

# Administration -- table of contents

## 6: Administer applications

### 6.1: Quick reference for administration

### 6.2: Preparing to host applications

#### 6.2.1: The default resources

### 6.3: Assembling applications and generating deployment code

#### 6.3.1: Assembling modules

#### 6.3.2: Setting properties for a module

#### 6.3.3: Generating deployment code for modules

#### 6.3.4: Verifying archives

### 6.4: Installing applications and setting classpaths

#### 6.4.1: Setting classpaths

### 6.5: Maintaining and updating applications

#### 6.5.1: Hot deployment and dynamic reloading

### 6.6: Tools and resources quick reference

#### 6.6.0: About user assistance

##### 6.6.0.3: Web administrative console overview

##### 6.6.0.3a: Starting, stopping, and logging into the Web administrative console

##### 6.6.0.3b: Working with server configuration files, using the Web administrative console

##### 6.6.0.3b01: Format of problems page for server configuration

##### 6.6.0.3d: Setting preferences in the Web administrative console

##### 6.6.0.3e: Using the Web administrative console

##### 6.6.0.4: Overview of editing property files by hand

##### 6.6.0.5: Using the Application Assembly Tool interface

##### 6.6.0.5.1: Modifying and adding modules to applications

##### 6.6.0.5.2: Generating deployment code for modules

##### 6.6.0.5.3: Verifying archives

##### 6.6.0.5.4: Viewing deployment descriptors

##### 6.6.0.5.5: Specifying JNDI bindings and resolving references

##### 6.6.0.5.6: Converting EJB JAR files from 1.0 to 1.1 format

##### 6.6.0.5a: Starting and stopping Application Assembly Tool

6.6.0.6: Installing applications with the application installer command line

6.6.0.7: Launching Java application clients in the J2EE application client container

6.6.0.8: Expanding .ear files

6.6.0.9: Application Client Resource Configuration Tool for configuring client resources

6.6.0.9.3: Removing objects from EAR files with the ACRCT

6.6.0.9a: Starting the ACRCT and opening an EAR file

6.6.0.10: SoapEarEnabler tool

6.6.0.13: Log Analyzer main window

6.6.0.13.1: Log Analyzer Find window

6.6.0.13.2: Log Analyzer Preferences notebook -- General

6.6.0.13.3: Log Analyzer Preferences notebook -- Appearance

6.6.0.13.4: Log Analyzer Preferences notebook -- Toolbars

6.6.0.13.5: Log Analyzer Preferences notebook -- Help

6.6.0.13.6: Log Analyzer Preferences notebook -- Logs

6.6.0.13.7: Log Analyzer Preferences notebook -- Severity

6.6.0.13.8: Log Analyzer Preferences notebook -- Record

6.6.0.13.9: Log Analyzer Preferences notebook -- Analyzer output

6.6.0.14: XML-SOAP Admin tool

6.6.0.15: Deployment Tool

6.6.0.15.1: Generating EJB deployment code from the command line

6.6.0.15.2: Meet-in-the-middle mapping support

6.6.0.15.3: Modifying the schema document

6.6.0.15.4: Modifying the map

6.6.0.15.5: Implementing custom finder helpers for CMP entity beans

6.6.0.16: Dynamic fragment cache configuration

6.6.0.16.1: Global administration

6.6.0.16.2: Policy configuration

6.6.0.16.3: Dynamic fragment cache XML examples

6.6.0.16.4: Dynamic fragment caching monitor

6.6.0.17: DrAdmin command reference

6.6.1: Administering applications (overview)

6.6.1.0: Enterprise application properties

6.6.1.0.1.a: Assembly properties for security roles (application)

6.6.1.0aa: Assembly properties for enterprise applications

6.6.1.3: Administering applications with the Web console

6.6.1.3.1: Installing applications with the Web console

- 6.6.1.3.1a: Mapping roles to users with the Web console
- 6.6.1.3.1b: Mapping EJB "Run As" roles to users with the Web console
- 6.6.1.3.1c: Mapping virtual hosts to Web modules with the Web console
- 6.6.1.3.1d: Modify the EJB to JNDI name mapping with the Web console
- 6.6.1.3.1e: Modify the EJB Reference to JNDI name mapping with the Web console
- 6.6.1.3.1f: Modify the Resource reference to JNDI name mapping with the Web console
- 6.6.1.3.1g: Specify CMP data source bindings with the Web console
- 6.6.1.3.2: Starting, stopping, and restarting applications and modules with the Web console
- 6.6.1.3.3: Uninstalling applications with the Web console
- 6.6.1.3.4: Updating applications with the Web console
- 6.6.1.3.5: Exporting application configurations with the Web console
- 6.6.1.3.6: Exporting DDL for EJB modules with the Web console
- 6.6.1.3.7: Viewing deployment descriptor information for enterprise applications (read-only)

#### 6.6.1.5: Administering applications with Application Assembly Tool

- 6.6.1.5.1: Creating an application

### 6.6.2: Administering nodes (overview)

#### 6.6.2.0: Node properties

#### 6.6.2.3: Administering nodes with the Web console

- 6.6.2.3.4: Updating nodes with the Web console

### 6.6.3: Administering application servers

#### 6.6.3.0: Application server properties

#### 6.6.3.3: Administering application servers with the Web console

- 6.6.3.3.1: Starting application servers with the Web console
- 6.6.3.3.2: Pinging application servers with the Web console
- 6.6.3.3.3: Stopping application servers with the Web console
- 6.6.3.3.4: Updating application server configurations with the Web console

### 6.6.4: Administering EJB containers (overview)

#### 6.6.4.0: EJB container properties

#### 6.6.4.3: Administering EJB containers with the Web console

- 6.6.4.3.4: Updating EJB container configurations with the Web console

### 6.6.5: Administering EJB modules (overview)

#### 6.6.5.0: EJB module properties

- 6.6.5.0.1: Assembly properties for entity beans
- 6.6.5.0.2: Assembly properties for CMP fields
- 6.6.5.0.3: Assembly properties for method extensions
- 6.6.5.0.4: Assembly properties for session beans
- 6.6.5.0.5: Assembly properties for security roles

- 6.6.5.0.6: Assembly properties for method permissions
- 6.6.5.0.7: Assembly properties for container transactions
- 6.6.5.0aa: Assembly properties for EJB modules
  
- 6.6.5.3: Administering EJB modules with the Web console
  - 6.6.5.3.2: Viewing deployment descriptor information for EJB modules (read-only)
  - 6.6.5.3.4: Updating EJB module configurations with the Web console
  
- 6.6.5.5: Administering EJB modules with Application Assembly Tool
  - 6.6.5.5.1: Creating an EJB module
  
- 6.6.7: Administering Web containers
  - 6.6.7.0: Web container properties
  
  - 6.6.7.3: Administering Web containers with the Web console
    - 6.6.7.3.4: Updating Web container configurations with the Web console
  
- 6.6.8: Administering Web modules (overview)
  - 6.6.8.0: Web module properties
    - 6.6.8.0.1: Assembly properties for Web components
    - 6.6.8.0.2: Assembly properties for initialization parameters
    - 6.6.8.0.3: Assembly properties for page lists
    - 6.6.8.0.4: Assembly properties for security constraints
    - 6.6.8.0.5: Assembly properties for Web resource collections
    - 6.6.8.0.8: Assembly properties for context parameters
    - 6.6.8.0.9: Assembly properties for error pages
    - 6.6.8.0.10: Assembly properties for MIME mapping
    - 6.6.8.0.11: Assembly properties for servlet mapping
    - 6.6.8.0.12: Assembly properties for tag libraries
    - 6.6.8.0.13: Assembly properties for welcome files
    - 6.6.8.0.14: Assembly properties for MIME filters
    - 6.6.8.0.15: Assembly properties for JSP attributes
    - 6.6.8.0.16: Assembly properties for file-serving attributes
    - 6.6.8.0.17: Assembly properties for invoker attributes
    - 6.6.8.0.18: Assembly properties for servlet caching configurations
  - 6.6.8.0aa: Assembly properties for Web modules
  
  - 6.6.8.3: Administering Web modules with the Web console
    - 6.6.8.3.1: Precompiling JSP files for Web modules of an application with the Web console
    - 6.6.8.3.2: Viewing deployment descriptor information for Web modules (read-only)
    - 6.6.8.3.4: Updating Web module configurations with the Web console
  
  - 6.6.8.5: Administering Web modules with Application Assembly Tool
    - 6.6.8.5.1: Creating a Web module
  
- 6.6.11: Administering HTTP session support (overview)

#### 6.6.11.0: Session Manager properties

#### 6.6.11.3: Administering session management with the Web console

##### 6.6.11.3.4: Updating session management settings with the Web console

#### 6.6.11.5: Procedure for configuring persistent session support

#### 6.6.12: Configuring user profile support

#### 6.6.13: Administering transports (overview)

#### 6.6.13.0: Properties of transports

#### 6.6.13.3: Administering transports with the Web console

##### 6.6.13.3.1: Configuring new HTTP transports with the Web console

##### 6.6.13.3.3: Removing HTTP transport configurations with the Web console

##### 6.6.13.3.4: Updating transport configurations with the Web console

#### 6.6.14: Administering database connections (overview)

#### 6.6.14.0: Properties of JDBC and data source providers

##### 6.6.14.0.1: Properties of data sources

#### 6.6.14.3: Administering database connections with the Web console

##### 6.6.14.3.1: Configuring new JDBC providers with the Web console

##### 6.6.14.3.2: Configuring new data source configurations with the Web console

##### 6.6.14.3.3: Updating JDBC provider configurations with the Web console

##### 6.6.14.3.4: Updating data source configurations with the Web console

##### 6.6.14.3.5: Removing JDBC provider configurations with the Web console

##### 6.6.14.3.6: Removing data source configurations with the Web console

#### 6.6.14.5: Additional administrative tasks for specific databases

#### 6.6.14.6: Notes about various databases

#### 6.6.14.8: Recreating database tables from the exported table DDL

#### 6.6.14.9: Administering data source providers and data sources with the ACRCT

##### 6.6.14.9.1: Configuring new data source providers with the ACRCT

##### Configuring new data sources with the ACRCT

##### 6.6.14.9.3: Removing data source providers (JDBC providers) and data sources with the ACRCT

##### 6.6.14.9.4: Updating data source and data source provider configurations with the ACRCT

#### 6.6.15: Administering custom services (overview)

#### 6.6.15.0: Properties of custom services

#### 6.6.15.3: Administering custom services with the Web console

##### 6.6.15.3.1: Configuring new custom services with the Web console

##### 6.6.15.3.3: Removing custom services with the Web console

##### 6.6.15.3.4: Updating custom service configurations with the Web console

## 6.6.16: Administering virtual hosts (overview)

### 6.6.16.0: Properties of virtual hosts

### 6.6.16.3: Administering virtual hosts with the Web console

#### 6.6.16.3.1: Configuring new virtual hosts with the Web console

#### 6.6.16.3.3: Removing virtual hosts with the Web console

#### 6.6.16.3.4: Updating virtual host configurations with the Web console

## 6.6.18: Securing applications

### 6.6.18.0: General security properties

#### 6.6.18.0.1: Properties for configuring Secure Socket Layer (SSL) support

#### 6.6.18.0.2: Properties for configuring security using local operating system

#### 6.6.18.1a07: Configuring SSL in WebSphere Application Server

### 6.6.18.3: Administering security with the Web console

#### 6.6.18.3.1: Enabling global security with the Web console

#### 6.6.18.3.3: Removing global security with the Web console

#### 6.6.18.3.6: Specifying user IDs for the server and administrator with the Web console

### 6.6.18.6: Avoiding known security risks in the runtime environment

### 6.6.18.7: Protecting individual application components and methods

### 6.6.18.9: Specifying authentication options in sas.client.props

### 6.6.18.10: The demo keyring

### 6.6.18.12: Cryptographic token support

## 6.6.19: Administering the product messages, logs, and traces (overview)

### 6.6.19.0: Properties for tracing, logging, and messages

#### 6.6.19.0.3: Server trace properties

### 6.6.19.3: Administering server traces with the Web console

#### 6.6.19.3.1: Enabling and specifying trace strings with the Web console

#### 6.6.19.3.2: Starting (sending) server traces with the Web console

#### 6.6.19.3.3: Disabling server traces with the Web console

#### 6.6.19.3.4: Retrieving trace strings from the server with the Web console

#### 6.6.19.3.5: Specifying server trace logs with the Web console

#### 6.6.19.3.6: Viewing trace logs with the Web console

#### 6.6.19.3.7: Specifying the trace host name and port with the Web console

## 6.6.20: Administering transactions (overview)

### 6.6.20.0: Transaction properties

### 6.6.20.3: Administering the transaction service with the Web console

#### 6.6.20.3.1: Enabling the transaction service with the Web console

#### 6.6.20.3.3: Disabling the transaction service with the Web console

## 6.6.24: Administering application client modules (overview)

### 6.6.24.0: Application client module properties

#### 6.6.24.0aa: Assembly properties for application client modules

### 6.6.24.5: Administering application clients with Application Assembly Tool

#### 6.6.24.5.1: Creating an application client

## 6.6.25: Administering resource providers (overview)

### 6.6.25.0: J2EE resource and resource provider properties

### 6.6.25.3: Administering resource providers with the Web console

## 6.6.26: Administering application server process definitions (overview)

### 6.6.26.0: Process definition properties

### 6.6.26.3: Administering application server process definitions with the Web console

#### 6.6.26.3.4: Updating process definition configurations with the Web console

## 6.6.27: Administering path maps (overview)

### 6.6.27.0: Path map properties

### 6.6.27.3: Administering path maps with the Web console

#### 6.6.27.3.1: Configuring new path maps with the Web console

#### 6.6.27.3.3: Removing path maps with the Web console

#### 6.6.27.3.4: Updating path maps with the Web console

## 6.6.28: Administering Object Level Trace and Distributed Debugger support (overview)

### 6.6.28.0: Object Level Trace and Distributed Debugger properties

### 6.6.28.3: Administering OLT and Debugger with the Web console

#### 6.6.28.3.4: Updating IBM Object Level Trace and Distributed Debugger configurations with the Web console

## 6.6.29: Administering Location Service Daemons (overview)

### 6.6.29.0: Location Service Daemon properties

### 6.6.29.3: Administering the Location Service Daemon with the Web console

#### 6.6.29.3.4: Updating Location Server Daemon configurations with the Web console

## 6.6.30: Administering Object Request Brokers (ORBs)

### 6.6.30.0s: Object Request Broker and ORB thread pool properties for Advanced Single Server Edition

### 6.6.30.3: Administering Object Request Brokers (ORBs) with the Web console

- 6.6.30.5: Setting the ORB timeout value
- 6.6.32: Administering name service support (overview)
- 6.6.34: Administering environment entries
  - 6.6.34.0: Properties of environment entries
    - 6.6.34.0aa: Assembly properties for environment entries
- 6.6.35: Administering generation of deployment code
  - 6.6.35.0: Assembly properties for generating deployment code
- 6.6.36: Administering Java Virtual Machine settings (overview)
  - 6.6.36.0: JVM properties
    - 6.6.36.0.1: Java command line arguments reference
  - 6.6.36.3: Administering JVM settings with the Web console
- 6.6.37: Administering mail providers and mail sessions
  - 6.6.37.0: Properties of JavaMail providers
    - 6.6.37.0.1: Properties of JavaMail sessions
  - 6.6.37.3: Administering mail providers and mail sessions with the Web console
    - 6.6.37.3.1: Configuring new mail providers with the Web console
    - 6.6.37.3.3: Updating mail provider configurations with the Web console
    - 6.6.37.3.4: Updating mail session configurations with the Web console
    - 6.6.37.3.5: Removing mail provider configurations with the Web console
    - 6.6.37.3.6: Removing mail session configurations with the Web console
  - 6.6.37.9: Administering JavaMail providers and sessions with the ACRCT
    - 6.6.37.9.1: Configuring new JavaMail sessions with the ACRCT
    - 6.6.37.9.3: Removing JavaMail sessions with the ACRCT
    - 6.6.37.9.4: Updating JavaMail session configurations with the ACRCT
- 6.6.38: Administering URL providers and URLs (overview)
  - 6.6.38.0: Properties of URL providers
    - 6.6.38.0.1: Properties of URLs
  - 6.6.38.3: Administering URL providers with the Web console
    - 6.6.38.3.1: Configuring new URL providers with the Web console
    - 6.6.38.3.3: Updating URL provider configurations with the Web console
    - 6.6.38.3.4: Updating URL configurations with the Web console
    - 6.6.38.3.5: Removing URL provider configurations with the Web console
    - 6.6.38.3.6: Removing URL configurations with the Web console
  - 6.6.38.9: Administering URL providers and URLs with the ACRCT



- 6.6.38.9.1: Configuring new URL providers and URLs with the ACRCT
  - Configuring new URLs with the ACRCT
- 6.6.38.9.3: Removing URL providers and URLs with the ACRCT
- 6.6.38.9.4: Updating URL and URL provider configurations with the ACRCT

## 6.6.39: Administering messaging and JMS providers (overview)

### 6.6.39.0: Properties of JMS providers

- 6.6.39.0.1: Properties of JMS connection factories
- 6.6.39.0.2: Properties of JMS destinations

### 6.6.39.3: Administering JMS providers with the Web console

- 6.6.39.3.1: Configuring new JMS providers with the Web console
- 6.6.39.3.2: Configuring new JMS connection factories with the Web console
- 6.6.39.3.3: Configuring new JMS destinations with the Web console
- 6.6.39.3.4: Updating JMS provider configurations with the Web console
- 6.6.39.3.5: Updating JMS connection factory configurations with the Web console
- 6.6.39.3.6: Updating JMS destination configurations with the Web console
- 6.6.39.3.7: Removing JMS provider configurations with the Web console
- 6.6.39.3.8: Removing JMS connection factory configurations with the Web console
- 6.6.39.3.10: Removing JMS destination configurations with the Web console

### 6.6.39.9: Administering JMS providers, connection factories, and destinations with the ACRCT

- 6.6.39.9.1: Configuring new JMS providers with the ACRCT
  - Configuring new JMS connection factories with the ACRCT
  - Configuring new JMS destinations with the ACRCT
- 6.6.39.9.3: Removing JMS providers, connection factories, and destinations with the ACRCT
- 6.6.39.9.4: Updating JMS provider, connection factory, and destination configurations with the

ACRCT

## 6.6.41: Administering WebSphere administrative domains (overview)

### 6.6.41.0: Administrative domain properties

### 6.6.41.3: Administering administrative domains with the Web console

- 6.6.41.3.4: Updating administrative domain configurations with the Web console

### 6.6.41.5: Establishing multiple administrative domains on a machine

## 6.6.43: Administering references

### 6.6.43.0: Properties of references

- 6.6.43.0.1: Assembly properties for EJB references
- 6.6.43.0.2: Assembly properties for resource references
- 6.6.43.0.3: Assembly properties for security role references

## 6.6.45: Administering WebSphere plug-ins for Web servers

### 6.6.45.0: Properties of WebSphere plug-ins for Web servers

#### 6.6.45.0.1: Modifications to Web server configuration files during product installation

6.6.45.5: Controlling where the WebSphere plug-ins for Web servers are installed

6.6.45.6: Regenerating the Web server plug-in configuration

6.6.45.7: What to do after changing Web server ports

6.6.45.8: Checking your IBM HTTP Server version

6.6.45.9: Manually updating the Domino Web server configuration file

6.6.48: Administering ports

6.6.49: Administering National Language Support

6.6.51: Administering network configurations

6.6a: Starting and stopping servers

6.6a01: Running the product servers and consoles as non-root

6.7: Tutorials

6.7.1: Application assembly tutorial

6.7.2: Application deployment tutorial

6.7.3: Application testing tutorial

6.7.4: Application security tutorial

6.7.5: Advanced application assembly and deployment tutorial -- CMP bean application

6.7.6: Application cleanup and removal tutorial

6.7.soap: Deploying a Java class as a Web service, using SOAP

6.7.hc: HitCount tutorial for using Debugger and OLT

6.7.sq: StockQuote tutorial for using Debugger and OLT

6.10: Backing up and restoring administrative configurations

# 6: Administer applications

Typically, one or more application developers with different areas of expertise (such as architects, Java programmers, legacy programmers, and Web programmers) [design and create a new application](#) or [migrate an existing application to support new Java specifications](#) based on input from business users.

After the files comprising an application have been developed, then the application can be added to the application server for access by users. This section of documentation helps you learn the flow of administrative activities, whether you are a code developer introducing an application into a test environment, or an administrator responsible for the production environment.

## A feasible end-to-end administrative procedure

The following procedure provides links to more information about each step. To practice the procedure using an application provided with WebSphere Application Server, perform the application configuration and deployment tutorials.

1. **Install the product.** Plan and install a topology comprised of one or more product installations and the necessary prerequisites.
  - Who: Lead architect, network administrator, systems administrator
  - How: See [components](#) section, [installation](#) section .
2. **Assemble the application.** Assemble the application modules from the application files. Set values for the deployment descriptor. For applications containing EJB modules, generate code for deployment.
  - Who: Web designer, Java programmer
  - How: [Using the Application Assembly Tool](#), and possibly the [Deployment Tool for Enterprise JavaBeans](#)

For example, for an EJB module, the enterprise bean developer writes and compiles the enterprise bean components. The developer assembles the components and a deployment descriptor into an EJB JAR file and then generates code for deployment.

For entity beans (BMP or CMP), the developer also generates the database tables the beans will use to store their data.

The developer transfers the JAR file to the WebSphere administrator, or informs the administrator of its location on a machine in the WebSphere administrative domain.

3. **Secure the application.** Configure security in the application.
  - Who: Systems administrator
  - How: [Using the Application Assembly Tool](#)
4. **Configure the runtime and resources.** Create application servers and other resources that will support applications.
  - Who: Systems administrator
  - How: [Using the administrative console](#)
5. **Install the application.** Install the application into the test environment.
  - Who: Java programmer, systems administrator
  - How: [Using the administrative console](#)

Installing an application refers to the process of placing the Enterprise Archive (EAR) file in a runtime

environment comprised of an application server.

During this step, the administrator can optionally edit the application's deployment descriptor.

6. **Configure resources for application clients (if applicable).** Create resources for the Java clients that will access your application.
  - Who: Systems administrator
  - How: Using the Application Client Resource Configuration Tool
7. **Test the application.** Test the application prior to enabling runtime security so that you can tell the difference between security-related problems and other problems.
  - Who: Systems administrator
  - How: Using a Web browser or other application client
8. **Configure runtime security.** Configure and enable security in the application server runtime.
  - Who: Systems administrator
  - How: [Using the administrative console](#)

Note that because security is presented as a scenario, the instructions (such as suggested field values) refer to a particular sample application. Substitute the names of your own application components.
9. **Test application security.** Test the application again, this time with security enabled.
  - Who: Tester, systems administrator
  - How: [Using the administrative console](#)
10. **Perfect the test environment.** Debug and verify the application in the test environment. Perform preliminary tuning.
  - Who: Tester, systems administrator
  - How: Using the administrative console
11. **Port the application to production environment.** Configure the production environment and move the application there.
  - Who: Network administrator, systems administrator
  - How: Using the administrative console
12. **Manage the production environment.** Manage the production application, tuning the application files and configuration as needed. Update the production application as needed.
  - Who: Maintenance -- Network administrator, tester, Updates -- Java programmer, systems administrator
  - How: [Using the administrative console](#)
13. **Uninstall applications as needed.**
  - Who: Network administrator, systems administrator
  - How: [Using the administrative console](#)
14. **Remove security, as needed.**
  - Who: Network administrator, systems administrator
  - How: [Using the administrative console](#)

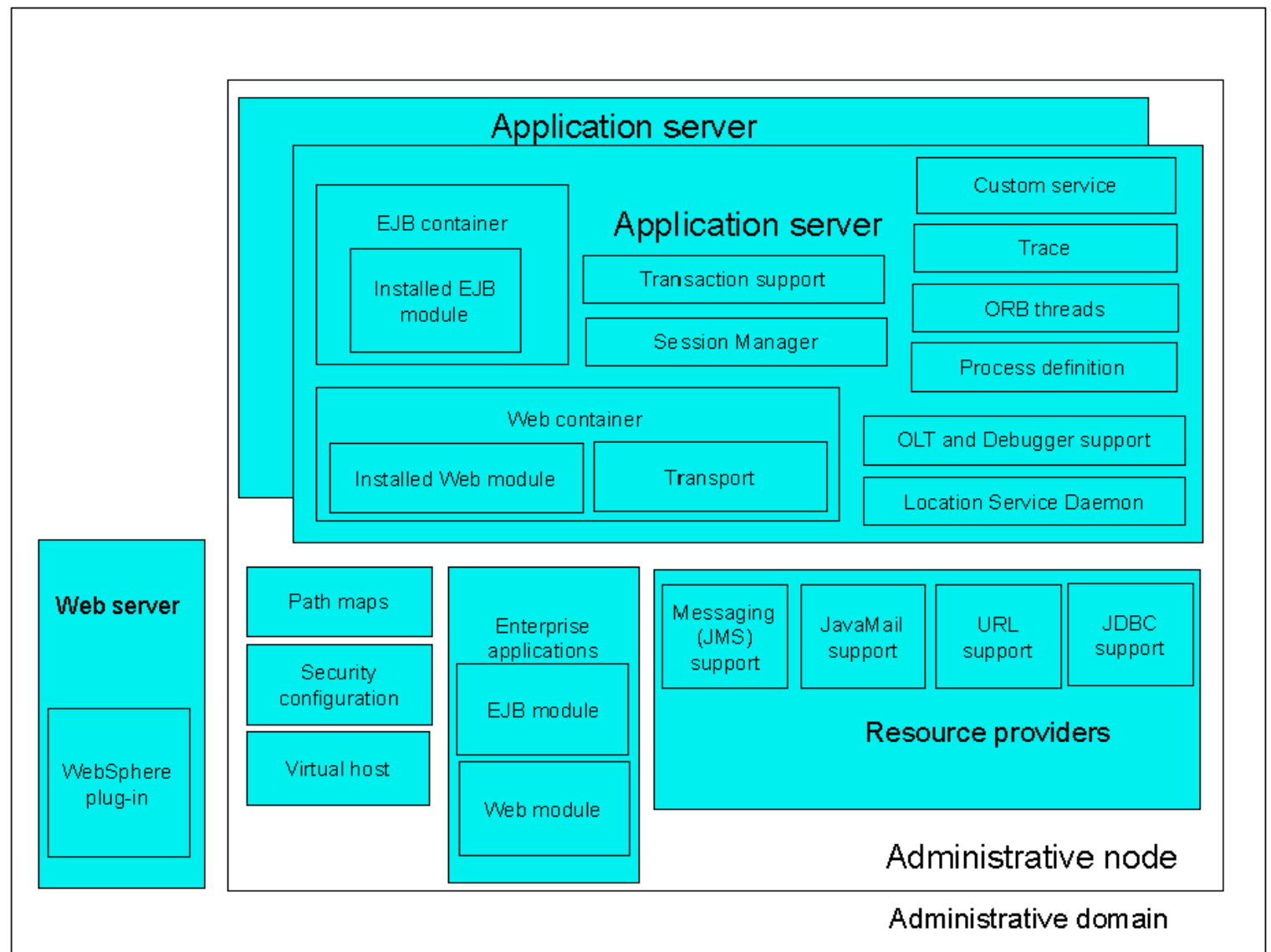
## More information

To obtain more information about a step in the above procedure, see the links to sub-topics.

## 6.1: Quick reference for administration

The administrative model for Advanced Single Server Edition is summarized graphically below. Click the name of a resource to view the entry point to the documentation for the administering the resource.

See below for further discussion of the administrative model depicted here.



As shown above, and in the tree views of the administrative console and other graphical administrative tools, the WebSphere administrative domain is comprised of several resource types, arranged in a hierarchy of containment relationships.

The administrative node contains an application server, plus several resources that transcend application servers (such as those related to JDBC support). The WebSphere plug-ins for the supported Web servers also require administration.

In this documentation, the term *resource* is used loosely to describe a logical set of properties that can be administered, such as settings for session support using the Session Manager. Some resources, such as application servers, are "live objects," meaning they can be started and stopped. Other resources are simply groups of related settings, such as the properties for configuring transaction support.

The set of resources outlined in the documentation varies somewhat in granularity from the administrative model portrayed in the administrative tools, geared towards helping you map activities (to administer transaction support, to administer HTTP session support, to administer Java messaging support, and so on) onto the specific resources and hierarchy of the WebSphere administrative model.

The documentation entry point for each resource type includes:

- An overview of administering the resource type
- Links to instructions for using the relevant administrative tools to administer the resource
- Links to field descriptions for configuring the properties of the resource
- Links to conceptual information answering "What is *this resource type*?"

## 6.2: Preparing to host applications

If you are using the default [application server XML configuration file](#), then you already have a default application server and several other default resources. You are ready to add your first application to the runtime environment. You can stop reading at this point!

Also, for your convenience, in addition to the default file, another `template-server-cfg.xml` file is shipped for configuring a server from a basic set of definitions.

The default application server and a set of default resources are available to help you begin quickly (see the [related topics links](#)). Suppose you choose instead to configure a new server and set of resources. Here is what you need to do in order to set up a runtime environment to support applications.

1. [Create an application server](#).
2. Create other supporting resources.
  - [Create a virtual host](#).
  - [Configure a Web container](#).
  - [Configure an EJB container](#).
3. Create resources for data access.
  - [Create a JDBC provider and data source](#).
4. Create additional resources for J2EE services.
  - [Create a URL and URL provider](#).
  - [Create a JMS destination, connection, and provider](#).
  - [Create a JavaMail session](#).
5. Create resources for personalization support.
  - [Configure a Session Manager](#).

Configuring the application server runtime and resources comprises one step of [a recommended end-to-end administrative procedure](#).

### Configuring application client resources

Now is also a good time to prepare the resources needed by any Java clients that will use your application. See the [Related information](#), or return to the main administrative procedure, which contains a step reminding you to perform this task (if it applies).

## 6.2.1: The default resources and configurations

As described in the article about [preparing to host applications](#), the default [application server XML configuration file](#) includes a default application server and several other default resources for your environment.

## 6.3: Assembling applications and generating deployment code

The Application Assembly Tool is a graphical user interface for assembling enterprise (J2EE) applications. Use this set of articles to become familiar with the steps needed to assemble an application. Then use the related tasks to create the application's modules, generate code for deployment, and verify the application's archive files.

Application assembly is the process of creating an archive file that groups all files related to an application (the archive file can include other archive files as well as individual class files, HTML files, GIF files, and soon). A single archive file can represent an application. Multiple archive files can be further bundled into a higher-level archive if the application has many components. Each bundled entity is called a *module*. The assembly process consists of selecting all of the files to be included in the module, creating a deployment descriptor (XML file) for the module, and then packaging the files into a single archive file. The archive file lists all files in the module. The deployment descriptor lists the characteristics of the module and contains instructions for how the module is to be deployed.

Assembling an application consists of the following steps:

1. Launch the Application Assembly Tool.
2. Create modules.
  - Specify the location of files that make up the application.
  - Enter or edit properties used to generate the deployment descriptor file.
  - Optionally, add JNDI binding information for and resolve references to the application's external resources.
3. Save the archive file. This automatically generates the XML deployment descriptor files for the application.
4. Verify the archive file and, optionally, view the content of the XML files.
5. If the application contains EJB modules, or is a stand-alone EJB module, generate deployment code for the application.

The Application Assembly Tool invokes the Deployment Tool for Enterprise JavaBeans (ejbdeploy) for generating code and validating archives. For CMP entity beans, the tool can also generate a map and schema document, and DDL file containing SQL code for creating a database table.

After an application is assembled, it can be installed in an application server by using the administrative console. At installation time, the console is used to carry out the security instructions defined in the deployment descriptor and to locate the required external resources that it will use, such as enterprise beans and databases. In the console, additional configuration properties can be added and binding properties defined in the Application Assembly Tool can be redefined.

After the application is deployed, you can use the Application Assembly Tool to modify the application by adding or removing modules, editing deployment descriptor properties, and regenerating code for deployment.

**Note:** If you are using the Application Assembly Tool to create application client modules, you must also use the Application Client Resource Configuration Tool. This tool allows you to define references to resources (other than enterprise beans) on the machine where the application client resides.



## 6.3.1: Assembling modules

A module is a group of related files that represent an application. The files can include class files, HTML files, and GIF files. Assembly is the process of specifying which files make up the module, creating deployment descriptor files for the module, and packaging these files in an archive. Modules are used to represent any of the following:

- One or more Web components. Web components are servlets or JavaServer Pages (JSP) files.
- One or more enterprise beans.
- Application clients.

A J2EE application consists of any combination of the above.

The standard JAR file format is used to package modules, regardless of the type of module. However, the archive file is referred to according to its content:

- EJB modules--packaged in JAR files
- Web modules--packaged in Web archive (WAR) files
- Application-client modules--packaged in JAR files
- Enterprise applications--packaged in Enterprise Archive (EAR) files

An EJB module, Web module, or application-client module can be installed as a stand-alone application or can be combined with other modules into an enterprise application. When a stand-alone application is installed in WebSphere Application Server, it is automatically packaged in an EAR file.

## 6.3.2: Setting properties for a module

A module's properties are used to create a deployment descriptor for the module. The properties represent the deployment descriptor elements as defined in the J2EE specifications. The Application Assembly tool automatically creates deployment descriptor files based on values entered in the property dialog boxes and wizards. The deployment descriptor is an XML document that contains application configuration data that the run-time uses. In general, a deployment descriptor contains the following information:

- Information about the content of the module being assembled. For example, for an EJB module, the deployment descriptor lists each enterprise bean's class, home interface class, remote interface class, whether the bean is an entity or session bean, and the bean's attributes (such as persistence management type and primary key class for entity beans).
- References to a module's internal and external dependencies (such as enterprise beans, databases, and resource connection factories needed by the module). Internal dependencies are dependencies on other components within the same module. External dependencies are dependencies on components residing outside of the module. For example, an enterprise bean in an EJB module can require another enterprise bean that is not packaged in the same module. The deployment descriptor can contain references to the remote bean's home interface. The deployment descriptor can also include JNDI binding information that the container uses to locate the remote bean at install time.
- Run-time-specific information needed by the application. For example, the servlet mappings needed for a Web application or the persistence management (BMP or CMP) to be used by an entity bean.
- References to security roles. Security information is used when the module is deployed.

In addition to XML files, the Application Assembly Tool automatically generates XMI files to store binding and IBM extension information. Binding information maps a resource (enterprise bean or resource connection factory object) from the logical name specified in the deployment descriptor to its actual name in the global JNDI namespace. IBM deployment descriptor extensions are additions to the standard descriptors for J2EE applications, Web applications, and enterprise beans. The extensions allow you to specify properties that enable Enterprise Edition or legacy (older) systems to work in the WebSphere Application Server environment. For example, an extension property is used to define how to manage transaction scoping for an enterprise bean's methods.

A J2EE application contains one application-level deployment descriptor file, governing the application as a whole. It also contains a component-level deployment descriptor file, one governing each module in the application. For example, the application-level deployment descriptor specifies security information for the application and defines how often the application is to be reloaded. The component-level deployment descriptor (for example, for an EJB module) specifies the following:

- General properties of the EJB module, such as the location of class files needed for a client program to access the enterprise beans in the module and the icons to be associated with the module.
- The deployable enterprise beans that the module will contain.
- Security roles used to access resources in the module.
- Transaction attributes for the enterprise bean methods.
- A default datasource to be used by entity beans in the module, if one is not specified.

## 6.3.3: Generating deployment code for modules

Deployment code for an application can be generated when the application is installed in WebSphere Application Server. Alternatively, it can be manually generated before installation. During code generation, the Application Assembly Tool invokes the Deployment Tool for Enterprise JavaBeans (ejbdeploy) to do the following:

1. Generate and compile container implementation classes.
2. Process the implementation classes by using the RMIC compiler, which generates the client-side stub and server-side skeletons needed for remote method invocation (RMI).
3. Validate the JAR file and display error messages if there are any J2EE specification violations within the JAR file. Verifying archive files is important for successful generation of deployment code for the application.
4. For CMP entity beans, generate persistence code.
5. Insert the files back into the JAR file.

For CMP entity beans, if the JAR file contains a map and schema document, that schema is used. If the JAR file does not contain a map and schema document, the Application Assembly Tool uses a top-down mapping to generate the following files:

- META-INF\Table.ddl
- META-INF\map.mapxmi
- META-INF\Schema\schema.rdbxmi

**Note:**

For Advanced Single Server Edition of this product, only the Table.ddl file is generated.

The user must create the database table manually. The generated Table.ddl file can be used as a guide, or the user can create the table by some other means.

If you do not wish to use the default top-down mapping, the recommended tool is VisualAge for Java. For information on default mappings, see the documentation for Deployment Tool for Enterprise JavaBeans.

To migrate CMP entity beans from WebSphere Application Server 3.5 for use in version 4.0, use the -35 option of the Deployment Tool for Enterprise JavaBeans. This option is recommended only if you need to preserve use of the mapping rules that were used in the 3.5 version of the Deployment Tool.

## 6.3.4: Verifying archives

Verifying archive files is important for successful generation of deployment code for the application. During verification, the Application Assembly Tool checks that an archive is complete, and that deployment descriptor properties and references contain appropriate values. Specifically, the verification process checks the following:

- Required deployment properties must contain values.
- Values specified for environment entries must match their associated Java types.
- In both EAR and WAR files:
  - For EJB references, the target enterprise bean of the link must exist.
  - For security role references, the target role must exist.
  - Security roles must be unique.
- For EAR files, each module listed in the deployment descriptor must exist in the archive.
- For WAR files, the files for icons, servlets, error, and welcome pages listed in the deployment descriptor must have corresponding files in the archive.
- For EJB modules:
  - All class files referenced in the deployment descriptor must exist in the JAR file.
  - Method signatures for enterprise bean home, remote, and implementation classes must be compliant with the EJB 1.1 specification.


If errors are found, a message is displayed in a scrolling window.

## 6.4: Installing applications and setting classpaths

This article describes how to install an application into the WebSphere Application Server environment. This includes placing the files comprising an application onto the physical system containing the WebSphere Application Server product and updating the application server configuration with information about the application.

Installing applications (or standalone modules) into the application server runtime involves the following tasks:

- Configure the application settings in the administrative console, by performing an application installation task (requires the server to be running)
  - [6.6.1.3.1: Installing applications with the Web console](#)
- Placing the application files in a directory location, usually relative to the IBM WebSphere Application Server *product\_installation\_root*
  - Place the files in the location of your choice. Specify the location when you are performing the application installation task.
- Setting classpaths as needed, in addition to those set during application assembly
  - [Setting classpaths](#)

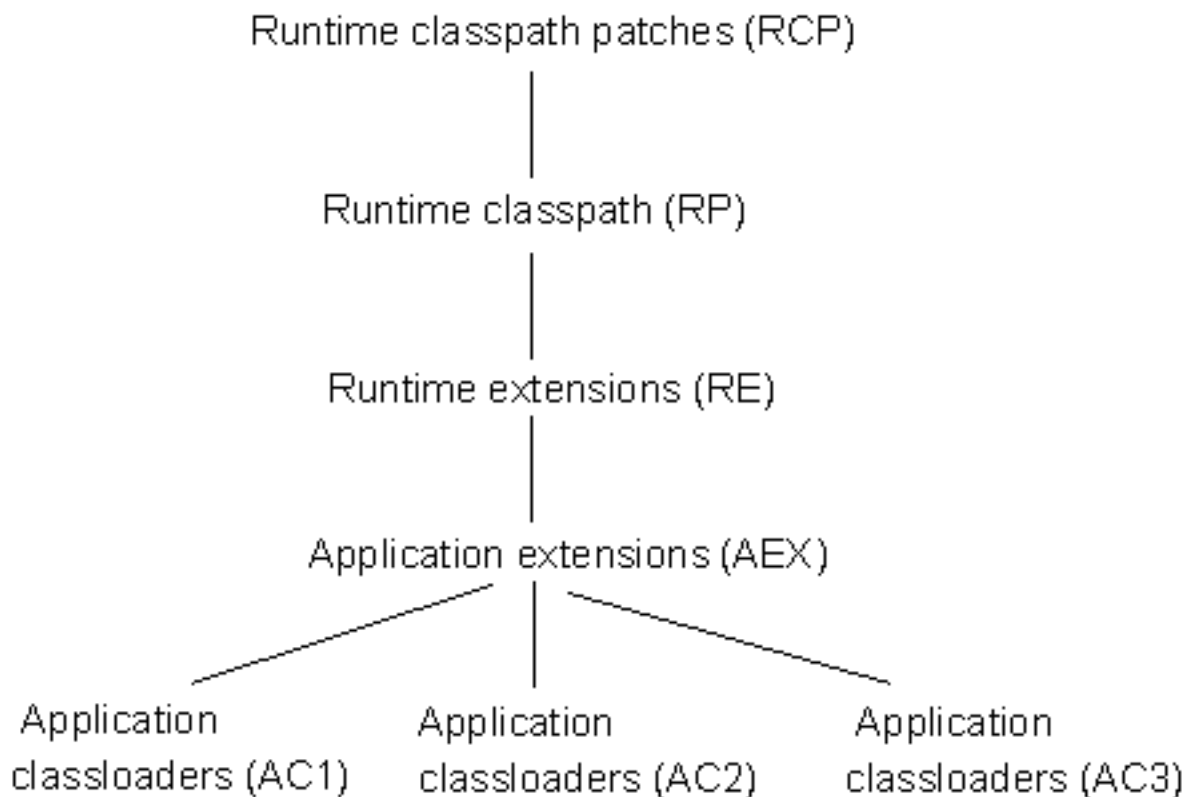
 The [application installer tool](#) is also available for installing application files into the directory structure. The command line-based application installer tool accomplishes the same thing as the application installation task in the administrative console, but does not require the application server and administrative console to be running at the time of application installation.

## 6.4.1: Setting classpaths

- [The classpaths and their search orders](#)
- [The system classpath](#)
- [Module visibility for setting isolation levels](#)
- [Guidelines for reloading classes](#)
- [Guidelines for packaging classes](#)

### The classpaths and their search orders

The relationship among the classloaders represents the fact that classes loaded by a particular classloader can reference other classes as long as these other classes can be loaded by the same classloader or any of its ancestors (but not its children).



#### Runtime classpath patches (RCP)

- **Description:** Classes and JAR files in the [product\\_installation\\_root](#)/classes directory
- **How to set and view contents:** The location is fixed by the System property `ws.ext.dirs`. You can view the directory contents to see the contents of this classpath.
- **Search order:** RCP -> RP -> RE. The runtime classpath patches override any copies of the same files lower in the classpath tree
- **Typical contents:** e-Fixes and other APARs to be applied to the application server runtime

#### Runtime classpath (RP)

- **Description:** Classes and JAR files in the [product\\_installation\\_root/lib](#) directory
- **How to set and view contents:** The location is fixed by the System property `ws.ext.dirs`. You can view the directory contents to see the contents of this classpath
- **Search order:** RCP -> RP -> RE. The runtime classloader checks the RCP, and then the RP
- **Typical contents:** The core WebSphere Application Server runtime

## Runtime extensions (RE)

- **Description:** Classes and JAR files in the [product\\_installation\\_root/lib/ext](#) directory
- **How to set and view contents:** The location is fixed by the System property `ws.ext.dirs`. You can view the directory contents to see the contents of this classpath
- **Search order:** RCP -> RP -> RE. The runtime classloader checks the RCP, then the RP, then the RE
- **Typical contents:** Extensions to the core WebSphere Application Server runtime

## Application extensions (AEX)

- **Description:** Classes and JAR files in the [product\\_installation\\_root/lib/app](#)
- **How to set and view contents:** The location is fixed by the WebSphere Application Server runtime and cannot be configured. You can view the directory contents to see the contents of this classpath
- **Search order:** The runtime classloader checks the AEX, then RCP -> RP -> RE.
- **Typical contents:** Class libraries that need to be shared among all J2EE applications installed on the server. Because these classes are not visible to the WebSphere Application Server runtime classloaders (RCP, RP, and RE), this classpath can contain updated versions of common libraries (such as JAXP) that are present in the runtime.

Place dependent JAR files in this directory, too.

## Application classloaders (AC)

- **Description:** Classpaths as specified for WAR and JAR modules
- **How to set and view contents:** The classpath for WAR and EJB JAR files are set using MANIFEST Class-Path entries. You can modify these settings using the [Application Assembly Tool](#).
- **Search order:** Depends on the module type.
  - **WAR modules** follow the search order: module classloader -> AEX -> RCP -> RP -> RE

Within the module classloader for the WAR file, the search order is:

1. Root of the WAR file
2. WEB-INF/classes directory contents
3. JAR files in the WEB-INF/lib directory

The module classloader includes the JAR or WAR and its pieces, WEB-INF/classes and WEB-INF/lib, and all of the entries specified by the Modules' class-path entry.

- **EJB JAR modules** follow the search order: AE -> RCP -> RP -> RE -> module classloader

The **delegation mode** of WAR and EJB JAR files can be changed by defining the following system properties:

- `com.ibm.ws.classloader.warDelegationMode` (false by default)
- `com.ibm.ws.classloader.ejbDelegationMode` (true by default)

Set the value of each property to true or false:

- **true:** AE -> RCP -> RP -> RE -> module classloader
- **false:** module classloader -> AE -> RCP -> RP -> RE
- **Typical contents:** Application code, including servlets, enterprise beans, JavaServer Pages (JSP) files, JavaBeans, and supporting class libraries

## The system classpath


If you use the [classpath setting in the administrative console](#) to specify additional entries to be added to the classpath, these entries will be appended to the classpath. If you specify a System property, `ws.ext.dirs`, these entries will be appended to the `ws.ext.dirs`.

The WebSphere Application Server runtime is loaded by a bootstrap classloader that loads classes from the directories, JAR files in specified directories, and JAR files specified by the system property `ws.ext.dirs`. Because the system classloader is the parent of the WebSphere bootstrap classloader, it is important to note that the classes put in the system classpath will not be able to load classes in the WebSphere runtime (including J2EE APIs).

## Module visibility for setting isolation levels

The relationship among the classloaders implies that it is not usually possible to access the classes of one Web application from another Web application.

The [application server settings](#) include a property for specifying the module visibility, which determines the level of isolation among the classpaths for various application components.

 Note that the default value of the module visibility setting impacts whether you can port applications from previous WebSphere Application Server versions or other editions (from *Advanced Single Server Edition* Version 4.0 to *Advanced Edition* Version 4.0, for example) without reassembling the applications or resetting the field value.

See the [module visibility field description in the application server property reference](#) for more information.

## Guidelines for reloading classes

To update (reload) the classes being used by an application, it is best to stop that J2EE application and start it again. This ensures that dependent classes between cooperating modules will be reloaded in unison, eliminating potential `ClassCastException`s that occur when mismatched versions of classes are being used.

However:

- EJB JAR files can be reloaded by stopping just the EJB module and starting it again. Any dependent EJB or WAR modules will need to be restarted as well.
- You can also reload EJB modules by specifying a reload interval in the EAR extension of the EAR file. If the EJB classes change, then the entire J2EE application (EAR) is restarted, not just the EJB module.
- WAR modules can be configured to reload automatically when their contents change. This is accomplished by setting an [Web module assembly property](#) using the [Application Assembly Tool](#).



## Guidelines for packaging classes

- The EJB JAR modules and WAR modules comprising an application should be packaged together in the same EAR module
- When a Web module (WAR) accesses an EJB module, it should not package the EJB interfaces and stubs in the WAR. Rather, it should express the dependency on the EJB JAR using a MANIFEST Class-Path entry. When using the [Application Assembly Tool](#), set the [classpath property of the Web module](#).
- Common classes that need to be shared among Web modules and EJB JAR modules should be packaged into a separate JAR file. Add the JAR file to the EAR file. Reference the JAR file in the MANIFEST Class-Paths of both the EJB JAR and Web module (WAR) files.
- Class library JAR files to be used only by WAR modules should be added to the WEB-INF/lib directory of the Web module

## 6.5: Maintaining and updating applications

This section includes information about monitoring daily operations and the health of applications, updating applications, and performing necessary tasks after updating an application (such as stopping a process and starting it again).

To update the contents of an application, use the Application Assembly Tool. Typical maintenance tasks include adding or editing assembly properties, adding or importing modules into an application, and adding enterprise beans, Web components, and files.

### What should you do if an application changes?

Depending on what you have changed, you can sometimes replace an application with a newer version, or modify configuration settings or application bindings, without having to disrupt the application server or the installed application.

Article 6.5.1 discusses various changes and the minimum actions you must take to update the application in your server runtime.

In the very worst case, you will need to:

1. [Reassemble the application, using the Application Assembly Tool.](#)
2. [Uninstall the current version of the application.](#)
3. [Install the new version of the application.](#)

## 6.5.1: Hot deployment and dynamic reloading

This article outlines the types of changes that can be made to application servers and their contents with respect to hot deployment and dynamic reloading. In other words, it discusses what you can change, and how, without having to stop the server and start it again.

Hot deployment is the process of adding new components (such as enterprise beans, servlets, and JSP files) to a running server without having to stop the application server process and start it again.

Dynamic reloading is the ability to change an existing component without needing to restart the server in order for the change to take effect. Dynamic reloading involves:

- Changes to the implementation of a component of an application, such as changing the implementation of a servlet
- Changes to the settings of the application, such as changing the deployment descriptor for a Web module

View information about updating and redeploying:

- [WAR Files](#)
- [EJB Jar Files](#)
- [Application](#)
- [Server Configuration](#)
- [HTTP Plugin Configuration Changes](#)

### WAR Files

#### Change an existing JSP File

- User Action: None
- Comments: The changed JSP can be placed directly in the [product\\_installation\\_root](#)/installedApps/<app name>/<module name> directory, or the appropriate subdirectory. The change will be automatically detected and the JSP will be recompiled and reloaded.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

#### Add a new JSP File to an existing application

- User Action: None
- Comments: The new JSP can be placed directly in the [product\\_installation\\_root](#)/installedApps/<app name>/<module name> directory, or the appropriate subdirectory. The new file will be automatically detected and compiled on the first request to the page.
- Hot deployment: Yes
- Dynamic reloading: Yes

#### Change an existing Servlet class (edit and recompile)

- User Action: None (if reloading is enabled). If reloading is disabled, the server will have to be restarted to detect the change.

- Comments: The new version of the servlet .class file can be placed directly in the [product\\_installation\\_root](#)/installedApps/ <app name>/<module name>/WEB-INF/classes directory. If the .class file is part of a jar file, the new version of the jar file can be placed directly in [product\\_installation\\_root](#)/installedApps/<app name>/<module name>/WEB-INF/lib. In either case, the change will be detected, the web application will be shutdown and reinitialized, picking up the new class.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## Change a dependent class of an existing Servlet class

- User Action: None (if reloading is enabled). If reloading is disabled, the server will have to be restarted to detect the change.
- Comments: The new version of the dependent .class file can be placed directly in the [product\\_installation\\_root](#)/installedApps /<app name>/<module name>/WEB-INF/classes directory. If the .class file is part of a jar file, the new version of the jar file can be placed directly in [product\\_installation\\_root](#)/installedApps/<app name>/ <module name>/WEB-INF/lib. In either case, the change will be detected, the web application will be shutdown and reinitialized, picking up the new class.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## Add a new Servlet using the Invoker (Serve Servlets by class name) facility or add a dependent class to an existing application

- User Action: None (if reloading is enabled). If reloading is disabled, the server will have to be restarted to detect the change.
- Comments: The new .class file can be placed directly in the [product\\_installation\\_root](#)/installedApps/<app name>/<module name>/WEB-INF/classes directory. If the .class file is part of a jar file, the new version of the jar file can be placed directly in [product\\_installation\\_root](#)/installedApps/<app name>/<module name>/WEB-INF/lib. This case is treated the same as changing an existing class. The difference is that adding the servlet/class does not immediately cause the web application to reload, since the class has never been loaded before. The class simply becomes available for execution
- Hot deployment: Yes
- Dynamic reloading: Non-applicable

## Add a new Servlet, including a new definition of the servlet in the web.xml deployment descriptor for the application

- User Action: None (if reloading is enabled). If reloading is disabled, the server will have to be restarted to detect the change.
- Comments: The new .class file can be placed directly in the [product\\_installation\\_root](#)/installedApps/<app name>/<module name>/WEB-INF/classes directory. If the .class file is part of a jar file, the new version of the jar file can be placed directly in [product\\_installation\\_root](#)/installedApps/<app name>/<module name>/WEB-INF/ lib. The web.xml file can be edited in place or copied into the [product\\_installation\\_root](#)/installedApps/<app name>/<module name>/WEB-INF directory. The new .class file will not trigger a reload of the application. However,

updating the web.xml file will cause the application to reload at the next reload interval. Once the reload is complete, the new servlet will be available for service.

- Hot deployment: Yes
- Dynamic reloading: Non-applicable

## Modification to the web.xml file of an application

- User Action: None (if reloading is enabled). If reloading is disabled, the server will have to be restarted to detect the change.
- Comments: You can change most of the parameters of the web.xml file without restarting the server. However, there are exceptions, listed here:
  - <security-constraints> A security constraint of the application
  - <security-role> Change, add, or delete security roles
  - <login-config> The login config values
  - To change the listed properties, you will need to restart the module using the DrAdmin command.
- Hot deployment: Yes
- Dynamic reloading: Yes

## Modification to the ibm-web-ext.xmi of an application

- User Action: None (if reloading is enabled). If reloading is disabled, the server will have to be restarted to detect the change.
- Comments: You can change all of the extension settings. The only warning is if you set the reloadInterval property to zero (0) or you set reloadEnabled property to false, the application will reload once at the time of the change. However, after this final reload, the web module will no longer detect any changes of any kind. Both of these changes disable the reloading function. The only way to re-enable reloading and the Dynamic functions is to change the appropriate property and restart the module using the DrAdmin command.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## Modification to the ibm-web-bnd.xmi of an application

- User Action: Restart the Module
- Comments: You can use the DrAdmin command to issue a restart of module of the application. The bindings will be reloaded at that time.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## EJB Jar Files

### Modification to ejb-jar.xml of an application

- User Action: Restart the Module
- Comments: You can use the DrAdmin command to issue a restart of module of the application. The changes will be reloaded at that time.

- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## **Modification to ibm-ejb-jar-ext.xmi**

- User Action: Restart the Module
- Comments: You can use the DrAdmin command to issue a restart of module of the application. The changes will be reloaded at that time.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## **Modification to ibm-ejb-jar-bnd.xmi**

- User Action: Restart the Module
- Comments: You can use the DrAdmin command to issue a restart of module of the application. The changes will be reloaded at that time.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## **Modification to Table.ddl for an EJB Jar**

- User Action: Rerun DDL file on user database server
- Comments: This change really has no effect on the application server and is a change to the database table schema for the EJBs
- Hot deployment: Non-applicable
- Dynamic reloading: Non-applicable

## **Modification to the Map.mapxmi file for an EJB Jar**

- User Action: Redeploy the EJB and restart the module
- Comments: A change to this file requires you to regenerate the deployed code artifacts for the EJB, apply the new EJB jar to the server and restart the module using the DrAdmin program.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## **Modification to the Schema.dbxmi file for an EJB Jar**

- User Action: Redeploy the EJB and restart the module
- Comments: A change to this file requires you to regenerate the deployed code artifacts for the EJB, apply the new EJB jar to the server and restart the module using the DrAdmin program.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## **Update the Implementation class for an EJB or a dependent class of the implementation class for an EJB**

- User Action: Restart the Module and/or Application (This is not working on the current driver)
- Comments: After changing the Jar/Class File in the installedApps folder, you must restart the module or application that the EJB is part of. You should restart the application if other modules in the application use the changed module. If the changed module is used by modules in other applications, you should restart those applications/modules.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## Update the Home/Remote interface class for an EJB

- User Action: Redeploy the EJB and restart the application
- Comments: A changing the interfaces of the EJB requires you to regenerate the deployed code artifacts for the EJB, apply the new EJB jar to the server and restart the application that the EJB jar is a part of. This will always work as long as you are using Module or Application visibility for class loading on the server.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## Add a new EJB to a running Server

- User Action: Apply the new Jar to the installedApps folder and restart the application using the DrAdmin program.
- Comments: For this scenario, I am assuming that the EJB is part of an existing EJB Jar file.
- Hot deployment: Yes
- Dynamic reloading: Yes

# Application

## Modification to the application.xml for an application

- User Action: Restart the Application
- Comments: The application must be restarted with the DrAdmin program.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## Modification to the ibm-app-ext.xmi file for an application

- User Action: Restart the Application
- Comments: The application must be restarted with the DrAdmin program.
- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## Modification to the ibm-app-bnd.xmi file for an application

- User Action: Restart the Application
- Comments: The application must be restarted with the DrAdmin program.

- Hot deployment: Non-applicable
- Dynamic reloading: Yes

## **Modification to a non-module Jar contained in the EAR**

- User Action: Restart the Application
- Comments: Use the DrAdmin tool to restart the application containing the Jar that has changed
- Hot deployment: Yes
- Dynamic reloading: Yes

## **Add a new EJB module to an existing, running application**

- User Action: Restart the Server
- Comments: Because adding a new module requires a change to server-cfg.xml, the server must be restarted to pick up the change.
- Hot deployment: No
- Dynamic reloading: No

## **Add a new web module to an existing, running application**

- User Action: Restart the Server
- Comments: Because adding a new module requires a change to server-cfg.xml, the server must be restarted to pick up the change.
- Hot deployment: No
- Dynamic reloading: No

## **Addition of new application to server-cfg.xml of running server**

- User Action: Restart the Server
- Comments: Because adding a new application requires a change to server-cfg.xml, the server must be restarted to pick up the change.
- Hot deployment: No
- Dynamic reloading: No

# **Server Configuration**

## **Modification to server-cfg.xml file of running server**

- User Action: Restart the Server
- Comments: Any change to server-cfg.xml requires the server to be restarted to pick up the change.
- Hot deployment: No
- Dynamic reloading: No



# HTTP Plugin Configuration Changes

## Modifications to application.xml to change the context root of a war file

- User Action: Generate the Plugin config file through the Web-browser admin or through the GenPluginCfg.bat/sh script.
- Comments:
- Hot deployment: Yes
- Dynamic reloading: Yes

## Modifications to web.xml to add, remove, or modify a servlet mapping

- User Action: Possibly have to re-generate the http plugin configuration file through the Web-browser admin or through the GenPluginCfg.bat/sh script.
- Comments: If the web application has file serving enabled or has a servlet mapping of '/', you do not have to regenerate the plugin configuration. In all other cases the regeneration is required.
- Hot deployment: Yes
- Dynamic reloading: Yes

## Modification to server-cfg.xml to add, remove, or modify a HTTP transport, to add or remove a Virtual Host, or to add, remove, or modify a Virtual Host alias

- User Action: Generate the Plugin config file through the Web-browser admin or through the GenPluginCfg.bat/sh file.
- Comments: Any of these changes affect the plugin-cfg.xml file and require you to regenerate that file.
- Hot deployment: Yes
- Dynamic reloading: Yes

## 6.6: Tools and resources quick reference

This article provides various ways to access information about administering the product and your applications being hosted by the product. Find information based on:

- [The tasks that you want to perform](#)
- [The resources that you want to administer](#)
- [The tools that you want to use](#)

### Tasks quick reference

Click a check mark in the table to go to the instructions for the task.

	Application Assembly Tool	AEs administrative console	Application client resource configuration tool
Configuring client resources			✓
Creating an application client	✓		
Starting and stopping resources		✓	
Pinging resources		✓	
Validate administrative configuration		✓	
Removing objects from EAR files	✓		✓
Configuring enterprise applications	✓	✓	
Creating the EAR.WAR and JAR files in an application	✓		
Installing enterprise applications		✓	
Starting, stopping, and restarting applications and modules		✓	
Uninstalling applications		✓	
Updating application configurations	✓	✓	
Exporting application configurations		✓	
Viewing deployment descriptor information for enterprise applications (read-only)		✓	


Showing the status of enterprise applications		✓	
Exporting enterprise applications		✓	
Updating nodes		✓	
Regenerating WebSphere plug-in configurations		✓	
Configuring tracing on administrative nodes		✓	
Starting and stopping application servers		✓	
Updating application servers		✓	
Updating EJB containers		✓	
Creating an EJB module	✓		
Installing EJB modules		✓	
Viewing deployment descriptor information for EJB modules		✓	
Updating EJB module configurations		✓	
Exporting table DDLs of EJB modules		✓	
Updating Web container configurations		✓	
Creating a Web module	✓		
Installing Web modules		✓	
Precompiling JSP files for Web modules		✓	
Viewing deployment descriptor information for Web modules		✓	
Updating Web module configurations		✓	
Updating session management settings		✓	
Configuring new HTTP transports		✓	
Removing HTTP transport configurations		✓	
Updating transport configurations		✓	
Configuring new data source providers		✓	✓

Configuring new data sources		✓	✓
Removing data source providers (JDBC providers) and data sources		✓	✓
Updating data source and data source provider configurations		✓	✓
Configuring new custom services		✓	
Removing custom services		✓	
Updating custom service configurations		✓	
Administering virtual hosts		✓	
Configuring security	✓	✓	
Administering server traces		✓	
Administering the transaction service		✓	
Configuring application server process definitions		✓	
Updating process definition configurations		✓	
Administering path maps		✓	
Configuring OLT and Debugger		✓	
Configuring the Location Service Daemon		✓	
Configuring Object Request Brokers (ORBs)		✓	
Configuring JVM settings		✓	
Configuring new mail sessions		✓	✓
Updating mail session configurations		✓	✓
Removing mail session configurations		✓	✓
Configuring new URL providers		✓	✓
Updating URL provider configurations		✓	✓
Removing URL provider configurations		✓	✓
Removing URL configurations		✓	✓

Configuring new JMS providers		✓	✓
Configuring new JMS connection factories		✓	✓
Configuring new JMS destinations		✓	✓
Updating JMS provider, connection factory, and destination configurations		✓	✓
Removing JMS providers, connection factories, and destinations		✓	✓
Updating JMS provider, connection factory, and destination configurations		✓	✓
Administering WebSphere administrative domains		✓	
Regenerating the Web server plug-in configuration		✓	

## Resources quick reference

The following tables provide quick entry points into descriptions, tasks, and settings of various objects that you can administer.

 See also the [administrative model](#), which enables you to click a resource in a topology diagram to go straight to the information about the resource.

## Assembling applications and generating code for deployment

Description	Administrative overview	Settings
<a href="#">Application assembly</a>	See <a href="#">Application Assembly Tool</a>	See properties for applications, EJB modules, Web modules, and application client modules.
<a href="#">Generating code for deployment</a>	See <a href="#">Application Assembly Tool</a>	<a href="#">Assembly properties for generating deployment code</a>

## Configuring the overall administrative environment

Description	Administrative overview	Settings
<a href="#">Administrative domains</a>	<a href="#">Administering WebSphere administrative domains</a>	<a href="#">Administrative domain properties</a>
<a href="#">Nodes</a>	<a href="#">Administering nodes</a>	<a href="#">Node properties</a>
<a href="#">Virtual hosts</a>	<a href="#">Administering virtual hosts</a>	<a href="#">Virtual host properties</a>

## Configuring specific application servers

Description	Administrative overview	Settings
Application servers	Administering application servers	Application server properties
Application server process definitions	Administering application server process definitions	Application server process definition properties
JVMs	Administering JVMs	JVM properties
EJB containers	Administering EJB containers	EJB container properties
Web containers	Administering Web containers	Web container properties

## Configuring application components installed in a runtime

Description	Administrative overview	Settings
Applications	Administering applications	Application properties
EJB modules	Administering EJB modules	EJB module properties
Web modules	Administering Web modules	Web module properties
Application client modules	Administering application client modules	Application client module properties

## Configuring resource providers for data access and other functions

Description	Administrative overview	Settings
Resource providers	Administering resource providers	Resource provider properties
JavaMail resources	Administering JavaMail providers and sessions	JavaMail provider properties and JavaMail session properties
URL resources	Administering URL providers and URLs	URL provider properties and URL properties
JMS resources	Administering JMS providers	JMS provider properties and JMS connection factory properties and JMS destination properties
Data access	Administering database connections	JDBC provider properties Data source properties

## Configuring various services to support applications in the runtime

Description	Administrative overview	Settings
Security	Administering security	Security properties
Sessions and Session Managers	Administering Session Managers	Session Manager properties
User Profile Managers	Administering User Profile Managers	Non-applicable -- no settings in console
Tracing	Administering the product trace	Trace properties
Transactions	Administering transactions	Transaction properties
Custom services	Administering custom services	Custom service properties

<a href="#">Location Service Daemons</a>	<a href="#">Administering Location Service Daemons</a>	<a href="#">Location Service Daemon properties</a>
<a href="#">ORBs</a>	<a href="#">Administering ORBs</a>	<a href="#">ORB properties</a>
<a href="#">Path maps</a>	<a href="#">Administering path maps</a>	<a href="#">Path map properties</a>
<a href="#">OLT and Distributed Debugger</a>	<a href="#">Administering OLT and Distributed Debugger</a>	<a href="#">OLT and Distributed Debugger properties</a>

## Configuring resources related to routing and distributing application requests

Description	Administrative overview	Settings
<a href="#">Transports</a>	<a href="#">Administering transports</a>	<a href="#">Transport properties</a>
<a href="#">Plug-ins for Web servers</a>	<a href="#">Administering WebSphere plug-ins for Web servers</a>	<a href="#">Plug-in properties</a>

## Tools quick reference

Use the following information to compare, select among, and learn how to use the features of the various product interfaces.

- [Summary of available product interfaces \(consoles and tools\)](#)
- [Scope of administrative capabilities](#)

## Summary of available product interfaces (consoles and tools)

The table summarizes the available product interfaces (with the exception of the [installation program](#) and the problem determination tools).

Interface	Purpose
<a href="#">Script for starting the product</a>	Perhaps the most important interface you will use is the script for starting the product, which you need to do prior to starting the administrative console or the application server runtime.
<a href="#">Administrative console (Web based)</a>	Graphical interface for configuring and operating application servers and other resources that support applications in a runtime environment
<a href="#">Application assembly tool</a>	Graphical interface for assembling applications from application components and modifying their J2EE deployment descriptors
<a href="#">Configuration using property files</a>	Ability to manually edit various property files in the product directory structure for miscellaneous configuration purposes
<a href="#">Application installer (SEAppInstall)</a>	Command line interface for installing an application into a server configuration file, preparing it to run within an application server. You can also use it to uninstall an application from a server and to export an installed application containing configuration information.
<a href="#">Client launcher (launchClient)</a>	Command line interface for starting a client application.
<a href="#">EAR expander</a>	Command line interface for expanding .ear files into the directory structure typical for use in the runtime of this product. Should not need to use this tool on an ordinary basis.

<a href="#">Application client resource configuration tool</a>	Graphical interface for configuring resources that support client applications in a runtime environment. (See also <a href="#">information on developing a Java client</a> .)
<a href="#">SOAP EAR Enabler</a>	Command line interface for enabling a set of SOAP services within an Enterprise Application Archive (EAR) file.
<a href="#">XML-SOAP Admin tool</a>	A modified version of the Apache SOAP XML-Admin interface for managing each SOAP-enabled EAR file.
<a href="#">Deployment tool</a>	Interface for generating code for deployment. Useful for more complicated or custom scenarios. For simple scenarios, the use of this tool is hidden from the administrator. It is invoked by other consoles and tools. (See also <a href="#">the article about generating code for deployment</a> ).

## Scope of administrative capabilities

Administrators that have access to an administrative client can use the full capabilities of the client. At this time, it is **not** possible to allow administrators to perform some operations (such as starting and stopping servers) but deny them the ability to perform other operations (such as creating and configuring resources).



## 6.6.0: About user assistance

The InfoCenter provides a copy of the help files (or user assistance files) that are installed with the product. In some cases, the InfoCenter versions of the help files are more up to date.

Articles named 6.6.0.\* approach administration by describing how to use the available tools. Each set of articles about a particular administrative tool provides entry points into tasks that you can perform with the tool.


Articles named 6.6.1 through 6.6.N complement the tool-based documentation by providing entry points to administering various object types, such as nodes and application servers. Each entry point provides links to tasks and tools related to administering the object type.

## 6.6.0.3: Web administrative console overview

The Web administrative console is a *configuration editor* that runs in a Web browser. It provides the opportunity to work with [application server configuration files](#) encoded in eXtensible Markup Language (XML).

See the sub-topics and other related information for instructions for using the features of this administrative console. The remainder of this article will be used to highlight key points and provide brief usage tips.

### What to do if "Additional problems have been detected"

 If you see the "Additional problems have been detected" message at the top of the right side of the console, it signals problems in your server configuration file. You might not want to save your configuration file until you solve the problems.

See [the problem page topic](#) for more information, including a methodology for investigating problems.

### Enabling keyboard accessibility mode when using a Netscape browser

Netscape browsers have an inherent limitation wherein the Javascript tree on the left pane cannot be navigated using the keyboard alone.

To have keyboard accessibility to the tree, complete the following steps:

1. Click on Preferences at the top of the main panel.
2. Click on the "No Javascript tree" link to load the purely html, no Javascript tree in the left pane.

This tree now has keyboard accessibility.

## 6.6.0.3a: Starting, stopping, and logging into the Web administrative console

- [Starting the console](#)
- [Logging in](#)
- [Logging out](#)
- [Stopping the console](#)
- [Note to Linux users](#)

### Starting the console

To start the Web administrative console:

1. Ensure that the Web server is running on the machine containing the console.
2. [Ensure that the application server is also running.](#)
3. In a Web browser, type:

`http://your.server.name:9090/admin`

where *your.server.name* is `localhost` if the Web console is on the local machine. On Windows 2000, it has been found that `localhost` is not always recognized. In such a case, use the actual host name.

If the console is on a remote machine, enter the short or fully qualified host name for the machine containing the administrative server.

4. Wait for the console to load into the browser.

### Logging into the console

A **Login** screen is displayed whenever you open the console.

1. Enter your user ID. (See below for some notes).
2. If you have done the following, then the console has been *secured*, meaning you must enter a password:
  1. [Specified security user IDs and passwords](#)
  2. [Enabled global security](#)
3. Click **Submit**.

### Notes about the user ID for logging in

Even if you have not secured the console, you will be prompted for an ID. The server configuration file will be saved under the user ID if there is a [session timeout](#). If you do not specify an ID, a default username called "User" will be assigned to the session.

A user ID lasts for the duration of the session for which it was used to log in. If you enter an ID that is already in use (and in session), you will be prompted to do one of the following:

- Force the existing user ID out of session. The configuration file that was being used by the existing user ID will be saved in the temp area. See the [session timeout description](#) for further details.
- Wait for the existing user ID to log out or timeout of the session.

- Specify a different user ID.

## Logging out

When you are logged in, a **Logout** option is available in the menu on the Web console home page.

1. Select **Logout** from the menu.
2. If you already saved your changes, you are logged out right away. Otherwise, if you need to save your changes, the **Logout** panel is displayed.
3. Save your changes to the same file that you were editing or save it to another file.
4. Click **OK** to log out.

## Stopping the console

To stop the console, simply close the browser.

## Note to Linux users

If you have difficulty using the Web-based administrative console on Linux with a Netscape browser, one of the following workarounds could help:

- If the tree on the left side of the console is not being displayed, configure Netscape preferences to Enable Java and Enable JavaScript. Try to access the console again.
- Try the Blackdown 1.2.2 JDK plug-in for Linux
- Use a Web browser on a different operating system or machine to access the Linux machine remotely
- Try Mozilla with the JDK 1.3 plug-in on Linux

Specifically, problems have been encountered on SuSe 6.4 and SuSe 7.1 using Netscape Communicator Version 4.72. Other Linux systems are susceptible, as testing suggests that the problem is partially or wholly attributable to the JDK plug-in of the Web browser.

When one tries to start the console, the left frame freezes while loading, with the message "View no script tree" at the top.

## 6.6.0.3b: Working with server configuration files, using the Web administrative console

- Use the default configuration file
- Open another configuration file after starting the console
- Specify another configuration file to open when the console starts
- Review and correct potential problems with a configuration file before saving it
- Save a configuration file
- Save a configuration under a different name
- Create a new configuration file
- Temporary configuration files resulting from session timeouts

### Using the default configuration file

By default, when you open the console, the default [server configuration file](#) is loaded.

The following tasks describe how to switch to a different configuration file, either after opening the console, or for the next time you open the console.

### Opening other server configuration files after starting the console

To open a server configuration file other than one that is currently loaded:

1. *Recommended.* Save the configuration file that you are working with now, so that its changes are not lost.
2. View the **Configuration Files** panel by either:
  - Clicking **Open a configuration file to edit with the console** link on the console home page.
  - Clicking the **Configuration** menu item.
3. On the **Configuration Files** panel, specify the configuration file to open. To do so, you can either:
  - Select from among the files in the `config` directory.
  - Enter the full path to the file on the server.
4. Click **OK**. The specified file will be loaded.

### Specifying a server configuration file to load when the console opens

When you open the administrative console, you can specify to use a particular configuration file by passing the file name as a parameter in the browser URL.

For example, the browser URL could be:

```
http://localhost:9091/admin/edit?configFile=c:/temp/myConfigFile.xml
```

The above example will load `myConfig.xml` from the `c:/temp` directory.

### Reviewing and correcting potential problems with a configuration file

It is a good practice to review the problems page and resolve any problems prior to saving your server configuration.

If you make any changes that could result in potential problems with the server configuration file, a link specifying "additional problems have been reported" will be displayed at the top of every page that you view. Click the link to view the problems page.

Prior to saving the configuration file, read the descriptive text about the problems, as some types of problems can result in unwanted behavior if an application server is restarted using the configuration file that has the problems.

The [problem page format](#) includes an Object ID that you can use to locate each problem in the configuration file:

1. *Recommended.* Save a backup copy of the configuration to a temporary file (see below for instructions) in case subsequent changes aggravate the problem.
2. Open the original server configuration file in a text editor or XML editor and search for the problem ID.
3. The XML element with the problem ID is the element that has problems.
4. Return to the console, and look at the configuration for the object type pertaining to the element that was found to have problems. For example, it could be the XML element pertaining to the virtual host object in the console. Adjust the configuration to eliminate the error.
5. Determine whether the "additional problems have been reported" link is still visible (indicating that errors remain).
6. Repeat the above steps until all errors have been eliminated.
7. When all errors have been eliminated, save the configuration file.

## Saving configuration files

To save a configuration file:

1. View the **Save Configuration** panel by either:
  - Clicking **Save** link on the console home page.
  - Clicking the **Save** menu item.
2. Select the **Save** radio button.
3. Click **OK**.

## Saving configuration files under different names

To save a configuration file using a different name than its current name:

1. View the **Save Configuration** panel by either:
  - Clicking **Save** link on the console home page.
  - Clicking the **Save** menu item.
2. Select the **Save As** radio button.
3. Specify the full server path and name of the new file.
4. Click **OK**.

A copy of the current file is created under the new name and your changes are saved to the new file. (The original configuration file is left unchanged.) The new file is loaded into the console.

## Creating new configuration files

To create a new configuration file by using an existing file as a template.

1. *Recommended.* Save the configuration file that you are working with now, so that its changes are not lost.
2. Click the **Create** link on the console home page to display the **Configuration Files** panel.
3. Select an existing configuration file to use as template. To do so, you can either:
  - Select from among files in the `config` directory.
  - Enter the full path to the file on there server.
4. Specify the name of the new file to create. Specify the full path to the file on the server.
5. Click **OK**. The new file will be created and loaded into the console.

## Temporary configuration files resulting from session timeout

If the console is not used for 15 minutes or more, the session times out. The something happens if you close the browser window without saving the configuration file. The changes to the file will be saved to a temporary file when the session times out, after 15 minutes.

When a session times out, the configuration file in use is saved under the *userid*/timeout directory under the ServletContext's temp area. This is value of the `javax.servlet.context.tempdir` attribute of the `ServletContext`. By default, it is:

`product\_installation\_root/temp/hostname/Administration/admin/admin.war`

You can change the temp area by specifying it as a value for the `tempDir` init-param of the action servlet in the deployment descriptor (`web.xml`) of the administrative application.

The next time you log on to the console, you are prompted to load the saved configuration file. If you decide to load the saved file:

1. If a file with the same name exists in the `product\_installation\_root/config` directory, that file is moved to the *userid*/backup directory in the temp area.
2. The saved file is moved to the `product\_installation\_root/config` directory.
3. The file is then loaded.

If you decide not to load the saved file, it is deleted from the *userid*/timeout directory in the temp area.

The configuration file is also saved automatically when the same user ID [logs into the non-secured console](#) again, effectively starting a different session. This process is equivalent to forcing the existing user ID out of session, similar to a session timeout.

## 6.6.0.3b.1: Format of problems page for server configuration

Problems listed on the problems page produced in the Advanced Single ServerEdition administrative console contain the following information about problemsfound in the XML server configuration file.

### ID

An identifier for the type of problem. Message IDs follow the following format:

[CategoryId][Code][W/E]

where:

#### Category ID

A four-digit alpha numeric category ID which specifies a functional area of the system. The category will typically be CHKW.

#### Code

A four digit sequence number that specifies a particular type of problem.

#### W/E

The severity of the problem -- Warning or Error.

A warning message might or might not cause runtime problems. Even if the problem is left unresolved, the server should be able to start.

An error message will typically cause runtime problems, such as a feature not being available, or unreliable or inconsistent operation.

### Description

A brief description of the type of problem being reported.

### [Object ID]

An identifier for an element of the configuration that contains the problem.



## 6.6.0.3d: Setting preferences in the Web administrative console

To set preferences for the current work session, click the **Preferences** menu item to display the **Preferences** panel. The preferences are not saved. The next time you open the console, the default preferences are used.

Two preferences are available.

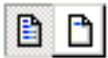
### Show IBM banner

By default, this option is enabled. If this option is deselected, the IBM banner displayed in the console header area is hidden.

### Show descriptions

By default, this option is enabled, meaning that field level help and other instructions are displayed on the console panels. If this option is deselected, the field level help and other instructions are not displayed on the console panel.

Alternatively, click the Show/Hide Description Toggle image displayed on the far right of the menu:



## 6.6.0.3e: Using the Web administrative console

The Web administrative console has two areas:

- A navigation area on the left
- A work area on the right that displays property forms and other worksheets

The navigation area contains a tree of administrative tasks and resources. To perform a task or configure a resource, click the task or resource in the tree.

When you *expand* a task or resource by clicking the expansion widget to its left, you can view the tasks and resources contained by that task or resource in the tree hierarchy. When the task or resource is *collapsed*, the tasks and resources it contains are hidden.

When you *select* a task or resource by clicking the label of the task or resource, information about it is displayed in the work area. In most cases, the work area contains property fields you can read or edit, with buttons at the bottom.

## 6.6.0.4: Overview of editing property files by hand

In general, it is recommended that you use the administrative toolsto edit configuration (property) files, rather than editing them directly. Here are references to the main opportunities for editing product property files by hand:

- [Server configuration file](#)
- [Dynamic caching feature](#)

## 6.6.0.5: Using the Application Assembly Tool interface

The Application Assembly Tool interface includes the following:

- A main window for creating, viewing, and editing the content and assembly properties of Web modules, EJB modules, application client modules, and J2EE applications.
- Property dialog boxes for creating new objects and for accessing and modifying properties for a module. The tool uses the property values to automatically generate deployment descriptor files for the module.
- Wizards for quickly creating modules. Wizards gather minimum requirements for creating the module and generating the deployment descriptor. They prompt for required information and use default values where applicable.

The main Application Assembly Tool window consists of two panes: the navigation pane on the left and the property-editing pane on the right. The navigation pane provides a view of the contents of the module being created or edited. The property-editing pane displays property dialog boxes for the component or property group selected in the opposite pane.

- The **navigation pane**. This is a hierarchical (tree) view of a module's contents, assembly properties, and files. The first object in the tree is the display name of the module or application. An icon in the tree represents each of the following:
  - Modules in the application (EJB modules, Web modules, or application client modules).
  - Components of a module (for example, entity beans and session beans for EJB modules).
  - Groups of properties for the module or component. For example, Web components have properties for security role references, initialization parameters, and page lists.
  - Files making up the module.

The navigation pane is used to review the content and structure of a module and to select components or property groups. Selection of a component or property group allows its properties to be entered or edited in the property pane.

- The **property pane**. By default, this pane displays property dialog boxes for the item selected in the opposite pane. If the selected item is an object type (for example, entity bean), the instances of the type are displayed in the top portion of the pane. The properties for the instance are displayed on the bottom portion of the pane. When an instance is selected, the corresponding property dialog box is displayed in the bottom portion of the pane.

When the property pane is disabled, its contents are dependent on the item that is selected in the navigation pane, as follows:

- If a module folder is selected (for example, Web Modules), the names of all WAR files are displayed.
- If an archive file is selected (for example, a JAR file), the property groups for that archive are displayed.
- If an object type is selected (for example, Web Components), the instances of that object type are displayed. In this example, servlets and JSP files are displayed.
- If an object instance is selected (for example, an entity bean instance), the property groups for the instance are displayed.
- If a property group icon is selected, the property values are displayed in table form.
- If the Files icon is selected, a complete list of files for the module are displayed.

The property pane can be disabled by clicking **View->Show Property Pane** to toggle the selection.

You can adjust the width of the panes by dragging the split bar left or right. In the property-editing pane, you can

also adjust the width of the columns of properties by dragging the edge of the column left or right.

---

## The navigation pane

The navigation pane displays contents and associated assembly properties in an indented tree outline. The tree hierarchy for an application (EAR file) is as follows:

- Application display name
  - EJB Modules (folder)
    - JAR files
      - Session beans
        - Session bean instances
        - Properties for session bean instances
      - Entity beans
        - Entity bean instances
        - Properties for entity bean instances
      - Security roles
      - Method permissions
      - Container transactions
      - Files
    - Web Modules (folder)
      - Web applications
        - Web components
          - Web component instances (servlets or JSP files)
          - Properties for the Web component instance
        - Properties for the Web application
        - Assembly property extensions
        - Files
    - Application Clients (folder)
      - Application client instances
      - Properties for application client instances
    - Security Roles (for the application)
    - Files

Clicking the plus (+) and minus (-) symbols (or double-clicking an item) expands and collapses the tree to reveal the content of each folder or icon. The tree can also be navigated by using the up- and down-arrow keys to move through the branches and leaves and the left- and right-arrow keys to expand and collapse the tree. Folders and icons can be expanded to reveal their contents. For example, when the Web Modules folder is expanded, the individual Web applications (WAR files) contained in the module are revealed. When the icon for a Web application is expanded, icons for the Web components (servlets or JSP files), assembly properties, and individual files in the application are revealed.

In a J2EE application, the first level of the hierarchy is the display name of the application. The second level consists of folders, one for each type of module in the application and one for the security properties that apply to the application as a whole. Beneath each folder representing a type of module, the module's components are

listed. Forexample, beneath the EJB modules folder, icons represent the individual JAR files in the module. Beneath the JAR file, icons represent session beans, entity beans, assembly properties, and files in the JAR. Clicking the icon for entity beans reveals all entity bean instances in the module. Expanding an entity bean instance reveals icons for various categories of properties for the bean.

Clicking an item (an archive file, enterprise bean instance, or property icon, forexample) in the navigation pane causes its corresponding property dialog box to be displayed in the property pane (if properties have been set).

---

## The property dialog boxes

The property dialog boxes allow you to define the contents of a new module, make modifications to an existing module, or duplicate a module in full or in part by cutting and pasting archive files and property values. This method is especially useful for assembling multiple modules with similar contents. Using a property dialog box also provides access to the complete listing of properties, whereas the wizards expose only the minimum required and more-frequently used properties that must be defined for the module. Default values are used where applicable. On the property dialog box, the fields for required properties have a red asterisk (\*) next to them. When using the property dialog box, you can access help for any of the properties or examine the definitions of the default properties already provided.

A property dialog box is used to enter, view, or modify values. Each property dialog box contains one or more of the following tabs. Clicking a tab opens a window containing a group of properties. Note that not all tabs are present for all property groups.

- **General.** These properties represent standard J2EE deployment descriptors.
  - **Icons.** These properties specify the location of JPEG or GIF files used as icons to represent the component in a GUI.
  - **IBM Extensions.** The IBM deployment descriptor extensions are additions to the standard descriptors for J2EE applications, Web applications, and enterprise beans. The extensions allow you to specify properties that enable Enterprise Edition or legacy (older) systems to work in the WebSphere Application Server environment. They also allow you to specify behavior that is vendor specific, undefined in a current specification, or expected to be included in a future specification. For example, isolation level attributes determine how isolated one transaction is from another. This property can be set for individual methods in an enterprise bean or for all methods in the enterprise bean. Isolation level attributes are defined in the extensions layer.
  - **Bindings.** Binding information contains references to the external resources and enterprise beans required by a module. This information is used by the container to locate a resource in the JNDI namespace. The logical names specified for binding properties are later resolved (in the administrative console) to the registered JNDI name for an enterprise bean's home interface and for resource manager connection factories. The bindings specified for a module in the Application Assembly Tool can be changed prior installing the module in an application server.
- 

## The wizards

The wizards allow you to quickly create a module with the minimum required information. Using wizards is ideal for testing and prototyping purposes. If you wish to specify additional properties for your module, such as extensions and binding information, use the property dialog boxes. (After using a wizard, you can continue to edit the properties of a module or add new properties; this is done by using the property dialog boxes.) The property dialog boxes allow you to specify values for all properties, whereas the wizards prompt you to specify required properties and other more frequently used properties. On wizards, the fields for required properties have a red asterisk next to them. When using the wizards, you can access help for any of the properties or examine the definitions of the default properties already provided.

## 6.6.0.5.1: Modifying and adding modules to applications

You can modify a module by adding or editing properties, adding components, or removing components. If you are modifying an application (EAR file), you can add new modules or import them. The steps used to modify a module are the same regardless of the type of module. After you make changes to a module, you must save the archive again. If the module contains enterprise beans, you must regenerate deployment code.

This file contains instructions for the following tasks:

- [Adding or editing assembly properties](#)
- [Adding or importing modules into applications](#)
- [Copying and deleting JAR and WAR files](#)
- [Adding enterprise beans and Web components](#)
- [Adding files](#)

Note that after deployment code has been generated for an application, the deployable archive is renamed with the prefix `Deployed_`. Any subsequent changes to the archive from within the Application Assembly Tool are applied to the version of the archive that existed prior to code generation. To see changes reflected in your application, you must regenerate deployment code and re-install the deployable archive.

---

### Adding or editing assembly properties

To add properties or edit the existing properties of a module, perform the following steps:

1. Make sure that the property pane is enabled. From the main menu, click **View**. If necessary, toggle the **Show Property Pane** menu item so that it is checked.
2. If the module is not already opened, click **File->Open** and select the archive file for the module. The contents of the module are displayed in an Application Assembly Tool window.
3. In the navigation pane, click the icon for the group of properties that you want to view or edit. If property values exist, a property dialog box is displayed in the property editing pane. If no property values exist, right-click the property icon and choose **New**. A property dialog box is displayed.
4. Enter or edit values. Click **OK** to close the property dialog box. Click **Apply** to apply the values but to keep the dialog open for additional edits. Click **Help** to view field help.

There are several alternative methods of accessing the property dialog boxes. These methods apply only to archive files (JAR, WAR, and EAR files) or to module components (enterprise beans or JSP files, for example).

- Right-click the icon representing the archive or component. Choose **Properties**. A property dialog box is displayed. Edit the properties and click **OK**.
  - Click the icon representing the archive or component. Click the **Properties** icon on the tool bar. A property dialog box is displayed. Edit the properties and click **OK**.
- 

### Adding or importing modules into applications

You can add any combination of JAR or WAR files to an enterprise application (EAR file). There are several ways to include archive files:

- Import an existing JAR or WAR file.

- Open an existing EJB or Web module and copy and paste individual beans or Web components into the application.
- Create a new JAR or WAR file.

The procedure for each task is as follows:

- To import archive files into the application, right-click the icon corresponding to the type of module (EJB module, Web module, or application client module) and choose **Import**. Use the file browser to locate and select the archive file. Click **Open**. The JAR or WAR file is displayed under the appropriate folder. The imported module's properties are also imported. Click the plus sign(+) next to the archive's icon to view its contents and edit its properties if needed.
- To copy portions of an existing module, open the module by clicking **File->Open** and selecting the file. Arrange the windows so that both the EAR file and the source archive are visible. Use a copy-and-paste operation to copy archives or components to the current module.
- To create a new JAR or WAR file to be included in the application, right-click the folder corresponding to the type of module to create (EJB Modules, Web Modules, and Application Clients), and choose **New**. Enter required values as indicated. Optionally, enter values for other properties. Click **OK**. The JAR or WAR file is displayed under the appropriate folder. Click the plus sign (+) to verify contents and enter assembly properties.

For EJB modules, you must add at least one enterprise bean. Right-click the folder corresponding to the type of bean to create (session bean or entity bean), and choose **New** or **Import**. For entity beans, choose whether you are creating a CMP or BMP bean. A property dialog box is displayed. Enter required values as indicated. Optionally, enter values for other properties. Click **OK**. The enterprise bean is displayed in the navigation pane. Click the plus sign (+) to verify the contents and enter assembly properties.

## Copying and deleting JAR and WAR files

To copy a JAR or WAR file from one module to another:

1. In the navigation pane of the source module, right-click the icon representing the JAR or WAR file to be copied. Choose **Copy**.
2. In the navigation pane of the target module, right-click the folder representing EJB modules or Web modules and choose **Paste**. Confirm the values displayed in the dialog box. If the target module already contains an archive with the same name, a message is displayed stating that the archive cannot be overwritten. (You can ignore the warning and later rename the archive and modify its properties.)

To delete a JAR or WAR file from a module:

1. In the navigation pane, right-click the JAR or WAR file. Choose **Delete**.

## Adding enterprise beans and Web components

To add enterprise beans or Web components, use any of the following methods:

- Import the item into the module. In the navigation pane, right-click the Web modules, EJB modules, or Application Client folder and choose **Import**. Browse the file system and locate the desired archive file. When the file is located, click **Open**. If you are importing an EAR file, you are prompted to select which WAR or JAR files to import. Click the desired archive and click **OK**. As prompted, enter or verify properties for the imported archive.
- Use a copy-and-paste operation to copy components from an existing module.
- Create a new item by right-clicking the appropriate icon (for example, the icon representing entity beans



or Web components) and choose **New**. Click **Browse** to locate the class files of the enterprise bean or Web component. By default, the root directory or archive is the current archive. If needed, browse the file system for the directory or archive where class files reside. After you choose a directory or archive, its file structure is displayed. Expand the structure and locate the files that you need. Select the file and click **OK**. In the property dialog box, click **OK**. Verify that the enterprise beans or Web components are added to the module (expand the hierarchy in the navigation pane).

---

## Adding files

To add files, perform the following steps:

1. Right-click the Files icon and choose **Add Files**. If you are adding files to a Web module, right-click the **Class Files**, **Resource Files**, or **JAR Files** icon.
2. Select a root directory or archive. The root directory is used to determine the relative path of the files being added. If you are adding an entire archive, select the directory that contains the archive. First, click **Browse** to navigate to the desired directory or archive. Then select the directory and click **Select**. The directory structure is displayed in the left pane.
3. Use the left pane to navigate the contents of the directory. The right pane displays the contents of the selected directory in the left pane.
4. From the list of files displayed in the right pane, select the desired files, and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. Relative path names are maintained. To select multiple files, press and hold the Control key while selecting the files. To add a range of files, select the initial file, press and hold the Shift key, and then select the last file in the range.
5. The files to be added are displayed in the Selected Files pane. When the Selected Files pane contains the correct set of files, click **OK**.

## 6.6.0.5.2: Generating deployment code for modules

Before installing your application in WebSphere Application Server, you must generate deployment code for the application. This step is required for EJB modules and for any EAR files that contain EJB modules. During code generation, the Application Assembly Tool invokes the Deployment Tool for Enterprise JavaBeans (ejbdeploy) to do the following:

1. Generate and compile container implementation classes.
2. Process the implementation classes by using the RMIC compiler, which generates the client-side stub and server-side skeletons needed for remote method invocation (RMI). (You can bypass this step by selecting the Code Generation Only checkbox on the property dialog box.)
3. Validate the JAR file and display error messages if there are any J2EE specification violations within the JAR file.
4. For CMP entity beans, generate persistence code.

For CMP entity beans, if the JAR file contains a map and schema document, that schema is used. If the JAR file does not contain a map and schema document, the Application Assembly Tool uses a top-down mapping to generate the following files:

- META-INF\Table.ddl
- META-INF\map.mapxmi
- META-INF\Schema\schema.rdbxmi

### Note:

For Advanced Single Server Edition of this product, only the Table.ddl file is generated.

The user must create the database table manually. The generated Table.ddl file can be used as a guide, or the user can create the table by some other means.

If you do not wish to use the default top-down mapping, the recommended tool is VisualAge for Java. For information on default mappings, see the documentation for Deployment Tool for Enterprise JavaBeans.

To migrate CMP entity beans from WebSphere Application Server 3.5 for use in version 4.0, use the -35 option of the Deployment Tool for Enterprise JavaBeans. This option is recommended only if you need to preserve use of the mapping rules that were used in the 3.5 version of the Deployment Tool.

To generate deployment code for a module in the Application Assembly Tool, perform the following steps:

### Note:

A working directory is used to save temporary files while creating the deployed archive. On UNIX systems, if there is insufficient space in this directory, a segmentation violation error can occur. Make sure that there is sufficient space in the working directory. Required space varies depending on the application. By default, the working directory is /tmp on UNIX systems. You can specify a different directory in the Working directory field on the property dialog box.

1. Click **File->Open** and select the module. The module is displayed in the main window.
2. Right-click the name of the module at the top of the navigation pane and choose **Generate code for deployment**. A property dialog box is displayed. Enter or edit values for [6.6.35.0: Assembly properties for generating deployment code](#). By default, the newly created archive file has the same name as the current archive file with the prefix `Deployed_`. Click **Generate Now**.
3. View status messages in the window and resolve any errors reported.
4. To close the window, click **Close**.

If the modules in an application are already deployed, the Application Assembly Tool regenerates deployment

code for them.

Note: Ensure that your JAR files do not contain source code(.java files). On certain platforms, JAR files containing sourcecode can cause compilation errors during deployment code generation.

## 6.6.0.5.3: Verifying archives

Verifying archive files is important for successful generation of deployment code for the application. During verification, the Application Assembly Tool checks that an archive is complete, and that deployment descriptor properties and references contain appropriate values. Specifically, the verification process checks the following:

- Required deployment properties must contain values.
- Values specified for environment entries must match their associated Java types.
- In both EAR and WAR files:
  - For EJB references, the target enterprise bean of the link must exist.
  - For security role references, the target role must exist.
  - Security roles must be unique.
- For EAR files, each module listed in the deployment descriptor must exist in the archive.
- For WAR files, the files for icons, servlets, error, and welcome pages listed in the deployment descriptor must have corresponding files in the archive.
- For EJB modules:
  - All class files referenced in the deployment descriptor must exist in the JAR file.
  - Method signatures for enterprise bean home, remote, and implementation classes must be compliant with the EJB 1.1 specification.

To verify a module in the Application Assembly Tool:

1. Click **File->Open** and select the module to be verified.
2. Right-click the name of the module at the top of the navigation pane and choose **Verify**.
3. In the **Verify** window, click **Verify**. The tool displays a scrolling window for viewing status messages as the verification proceeds.

## 6.6.0.5.4: Viewing deployment descriptors

You can view the deployment descriptor for a J2EE application or for any module contained in the application. To view a deployment descriptor:

1. Click **File->Open** and select the module whose deployment descriptor file is to be viewed.
2. In the navigation pane, right-click the name of the EAR file to view the application-level deployment descriptor. Right-click the name of an EJB module or Web module to view a module-level deployment descriptor. Choose **View Descriptor**. The tool displays a window for viewing the XML file.
3. Click **Close** to close the window.
4. If necessary, modify the values of the deployment descriptor by editing the module's properties and then saving the archive. Repeat this procedure to view your modifications.

## 6.6.0.5.5: Specifying JNDI bindings and resolving references

Binding information is used by a container to locate external resources such as enterprise beans and resource connection factories required by an application. You can specify the JNDI names of resources when you install an application by using the administrative console. Alternatively, you can specify the JNDI names in the Application Assembly Tool. Binding properties specified in the tool can be changed in the administrative console prior to installing an application. You can specify binding information for the following:

- Enterprise beans.
- Data sources to be used by entity beans.
- References to enterprise beans and resource manager connection factories. For example, an application client, enterprise bean, or servlet can be a client to another enterprise bean. The referencing bean or servlet declares an EJB reference and optionally binds the reference.
- Virtual host names for Web applications.

If you are using the Application Assembly Tool to create application client modules, you must also use the Application Client Resource Configuration Tool if you have local resources. This tool allows you to define references to local resources (other than enterprise beans) on the machine where the application client resides.

References to enterprise beans, resource manager connection factories, and environment entries are made available in the JNDI naming context `java:comp/env`. References are relative to this context. The references can then be bound to the actual JNDI name in the target operational environment.

To specify binding information for an enterprise bean:

1. In the navigation pane, click the icon representing the enterprise bean instance. The properties for the enterprise bean are automatically displayed in the property pane.
2. In the property pane, click the **Bindings** tab.
3. Enter the JNDI name. This is the name to which the bean's home interface is bound in the global JNDI namespace. As a convention, use the fully qualified JNDI name. For example, `com/ibm/ejb/samples/ClaimHome`. You use this name when you bind EJB references to the bean from the referencing client.
4. Click **Apply**.
5. Save the archive.

To declare a reference and optionally bind the reference:

1. In the navigation pane, locate the property icon representing the reference (EJB reference, resource reference, or environment entry).
2. Right-click the icon and click **New**. The property dialog box is displayed. (If property values already exist for the item, the property dialog box is automatically displayed in the property pane.)
3. On the **General** tab, enter required properties. For example, for EJB references, enter a name, home interface, remote interface, and enterprise bean type. If you enter the name `ejb/ClaimHome`, the client looks up the bean's home interface by using the JNDI name `java:comp/env/ejb/ClaimHome`. View the help for [6.6.43.0.1: Assembly properties for EJB references](#), [6.6.43.0.2 Assembly properties for resource references](#), and [6.6.34.0.a: Assembly properties for environment entries](#).

If a referenced enterprise bean resides in the current module, or in the encompassing application, use the Link property to select the name of the bean. The binding information can be omitted.

4. Click the **Bindings** tab.

5. Enter the global JNDI name of the item being referenced. As a convention, use the fully qualified JNDI name. For the example reference `ejb/ClaimHome`, enter `com/ibm/ejb/samples/ClaimHome`. This name must match the JNDI name specified for the ClaimHome bean on its **Bindings** property dialog box.
6. Click **Apply**.
7. Save the archive.

Some additional notes on binding:

- For CMP entity beans, you can optionally specify the JNDI name of the datasource of the bean.
- Optionally, you can specify a default data source and default authorization information to be used by all entity beans in an EJB module. To do this, click the EJB Modules folder to display its property dialog box. Select the **Bindings** tab. Enter the appropriate values.

## 6.6.0.5.6: Converting EJB JAR files from 1.0 to 1.1 format

The Application Assembly Tool can convert an EJB 1.0 (serialized) deployment descriptor to an EJB 1.1 XML deployment descriptor. However, some code changes are required to make your EJB 1.0 application comply with the J2EE specification.

To convert a JAR file containing enterprise beans that conform to the 1.0 EJB specification, do the following:

1. Click **File->Open** to read the JAR file into the Application Assembly Tool. You are prompted whether to make the conversion.
2. Save the JAR file by clicking **File->Save As**. Specify a new name for the JAR file. An EJB module containing the converted JAR file is automatically created.

The resulting JAR file contains the following .xml and .xmi files instead of the original .ser files. Old information is mapped to the XML format of the EJB 1.1 specification.

- META-INF/MANIFEST.MF
- META-INF/ejb-jar.xml
- META-INF/ibm-ejb-jar-ext.xmi
- META-INF/ibm-ejb-jar-bnd.xmi



## 6.6.0.5a: Starting and stopping Application Assembly Tool

To start the Application Assembly Tool, run the assembly.bat file (on Windows NT systems) or assembly.sh on UNIX systems. The script is located in the directory where WebSphere Application Server is installed. For example, issue the following command on Windows NT systems:

```
C:\websphere\appserver\bin>assembly
```

If the tool cannot be started, verify that the assembly.bat file is in your PATH environment variable or specify the fully qualified path to the script.

To close the Application Assembly Tool, click **File->Exit**.

## 6.6.0.6: Installing applications with the application installer command line

The application installer, otherwise known as "SEAppInstall," is a tool for installing an application into a server configuration file and preparing this application to run within an application server.

- [Invoking the SE application installer](#)
- [Understanding SE application installer Syntax and parameters](#)
- [Installing an application into a server](#)
- [Uninstalling an application from a server](#)
- [Exporting an installed application containing configuration information](#)
- [Validating configuration information](#)

The application installer accomplishes the following:

- Locates and resolves bindings for each module in the EAR file
- Updates the [server configuration file](#) to reflect that the application is installed
- Puts the .ear file and its contents into the required arrangement in the directories under WebSphere Application Server
- Validates the configuration information in an EAR file and optionally, in the WebSphere Application Server

### Invoking the SE application installer

To invoke the SE application installer:

1. Open a command window.
2. Change directory to [product\\_installation\\_root/bin](#)
3. Run the tool by entering a command using the syntax and parameters described next.
4. Optionally, view the current command line parameters for the tool by typing:  
**SEAppInstall**

### Understanding SE application installer syntax and parameters

At the time of this writing, the tool displays the following information when invoked without parameters:

```
IBM WebSphere Application Server Release 4, AESJ2EE Application Installation Tool, Version
1.0 Copyright IBM Corp., 1997-2001 Required Operation Missing: You must specify either -install,
-uninstall, -export, -list, -extractDDL, or -validateUsage: Use one of
the following commands java com.ibm.websphere.install.se.SEApplicationInstaller -install <ear
file or directory> [-configFile <server configuration file>] [-expandDir <directory in which
to expand ear>] [-nodeName <name of node>] [-serverName <name of server>] [-ejbdeploy
<TRUE | false>] [-dbType <database type number>] [-schemaName <table schema name>]
[-precompileJsp <TRUE | false>] [-validate <app | server | both | NONE>] [-denyAll <TRUE |
false>] [-interactive {TRUE | false}] If you selected "-interactive false", you will not
be asked any questions and you will not be able to specify binding data java
com.ibm.websphere.install.se.SEApplicationInstaller -uninstall <application name> [-delete
<true | false>] [-configFile <server configuration file>] [-nodeName <name of node>]
[-serverName <name of server>] java com.ibm.websphere.install.se.SEApplicationInstaller -export
<application name> [-configFile <server configuration file>] -outputFile <name of the ear
file to create> java com.ibm.websphere.install.se.SEApplicationInstaller -list <apps | wars |
ejbjars | all> [-configFile <server configuration file>] java
com.ibm.websphere.install.se.SEApplicationInstaller -extractDDL <application name>
[-DDLPrefix <Prefix to apply to front of all DDL file names>] [-configFile <server configuration
file>] java com.ibm.websphere.install.se.SEApplicationInstaller -validate <app | server | both |
NONE> [-ear <ear file>] [-configFile <server configuration file>] If you specify
"-validate app" or "-validate both", you must include the "-ear" option. If you specify
"-validate server" or "-validate both", the "-configFile" option is optional. You can also
use the following syntax to execute the command: "java -jar SEAppInstall.jar" instead of
"java com.ibm.websphere.install.se.SEApplicationInstaller"
```

### Installing an application into a server

The **-install** option takes an EAR file, or expanded directory containing the EAR file contents and configures this file within a server configuration file specified via the optional **-configFile** parameter. If the **-configFile** parameter is not specified, the default is [product\\_installation\\_root/config/server-cfg.xml](#).

The optional parameters for this file include:

- **-dbType**: Applicable only during a deploy of a CMP EJB. This parameter forces the database type to use when generating the ddl.
- **-denyAll**: If not specified, results in denied access to all unprotected methods, if security is enabled.
- **-ejbdeploy**: If not specified, assumes that if the enterprise application EAR file contains one or more EJBs, these EJBs should be deployed during the installation.
- **-expandDir**: If not specified, defaults to what is specified under the APP\_INSTALL\_ROOT PathMap entry within the specified server configuration

file. If the `-configFile` is not specified, defaults to `product_installation_root/installedApps`

- **-interactive**: If not specified, defaults to true. This option walks through the installing application and looks for binding information that needs specification, including JNDI names for EJBs, EJB-REFs, and RESOURCE-REFs. Binding information to map roles to users and groups for security purposes is also included.
- **-nodeName**: If not specified, uses the node specified within the specified server configuration file. If no nodes, or if multiple nodes are found, an error is thrown.
- **-precompileJsp**: If not specified, defaults to true.
- **-schemaName**: Applicable only during a deploy of a CMP EJB. If specified, sets the database schema name to use with this application.
- **-serverName**: If not specified, uses the server specified within the specified server configuration file. If no servers, or if multiple servers are found, an error is thrown.
- **-validate**: If specified, Validates the configuration bindings and extensions in the server and the specified application. Possible values of `-validate` are:
  - app: Requires the `-ear` parameter. Specify the fully qualified path to the EAR file.
  - server
  - both: Requires the `-ear` parameter. Specify the fully qualified path to the EAR file.
  - NONE

The default is no validation.

The process of installing an application updates the server configuration file with the application, and installs the EJB JARs within the EJB container and the WARs within the Web Container. The installation process also expands the application under the `expandDir` location. The expanded directory is named the same as the installed application, but in directory format. There is also binding information under the appropriate META-INF and WEB-INF directories, within the appropriate modules of the application. For example (on a single line):

```
seappinstall -install c:\exportedApps\sampleApp.ear -configFile c:\websphere\appserver\config\server-cfg.xml -interactive false
```

This command installs the sampleApp.ear file into the appropriate server configuration file.

## Uninstalling an application from a server

The **-uninstall** option takes an EAR file and removes it within a server config file, specified via the optional **-configFile** parameter. Note that the location of the installed application in the expanded directory is not removed and contains two optional parameters, unless you specify the `-delete` true option:

- **-delete**: If a value of true is specified, the application is removed from the directory in which it was expanded during installation.
- **-nodeName**: If not specified, uses the node identified within the specified server configuration file. If no nodes, or if multiple nodes are found, an error is thrown.
- **-serverName**: If not specified, uses the server identified within the specified server configuration file. If no server, or if multiple servers are found, an error is thrown.

The process of uninstalling an application updates the server configuration file removing the application. This process removes the module references on the EJB and Web Containers. For example (on a single line):

```
seappinstall -uninstall sampleApp -configFile c:\websphere\appserver\config\server-cfg.xml
```

uninstalls the sampleApp application from the appropriate server configuration file.

## Exporting an installed application containing configuration information

The parameters for `seappinstall` include:

- **-export**: Takes an installed application within a server config file specified via the optional **-configFile** parameter and exports it to the location specified by the **-outputFile** parameter. This option is useful since the exported jar contains the binding information that the application uses within the server configuration. For example (on a single line):

```
seappinstall -export sampleApp -configFile c:\websphere\appserver\config\server-cfg.xml -outputFile c:\exportedApps\sampleApp.ear
```

exports the installed sampleApp application and collapses the application files into the EAR file specified by the `outputFile` parameter.

- **-extractDDL**: Extracts all table definitions for any CMP EJBs in the specified installed application. Optional parameters are **-configFile** and **-DDLPrefix**. `-DDLPrefix` specifies an optional prefix to prepend to all DDL file names. The output files are written to the current directory. For example:

```
seappinstall -extractDDL sampleApp
```

generates the DDL file sampleApp.ear\_Increment.jar\_Table.ddl in the bin directory.

- **-list**: Lists the installed entities as specified by the `-list` parameter. Possible values for `-list` are:
  - apps
  - wars
  - ejbjars
  - all

The optional parameter, `-configFile`, specifies the server configuration file against which the list request is issued. For example:

```
seappinstall -list all
```

lists all the applications, all the Web modules, and all the EJB modules installed in the specified server configuration.

## Validating configuration information

The -validation option validates the bindings and extensions in an enterprise application (EAR) and, optionally, the configuration in the application server configuration file.

It is not required that this application already be installed, or in its expanded form. The -configFile parameter is optional. If not specified, the [\*product\\_installation\\_root\*](#)/config/server-cfg.xml server configuration file is used.

For example:

```
seappinstall -validate both -ear c:\websphere\appserver\installableApps\sampleApp.ear
```

## 6.6.0.7: Launching Java application clients in the J2EE application client container

The J2EE specification requires support for a client container that runs standalone Java applications (known as J2EE application clients) and provides J2EE services to the applications. J2EE services include naming, security, and resource connections.

### Before using the tool

You are ready to run your client application using this tool after you have:

1. [Written the client application program](#)
2. [Assembled](#) and [installed](#) an application module (.ear file) in the application server runtime

### Invoking the launchClient tool

J2EE application clients are launched using the launchClient shell, located in:

[product\\_installation\\_root](#)/bin/launchClient.bat

This command instantiates an instance of the [launchClient](#) class.

### Application client runtime behavior

The launchClient batch command starts the application client runtime, which:

- Initializes the client runtime
- Loads the class that you designated as the main class with the [application assembly tool](#)
- Runs the main method of the client application program

When your program terminates, the application client runtime cleans up the environment and the Java virtual machine ends.

### Passing parameters

You have to pass parameters to the launchClient command. You can pass parameters to your client application program as well. The launchClient command allows you to do both.

The launchClient command requires that the first parameter is either:

- an Ear file specifying the client application to launch
- a request for launchClient usage information

All other parameters intended for the launchClient command must begin with the -CC prefix.

Parameters that are not Ear files, or usage requests, or that begin with the -CC prefix, are ignored by the application client runtime, and are passed directly to the client application program.

For more information on the launchClient parameters, refer to the main method in the [launchClient](#) javadoc.

launchClient retrieves parameters from three places:

1. The command line
2. A properties file
3. System properties

The parameters are resolved in the order listed above, with command line values having the highest priority and system properties the lowest. This prioritization allows you to set and override default values.

### Specifying the server name

By default, the launchClient command uses the environment variable COMPUTERNAME for the **BootstrapHost** property value. This setting is effective for testing your client application when it is installed on the same computer as the server. However, in other cases you need to override this value with the name of your server.

You can override the **BootstrapHost** value by invoking launchClient with the following parameters:

**launchClient myapp.ear -CCBootstrapHost=abc.midwest.mycompany.com**

You can also override the default by specifying the value in a properties file and passing the file name to launchClient.

### Running a client application with security enabled

Security is controlled by the server. You do not need to configure security on the client because the client assumes that security is enabled. If security is not enabled, the server ignores the security request, and the client application works as expected.

## Format of the launchClient properties file

You can store launchClient values in a properties file, a good method for distributing default values. A knowledgeable user can then override one or more values on the command line.

The format of the file is one launchClient-CC parameter per line without the -CC prefix. For example:

```
verbose=trueclasspath=c:\mydir\util.jar;c:\mydir\harness.jar;c:\production\G19\global.jarBootstrapHost=abc.westcoast.mycompany.comtracefile=c:\WebSphere\mylog.txt
```

## Command syntax reference

You can obtain additional help on this command by invoking it without arguments. Following is the result of one such invocation. The arguments are subject to change. For the definitive arguments and syntax, query the program.

```
IBM WebSphere Application Server, Release 4.0J2EE Application Client Tool, Version 1.0Copyright IBM Corp., 1997-2001WSCL0012I:
Processing command line arguments.Usage: launchClient [<userapp.ear> | -help | -?] [-CC<name>=<value>] [app
args]where:<userapp.ear>      = The path and name of the Ear file containing the client application.-help, -?      =
Print this help message.where the -CC properties are for use by the Application Client Runtime:-CCverbose      =
<true|false> Use this option to display additional      informational messages. The default is false.
-CCclasspath      = A classpath value. When an application is launched,      the system classpath is
not used. If you need to      access classes that are not in the ear file or part      of
the resource classpaths, specify the appropriate      classpath here. Multiple paths may be concatenated.
-CCjar      = The name of the client jar file within the EAR file      that contains the
application you wish to launch.      This argument is only necessary when you have
multiple client jar files in the ear file.      -CCBootstrapHost      = The name of the host server you
wish to connect to      initially. Format: your.server.ofchoice.com-CCBootstrapPort      = The server port
number. If not specified, the      WebSphere default value is used.-CCinitonly      = <true|false>
This option is intended for ActiveX      applications to initialize the Application Client
runtime without launching the client application.      The default is false.
-CCtrace      = <true|false> Use this option to have WebSphere write      debug trace information
to a file. You may need this      information when reporting a problem to IBM Service.
The default is false.      -CCtracefile      = The name of the file to write trace information.
The default is to output to the console.      -CCpropfile      = Name of a Properties file containing
launchClient      properties. In the file, specify the properties      without the -CC
prefix. For example: verbose=true.      where "app args" are used by the client application and are ignored
byWebSphere.
```

## 6.6.0.8: Expanding .ear files

The EARExpander expands Ear files into the format desired by the application server runtime, as described in [the application installation instructions](#). EARExpander can also collapse the expanded format back to a normal Ear (.jar or .zip) format.

### Invoking the tool

The tool is located in the following directory:

[product\\_installation\\_root](#)/bin/EARExpander.bat

To view syntax, open a command line and invoke the tool without arguments. Here is a typical result. The line breaks have been changed for better formatting in this documentation.

```
C:\seaa0122.02\bin>EARExpander.bat IBM WebSphere Application Server, Release 4.0 J2EE J2EE
Application Expansion Tool, Version 1.0 Copyright IBM Corp., 1997-2001 Required Argument Missing: ear
Usage: java com.ibm.websphere.install.commands.EARExpander-ear -expandDir -operation
[expansionFlags]
```

ExpansionFlags indicate whether you want every JAR file expanded, or just the contained WAR files within the EAR file. The default is all.

### Expanding files

The following example command expands the file my.ear into the [product\\_installation\\_root](#)/bin/myEAR directory:

```
EARExpander -ear my.ear -expandDir product\_installation\_root/bin/myEAR -operation expand
```

### Collapsing files

Using the collapse -operation reverses the format to normal.

```
EARExpander -ear my.ear -expandDir product\_installation\_root/bin/myEAR
-operation collapse
```

Type each of the above commands on a single line, despite their appearance in this documentation.

## 6.6.0.9: Application Client Resource Configuration Tool for configuring client resources

The *Application Client Resource Configuration Tool* defines the resources for the client application. These configurations are stored in the client application .ear file, and are used by the WebSphere application client runtime for resolving and creating an instance of the resources for the client application.

[Launching the tool](#)




## 6.6.0.9.3: Removing objects from EAR files with the Application Client Resource Configuration Tool

During this task, you will remove (delete) an object from an EARfile for your application client. You can remove any particular J2EE resource or resource provider, including data sources and data source providers; URLs and URL providers;JMS providers, connection factories, and destinations; and JavaMail sessions. You cannot removethe default JavaMail provider, however.

1. [Start the tool and open the EAR file](#) from which you want to remove an object. The EAR file contents will be displayed in a tree view.

*Recommended.* If you already had the EAR file open, and have made some changes, click **File -> Save** to save your work before preceding to delete an object. See below for a discussion.

2. In the tree, locate the object that you want to remove.
3. Do one of the following:
  - Click the object, then click **Edit -> Delete** from the tool menu bar.
  - Right-click the object to display its menu, then click **Delete**.
4. If you are sure that you want the deletion to take effect, click **File -> Save**.

 The option to delete an item does not offer a confirmation dialog. As soon as you delete the item, it is gone. As a safeguard, consider saving your work right before you begin this task. That way, if you change your mind after removing an item, you can close the EAR file without saving your changes, which will cancel your deletion. Be sure to do so immediately after the deletion, or you will also lose any unsaved work that you performed since the deletion.

## 6.6.0.9a: Starting the Application Client Resource Configuration Tool and opening an EAR file

Start the graphical interface with the command:

```
clientConfig
```

To open an EAR file into the tool:

1. On the menu bar of the tool, click **File -> Open**.
2. Browse for the file that you want to open.
3. When you have found the file and selected it, click **Open**.

To save your changes to the file and close the tool:

1. On the menu bar of the tool, click **File -> Save**.
2. Click **File -> Exit**.

Now begin administering a resource type:


- [JDBC providers and data sources](#)
- [JavaMail providers and sessions](#)
- [URL providers and URLs](#)
- [JMS providers, connection factories, and destinations](#)

## 6.6.0.10: SoapEarEnabler tool

The SoapEarEnabler tool is a Java application that enables a set of SOAP services within an Enterprise Application Archive (EAR).

**Note:** Start with an existing EAR file, either one created with the Application Assembly Tool (AAT), or a previously-created, valid, J2EE-compliant EAR.

The SoapEarEnabler guides you through the required steps to enable one or more services within an application. The SoapEarEnabler tool makes a backup copy of your original EAR in the event you need to remove or add services at a later time.

 SoapEarEnabler will **not** accept a "soap-enabled" EAR file as input.

After you enable web services, you must install the EAR in WebSphere Application Server.

You will be prompted as to whether you wish to add the administration client to the EAR. This is a Web-based client that will allow to list all active services for a specific context from a browser window. With this interface, you can stop and start existing services. You might choose to not add this interface for security reasons, or you might want to secure the interface before making a service available.

For more information on securing resources, see article [Securing SOAP services](#).

Before invoking SoapEarEnabler, first create an [4.8.2.1: Apache SOAP deployment descriptor](#) for each service to be enabled.

Invoke the SOAP EAR enabler tool from the WebSphere Application Server bin directory using the command, **SoapEarEnabler** on the Windows platform or **SoapEarEnabler.sh** on UNIX platforms.


The tool operates in two modes:

1. [interactive](#)
2. [silent](#)

Specify all required command line arguments to use the tool in *silent* mode.

### Interactive mode

The SoapEarEnabler tool prompts you for all required information. The following dialog is an example of using the tool in interactive mode.

 In this dialog, user input is in *italics*, and tool output is in **bold**.

*SoapEarEnabler* (On Windows NT)

*SoapEarEnabler.sh* (On UNIX platforms)

**Please enter the name of your ear file:**

*..\work\stockquote.ear*

**How many services would you like your application to contain (1...n)?**

*1*

**Now prompting for info for service #1:**

**Please enter the file name of the deployment descriptor xml file:**

*..\work\StockQuoteDD.xml*

**Is this service an EJB (y/n)?**

*n*

**How many jar files are required for this service (0...n)?**

*1*

**Classpath requirement #1: Please choose a file ([1] samples.jar, [2] stockquote.war):**

*1*

**Should this service be secured (y/n)?**

*n*

**Please enter a context root for your non-secured services (e.g. /soap):**

*/soapsamples*

**Do you wish to install the administration client?**

**Warning! You should not install this client in a production ear unless you intend to secure the URI to it.**

**Install the administration client (y= yes/n= no)?**

*y*

### Silent Mode

In silent mode, supply the arguments in the same order as for the interactive prompts.

 In silent mode, the SOAP Admin GUI will not install. Also, you will not be prompted for the SOAP Admin GUI.

The following example describes how to use the tool in silent mode:

**soapearenabler [args]**

where the arguments must be specified in the following order:

**<ear-file-name>**

**<number-of-services>**

*The following block is repeated based on the number of services specified.*

**<deployment-descriptor-file-name>**

**<service-is-an-ejb-(y/n)>**

*The following argument should be supplied only if service-is-an-ejb-(y/n) is y.*

**<ejb-jar-file-uri-(already-in-ear)>**

**<number-of-additional-jar-files(0, 1, 2...)>**

*The following argument is repeated number-of-additional-jar-file times.*

**<classpath-entry-uri-(already-in-ear)>**

**<secure-this-service-(y/n)>**

*This following argument is supplied only if secure-this-service-(y/n) is n for any service.*

**<context-root-for-non-secured-services, ex: /soap>**

*This following argument is supplied only if secure-this-service-(y/n) is y for any service.*

**<context-root-for-secured-services, ex: /soapsec>**

## Silent Mode examples

The following is an example of deploying one ejb as a non-secured service:

```
soapearenabler soap.ear 1 d:\xml-soap\java\samples\ejbadder\deploymentdescriptor.xml
addservice-ejb.jar 1 samples.jar n /soap
```

The following is an example of deploying one ejb as a non-secured service, and one java class as a secured service:

```
soapearenabler soap.ear 2 d:\xml-soap\java\samples\ejbadder\deploymentdescriptor.xml
addservice-ejb.jar 1 samples.jar n
d:\xml-soap\java\samples\stockquote\deploymentdescriptor.xml n 1 samples.jar y /soap /soap-sec
```

The following is an example of deploying 2 java classes as non-secured services:

```
soapearenabler soap.ear 2 d:\xml-soap\java\samples\stockquote\deploymentdescriptor.xml n 1
samples.jar n d:\xml-soap\java\samples\addressbook\deploymentdescriptor.xml n 1 samples.jar
n /soap
```

The line breaks in the above examples have been modified for this documentation. Typically, commands are issued on a single line.

## 6.6.0.13: Log Analyzer main window


The Log Analyzer takes one or more activity or trace logs, merges all the data, and, by default, displays the entries in unit of work groupings. It analyzes error conditions in the log entries to provide error message explanations. The Log Analyzer's main window has the following interface:

- Three window panes
- Status line
- Menu bar
- Pop-up actions

### Window panes


The Log Analyzer window has three panes:

- **Logs pane (left)**

You may find the unit of work UOW grouping useful when you are trying to find related entries in the activity log or when you are debugging problems across multiple machines. By default, Log Analyzer's Logs pane displays log entries by unit of work. It lists all the unit of work (UOW) instances and its associated entries from the logs that you have opened. The file name of the first log that you opened is shown in the pane's title bar. By default, the log entries are shown in UOW sequence. There is a root folder and under it, each UOW has a folder  icon which you can expand to show all the entries for that UOW. All log entries *without* any UOW identification are grouped into a single folder in this tree view. The UOW folders are sorted to show the UOW with the latest timestamp at the top of the list. The entries *within* each UOW are listed in the reverse sequence, that is the first (earliest) entry for that UOW is displayed at the top of the list. If you have merged several logs in the Log Analyzer, all the log entries are merged in timestamp sequence within each UOW folder, as if they all came from the same log.

Each UOW line has the following format:

12/8/98 15:02:45.613064597	(1)	32693:mfok01.torolab.ibm.com
└──────────┴──────────┴──────────┘		
Last timestamp	No. of entries	UnitOfWork

Click the  icon next to the UOW folder to see all the log entries for the UOW. Each log entry's identification has the following format:

For the WebSphere Enterprise Edition:

Rec_5_	SOMSGClientStarter	@SOMSGClientStarter
└──┘	└──────────┘	└──────────┘
Entry#	Function Name	Class Name

For WebSphere Advanced Edition:

Rec\_5\_com.im.ejs.sm.server.AdminServer

Entry #

Class Name

Every log entry is assigned an entry number, Rec\_####, when a log is opened in the Log Analyzer. If more than one file is opened in the Log Analyzer (merged files), the Rec\_#### identification will not be unique because the number is relative to the entry sequence in the original log file and not to the merged data that the Log Analyzer is displaying. This Rec\_#### appears in the first line (**RecordedId**) in the Records pane.





By default, each entry in this pane is color-coded to help you quickly identify the ones that have high severity errors.

- Non-selected log entry with background color of:
  - Pink indicates that it has a severity 1 error.
  - Yellow indicates that it has a severity 2 error.
  - White indicates that it has a severity 3 error.
- Selected log entry with background color of:
  - Red indicates that it has a severity 1 error.
  - Green indicates that it has a severity 2 error.
  - Blue indicates that it has a severity 3 error.

These colors are configurable and can be changed in the Log Analyzer's Preferences Log page. See [Background color](#) for different error severity levels and for more information on how to do this.

The Log Analyzer can display the log entries in different groupings. Use the [Log Analyzer Preferences notebook: Logs](#) page to set the grouping filters.

After the Analyze action has been invoked, each analyzed log entry has the following icons:

-  indicates that the entry has some analysis information in one or more pages in the Analysis pane.
-  indicates that the entry has some analysis information and that it has a reraised or remapped exception. You may want to look at the log entry prior to this one when diagnosing problems.
-  indicates that the entry has either a severity 1 or 2 error but no additional analysis information is available for it.
-  indicates that the entry has a severity 3 error and it has no analysis information.

#### ● Record pane (upper right)

When you select an entry under the unit of work in the Logs pane, you see the entry in the Record pane. The entry's identification is shown in the pane's title bar. Right-click in this Record pane to see the actions that you can perform on the entry. There is a drop down arrow next to **Record** which allows you to go back to look at the last ten records that you have viewed. The cache for the historical data (10, by default) is set in the Preferences General page.

#### Note:

1. The associated analysis data for these cached records are not shown. To see analysis information for cached data, reselect the entry from the Logs pane.

2. The timestamp format that appears in the Record pane must match the format that was used by the showlog command when formatting the log.

You can enable/disable line wrap mode for the Record Pane using the **Log Analyzer Preferences notebook: Record**. To print contents of this pane, select **Record > Print** when the Record pane is in focus.

- **Analysis pane (lower right)**

When analyze action has been invoked *and* additional information is available, the information will appear in the Symptom page. The page, GPF, is filled in only when the analyze action is invoked *and* there is a GPF dump. If the page tab is grayed out, there is no information in that page. The pages of the Analysis pane are:

- **Symptom**

The Log Analyzer provides a database of information to help you recover from some common WebSphere errors. As a part of the analyze action, if such recovery information is found in the database for the selected log entry, the information is displayed in this page.

You can set line wrap for information that appears in the Symptom page in the Analysis pane using the **Log Analyzer Preferences notebook -- Analyzer output**.

The Symptoms page can be detached into separate windows to make it easy for you to view all the information by selecting **View > Detach into a new window**.

You can set line wrap for information that appears in the Symptoms page in the Analysis pane using the **Log Analyzer Preferences notebook -- Analyzer output**.

## **Status line**

There is a status line at the bottom of the window showing the status of actions.

## **Menu bar**

The menu bar in the Log Analyzer's main window, has the following selections:

- **File**

- **Open...**

Opens a new log file in the analyzer. You can select either a formatted activity/trace log or a previously saved XML file. If you want the Log Analyzer to format a raw log file (by running the showlog command) prior to opening it, name the log file with suffix.log. If the Log Analyzer finds that the .log file contains formatted data, it skips the showlog formatting step.

If you want to merge data from another log, select **Merge with**.

- **Merge with...**

When another log file is already opened in the Log Analyzer, use the **Merge with** action to open subsequent logs. The Log Analyzer merges the data from all the logs that it opens and displays all the entries within timestamp sequence in the UOW folders. The data appears as if they came from one log.

If you want the Log Analyzer to run the showlog utility prior to opening it, name the log file **activity.log**.

- **Redisplay logs**

To redisplay the logs using the recently set filters.

- **Save as...**

Saves the log as an XML file (or text file). If analyze action has been performed, all the Symptom analysis information is also saved. If logs are merged in the Log Analyzer, the saved file contains entries of all the merged logs in the sequence that is shown in the Logs pane.

**Note:** If the merged logs have different timestamp formats, you should not save the merged information because the Log Analyzer only recognizes a single timestamp format for each file that it opens.

- **Save**

Is only enabled if the first file that you opened is an XML file. It resaves the XML file with all the data that is currently displayed in the Log Analyzer. Any Symptom information is also saved. If logs are merged in the Log Analyzer, the saved file contains entries of all the merged logs in the sequence that is shown in the Logs pane. **Note:** If the merged logs have different timestamp formats, you should not save the merged information because the Log Analyzer only recognizes a single timestamp format for each file that it opens.

- **Print Log...**

Prints all the entries that the Log Analyzer is displaying. If logs are merged in the Log Analyzer, the output contains entries of all the merged logs in the sequence that is shown in the Logs pane. If analyze action has been performed, all Symptom analysis information is also printed. To print parts of the log, use **Record > Print**.

- **Close**

Closes the opened log.

- **Update Database**

Updates the symptom database which is used for Symptom analysis. It downloads the latest version of the symptom database from the URL specified in the ivblogbr.properties file.

- **Preferences...**

Lets you configure and change the appearance of the Log Analyzer window and its contents.

- **Exit**

Exits the Log Analyzer and closes its window.

- **Edit**

- **Copy**

Copies the selected text in the Record or Analysis pane to the clipboard. If you have not selected any text, **Copy** does not appear in the menu.

- **Find**

Allows you to find text strings in the focused pane.

- **View**

- **Logs**

Toggles the visibility of the Logs pane.

- **Record**

Toggles the visibility of the Record pane.

- **Symptom**

Toggles the visibility of the Symptom page in the Analysis pane.

- **Record**



All the actions under this menu applies to the focused pane.

To select several entries, hold down the **Ctrl** key when making the selection. When a folder is selected, the action applies to all the entries in that folder.

- **Analyze**

Retrieves and displays additional documentation on known errors and error messages in the Analysis pane (Symptom page). Select the folder(s) and/or entries in the Logs pane, right-click to select the **Analyze** action, or from the menu bar, select **Record > Analyze**. **Note:** If you invoke **Analyze** for the root folder, then all the entries in the log that you are viewing will be analyzed.



If some analysis information is available for an entry, it will either have a  or  icon next to it in the Logs pane. If the analyze action has already been performed, the selection will be grayed out.

- **Save to file**

Saves the selected entries in the Logs pane. If UOW folders are selected, all the entries in the folder are saved. Any retrieved analysis information is also saved. If the focused pane is either the Records or Analysis pane, then only that pane's information is saved.

- **Print**

- If the focused pane is Logs, the action prints the selected folder(s) and/or entries. Any retrieved analysis information for those entries is also printed.
- If the focused pane is Record, the action prints the entry that is currently in the Record pane. Any retrieved analysis information is not printed.
- If the focused pane is Analysis, the action prints the Symptom page contents.

- **Windows**

If you have detached the Symptom page in the Analysis pane into separate windows, all the windows will be listed under this menu and you can select the windows that you want to bring to the foreground.

- **Help**

Provides a list of WebSphere's on-line documentation for additional information.

### **Pop-up actions**

In the focused pane, right-click to bring up a list of actions in a pop-up menu. Actions that you cannot perform are grayed out. When a folder is selected in Logs pane, the action applies to all the entries in that folder. To select several folders or entries in the Logs pane, hold down the **Ctrl** key when making the selection.

## 6.6.0.13.1: Log Analyzer Find window

The Log Analyzer Find window allows you to look for text strings in the focused pane. For example, if you remember the Unit of Work identification, you can enter that text string in the Find window to quickly locate the Unit of Work folder in the Logs pane.

## 6.6.0.13.2: Log Analyzer Preferences notebook -- General

The Log Analyzer Preferences notebook's General page lets you specify the behavior of panes in the Log Analyzer's window:

- **Show title bars**  
Shows title bars of window and its panes.
- **Highlight selected pane**  
Highlights the pane that is in focus.
- **Pane history cache size**  
Specify a number of records to save in the cache. The Log Analyzer keeps a history of the (specified number of) records that you have viewed. You can use the drop down list next to **Record** in the pane's title bar to see these cached entries.

**Note:** The associated analysis data for these records are not saved. To see analysis information, reselect the entry from the Logs pane.

- **Show logo at startup**  
Shows the logo when you start-up the Log Analyzer.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

## 6.6.0.13.3: Log Analyzer Preferences notebook -- Appearance

The Log Analyzer Preferences notebook's Appearance page lets you define the overall appearance of the Log Analyzer. You can select the family of products and its textureschemes that you want the Log Analyzer's window to emulate.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

## 6.6.0.13.4: Log Analyzer Preferences notebook -- Toolbars

The Log Analyzer Preferences notebook's Toolbars page lets you customize the appearance and contents of the toolbar in the Log Analyzer window. You can select whether there is text and/or icon in the toolbar, as well as, the functions that you want in the toolbar.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

## 6.6.0.13.5: Log Analyzer Preferences notebook -- Help

The Log Analyzer Preferences notebook's Help page lets you select the browser that will be used to display online help files.

For Windows NT, the default web browser will be used and you do not need to update any settings unless there are problems when bringing up the default browser.

For AIX, HP-UX, and Solaris, you have to update the following settings, especially the browser's full path in the **Browser location** entry.

- **Help browser**  
Select the web browser you want to use.
- **Browser location**  
Select the location of the browser executable file. This should be correct by default, but if you cannot access help then you may need to explicitly enter the browser location.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

## 6.6.0.13.6: Log Analyzer Preferences notebook -- Logs

The Logs page of the Log Analyzer Preferences notebook lets you group the entries in the logs by different entry fields for viewing. For example, you can select to group the log entries by TimeStamp or clientHostName when they are displayed in the Logs pane.

### Filter 1

Use this filter to set the first level of grouping when log entries are displayed in the Logs pane. By default, the log entries are grouped by UnitOfWork.

### Filter 2

Use this filter to set the second level of grouping (that is, within the grouping of filter 1 folder) when log entries are displayed in the Logs pane.

All the entries within the grouped folders are always sorted in timestamp sequence with the earliest entry at the top of the list.

### Redisplay log file immediately

Select this box to immediately regroup the logs entries (after you have clicked **OK**) based on the new filter settings. The entries in the Logs Pane are redisplayed according to the new grouping. If you want to delay the grouping, then do not select this box and, at a later time, you can use the **File > Redisplay logs...** menu selection to regroup and display the log entries based on the changed filter settings.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

## 6.6.0.13.7: Log Analyzer Preferences notebook -- Severity

The Log Analyzer Preferences notebook lets you change the background colors of log entries that appear in the Logs pane. The colors are useful to help you quickly identify entries that have high severity errors and the entry that you have currently selected.

### Use colors to indicate level of error severity

Select this checkbox to color-code the background of log entries and folders. When selected, the radio button selections in this page are enabled.

### Background color

For each folder and entry in the Logs pane, there is some text describing the entry.

To choose a background color for selected log entry that has a severity 1 error, do the following:>

1. Select **Selected node**.
2. Select **Severity 1**.
3. Select the color by clicking on the color **Swatches**. To use the default setting, click **Restore Default**. To see the results of your change, look at the **Preview** box.
4. Click **Apply** to save that setting.

Repeat similar steps to change the background color for selected log entries that have severity 2 and 3 errors.

To choose a background color for an unselected log entry that has a severity 1 error, do the following:

1. Select **Unselected node**.
2. Select **Severity 1**.
3. Select the color by clicking on the color **Swatches**. If you want to use the default setting, click **Restore Default**. If you want to see the result of your change, look at the **Preview** box.
4. Click **Apply** to save that setting.

Repeat similar steps to change the background color for unselected log entries that have severity 2 and 3 errors.

### Sample

You can see the result of your color change prior to applying the change. Look at the nodes shown in the **Sample** box. For color changes of *selected* nodes, click on the node in the sample box to see the color change.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.



## 6.6.0.13.8: Log Analyzer Preferences notebook -- Record

The Log Analyzer Preferences notebook lets you set the line wrap mode for the data that is displayed in the Record pane. It also lets you set the time and date format of the timestamp so that it matches the timestamp format of the log that you are opening.

### **Enable line wrap mode for record pane**

Select this checkbox to enable line wrapping for the information that appears in the Record pane.

### **Date format**

When viewing logs, this date format only changes the timestamp format that is displayed in the Record pane. The timestamp format in the log file and the timestamp shown in the Logs pane are not affected by this setting.

### **Time format**

When viewing logs, this time format only changes the timestamp format that is displayed in the Record pane. The timestamp format in the log file and the timestamp shown in the Logs pane are not affected by this setting.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

## 6.6.0.13.9: Log Analyzer Preferences notebook -- Analyzer output

The Log Analyzer Preferences notebook lets you enable line wrap for information that appears in the Analysis pane.

### Set line wrap

Select the appropriate checkbox to enable line wrap for the following page that appears in the Analysis pane:

- Symptom page

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

## 6.6.0.14: XML-SOAP Admin tool

Use the [SOAPEarEnabler](#) tool to add administrative interfaces to your EAR files. You can then use the XML-SOAP Admin tool with these EAR files.

WebSphere Application Server provides a modified version of the Apache SOAP XML-Admin interface (or XML-SOAP Admin tool) for each SOAP-enabled EAR file. This interface allows you to do the following for each context root:

- List configured services, showing active and stopped services
- Stop a service
- Start a service
- View the Apache Soap deployment descriptor for a service

### Accessing the XML-SOAP Admin tool

Access the XML-SOAP Admin tool through a Web browser by specifying:

```
http://localhost/<contextroot>/admin/index.html
```

**Note:** The *context root*, in this example, is the context specified when installing the SOAP-enabled .ear file. The context root for SOAP samples is `soapsamples`.

Therefore to use this interface with the SOAP samples, enter:

```
http://localhost/soapsamples/admin/index.html
```

You cannot use the XML-SOAP Admin tool to add or remove a service. Use the SOAP Ear Enabler tool to add or remove services. A "stopped" service is persisted across starts and stops of the application server. Therefore, if you stop a service, it will remain stopped until the next time you use the XML-SOAP Admin tool to start it again.

You can add the XML-SOAP Admin tool interface to an enterprise application when you SOAP-enable the EAR file. In interactive mode, you are asked whether you want to add the XML-SOAP Admin tool interface. Replying "yes" will add the necessary JSP files and bindings that allow you to access the XML-SOAP Admin tool interface for the application. The interface is an optional addition because you may not want to expose it in a production environment. Optionally, you may choose to use the application assembly tool to assign roles to the XML-SOAP Admin tool so that it is secure.

### Updating an existing SOAP-enabled Enterprise Application

The Application Assembly tool is used to update the contents and configuration of an enterprise application. For example, to secure the XML-SOAP Admin tool interface for a particular EAR, use the application assembly tool to secure the resource. (See article [Securing applications](#) for security information.) However, you cannot use the application assembly tool to add or remove a Web service.

To add or remove a service to the XML-SOAP Admin tool, start with the original EAR file and execute the enabling process again.

**Note:** The SOAPEar Enabler tool saves a backup copy of the EAR file.

# 6.6.0.15: Deployment Tool

## Main topics

- [Generating deployment code for EJB enterprise beans](#)

## Other topics

- [Meet-in-the-middle mapping support](#)
- [Modifying the schema document](#)
- [Modifying the map document](#)
- [Implementing custom finders in home interfaces for CMP entity beans](#)

## Migrating CMP beans from Version 3.5

To preserve the existing (older) table/schema rules when migrating CMP entity beans from WebSphere Application Server Version 3.5 to Version 4.0, use the -35 option of the deployment tool. The deployment tool is used to generate code for deployment on Version 4.0. Its -35 option will generate table mappings based on the 3.5 rules. Combine the -35 option with the -dbschema EJB option to map to the 3.5 generated schema that was used.

To migrate manually, the EJB JAR requiring migration must be invoked through the deployment tool command line, using the arguments:

```
-35 -dbschema EJB
```

Without these options, the database table will follow a new set of rules that is more aligned with the defaults for VisualAge for Java.

## The .java and .class file time stamps must differ

If an error occurs when you are generating code for deployment, ensure that the time stamps of the .class files and corresponding .java files in your EJB JAR file do not match. If the .class and .java files have the same time stamp, one solution is to perform an operation on the .java files (such as "touch" on a UNIX system) to cause their time stamps to change without changing the file content.

# 6.6.0.15.1: Generating EJB deployment code from the command line

Before you can successfully run your enterprise beans on either a test or production server, you need to generate deployment code for the enterprise beans. The Deployment Tool for Enterprise JavaBeans (referred to in the remainder of this topic as simply the EJB Deploy Tool) provides a command-line interface that you can use to generate enterprise bean deployment code. The tool employs a command-line environment that allows you to run a build process overnight and have the deployment tool automatically invoked to generate your deployment code in batch mode.

The EJB Deploy Tool is invoked from the command line using the **ejbdeploy** command, which accepts an input EJB JAR file that contains one or more enterprise beans. It then generates an output deployed JAR file that contains deployment code in the form of .class files.

Specifically, when you run the **ejbdeploy** command, the following activities occur:

- Code is imported from the input JAR file
- Deployment code is generated
- RMIC is run
- The deployment code is compiled
- Code is exported to the output JAR file

These activities are discussed further in some of the descriptions for the options that can be specified with the **ejbdeploy** command.

The EJB Deploy Tool supports meet-in-the-middle mapping, EJB single and multiple table inheritance, and associations that have been defined in VisualAge for Java. It also supports the use of converters, which translate a database representation to a Java object type, and composers, which are used to map a single complex bean field to multiple database columns.

For information about migrating entity beans from WebSphere Application Server Version 3.5, see the section below entitled "Migrating CMP entity beans from WebSphere Application Server, Version 3.5". For information about generating deployment code for VisualAge for Java enterprise beans and running them in WebSphere Application Server, see the section below entitled "Deploying and running VisualAge for Java enterprise beans in WebSphere Application Server".

## Syntax

```
ejbdeploy input_JAR_name working_directory output_JAR_name [-codegen] [-cp classpath][-dbname name]  
[-dbschema name][-dbvendor name][-ignoreErrors] [-keep] [-noinform] [-novalidate] [-nowarn] [-quiet]  
[-rmic "options"] [-trace] [-35]
```

## Command arguments

*ejbdeploy*

The command to generate deployment code. If run without any arguments, the **ejbdeploy** command displays a list of arguments that can be run with the command.

*input\_JAR\_name*

The fully qualified name of the input JAR file that contains the enterprise beans for which you want to generate deployment code; for example, `c:\ejb\inputJARs\myEJBs.jar`. (This argument is required.) If your input JAR file contains CMP beans, the `ejbdeploy` command automatically creates the schemas and mappings for the CMP beans. It is important to note that the `ejbdeploy` command no longer uses what is specified on the system class path. Instead, the dependent classes need to be contained in a JAR file and included in the command processing using the **-cp** option. You must ensure that the .class files of each enterprise bean's home and remote classes are packaged in the input JAR file. (Note that the best way to specify dependent JAR files is through the [Java extension mechanism](#) and by listing the required JAR files in the manifest file of the input JAR file. The EJB Deploy Tool will respect these extensions and you won't need to specify them using a command-line option.)

#### *working\_directory*

The name of the directory where temporary files are stored that are required for code generation. (This argument is required.) If the working directory that you specify already exists prior to running the `ejbdeploy` command, the temporary files are generated into the working directory and are not removed. However, if the working directory does not already exist prior to running the command, the directory is created and the temporary files are generated into it, but the directory and all of the temporary files are later automatically removed when the deployment code generation is complete. (If you do not want the working directory to be removed when the deployment code generation is complete, use the `-keep` option.)

#### *output\_JAR\_name*

The fully qualified name of the output JAR file that is created by the `ejbdeploy` command and that contains the generated classes required for deployment; for example: `c:\ejb\outputJARs\myEJBs.jar`. (This argument is required.) The directories specified in the fully-qualified name must already exist before you run the `ejbdeploy` command. (Note that when you specify a name for the output JAR file and then run the `ejbdeploy` command, any existing output JAR file of the same name will be overwritten without warning.)

#### **-codegen**

Restricts the `ejbdeploy` command to just (a) importing code from the input JAR file (b) generating the deployment code, and (c) exporting code to the output JAR file. (If you do not specify the `-codegen` option, the `ejbdeploy` command performs the same operations as those performed using the `-codegen` option, but it will additionally compile the generated deployment code and run RMIC.)

#### **-cp classpath**

If you intend to run the `ejbdeploy` command against JAR files that have dependencies on other JAR files, you can use the `-cp` option to specify the classpath of the other JAR files. Using the `-cp` option, you can specify multiple JAR files as arguments. However, the JAR file names must be fully qualified, separated by semicolons, and enclosed in double quotation marks. For example:

```
-cp "path\myJar1.jar; path\myJar2.jar; path\myJar3.jar"
```

#### **-dbname name**

The name of the database you want to create. If the name of the database contains any spaces, the entire name must be enclosed in double quotes. For example:

```
-dbname "my database"
```

#### **-dbschema name**

The name of the schema you want to create. If the name of the schema contains any spaces, the entire name must be enclosed in double quotes. For example:

**-dbschema** "my schema"

(Note that this option is only used when creating a database definition in the top-down mode of operation. The database information is then saved in the schema document in the JAR file, meaning that the options do not need to be re-specified. It also means that when a JAR is generated by VisualAge for Java or WebSphere Studio Application Developer, the correct database must be defined at that point because it cannot be changed later.)

**-dbvendor** *name*

The name of the database vendor, which is used to determine database column types, mapping information, Table.ddl, and other information. The valid database vendor names are:

**SQL92** (Generic SQL 92)

**SQL99** (Generic SQL 99)

**DB2UDBWIN\_V61** (DB2 for Windows, V6.1)

**DB2UDBWIN\_V71** (DB2 for Windows, V7.1)

**DB2UDBOS390\_V6** (DB2 for OS/390, V6)

**DB2UDBAS400\_V4R5** (DB2 for OS/400, V4 R5)

**ORACLE\_V8** (Oracle, V8.0)

**INFORMIX\_V92** (Informix Dynamic Server.2000, V9.2)

**SYBASE\_V1192** (Sybase Adaptive Server Enterprise, V11.9.2)

**MSSQLSERVER\_V7** (MS SQL Server, V7)

**MYSQL\_V323** (MySQL, V3.23)

**-ignoreErrors**

Specifies that processing should continue even if compilation or validation errors are detected.

**-keep**

Retains the working directory after the ejbdeploy command has run.

**-noinform**

Suppresses informational messages.

**-novalidate**

Prevents the validators from running. (When you generate deployed code in batch mode from the command line, all of the installed validators are automatically run on the specified JAR file unless you specify this **-novalidate** option, which prevents all validators from running.) Note that the **-nowarn** option is generally a better options to use. It will hide any warnings but still display errors that may prevent the bean from deploying or running on the server.

**-nowarn**

Suppresses warning and informational messages.

**-quiet**

Suppresses status messages (but does not suppress error messages).

**-rmic** "*options*"

Enables you to pass RMIC options to RMIC. The options, which are described in Sun's RMI Tools documentation, must be separated by a space and enclosed in double quotation marks. For example:

**-rmic** "-nowarn -verbose"

**-trace**

Generates trace information.

For CMP entity beans, the EJB Deploy Tool looks in the JAR file for an existing schema and map to use when generating deployment code. If no existing schema and map are found, a schema and map are created using improved top-down mapping rules. If you want to use the same top-down mapping rules for CMP entity beans that are used in the EJB Deploy Tool provided with WebSphere Application Server, Version 3.5, then specify the -35 option. This may be desirable in some situations. However, if you do not specify the -35 option, a form of top-down mapping is used that is an improvement over what has previously been available. (In the top-down mapping approach, you already have existing enterprise beans and their design determines the database design. You generate a schema and map and the generated schema contains one table for each entity bean with container-managed persistence. In these tables, each column corresponds to a CMP field of the enterprise bean, and the generated mapping maps the field to the column.)

## Migrating CMP entity beans from WebSphere Application Server, Version 3.5

To migrate CMP entity beans from WebSphere Application Server, Version 3.5, to WebSphere Application Server, Version 4.0, the existing EJB applications should be migrated over and redeployed using the **-35** option of the EJBDeploy Tool so that the existing (older) table rules are kept. To migrate the beans manually, the EJB JAR requiring migration must be invoked through the command line and the -35 option should be used. Without this option, the database table will follow a new set of rules which is more aligned with the defaults for VisualAge for Java.

## Deploying and running VisualAge for Java enterprise beans in WebSphere Application Server

To run an enterprise bean created in VisualAge for Java on WebSphereApplication Server Version 4.0, you must first generate deployment code for it using the EJB Deploy Tool.

(Note that you cannot use the **EJB => Generate deployed code** menu item in VisualAge for Java, because it is designed to support the embedded VisualAge for Java WebSphere Test Environment that is based on WebSphereApplication Server, Version 3.5. Also, you cannot use the EJB Deploy Tool that is shipped with the June 2001 release of WebSphere Application Server, Version 4.0, because it does not support all of the function that is available in VisualAge for Java.)

To deploy and run a VisualAge for Java enterprise bean on WebSphereApplication Server, Version 4.0:

1. In the VisualAge for Java EJB Development Environment, select the **EJB => Export => Export EJB 1.1 JAR** menu item. This calls the Export Tool for Enterprise Java Beans 1.1, which enables you to create a JAR file that can be deployed to WebSphere Application Server. (If you don't use the Export Tool for Enterprise JavaBeans 1.1, no mappings, inheritance, or associations will be exported.)
2. Open a command window and change to the directory where you installed the EJB Deploy Tool.
3. Run the ejbdeploy command and use the JAR that you created in VisualAge for Java as your input JAR



file. This will create an output deployed JAR file.

4. Install the deployed JAR file into WebSphere Application Server, Version 4.0. Depending on your edition of WebSphere Application Server, this can be accomplished using either the SEAppInstall tool or the WebSphere Application Assembly Tool. If your edition of WebSphere Application Server includes the SEAppInstall tool and you want to use the SEAppInstall tool from the command line, ensure that you specify the "-ejbdeploy false" option, so that the earlier version of the EJB Deploy Tool that shipped with WebSphere Application Server Version 4.0 is not invoked.

Note that for CMP entity beans, a data definition language (DDL) that can be used to create corresponding database tables for a JAR file is contained within the META-INF directory and entitled Table.ddl. (In VisualAge for Java, Version 3.5, this information used to be directed into the generated persister. The table creation is now a task left up to the user, but it is made more convenient by the Table.ddl file.)

# 6.6.0.15.2: Meet-in-the-middle mapping support

By default, the `ejbdeploy` command automatically generates top-down mapping. However, after completion of the top-down operation, you can modify the resulting map and schema to fit your existing data or models (meet-in-the-middle). Follow the steps below if you want to modify the default schema and/or map.

To start this process, you need a standard EJB-1.0 or 1.1 JAR file.

1. Run the `ejbdeploy` utility using the `-codegen` argument.
2. Modify the database `schema`.
3. Modify the `map`.
4. `Generate deployment code` for your enterprise beans by running the `ejbdeploy` utility again. This produces a deployed JAR that can be installed directly into WebSphere Application Server 4.0.

# 6.6.0.15.3:Modifying the schema document

A schema document is created during the top-down mapping generation done by the ejbdeploy tool. The schema document contains descriptions of tables and their columns. After completing the top-down operation, the resulting schema may need to be modified to fit existing data or models (meet-in-the-middle mapping). This topic discusses that process.

## XMI vocabulary: Schema.rdbxmi

This topic discusses the structure and vocabulary of an XMI schema document. The following is a sample XMI code block from the schema.rdbxmi file that describes the BankAccount table, all its columns and constraints (Primary Key).

```
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:RDBSchema="RDBSchema.xmi">
<RDBSchema:RDBDatabase xmi:id="TopDownDB_ID" name="TopDownDB" schemata="USERID_ID"
tableGroup="BankAccount">    <dataTypes href="UDBV7_Primitives.xmi#SQLPrimitives_1"/>
</RDBSchema:RDBDatabase>    <RDBSchema:RDBSchema xmi:id="USERID_ID" name="USERID"
database="TopDownDB_ID" tables="BankAccount"/>    <RDBSchema:RDBTable xmi:id="BankAccount"
name="BankAccount" primaryKey="SQLReference_1" database="TopDownDB_ID" schema="USERID_ID">
<namedGroup xsi:type="RDBSchema:SQLReference" xmi:id="SQLReference_1" members="RDBColumn_1"
table="BankAccount" constraint="Constraint_BankAccountPK"/>    <constraints
xmi:id="Constraint_BankAccountPK" name="BankAccountPK" type="PRIMARYKEY"
primaryKey="SQLReference_1"/>    <columns xmi:id="RDBColumn_1" name="accountNum" allowNull="false"
group="SQLReference_1">    <type xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_1">
<originatingType xsi:type="RDBSchema:SQLExactNumeric"
href="UDBV7_Primitives.xmi#SQLExactNumeric_1"/>    </type>    </columns>    <columns
xmi:id="RDBColumn_2" name="customerName">    <type xsi:type="RDBSchema:SQLCharacterStringType"
xmi:id="SQLCharacterStringType_1" characterSet="800" length="32">    <originatingType
xsi:type="RDBSchema:SQLCharacterStringType" href="UDBV7_Primitives.xmi#SQLCharacterStringType_3"/>
</type>    </columns>    <columns xmi:id="RDBColumn_3" name="balance">    <type
xsi:type="RDBSchema:SQLNumeric" xmi:id="SQLNumeric_1" precision="5" scale="0">
<originatingType xsi:type="RDBSchema:SQLNumeric" href="UDBV7_Primitives.xmi#SQLNumeric_3"/>
</type>    </columns>    </RDBSchema:RDBTable> </xmi:XMI>
```

## XML element descriptions

**RDBSchema:RDBDatabase** - defines the name and id of the database.

**RDBSchema:RDBTable** - defines the name and id of the table.

**RDBSchema:SQLConstraint** - defines the constraints, this sample only includes the primary key constraint. The constraint points to another object that describes the type of constraint. In this case, the SQLReference "BankAccountPK" describes the primary key, and what members (columns) are included.

**RDBSchema:RDBColumn** - defines the column type by referencing a type instance in a sqlprimitives document. The primitives documents are located in the /plugins/sqlmodel/runtime/primitives directory.

## Changing type definitions

If you need to change between two closely-related types, such as, VARCHAR to LONGVARCHAR, you only need to change the type id in the href. A column might look like this:

```
<columns xmi:id="RDBColumn_52" name="location">
<type xsi:type="RDBSchema:SQLCharacterStringType" xmi:id="SQLCharacterStringType_20" characterSet="800" length="1">
  <originatingType xsi:type="RDBSchema:SQLCharacterStringType"
  href="ORACLEV8i_Primitives.xmi#SQLCharacterStringType_2"/>
</type>
</columns>
```

In this case, the href is referring to the following type in ORACLEV8i\_Primitives.xmi:

```
<types xsi:type="RDBSchema:SQLCharacterStringType"
xmi:id="SQLCharacterStringType_2" externalName="CHARACTERVARYING" name="VARCHAR2"
dbcEnumType="12 domain="ORACLE_V8" requiredUniqueInstance="true" renderedString="VARCHAR2"
typeEnum="CHARACTERVARYING" formatterClassName="com.ibm.etools.rdbschema.formatter.oracle.CharacterTextFormatter"
characterSet="800" length="1"/>
```

If you just need to change the length, you can do so by directly changing the length (seen in red in the code snippet above) on the type object. If you want to switch to one of the other character types, you can change the type reference (seen in red in the code snippet above) to one of the other similar

types in the primitives doc. For example, SQLCharacterStringType\_1 to SQLCharacterStringType\_6.As long as you are switching between closely-related types, you do not need to change the syntax of the <type>.

For numerics, the same thing will apply. You should only have to modify the precision and scale. However, you can get the more specific numeric type you want by switching between the SQLNUMERIC\_ types.

# Adding table columns

Adding a new column to the table is easy if you use the following steps:

1. Add the new column under the columns tag. Make sure the order of the column in the list corresponds to the order in the database.

```
<columns xmi:id="RDBColumn_1" name="accountNum" allowNull="false" group="SQLReference_1">
```

2. Add the column type.

```
<type xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_1"> <originatingType  
xsi:type="RDBSchema:SQLExactNumeric" href="UDBV7_Primitives.xmi#SQLExactNumeric_1"/></type>
```

Make sure all of the id's chosen are unique in the entire XMIdocument.

# More examples of type definitions

The following type definition examples were taken from UDBV6\_Primitives.xmi. This and other type definition example files can be found in the following location: X:\..\eclipse\plugins\sqlmodel\runtime\primitives\

CHARACTER FOR BIT DATA	<types xmi:type="RDBSchema:SQLBinaryLargeObject" xmi:id="SQLBinaryLargeObject_3" externalName="BINARY LARGE OBJECT" name="CHARACTER () FOR BIT DATA" jdbcEnumType="-2" domain="DB2UDBNT_V61" requiredUniqueInstance="true" renderedString="CHARACTER" typeEnum="BINARYLARGEOBJECT" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.BinaryTextFormatter" length="1"/>
VARCHAR FOR BIT DATA	<types xsi:type="RDBSchema:SQLBinaryLargeObject" xmi:id="SQLBinaryLargeObject_5" externalName="BINARY LARGE OBJECT" name="VARCHAR () FOR BIT DATA" jdbcEnumType="-3" domain="DB2UDBNT_V61" requiredUniqueInstance="true" renderedString="VARCHAR" typeEnum="BINARYLARGEOBJECT" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.BinaryTextFormatter" length="1"/>
SMALLINT	<types xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_4" externalName="SMALLINT" name="SMALLINT" jdbcEnumType="5" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="SMALLINT" typeEnum="SMALLINT" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
FLOAT	<types xsi:type="RDBSchema:SQLApproximateNumeric" xmi:id="SQLApproximateNumeric_4" externalName="DOUBLE PRECISION" name="FLOAT" jdbcEnumType="8" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="DOUBLE" typeEnum="DOUBLEPRECISION" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
DOUBLE	<types xsi:type="RDBSchema:SQLApproximateNumeric" xmi:id="SQLApproximateNumeric_2" externalName="DOUBLE PRECISION" name="DOUBLE" jdbcEnumType="8" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="DOUBLE" typeEnum="DOUBLEPRECISION" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
LONG VARCHAR FOR BIT DATA	<types xsi:type="RDBSchema:SQLBinaryLargeObject" xmi:id="SQLBinaryLargeObject_8" externalName="BINARY LARGE OBJECT" name="LONG VARCHAR FOR BIT DATA" jdbcEnumType="-4" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="LONG VARCHAR" typeEnum="BINARYLARGEOBJECT" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.BinaryTextFormatter" length="0"/>

DATE	<types xsi:type="RDBSchema:SQLDate" xmi:id="SQLDate_1" externalName="DATE" name="DATE" jdbcEnumType="91" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="DATE" typeEnum="DATE" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
SMALLINT	<types xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_4" externalName="SMALLINT" name="SMALLINT" jdbcEnumType="5" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="SMALLINT" typeEnum="SMALLINT" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
DECIMAL ( 20 , 2 )	<types xsi:type="RDBSchema:SQLNumeric" xmi:id="SQLNumeric_1" externalName="NUMERIC" name="DECIMAL" jdbcEnumType="2" domain="DB2UDBNT_V61" requiredUniqueInstance="true" renderedString="NUMERIC" typeEnum="NUMERIC" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.NumericTextFormatter" precision="5" scale="0"/>
REAL	<types xmi:type="RDBSchema:SQLApproximateNumeric" xmi:id="SQLApproximateNumeric_1" externalName="REAL" name="REAL" jdbcEnumType="7" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="REAL" typeEnum="REAL" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
INTEGER	<types xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_1" externalName="INTEGER" name="INTEGER" jdbcEnumType="4" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="INTEGER" typeEnum="INTEGER" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
BIGINT	<types xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_3" externalName="INTEGER" name="BIGINT" jdbcEnumType="-5" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="BIGINT" typeEnum="INTEGER" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
TIME	<types xsi:type="RDBSchema:SQLTime" xmi:id="SQLTime_1" externalName="TIME" name="TIME" jdbcEnumType="92" domain="DB2UDBNT_V61" requiredUniqueInstance="true" renderedString="TIME" typeEnum="TIME" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter" precision="2" timezone="false"/>
TIMESTAMP	<types xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_1" externalName="INTEGER" name="INTEGER" jdbcEnumType="4" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="INTEGER" typeEnum="INTEGER" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
BIGINT	<types xsi:type="RDBSchema:SQLExactNumeric" xmi:id="SQLExactNumeric_3" externalName="INTEGER" name="BIGINT" jdbcEnumType="-5" domain="DB2UDBNT_V61" requiredUniqueInstance="false" renderedString="BIGINT" typeEnum="INTEGER" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.SimpleTextFormatter"/>
CLOB	<types xmi:type="RDBSchema:SQLCharacterLargeObject" xmi:id="SQLCharacterLargeObject_1" externalName="CHARACTER LARGE OBJECT" name="CLOB" jdbcEnumType="2005" domain="DB2UDBNT_V61" requiredUniqueInstance="true" renderedString="CLOB" typeEnum="CHARACTERLARGEOBJECT" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.LargeObjectTextFormatter" characterSet="800" length="1" multiplier="K"/>
BLOB	<types xsi:type="RDBSchema:SQLBinaryLargeObject" xmi:id="SQLBinaryLargeObject_1" externalName="BINARY LARGE OBJECT" name="BLOB" jdbcEnumType="2004" domain="DB2UDBNT_V61" requiredUniqueInstance="true" renderedString="BLOB" typeEnum="BINARYLARGEOBJECT" formatterClassName="com.ibm.etools.rdbschema.formatter.db2nt.LargeObjectTextFormatter" length="1" multiplier="K"/>

# 6.6.0.15.4: Modifying the map document

A map consists of mapping the model attributes and relationships to the schema. The map connects the object model description with the schema description. Once the map is defined, the persistence support for your data store is created during code generation.

## XML vocabulary: Map.mapxmi

After completing the top-down operation, the resulting map may need to be modified to fit existing data or models.

This topic discusses the structure and vocabulary of an XML map document. The following is a sample XML code block from the map.mapxmi file that shows the mapping between the BankAccount CMP, and the BANKACCOUNT table.

```
<ejbrdbmapping:EjbRdbDocumentRoot xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:ejbrdbmapping="ejbrdbmapping.xml" xmlns:Mapping="Mapping.xml" xmlns:ejb="ejb.xml"
xmlns:RDBSchema="RDBSchema.xml" xmlns:java="java.xml" xmi:id="EjbRdbDocumentRoot_1"
outputReadOnly="false" topToBottom="true"> <helper xsi:type="ejbrdbmapping:RdbSchemaProperties"
xmi:id="RdbSchemaProperties_1" primitivesDocument="DB2UDBNT_V71"/> <inputs xsi:type="ejb:EJBJar"
href="META-INF/ejb-jar.xml#ejb-jar_ID"/> <outputs xsi:type="RDBSchema:RDBDatabase"
href="META-INF/Schema/Schema.dbxmi#TopDownDB_ID"/> <nested xsi:type="ejbrdbmapping:RDBEjbMapper"
xmi:id="RDBEjbMapper_1"> <helper xsi:type="ejbrdbmapping:PrimaryTableStrategy"
xmi:id="PrimaryTableStrategy_1"> <table href="META-INF/Schema/Schema.dbxmi#BankAccount"/>
</helper> <inputs xsi:type="ejb:ContainerManagedEntity" href="META-INF/ejb-jar.xml#BankAccount"/>
<outputs xsi:type="RDBSchema:RDBTable" href="META-INF/Schema/Schema.dbxmi#BankAccount"/> <nested
xmi:id="BankAccount_accountNum---BankAccount_accountNum"> <inputs xsi:type="ejb:CMPAttribute"
href="META-INF/ejb-jar.xml#BankAccount_accountNum"/> <outputs xsi:type="RDBSchema:RDBColumn"
href="META-INF/Schema/Schema.dbxmi#RDBColumn_1"/> <typeMapping
href="JavatoDB2UDBNT_V71TypeMaps.xml#int-INTEGER"/> </nested> <nested
xmi:id="BankAccount_customerName---BankAccount_customerName"> <helper
xsi:type="ejbrdbmapping:EJBConverter" xmi:id="EJBConverter_1"> <targetClass
href="java:/java.lang#String"/> <transformerClass
href="java:/com.ibm.vap.converters#VapTrimStringConverter"/> </helper> <inputs
xsi:type="ejb:CMPAttribute" href="META-INF/ejb-jar.xml#BankAccount_customerName"/> <outputs
xsi:type="RDBSchema:RDBColumn" href="META-INF/Schema/Schema.dbxmi#RDBColumn_2"/> <typeMapping
href="JavatoDB2UDBNT_V71TypeMaps.xml#String-VARCHAR"/> </nested> <nested
xmi:id="BankAccount_balance---BankAccount_balance"> <inputs xsi:type="ejb:CMPAttribute"
href="META-INF/ejb-jar.xml#BankAccount_balance"/> <outputs xsi:type="RDBSchema:RDBColumn"
href="META-INF/Schema/Schema.dbxmi#RDBColumn_3"/> <typeMapping
href="JavatoDB2UDBNT_V71TypeMaps.xml#BigDecimal-DECIMAL"/> </nested> </nested> <typeMapping
xsi:type="Mapping:MappingRoot"
href="JavatoDB2UDBNT_V71TypeMaps.xml#Java_to_DB2UDBNT_V71_TypeMaps"/></ejbrdbmapping:EjbRdbDocumentRoot>
```

## XML element descriptions

**helper** - defines helper objects that are used to set schema properties, table strategy, and as converters between attribute and column types.

**inputs** - defines CMP references using an href tag that uses a path statement relative to the META-INF directory.

**outputs** - defines database table references using an href tag that uses a path statement relative to the META-INF directory.

**nested** - defines children mappings for attributes and columns. These mappings use the mapping object to describe the inputs, outputs, and helpers.

**typeMapping** - an optional element in the map document.

## Adding mappings

The procedure for adding a new mapping is easy if you use the following steps:

1. Under the nested tag, add a new Mapping object.

```
<helper xsi:type="ejbrdbmapping:RdbSchemaProperties" xmi:id="RdbSchemaProperties_1"
primitivesDocument="DB2UDBNT_V71"/> <inputs xsi:type="ejb:EJBJar"
href="META-INF/ejb-jar.xml#ejb-jar_ID"/> <outputs xsi:type="RDBSchema:RDBDatabase"
href="META-INF/Schema/Schema.dbxmi#TopDownDB_ID"/> <nested xsi:type="ejbrdbmapping:RDBEjbMapper"
xmi:id="RDBEjbMapper_1"> <helper xsi:type="ejbrdbmapping:PrimaryTableStrategy"
xmi:id="PrimaryTableStrategy_1"> <table href="META-INF/Schema/Schema.dbxmi#BankAccount"/>
</helper>
```

2. Add the CMPAttribute and RDBColumn references.

```
<nested xmi:id="BankAccount_accountNum---BankAccount_accountNum"> <inputs
xsi:type="ejb:CMPAttribute" href="META-INF/ejb-jar.xml#BankAccount_accountNum"/> <outputs
xsi:type="RDBSchema:RDBColumn" href="META-INF/Schema/Schema.dbxmi#RDBColumn_1"/> <typeMapping
```

```
href="JavatoDB2UDBNT_V71TypeMaps.xmi#int-INTEGER"/> </nested>
```

Make sure all of the id's chosen are unique in the entire document.

# 6.6.0.15.5: Implementing custom finder helpers for CMP entity beans

To work with custom finder helpers for CMP entity beans, there are two tasks that you need to complete:

- Implement the custom finder helpers
- Add the custom finder helpers to the home interface

The task of adding custom finder helpers to the home interface is found in the WebSphere Studio Application Developer help topic "Adding methods to the home and remote interfaces". The task of implementing custom finder helpers is described in the remainder of this topic in the following sections:

- [Introduction to working with custom finder helpers](#)
- [Implementing custom finders using EJB query language and SQL](#)
- [Defining custom finder helpers in deployment descriptor extension documents](#)
- [The AutoWorld example](#)
  - [Using the finder helper interface](#)

## Introduction to working with custom finder helpers

To implement custom finder helpers, you can use one of two supported query languages:

- EJB query language
- SQL

EJB query language is recommended for implementing custom finder helpers, but the use of SQL is still supported.

To implement custom finder helpers, you should use a deployment descriptor extension document rather than a helper finder interface. Both EJB query language strings and SQL query strings and method declarations can be used in a deployment descriptor extension document.

The use of helper finder interfaces has been deprecated but is still supported to some extent. If you are working with existing EJB 1.0 JAR files, you can continue to define SQL query strings or method declarations in the helper finder interface. (The SQL query string is actually a field on the interface describing the full SELECT statement or just the WHERE clause.)

However, for any new development work that requires you to work with EJB JAR files at a level higher than 1.0, it is required that you use a deployment descriptor extension document rather than the finder helper interface to define your queries or method declarations.

## Implementing custom finders using EJB query language and SQL

The EJB query language defines finder methods for entity beans with container-managed persistence. The definition uses a language based on SQL that allows searches on the persistent attributes of an Enterprise Java Bean and associated bean attributes. The query language is independent of the bean's mapping to a relational data store and is portable. The query language is compiled into SQL at deployment time based on the schema mapping for the bean.

An EJB query is a string that contains

- an optional SELECT clause that specifies the EJB objects to return
- a FROM clause that names the bean collections
- an optional WHERE clause that contains search predicates over the collections
- an optional ORDER BY clause that specifies the ordering of the result collection.

An EJB query also contains input parameters that correspond to the arguments of the finder method.

Additional information is found in the WebSphere Studio Application Developer help topic "EJB query language - overview".

There are four types of custom finders that are currently supported for enterprise beans and that can be used in combination:

- SQL SELECT
- SQL WHERE
- SQL Method
- EJB query language

The SELECT, WHERE, and method custom finders are standard SQL techniques used to query a database column.

For each finder method that is defined in the EJB home interface (other than `findByPrimaryKey` and those finder methods generated to support associations), one of the following queries or declarations must be defined in the deployment descriptor extension document or in the finder helper interface (in file `beanClassNameFinderHelper.java`):

- An SQL SELECT or WHERE query tag (for extension documents) or string (for finder helper interfaces)
- A method declaration (for method custom finders)



- An EJB query language tag or string

The return type `java.util.Enumeration` or `java.util.Collection` on the finder in the home interface indicates that this finder may return more than one bean. Using the remote interface as the return type indicates that a single bean is returned. This is true for all of the supported types of custom finders. The code generated into the persister handles this distinction.

For SQL SELECT, WHERE, and method custom finders, there may be situations where the finder methods access different databases. In this case, it is necessary to ensure that SQL compatibility is maintained across the different databases. For instance, it is possible that the SQL syntax used by each database is different. In these situations, use the SQL extensions defined by JDBC to resolve the database differences. For example, assume that you are developing a CMP entity bean that requires a finder method that involves a timestamp/date field. Also assume that this bean will be deployed to DB2 and Oracle databases. The problem is that the format of the timestamp/date fields in DB2 and Oracle are different, which causes difficulties in defining one WHERE clause for use with both DB2 and Oracle. The solution to this particular problem is to use the SQL Escape sequence.

## Defining custom finder helpers in deployment descriptor extension documents

If you are working with existing EJB 1.0 JAR files, you can continue to define SQL query strings, method declarations, or EJB query language strings in the helper finder interface. However, this mechanism for defining queries has several restrictions, such as the inability to create polymorphic finders. Therefore, for any new development work, it is recommended that you use a deployment descriptor extension document rather than the finder helper interface to define your queries and method declarations. The extension document you need to use is saved in the JAR file and is named:

`META-INF/ibm-ejb-jar-ext.xml`

The extension document contains query tags rather than query strings. You can edit the extension document much like a deployment descriptor file. The following example uses standard SQL and EJB query language finder syntax and will help you understand the format of the extensions document. You can use the example as a template and copy/paste the tagging for your own use.

```
<ejb-ext:EJBJarExtension xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:ejb-ext="ejb-ext.xml" xmlns:ejb="ejb.xml" xmlns:ecore="ecore.xml"
xmi:id="ejb-jar_ID_Ext"><ejbJar href="META-INF/ejb-jar.xml#ejb-jar_ID"/><ejbExtensions
xsi:type="ejb-ext:ContainerManagedEntityExtension" xmi:id="Department_Ext"
name="Department"><enterpriseBean xsi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#Department"/><finderDescriptors
xsi:type="ejb-ext:FullSelectFinderDescriptor"
xmi:id="Department_findAllQueryString_FullSelect_ID" selectStatement="SELECT * FROM
T1.DEPARTMENT"><finderMethodElements xmi:id="MethodElement_10" name="findAll"
parms="" type="Home"><enterpriseBean xsi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#Department"/></finderMethodElements></finderDescriptors><finderDescriptors
xsi:type="ejb-ext:WhereClauseFinderDescriptor" xmi:id="Department_findByName_WhereClause_ID"
whereClause="T1.NAME LIKE ?"><finderMethodElements xmi:id="MethodElement_11" name="findByName"
parms="java.lang.String" type="Home"><enterpriseBean xsi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#Department"/></finderMethodElements></finderDescriptors><finderDescriptors
xsi:type="ejb-ext:EjbqlFinderDescriptor" xmi:id="Department_findByName_ejbql_ID"
ejbqlQueryString="select e from DepartmentBean name like ?"><finderMethodElements
xmi:id="MethodElement_12" name="findByEjbName" parms="java.lang.String"
type="Home"><enterpriseBean xsi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#Department"/></finderMethodElements></finderDescriptors><finderDescriptors
xsi:type="ejb-ext:UserFinderDescriptor"
xmi:id="Department_findSpecial_User_ID"><finderMethodElements xmi:id="MethodElement_13"
name="findSpecial" type="Home"><enterpriseBean xsi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#Department"/></finderMethodElements></finderDescriptors></ejbExtensions></ejbJar>
```

For each finder method in the extension document, you need to provide a `finderDescriptor` element that consists of two parts: the `finderMethodElement`, which identifies the method to which the finder descriptor applies, and the query string, which can be specified as one of the following types:

- SQL "whereClause" (corresponds to finder helper interface strings)
- SQL "selectStatement" (corresponds to finder helper interface strings)
- EJB query language string (specifies the query in terms of the EJB 2.0 EJB Query Language)

These three types of query string have the following unique `xsi:type`:

- `ejb-ext:WhereClauseFinderDescriptor`
- `ejb-ext:FullSelectFinderDescriptor`
- `ejb-ext:EjbqlFinderDescriptor`

The name of the class that contains the custom finder method is *beanClassNameFinderObject*, where *beanClassName* is the name of the implementation class for the bean. The `FinderObject` class must be in the same package as the bean implementation class.

In the extensions document, you can also employ a user finder descriptor, which the deployment tool uses to determine the finder implementation method provided by the user. In the above example, the last `finderDescriptors` tag specifies a user finder descriptor. This descriptor simply serves as a placeholder, because it contains no "string" like the other descriptors. The name of the custom finder is there, however. It is named `findSpecial`. As a result, in the *beanClassNameFinderObject* class provided by the bean developer, there must be a corresponding method called `findSpecial`.

Since two separate `findSpecial` methods are used, care must be taken to understand the difference between the two. The essential difference is that the bean developer defines one `findSpecial` method in the home interface and the other `findSpecial` method in the `FinderObject` class. Both take the same arguments, but they return different types and serve different purposes. They are given the same name as a convenient way to tie them together.

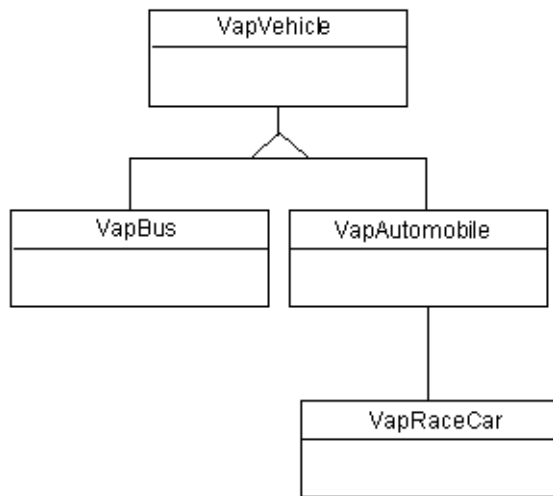
When a bean developer needs a custom finder, the developer defines it in the `finder` interface. For CMP entity beans, the EJB Deploy Tool generates the code that implements the custom finder. For all but the custom method finder, all that is needed is a string. For the custom method finder, the bean developer needs to actually implement a method in the `FinderObject`. The deploy tool still generates the code that implements the `findSpecial` custom finder, but that generated code will call the `findSpecial` method in the `FinderObject`.

During deployment, if a JAR file contains finder helper interfaces, the EJB Deploy Tool will migrate the appropriate SQL strings into a field contained within the extensions document (`ibm-ejb-jar-ext.xml`). This string is what is used during deployment to emit code contained within the emitted JDBC persister. In WebSphere Application Server, Version 3.5, the finder helper interface was used directly. Once the information exists in the extensions document, that information will be used and any subsequent changes to the finder helper interface will be ignored. This field can be modified by hand or by using the Application Assembly tool.

Note that when enterprise beans are exported from VisualAge for Java, Enterprise Edition, Version 4.0 using the Export Tool for Enterprise Java Beans 1.1, or when 1.0 EJB JAR files are deployed with the deployment tool, the finder meta data is migrated to the new form from the finder helper interfaces. If the enterprise beans are exported using the Export Tool for Enterprise Java Beans 1.1, the redundant classes are filtered from the exported JAR. If the enterprise beans are not exported using the Export Tool for Enterprise Java Beans 1.1 and are imported along with the redundant classes, the classes are simply ignored.

## The AutoWorld example

To introduce this AutoWorld example, consider the following hierarchy of CMP entity beans:



Outside of this CMP bean hierarchy, there are two other existing CMP entity beans: `VapGarage` and `VapMotorVehiclePart`. An association exists between `VapGarage` and `VapVehicle`, in which a garage can hold many vehicles. Another association exists between `VapMotorVehiclePart` and `VapVehicle`, in which a vehicle can contain many parts.

In the remainder of this topic, the AutoWorld example is used to show you how to work with queries defined in the finder helper interface.

## Using the finder helper interface

In this section, the AutoWorld example is used to show you how the following custom finders are defined in the finder helper interface:

- [SQL SELECT custom finders](#)
- [SQL WHERE custom finders](#)
- [Method custom finders](#)

### SQL SELECT custom finders

A SELECT custom finder is used to enter an entire select statement in the finder helper interface to define the SQL query.

The use of SELECT custom finders is supported for compatibility with earlier releases. Using SELECT custom finders is discouraged in this and future releases.

### SQL WHERE custom finders

A custom finder in which you enter only the filtering WHERE clause into the finder helper interface is called a WHERE custom finder. Using the same garage example as before, the finder helper interface would look like this:

```
public interface VapGarageBeanFinderHelper {    public final static String
findCapacityGreaterThanWhereClause =          "T1.CAPACITY > ?"; }
```

Notice that any dependency on the shape of the results is removed from the string. Two dependencies still exist:

- The name of the column (CAPACITY)
- The alias for the table (T1)

The name of the column would change only if you took action to change it.

The alias for the table will be the same from one generation to the next unless tables are added to or removed from the mapping. In single-table cases, this may not seem significant (the alias is always T1). When multiple tables are used, this is very important.

For example, consider the VapVehicle hierarchy. When it is mapped using root/leaf inheritance mapping, there is at least one table for each class in the hierarchy, and the query into which the WHERE clause would be inserted would have multiple subselects. The WHERE clause would be inserted into each subselect, and it would be valid SQL syntax for each subselect to use a different alias for the same table. However, our query generation ensures that the same table has the same alias across the entire query. If it did not, this WHERE clause substitution technique could not work. In these cases, the code generated into the persister knows to inject any finder parameters into the query multiple times.

There is one very important restriction when using this form of custom finder in a mapped hierarchy: The WHERE clause can only reference tables that map the bean in which the finder is defined or tables that map one of bean's parent beans.

For example, a WHERE clause in the VapVehicleBeanFinderHelper can only reference columns in the table used to map the VapVehicle bean. However, a WHERE in the VapAutomobileBeanFinderHelper can reference columns from either the table that maps VapAutomobile or VapVehicle (its superclass).

Any table references in your handwritten SQL code must match the table aliases set up in the genericFindSqlString field. This is declared in the enterprise bean's generated persister.

As in the SELECT custom form, the number of finder parameters must match the number of injection points (the ? characters) in the WHERE clause. Also, as in the SELECT form, the type of the parameter will be used to determine which java.sql.PreparedStatement set call will be used to inject each parameter. The parameter types must be compatible with the column types. If the type is an object type and the parameter is null, a setNull call will be used.

For example, the home interface may contain the following method:

```
public java.util Enumeration findGreaterThan (int threshold) throws java.rmi.RemoteException,
javax.ejb.FinderException;
```

In this finder helper interface, the WHERE custom finder is one of the forms that you can provide. For example (line broken for publication):

```
public static final String findGreaterThanWhereClause = "T1.VALUE > ?";
```

Note, however, that if you have an SQL statement that contains no WHERE clause, such as SELECT \* FROM MYTABLE, you should use a query string that always evaluates to true. For example:

```
public static final String findALLWhereClause = "1 = 1";
```

## Method custom finders

A custom finder in which you enter a method signature into the finder helper interface is called a method custom finder. It is the most flexible type of custom finder, but it requires more work on your part. Using the same garage example as before, the finder helper interface would look like this:

```
public interface VapGarageBeanFinderHelper { public java.sql.PreparedStatement
findCapacityGreaterThan(int threshold) throws Exception;}
```

Unlike the SELECT and WHERE forms, however, this is not enough for method custom finders. An implementation of this method is needed. You provide your implementation of the method in a class that follows these rules:

- The name of the class is *beanClassNameFinderObject* (VapGarageBeanFinderObject, in this example) and is in the same package as the bean class.
- The class must extend com.ibm.vap.finders.VapEJSJDBCFinderObject and must implement the bean's finder helper interface.
- Any table references in your handwritten SQL code must match the table aliases set up in the genericFindSqlString field. This is declared in the enterprise bean's generated persister.

To finish off our example, the finder object would look like this.

```
/** * Implementation class for methods in * VapGarageBeanFinderHelper. */ public class
VapGarageBeanFinderObject extends com.ibm.vap.finders.VapEJSJDBCFinderObject implements
VapGarageBeanFinderHelper { public java.sql.PreparedStatement findCapacityGreaterThan(int
threshold) throws Exception { PreparedStatement ps = null; int mergeCount =
getMergedWhereCount(); int columnCount = 1; ps = getMergedPreparedStatement("T1.CAPACITY
> ?"); for (int i=0; i<(columnCount*mergeCount); i=i+columnCount) { ps.setInt(i+1,
threshold); } return ps; }}
```

In the case of any method custom finder, the generated persister uses your implementation to create the PreparedStatement to be executed. The persister will execute the PreparedStatement and handle the results. The implementation needs help from the persister to make sure the result set for the query has the correct shape. The com.ibm.vap.finders.VapEJSJDBCFinderObject base class provides several important helper methods, some of which are shown in the above example. In the following table, the complete set of helper methods are listed and described:

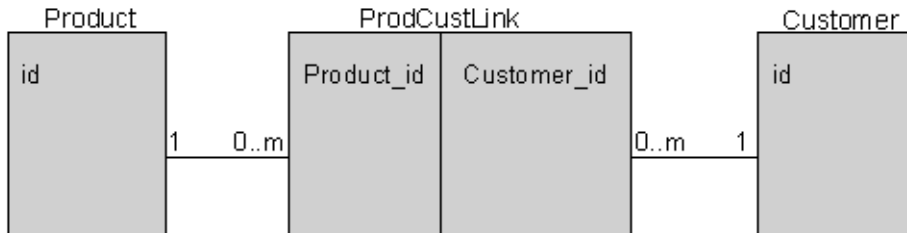
Method	Description
getMergedPreparedStatement	Takes a WHERE clause and returns a PreparedStatement with the WHERE clause merged into the appropriate places. The PreparedStatement will have the correct result set shape.
getMergedWhereCount	Returns the number of times the WHERE clause is merged into the PreparedStatement. This is needed to know how many times to inject your query parameters into the PreparedStatement.
getPreparedStatement	Takes a complete query string and returns a PreparedStatement. This can be used if for some reason you need to do your own WHERE clause merging. This should be a very rare case. The next two functions are provided to help in these extreme cases.
getGenericFindSqlString	Returns the query string into which WHERE clauses are merged.
getGenericFindInsertPoints	Returns an array of integers that defines the point or points in the getGenericFindSqlString returned query string at which the WHERE clause is merged. The first point in the array is the last point in the string. It is usually best to merge from the end of the query string since a merge at the end will not change the location of merges earlier in the string. The size of the array is the same as the value returned from getMergedWhereCount.

This is a rather simple case that can better be handled by a WHERE custom finder. More complex examples are possible that a WHERE custom finder simply could not handle. For example, suppose you wanted a finder that took a more complex object and injected it into multiple columns in a WHERE clause. You could end up with a finder method that looked like this:

```
public java.sql.PreparedStatement findWithComplexObject(BigObject big) throws Exception {
PreparedStatement ps = null;    int mergeCount = getMergedWhereCount();    int columnCount = 3;
int anInt = big.getAnInt();    String aString = big.getAString();    String aLongAsString =
com.ibm.vap.converters.VapStringToLongConverter.    singleton().dataFrom(big.getLongObject());
ps = getMergedPreparedStatement("(T1.ANINT > ?) AND (T1.ASTRING = ?) AND (T2.ALONGSTR <
?)");    for (int i=0; i<(columnCount*mergeCount); i=i+columnCount) {        ps.setInt(i+1, anInt);
        if (aString == null)            ps.setNull(1, java.sql.Types.VARCHAR);        else
ps.setString(i+2, aString);            if (aLongAsString == null)            ps.setNull(1,
java.sql.Types.VARCHAR);        else            ps.setString(i+3, aLongAsString);    }    return ps; }
```

Even more complex examples are possible. For instance, an object could be passed that contains the WHERE clause (or instructions on how to create it) in addition to the data. Or, there could be multiple parameters, each representing different conditions in the WHERE clause.

The following example is a logical representation of how a many-to-many association could be accomplished using a complex method custom finder. The example involves a many-to-many association between Product and Customer beans, using an intermediary bean (ProdCustLink) and two 1:m associations:



You can write method custom finders to span the relationship in either direction with just one method call. For this example, consider one direction only: a finder in Customer that retrieves all Customer instances that are associated with a given product key.

Customer's home interface contains the appropriate method signature, as follows:

```
java.util Enumeration    findCustomersByProduct(prod.cust.code.ProductKey inKey)    throws
java.rmi.RemoteException, javax.ejb.FinderException;
```

Customer's finder helper interface contains the signature for the corresponding finder method:

```
public java.sql.PreparedStatement    findCustomersByProduct(prod.cust.code.ProductKey inKey)
throws Exception;
```

The finder object (CustomerBeanFinderObject) builds and caches the query string for the finder as well as implements the finder method.

```
public class CustomerBeanFinderObject    extends com.ibm.vap.finders.VapEJSJDBCFinderObject
implements CustomerBeanFinderHelper {    private String cachedFindCustomersByProductQueryString =
null;    .    .    . }
```

Through lazy initialization in the finder object, the accessor method for the query-string field builds up the query string by first merging the WHERE condition into the query template and then adding a reference to the intermediate table into the FROM clause.

The first half of the accessor method uses a genericFindInsertPoints array to locate and update each WHERE clause. Then, the second half of the method counts forward from the beginning of each FROM clause, inserts the reference to the intermediate table into the query string as needed, and updates the query-string field.

```
protected String getFindCustomersByProductQueryString() {    if
(cachedFindCustomersByProductQueryString == null) {        // Do the WHERE first        // so that the
genericFindInsertPoints are correct.        int i;        int[] genericFindInsertPoints =
getGenericFindInsertPoints();        StringBuffer sb = new StringBuffer(getGenericFindSqlString());
for (i = 0; i < genericFindInsertPoints.length; i++) {            sb.insert(genericFindInsertPoints[i],
"(T1.id = T2.Customer_id) AND (T2.Product_id = ?)");        }        // Make sure to update every FROM
clause.        String soFar = sb.toString();        int fromOffset = soFar.indexOf(" FROM ");
while (fromOffset != -1) {            sb.insert((fromOffset+5), " ProdCustLink T2, ");            soFar =
sb.toString();            fromOffset = soFar.indexOf(" FROM ", (fromOffset+5));
        }        cachedFindCustomersByProductQueryString = sb.toString();    }    return
cachedFindCustomersByProductQueryString; }
```

After this method call, the query string looks something like the following:

```
SELECT <columns> FROM ProdCustLink T2, CUSTOMER T1    WHERE((T1.id = T2.Customer_id) AND
(T2.Product_id = ?))
```

Also in the finder object, the implemented finder uses the query string to create a PreparedStatement. Last but not least, the product ID value is added into each WHERE clause by using the superclass method getMergedWhereCount() in the iteration loop.

```
public java.sql.PreparedStatement    findCustomersByProduct(ProductKey inKey)    throws
java.lang.Exception {    // Get the full query string and make a PreparedStatement.
java.sql.PreparedStatement ps =        getPreparedStatement(getFindCustomersByProductQueryString());
// Inject the product id parameter into each merged WHERE clause.    for (int i = 0; i <
getMergedWhereCount(); i++) {        if (inKey != null)            ps.setInt(i+1, inKey.id);        else
ps.setNull(i+1, 4);    }    return ps; }
```

## 6.6.0.16: Dynamic fragment cache configuration

Dynamic fragment caching can be enabled and configured using two XML configuration files, or through WebSphere Application Server's administrative console. The articles in this section describe building the XML files, although all the values discussed (except where specifically documented otherwise) have identical counterparts in the administrative console.

To enable dynamic caching, you can build two XML configuration files:

- *[dynacache.xml](#)* - the global caching administration file
- *[servletcache.xml](#)* - the individual cache policy configuration file

Both XML files use UTF-8 (the 8-bit Universal Character Set transformation format) character encoding.



## 6.6.0.16.1: Global administration

You can enable and configure dynamic fragment caching through a XML configuration file or through the [administrative console](#).

### XML configuration file

The *dynacache.xml* file enables and configures dynamic fragment (also known as servlet) caching. If this file is not found in the [product\\_installation](#)\properties directory, the Web container processes servlets in the usual manner. However, if this file is found in the [product\\_installation](#)\properties directory, then dynamic fragment caching is enabled.

Within this file, users configure the overall operation of the cache, such as its size, and register the external caches that are used by the application.

**Note:** The *dynacache.xml* file is read only once, at WebSphere Application Server startup. So if changes are made to the file, WebSphere Application Server must be restarted for the changes to take effect.

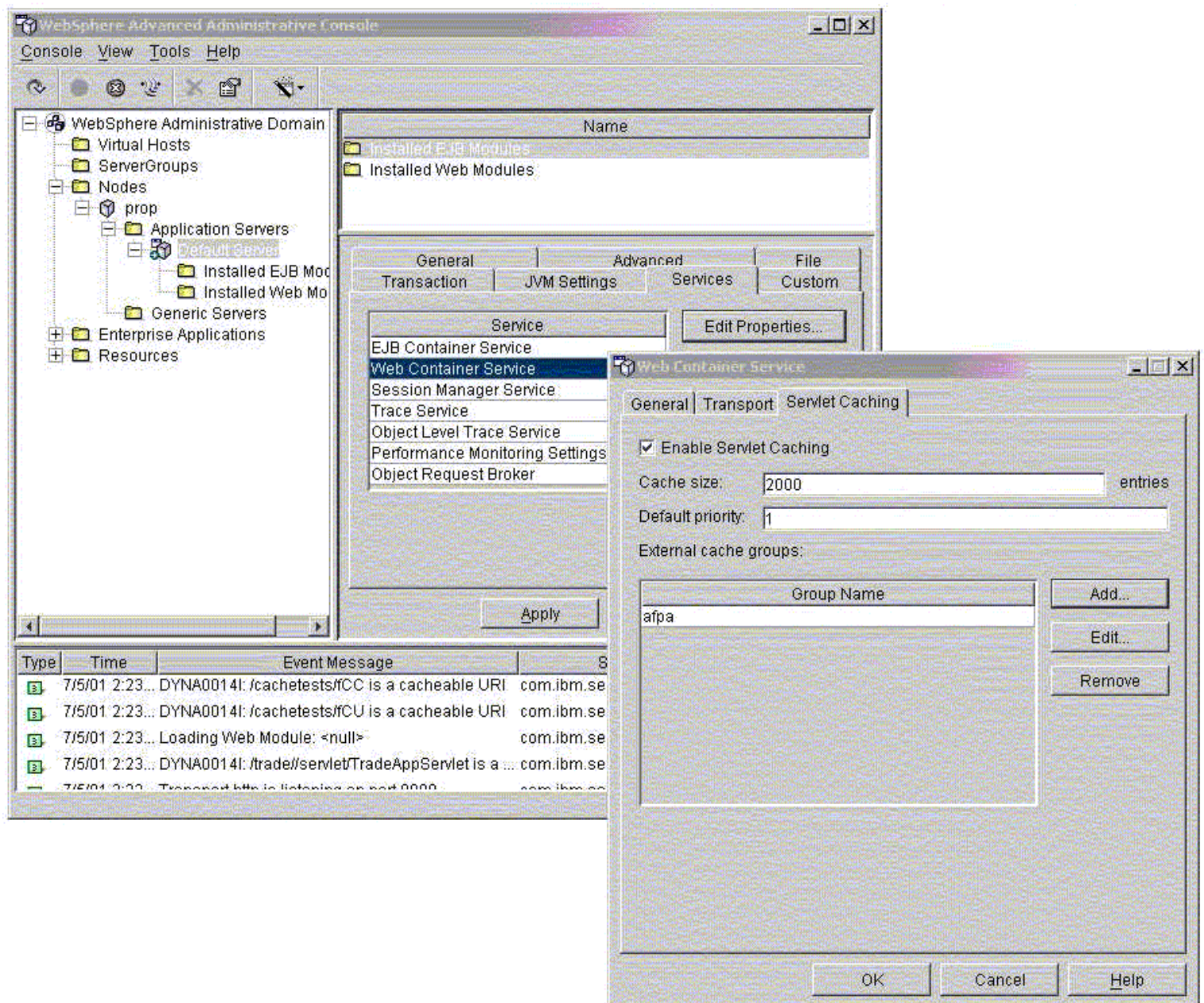
The DTD for the *dynacache.xml* file is specified in the *dynacache.dtd* file, which ships with WebSphere Application Server and is located in the [product\\_installation](#)\properties directory. The *dynacache.dtd* file defines the root element, <cacheUnit>, and should be included in the *dynacache.xml* file through the DOCTYPE declaration.

The beginning of the *dynacache.xml* should have the following processing instructions:

```
<?xml version="1.0"?><!DOCTYPE cacheUnit SYSTEM "dynacache.dtd">
```

### Console configuration

You can also enable dynamic fragment caching through the administrative console. Cache properties are located under a server's Web Container:



### Description of elements in the *dynacache.xml* file

The elements defined in the *dynacache.xml* file provide the following two categories of function:

- Configuring global cache operations
- Controlling external caches

## Configuring global cache operations

The root `<cacheUnit>` element contains one `<cache>` element, which provides global settings for all Application Servers on a node.

### Cache

```
<cache size="entries" priority="default_priority" />
```

*entries:* An integer defining the maximum number of entries the cache will hold. Values are usually in the thousands, with no set maximum or minimum

The size of the cache is limited not by the amount of available memory (for example, 20 megabytes), but by the number of distinct elements the available memory can hold. The cache only adds a few bytes of overhead for each entry (less than 32 bytes), so the size of an entry in memory is effectively equal to the amount of data being stored in that entry.

*default\_priority:* An integer that defines the default priority for cacheable servlets defined in `servletcache.xml`. The recommended value for *default\_priority* is one.

Priority is an extension of the Least Recently Used (LRU) caching algorithm. It represents the number of cycles through the LRU algorithm an entry is guaranteed to stay in the cache. See article [Dynamic fragment cache policy configuration](#) for more information on priority settings.

So, a cache storing at most 10,000 entries, giving all entries a default priority of one, would be declared as:

```
<cache size="10000" priority="1" />
```

## Controlling external caches

WebSphere Application Server can control external caches. You can define different groups of external caches, each with its own set of member caches. The interface between WebSphere Application Server and one of the member caches is an adapter bean, typically written by the vendor of the external cache. An administrator registers external cache groups within an `<externalCacheGroups>` element, which can then be used in cache policies. Each group is defined within a `<group>` element, which can contain `<member>` elements. See the [External cache adapter building](#) article for information on configuring WebSphere Application Server to control IBM Edge Server or IBM HTTP Server.

### Group

```
<group id="group_name" > ... </group>
```

*group\_name:* A string identifier for this group of external caches.

This name is used to specify the external group. For a servlet to be externally cacheable, it must be configured in the `servletcache.xml` file that is included in a group.

### Member

```
<member address="address" adapterBeanName="adapter_class" />
```

*address:* The hostname or IP address for this external cache.

This string will be passed to the adapter bean's `setAddress` method.

*adapter\_class:* The package and class name of the adapter for this external cache.

## Example of the *dynacache.xml* file

An entire `dynacache.xml` file based on the element definitions previously described might look like the following example:

```
<?xml version="1.0"?>

<!DOCTYPE cacheUnit SYSTEM "dynacache.dtd">

<cacheUnit>

<cache size="10000" priority="1" />

<externalCacheGroups>

<group id="edgeservers" type="shared">

<member address="edgeone" adapterBeanName="my.package.EdgeAdapter" />

<member address="edgetwo" adapterBeanName="other.package.OtherAdapter" />

</group>

</externalCacheGroups>

</cacheUnit>
```

## Quick Reference of the *dynacache.xml* file

1. `<?xml version="1.0"?>`
2. `<!DOCTYPE cacheUnit SYSTEM "dynacache.dtd">`
3. `<cacheUnit>`
4. `<cache size="number of elements" priority="default priority" />`
5. `<externalCacheGroups>`
6. `<group id="group name">`
7. `<member address="server name" adapterBeanName="some.package.Adapter"/>`
8. `</group>`
9. `</externalCacheGroups>`
10. `</cacheUnit>`

## 6.6.0.16.2: Policy configuration

Cache policies can be configured using the XML configuration file, *servletcache.xml*, or the Application Assembly Tool. This article describes building the XML file, which currently supports more caching features than the AAT. Unless stated otherwise, the XML elements have identical counterparts in the AAT. See the [Application Assembly Tool](#) documentation to learn how to build cache policies with the AAT.

### XML configuration

Administrators define the servlets to be cached inside the *servletcache.xml* file. The *servletcache.xml* file, like the *dynacache.xml* file, is located in the [product\\_installation](#)/properties directory.

The root element of this XML file is <servletCache>, which contains <servlet> elements. Within the <servlet> element, you specify parameters that:

1. Identify the servlets to be cached
2. Govern the creation of entry ids for servlets
3. Describe how entries generated from servlets are deleted from the cache

Typically you declare several <servlet> elements inside a *servletcache.xml* file.

Just like the *dynacache.xml* file, the *servletcache.xml* file is read only once, when WebSphere Application Server is first started. So if changes are made to the file, WebSphere Application Server must be restarted for the changes to take effect.

The DTD for the *servletcache.xml* file is specified in the *servletcache.dtd* file, which ships with WebSphere Application Server and is located in the [product\\_installation](#)\properties directory. The *servletcache.dtd* file defines the root element <servletCache>. The *servletcache.dtd* file is included in the *servletcache.xml* file through the DOCTYPE declaration.

The beginning of the *servletcache.xml* should have the following processing instructions:

```
<?xml version="1.0"?><!DOCTYPE servletCache SYSTEM "servletcache.dtd">
```

### Identifying What to Cache

The cache will parse the *servletcache.xml* file on startup, and extract from each <servlet> element a set of configuration parameters. Then, every time a new servlet is initialized, the cache attempts to match that servlet to each of the different servlet elements to find the configuration information for that servlet.

You can specify which servlets to cache in two ways:

1. By specifying the class name of the servlet, or
2. By using the servlet's full URI, beginning after the host name/IP address

To cache a servlet class, use the <servletimpl> tag.

#### Servletimpl

```
<servletimpl class="ClassName" />
```

*ClassName:* The class to be cached. This class must extend HttpServlet.

Whenever a servlet of the specified class is initialized, the dynamic caching function will match that servlet with the element configuration.

**Note:** This comparison is done by matching strings; subclasses of the specified



servlet implementation will not match this declaration.

Defining servlets by class name is ***not*** supported by the Application Assembly Tool.

Alternatively, you can specify a web path for your servlets. You do this with the `<path>` tag.

## Path

```
<path uri="web_path" />
```

*web\_path*: The full URI of the servlet, from the hostname to the CGI variable, but not including the CGI variable. The Web application's Web path must be a part of this parameter.

So if you access a cacheable servlet with the URL

*http://servername.ibm.com/webappname/servlet/MyServlet?arg1=1&....* then that servlet's path element should read `<path uri="/webappname/servlet/MyServlet" />`

You can specify different path elements referring to the same servlet. Also, the same path can appear in two different `<servlet>` elements, but when configuring a servlet invoked from that path, dynacache will use the configuration from the first valid `<servlet>` element that matches.

```
<servlet>
  <servletImpl class="CalcServlet" />
  :
  (other config info)
  :
</servlet>
```

This configuration will match any servlet of class **CalcServlet**. Also, if class **ScientificCalculatorServlet** extends **CalcServlet**, **ScientificCalculatorServlets** will not match this element

This example of a general setup will cache every servlet of class **CalcServlet** across the entire server, regardless of the URI

```
<servlet>
  <path uri="/tools/Calc" />
  <path uri="/tools/servlet/CalcServlet" />
  <path uri="/tools/Calculator.jsp" />
  :
  (other config info)
  :
</servlet>
```

This configuration yields slightly different behavior. In it we are working within a hypothetical tools Web application. The first path will catch any request for the `/tools/Calc` URI, regardless of that servlet's class. The second path will catch any calls to the **InvokerServlet** for the **CalcServlet** class. The third path allows you to cache on a JSP implementation of the calculator.

This example of a specific setup will cache the GUI defined servlet **Calc** (which may or may not be of the **CalcServlet** class), and will only cache **CalcServlets** if they are specifically invoked

## Removing entries from the cache

After you declare which servlets to cache, defining how to cache those servlets has two parts:

- Building unique entries for different requests, and
- Removing those entries at the appropriate time

You can do this using the *servletcache.xml* file, or by writing your own class to handle invalidation.

The cache removes entries in the following circumstances:

1. One of the cache's invalidation methods (see [com.ibm.websphere.servlet.cache.Cache](#) in Javadoc) was called directly, inside the servlet code
2. Running a servlet triggered a rule based invalidation

3. The timeout for the entry expired
4. The cache was full and a new entry replaced an old one

The first case is not relevant in this context, as it is done programmatically inside an application. The second case results from the "invalidate" attribute used with a cache variable, and third and fourth cases are configured using the <timeout> and <priority> tags.

## Timeout

<timeout seconds="*time\_in\_cache*" />

*time\_in\_cache*: The length of time, in seconds, after creation of an entry, that it should be removed from the cache. This value is required. If this value is zero or negative, the entry will not timeout, and can only be removed when the cache is full or programmatically from within an application.

If the <timeout> element is not present, then the <servlet> element in which it is contained will be invalidated, and will not be cached. When entries with a positive timeout are created, the timeout is added to the current time to determine when the entry will be invalidated. When that time is reached, if the entry remains in the cache, the cache will force its invalidation.

## Priority

<priority val="*priority*" />

*priority*: This value is a zero or positive integer, designating the length of time an entry will stay in the cache before becoming eligible for removal. If this element is not present, then the default value, defined in the dynacache.xml file, will be used.

Priority is an extension of the Least Recently Used caching algorithm. It represents the number of cycles through the LRU algorithm an entry is guaranteed to stay in the cache. On each cycle of the algorithm, the priority of an entry is decremented. Once it reaches zero, it is eligible to be invalidated. If an entry is requested while in the cache, its priority is reset to this value, keeping it in the cache longer. Therefore, a higher priority will provide a relatively longer availability for an entry, and more frequently requested entries will stay in the cache longer, with an entry's timeout as a hard cap.

In a high volume application where space in the cache is at a premium, designers should consider increasing the priority of a servlet or JSP if calculating its output is significantly harder than average, or if it will be executed much more often than average.

**Note:** Priority values should be kept low, as higher values will not yield a relative improvement, but will use extra LRU cycles. Declaring a servlet with priority 3 and another with priority 4 will generate the same invalidation behavior as servlets with priority 1 and 2, only marginally slower. When dealing with caches in the thousands of entries and greater, that slowdown becomes significant.

Use the timeout variable to guarantee the validity of an entry. Use priority to rank the relative importance of that entry. Giving all elements equal priority results in a standard LRU cache that increases performance significantly, but you can tailor the cache's operation to the application with careful use of the priority variable.

## Externalcache

<externalcache id="*group\_name*" />

*group\_name*: The name of an external cache group defined in the global configuration of the cache.

When this page is cached, a copy of it will now be pushed to this external cache group. See the [external cache](#) article for more information.

## IdGenerator

`<idgenerator class="class_name" />`

*class\_name*: The full package and class name of a class extending `com.ibm.servlet.dynacache.IdGenerator`.

See the "Custom Id and MetaData Generators" section for more information.

## MetaDataGenerator

`<metadatagenerator class="class_name" />`

*class\_name*: The full package and class name of a class extending `com.ibm.servlet.dynacache.MetaDataGenerator`.

See the "Custom Id and MetaData Generators" section for more information.

## Specifying how to cache

Each time a servlet is called and WebSphere Application Server generates a corresponding `HttpServletRequest` object, the cache uses information in that object to build an id string to represent the call.

A servlet's cache policy contains the rules that determine which pieces of information are used in that id string. It always contains certain default information, such as the URI of the requested fragment and its character encoding. The rules give extra information about what variables should be used in the id, and how they should be treated.

## Using cache variables

A cache variable is a generic term for a variable whose data should be used in caching a fragment. The four types of cache variables that correspond to the main sources of input for a fragment are:

- request parameters
- request attributes
- session attributes
- cookies

A client browser can set request parameters with CGI when submitting a form, while a servlet can set attributes on an `HttpServletRequest` object, then forward or include that request to another servlet. Finally, WebSphere Application Server can maintain session attributes that a servlet might want to access, and set cookies for later servlets to process.

In addition to building cache ids, these variables are used to control the following:

- grouping of cache entries
- the invalidation of other groups
- determining whether to cache a servlet based on that variable's value

The cache can decide whether or not to cache an invocation depending on the value of its input variables using the `<exclude>` tag.

Groups are handled by data ids, which consist of strings specified in the cache variable combined with the variable's value. Using the `data_id` attribute builds a group name and adds a servlet's cache entries to that group.

The `invalidate` attribute causes the eviction of a group from the cache. It is possible to have an entry that does no caching, and only invalidates groups. The `<invalidateonly>` tag is provided to save processing time for this case. The syntax for the various attributes follow.

## Request, Parameter, and Attribute

```
<request>
  <parameter id="parameter_name"
    data_id="group_identifier"
    invalidate="group_identifier"
    ignorevalue="true|false"
    required="true|false" >
    <exclude value="exclude_value"/>
  </parameter>
  <attribute id="attribute_name"
    method="method_name"
    data_id="group_identifier"
    invalidate="group_identifier"
    ignorevalue="true|false"
    required="true|false" >
    <exclude value="exclude_value"/>
  </attribute>
</request>
```

*parameter\_name*: The name of a request parameter, the value of which will be used to generate cache entry ids.

*attribute\_name*: The name of a request attribute. The output of one or more of this object's methods will be used to generate cache entry ids.

*method*: The name of a method in this request attribute. The output of this method will be used to generate cache entry ids. If this value is not specified "toString" is assumed. This method may not take any arguments, and parenthesis should not be included in the method.

*group\_identifier*: A string that, when combined with the value of this request variable, generates a group name for this cache entry. If used by a "data\_id" attribute, this cache entry will be placed in that group. If used in an "invalidate" entry, then that group will be invalidated, and this entry will then be placed in the cache. If this variable is not present, the group id will not be used, and neither action will occur.

*ignorevalue*: Indicates whether this variable's value is relevant to the cache id, or whether only the variable's presence is important. If true, the cache id will reflect that the variable was present, but its value will not be used. This value defaults to false when not set.

*required*: Indicates whether this value must be present in the request. If this is set to true and either the parameter/attribute is not present or (when defining an attribute) does not contain the specified method, then this request will not be cached. This value defaults to false when not set.

You can define unlimited parameter and attribute objects within a <request> element, but you should only define one <request> element per <servlet> element.

The cache's handling of session attributes is identical to its handling of request attributes.

## Session

```
<session id="session_attribute_name"
  method="method_name"
  data_id="group_identifier"
  invalidate="group_identifier"
  ignorevalue="true|false"
  required="true|false" >
  <exclude value="exclude_value"/>
</session>
```

*session\_attribute\_name*: The name of a session attribute, the value of which will be used to

generate cache entry ids.

*method*: The name of a method in this session attribute. The output of this method is used to generate cache entry ids. If this value is not specified "toString" is assumed. This method may not take any arguments, and parenthesis should not be included in the method.

*group\_identifier*: A string that, when combined with the value of this request variable, generates a group name for this cache entry. If used by a "data\_id" attribute, this cache entry will be placed in that group. If used in an "invalidate" entry, then that group will be invalidated, and this entry will then be placed in the cache. If this variable is not present, the group id will not be used, and neither action will occur.

*ignorevalue*: Indicates whether this variable's value is relevant to the cache id, or if only the variable's presence is important. If true, the cache id will reflect that the variable was present, but its value will not be used. This value defaults to false when not set.

*required*: Indicates whether this value must be present in the session. If this is set to true and either the session parameter is not present or does not contain the specified method, then this request is not cached. This value defaults to false when not set.

As with request parameters/attributes, there can be any number of session attributes declared in a servlet.

### Invalidateonly

```
<invalidateonly />
```

When this tag is used, no caching is performed for this servlet, though invalidations triggered by it will take effect.

### Exclude

The <exclude/> tag attaches to a cache variable and is used to keep a servlet from being cached when that variable is present on the request. It can only be applied for certain values, or for all values of a cache variable. When the exclude tag is used without its "value" attribute, no caching is performed for this servlet when the variable is present. Alternatively, users can repeat the <exclude> tag with the "value" attribute defined several times to specify a set of values that keep the servlet from being cached. This helps prevent caching of control servlets.

*exclude\_value*: When the cache variable is present on a request and its value is equal to this, the servlet is not cached.

For example, the following setup indicates that whenever the request parameter "nocache" is present, the servlet should not be cached:

```
<servlet>
  <path uri="/myServlet" />
  <timeout seconds="-1" />
  <request>
    <parameter id="nocache" >
      <exclude/>
    </parameter>
  </request>
</servlet>
```

In this example, the servlet will be cached unless the request parameter "cache" is present and its value equals "no" or "false":

```
<servlet>
  <path uri="/myServlet" />
  <timeout seconds="-1" />
  <request>
```

```

        <parameter id="cache" >
            <exclude value="no"/>
            <exclude value="false"/>
        </parameter>
    </request>
</servlet>

```

Exclude tags can be applied to any cache variable.

### Notes:

Invalidating has a higher performance cost than caching. Avoid using the same variable to invalidate a group and to group an entry. You will see less performance benefit than when just using the variable to define a group id.

Data\_id and invalidate tags that are within the same <servlet> element should have different values, usually. If they have the same value, the group is invalidated, and the entry for the current servlet is put into that group. The invalidate tag that corresponds to a data\_id will occur in different <servlet> elements.

## Quick Reference of the *servletcache.xml* file

1. <?xml version="1.0"?>
2. <!DOCTYPE cacheUnit SYSTEM "dynacache.dtd">
3. <servletCache>
4. <servlet>
5. <invalidateonly/>

```
<servletimpl class="some.package.SomeClass"/>
```

6. <timeout seconds= "*timeout value*" />
- <priority value="*priority value*" />
7. <externalcache id="external cache group name">
8. OR
9. <request>
10. <parameter id="*parameter name*"

```

        <exclude value="exclude value"/>
    </parameter>

```

11. <attribute id= "*attribute name*"

```

    OR  <path uri="MyServlet" />
    <path uri="servlet/SomeClass"/>

```

```

    OR  <metadatagenerator
class="package.GeneratorClass" />

```

```

data_id="group identifier 1"
invalidate="group identifier 2"
ignorevalue="true|false"
required="true|false" >

```

```

method= "aMethodName"
data_id="group identifier 1"
invalidate="group identifier 2"
ignorevalue="true|false"
required="true|false" />

```

12. `</request>`

13. `<session id="session attribute name"`

```
method="aMethodName"  
data_id="group identifier 1"  
invalidate="group identifier 2"  
ignorevalue="true|false"  
required="true|false" />
```

14. `cookie id="cookie name"`

```
method="aMethodName"  
data_id="group identifier 1"  
invalidate="group identifier 2"  
ignorevalue="true|false"  
required="true|false" />
```

15. `<idgenerator class="package.IdGeneratorClass" />`

16. `<metadatagenerator class="package.MetaDataGeneratorClass"/>`

17. `</servlet>`

18. `</servletCache>`

#### Notes:

Line 6: Timeout < 1 implies the value will not time out. Required.

Line 10: The method, called from the request parameter, defaults to `toString`, and `required` defaults to `false`.

Line 10: The `exclude` tag can be used in any of the four cache variable types.

Line 10: The method called on the request attribute defaults to `toString`, and `required` defaults to `false`.

Line 13: The method called on the session attribute defaults to `toString`, and `required` defaults to `false`.

Line 14: The method called on the cookie defaults to `toString`, and `required` defaults to `false`.

Line 8: OR applies to lines 10-15

## 6.6.0.16.3: Dynamic fragment cache XML examples

In this example, Calculator Servlet is defined inside a 'tools' Web application. It takes an operation 'operation' and two arguments, 'arg1' and 'arg2'. The values for these variables are received in the query string from an external user, that is, request parameters from a browser or applet. You invoke the servlet to calculate 2+3 to get the answer 5, with the URI:

```
/tools/Calc?arg1="2"&arg2="3"&operation="+"
```

To cache the output of this servlet, you distinguish results using the request parameters, so that 2,3,+ has a different cache entry than 4,5,\*. For this example, you would define the following <servlet> element:

```
<servlet>
  <path uri="/tools/Calc" />
  <request>
    <parameter id="arg1"      required="true" />
    <parameter id="arg2"      required="true" />
    <parameter id="operation" required="true" />
  </request>
  <timeout seconds="-1" />
</servlet>
```

In a second example, the news servlet of class CoastalNewsServlet displays either west coast news or east coast news, depending on a user's location. This servlet has a session object named 'location' of class 'LocationBean' with a method getCoast() that returns "east" or "west". If this object is not present on the session, then the servlet returns the news for both coasts, which you also want to cache. So whether or not the location is present on the session, you want to cache the output of the servlet, and you do not want the entries to time out:

```
<servlet>
  <servletimpl class="CoastalNewsServlet" />
  <session id="location" method="getCoast" />
  <timeout seconds="-1" />
</servlet>
```

To group cache entries based on the coast, or to invalidate the cache entries for a coast whenever the location bean gets updated by a control servlet, you must consider the design of the application.

If a variable is not available to a servlet at execution (that is, the request/session variable has not been set), then, even if the servlet is cacheable, no group id will be generated based on that variable. In the CoastalNewsServlet example, a missing session parameter causes the servlet to display the news for both coasts. Naturally, in this case, you want the cache entry to belong to both the east and west coast groups. However, because the cache does not handle data ids when variables are missing, this is not possible.

The simplest solution to this problem marks the location bean as a required variable for caching. This means the output of the news servlet will not be cached if location is undefined. Therefore, you can now do all the grouping knowing that the location bean will be present to place entries into the correct groups. Servletcache.xml needs two sets of changes to finish configuring groups:

- the CoastalNewsServlet entry must be modified to put entries into groups, and
- a new entry for the control servlet that updates the location bean must be added to allow invalidation of these groups.

```
<servlet>
  <servletimpl class="CoastalNewsServlet" />
  <session id="location" method="getCoast" data_id="coast" required="true" />
  <timeout seconds="-1" />
</servlet>
```

```
<servlet>
  <invalidateonly/>
  <servletimpl class="LocationUpdateServlet" />
  <request>
    <attribute id="new_coast" invalidate="coast" />
  </request>
</servlet>
```

Now, when the news servlet is invoked, the cache will take the data\_id "coast" and append "=" and the value of location.getCoast() to create a group name to identify that entry. In the update servlet, a new\_coast string is expected as a request attribute, and will be used in the same fashion to build group names for removal from the cache.

## Using HttpSession and request attributes

The dynamic cache can use objects stored in an HttpSession or request attribute in caching requests. The Servlet specification allows the Servlet or JSP programmer to put any number of objects into the session or request, and index these values with a String key name. Possible uses range from the storage of minimal data and simple types (for example, String, Integer, Boolean) to very complex and large (Vectors and Hashtables of complex User Objects).

Given this background, this is what the WebSphere Application Server Servlet/JSP cache supports:

- When building cache policies the user can specify:
  1. A key name for a session/request attribute value and



2. An [optional] method name used for retrieving a value from the stored object.
- In the case where the method name is not specified, the default method toString() is used to transform the object into a String for use in the cache id.

Consider the following object:

```
class User {    public String getName() {...};           public String getPrimaryGroup() {...};    }
```

If an instance of this object is stored in the session using the key "user", you might specify the following cache policy:(in this example, imagine that different user groups view a different home page):

```
<servlet>
    <path uri="/index.jsp" />
    <session id="user" method="getPrimaryGroup" />
</servlet>
```

## Caching personalized pages

Let's say we want a fragment of a page to be a stock list owned by an user. The fragment has two servlets: the first obtains the stock list from the database and forwards the list of stock symbols to the second servlet, which gets quotes for the stocks from the back end. How can we cache this fragment?

Depending on exactly how this stock list is generated, caching will have varying effectiveness, but it will provide a benefit. The answer comes from applying a key concept in servlet caching. The biggest performance wins come from caching servlets that obtain information from outside WebSphere Application Server. This means that while caching a simple presentation JSP file will give moderate performance gains, caching servlets that request information from Enterprise Java Beans or databases, saves WebSphere Application Server processing power and decreases load on the back end.

In this example, both sets of actions can be cached (since they are both reads from the database). The first servlet is a classic example of a cacheable servlet. It is a simple database lookup, using one value as input (a user id), and producing the list as output. The interesting case (an example of designing an application with caching in mind) is how the servlets go from a list of symbols to actually looking up the quotes.

The simple design involves one servlet that does all the work and a presentation JSP. The servlet gets a list of stock symbols for a user from the database (presumably that user's database record holds the symbols in their portfolio), then immediately goes back to the database and looks up current quotes for those symbols. It builds a list of quotes and forwards them to a presentation JSP.

This design can be cached. You base everything off the user id, and cache the current quotes for that user. There are missed opportunities here, though. First, since everything is based off of an individual user, you cannot reuse the stock quotes gathered for other users who own the same stocks. The usefulness of this cache entry is limited to one user. Second, whenever one of the quotes, or the list of symbols changes, the whole page changes. This entry is unstable, and will not be correct very long.

A better design fragments the data requests into 3 different servlets/JSP files, separating each of the two database reads into its own servlet/JSP file. The first servlet would use the user id to get the list of symbols. It would forward the list to a presentation JSP that would format the list, and get individual quotes from a third servlet that takes a symbol as input.

All three of these servlets/JSP files can be cached individually. The first servlet is caching the user specific list of symbols only. Changes to a user's portfolio will not affect the cache entries for the stock quotes. The presentation JSP file only changes the list of symbols supplied to it. Caching in this example is as useful as caching the first servlet, and could be even more useful if different users have the same list of stocks. The quotes are cached with the last servlet, which will have a different entry for each stock. If one stock quote changes, then the request for that individual stock will have to be reinvoked from the application server. The rest of the requests can all be served from the cache. Most importantly, these quotes are reusable in any request for a customer's portfolio. By narrowing the responsibility of individual pieces of the application, you can provide selective caching, and achieve bigger performance gains.

## 6.6.0.16.4: Dynamic fragment caching monitor

WebSphere Application Server provides the Servlet Cache Monitor application for inspecting the contents and behavior of the fragment cache. To use the Servlet Cache Monitor, install the *ServletCacheMonitor.ear* file, located in the [product\\_installation/installableApps](#) directory, on each application server that uses the dynamic fragment cache function.

After the *ServletCacheMonitor.ear* file is installed, access the Servlet Cache Monitor through the `/servletcache` URI.

To monitor different application servers on the same node, modify the *ServletCacheMonitor.ear* file's Web path using the [Application Assembly Tool](#). This allows you to access the different servers' cache monitors with different URIs.

### Cache statistics

Cache Statistics	
Cache Size	10
Used Entries	10
Cache Hits	72
Cache Misses	13
LRU Evictions	2
Explicit Removals	1
Default Priority	1

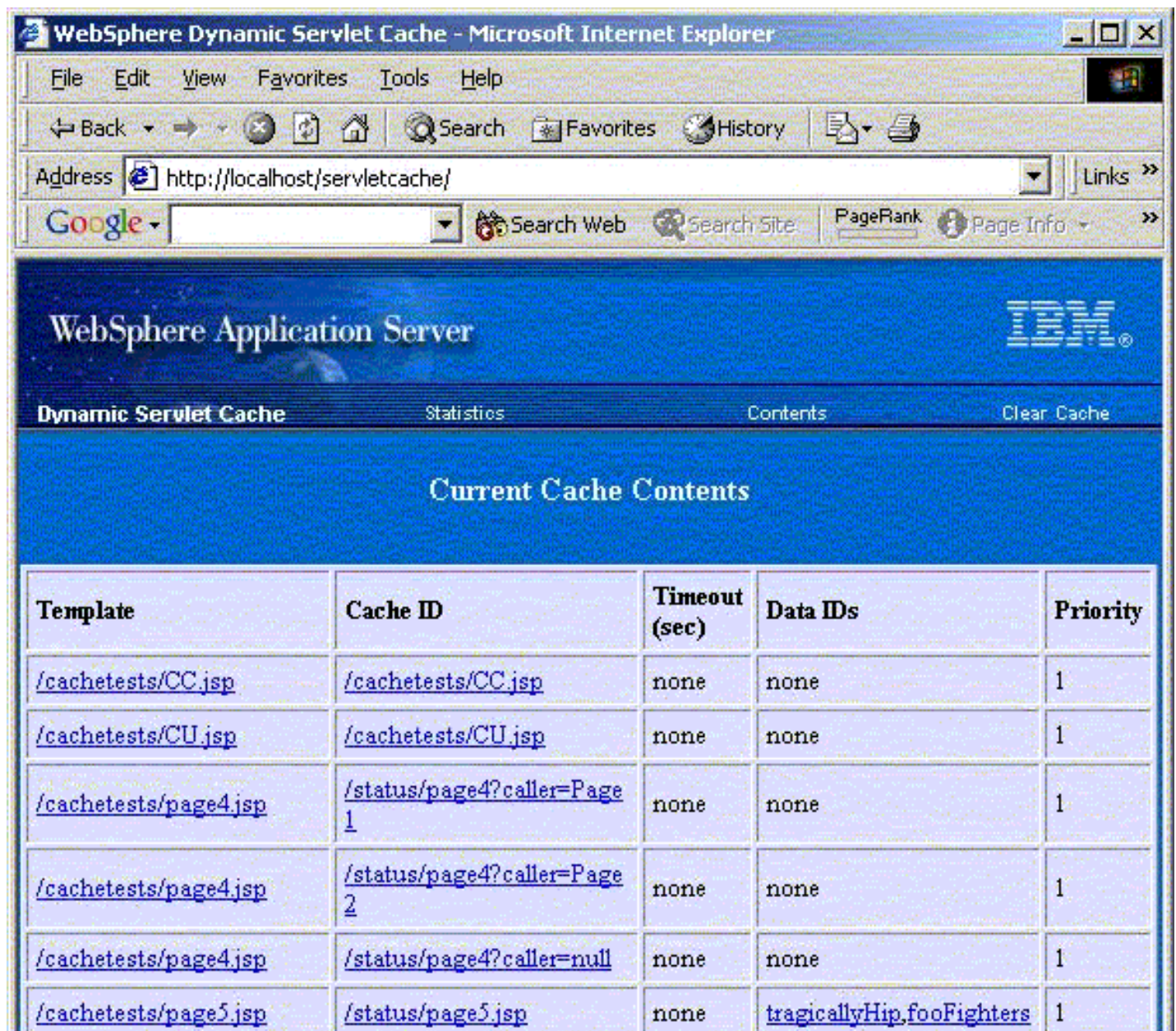


The statistics page is the main page of the monitor and describes the following properties:

- Cache Size - The maximum number of cache entries as defined in the global properties of the fragment cache.
- Used Entries - The number of entries currently contained in the cache
- Cache Hits - The number of servlet/JSP requests, both external (from a browser) and internal (included/forwarded from another servlet/JSP) that have been served from the cache since startup.
- Cache Misses - The number of requests for cacheable servlets or JSPs, that were not served from the cache, i.e. resulted in executing the fragment and storing the result in the cache.
- LRU Evictions - The number of entries that have been removed from the cache by the LRU algorithm when the cache was full and new entries needed to be added.
- Explicit Removals - The number of entries removed from the cache through cache policy rules, or through the cache monitor.
- Default Priority - The default value for a cache entry's priority as defined in the global properties of the fragment cache. See the Cache Policies section of the fragment cache documentation for more information about priority.

## Cache contents

The contents of the fragment cache can be viewed in the "Cache contents view" panel. A list of all entries, an individual entry, or lists of entries with identical group ids (also referred to as data ids) are available.



Template	Cache ID	Timeout (sec)	Data IDs	Priority
<a href="#">/cachetests/CC.jsp</a>	<a href="#">/cachetests/CC.jsp</a>	none	none	1
<a href="#">/cachetests/CU.jsp</a>	<a href="#">/cachetests/CU.jsp</a>	none	none	1
<a href="#">/cachetests/page4.jsp</a>	<a href="#">/status/page4?caller=Page 1</a>	none	none	1
<a href="#">/cachetests/page4.jsp</a>	<a href="#">/status/page4?caller=Page 2</a>	none	none	1
<a href="#">/cachetests/page4.jsp</a>	<a href="#">/status/page4?caller=null</a>	none	none	1
<a href="#">/cachetests/page5.jsp</a>	<a href="#">/status/page5.jsp</a>	none	<a href="#">tragicallyHip.fooFighters</a>	1



<a href="#">/cachetests/page6.jsp</a>	<a href="#">/status/page6.jsp</a>	none	<a href="#">tragicallyHip</a>	1
<a href="#">/cachetests/page7.jsp</a>	<a href="#">/status/page7.jsp</a>	none	<a href="#">fooFighters</a>	1
<a href="#">/cachetests/TimeStamp.jsp</a>	<a href="#">/cachetests/TimeStamp.jsp</a>	none	none	1
<a href="#">/cachetests/TimeStamp.jsp</a>	<a href="#">/cachetests/page2.jsp</a>	none	none	1

For each entry you can see several of its properties, including template, cache id, and data id. Clicking a template or data id displays a list of entries with the same template/data id.

**WebSphere Application Server**

**Dynamic Servlet Cache**    Statistics    Contents    Clear Cache

You are viewing template [/cachetests/page4.jsp](#)

[Return](#)    [Invalidate](#)

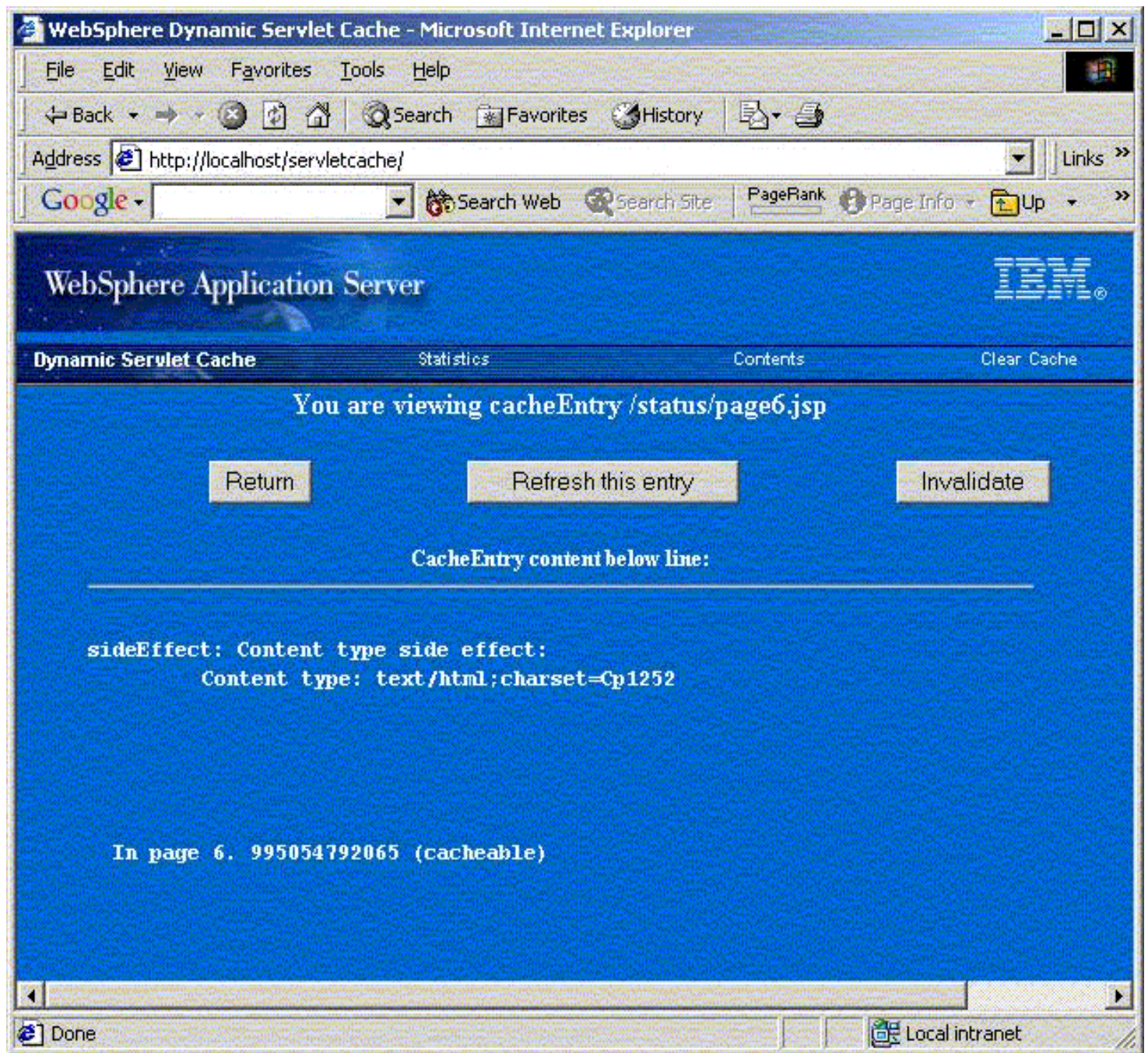
Fragments for template [/cachetests/page4.jsp](#)

cacheEntry id	timeLimit	dataIds	priority
<a href="#">/status/page4?caller=null</a>	none	none	1
<a href="#">/status/page4?caller=Page 1</a>	none	none	1
<a href="#">/status/page4?caller=Page 2</a>	none	none	1

Fri Jul 13 16:28:59 EDT 2001

Clicking a cache id in any of the menus in the "Cache list of entries" view, displays a detailed description of the entry. From this panel, you can invalidate the entry, reset its position in the LRU algorithm (mimicking a cache hit on that entry), or return to the main list of entries.





## Clear the cache

This removes every entry currently in the cache.

## 6.6.0.17: DrAdmin command reference

Use the following command program to perform various tasks in *Advanced Single ServerEdition*, such as starting and stopping servers, applications, and modules.

`product_installation_root/bin/DrAdmin`

For complete DrAdmin usage, enter `DrAdmin -usage`. For comprehensive DrAdmin help, enter `DrAdmin -help`. Here are some common commands:

- To restart an application:
  - `DrAdmin -serverPort DrAdmin port number -restart -application ear file name`
- To stop a running application:
  - `DrAdmin -serverPort DrAdmin port number -stop -application ear file name`
- To start a stopped application:
  - `DrAdmin -serverPort DrAdmin port number -start -application ear file name`
- To list the status of an application
  - `DrAdmin -serverPort DrAdmin port number -long -application ear file name`
- To restart a module within an application:
  - `DrAdmin -serverPort DrAdmin port number -restart -application ear file name -module module file name`
- To stop a module within an application:
  - `DrAdmin -serverPort DrAdmin port number -stop -application ear file name -module module file name`
- To start a module within an application:
  - `DrAdmin -serverPort DrAdmin port number -start -application ear file name -module module file name`

## 6.6.1: Administering applications (overview)

Administration of applications consists of the following:

- Assembling the modules of the application, setting deployment descriptor properties, and generating code for deployment using the Application Assembly Tool
- Performing additional configuration tasks using the console

### Classpath considerations

An important classpath-related setting to note is the Module Visibility. This application server setting impacts the portability of applications and standalone modules from other WebSphere Application Server versions and editions. If your existing module does not run as-is when you transfer it to Version 4.0, you might need to reassemble an existing module or change the module visibility setting.

See [the information on setting classpaths](#) for a full discussion of classpath considerations. See the [applicationserver property reference](#) for information about the module visibility setting.

# Enterprise application properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

## Archive URL



The path to the EAR file for the application

## Context Root



The context root of the Web application contained in this enterprise application.

The context root is combined with the defined servlet mapping (from the WAR file) to compose the full URL that users type to access the servlet. For example, if the context root is /gettingstarted and the servlet mapping is MySession, then the URL is http://host:port/gettingstarted/MySession.

## EnterpriseApp Name



or Name



A logical name for the application. The name will be displayed in administration and configuration tools.

## Execution State



*For Advanced Edition:* The current state of the application, such as running or stopped

*For Advanced Single Server Edition:* The execution state that you would like the application to have, the next time you start the product

## Node



The administrative node with which the application is associated.

## Path, or Remote Path on Server



The fully qualified path to the .ear, .jar, or .war file for which you are configuring an enterpriseapplication

Use the first set of fields if the console and application files are on the same machine (whether or not the server is on that machine, too).

Use the second set of fields (including **Remote Path...**) if the application files already reside on the server machine, and the console is being run from a remote machine.

During application installation, application files are typically uploaded from a client machine containing the administrative console to the server machine, where they are deployed. In such cases, the Web browser running the administrativeconsole is used to select EAR, WAR, or JAR modules to upload to the servermachine.

In some cases, however, the application files will already reside on the file system of the machine running the application server. To have the application server install these files, (bypassing the upload stage), use the Remote Path option.



An example of the field value on Windows NT is C:\WebSphere\AppServer\installableApps\test.ear. You can optionally enter the application name to use for standalone modules (JAR, WAR) and the context root (used only for Web modules).

### **User/Role Mappings, Run As Mappings, Roles, Users, and Groups**



See the properties for mapping security roles and "run as" roles to users and groups. This applies to *Advanced Edition*, butnot to *Advanced Single Server Edition*.

## 6.6.1.0.1.a: Assembly properties for security roles (application)

A security role is a logical grouping of principals. Access to operations (such as EJB methods) is controlled by granting access to a role.

### **Role Name (Required, String)**

Specifies the name of a security role that is unique to the application.

### **Description**

Contains a description of the security role.

### **Binding -- Groups -- Name**

Specifies user groups that are granted the security role.

### **Binding -- Users -- Name**

Specifies users that are granted the security role.

### **Binding -- Special Subjects -- Name**

Specifies one of two special categories of users to which roles can be granted: Everyone or All authenticated users. If the special subject Everyone is granted a role, then all users, including those who did not authenticate, are granted the role. In other words, a method on an enterprise bean or a URI is unprotected if any of the required roles for that method are granted to the special subject Everyone. In the case of All authenticated users, any user who can authenticate by using a valid user ID and password is considered to be granted that role.

## 6.6.1.0.aa: Assembly properties for enterprise applications

### **File name**

Specifies the file name of the application.

### **Alternative DD**

Specifies the file name for an alternative deployment descriptor file to use instead of the original deployment descriptor file in the application's JAR file. This file is the postassembly version of the deployment descriptor file. (The original deployment descriptor file can be edited to resolve dependencies and security information. Directing the use of the alternative deployment descriptor allows you to keep the original deployment descriptor file intact). The value of the Alternative DD property must be the full path name of the deployment descriptor file relative to the application's root directory. By convention, the file is in the ALT-INF directory. If this property is not specified, the deployment descriptor file is read directly from the module's JAR file.

### **Display name (Required, String)**

Specifies a short name intended to be displayed by GUIs. It is used to identify the application at deployment time.

### **Small icon**

Specifies a JPEG or GIF file containing a small image (16x16 pixels). The image is used as an icon to represent the application in a GUI.

### **Large icon**

Specifies a JPEG or GIF file containing a large image (32x32 pixels). The image is used as an icon to represent the application in a GUI.

### **Description**

Contains a description of the application.

### **Reload interval**

Specifies the time interval, in seconds, at which the EJB modules are to be reloaded.

### **Enterprise application name**

Specifies the global JNDI name of the application.

### **Run-as bindings - Authorization Data - User ID**

Specifies the security principal to be used for the execution of the application's enterprise bean methods. This security principal is granted the named role.

### **Run-as bindings - Authorization Data - Password**

Specifies the password of the security principal.

### **Run-as bindings - Role name**

Specifies a security role name. This name appears as the value of the Use Identity Assigned to Specified Role property of the enterprise bean.

## 6.6.1.3: Administering applications with the Web console

Use the Web console to install valid applications (.ear files) and module files (.war and .jar files) into the application server runtime and manage their configurations. If you install a module, it will be wrapped in an .ear file for you.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Enterprise Applications**

## 6.6.1.3.1: Installing applications with the Web console

During this task, you will install the application files (.ear, .jar, and .war) into the product directory structure.

**Important:** Users and groups defined in the application EAR file will be removed during application installation if they do not exist in the specified system registry.

To install an application:

1. Expand the tree on the left side of the console to locate **Nodes** -> *hostname* -> **Enterprise Applications**.
2. Click **Enterprise Applications**.

The right side of the console will display the list of zero or more installed applications (as .ear files).

3. Click the **Install** button displayed on the right side of the console, above the list of installed applications.
4. Specify the location of the application or module, referring to the [application property reference](#) as needed to fill in the field values.
  - Use the first set of fields if the console and application files are on the same machine (whether or not the server is on that machine, too)
  - Use the second set of fields (including **Remote Path...**) if the application files already reside on the same machine as the server, and the console is being run from a remote machine.

See the field help for further discussion of the remote file path.

5. Click **Next**.
6. Follow the instructions on the resulting task wizard. Depending on the components in your application, various panels will be displayed:
  - [Modify Role to User Mapping](#)  
(all applications)
  - [Modify EJB Run as Role to User Mapping](#)  
(applications containing EJB modules with one or more entity beans that use the IBM deployment descriptor extension for Run As Settings, Run As Mode:Run As Specified Identity)
  - [Modify EJB to JNDI Name Mapping](#)  
(applications containing EJB modules)
  - [Modify EJB Reference to JNDI Name Mapping](#)  
(applications containing EJB modules with EJB references in their deployment descriptors)
  - [Modify Resource Reference to JNDI Name Mapping](#)  
(applications containing Web modules with Resource references in their deployment descriptors)
  - [Specify Virtual Host Mapping](#)  
(applications containing Web modules) -- includes JSP precompilation option
  - [Specify CMP Data Source Binding](#)  
(applications containing EJB modules with container managed entity beans)

As you complete the wizard and wait for the application to be installed, please be patient if the application you are installing contains EJB modules for which code must be generated for deployment. This step can take a while.

7. Verify that the new application is displayed in the tree on the left side of the console. It should be located at **Nodes** -> *hostname* -> **Enterprise Applications** -> *application\_name* where *application\_name* is its name.

8. [Save your configuration.](#)
9. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

If you installed an application while the server was running, please note that the newly installed application and its modules can be viewed in the list of installed Enterprise Applications, from which applications and their modules can be [started, stopped, and restarted](#). However, the application and modules will remain in a "cannot be run" state until the server is stopped and started again.

## 6.6.1.3.1a: Mapping roles to users with the Web console

This task is a sub-task of the application installation wizard. You might also perform it to modify an application that is already installed. During this task, you will map existing roles to users. It is very important that all roles in the application be assigned to subjects (users, groups, "All Authenticated," or "Everyone") prior to starting the application in the application server.

New roles must be added at application assembly time (prior to installation). If you are installing this application for the first time, you might see user or group names already assigned to the roles on this page. To assist you, these users and groups are provided as Role to Subject "hints" by the application assembler (using the application assembly tool).

1. Display the **Mapping Roles to Users** panel. If you are using the installation wizard to install an application, the panel will be displayed already.

To display the panel for modifying an already installed application:

1. Expand the tree on the left side of the console to display **Nodes** -> *hostname* -> **Enterprise Applications** -> *application\_name*.
  2. In the property sheet for the application, locate the **Modify Application Bindings** list.
  3. Click the **Mapping Roles to Users** task.
2. For each role listed on the panel, specify values for the users, groups, and special subjects.

When entering multiple user or group names in the text field, delimit each name with a comma.

For the special subjects for each role, select "Everyone" to allow any user to perform actions matching the specified role. Select "All Authenticated" to allow any user who has already been authenticated with the underlying security system to perform the actions associated with the specified role.

3. Click **Next** to display the confirmation page.
4. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.1.3.1b: Mapping EJB "Run As" roles to users with the Web console

This task is a sub-task of the application installation wizard. You might also perform it to modify an application that is already installed. New Run As mappings must be added before this stage -- before installing the application.

1. Display the **Mapping EJB "Run As" roles** panel. If you are using the installation wizard to install an application, the panel will be displayed already.

To display the panel for modifying an already installed application:

1. Expand the tree on the left side of the console to display **Nodes** -> *hostname* -> **Enterprise Applications** -> *application\_name*.
2. In the property sheet for the application, locate the **Modify Application Bindings** list.
3. Click the **Mapping EJB "Run AS" Roles to Users** task.
2. For each EJB module listed on the panel, specify a Run As role.
3. Click **Next** to display the confirmation page.
4. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).



## 6.6.1.3.1c: Mapping virtual hosts to Web modules with the Web console

This task is a sub-task of the application installation wizard. You might also perform it to modify an application that is already installed. You will not need to perform this task if the application does not contain Web modules.

1. Display the **Mapping Virtual Hosts to Web Modules** panel. If you are using the installation wizard to install an application, the panel will be displayed already.

To display the panel for modifying an already installed application:

1. Expand the tree on the left side of the console to display **Nodes** -> *hostname* -> **Enterprise Applications** -> *application\_name*.
  2. In the property sheet for the application, locate the **Modify Application Bindings** list.
  3. Click the **Mapping Virtual Hosts to Web Modules** task.
2. For each Web module listed on the panel, specify a virtual host.

The list contains existing virtual hosts. If the list is empty, or to create a new virtual host, see the **Virtual Hosts** entry in the tree on the left side of the console.
  3. The virtual host panel will also contain a choice about [precompiled JSP options](#) for the Web modules. Selecting Yes will precompile JSP files for the Web module while installing the application or saving changes to the application (whichever you happen to be doing). The precompiling will begin when you click **Finish** to end the wizard.
  4. Click **Next** to display the confirmation page.
  5. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
  6. [Save your configuration](#).
  7. (Optional) To have the configuration take effect:
    1. [Stop the server](#)
    2. [Start the server again](#).

## 6.6.1.3.1d: Modify the EJB to JNDI name mapping with the Web console

This task is a sub-task of the application installation wizard. During this task, you will map existing enterprise beans to JNDI names. New enterprise bean modules (EJB JAR files) must be added or removed using the application assembly tool prior to installing the application.

1. Display the **Modify the EJB to JNDI Name Mapping** panel. If you are using the installation wizard to install an application, the panel will be displayed already.
2. For each EJB JAR file listed on the panel, specify a JNDI name that the application server should use when binding the enterprise bean into the JNDI naming context.
3. Click **Next** to display the confirmation page.
4. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.1.3.1e: Modify the EJB Reference to JNDI name mapping with the Web console

This task is a sub-task of the application installation wizard. During this task, you will map existing EJB references to JNDI names. EJB References must be added or removed using the application assembly tool (prior to installing the application).

It is very important that all EJB References have JNDI names specified prior to running the application in the application server.

1. Display the **Modify the EJB Reference to JNDI Name Mapping** panel. If you are using the installation wizard to install an application, the panel will be displayed already.
2. For each EJB Reference listed on the panel, specify a JNDI name.

The application must be able to resolve the JNDI names to actual enterprise beans that have been (or will be) bound into the naming context of the application server. The JNDI name can identify an EJB that either resides in the same application or a different application.

3. Click **Next** to display the confirmation page.
4. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.1.3.1f: Modify the Resource reference to JNDI name mapping with the Web console

This task is a sub-task of the application installation wizard. During this task, you will map existing Resource references to JNDI names. Resource References must be added or removed using the application assembly tool (prior to installing the application).

It is important for all Resource References to have JNDI names specified prior to running the application in the application server.

1. Display the **Modify the Resource Reference to JNDI Name Mapping** panel. If you are using the installation wizard to install an application, the panel will be displayed already.
2. For each Resource Reference listed on the panel, specify a JNDI name.

These JNDI names must resolve to actual resource factories (Resources) which have been (or will be) bound into the application server environment. The JNDI names for Resource References enable the application to locate a resource factory (Data Source, Mail Session, JMS Connection Factory or Destination, URL, and so on) that has been (or will be) bound into the JNDI naming context by the application server.

If you have not yet configured a resource prior to installing the application, you can still specify the JNDI name of the resource, and then configure a new resource having the specified JNDI name at a later time using the Web console.

3. Click **Next** to display the confirmation page.
4. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.1.3.1g: Specify CMP data source bindings with the Web console

This task is a sub-task of the application installation wizard. During this task, you will specify data source bindings for existing CMP beans. EJB Modules containing Container Managed Entity (CMP) beans must be added or removed using the application assembly tool (prior to installing the application).

1. Display the **Specify CMP Data Source Bindings** panel. If you are using the installation wizard to install an application, the panel will be displayed already.
2. For each CMP bean listed on the panel, specify a data source.

It is highly recommended that each container managed entity bean in an application (EJB module) have a default resource JNDI name specified prior to running the EJB module in the application server. However, if all container managed entity beans in an EJB module will utilize the same data source, a single JNDI name can be specified for the EJB module as a whole.

If no data source JNDI names are specified either at the entity bean level or the EJB module level, the default data source of the EJB container will be used for all unmapped container managed entity beans in the application.

3. Click **Next** to display the confirmation page.
4. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.1.3.2: Starting, stopping, and restarting applications with the Web console

During this task, you will manage applications or individual modules within applications. To start, stop, or restart an application or its modules:

1. Click **Nodes** -> *hostname* -> **Enterprise Applications** to display the applications that are installed in the current server configuration.

Each application is in one of the following states, as indicated by a graphic next to each application name:

- Running - at least one of the modules in the application is running
- Stopped - all of the modules in the application are stopped
- Cannot be run - the server must be stopped and started again with the current configuration before further management operations can be performed on the application (see below for discussion)
- Unknown - the administrative application cannot determine the current application state (see below for discussion)

(If the server is stopped, then all of the applications or modules will be in the state in which they cannot be run. [Start the server](#) before proceeding.)

2. Select the check boxes next to one or more applications or modules in the list.
3. Click the button corresponding to the action that you want to perform on the selected applications or modules:
  - **Start**
  - **Stop**
  - **Restart** (stop and start again)

Starting or stopping an application or module also sets the execution state to match. The execution state determines the initial state of the application the *next time* the server is started with the current configuration.

A "please wait" page will be displayed until all of actions (starting, stopping, or restarting) against the selected applications again, and the console message area will contain a message that the operation has been completed. If there was a problem, the message area will display an error message instead.

### Details and troubleshooting

This remainder of this article discusses the behavior of applications or modules in various states as you try to start, stop, or restart them. It includes guidelines for recognizing and correcting error situations.

### Handling of simultaneous requests

The administrative application will serialize simultaneous requests issued by users engaged in separate administration sessions. One administrator might perceive a delay while another administrator's command is completing.

### Possible causes that an application cannot be managed, or has an unknown state

An application or module can be in an unknown state if:

- The trace service diagnostic port is set to -1
- The trace service is disabled for the server in which the applications are installed
- The administrative application encounters an error when querying the status of the application
- The administrative application is unable to connect to the server at all

You might not be able to manage an application if:

- The trace diagnostic port for the server is configured, but the administrative application cannot connect to the specified port
- The application or module being managed is not defined in the configuration of the running server. This can happen if the server is not stopped and started again after installing a new application.

## 6.6.1.3.3: Uninstalling applications with the Web console


During this task, you will uninstall the application files (.ear, .jar, and .war) from the product directory structure and remove the application configuration.

To remove an application:

1. [Stop the application](#).
2. Expand the tree on the left side of the console to locate **Nodes** -> *hostname* -> **Enterprise Applications**.
3. Click **Enterprise Applications**.

The right side of the console will display the list of zero or more installed applications (as .ear files).

4. Select the application by clicking the check box to the left of the application name.
5. Click the **Uninstall** button displayed on the right side of the console, above the list of installed applications.

 [Export the application](#) before uninstalling it, in order to save the application bindings and other configuration settings from when you previously installed the application.

6. Follow the instructions on the resulting task wizard for uninstalling the application.
7. Verify that the application has been removed from the list of applications located at **Nodes** -> *hostname* -> **Enterprise Applications**.
8. [Stop the server](#).
9. Manually delete the directory corresponding to the application as described in article 6.4.2 of the InfoCenter.

The directories were *not* deleted automatically during the earlier uninstallation step because:

- Some of the resources might have been in use, because the application server could have been running as you uninstalled the application
- You might want to back up the files manually after using the console to uninstall the application



## 6.6.1.3.4: Updating applications with the Web console

During this task, you will install updated application files for an existing application, modify the configuration of an existing application, or both.

It is recommended that you review article 6.5.1 of the InfoCenter, regarding dynamic reloading options.

### Updating bindings

Use the installation sub-tasks to modify the bindings of an already installed application. To do so:

1. Locate the application in the tree on the left side of the console.
2. Click it to display its properties.
3. Click one of the links listed under **Modify Application Bindings** to launch a panel or wizard for modifying the type of bindings that you chose.

### Making bigger changes

If you need to update the contents of the .ear, .jar, or .war files comprising the application or module, or make any other configuration changes not available in the administrative console:

1. Export the application to a location other than its current location under the WebSphere product directory structure, because you are going to manually delete those directories associated with the installed application.

To export the application, uninstall its configuration settings, and delete the installed application directories, follow the instructions for [removing an application](#).

2. Modify the application .ear, .jar, or .war files if you have not done so already.
3. [Install the application or module](#) again, using the updated files.

## 6.6.1.3.5: Exporting application configurations with the Web console

If you need to make further assembly or deployment modifications after installing an application, use this feature to export the EAR file of the application. You can then use the application assembly tool to edit the EAR file.

To export an application:

1. Expand the tree on the left side of the console to locate **Nodes** -> *hostname* -> **Enterprise Applications**.
2. Click **Enterprise Applications**.

The right side of the console will display the list of zero or more installed applications (as .ear files).

3. Select the application by clicking the check box to the left of the application name. The application needs to contain EJB JAR files which have code generated for deployment, meaning the JAR files will contain Table.DDL files.
4. Click the **Export DDL** button displayed on the right side of the console, above the list of installed applications.
5. Follow the instructions on the resulting dialog. The Table.DDL file for each EJB JAR file will be available for download. Each file is <earName>\_<ejbJarName>.ddl. Click the file name to download the file.
6. Click **OK** when you are finished.
7. [Save your configuration](#).
8. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.1.3.6: Exporting DDL for EJB modules with the Web console

The Web console enables you to export the DDL for each EJB module of your application that contains entity beans. The console uses the download capability of your Web browser to save the DDL files to the location of your choice. You (or a database administrator) can then use the downloaded DDL with the database vendor product to create database tables that correspond to the EJB modules of your application.

The data store that the DDL schema is loaded into should be the same data store specified in the container managed entity bean data source bindings. The bindings should be those that were specified for each enterprise bean, using the application installation wizard.

To export a DDL:

1. Expand the tree on the left side of the console to locate **Nodes** -> *hostname* -> **Enterprise Applications**.
2. Click **Enterprise Applications**.

The right side of the console will display the list of zero or more installed applications (as .ear files).

3. Select the application by clicking the check box to the left of the application name.
4. Click the **Export DDL** button displayed on the right side of the console, above the list of installed applications.
5. For each application that you selected, if:
  - The application contains EJB JAR files
  - The EJB JAR files have had code generated for deployment

then the application will have a Table.DDL file displayed in a list of files available for download. Each file is named with the pattern *earName\_ejbJarName.ddl*.

Click a file to download it.

6. Click **OK**.
7. [Save your configuration](#).
8. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.1.3.7: Viewing deployment descriptor information for enterprise applications (read-only)

To view the deployment descriptor information for an enterprise application:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Enterprise Applications** to display the enterprise application view.
2. Click a particular enterprise application to view its details on right side of the console.
3. Click the link named **View Deployment descriptor (*application.xml*)** where *application* is the application name.

The deployment descriptor information will be displayed.

## 6.6.1.5: Administering applications with Application Assembly Tool

A J2EE<sup>(TM)</sup> application consists of one or more Web modules, EJB modules, or application client modules. The Application Assembly Tool is used to create and edit an application's modules, verify the archive files, and generate deployment code. See the related topics for links to concepts, instructions for creating an application, and field help.

## 6.6.1.5.1: Creating an application

J2EE applications can be created by using property dialog boxes or by using a wizard.

- [Using the property dialog boxes](#)
  - [Using the Create Application wizard](#)
- 

### Using the property dialog boxes

Creating a new enterprise application consists of specifying the archive files that are to be included in the application and then adding assembly instructions. Follow these steps to create a new application:

1. Click **File->New-> Application**. Then the navigation pane displays a hierarchical structure used to build the contents of the application. The icons represent EJB modules, Web modules, application client modules, assembly properties, and files comprising the application. A property dialog box containing general information about the application is displayed in the property pane.
2. By default, the EAR file name and the application display name are the same. It is recommended that you change the display name in the property pane.
3. By default, the temporary location of the EAR file is `installation_directory/bin`. You must specify a new file name and location by clicking **File->Save**. You must add at least one module (archive) before saving the application.
4. Enter values for other properties as needed. View the help for [6.6.1.0.a: Assembly properties for enterprise applications](#).
5. Add archive files to the application. You can add any combination of JAR or WAR files. There are several ways to include archive files:
  - Import an existing JAR or WAR file.
  - Open an existing EJB or Web module and copy and paste individual beans or Web components into the application.
  - Create a new JAR or WAR file.
6. To import archive files into the application, right-click the icon corresponding to the type of module (EJB module, Web module, or application client module) and choose **Import**. Use the file browser to locate and select the archive file. Click **Open**. The JAR or WAR file is displayed under the appropriate folder. The imported module's properties are also imported. Click the plus sign (+) next to the archive's icon to view its contents and edit its properties if needed.
7. To create a new JAR or WAR file to be included in the application, right-click the folder corresponding to the type of module to create (EJB Modules, Web Modules, and Application Clients), and choose **New**. Enter required values as indicated. Optionally, enter values for other properties. Click **OK**. The JAR or WAR file is displayed under the appropriate folder. Click the plus sign (+) to verify contents and enter assembly properties.

For EJB modules, you must add at least one enterprise bean. Right-click the folder corresponding to the type of bean to create (session bean or entity bean), and choose **New** or **Import**. For entity beans, choose whether you are creating a CMP or BMP bean. A property dialog box is displayed. Enter required values as indicated. Optionally, enter values for other properties. Click **OK**. The enterprise bean is displayed in the navigation pane. Click the plus sign (+) to verify the contents and enter assembly properties.
8. To copy portions of an existing module, open the module by clicking **File->Open** and selecting the file. Arrange the windows so that both the EAR file and the source archive are visible. Use a copy-and-paste operation to copy archives or components to the current module.
9. Repeat the procedure for additional JAR files or WAR files representing EJB modules, Web modules, or

application clients.

10. Define security properties for the application. Right-click the **Security Roles** icon and click **New**. Enter values on the property dialog box and then click **OK**. View the help for [6.6.1.0.1.a: Assembly properties for security roles \(application\)](#).
  11. Add supplementary files needed by the application. Right-click the **Files** icon in the navigation pane and choose **Add Files**. Click **Browse** to navigate to the desired directory or archive. Click **Select**. If you are adding an entire archive, select the directory that contains the archive. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. The selected files are displayed in the **Selected Files** window. Relative path names are maintained. When the **Selected Files** window contains the correct set of files, click **OK**.
  12. Review the contents of the application and make any desired changes.
  13. Save the application EAR file. Click **File->Save**. Enter a name for the application, for example, temp.ear. Click **Save**. A dialog box confirming the save operation is displayed. Click **OK**.
- 

## Using the Create Application wizard

Use this wizard to create an enterprise application. An enterprise application can consist of one or more EJB modules, Web modules, or application clients, and a deployment descriptor. During creation of the application, you specify the archive files for each module to be included in the application. You also specify other information about the application, such as security roles.

You can create new EJB modules, Web modules, and application client modules, or import existing modules. When the wizard is completed, your application is packaged in an EAR file and resides in the directory location that you specified.

To create an enterprise application, click the **Wizards** icon on the tool bar and then click **Application**. Follow the instructions on each panel.

- [Specifying application properties](#)
- [Adding supplementary files](#)
- [Choosing application icons](#)
- [Adding EJB modules](#)
- [Adding Web modules](#)
- [Adding application client modules](#)
- [Adding security roles](#)
- [Setting additional properties and saving the archive](#)

## Specifying application properties

On the **Specifying Application Properties** panel:

1. Specify a display name for the application (required). The display name is used to identify your application in the Application Assembly Tool and can be used by other tools.
2. Specify a file name for the application (required). The file name specifies a location on your system where the EAR file is to be created.
3. Provide a short description of the application (optional).

4. Click **Next**.

## Adding supplementary files

On the **Adding Supplementary Files** panel, specify the location of additional files to be included in your application. These are files other than the J2EE modules to be included--for example, libraries and utilities needed by the application. To add or remove files:

1. Click **Add**. Use the file browser to locate and choose one or more files to add to the application. First, browse for the root directory or archive where the files are located and click **Select**. If you are adding an entire archive, select the directory that contains the archive.
2. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. Relative path names are maintained. The selected files are displayed in the Selected Files window. Click **OK**. The files are displayed in a table on the wizard panel.
3. If you want to remove a file, select the file in the table and then click **Remove**.
4. Continue to add or remove files until you have the correct set of files. Click **OK**.
5. Click **Next**.

## Choosing application icons

On the **Choosing Application Icons** panel:

1. Specify the full path name of a small icon, or click **Browse** to locate and select the file. The icon must be a GIF or JPEG image 16x16 pixels in size.
2. Specify a full path name of a large icon, or click **Browse** to locate and select the file. The icon must be a GIF or JPEG image 32x32 pixels in size.
3. Click **Next**.

## Adding EJB modules

On the **Adding EJB Modules** panel, add one or more EJB JAR files.

1. To add JAR files, click **Add**. Use the file browser to locate and select the files, and then click **Open**. As prompted, verify or enter a file name and alternate deployment descriptor for the JAR file. Click **OK**. The JAR files are displayed in a table on the wizard panel.
2. To remove a JAR file, select the file and then click **Remove**.
3. Continue adding or removing JAR files as needed. When you are finished adding JAR files, click **Next**.

## Adding Web modules

On the **Adding Web Modules** panel, add one or more WAR files to the application.

1. To add WAR files, click **Add**. Use the file browser to locate and select the files and then click **Open**. As prompted, verify or enter a file name and context root for the WAR file. Click **OK**. The WAR files are displayed in a table on the wizard panel.
2. To remove a WAR file, select the file and then click **Remove**.
3. Continue adding or removing files as needed. When you are finished adding WAR files, click **Next**.

## Adding application client modules



On the **Adding Application Client Modules** panel, add one or more application client JAR files to the application.

1. To add JAR files, click **Add**. Use the file browser to locate and select the files and then click **Open**. As prompted, verify or enter a file name and alternate deployment descriptor for the JAR file. Click **OK**. The JAR files are displayed in a table on the wizard panel.
2. To remove a JAR file, select the file and click **Remove**.
3. Continue adding or removing files as needed.
4. When you are finished adding application client JAR files, click **Next**.

## Adding security roles

On the **Adding Security Roles** panel, specify a security role that a principal must be granted in order to access the application. These security roles must be unique to the application. Any security roles defined in the application's modules are also displayed in the table on this panel.

1. Click **Add**. Type a role name (required). Optionally, enter a description of the role. View the help for [6.6.5.0.5: Assembly properties for security roles](#). Click **OK**. The role and its description are displayed in a table on the wizard panel.
2. Continue to add security roles as needed. If you need to remove a role, select the role in the table and then click **Remove**.

## Setting additional properties and saving the archive

Click **Finish** to complete the wizard. To change settings for properties, click **Back** to return to the appropriate panel. Make any needed changes, and then click **Finish**.

After you click **Finish**, the contents of the archive are displayed in the navigation pane. You can continue adding or modifying properties as needed. For example, you can add binding information. When you are finished editing the archive, click **File->Save** to save the archive file.

## 6.6.2: Administering nodes (overview)

Administrative server node configurations provide information about machines (physical hosts) involved in the IBM WebSphere Application Server environment.

WebSphere Application Server supports one administrative server per physical machine. Therefore, it does not matter whether the administrator considers the physical machine or the administrative server to be the "node."

(The current exception is for the AS/400 operating system, on which the product supports multiple administrative servers on each physical machine).

### Regenerating the WebSphere plug-in for the Web server

It is suggested you review the information about [administering WebSphere plug-ins for Web servers](#), because regenerating (updating) the plug-in is a task associated with the administrative node.

## 6.6.2.0: Node properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Name**



**or Node Name**



A logical name for the node, such as "localhost" or the short (unqualified) DNS name of the node. Machine names must be unique within an administrative domain.

**Status**



The current state of the node, such as running or stopped

## 6.6.2.3: Administering nodes with the Web console

Use the Web console to edit the configurations of nodes. At present, you can define one domain, onenode, and one application server in a configuration.

Work with objects of this type by locating them in the tree on the left side of the console:

Click the **Nodes** entry in the tree.

## 6.6.2.3.4: Updating nodes with the Web console

During this task, you will update the configuration of an existing node.

To update a node configuration:

1. Expand the tree on the left side of the console to locate **Nodes** -> *hostname*, where *hostname* is either literally "localhost" or is the hostname of the machine on which the product is running.
2. Click *hostname* to display its properties on the right side of the console.
3. Adjust the node properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.3: Administering application servers

An application server configuration provides information for starting and managing a server process to handle requests for enterprise applications and their components.

The WebSphere administrator configures one or more application servers with settings for:

- Giving the server a unique name by which to manage it
- Associating the server with a unique transport port
- Specifying Java command line arguments and environment variables for starting the application server process
- Enabling WebSphere security for the server
- Associating the server with an operating system user ID and security group
- Specifying how many times to try to start or ping the server before giving up
- Specifying the process priority on the operating system
- Defining standard in, standard error, standard out streams for the server runtime
- Specifying a working directory
- Conducting transactions
- Specifying tracing of the server and its child components

### Regenerating the WebSphere plug-in for the Web server

It is suggested you review the information about [administering WebSphere plug-ins for Web servers](#), because you will need to regenerate the plug-in configuration each time you create a new application server.

## 6.6.3.0: Application server properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

An application server contains a variety of configurations:

- General, Advanced, and File properties  or Process Definition 
- Transaction properties  
- Custom service properties  
- JVM properties  
- EJB container service properties  
- Web container service properties  
- Transport configuration properties  
- Session Manager service properties  
- Trace service properties  
- Object Level Trace service properties  
- Object Request Broker (ORB) service properties 
- Location Service Daemon properties 
- Security service properties 

... as well as these additional settings:

**Application Server Name**  or **Name** 

A logical name for the application server. The name must be unique within the node (physical machine) containing the application server.

**Execution State**  or **Node Startup State** 

*For the Advanced Single Server Edition:* The state that you would like for the application server to be in,

the next time the *product* is stopped and started again.

*For the Advanced Edition:* The state you would like the application server to be in, the next time *the node containing the application server* is stopped and started again.

Choices are Stopped, Running, and Last State (meaning the last state that the applicationserver was in, before node was stopped).

## Module Visibility

The classloader isolation mode to use for the application server. By changing the default visibility level, it is possible to have visibility of classes in other modules, or even other applications.

- Specify "SERVER" to allow all application classloaders on the system to have visibility of all other application classloaders in the system. Search order is the same order as when the modules were initialized into the system.
- Specify "APPLICATION" to allow all classloaders in a J2EE application to have visibility of other classloaders in the same application. Search order is the order the modules are defined in the application.xml for the EAR.
- Specify "MODULE" to use one classloader per module. Each module (EAR, JAR, or WAR) has its own unique classloader. Visibility to other modules in the application is only achieved when MANIFEST Class-Path entries are added to a module.
- Specify "COMPATIBILITY" for compatibility with applications from WebSphere Application Server 3.5.x and 3.0.2.x. In this mode, all EJB module classloaders have visibility of all other EJB module classloaders and all Web Application modules have visibility of the EJB classloaders. Search order for the EJB classloaders is determined by the order in which the EJB modules were initialized.

Portable J2EE applications should be written with Module-level visibility. The other modes are provided to accomodate applications that have different visibility requirements that cannot be modified.

The default module visibility differs between *Advanced Edition* and *Advanced Single Server Edition*.

- The *Advanced Edition* default is MODULE
- The *Advanced Single Server Edition* default is COMPTABILITY

Suppose you have a WAR module that depends on an EJB JAR installed in a different application or the same application, but the WAR module lacks a manifest classpath entry. Such a WAR module will run on *Advanced Single Server Edition* using the default settings, but will not run on *Advanced Edition* until you reset the Module Visibility field to COMPATIBILITY.

If you do not want to run *Advanced Edition* WAR modules in COMPATABILITY mode, then specify the classpath during assembly for the WAR module that you are going to run on *Advanced Edition*.

## Name

See Application Server Name

## Node

The administrative node on which the application server resides

## Node Startup State

See Execution State



## 6.6.3.3: Administering application servers with the Web console

Use the Web console to edit the configurations of application servers. You *cannot* use this console to start and stop application servers. At present, you can define one domain, one node, and one application server in a configuration.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

## 6.6.3.3.1: Starting application servers with the Web console

During this task, you will start a new server process based on the current server configuration's process definition settings.

1. Select the Configuration menu listed at the top of the right side of the console.
2. Select configuration file of the server that you want to stop.
3. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** to display the application servers view.
4. Select the application server that you want to start.
5. Click the **Start** button to launch a new server process based on the current server configuration's process definition settings.

You can view logs to check whether the server has started.

### Variations -- starting the server for tracing and debugging

To start the server with standard Java debugger:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Process Definition** -> **JVM Settings** to view the JVM settings.
2. In the JVM settings, check Debug mode to enable the standard Java debugger. Set the debug arguments if needed.
3. [Save the changes to a configuration file.](#)
4. [Stop the server.](#)
5. Start the server again (see above).

To start the server with IBM Distributed debugger or Object Level Trace:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **IBM Debug and OLT**.
2. In IBM Debug and OLT settings, enable IBM Distributed debugger, enable Object Level Trace, or both.

In order to use Distributed Debugger, you must enable both.

3. [Save the changes to a configuration file.](#)
4. [Stop the server.](#)
5. Start the server again (see above).

## 6.6.3.3.2: Pinging application servers with the Web console

During this task, you will ping a server process based on the current server configuration's process definition settings. To determine if the application server is already running on the network, the administrative application provides a Ping command that will attempt to connect to the application server on the DrAdmin port configured for that server (typically, port 7000).

1. Select the Configuration menu listed at the top of the right side of the console.
2. Select configuration file of the server that you want to stop.
3. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** to display the application servers view.
4. Select the application server that you want to ping, to display its properties on the right side of the console.
5. Click the **Ping** button to ping the server process specified by the current server configuration file.

The ping will be successful if the server is running. It will fail if the server is not running, or the administrative application can not connect with the server on the administrative port.

## 6.6.3.3.3: Stopping application servers with the Web console

During this task, you will stop a server process based on the current server configuration's process definition settings.

1. Select the Configuration menu listed at the top of the right side of the console.
2. Select configuration file of the server that you want to stop.
3. In the tree on the left side of the console, click **Nodes** -> *hostname* -> -> **Application Servers** to display the application servers view.
4. Select the application server that you want to stop, to display its properties on the right side of the console.
5. Click the **Stop** button to stop the server process.
6. The command will attempt to stop the server.

If the server is not running, or the administrative application could not establish communications with the server process, an error message will be displayed at the top of the page.

A warning message will be displayed if you are stopping the same server that is running the administrative application.

7. Click **Continue** on the confirmation screen to stop the server. You will be automatically logged out of the administrative application, and will need to [log in](#) again when the server has started.

## 6.6.3.3.4: Updating application server configurations with the Web console

During this task, you will update the configuration of an existing application server.

To update an application server configuration:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* where *application\_server\_name* is the name of the existing application server.
2. Adjust the application server properties.
3. When you are finished, click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.4: Administering EJB containers (overview)

A container configuration provides information about an enterprisebean container. The administrator can specify several properties to address basic questions about the container location and behavior.

### Specifying the server in which the container will reside

Each enterprise bean container resides in a particular application server.

When the administrator adds a new container to the WebSphere administrativedomain, he or she must associate the container with a particular server (also known as the container's parent).

An application server can host multiple containers.

### Specifying how beans in the container will get database connections

Every container can support the two main bean types, session beans and entitybeans:

- Entity beans require database connections because they store permanent data.
- Session beans do not *require* database access, though they can obtain it indirectly (as needed) by accessing entity beans.

A data source is an administrative resource that defines a pool of database connections. Servlets and enterprise beans use data sources to obtain database connections.

When configuring a container, the administrator can specify a default data source for the container. This data source will be the default data source used by any entity beans installed in the container that use container managed persistence (CMP).

When configuring a CMP entity bean, the administrator can specify which data source the container must use for managing the persistent state of the entity bean. If the administrator specifies a data source for an individual CMP entity bean then this data source will override any data source specified on the container.

Specifying a default data source is optional if each CMP entity bean in the container has a data source specified in its configuration. If a default data source is not specified and a CMP entity bean is installed in that container without specifying a data source for that bean then it will not be possible to start that CMP entity bean.

The default data source for a container is secure. When specifying it, the administrator must provide the user ID and password for accessing the data source.

### Specifying how the container will manage cached bean instances

Each container keeps a cache of bean instances for ready access. The WebSphere administrator specifies settings governing the cache size and a policy for removing unused items from the cache.

### Specifying where the container will passivate beans to make room in its cache

A container can *passivate* session beans to make room in its cache. The container saves a serialized session bean to a file. It restores the bean to the cache when more room is available.

The WebSphere administrator specifies a passivation directory in which to keep the files.

## 6.6.4.0: EJB container properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Application Server



The application server of which the EJB container is a part

### Default Data Source



The data source to use to connect to a JDBC-compliant database, such as DB2

### Cache clean-up interval



### or Clean-up Interval



The interval at which the container attempts to remove unused items from the cache to reduce the total number of items in the cache to the value of the Cache preferred limit property.

The cache manager tries to maintain some unallocated entries that can be allocated quickly, as needed. A background thread attempts to free some entries while ensuring that some unallocated entries are maintained. If the thread runs while the application server is idle, then when the application server needs to allocate new cache entries, it does not pay the performance cost of removing entries from the cache.

In general, increase this parameter as the cache size increases.

*For the Advanced Single Server Edition:* This value must be a positive integer specified in milliseconds.

### Cache size



The number of buckets in the cache hash table

If you change this value, change the Cache absolute limit property to correspond. For example, if you change the cache size to 3000, change the cache absolute limit to 3000, unless for some reason you do not want all of the available cache to be used.

*For Advanced Single Server Edition:* This value must be a positive integer.

### Inactive Pool Cleanup Interval



The interval at which inactive pools will be reduced to their minimum size.

*For Advanced Single Server Edition:* The value must be a positive integer specified in milliseconds.

### Installed EJB Modules



The EJB modules that are installed in the EJB container of this server

### Node



The node with which the application server is associated

### Passivation Directory



The directory into which the container will save the persistent state of passivated session beans

Session beans are passivated when the container needs to reclaim space in the bean cache. At passivation time, the container serializes the bean instance to a file in the passivation directory and discards the instance from the bean cache. If, at a later time, a request arrives for the passivated bean instance, the container retrieves it from the passivation directory, deserializes it, returns it to the cache, and dispatches the request to it. If any step fails (for example, if the bean instance is no longer in the passivation directory), then the method invocation fails.



## 6.6.4.3: Administering EJB containers with the Web console

Use the Web console to edit the configurations of EJB containers, which are responsible for providing services needed by running EJB modules and the enterprise beans that they contain. Each application server has one logical EJB container, which you can modify but not create or remove.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **EJB Containers**

## 6.6.4.3.4: Updating EJB container configurations with the Web console

During this task, you will update the configuration of an existing EJB container, which is part of an application server configuration.

To update an EJB container configuration:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **EJB Container** where *application\_server\_name* is the name of the existing application server.
2. Click the EJB container to display its properties on the rightside of the console.
3. Modify the properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.5: Administering EJB modules (overview)

Administration of EJB modules consists of the following:

- Creating the module, setting deployment descriptor properties, and generating code for deployment using the Application Assembly Tool
- Setting additional configuration properties using the console

### Classpath considerations

An important classpath-related setting to note is the Module Visibility. This application server setting impacts the portability of applications and standalone modules from other WebSphere Application Server versions and editions. If your existing module does not run as-is when you transfer it to Version 4.0, you might need to reassemble an existing module or change the module visibility setting.

See [the information on setting classpaths](#) for a full discussion of classpath considerations. See the [applicationserver property reference](#) for information about the module visibility setting.

## 6.6.5.0: EJB module properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Application or  Application Ref **

The application installation binding within which the module-to-server installation binding is contained. This is typically the logical name of the enterprise application you configured to contain this EJB module.

**Default Data Source **

The data source to use for this EJB module, unless a different one is specified

**Execution State **

The state that you would like the enterprise bean module to be in, the next time the product is stopped and started again

**Name  or Module Name **

An administrative name for the EJB module

**Password **

The password corresponding to the specified user ID

**Server **

The application server on which the EJB module is installed

**URI **

A URI that, when resolved relative to the application URL, specifies the location of the module archive on a file system. The URI must match the URI of a ModuleRef URI in the deployment descriptor of an application if the module was packaged as part of a deployed application (EAR).

**User ID **

The user ID for accessing the default data source

See also the other [application server properties](#).

## 6.6.5.0.1: Assembly properties for entity beans

### **EJB name (Required, String)**

Specifies a logical name for the enterprise bean. This name must be unique within the EJB module. There is no relationship between this name and the JNDI name.

### **Display name**

Specifies a short name that is intended to be displayed by GUIs.

### **Description**

Contains text describing the entity bean.

### **Home Interface (Required, String)**

Specifies the full name of the enterprise bean's home interface class--for example, `com.ibm.ejs.doc.account.AccountHome`.

### **Remote Interface (Required, String)**

Specifies the full name of the enterprise bean's remote interface class--for example, `com.ibm.ejs.doc.account.Account`.

### **EJB class (Required, String)**

Specifies the full name of the enterprise bean class--for example, `com.ibm.ejs.doc.account.AccountBean`.

### **Persistence**

Specifies whether an entity bean manages its own persistent storage or whether storage is managed by the container. The valid values are `Bean-managed` and `Container-managed`.

### **Primary key field**

Specifies the name of a simple primary key. Simple primary keys map to a single field in the entity bean class and are made up of primitive Java data types (such as integer or long). If exactly one CMP field is the primary key, it can be specified here.

### **Primary key class (Required, String)**

Specifies the full name of the bean's primary key class--for example, `com.ibm.ejs.doc.account.AccountKey`. Composite primary keys map to multiple fields in the entity bean class (or to data structures built from the primitive Java data types), and must be encapsulated in a primary key class. More complicated enterprise beans are likely to have composite primary keys, with multiple instance variables representing the primary key. A subset of the container-managed fields is used to define the primary key class associated with each instance of an enterprise bean.

### **Reentrant**

Specifies whether the entity bean is reentrant or not. If an enterprise bean is reentrant, it can invoke methods on itself or call another bean that invokes a method on the calling bean. Only entity beans can be reentrant. If an entity bean is not reentrant and a bean instance is executing a client request in a transaction context and another client using the same transaction context makes a request on the same bean instance, the EJB container throws the `java.rmi.RemoteException` exception to the second client. If a bean is reentrant, the container cannot distinguish this type of illegal loopback call from a legal concurrent call and the bean must be coded to detect illegal loopback calls.

### **Small icon**

Specifies a JPEG or GIF file containing a small image (16x16 pixels). The image is used as an icon to represent the entity bean in a GUI.

### **Large icon**

Specifies a JPEG or GIF file containing a large image (32x32 pixels). The image is used as an icon to

represent the entity bean in a GUI.

### **Inheritance root**

Specifies whether or not the enterprise bean is at the root of an inheritance hierarchy.

### **Bean Cache -- Activate at**

Specifies the point at which an enterprise bean is activated and placed in the cache. Removal from the cache and passivation is also governed by this setting. Valid values are Once and Transaction. Once indicates that the bean is activated when it is first accessed in the server process, and passivated (and removed from the cache) at the discretion of the container--for example, when the cache becomes full. Transaction indicates that the bean is activated at the start of a transaction and passivated (and removed from the cache) at the end of the transaction. The default value is Transaction.

### **Bean Cache -- Load at**

Specifies when the bean loads its state from the database. The value of this property implies whether the container has exclusive or shared access to the database. Valid values are Activation and Transaction. Activation indicates that the bean is loaded when it is activated (Once or Transaction) and implies that the container has exclusive access to the database. Transaction indicates that the bean is loaded at the start of a transaction and implies that the container has shared access to the database. The default is Transaction.

The settings of the Activate at and Load at properties govern which commit options are used. The commit options are described in detail in the Enterprise JavaBeans specification, version 1.1.

- For Commit Option A (implies exclusive DB access), use Activate at = Once and Load at = Activation. This option reduces database I/O (avoids calls to the `ejbLoad` function) but serializes all transactions accessing the bean instance. Option A can increase memory usage by maintaining more objects in the cache, but could provide better response time if bean instances are not generally accessed concurrently by multiple transactions.
- For Commit Option B (implies shared DB access), use Activate at = Once, Load at = Transaction. Option B can increase memory usage by maintaining more objects in the cache. However, because each transaction creates its own copy of an object, there can be multiple copies of an instance in memory at any given time (one per transaction), requiring that the database be accessed at each transaction. If an enterprise bean contains a significant number of calls to the `ejbActivate` function, using Option B can be beneficial because the required object is already in the cache. Otherwise, this option does not provide significant benefit over Option A.
- For Commit Option C (implies shared DB access), use Activate at = Transaction and Load at = Transaction. This option can reduce memory usage by maintaining fewer objects in the cache, however, there can be multiple copies of an instance in memory at any given time (one per transaction). This option can reduce transaction contention for enterprise bean instances that are accessed concurrently but not updated.

### **Locale location**

Specifies the language used when the enterprise bean retrieves and displays message catalogs: the local language of the client that invoked the bean method or the local language of the server where the bean is running. Valid values are server and caller.

### **Local Transactions -- Boundary**

Specifies when a local transaction begins. The default behavior is that the local transaction begins when the method begins and ends when the method ends.

### **Local Transactions -- Unresolved action**

Specifies the action the container must take if resources are uncommitted by an application in a local transaction. A local transaction context is created when a method executes in what the EJB specification refers to as an unspecified transaction context. Valid values are Rollback and Commit. The default is Rollback.

**Local Relationship Roles - Name**

Information is not available.

**Local Relationship Roles - Source EJB Name**

Information is not available.

**Local Relationship Roles - is Forward**

Information is not available.

**Local Relationship Roles - is Navigable**

Information is not available.

**JNDI name**

Specifies the JNDI name of the bean's home interface. This is the name under which the enterprise bean's home interface is registered and therefore is the name that must be specified when an EJB client does a lookup of the home interface.

**Data Source - JNDI name**

Specifies the JNDI name for the bean's data source.

**Default Authorization - User ID**

Specifies the default user ID for connecting to a data source.

**Default Authorization - Password**

Specifies the default password for connecting to a data source.

## 6.6.5.0.2: Assembly properties for CMP fields

Container-managed persistence (CMP) fields define the variables in the beanclass for which the container must handle persistence management.

**Name (Required, String)**

Specifies a subset of public variables in an enterprise bean's implementation class.



## 6.6.5.0.3: Assembly properties for method extensions

### Method type

Specifies the type of the enterprise bean method. Valid values are Home methods, Remote methods, and All methods.

### Name

Specifies the name of an enterprise bean method, or the asterisk character(\*) . The asterisk is used to denote all the methods of the specified interface--for example, all methods in the remote interface.

### Parameters

Contains a list of fully qualified Java type names of the method parameters. Used to identify a single method among multiple methods with an overloaded method name.

### Isolation level attributes

The transaction isolation level determines how isolated one transaction is from another. This property can be set for individual methods in an enterprise bean or for all methods in the enterprise bean. An asterisk is used to indicate all methods in the bean. However, within a transactional context, the isolation level associated with the first method invocation becomes the required isolation level for all methods invoked within that transaction. If a method is invoked with a different isolation level from that of the first method, the `java.rmi.RemoteException` exception is thrown.

### Isolation level

Specifies the level of transactional isolation. The valid values are as follows:

- Serializable. This level prohibits the following types of reads:
  - Dirty reads, where a transaction reads a database row containing uncommitted changes from a second transaction.
  - Nonrepeatable reads, where one transaction reads a row, a second transaction changes the same row, and the first transaction rereads the row and gets a different value.
  - Phantom reads, where one transaction reads all rows that satisfy an SQL WHERE condition, a second transaction inserts a row that also satisfies the WHERE condition, and the first transaction applies the same WHERE condition and gets the row inserted by the second transaction.
- Repeatable read. This level prohibits dirty reads and nonrepeatable reads, but it allows phantom reads.
- Read committed. This level prohibits dirty reads, but allows nonrepeatable reads and phantom reads.
- Read uncommitted. This level allows dirty reads, nonrepeatable reads, and phantom reads.

The container uses the transaction isolation level attribute as follows:

- Session beans and entity beans with bean-managed persistence (BMP)--For each database connection used by the bean, the container sets the transaction isolation level at the start of each transaction unless the bean explicitly sets the isolation level on the connection.
- Entity beans with container-managed persistence (CMP)--The container generates database access code that implements the specified isolation level.

### Security Identity

Specifies that a principal's credential properties are to be handled as indicated in the Run-As mode property. Checking this box makes the Run-as Mode properties editable.

### Description

Contains text describing or commenting on the security instructions.

## Run-As Mode

Credential information is used by the security service to determine the permissions that a principal has on various resources. At appropriate points, the security service determines whether the principal is authorized to use a particular resource based on the principal's permissions. If the method invocation is authorized, the security service does the following with the principal's credential properties based on the value of the Run-as Mode property of the enterprise bean:

- Use Identity of Caller -- the security service makes no changes to the principal's credential properties.
- Use Identity of EJB Server -- the security service alters the principal's credential properties to match the credential properties associated with the EJB server.
- Use Identity Assigned to Specified Role -- A principal that has been assigned to the specified security role is used for the execution of the bean's methods. This association is part of the application binding where the role is associated with a user ID and password of a user who is granted that role.

## Role Name

Specifies the name of a security role. If the Use Identity Assigned to Specified Role button is selected, a principal that has been granted this role will be used.

## Description

Contains a description of the security role.

## Access Intent - Intent type

Specifies whether the method is a read-only method or whether the method can update data (or invoke other methods that can update data, in the same transaction). The legal values are read or update (read/write).

## Finder descriptor - User

Specifies that the user has provided a finder helper class in the entity bean's home interface. The class contains specialized finder methods.

## Finder descriptor - EJB QL

Describes the semantics of a finder method using the EJB QL (Enterprise JavaBeans query language). EJB QL is a declarative, SQL-like language intended to be compiled to the target language of the persistent data store used by a persistence manager. The language is independent of the bean's mapping to a relational data store and is therefore portable. The EJB query specifies a search based on the persistent attributes and relationships of the bean. An EJB query contains a SELECT clause (optional), a FROM clause (required), a WHERE clause (optional), and an ORDER BY clause (optional). The SELECT clause specifies the EJB objects to return. The FROM clause specifies the collections of objects to which the query is to be applied. The WHERE clause contains search predicates over the collections. The ORDER BY clause specifies the ordering of the resulting collection.

## Finder descriptor - Full select

Note: For information on restrictions, see the documentation for Deployment Tool for Enterprise JavaBeans.

Describes the semantics of the finder method using a SQL SELECT statement. The SELECT statement indicates the EJB objects to return.

## Finder descriptor - Where clause

Note: For information on restrictions, see the documentation for Deployment Tool for Enterprise JavaBeans.

Describes the semantics of the finder method using a SQL WHERE clause. This clause restricts the

results that are returned by the query.

## 6.6.5.0.4: Assembly properties for session beans

### **EJB name (Required, String)**

Specifies a logical name for the enterprise bean. This name must be unique within the EJB module. There is no relationship between this name and the JNDI name.

### **Display name**

Specifies a short name that is intended to be displayed by GUIs.

### **Description**

Contains text describing the session bean.

### **Home interface (Required, String)**

Specifies the full package name of the enterprise bean's home interface class, for example, `com.ibm.ejs.doc.account.AccountHome`.

### **Remote interface (Required, String)**

Specifies the full package name of the enterprise bean's remote interface class, for example, `com.ibm.ejs.doc.account.Account`.

### **EJB class (Required, String)**

Specifies the full package name of the enterprise bean class, for example, `com.ibm.ejs.doc.account.AccountBean`.

### **Session type**

Specifies whether the enterprise bean maintains a conversational state (is a stateful session bean) or does not (is a stateless session bean). Valid values are `stateful` and `stateless`.

### **Transaction type**

Specifies whether the enterprise bean manages its own transactions or whether the container manages transactions on behalf of the bean. Valid values are `container` or `bean`.

### **Small icon**

Specifies a JPEG or GIF file containing a small image (16x16 pixels). The image is used as an icon to represent the session bean in a GUI.

### **Large icon**

Specifies a JPEG or GIF file containing a large image (32x32 pixels). The image is used as an icon to represent the session bean in a GUI.

### **Timeout**

Specifies the idle timeout value for the enterprise bean in seconds. A zero (0) value indicates that idle bean instances time out after the maximum allowable timeout period elapses. By default, the timeout is 600 seconds or 10 minutes. This property does not apply to session beans.

### **Inheritance root**

Specifies whether the enterprise bean is at the root of an inheritance hierarchy.

### **Bean Cache -- Activate at**

Applies to stateful session beans only, not to stateless beans. Specifies the point at which an enterprise bean is activated and placed in the cache. Removal from the cache and passivation are also governed by this setting. Valid values are `Once` and `Transaction`. `Once` indicates that the bean is activated when it is first accessed in the server process and passivated (and removed from the cache) at the discretion of the container--for example, when the cache becomes full. `Transaction` indicates that the bean is activated at the start of a transaction and passivated (and removed from the cache) at the end of the transaction. The default value is `Once`.

**Bean Cache -- Load at**

This property does not apply to session beans.

**Locale location**

Specifies the language used when the enterprise bean retrieves and displays message catalogs: the local language of the client that invoked the bean method or the local language of the server where the bean is running. Valid values are server and caller.

**Local Transactions -- Boundary**

Specifies when a local transaction begins. The default behavior is that the local transaction begins when the method begins and ends when the method ends. This property is not applicable for session beans.

**Local Transactions -- Unresolved action**

Specifies the action the container must take if resources are uncommitted by an application in a local transaction. A local transaction context is created when a method executes in what the EJB specification refers to as an unspecified transaction context. Valid values are Rollback and Commit. The default is Rollback.

**JNDI name**

Specifies the JNDI name of the bean's home interface. This is the name under which the enterprise bean's home interface is registered and therefore is the name that must be specified when an EJB client does a lookup of the home interface.

## 6.6.5.0.5: Assembly properties for security roles

A security role is a logical grouping of principals. Access to operations (such as EJB methods) is controlled by granting access to a role.

### **Role name (Required, String)**

Specifies the name of a security role.

### **Description**

Contains text describing the security role.

If specifying security roles at the application level (EAR file), the following properties apply:

### **Role Name (Required, String)**

Specifies the name of a security role that is unique to the application.

### **Description**

Contains text describing the security role.

### **Binding -- Groups -- Name**

Specifies user groups that are granted the security role.

### **Binding -- Users -- Name**

Specifies users that are granted the security role.

### **Binding -- Special Subjects -- Name**

Specifies one of two special categories of users to which roles can be granted: Everyone or All authenticated users. If the special subject Everyone is granted a role, then all users, including those who did not authenticate, are granted the role. In other words, a method on an enterprise bean or a URI is unprotected if any of the required roles for that method are granted to the special subject Everyone. In the case of All authenticated users, any user who can authenticate by using a valid user ID and password is considered to be granted that role.

## 6.6.5.0.6: Assembly properties for method permissions

A method permission is a mapping between one or more security roles and one or more methods that a member of the role can invoke. Assembly properties for method permissions include an optional description, a list of security role names, and a list of methods. The security roles must be defined, and the methods must be methods defined in the enterprise bean's remote or home interfaces.

### **Method permission name**

Specifies a name for the mapping between method permissions and security roles.

### **Description**

Contains text describing the mapping between method permissions and security roles.

### **Methods - Name**

Specifies the name of an enterprise bean method, or the asterisk (\*) character. The asterisk is used to denote all the methods of the specified interface--for example, all methods of the remote interface.

### **Methods - Enterprise Bean**

Specifies the name of the enterprise bean that contains the method.

### **Methods - Type**

Distinguishes between a method with the same signature that is defined in both the home and remote interface. Valid values are All methods, Remote methods, or Home methods.

### **Methods - Parameters**

Contains a list of fully qualified Java type names of the method parameters. This property is used to identify a single method among multiple methods with an overloaded method name.

### **Roles -- Role Name**

Specifies the security role that must be granted in order to invoke the method.

## 6.6.5.0.7: Assembly properties for container transactions

Container transaction properties specify how the container must manage transaction scopes for the enterprise bean's method invocations. Specify one or more methods and associate a transaction attribute with each method.

### Name

Specifies a name for the mapping between a transaction attribute and one or more methods.

### Description

Contains text describing the mapping.

### Transaction Attribute

Specifies how the container must manage the transaction boundaries when delegating a method invocation to an enterprise bean's business method. The legal values are Never, Mandatory, Requires New, Required, Supports, Not Supported, and Bean Managed. The default is Not Supported.

- **Mandatory.** Directs the container to always invoke the bean method within the transaction context associated with the client. If the client attempts to invoke the bean method without a transaction context, the container throws the `javax.jts.TransactionRequiredException` exception to the client. The transaction context is passed to any EJB object or resource accessed by an enterprise bean method.  
  
EJB clients that access these entity beans must do so within an existing transaction. For other enterprise beans, the enterprise bean or bean method must implement the Bean Managed value or use the Required or Requires New value. For non-enterprise bean EJB clients, the client must invoke a transaction by using the `javax.transaction.UserTransaction` interface.
- **Supports.** Directs the container to invoke the bean method within a transaction context if the client invokes the bean method within a transaction. If the client invokes the bean method without a transaction context, the container invokes the bean method without a transaction context. The transaction context is passed to any enterprise bean objects or resources that are used by this bean method.
- **Never.** Directs the container to invoke bean methods without a transaction context.
  - If the client invokes a bean method from within a transaction context, the container throws the `java.rmi.RemoteException` exception.
  - If the client invokes a bean method from outside a transaction context, the container behaves in the same way as if the Not Supported transaction attribute was set. The client must call the method without a transaction context.
- **Requires New.** Directs the container to always invoke the bean method within a new transaction context, regardless of whether the client invokes the method within or outside a transaction context. The transaction context is passed to any enterprise bean objects or resources that are used by this bean method.
- **Not Supported.** Directs the container to invoke bean methods without a transaction context. If a client invokes a bean method from within a transaction context, the container suspends the association between the transaction and the current thread before invoking the method on the enterprise bean instance. The container then resumes the suspended association when the method invocation returns. The suspended transaction context is *not* passed to any enterprise bean objects or resources that are used by this bean method.
- **Required.** Directs the container to invoke the bean method within a transaction context. If a client invokes a bean method from within a transaction context, the container invokes the bean method within the client transaction context. If a client invokes a bean method outside a transaction context, the container creates a new transaction context and invokes the bean method from within



that context. The transactioncontext is passed to any enterprise bean objects or resources that are used bythis bean method.

- Bean Managed. Notifies the container that the bean class directlyhandles transaction demarcation. This property can be specified onlyfor session beans, and it cannot be specified for individual beanmethods.

### **Method Elements - Name**

Specifies the name of an enterprise bean method, or the asterisk character(\*). The asterisk is used to denote all the methods of the specifiedinterface--for example, all methods of the remote interface.

### **Method Elements - Enterprise bean**

Specifies which enterprise bean contains the methods indicated in the Nameproperty.

### **Method Elements - Type**

Used to distinguish between a method with the same signature that isdefined in both the home and remote interface. Valid values are Allmethods, Remote methods, or Home methods. Use All methods if atransaction attribute applies to all methods of the bean.

### **Method Elements - Parameters**

Contains a list of fully qualified Java type names of the methodparameters. This property is used to identify a single method amongmultiple methods with an overloaded method name.

## 6.6.5.0.aa: Assembly properties for EJB modules

### File name (Required, String)

Specifies the file name of the EJB module, relative to the top level of the application package.

### Alternative DD

Specifies the file name for an alternative deployment descriptor file to use instead of the original deployment descriptor file in the module's JAR file. This file is the postassembly version of the deployment descriptor file. (The original deployment descriptor file can be edited to resolve dependencies and security information. Directing the use of the alternative deployment descriptor allows you to keep the original deployment descriptor file intact). The value of the Alternative DD property must be the full path name of the deployment descriptor file relative to the module's root directory. By convention, the file is in the ALT-INF directory. If this property is not specified, the deployment descriptor file is read directly from the module's JAR file.

### Classpath

The path containing additional classes required by the application but not contained in the module's archive file. The class loader uses this path. Specify the values relative to the root of the EAR file and separate the values with spaces. Absolute values that reference files or directories on the hard drive are ignored. To specify classes that are not in JAR files but are in the root of the EAR file, use a period and forward slash (.). Consider the following example directory structure in which the file myapp.ear contains an EJB module named myejb.jar. Additional classes reside in class1.jar and class2.zip. A class named xyz.class is not packaged in a JAR file but is in the root of the EAR file.

```
myapp.ear/myejb.jar myapp.ear/class1.jar myapp.ear/class2.zip myapp.ear/xyz.class
```

Specify class1.jar class2.zip ./ as the value of the Classpath property. (Name only the directory for class files.)

### Display name

Specifies a short name that is intended to be displayed by GUIs.

### EJB client JAR

Specifies the location of the JAR file containing a subset of deployed classes needed by the client.

### Description

Contains text describing the module.

### Small icon

Specifies a JPEG or GIF file containing a small image (16x16 pixels). The image is used as an icon to represent the module in a GUI.

### Large icon

Specifies a JPEG or GIF file containing a large image (32x32 pixels). The image is used as an icon to represent the module in a GUI.

### Generalizations -- Subtype

Information is not available.

### Generalizations -- Supertype

Information is not available.

### EJB Relationships -- Name

Information is not available.

### EJB Relationships -- Source EJB name

Information is not available.

### EJB Relationships -- Is forward

Information is not available.

### EJB Relationships -- Is navigable

Information is not available.

### Default Data Source -- JNDI name

Specifies the default JNDI name for the data source. This default is used if binding information is not specified in the deployment descriptor for an individual enterprise bean.

**Default Authorization - User ID**

Specifies the default user ID for connecting to an enterprise bean's data store.

**Default Authorization - Password**

Specifies the default password for connecting to an enterprise bean's data store.

## 6.6.5.3: Administering EJB modules with the Web console

Use the Web console to edit the configurations of EJB modules. Because EJB modules are configured, added, and removed as part of installed applications (.ear files), most of their settings displayed in this console are read-only.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**  
-> *application\_server\_name* -> **EJB Containers**  
-> **Installed EJB modules**

The remainder of this article outlines key administrative points.

### Specifying a target database vendor type and schema name for EJB jars containing CMP beans

When installing EJB modules that contain CMP entity beans onto the application server, you can specify a target database vendor type, and a name to use as the table qualifier (schema name). Enter this information on the [Mapping EJB to JNDI Name Mapping](#) panel of the installation wizard.

## 6.6.5.3.2: Viewing deployment descriptor information for EJB modules (read-only)

To view the deployment descriptor information for an EJB module:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **EJB Containers** -> **Installed EJB modules** to display the EJB module view.
2. Click a particular EJB module to view its details on right side of the console.
3. Click the link named **View Deployment descriptor (*ejb-jar.xml*)** where *ejb-jar* is the application name.

The deployment descriptor information will be displayed.

## 6.6.5.3.4: Updating EJB module configurations with the Web console

During this task, you will update the configuration of an existing EJB module that is installed on an application server.

To update an EJB module configuration:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **EJB Container** -> **Installed EJB Modules** where *application\_server\_name* is the name of the existing application server.
2. In the resulting list, click the EJB module that you want to configure.

Its properties will be displayed on the right side of the console.

3. Modify the properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.5.5: Administering EJB modules with Application Assembly Tool

An EJB module is used to assemble one or more enterprise beans into a single deployable unit. The Application Assembly Tool is used to create and edit EJB modules, verify the archive files, and generate deployment code. See the related topics for links to concepts, instructions for creating an EJB module, and field help.

## 6.6.5.5.1: Creating an EJB module

EJB modules can be created by using the property dialog boxes or by using awizard.

- [Using the property dialog boxes](#)
  - [Using the Create EJB Module wizard](#)
- 

### Using the property dialog boxes

The steps for creating an EJB module are as follows:

1. Click **File->New->EJB Module**. Thenavigation pane displays a hierarchical structure used to build the contentsof the module. The icons represent the components, assembly properties,and files for the module. A property dialog box containing generalinformation about the module is displayed in the property pane.
2. By default, the archive file name and the module display name are thesame. It is recommended that you change the display name in theproperty pane.
3. By default, the temporary location of the EJB module isinstallation\_directory/bin. You must specify a newfile name and location by clicking **File->Save**. You must add at least one enterprise bean to the module before savingit. This is a requirement for a valid archive file.
4. Enter values for other properties as needed. View the help for [6.6.5.0.a: Assembly properties for EJB modules](#).
5. Add enterprise beans to the module. You must add at least oneenterprise bean. First, click the icon representing the type of beanbeing added (Session Beans or Entity Beans). There are several ways ofadding beans to a module:
  - Import an existing JAR or EAR file containing enterprise beans.Right-click the icon representing the enterprise bean type and choose**Import**. Click **Browse** to browse the file systemand locate the desired JAR file. When the file is located, click**Open**. Select the JAR file in the left window. Theenterprise beans in the selected JAR file are displayed in the rightwindow. Select the beans to be added and click **Add**.The selected items are displayed in the Selected Components window.Click **OK**. The property dialog box for the enterprise bean isautomatically populated with required values. Click the plus sign(+) next to the icon representing the bean type to verify that thebeans are included in the module.
  - Use a copy-and-paste operation to copy archive files from an existingmodule.
  - Create a new enterprise bean. Right-click the icon representing theappropriate bean type and choose **New**. For entity beans,choose whether the bean has container-managed or bean-managedpersistence. In the property dialog box, browse for and select theclass files that make up the bean. By default, the root directory orarchive is the current archive. If needed, browse the file system forthe directory or archive where the class files reside. After you choosea directory or archive, its file structure is displayed. Expand thestructure and locate the files that you need. Select the file and click**OK**. In the property dialog box, click **OK**.Verify that the beans are added to the module (expand the hierarchy for thebean type in the navigation pane). If there are one or more beans,display the properties for each bean by clicking the bean in the top part ofthe property pane. The corresponding property dialog box is displayedin the bottom part of the pane.
6. Specify properties for each enterprise bean. Expand the hierarchyfor each bean type. Click a bean instance and, if needed, edit or enterproperties for that bean. View the help for [6.6.5.0.1: Assembly properties for entity beans](#) or [6.6.5.0.4: Assembly properties for session beans](#).
7. Add assembly properties for each bean. Click the plus sign(+) next to the bean instance to reveal



property groups. Click the icon representing a group of properties. If properties are already defined (for example, for an imported bean), edit the properties in the property pane. If properties are not defined, right-click the property icon and click **New**. A property dialog box is displayed. Enter values for the properties and click **OK**.

- Specify Environment Entries. View the help for [6.6.34.0.a: Assembly properties for environment entries](#).
  - Specify EJB References. View the help for [6.6.43.0.1: Assembly properties for EJB references](#).
  - Specify Resource References. View the help for [6.6.43.0.2 Assembly properties for resource references](#).
  - Specify Security Role References. View the help for [6.6.43.0.3: Assembly properties for security role references](#).
  - Specify CMP Fields. View the help for [6.6.5.0.2: Assembly properties for CMP fields](#).
  - Specify Method Extensions. View the help for [6.6.5.0.3: Assembly properties for method extensions](#).
8. Add assembly properties for the EJB module. In the navigation pane, right-click each property group's icon. Choose **New** to add new values.
- Specify Security roles. View the help for [6.6.5.0.5: Assembly properties for security roles](#).
  - Specify Method permissions. View the help for [6.6.5.0.6: Assembly properties for method permissions](#).
  - Specify Container transactions. View the help for [6.6.5.0.7: Assembly properties for container transactions](#).
9. Add files needed by the application. Right-click the Files icon in the navigation pane and choose **Add Files**. Click **Browse** to navigate to the desired directory or archive. Click **Select**. If you are adding an entire archive, select the directory that contains the archive. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. These selected files are displayed in the Selected Files window. Relative path names are maintained. When the Selected Files window contains the correct set of files, click **OK**.
10. Click **File->Save** to save the archive.
- 

## Using the Create EJB Module wizard

Use this wizard to create an EJB module. The module can then be used as a stand-alone application, or it can become part of a J2EE application containing other modules. An EJB module consists of one or more enterprise beans. You can use existing EJB JAR files (import them), or create new ones.

During creation of the EJB module, you specify the files for each enterprise bean to be included in the module. You also specify other information about the bean, such as security roles and references to other enterprise beans and to resource connection factories. After defining the enterprise beans to be included in the module, you specify assembly properties that apply to the module as a whole. Both bean and module information are used to create a deployment descriptor.

You can specify either of the following:

- One or more EJB 1.1 JAR files, either created manually or with VisualAge for Java. The enterprise beans must conform to J2EE specifications. The JAR files can be deployed or undeployed.
- One or more EJB 1.0 JAR files. The Application Assembly Tool automatically converts the file to the

EJB 1.1 specification format, but you must specify dependent classpaths if any.

- Enterprise bean class files (not residing in a JAR file).

The wizard creates an EJB module in the file location you specify.

To create an EJB module, click the **Wizards** icon on the tool bar and then click **EJB Module**. Follow the instructions on each panel.

- [Specifying EJB module properties](#)
- [Adding files](#)
- [Specifying EJB Client JAR and classpath](#)
- [Choosing EJB module icons](#)
- [Adding enterprise beans](#)
- [Adding security roles](#)
- [Adding method permissions](#)
- [Adding container transactions](#)
- [Setting additional properties and saving the archive](#)

## Specifying EJB module properties

On the **Specifying EJB Module Properties** panel:

1. Indicate the application to which this module is to be added. If a parent application is not indicated, the module is created as a stand-alone application.
2. Specify a display name for the module. The display name is used by the Application Assembly Tool to identify the module and can be used by other tools.
3. Specify a file name for the module. The file name specifies allocation on your system for the JAR file to be created.
4. Provide a short description of the module (optional).
5. Click **Next**.

## Adding files

On the **Adding Files** panel, specify supplementary files (such as library and utility files) that are to be included in your EJB module. To add or remove files:

1. Click **Add**. Use the file browser to choose one or more files. First, browse for the root directory or archive where the files are located and click **Select**. If you are adding an entire archive, select the directory that contains the archive. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. Relative path names are maintained. The selected files are displayed in the Selected Files window. Click **OK**. The files are displayed in a table on the wizard panel.
2. If you want to remove a file, select the file in the table and then click **Remove**.
3. Continue to add or remove files until you have the correct set of files.
4. Click **Next**.

## Specifying EJB Client JAR and classpath

On the **Specifying EJB Client JAR and Classpath** panel:

1. Specify the EJB Client JAR file. This is the location of the JARfile containing deployed classes needed by a client program for accessing the enterprise beans in this module.
2. Specify the path containing additional classes required by the application. This path is used by the classpath loader.
3. Click **Next**.

## Choosing EJB module icons

On the **Choosing EJB Module Icons** panel, specify icons for your module.

1. Specify the full path name of a file containing a small icon. The icon must be a GIF or JPEG image 16x16 pixels in size.
2. Specify a full path name of a file containing a large icon. The icon must be a GIF or JPEG image 32x32 pixels in size.
3. Click **Next**.

## Adding enterprise beans

The **Adding Enterprise Beans** panel is used to add new enterprise beans, import existing beans, or (if you are modifying this module or make a mistake) remove beans.

To add a new enterprise bean:

1. Click **New**. On the dialog box, choose a bean type (session bean, entity bean with BMP, or entity bean with CMP). Click **OK**.
2. On the **Specifying Enterprise Bean Properties** panel, enter values for the properties of the bean. Click **Browse** to locate the root directory or archive where the bean class files reside. The files are displayed in a window. Locate and click the appropriate file. Click **OK**. View the help for [6.6.5.0.1: Assembly properties for entity beans](#) or [6.6.5.0.4: Assembly properties for session beans](#). Click **Next**.
3. On the **Specifying Container-managed persistence (CMP) fields** panel, define the variables for which the container must manage persistence management. View the help for [6.6.5.0.2: Assembly properties for CMP fields](#). Click **Next**. This panel is visible only for entity beans with CMP.
4. On the **Specifying Specific Enterprise Bean Type Properties** panel, enter values for the properties of the bean. View the help for [6.6.5.0.1: Assembly properties for entity beans](#) or [6.6.5.0.4: Assembly properties for session beans](#). Click **Next**.
5. On the **Choosing Enterprise Bean Icons** panel, specify icons for the bean. Specify the full path name of a file containing a small icon and large icon. The icon must be a GIF or JPEG image (16x16 pixels or 32x32 pixels in size).
6. On the **Adding Environment Entries** panel, enter values for environment entries. Click **Add** and enter a name and type (required). Click **OK**. The entry is displayed in the table on the wizard panel. To remove an entry, select the entry and then click **Remove**. View the help for [6.6.34.0.a: Assembly properties for environment entries](#). Click **Next**.
7. On the **Adding Security Role References** panel, enter values for security role references. Click **Add** to enter a role name. Click **OK**. The role name is displayed in a table on the wizard panel. To remove a role, select the role in the table and then click **Remove**. View the help for [6.6.43.0.3: Assembly properties for security role references](#). Click **Next**.
8. On the **Adding Resource References** panel, enter references for resource connection factories. Click

**Add** to add a reference. You must enter values for a name, type, and authorization mode. Click **OK**. The reference is displayed in a table on the wizard panel. To remove a reference, select the reference in the table and then click **Remove**. View the help for [6.6.43.0.2 Assembly properties for resource references](#). Click **Next**.

9. On the **Adding EJB References** panel, enter values for EJB references. Click **Add** to add a reference. You must enter a value for the name, home interface, remote interface, and type. Click **OK**. The reference is displayed in a table on the wizard panel. To remove a reference, select the reference in the table and then click **Remove**. View the help for [6.6.43.0.1: Assembly properties for EJB references](#).
10. Click **Finish**.
11. Continue to add more enterprise beans as needed.

To import an existing enterprise bean:

1. Click **Import**.
2. Browse the file system to locate the desired archive. The contents of the archive are displayed in a window. Select one or more JAR files. The enterprise beans in the JAR file are displayed in the right window. Select an enterprise bean and then click **Add**. The enterprise beans are added to the Selected Components window. Click **OK**.

To remove an enterprise bean, select the enterprise bean in the table and then click **Remove**.

Continue adding and removing enterprise beans as necessary. Click **Next**.

## Adding security roles

On the **Adding Security Roles** panel:

1. Click **Add**. Type a role name and, optionally, type a description. Click **OK**. The role name is displayed in a table on the wizard panel. View the help for [6.6.5.0.5: Assembly properties for security roles](#).
2. Continue to add security roles as needed. If you need to remove a role, select the role in the table and then click **Remove**.
3. Click **Next**.

## Adding method permissions

On the **Adding Method Permissions** panel, indicate which security roles are permitted to invoke which methods.

1. To add method permissions, click **Add**. Enter a name for the method permission. View the help for [6.6.5.0.6: Assembly properties for method permissions](#).
2. Click **Add** next to the table of methods. Locate the method in the JAR file, select it, and then click **OK**.
3. Click **Add** next to the table of security roles. Select the appropriate security role and click **OK**.
4. Verify the information and click **OK**. The method permission is displayed in a table on the wizard panel.
5. To add multiple method permissions, click **Add** on the wizard panel and repeat the process.
6. Continue to add and remove methods and corresponding security roles as needed. If you need to remove a method permission, select the item and then click **Remove**.
7. Click **Next**.

## Adding container transactions

On the **Adding Container Transactions** panel, indicate transaction attributes for the methods of the enterprise

bean.

1. To add a container transaction, click **Add**. Enter a name and choose a transaction attribute from the menu. View the help for [6.6.5.0.7: Assembly properties for container transactions](#).
2. Click **Add** to choose which methods are to be governed by this attribute. Locate the method in the JAR file, select it, and then click **OK**.
3. Verify the information and click **OK**. The container transaction is displayed in a table on the wizard panel.
4. To add multiple container transactions, click **Add** on the wizard panel and repeat the process.
5. Continue adding or removing container transactions as needed. If you need to remove a container transaction, select the item and then click **Remove**.

## Setting additional properties and saving the archive

Click **Finish** to complete the wizard. To change settings for properties, click **Back** to return to the appropriate panel. Make any needed changes, and then click **Finish**.

After you click **Finish**, the contents of the archive are displayed in an Application Assembly Tool window. Review the contents in the navigation pane. You can continue adding or modifying properties as needed. For example, you can add binding information. When you are finished editing the archive, click **File->Save** to save the archive file.

## 6.6.7: Administering Web containers (overview)

A Web container configuration provides information about the applicationserver component that handles servlet requests forwarded bythe Web server. The administratorspecifies Web container properties including:

- Application server on which the Web container runs
- Number and type of connections between the Web server and Web container
- Port on which the Web container listens

## 6.6.7.0: Web container properties

Web containers contain other resource types, whose properties are listed in separate property reference files. If you do not find a property in the following list, see below for [links to the property references of other resource types comprising Web containers](#).

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Allow thread allocation beyond maximum



Allows the number of threads to increase beyond the maximum size configured for the thread pool

### Application Server



The application server associated with this Web container

### Cache Size



A positive integer defining the maximum number of entries the cache will hold. Values are usually in the thousands, with no maximum or minimum.

### Can Be Grown



Allows the number of threads to increase beyond the maximum size configured for the pool

### Default Priority



The default priority for servlets that can be cached. It determines how long an entry will stay in a full cache. The recommended value is 1.

### Dynamic Properties



A set of name-value pairs for configuring properties beyond those displayed in the interface

### Enable Dynamic Cache



### or Enable Servlet Caching



Enable the servlet and JSP dynamic JNDI caching feature

### External Cache Groups



For servlets that can be cached, specifies the external groups to which their entries should be sent

### Enable Servlet Caching



See Enable Dynamic Cache

### HTTP Transports



### or Transport



The HTTP transports associated with this Web container. See also [transport properties](#)

### Inactivity Timeout



### or Thread Inactivity Timeout



The period of time after which a thread should be reclaimed due to inactivity

### **Installed Web Modules**

The Web modules that are installed into the Web container of this server

### **Maximum Size** **or Maximum Thread Size**

The maximum number of threads to allow in the pool

### **Minimum Size** **or Minimum Thread Size**

The minimum number of threads to allow in the pool

### **Name (External Cache Group)**

See External Cache Groups

### **Node**

The node with which this Web container is associated

### **Session Manager**

The Session Manager associated with this Web container. See also [Session Manager properties](#)

### **Thread Inactivity Timeout**

See Inactivity Timeout

### **Thread Pool**

The thread pool settings for the Web container

### **Transport**

See HTTP Transports

### **Type**

Only shared external cache groups are supported at this time

## **Additional properties related to Web containers**

Web containers contain other resource types, whose properties are listed in separate property reference files. If you do not find the property in the above list ...

See also the:

- [application server properties](#)
- [HTTP Transport properties](#)

For Advanced Single Server Edition, see also the:

- [Session Manager properties](#)
- [Web module properties](#)



## 6.6.7.3: Administering Web containers with the Web console

Use the Web console to edit the configurations of Web containers, which are responsible for provided needed service to running Web modules and their contained servlets and JSP files. Each application server runtime has one logical Web container, which you can modify but not create or remove.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**  
-> *application\_server\_name* -> **Web Containers**

## 6.6.7.3.4: Updating Web container configurations with the Web console

During this task, you will update the configuration of an existing Web container, which is part of an application server configuration.

To update a Web container configuration:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Web Container** where *application\_server\_name* is the name of the existing application server.
2. Click Web Container. Its properties will be displayed on the rightside of the console.
3. Modify the properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.8: Administering Web modules (overview)

### Classpath considerations

An important classpath-related setting to note is the Module Visibility. This application server setting impacts the portability of applications and standalone modules from other WebSphere Application Server versions and editions. If your existing module does not run as-is when you transfer it to Version 4.0, you might need to reassemble an existing module or change the module visibility setting.

See [the information on setting classpaths](#) for a full discussion of classpath considerations. See the [applicationserver property reference](#) for information about the module visibility setting.

### Identifying a welcome page for the Web application


The default welcome page for your Web application is assumed to be named index.html. For example, if you have an application with a Web path of: /webapp/myapp

then the default page named index.html can be implicitly accessed using the following URL:

`http://hostname/webapp/myapp`

To identify a different welcome page, modify the properties of the Web module when you are assembling it. See the article about [assembling Web modules with the Application Assembly Tool \(article 6.6.8.5\)](#).

### Web application URLs are now case-sensitive on all operating systems

 Please note that in Version 4.0.x, Webapplication URLs are now case-sensitive on all operating systems, for security and consistency.

For example, suppose you have a Web client application that runs successfully on Version 3.5.x. When running the same application on Version 4.0, you encounter an error that the welcome page (typically index.html), or HTML files to which it refers, cannot be found:

Error 404: File not found: Banner.html      Error 404: File not found: HomeContent.html

Suppose the content of the index page is as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"><HTML><TITLE>Insurance Home Page</TITLE>
<frameset rows="18,80">      <frame src="Banner.html"                      name="BannerFrame"
SCROLLING=NO>      <frame src="HomeContent.html"                      name="HomeContentFrame">
</frameset></HTML>
```

but the actual file names in \WebSphere\AppServer\installedApps\... directory in which the application is deployed are:

`banner.html``homecontent.html`

To correct the problem, modify the index.html file to change the names "Banner.html" and "HomeContent.html" to "banner.html" and "homecontent.html" to match the names of the files in the deployed application.

## 6.6.8.0: Web module properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Application or  Application Ref **

The application installation binding within which the module-to-server installation binding is contained. This is typically the logical name of the enterprise application you configured to contain this Web module.

**Context Root **

The context root of the Web application contained in an enterprise application.

The context root is combined with the defined servlet mapping (from the WAR file) to compose the full URL that users type to access the servlet. For example, if the context root is /gettingstarted and the servlet mapping is MySession, then the URL is http://host:port/gettingstarted/MySession.

**Execution State **

The state that you would like the Web module to be in, the next time the product is stopped and started again

**Name  or Module Name **

An administrative name for the Web module

**Server **

The application server on which the Web module is installed

**URI **

A URI that, when resolved relative to the application URL, specifies the location of the module archive on a file system. The URI must match the URI of a ModuleRef URI in the deployment descriptor of an application if the module was packaged as part of a deployed application (EAR).

## 6.6.8.0.1: Assembly properties for Web components

### **Component name (Required, String)**

Specifies the name of the servlet or JavaServer Pages<sup>(TM)</sup> (JSP)file. This name must be unique within the Web module.

### **Display name**

Specifies a short name that is intended to be displayed by GUIs.

### **Description**

Contains a description of the servlet or JSP file.

### **Component type**

Specifies the type of Web component. Valid values are servlet orJSP file.

### **Class name (Required, String)**

Specifies the full path name for the servlet's class.

### **JSP file (Required, String)**

Specifies the full path name for the JSP file.

### **Load on startup**

Indicates whether this servlet is to be loaded at the startup of the Webapplication. The default is false (the checkbox is notselected). Also specifies a positive integer indicating the order inwhich the servlet is to be loaded. Lower integers are loaded beforehigher integers. If no value is specified, or if the value specified isnot a positive integer, the container is free to load the servlet at any timein the startup sequence.

### **Small icon**

Specifies a JPEG or GIF file containing a small image (16x16pixels). The image is used as an icon to represent the Web component ina GUI.

### **Large icon**

Specifies a JPEG or GIF file containing a large image (32x32pixels). The image is used as an icon to represent the Web component ina GUI.

## 6.6.8.0.2: Assembly properties for initialization parameters

Initialization parameters are sent to a servlet in its `HttpConfig` object when the servlet is first started.

### **Parameter name (Required, String)**

Specifies the name of an initialization parameter.

### **Parameter value (Required, String)**

Specifies the value of the initialization parameter.

### **Description**

Contains text describing the use of the parameter.

## 6.6.8.0.3: Assembly properties for page lists

Page lists allow you to avoid hardcoding URLs in servlets and JSPfiles. A page list specifies the location where a request is to be forwarded, but automatically tailors that location depending on the MIME type of the servlet. These properties allow you to specify a markup language and an associated MIME type. For the given MIME type, you also specify a set of pages to invoke. For example, if you define a markup language named VXML and associate it with a vxml MIME type, you can then define Page names and URIs to be invoked for that particular MIME type. The Page names end in .page and are the same name for all markup languages. However, the URIs are set to point to files that are particular to the given MIME type. For example, if a page is called ShowAccount.page and is in a markup language named VXML, the URI is ShowAccountVXML.jsp. In a markup language named HTML, the URI is ShowAccountHTML.jsp. When the servlet refers to ShowAccount.page, the actual file to which the request maps depends on the servlet's MIME type.

### **Name**

Specifies the name of the markup language--for example, HTML, WML, and VXML.

### **MIME Type**

Specifies the MIME type of the markup language--for example, text/html and text/x-vxml.

### **Error Page**

Specifies the name of an error page.

### **Default Page**

Specifies the name of a default page.

### **Pages - Name**

Specifies the name of the page to be served, for example, StockQuoteRequest.page.

### **Pages - URI**

Specifies the URI of the page to be served, for example, examples/StockQuoteHTMLRequest.jsp.

## 6.6.8.0.4: Assembly properties for security constraints

Security constraints declare how Web content is to be protected. These properties associate security constraints with one or more Web resource collections. A constraint consists of a Web resource collection, an authorization constraint, and a user data constraint.

- A Web resource collection is a set of resources (URL patterns) and HTTP methods on those resources. All requests that contain a request path that matches the URL pattern described in the Web resource collection is subject to the constraint. If no HTTP methods are specified, then the security constraint applies to all HTTP methods.
- An authorization constraint is a set of roles that users must be granted in order to access the resources described by the Web resource collection. If a user who requests access to a specified URI is not granted at least one of the roles specified in the authorization constraint, the user is denied access to that resource.
- A user data constraint indicates that the transport layer of the client/server communications process must satisfy the requirement of either guaranteeing content integrity (preventing tampering in transit) or guaranteeing confidentiality (preventing reading while in transit).

If multiple security constraints are specified, the container uses the "first match wins" rule when processing a request to determine what authentication method to use, or what authorization to allow.

### **Security constraint name**

Specifies the name of the security constraint.

### **Authorization Constraints - Roles**

Specifies the user roles that are permitted access to this resource collection.

### **Authorization Constraints - Description**

Contains a description of the authorization constraints.

### **User Data Constraints - Transport guarantee**

Indicates how data communicated between the client and the server is to be protected. Specifies that the protection for communications between the client and server is None, Integral, or Confidential. None means that the application does not require any transport guarantees. Integral means that the application requires that the data sent between the client and the server must be sent in such a way that it cannot be changed in transit. Confidential means that the application requires that the data must be transmitted in a way that prevents other entities from observing the contents of the transmission. In most cases, Integral or Confidential indicates that the use of SSL is required.

### **User Data Constraints - Description**

Contains a description of the user data constraints.



## 6.6.8.0.5: Assembly properties for Web resource collections

A Web resource collection defines a set of URL patterns (resources) and HTTP methods belonging to the resource. HTTP methods handle HTTP-based requests, such as GET, POST, PUT, and DELETE. A URL pattern is a partial Uniform Resource Locator that acts as a template for matching the pattern with existing full URLs in an attempt to find a valid file.

### **Web resource name (Required, String)**

Specifies the name of a Web resource collection.

### **Web resource description**

Contains a description of the Web resource collection.

### **HTTP methods**

Specifies the HTTP methods to which the security constraint applies. If no HTTP methods are specified, then the security constraint applies to all HTTP methods. The valid values are GET, POST, PUT, DELETE, HEAD, OPTIONS, and TRACE.

### **URL pattern**

Specifies URL patterns for resources in a Web application. All requests that contain a request path that matches the URL pattern are subject to the security constraint.

## 6.6.8.0.8: Assembly properties for context parameters

A servlet context defines a server's view of the Web application within which the servlet is running. The context also allows a servlet to access resources available to it. Using the context, a servlet can log events, obtain URL references to resources, and set and store attributes that other servlets in the context can use. These properties declare a Web application's parameters for its context. They convey setup information, such as a webmaster's e-mail address or the name of a system that holds critical data.

### **Parameter name (Required, String)**

Specifies the name of a parameter--for example, `dataSourceName`.

### **Parameter value (Required, String)**

Specifies the value of a parameter--for example, `jdbc/sample`.

### **Description**

Contains a description of the context parameter.

## 6.6.8.0.9: Assembly properties for error pages

Error page locations allow a servlet to find and serve a URI to a client based on a specified error status code or exception type. These properties are used if the error handler is another servlet or JSP file. The properties specify a mapping between an error code or exception type and the path of a resource in the Web application.

The container examines the list in the order that it is defined, and attempts to match the error condition by status code or by exception class. On the first successful match of the error condition, the container serves back the resource defined in the Location property.

### **Error Code**

Indicates that the error condition is a status code.

### **Error Code (Required, String)**

Specifies an HTTP error code, for example, 404.

### **Exception**

Indicates that the error condition is an exception type.

### **Exception type name (Required, String)**

Specifies an exception type.

### **Location (Required, String)**

Contains the location of the error-handling resource in the Web application.

## 6.6.8.0.10: Assembly properties for MIME mapping

A Multi-Purpose Internet Mail Extensions (MIME) mapping associates a filename extension with a type of data file (text, audio, image). These properties allow you to map a MIME type to a file name extension.

### **Extension (Required, String)**

Specifies a file name extension, for example, .txt.

### **MIME type (Required, String)**

Specifies a defined MIME type, for example, text/plain.

## 6.6.8.0.11: Assembly properties for servlet mapping

A servlet mapping is a correspondence between a client request and a servlet. Servlet containers use URL paths to map client requests to servlets, and follow the URL path-mapping rules as specified in the JavaServlet specification. The container uses the URI from the request, minus the context path, as the path to map to a servlet. The container chooses the longest matching available context path from the list of Webapplications that it hosts.

### **URL pattern (Required, String)**

Specifies the URL pattern of the mapping. The URL pattern must conform to the Servlet specification. The following syntax must be used:

- A string beginning with a slash character (/) and ending with the slash and asterisk characters (/\*) is used as a path mapping.
- A string beginning with the characters \*. is used as an extension mapping.
- All other strings are used as exact matches only.
- A string containing only the slash character (/) indicates that the servlet specified by the mapping becomes the default servlet of the application. In this case, the servlet path is the request URI minus the context path, and the path info is null.

### **Servlet (Required, String)**

Specifies the name of the servlet associated with the URL pattern.

## 6.6.8.0.12: Assembly properties for tag libraries

Java ServerPages (JSP) tag libraries contain classes for common tasks such as processing forms and accessing databases from JSP files.

### **Tag library file name (Required, String)**

Specifies a file name relative to the location of the web.xml document, identifying a tag library used in the Web application.

### **Tag library location (Required, String)**

Contains the location, as a resource relative to the root of the Web application, where the Tag Library Definition file for the tag library can be found.

## 6.6.8.0.13: Assembly properties for welcome files

A Welcome file is an entry-point file (for example, index.html) for a group of related HTML files. Welcome files are located by using a group of suggested partial URIs. A Welcome file is an ordered list of partial URIs that the container uses to attempt to find a valid file when the initial URI cannot be found. The container appends these partial URIs to the requested URI to arrive at a valid URI. For example, the user can define a Welcome file of index.html so that a request to a URL such as host:port/webapp/directory (where directory is a directory entry in the WAR file that is not mapped to a servlet or JSP file) can be fulfilled.

### **Welcome file (Required, String)**

The Welcome file list is an ordered list of partial URLs with no trailing or leading slash characters (/). The Web server appends each file in the order specified and checks whether a resource in the WAR file is mapped to that request URI. The container forwards the request to the first resource in the WAR that matches.

## 6.6.8.0.14: Assembly properties for MIME filters

Filters transform either the content of an HTTP request or response and can also modify header information. MIME filters forward HTTP responses with a specified MIME type to one or more designated servlets for further processing.

### **MIME Filter - Target**

Specifies the target virtual host for the servlets.

### **MIME Filter - Type**

Specifies the MIME type of the response that is to be forwarded.



## 6.6.8.0.15: Assembly properties for JSP attributes

JSP attributes are used by the servlet that implements JSP processing behavior.

### **JSP Attribute (Name)**

Specifies the name of an attribute.

### **JSP Attribute (Value)**

Specifies the value of an attribute.

## 6.6.8.0.16: Assembly properties for file-serving attributes

File serving allows a Web application to serve static file types, such asHTML. File-serving attributes are used by the servlet that implementsfile-serving behavior.

### **File Serving Attribute (Name)**

Specifies the name of an attribute.

### **File Serving Attribute (Value)**

Specifies the value of an attribute.

## 6.6.8.0.17: Assembly properties for invoker attributes

Invoker attributes are used by the servlet that implements the invocation behavior.

### **Invoker Attribute (Name)**

Specifies the name of an attribute.

### **Invoker Attribute (Value)**

Specifies the value of an attribute.

## 6.6.8.0.18: Assembly properties for servlet caching configurations

Dynamic caching can be used to improve the performance of servlet and JavaServer Pages (JSP) files by serving requests from an in-memory cache. Cache entries contain the servlet's output, results of the servlet's execution, and metadata.

The properties on the General tab define a cache group and govern how long an entry remains in the cache. The properties on the ID Generation tab define how cache IDs are built and the criteria used to cache or invalidate entries. The properties on the Advanced tab define external cache groups and specify custom interfaces for handling servlet caching.

### **Caching group name (Required, String)**

Specifies a name for the group of servlets or JSP files to be cached.

### **Priority**

An integer that defines the default priority for servlets that are cached. The default value is 1. Priority is an extension of the Least Recently Used (LRU) caching algorithm. It represents the number of cycles through the LRU algorithm that an entry is guaranteed to stay in the cache. The priority represents the length of time that an entry remains in the cache before being eligible for removal. On each cycle of the algorithm, the priority of an entry is decremented. When the priority reaches zero, the entry is eligible for invalidation. If an entry is requested while in the cache, its priority is reset to the priority value. Regardless of the priority value and the number of requests, an entry is invalidated when its timeout occurs. Consider increasing the priority of a servlet or JSP file when it is difficult to calculate the output of the servlet or JSP file or when the servlet or JSP file is executed more often than average. Priority values should be low. Higher values do not yield much improvement but use extra LRU cycles. Use timeout to guarantee the validity of an entry. Use priority to rank the relative importance of one entry to other entries. Giving all entries equal priority results in a standard LRU cache that increases performance significantly.

### **Timeout**

Specifies the length of time, in seconds, that a cache entry is to remain in the cache after it has been created. When this time elapses, the entry is removed from the cache. If the timeout is zero or a negative number, the entry does not time out. It is removed when the cache is full or programmatically, from within an application.

### **Invalidate only**

Specifies that invalidations for a servlet are to take place, but that no caching is to be performed for the servlet. For example, this property can be used to prevent caching of control servlets. Control servlets treat HTTP requests as commands and execute those commands. By default, this checkbox is not selected.

### **Caching group members**

Specifies the names of the servlets or JSP files to be cached. The URIs are determined from the servlet mappings.

### **Use URIs for cache ID building**

Specifies whether or not the URI of the requested servlet is to be used to create a cache ID. By default, URIs are used.

### **Use specified string**

Specifies a string representing a combination of request and session variables that are to be used to create cache IDs. (This property defines request and session variables, and the cache uses the values of these variables to create IDs for the entries.)

### **Variables - ID**

The name of a request parameter, request attribute, session parameter, or cookie.

### **Variables - Type**

Indicates the type of variable specified in the ID field. The valid values are Request parameter, Request attribute, Session parameter, or Cookie.

### **Variables - Method**

The name of a method in the request attribute or session parameter. The output of this method is used to generate cache entry IDs. If this value is not specified, the toString method is used by default.

### **Variables - Data ID**

Specifies a string that, combined with the value of the variable, generates a group name for the cache entry. The cache entry is placed in this group. This group can later be invalidated.

### **Variables - Invalidate ID**

Specifies a string that is combined with the value of the variable on the request or session to form a group name. The cache invalidates the group name.

### **Required**

Indicates whether a value must be present in the request. If this checkbox is selected, and either the request parameter, request attribute, or session parameter is not specified, or the method is not specified, the request is not cached.

### **External cache groups - Group name**

Specifies the name of the external cache group to which this servlet will be published.

### **ID generator**

Specifies a user-written interface for handling parameters, attributes, and sessions. The value must be a full package and class name of a class extending `com.ibm.websphere.servlet.cache.IdGenerator`. The properties specified in the Application Assembly Tool will still be used and passed to the `IdGenerator` in the `initialize` method inside `com.ibm.websphere.servlet.cache.CacheConfig` object.

### **Meta data generator**

Specifies a user-written interface for handling invalidation, priority levels, and external cache groups. The value must be the full package and class name of a class extending `com.ibm.websphere.servlet.cache.MetaDataGenerator`. The properties specified in the Application Assembly Tool will still be used and passed to the `MetaDataGenerator` in the `initialize` method inside `com.ibm.websphere.servlet.cache.CacheConfig` object.

## 6.6.8.0.aa: Assembly properties for Web modules

### File name (Required, String)

Specifies the file name of the Web module, relative to the top level of the application package.

### Alternative DD

Specifies the file name for an alternative deployment descriptor file to use instead of the original deployment descriptor file in the module's JAR file. This file is the postassembly version of the deployment descriptor file. (The original deployment descriptor file can be edited to resolve dependencies and security information. Directing the use of the alternative deployment descriptor allows you to keep the original deployment descriptor file intact). The value of the Alternative DD property must be the full path name of the deployment descriptor file relative to the module's root directory. By convention, the file is in the ALT-INF directory. If this property is not specified, the deployment descriptor file is read directly from the module's JAR file.

### Context root (Required, String)

Specifies the context root of the Web application. The context root is combined with the defined servlet mapping (from the WAR file) to compose the full URL that users type to access the servlet. For example, if the context root is /gettingstarted and the servlet mapping is MySession, then the URL is http://host:port/gettingstarted/MySession.

### Classpath

Specifies the full class path for the Web application. Specify the values relative to the root of the EAR file and separate the values with spaces. Absolute values that reference files or directories on the hard drive are ignored. To specify classes that are not in JAR files but are in the root of the EAR file, use a period and forward slash (/). Consider the following example directory structure in which the file myapp.ear contains a Web module named mywebapp.war. Classes reside in class1.jar and class2.zip. A class named xyz.class is not packaged in a JAR file but is in the root of the EAR file.

`myapp.ear/mywebapp.war myapp.ear/class1.jar myapp.ear/class2.zip myapp.ear/xyz.class`

Specify `class1.jar class2.zip ./` as the value of the Classpath property. (Name only the directory for class files.)

### Display name

Specifies a short name that is intended to be displayed by GUIs.

### Description

Contains a description of the Web module.

### Distributable

Specifies that this Web application is programmed appropriately to be deployed into a distributed servlet container.

### Small icon

Specifies a JPEG or GIF file containing a small image (16x16 pixels). The image is used as an icon to represent the module in a GUI.

### Large icon

Specifies a JPEG or GIF file containing a large image (32x32 pixels). The image is used as an icon to represent the module in a GUI.

### Session configuration

Indicates that session configuration information is present. Checking this box makes the Session timeout property editable.

### Session timeout

Specifies a time period, in seconds, after which a client is considered inactive. The default value is zero, indicating that the session timeout never expires.

### Login configuration -- Authentication method

Specifies an authentication method to use. As a prerequisite to gaining access to any Web resources protected by an authorization constraint, a user must authenticate by using the configured mechanism. A Web application can authenticate a user to a Web server by using one of the following mechanisms: HTTP basic authentication, HTTP digest authentication, HTTPS client authentication, and form-based authentication.

- HTTP basic authentication is not a secure protocol because the user password is transmitted with a simple Base64 encoding and the target server is not authenticated. In basic authentication, the Web server requests a Web client to authenticate the user and passes a string called the realm of the request in which the user is to be authenticated.

- HTTP digest authentication transmits the password in encrypted form.
- HTTPS client authentication uses HTTPS (HTTP over SSL) and requires the user to possess a public key certificate.
- Form-based authentication allows the developer to control the appearance of login screens.

The Login configuration properties are used to configure the authentication method that should be used, the realm name that should be used for HTTP basic authentication, and the attributes that are needed by the form-based login mechanism. Valid values for this property are Unspecified, Basic, Digest, Form, and Client certification.

Note: HTTP digest authentication is not supported as a login configuration in this product. Also, not all login configurations are supported in all of the product's global security authentication mechanisms (Local Operating system, LTPA, and custom pluggable user registry). HTTP basic authentication and form-based login authentication are the only authentication methods supported by the Local Operating system user registry. Because Advanced Single Server Edition uses the local operating system as the user registry for authentication, it can only support these two login methods. LTPA and the custom pluggable user registry are capable of supporting HTTP basic authentication, form-based login, and HTTPS client authentication. LTPA and the custom pluggable user registry is available only in Advanced Edition.

### **Login configuration -- Realm name**

Specifies the realm name to use in HTTP basic authorization. It is based on a user name and password, sent as a string (with a simple Base64 encoding). An HTTP realm is a string that allows URIs to be grouped together. For example, if a user accesses a secured resource on a Web server within the "finance realm," subsequent access to the same or different resource within the same realm does not result in a repeat prompt for a user ID and password.

### **Login configuration -- Login page (Required, String)**

Specifies the location of the login form. If form-based authentication is not used, this property is disabled.

### **Form Login Config -- Error page (Required, String)**

Specifies the location of the error page. If form-based authentication is not used, this property is disabled.

### **Reload interval**

Specifies a time interval, in seconds, in which the Web application's file system is scanned for updated files. The default is 0 (zero).

### **Reloading enabled**

Specifies whether file reloading is enabled. The default is false.

### **Default error page**

Specifies a file name for the default error page. If no other error page is specified in the application, this error page is used.

### **Additional classpath**

Specifies an additional class path that will be used to reference resources outside of those specified in the archive. Specify the values relative to the root of the EAR file and separate the values with spaces. Absolute values that reference files or directories on the hard drive are ignored. To specify classes that are not in JAR files but are in the root of the EAR file, use a period and forward slash (. /). Consider the following example directory structure in which the file myapp.ear contains a Web module named mywebapp.war. Additional classes reside in class1.jar and class2.zip. A class named xyz.class is not packaged in a JAR file but is in the root of the EAR file.

`myapp.ear/mywebapp.war myapp.ear/class1.jar myapp.ear/class2.zip myapp.ear/xyz.class`

Specify `class1.jar class2.zip ./` as the value of the Additional classpath property. (Name only the directory for .class files.)

### **File serving enabled**

Specifies whether file serving is enabled. File serving allows the application to serve static file types, such as HTML and GIF. File serving can be disabled if, for example, the application contains only dynamic components. The default value is true.

### **Directory browsing enabled**

Specifies whether directory browsing is enabled. Directory browsing allows the application to browse disk directories. Directory browsing can be disabled if, for example, you want to protect data. The default value is true.

### **Serve servlets by classname**

Specifies whether a servlet can be served by requesting its classname. Usually, servlets are served only through a URI reference. The class name is the actual name of the servlet on disk. For example, a file named SnoopServlet.java compiles

intoSnoopServlet.class. (This is the class name.)SnoopServlet.class is normally invoked by specifying snoop in theURI. However, if Serve Servlets by Classname is enabled, the servlet isinvoked by specifying SnoopServlet. The default value is true.

**Virtual hostname**

Specifies a virtual host name. A virtual host is a configurationenabling a single host machine to resemble multiple host machines. Thisproperty allows you to bind the application to a virtual host in order toenable execution on that virtual host.



## 6.6.8.3: Administering Web modules with the Web console

Use the Web console to edit the configurations of Web modules. Because Web modules are configured, added, and removed as part of installed applications (.ear files), most of their settings displayed in this console are read-only.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**  
-> *application\_server\_name* -> **Web Containers**  
-> **Installed Web modules**

## 6.6.8.3.1: Precompiling JSP files for Web modules of an application with the Web console

You can precompile the JSP files in a Web module either while you are installing the Web module (or the application containing it), or after installation.

To precompile the JSP files during application installation, follow the [instructions for installing an application](#).

To precompile the JSP files of an already installed application, follow the instructions for [mapping virtual hosts to Web modules](#) task.

In either case, you will end up at the "Mapping virtual hosts to Web modules" panel of the application installation wizard, from which you can specify to precompile JSP files.

## 6.6.8.3.2: Viewing deployment descriptor information for Web modules (read-only)

To view the deployment descriptor information for a Web module:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Web Containers** -> **Installed Web modules** to display the Web module view.
2. Click a particular Web module to view its details on right side of the console.
3. Click the link named **View Deployment descriptor (*web.xml*)** where *web* is the application name.

The deployment descriptor information will be displayed.

## 6.6.8.3.4: Updating Web module configurations with the Web console

During this task, you will update the configuration of an existing Web module installed on an application server.

To update a Web module configuration:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Web Container** -> **Installed Web Modules** where *application\_server\_name* is the name of the existing application server.
2. In the list of installed Web modules, click Web module that you want to configure. Its properties will be displayed on the rightside of the console.
3. Modify the properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.8.5: Administering Web modules with Application Assembly Tool

A Web module is used to assemble one or more servlets, JavaServer Pages(JSP) files, Web pages, and other static content into a single deployable unit. The Application Assembly Tool is used to create and edit modules, verify the archive files, and generate deployment code. See the related topics for links to concepts, instructions for creating a Web module, and field help.

## 6.6.8.5.1: Creating a Web module

Web modules can be created by using property dialog box or by using awizard.

- [Using the property dialog boxes](#)
  - [Using the Create Web Module wizard](#)
- 

### Using the property dialog boxes

The steps for creating a Web module are as follows:

1. Click **File->New->Web Module**. Thenavigation pane displays a hierarchical structure used to build the contents of the module. The icons represent the components, assembly properties, and files for the module. A property dialog box containing general information about the module is displayed in the property pane.
2. By default, the archive file name and the module display name are the same. It is recommended that you change the display name in the property pane. Enter values for other properties as needed. View the help for [6.6.8.0.a: Assembly properties for Web modules](#).
3. By default, the temporary location of the Web module is `installation_directory/bin`. You must specify a new file name and location by clicking **File->Save**. You must first add at least one Web component (servlet or JSP file) before saving the archive.
4. Add Web components (servlets or JSP files) to the module. You must add at least one Web component. There are several ways of adding components to a module:
  - Import an existing WAR file containing Web components. In the navigation pane, right-click the Web Components icon and choose **Import**. Click **Browse** to browse the file system and locate the desired archive file. When the file is located, click **Open**. The Web applications in the selected archive file are displayed. Select a Web application. Its Web components are displayed in the right window. Select the servlets or JSP files to be added and click **Add**. The components are displayed in the Selected Components window. Click **OK**. The properties associated with the archive are also imported and the property dialog boxes are automatically populated with values. Double-click the WebComponents icon to verify that the servlets or JSP files are included in the module.
  - Use a copy-and-paste operation to copy archive files from an existing module.
  - Create a new Web component. Right-click the Web Components icon and choose **New**. Enter a component name and choose a component type. Browse for and select the class files. By default, the root directory or archive is the current archive. If needed, browse the file system for the directory or archive where the class files reside. After you choose a directory or archive, its file structure is displayed. Expand the structure and locate the files that you need. Select the file and click **OK**. In the New WebComponent property dialog box, click **OK**. Verify that the Webcomponent has been added to the module (double-click the Web Components icon in the navigation pane). The Web components are also listed in the top portion of the property pane. Click the component to view its corresponding property dialog box in the bottom portion of the pane.
5. Enter properties for the Web component as needed. View the help for [6.6.8.0.1: Assembly properties for Web components](#).
6. Enter assembly properties for each Web component. Click the plus sign (+) next to the component instance to reveal property groups. Right-click each property group's icon. Choose **New** to add new values, or edit existing values in the property pane.
  - Specify Security Role References. View the help for [6.6.43.0.3: Assembly properties for security](#)

role references.

- Specify Initialization Parameters. View the help for [6.6.8.0.2: Assembly properties for initialization parameters](#).
  - Specify Page List Extensions. View the help for [6.6.8.0.3: Assembly properties for page lists](#).
7. Specify additional properties for the Web module. Right-click each property group's icon. Choose **New** to add new values, or edit existing values in the property pane.
- Specify Security Constraints. View the help for [6.6.8.0.4: Assembly properties for security constraints](#). If you add a security constraint, you must add at least one Web resource collection.
  - Specify Web resource collections, HTTP methods, and URL patterns. View the help for [6.6.8.0.5: Assembly properties for Web resource collections](#).
  - Specify Context Parameters. View the help for [6.6.8.0.8: Assembly properties for context parameters](#).
  - Specify EJB references. View the help for [6.6.43.0.1: Assembly properties for EJB references](#).
  - Specify Environment Entries. View the help for [6.6.34.0.a: Assembly properties for environment entries](#).
  - Specify Error Pages. View the help for [6.6.8.0.9: Assembly properties for error pages](#).
  - Specify MIME Mappings. View the help for [6.6.8.0.10: Assembly properties for MIME mapping](#).
  - Specify Resource References. View the help for [6.6.43.0.2: Assembly properties for resource references](#).
  - Specify Security Roles. View the help for [6.6.5.0.5: Assembly properties for security roles](#).
  - Specify Servlet Mapping. View the help for [6.6.8.0.11: Assembly properties for servlet mapping](#).
  - Specify Tag Libraries. View the help for [6.6.8.0.12: Assembly properties for tag libraries](#).
  - Specify Welcome Files. View the help for [6.6.8.0.13: Assembly properties for welcome files](#).
8. Optionally, specify assembly property extensions. In the navigation pane, double-click the icon for Assembly Property Extensions.
- Specify MIME filters. View the help for [6.6.8.0.14: Assembly properties for MIME filters](#).
  - Specify JSP Attributes. View the help for [6.6.8.0.15: Assembly properties for JSP attributes](#).
  - Specify File Serving Attributes. View the help for [6.6.8.0.16: Assembly properties for file-serving attributes](#).
  - Specify Invoker Attributes. View the help for [6.6.8.0.17: Assembly properties for invoker attributes](#).
  - Specify Servlet Caching Configurations. View the help for [6.6.8.0.18: Assembly properties for servlet caching configurations](#).
9. Add any other files needed by the application. In the navigation pane, click the plus sign (+) next to the Files icon. Right-click **Add Class Files**, **Add JAR Files**, or **Add Resource Files**. Choose **Add Files**. Click **Browse** to navigate to the desired directory or archive and then click **Select**. If you are adding an entire archive, select the directory that contains the archive. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. Relative path names are maintained. When the Selected Files window contains the correct set of files, click **OK**.

10. Click **File->Save** to save the archive.
- 

## Using the Create Web Module wizard

Use this wizard to create a Web module. The module can then be used as a stand-alone application, or it can become part of a J2EE application containing other modules. A Web module consists of one or more servlets and JSP files. You can use existing archives (by importing them), or create new ones.

During creation of the Web module, you specify the files for each servlet or JSP file to be included in the module. You also specify assembly properties for the servlets and JSP files, such as references to enterprise beans and resource connection factories, and security roles. The content information and assembly properties are used to create a deployment descriptor.

Before you start the wizard, you must have the required files for your servlet or JSP file. When the wizard is completed, your Web module (WAR file) is created in the directory that you specify.

To create a Web module, click the **Wizards** icon on the tool bar and then click **Web Module**. Follow the instructions on each panel.

- [Specifying Web module properties](#)
- [Adding files](#)
- [Specifying optional Web module properties](#)
- [Choosing Web Module icons](#)
- [Adding Web components](#)
- [Adding security roles](#)
- [Adding servlet mappings](#)
- [Adding resource references](#)
- [Adding context parameters](#)
- [Adding error pages](#)
- [Adding MIME mappings](#)
- [Adding Tag Libraries](#)
- [Adding Welcome Files](#)
- [Adding EJB references](#)
- [Setting additional properties and saving the archive](#)

## Specifying Web module properties

On the **Specifying Web Module Properties** panel:

1. Indicate the application to which this module is to be added. If a parent application is not indicated, the module is created as a stand-alone application.
2. Specify a file name and display name for the module. The display name is used to identify your module in the Application Assembly Tool and can be used by other tools. The file name specifies a location on your system where the WAR file is to be created.
3. Provide a short description of the module.
4. Click **Next**.



## Adding files

On the **Adding Files** panel, specify the files that are to be assembled for your Web module.

1. Click **Add Resource Files**, **Add Class Files**, or **Add JAR files**, depending on the type of file you are adding. First, browse for the root directory or archive where the files are located and click **Select**. If you are adding an entire archive, select the directory that contains the archive. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. Relative path names are maintained. The selected files are displayed in the **Selected Files** window. Click **OK**. The files are listed in a table on the wizard panel.
2. If you want to remove a file, select the file in the table and then click **Remove**.
3. Continue to add or remove files until you have the correct set of files.
4. Click **Next**.

## Specifying optional Web module properties

On the **Specifying Optional Web Module Properties** panel:

1. Indicate whether the module can be installed in a distributable Web container. The default value is false.
2. Specify the full classpath for the Web application.
3. Click **Next**.

## Choosing Web Module icons

On the **Choosing Web Module icons** panel, specify icons for your module.

1. Specify the full path name of a GIF or JPEG file. The icon must be 16x16 pixels in size.
2. Specify a full path name of a GIF or JPEG file. The icon must be 32x32 pixels in size.
3. Click **Next**.

## Adding Web components

On the **Adding Web components** panel, add new servlets or JSP files or import existing ones.

To add a new Web component:

1. Click **New**.
2. On the **Specifying Web Component Properties** panel, specify the component name and enter values for other properties. View the help for [6.6.8.0.1: Assembly properties for Web components](#).
3. On the **Specifying Web Component Type** panel, indicate the type of Web component and specify the servlet class name or JSP file.
4. On the **Choosing Web Component Icons** panel, specify a file containing a JPEG or GIF image.
5. On the **Adding Security Role References** panel, enter values for security role references. Click **Add** to enter a role name. Click **OK**. The role name is displayed in the table on the wizard panel. To remove a role, select the role in the table and then click **Remove**. Repeat as necessary. View the help for [6.6.43.0.3: Assembly properties for security role references](#). Click **Next**.
6. On the **Adding Initialization Parameters** panel, enter values for the Web component's initialization parameters. Click **Add** to add a parameter. You must enter a name and value. Click **OK**. The parameter is displayed in a table on the wizard panel. To remove a parameter, select the parameter and click **Remove**. Repeat as necessary. View the help for [6.6.8.0.2: Assembly properties for initialization parameters](#).

7. Click **Finish**.

To import an existing Web component:

1. Click **Import**.
2. Browse the file system to locate the desired archive. The contents of the archive are displayed in a window. Select the desired component and then click **Add**. The components are added to the SelectedComponents window. Click **OK**.

To remove a Web component, select the component name in the table and click **Remove**.

When you are finished adding Web components, click **Next**.

## Adding security roles

On the **Adding Security Roles** panel:

1. Click **Add**. Type a role name and, optionally, type a description. Click **OK**. The role name is displayed in a table on the wizard panel. View the help for [6.6.5.0.5: Assembly properties for security roles](#).
2. Continue to add security roles as needed. If you need to remove a role, select the role in the table and then click **Remove**.
3. Click **Next**.

## Adding servlet mappings

On the **Adding Servlet Mappings** panel:

1. Click **Add**. Enter a URL pattern and select a servlet from the menu. View the help for [6.6.8.0.11: Assembly properties for servlet mapping](#). Click **OK**. The servlet mappings are displayed in a table on the wizard panel.
2. Continue to add and remove URL patterns and corresponding servlets as needed. If you need to remove mapping, select the entry in the table and then click **Remove**.
3. Click **Next**.

## Adding resource references

On the **Adding Resource References** panel, enter references for resource connection factories.

1. Click **Add** to add a reference. You must enter a value for a name, type, and authorization mode. View the help for [6.6.43.0.2 Assembly properties for resource references](#). Click **OK**. The reference is displayed in the table on the wizard panel.
2. To remove a reference, select the reference in the table and then click **Remove**.
3. Continue to add and remove references as needed.
4. Click **Next**.

## Adding context parameters

On the **Adding Context Parameters** panel, enter values for context parameters.

1. Click **Add** to add a parameter. You must enter a name and value. View the help for [6.6.8.0.8: Assembly properties for context parameters](#). Click **OK**. The parameter is displayed in the table on the wizard panel.
2. To remove a parameter, select the parameter and then click **Remove**.

3. Continue to add and remove parameters as needed.
4. Click **Next**.

## Adding error pages

On the **Adding Error Pages** panel, enter values for errorpages.

1. Click **Add** to add a page. You must enter a location. Then choose **Error Code** or **ErrorException**. Enter a name for the error code or exception. View the help for [6.6.8.0.9: Assembly properties for error pages](#). Click **OK**. The error page is displayed in the table on the wizard panel.
2. To remove an error page, select the item in the table and then click **Remove**.
3. Continue to add and remove error pages as needed.
4. Click **Next**.

## Adding MIME mappings

On the **Adding MIME Mappings** panel, enter values for MIME mappings.

1. Click **Add** to add a mapping. You must enter an extension and a MIME type. View the help for [6.6.8.0.10: Assembly properties for MIME mapping](#). Click **OK**. The mapping is displayed in the table on the wizard panel.
2. To remove a mapping, select the mapping and then click **Remove**.
3. Continue to add and remove mappings as needed.
4. Click **Next**.

## Adding Tag Libraries

On the **Adding Tag Libraries** panel, enter values for tag libraries.

1. Click **Add** to add a tag library. You must enter a tag file name and library location. View the help for [6.6.8.0.12: Assembly properties for tag libraries](#). Click **OK**. The tag library is displayed in the table on the wizard panel.
2. To remove a tag library, select the library and then click **Remove**.
3. Continue to add and remove tag libraries as needed.
4. Click **Next**.

## Adding Welcome Files

On the **Adding Welcome Files** panel, enter values for welcome files.

1. Click **Add**. Enter a file name or use the file browser to locate the file. View the help for [6.6.8.0.13: Assembly properties for welcome files](#). Click **OK**. The file name is displayed in the table on the wizard panel.
2. To remove a file, select the file in the table and then click **Remove**.
3. Continue to add and remove files as needed.
4. Click **Next**.

## Adding EJB references

On the **Adding EJB References** panel, enter values for EJBReferences.

1. Click **Add** to add a reference. You must enter a value for the name, home interface, remote interface, and type. View the help for [6.6.43.0.1: Assembly properties for EJB references](#). Click **OK**. The reference is displayed in the table on the wizard panel.
2. To remove a reference, select the entry in the table and then click **Remove**.
3. Continue to add and remove references as needed.

## Setting additional properties and saving the archive

Click **Finish** to complete the wizard. To change settings for properties, click **Back** to return to the appropriate panel. Make any needed changes, and then click **Finish**.

After you click **Finish**, the contents of the archive are displayed in the Application Assembly Tool window. In the navigation pane, continue adding or modifying properties as needed. For example, you can add binding information. When you are finished editing the archive, click **File->Save** to save the archive file.

## 6.6.11: Administering HTTP session support (overview)

When configuring the Session Manager, the WebSphere administrator can:

- Specify which session tracking mechanism to use to pass the sessionid between the browser and the servlet
- Specify whether to save session data in a database (persistent sessions)
- Define the persistent sessions characteristics
- Define other characteristics of the Session Manager environment

## 6.6.11.0: Session Manager properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Application Server



The application server with which the Session Manager is associated

### Allow Overflow



Whether to allow the number of sessions in memory to exceed the value specified by Max In Memory Session Count property

### Cookie Domain



or Domain



The value of the domain field of a session tracking cookie. This value will dictate to the browser whether or not to send a cookie to particular servers. For example, if you specify a particular domain, session cookies will be sent only to hosts in that domain. The default domain is the server.

### Cookie Maximum Age



or Maximum Age



The amount of time that the cookie will live on the client browser.

Specify that the cookie will live only as long as the current browser session, or to a maximum age. If you choose the maximum age option, specify the age in seconds.

This value corresponds to the Time to Live (TTL) value described in the Cookie specification.

For *Advanced Single Server Edition*, use -1 to specify the current browser session.

### Cookie Name



or Name



A unique name for the cookie. The name must be JSESSIONID as specified in the Servlet 2.2 API

### Confirm Password



Enter the password a second time to ensure it recorded correctly

### Cookie Path



or Path



This dictates browser whether cookie is sent to the URI requested based on the path. Specify any string representing a path on the server. "/" indicates root directory.

Specify a value in order to restrict the paths to which the cookie will be sent. By restricting paths, you can keep the cookie from being sent to certain URLs on the server. If you specify the root directory, the cookie will be sent no matter which path on the given server is accessed.

### Data Source JNDI Name



or Data Source



The JNDI name of the non-JTA enabled data source from which the Session Manager will obtain database connections.

For example, if the JNDI name of the datasource is "jdbc/sessions", specify "jdbc/sessions."

The data source represents a pool of database connections and a configuration for that pool (such as the pool size). The data source must already exist as a configured resource in the environment.

## Data Source



See Data Source JNDI Name

## DB2 Row Size



The tablespaces page size configured for the sessions table, if using a DB2 database. Possible values are 4, 8, 16, and 32 kilobytes (K).

The default row size is 4K. In DB2, it can be updated to a larger value. This can help database performance in some environments.

When this value is other than 4, you must specify Table Space Name to use. For 4K pages, the Table Space Name is optional.

## Domain




See Cookie Domain

## Enable Cookies



Whether session tracking will use cookies to carry session IDs. If cookies are enabled, session tracking will recognize session IDs that arrive as cookies and try to use cookies for sending session IDs. If cookies are not enabled, session tracking will use URL rewriting instead of cookies (if URL rewriting is enabled). Enabling cookies takes precedence over URL rewriting.

 For *Advanced Single Server Edition*, do not disable cookies in the Session Manager of the application server that is running the administrative application, because it will cause the administrative application not to function after a restart of the server. An alternative is to run the administrative application in a separate process from your applications.

## Enable Persistent Sessions



When persistent sessions are enabled, the Session Manager will persist session information into the data source specified by the data source connection settings. Otherwise, the Session Manager will discard the session data when the server shuts down.

## Enable Protocol Switch Rewriting



Whether the session ID is added to a URL when the URL requires a switch from HTTP to HTTPS or from HTTPS to HTTP. If rewriting is enabled, the session ID is required in order to go between HTTP and HTTPS.


## Enable Security Integration



or Integrate with WebSphere Security



When security integration is enabled, the Session Manager will associate the identity of users with their HTTP sessions.

 Do not enable this property if the application server contains a Web application that has Form Login configured as the authentication method and local operating system is the authentication mechanism. It will cause authorization failures when user agents try to use the Web application.

## **Enable SSL Tracking** or **Enable SSL ID Tracking**

Whether session tracking uses SSL to carry session IDs. Enabling SSL tracking takes precedence over cookie-based session tracking and URL rewriting.

## **Enable SSL ID Tracking**

See Enable SSL Tracking

## **Enable URL Rewriting**

Specifies whether the Session Manager uses rewritten URLs to carry the session IDs. If URL rewriting is enabled, the Session Manager recognizes session IDs that arrive in the URL if the `encodeURL` method is called in the servlet.

## **First Hour** or **First Time of Day (0 - 23)**

The first hour during which the invalidated persistent sessions will be cleared from the database. This value must be a positive integer between 0 and 23. This is valid only when schedule invalidation is enabled.

## **First Time of Day (0 - 23)**

See First Hour

## **Integrate with WebSphere Security**

See Enable Security Integration

## **Invalidation Timeout**

Specifies how long a session is allowed to go unused before it will be considered valid no longer.

*For Advanced Edition:* Specify either "Set timeout" or "No timeout." If you select to set the timeout, the value must be at least two minutes, specified in minutes.

*For Advanced Single Server Edition:* Use a -1 to specify that the session will not be invalidated.

The value of this setting is used as a default when the session timeout is not specified in a Web module deployment descriptor.

Note, to preserve performance, the invalidation timer is not accurate "to the second." It is safe to assume that the timer is accurate to within two minutes. When the Write Frequency is time based, this value should be at least twice as large as the write interval.

## **Maximum Age**

See Cookie Maximum Age

## **Max In Memory Session Count** or **Maximum In Memory Session Count**

Specifies the maximum number of sessions to maintain in memory.

The meaning differs depending on whether you are using in-memory or persistent sessions. For in-memory sessions, this value specifies the number of sessions in the base session table. Use the Allow Overflow property to specify whether to limit sessions to this number for the entire Session Manager, or to allow additional sessions to be stored in secondary tables.



For persistent sessions, this value specifies the size of the memory cache for sessions. When the session cache has reached its maximum size and a new session is requested, Session Manager removes the least recently used session from the cache to make room for the new one.

**Name** 

See Cookie Name

**Node** 

The administrative node with which the Session Manager is associated

**Password**  

The password for database access

**Path** 

See Cookie Path

**Restrict exchange of cookies to secure sessions**  or **Secure** 

Whether session cookies include the secure field. Enabling the feature will restrict the exchange of cookies only to HTTPS sessions.

**Schedule Invalidation**  or **Specify session database cleanup schedule for invalidated sessions** 

Enables the scheduled invalidation process for cleaning up the invalidated HttpSession from the sessions database. Enable this option to reduce the number of database updates that are required to keep the HttpSession alive.

When this option is not enabled, the invalidator process runs every few minutes to remove invalidated HttpSession.


When this option is enabled, this setting specifies the two hours of a day in which the invalidator process cleans up the invalidated persistent sessions in the database. Specify the times at which there will be less activity in the session database.

**Second Hour**  or **Second Time of Day (0 - 23)** 

The second hour during which the invalidated persistent sessions will be cleared from the database. This value must be a positive integer between 0 and 23. This is valid only when schedule invalidation is enabled.

**Secure** 

See Restrict exchange of cookies to secure sessions

 For *Advanced Single Server Edition*, do not enable this feature in the Session Manager of the application server that is running the administrative application, because it will cause the administrative application not to function after a restart of the server. An alternative is to run the administrative application in a separate process from your applications.

**Specify session database cleanup schedule for invalidated sessions** 

See Schedule Invalidation

## Table Space Name



Tablespace to be used for the sessions table. This value is required when the DB2 Page Size is other than 4K and session persistence is enabled.

## User ID or Username



The user ID for database access

## Use Multirow Sessions or Using Multirow Schema



Whether to place each instance of application data in a separate row in the database, allowing larger amounts of data to be stored for each session. This can yield better performance in certain usage scenarios. If using multirow schema is not enabled, instances of application data can be placed in the same row.

## Username



See User ID

## Using Multirow Schema



See Use Multirow Sessions

## Write Contents



Whether only updated attributes should be written to the database. Otherwise, all of the session attributes will be written to the database, whether or not they have changed.

- If you specify only updated attributes, only the updated attributes will be written to the database
- If you specify all session attributes, all attributes will be written to the database

## Write Frequency



When the session will be written to the database.

- If you specify end of servlet service (END\_OF\_SERVLET\_SERVICE), a session will be written to the database after the servlet completes execution.
- If you specify manual update (MANUAL\_UPDATE), programmatic sync on the IBMSession object will be required in order to write the session data to the database.
- If you specify time based (TIME\_BASED\_WRITE), session data will be written to the database based on the specified Write Interval value.

## Write Interval



If the Write Frequency is specified as TIME\_BASED\_WRITE, this value specifies how often the session data will be written to the database. The value must be a number of seconds, specified as a positive integer in the range from 5 to 9999. At minimum, a configured Invalidation Timeout should be twice as large as the configured write interval.

See also the other [application server properties](#).

## 6.6.11.3: Administering session management with the Web console

Use the Web console to edit the configurations of Session Managers. Each application server has one Session Manager for enabling, configuring, and tuning HTTP session support.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**  
-> *application\_server\_name* -> **Web Containers**  
-> **Session Manager**

## 6.6.11.3.4: Updating session management settings with the Web console

During this task, you will update the configuration of an existing Session Manager, which is part of an application server configuration.

To update a Session Manager configuration:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Web Container** -> **Session Manager** where *application\_server\_name* is the name of the existing application server.
2. Click **Session Manager**. Its properties will be displayed on the rightside of the console.
3. Modify the properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  - [Stop the server](#)
  - [Start the server again](#).

## 6.6.11.5: Procedure for configuring persistent session support

The default [application server configuration file](#) for Advanced Single Server Edition assumes that the name of the database for session persistence will be "Session". The administrator must either create the database or the update the configuration to reference a different database.

To configure Session Manager for persistent sessions:

1. [Create a JDBC Driver](#).
2. [Create a data source](#) pointing to existing database, using the JDBC driver that you created. Note the JNDI name of the data source.
3. [Configure the Session Manager and save your changes](#):
  1. Enable session persistence.
  2. Specify the JNDI name from the previous step.
  3. Specify the database user ID and password for accessing the database.

The Session Manager provides a pre-configured data source for session persistence, appropriately named the Session Persistence data source. When using it, ensure that:

- The Session database referenced by the application server configuration file exists. Or, change the configuration to point to a different existing database.
- The JDBC provider for the data source is configured properly. That is, it must point to the location of db2java.zip (in the case of DB2).
- The correct user ID and password are specified for accessing the database. By default db2admin is specified as both the user ID and password in the Session Manager. If you are using a different ID or password, specify it in the Session Manager configuration.

## 6.6.12: Configuring user profile support

When configuring the User Profile Manager, the administrator can specify:

- A user ID and password for accessing the user profile database and table
- The "data wrapper" class that implements user profile support
- The data source for storing user profile data in a table
- The enterprise bean classes for accessing user profiles
- Whether to enable user profiles at this time

The User Profile Manager settings require significant understanding of the classes that implement user profilesupport, particularly if the implementation involves enterprise beans.

To configure a user profile, follow these steps:

1. Develop a servlet that accesses the User Profile Manager API.
2. Create an enterprise application that uses the user profile implementation beans.(These beans can be found in installation\_root/lib/userprofile.jar.) Use the same JNDI datasource name for both UP\_ReadOnly and UP\_ReadWrite.Make a note of the JNDI datasource name, because you must also specify this in userprofile.xml (step 4).
3. Deploy the enterprise application in the application server.
4. To installation\_root/properties, add a file named userprofile.xml in the following format.Specify enterprise bean class names;data wrapper class name; andJNDI names for the read-only bean, read/write bean, and datasource (from step 2). You must also add user ID and password information for the JNDI datasource.An example of userprofile.xml follows:

```
<?xml version="1.0"?> <userprofile>      <userprofile-enabled>true</userprofile-enabled>
<userprofile-wrapper-class>      <classname>
com.ibm.servlet.personalization.userprofile.UserProfile      </classname>
</userprofile-wrapper-class>      <userprofile-manager-name>      User Profile Manager
</userprofile-manager-name>      <userprofile-bean>      <readonly-interface>
com.ibm.servlet.personalization.userprofile.UP_ReadOnly      </readonly-interface>
<readwrite-interface>      com.ibm.servlet.personalization.userprofile.UP_ReadWrite
</readwrite-interface>      <readonlyhome-interface>
com.ibm.servlet.personalization.userprofile.UP_ReadOnlyHome      </readonlyhome-interface>
<readwritehome-interface>      com.ibm.servlet.personalization.userprofile.UP_ReadWriteHome
</readwritehome-interface>
<readonly-JNDI-lookupName>UP_ReadOnlyHome</readonly-JNDI-lookupName>
<readwrite-JNDI-lookupName>UP_ReadWriteHome</readwrite-JNDI-lookupName>      </userprofile-bean>
<userprofile-store>      <database-userid></database-userid>
<database-password></database-password>      <database-datasource></database-datasource>
</userprofile-store>      </userprofile>
```

5. Start the enterprise application.

## 6.6.13: Administering transports (overview)

Transports define the characteristics of the connections between the Web server and the application server, across which requests for applications are routed. Specifically, they define the connection between the Web server plug-in and the Web container of the application server.

Use transport properties to specify:

- How to manage the set of connections; for example, how many concurrent requests to allow
- Whether to secure the connections with SSL
- Host and IP information for the transport participants

Administering the transport is closely related to administering the WebSphere plug-ins for Web servers. Indeed, without the plug-in configuration, the transport configuration is of little use.

### The internal transport

For applications in a test or development environment (in other words, a non-production environment), you can use the *internal HTTP transports* system to serve servlets without an Web server plug-in. Simply use the internal HTTP transport port (typically on port 9080).

For example, to serve the "snoop" servlet without an HTTP server, use the URL:

`http://your.server.name:port/servlet/snoop`

with *port* being the internal transport port number (typically 9080) and *your.server.name* being localhost if the Application Server is on the local machine.

For a product environment, *do not* use the internal transport, as it lacks the performance available when using a Web server plug-in.

At times, the transport might be configured to use a port other than 9080. (The transport configuration is a part of the [Web container configuration](#). In such cases, you need adjust your virtual host alias and what you type into the Web browser, as described in the [virtual host administrative overview](#).

### Regenerating the WebSphere plug-in for the Web server

It is suggested you review the information about [administering WebSphere plug-ins for Web servers](#), because you will need to regenerate the plug-in configuration each time you specify or change an HTTP transport.

## 6.6.13.0: Properties of transports

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### HTTP transports

#### Dynamic Properties or Properties

Special configuration properties for the transport. Each transport type can contain custom properties specified by the transport provider.

*For the Advanced Edition:* The properties are displayed as fields into which you can enter values.

*For the Advanced Single Server Edition:* Enter the property names and their corresponding values into the Properties area.

The properties are as follows:

#### Connection Backlog or MaxConnectBacklog

The maximum number of outstanding connect requests that the operating system will buffer while it waits for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected.

Set this value to the number of concurrent connections that you would like to allow. Keep in mind that a single client browser might need to open multiple concurrent connections (perhaps 4 or 5); however, also keep in mind that increasing this value consumes more kernel resources.

The value of this property is specific to each transport.

#### ConnectionIOTimeout or I/O Timeout

The maximum time (in seconds) to wait when trying to read or write data DURING a request.

The value of this property, after it is specified for the first transport, will be applied globally to all other HTTP transports in this administrative domain.

#### ConnectionKeepAliveTimeout or Keep Alive Timeout

The maximum time (in seconds) to wait for the next request on a keep alive connections.

The value of this property, after it is specified for the first transport, will be applied globally to all other HTTP transports in this administrative domain.

#### I/O Timeout

See ConnectionIOTimeout



## Keep Alive Timeout

See ConnectionKeepAliveTimeout

## MaxConnectBacklog

See Connection Backlog

## Maximum Keep Alives or MaxKeepAliveConnections

The maximum number of concurrent keep alive (persistent) connections across all HTTP transports. (HTTP 1.1 allows multiple requests to be sent on the same TCP/IP connection.) The default value is 90% of the maximum number of threads in the Web container thread pool. This prevents all of the threads from being held by keep alive connections so that there are threads available to handle new incoming connect requests.

The value of this property, after it is specified for the first transport, will be applied globally to all other HTTP transports in this administrative domain.

## Maximum Requests per Keep Alive or MaxKeepAliveRequests

The maximum number of requests which can be processed on a single keep alive connection.

The value of this property, after it is specified for the first transport, will be applied globally to all other HTTP transports in this administrative domain.

## Enable SSL or SSL Enabled

Whether to protect connections between the WebSphere plug-in and application server with Secure Socket Layer (SSL). See also the [security properties](#)

## External

Whether this transport is for internal or external use

## Host or Host Name or Transport Host

The host IP address to which to bind for the transport

## Host Name

See Host

## Port or Transport Port

The port to which to bind for the transport

## Properties

See Dynamic Properties

## SSL Enabled

See Enable SSL

## Transport Host

See Host

**Transport Port** 

See Port

**Use global SSL default configuration** 

See the [general security properties](#)

See also the other [application server properties](#).

## 6.6.13.3: Administering transports with the Web console

Use the Web console to create and edit transport configurations. Transports describe how application requests flow from the WebSphere plug-ins for Web servers to the application server Webcontainer.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Web Containers**

-> **HTTP Transports**

## 6.6.13.3.1: Configuring new HTTP transports with the Web console

During this task, you will configure a new HTTP transport.

1. In the tree on the left side of the console, click

**Nodes** -> *hostname* -> **Application Servers**  
-> *application\_server\_name* -> **Web Containers**  
-> **HTTP Transports**

to see a list of available transports.

2. Click the **New** button located above the list.
3. Specify settings for the new transport.
4. When finished, click **OK**.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.13.3.3: Removing HTTP transport configurations with the Web console

During this task, you will remove a transport configuration.

To remove the configuration:

1. In the tree on the left side of the console, click

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Web Containers**

-> **HTTP Transports** to see a list of available transports.

2. Locate the transport that you want to remove. Click the checkbox to the left of its name, such that the check box is selected.

Make sure that no other transports have their check boxes selected, unless you also want to delete those providers.

3. Click the **Delete** button above the list of transports.
4. Click **OK**.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.13.3.4: Updating transport configurations with the Web console

During this task, you will update a transport configuration.

1. In the tree on the left side of the console, click

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Web Containers**

-> **HTTP Transports** to see a list of available transports.

2. Click the transport that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the transport properties.
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.14: Administering database connections (overview)

The administrator of Advanced Single Server Edition manages database connections for applications installed in the application server runtime. For example, applications might require data access in order to persist sessions.

Note, unlike Advanced Edition Version 4.x, and both Advanced and Standard Edition Version 3.5.x, Advanced Single Server Edition Version 4.x does *not* keep its administrative data in a database. There is no administrative database to install or administer. (However, the WebSphere samples still require the installation and configuration of a database -- see the [samples documentation](#) for details).

The WebSphere administrator has an important role in establishing and maintaining connection pools through data sources and drivers:

- Configuring the resources used in connection pooling -- JDBC drivers and data sources

The administrator needs to configure data sources and JDBC drivers for each brand and version of database from which enterprise applications or individual resources will require connections.

- Adjusting connection pooling parameters for optimal performance.

Connection pools provide a way to share the connection overhead among multiple requests, but it is possible to configure too large a connection pool. The administrator needs to determine the optimal value for the pool size and other settings, based on environmental factors such as the operating system in use.

JDBC Providers and data sources are used by Java components requiring database access, such as servlets.

## 6.6.14.0: Properties of JDBC and data source providers

The administrative tools use the following terms for the same type of configuration:  
JDBC provider or data source provider

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Classpath**



**or Server Class Path**



The path to the JAR file containing the implementation code for the database driver, such as db2java.zip file for DB2.

See [the reference below](#) for a list of default locations.

**Custom Properties**



Name-value pairs for setting additional properties

**Description**



A description of the driver, for your administrative records

**Implementation Class**



**or Implementation Classname**



The name of the Java data source class provided by the database vendor.

DB2:

- COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource (one phase commit protocol)
- COM.ibm.db2.jdbc.DB2XADataSource (two phase commit protocol)

DB2 on iSeries - toolbox driver

- com.ibm.as400.access.AS400JDBCCConnectionPoolDataSource (one phase commit protocol)
- com.ibm.as400.access.AS400JDBCXADataSource (two phase commit protocol)

DB2 on iSeries - native driver

- com.ibm.db2.jdbc.app.DB2StdConnectionPoolDataSource (one phase commit protocol)
- com.ibm.db2.jdbc.app.DB2StdXADataSource (two phase commit protocol)

Oracle:

- oracle.jdbc.pool.OracleConnectionPoolDataSource (one phase commit protocol)
- oracle.jdbc.xa.client.OracleXADataSource (two phase commit protocol)

Sybase:

- com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource (one phase commit protocol)
- com.sybase.jdbc2.jdbc.SybXADataSource (two phase commit protocol)



Merant:

- com.merant.sequelink.jdbcx.datasource.SequeLinkDataSource (one and two phase commit protocol)

Informix

- com.informix.jdbcx.IfxCConnection PoolDataSource (one phase commit protocol)
- com.informix.jdbcx.IfXXADataSource (two phase commit protocol)

InstantDB (NOT supported as an administrative database)

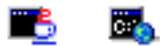
- com.ibm.ejs.cm.portability.IDBConnectionPoolDataSource (one phase commit protocol)

**Name**



A name for the driver. It is recommended you enter a name that is suggestive of the database product you are using, such as DB2JdbcDriver.

**Node**



The node (machine) on which to install the driver.

*For Advanced Edition:* Use the buttons on the Nodes panel to access dialogs for installing drivers on specific nodes and for uninstalling drivers.

**Property Set**



See the resource provider properties (only valid if you are using *Advanced Single Server Edition*).

**Server Class Path**



See Classpath

**URL prefix**



The URL prefix with which this driver is associated. The URL prefix is comprised of the protocol and subprotocol, separated by a colon (":").

The Database Name of the data source is appended to the URL prefix to produce the full JDBC URL of the database, such as jdbc:db2:was

## Locating JDBC providers on your operating system

The following table lists the default locations of JDBC provider files. See the product prerequisites Web site for the most up-to-date information on the operating system brands and databases supported by IBM WebSphere Application Server. Column 1 in the table lists the operating system, and column 2 shows a list of the drivers that are available for use with each database supported for the operating system.

Operating system	Drivers

AIX	<ul style="list-style-type: none"> <li>● DB2: <i>\$DB2_HOME/java12/db2java.zip</i> or <i>\$DB2_HOME/java/db2java.zip</i></li> <li>● Oracle: <i>\$ORACLE_HOME/jdbc/lib/classes12.zip</i></li> <li>● Sybase: <i>sybase_install_root/jConnect-5_2/classes/jconn2.jar</i> Merant SequelLink: <ul style="list-style-type: none"> <li>○ Complimentary copies with WebSphere Application Server: <ul style="list-style-type: none"> <li>■ <i>product_installation_root/lib/sljc.jar</i></li> <li>■ <i>product_installation_root/lib/sljcx.jar</i></li> </ul> </li> </ul> </li> </ul>
HP-UX	<ul style="list-style-type: none"> <li>● DB2: <i>\$DB2_HOME/java12/db2java.zip</i> or <i>\$DB2_HOME/java/db2java.zip</i></li> <li>● Oracle: <i>\$ORACLE_HOME/jdbc/lib/classes12.zip</i></li> <li>● Merant SequelLink: <ul style="list-style-type: none"> <li>○ Complimentary copies with WebSphere Application Server: <ul style="list-style-type: none"> <li>■ <i>product_installation_root/lib/sljc.jar</i></li> <li>■ <i>product_installation_root/lib/sljcx.jar</i></li> </ul> </li> </ul> </li> </ul>
Linux (Intel)	<ul style="list-style-type: none"> <li>● DB2: <i>\$DB2_HOME/java12/db2java.zip</i> or <i>\$DB2_HOME/java/db2java.zip</i></li> <li>● Oracle: <i>\$ORACLE_HOME/jdbc/lib/classes12.zip</i></li> </ul>
Linux on S/390	<ul style="list-style-type: none"> <li>● Oracle: <i>\$ORACLE_HOME/jdbc/lib/classes12.zip</i></li> </ul>
Solaris	<ul style="list-style-type: none"> <li>● DB2: <i>\$DB2_HOME/java12/db2java.zip</i> or <i>\$DB2_HOME/java/db2java.zip</i></li> <li>● Oracle: <i>\$ORACLE_HOME/jdbc/lib/classes12.zip</i></li> <li>● Sybase: <i>sybase_install_root/jConnect-5_2/classes/jconn2.jar</i> Merant SequelLink: <ul style="list-style-type: none"> <li>○ Complimentary copies with WebSphere Application Server: <ul style="list-style-type: none"> <li>■ <i>product_installation_root/lib/sljc.jar</i></li> <li>■ <i>product_installation_root/lib/sljcx.jar</i></li> </ul> </li> </ul> </li> </ul>
Windows	<ul style="list-style-type: none"> <li>● DB2: <i>C:\SQLLIB\java\db2java.zip</i></li> <li>● Oracle: <i>C:\Oracle\Ora81\jdbc\lib\classes12.zip</i></li> <li>● Merant SequelLink: <ul style="list-style-type: none"> <li>○ Complimentary copies with WebSphere Application Server: <ul style="list-style-type: none"> <li>■ <i>product_installation_root\lib\sljc.jar</i></li> <li>■ <i>product_installation_root\lib\sljcx.jar</i></li> </ul> </li> </ul> </li> </ul>

## 6.6.14.0.1: Properties of data sources

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Category



An optional category string that can be used to classify or group the resource

### Confirm Password



### or Re-Enter Password



Confirm the password that you entered in the preceding field

### Connection timeout



The maximum time in seconds that requests for a connection wait if the maximum number of connections is reached and all connections are in use

This value must be a positive integer.

### Custom Properties



### or Property Set



A set of custom name-value pairs describing properties of the data source

**The following are the required properties for each database type:**

DB2:

No required properties.

DB2 on iSeries -- toolbox driver:

serverName

The name of the server from which the data source will obtain connections, such as "MyServer"

DB2 on iSeries -- native driver:

No required properties.

Oracle:

URL

The url indicating the database from which the data source will obtain connections, such as "jdbc:oracle:thin:@myServer:1521:myDatabase," where "myServer" is the server name, "1521" is the port it is using for communication, and "myDatabase" is the database name.

Set the user and password in the field provided in the console.

Sybase:

serverName

The name of the database server, such as "db\_machine"

portNumber

The tcpip port number through which all communications to the server take place, such as 4100.

Merant:

serverName

The name of the server where SequeLinkServer resides, such as "MyServer"

portNumber

The TCP/IP port SequeLinkServer uses for communication. By default, SequeLinkServer uses port 19996, such as "19996".

disable2Phase

By default, two phase connections are used by Merant always, because the same data source class is used for one phase and two phase commit protocols.

To use one phase connections, set disable2Phase to true.

Set the user and password in the field provided in the console.

InstantDB (NOT supported as an administrative database):

url

The url indicating the database from which the datasource will obtain connections. Example: "jdbc:idb:"*configurationFile* where *configurationFile* is the name of the IDB configuration file.

Informix:

ifxIFXHOST

The physical machine name

serverName

The name of the Informix instance on the physical machine

portNumber

The port number of the Informix instance

informixLockModeWait

By default, Informix throws an exception when it cannot acquire a lock, rather than waiting for the current owner of the lock to release it. To modify this behavior, set this property to the number of seconds to wait for a lock. The default is 0 seconds. Any negative value means to wait forever.

The following are some additional, optional properties for various database types:

Sybase:

connectionProperties

CHARSET\_CONVERTER\_CLASS=com.sybase.jdbc2.utils.TruncationConverter

Setting the CHARSET\_CONVERTER\_CLASS can prevent exceptions such as this one when performing a dataSource.getConnection() call:

```
java.io.IOException: JZ0I6: An error occurred converting
UNICODE to the charset used by the server. Error
message: java.io.CharConversionException:
```

`java.io.UnsupportedEncodingException: hp-roman8`

Set additional connectionProperties by specifying them using the same pattern, separated by commas: *PROPERTY\_NAME=value;PROPERTY\_NAME=value; ...*

#### Database Name



The name of the database used to store entity bean data

This is required for DB2, and sometimes required for Sybase, Merant, and Informix (depending on your database configuration), and ignored for Oracle.

#### Description



A description of the data source, for your administrative records

#### Default Password



or Password



The password for connecting to the database when no user ID and password pair is specified by the application. If the default password is specified, the default user ID must also be specified.

#### Default User ID



or User



The user name for connecting to the database when no user ID and password pair is specified by the application. If the default user ID is specified, the default password must also be specified.

#### Disable Auto Connection Cleanup



Keeps the connection pooling software from automatically closing connections from this data source at the end of a transaction. This behavior is needed if you want to reuse the same connection across multiple transactions. When this is set, you must be sure to close the connection programmatically when you are through using it.

#### Idle timeout



The maximum time in seconds that an idle (unallocated) connection can remain in the pool before being removed to free resources.

This value must be a positive integer.

#### JDBC Provider



The JDBC driver (also known as data source provider) with which this data source is associated. It is used to connect to a relational database.

#### JNDI Name



The JNDI name for the resource, including any naming subcontexts. This name is used as the linkage between the platform's binding information for resources defined in the client application's deployment descriptor and actual resources bound into JNDI by the platform.

#### Maximum Connection Pool Size



or Maximum Pool Size



The maximum number of connections that can be in the pool. If the maximum number of connections is reached and all connections are in use, additional requests for a connection wait up to the number of seconds specified in the Connection timeout property.

This value must be a positive integer.

## Maximum Pool Size

See Maximum Connection Pool Size

## Minimum Connection Pool Size or Minimum Pool Size

The minimum number of connections in the pool.

This value must be a positive integer.

## Minimum Pool Size

See Minimum Connection Pool Size

## Name

A name by which to administer the data source.

It is recommended that you enter a name that is suggestive of the database you will use to store entity bean data, such as WASDataSource, where WAS is the database name. The default value for this property is the value of the Name property prefixed with "jdbc/" (such as "jdbc/DataSourceName").

## Orphan timeout

The maximum number of seconds that an application can hold a connection without using it before the connection can be returned to the pool.

This value must be a positive integer.

Note that the actual amount of time before a connection is closed is approximately twice the orphan timeout value.

## Password

See Default Password

## Property Set

See the [custom properties](#)

## Re-Enter Password

See Confirm Password

## Statement Cache or Statement Cache Size

The maximum number of prepared statements to cache for the data source. The limit is shared among all connections. The default value is 100.

## User

See Default User ID

## 6.6.14.3: Administering database connections with the Web console

Use the Web console to edit the configurations of JDBC providers and data sources, which are used by your installed applications to access data from databases.

Work with objects of this type by locating them in the tree on the left side of the console:

Click **Resources** -> **JDBC Providers** to display the JDBC providers view on the right side of the console.

If you expand the tree further, you will also see **Data Sources** for each JDBC Provider instance.

## 6.6.14.3.1: Configuring new JDBC providers with the Web console

During this task, you will configure a new JDBC provider.

1. In the console, click **Resources** -> **JDBC Providers**.
2. When the JDBC providers are displayed on the right side of the console, click the **New** button.
3. Specify the [properties](#) for the new driver.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).



## 6.6.14.3.2: Configuring new data source configurations with the Web console

During this task, you will create a new data source configuration.

1. Click **Resources** -> **JDBC Providers** -> *provider\_name* -> **Data Sources** in the tree on the left side of the console, where *provider\_name* is the provider with which the new data source should be associated.
2. When the Data Sources view is displayed on the right side of the console, click the **New** button.
3. Specify the [properties](#) for the new data source.
4. When you are finished, click **Apply**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.14.3.3: Updating JDBC provider configurations with the Web console

During this task, you will update an existing JDBC provider configuration.

1. Click **Resources** -> **JDBC Providers** -> *provider\_name* in the tree on the left side of the console. The properties of *provider\_name* will be displayed on the right side of the console.
2. Modify the [JDBC provider properties](#).
3. When you are finished, click **Apply**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.14.3.4: Updating data source configurations with the Web console

During this task, you will update an existing data source configuration.

1. Click **Resources** -> **JDBC Providers** -> *provider\_name* -> **Data Sources** -> *data\_source\_name* in the tree on the left side of the console. The data source properties will be displayed on the rightside of the console.
2. Modify the [data source properties](#).
3. When you are finished, click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.14.3.5: Removing JDBC provider configurations with the Web console

During this task, you will remove an existing JDBC provider configuration.

1. Click **Resources** -> **JDBC Providers** in the tree on the left side of the console.
2. From the JDBC provider view, select the JDBC provider to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.14.3.6: Removing data source configurations with the Web console

During this task, you will remove an existing data source configuration.

1. Click **Resources** -> **JDBC Providers** -> *provider\_name* -> **Data Sources** in the tree on the left side of the console.
2. From the data source view, select the data source to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.14.5: Additional administrative tasks for specific databases

For your convenience, this article provides instructions for enabling some popular database drivers, and performing other administrative tasks often required in order to provide data access to applications running on WebSphere Application Server. These tasks are performed outside of the WebSphere Application Server administrative tools, often using the database product tools. Always refer to the documentation accompanying your database driver as the authoritative and complete source of driver information.

See the [WebSphere Application Server product prerequisites](#) for the latest information about supported databases, drivers, and operating systems.

- [Enabling JDBC 2.0](#)
  - [For DB2 on Windows NT](#)
  - [For DB2 on UNIX](#)
- [Sourcing the db2profile script on UNIX](#)
- [Using JTA drivers](#)
  - [For DB2 on Windows NT](#)
  - [For DB2 on UNIX](#)
- [For Oracle 8.1.7 two phase commit support](#)
- [For Sybase on AIX](#)

### Enabling JDBC 2.0

Ensure that your operating system environment is set up to enable JDBC 2.0 use. This is required in order to use data sources created through WebSphere Application Server.

The following steps make it possible to find the appropriate JDBC 2.0 driver for use with WebSphere Application Server administration:

#### Enabling JDBC 2.0 with DB2 on Windows NT

To enable JDBC 2.0 use on Windows NT systems:

1. Stop the DB2 JDBC Applet Server service.
2. Run the following batch file:  
`SQLLIB\java12\usejdbc2.bat`
3. Stop WebSphere Application Server (if it is running) and start it again.

Perform the steps once for each system.

#### Determining the level of JDBC in use for DB2 on Windows NT

To see whether the JDBC level in use on your system:

- If JDBC 2.0 is in use, this file will exist:  
`SQLLIB\java12\inuse`
- If JDBC 1.0 is in use, this file will exist:  
`SQLLIB\java11\inuse`  
or there will be no java11 directory

#### Enabling JDBC 2.0 with DB2 on UNIX

Before starting WebSphere Application Server, you need to call `$INSTHOME/sql/lib/java12/usejdbc2` to use JDBC 2.0. For convenience, you might want to put this in your root user's startup script. For example on AIX, add the following to the root user's .profile:

```
if [ -f /usr/lpp/db2_07_01/java12/usejdbc2 ] ; then . /usr/lpp/db2_07_01/java12/usejdbc2fi
```

#### Determining the level of JDBC in use for DB2 on your UNIX system

To determine if you are using JDBC 2.0, you can echo `$CLASSPATH`. If it contains

```
$INSTHOME/sql/lib/java12/db2java.zip
```

then JDBC 2.0 is in use.

If it contains

```
$INSTHOME/sql/lib/java/db2java.zip
```

then JDBC 1.0 is in use.

### Sourcing the db2profile script on UNIX

Before starting WebSphere Application Server to host applications requiring data access, source the db2profile:

`~db2inst1/sqllib/db2profile`

where *db2inst1* is the user created during DB2 installation.

## Using JTA drivers

Instructions are available for using JTA drivers on particular operating systems. See your operating system's documentation for more information.

With the JDBC programming model underlying WebSphere Application Server Version 4.0x, the term "JTA enabled" becomes somewhat odd, with respect to its Version 3.5 meaning. The goal of this section about "Using JTA drivers" is to provide information about the steps that make DB2 work nicely with applications utilizing XA classes -- that is, those whose `DataSource` classes implement `javax.sql.XADataSource`.

### Using JTA drivers for DB2 on Windows NT

To enable JTA drivers for DB2 on Windows NT, follow these steps:

1. Stop all DB2 services.
2. Stop any other processes that use the `db2java.zip` file. (Note: If the **JVIEW** process is active, you must use the **Task Manager** utility to stop it.)
3. Make sure that you already [enabled JDBC 2.0](#).
4. Start the DB2 services.
5. Configure DB2 to use JTS as the transaction processing (TP) monitor. From the DB2 Control Center, follow these steps:
  - a. Right-click the DB2 instance that contains the database that is to be enabled for JTA access.
  - b. Click **Multisite Update, Configure** to start the Smartguide utility.
  - c. Click the **Use the TP monitor named below** radio button.
  - d. Select **JTS** as the TP monitor.
  - e. Click **Done**.
6. Bind the necessary packages to the database. From the **DB2 Command Line Processor** window, issue the following commands:

```
db2=> connect to mydb2jta
db2=> bind db2home\bnd\@db2cli.lst
db2=> bind db2home\bnd\@db2ubind.lst
db2=> disconnect mydb2jta
```

where *mydb2jta* is the name of the database that is to be JTA enabled, and *db2home* is the DB2 root installation directory path (for example, `D:\ProgramFiles\SQLLIB\bnd\@db2cli.lst`).
7. When you use an IBM WebSphere Application Server administrative client (such as the WebSphere Administrative Console) to configure a JDBC driver, specify the following settings:
  - **Server class path** = `%DB2_ROOT%/Sqllib/java/db2java.zip`
  - **Implementation class name** = `COM.ibm.db2.jdbc.DB2XADataSource`

### Using JTA drivers for DB2 on UNIX

To enable JTA drivers on UNIX, follow these steps:

1. Stop all DB2 services.
2. Stop the IBM WebSphere Application Server administrative service.
3. Stop any other processes that use `db2java.zip` file.
4. Make sure that you already [enabled JDBC 2.0](#).
5. Start the DB2 services.
6. Bind the necessary packages to the database. From the **DB2 Command Line Processor** window, issue the following commands:

```
db2=> connect to mydb2jta
db2=> bind db2home\bnd\@db2cli.lst
db2=> bind db2home\bnd\@db2ubind.lst
db2=> disconnect mydb2jta
```
7. When you use an IBM WebSphere Application Server administrative client (such as the WebSphere Administrative Console) to configure a JDBC driver, specify the following settings:
  - **Server class path** = `$INSTHOME/sqllib/java12/db2java.zip`  
For example, if `$INSTHOME` is `/home/test`, the path will be `/home/test/sqllib/java12/db2java.zip`
  - **Implementation class name** = `COM.ibm.db2.jdbc.DB2XADataSource`

### For Oracle 8.1.7 two phase commit support

The Oracle 8.1.7 thin driver can be used for JTA two phase commit support with the following restrictions:

- The thin driver that comes shipped with 8.1.7 may or may not work. Future patches from Oracle may work as well, but have not been tested. The driver that was available from the Oracle Technology Network Web site as of February 20, 2001 does work and is the recommended driver. Later versions on this Web site are expected to work, but have not been tested.

To obtain the driver from the Oracle support Web, visit:

<http://technet.oracle.com/>

You will need to be a registered user for the Oracle Technology Network to get the driver from this site. Contact Oracle for how to get access. After

you have access:

1. On the left hand side of the screen, select "Software".
2. On "Download Oracle Products, Drivers, and Utilities"
3. On the "Select a Utility or Driver" selection, select "Oracle JDBC Drivers".
4. Download the 8.1.7 driver for the platforms you use and follow the instructions for installing the new driver.

The above instructions are subject to change with no notice to IBM. This version of the instructions could become inaccurate. Consult the Oracle site for the definitive instructions.

- The 8.1.7 driver must be used with 8.1.7 databases. 8.1.6 databases do not support the recover() and forget() methods and other problems have been encountered running with 8.1.6. Oracle does not support JTA with 8.1.6.
- For Oracle, JTA can only be used with container managed (CMP) beans.
- In order for the bean to create the table, the bean must first be started the JTA set to false. After the bean has created the table, JTA can be set back to true.
- An entity bean that accesses Oracle with JTA set to true must be configured as follows:
  - In the deployment descriptor properties, under Transactions, under the Remote tab, set the Transaction Attribute to TX\_REQUIRED.
  - Under Isolation, under the Remote tab, set the Isolation Level to TRANSACTION\_READ\_COMMITTED.
- A session bean that is used with an entity bean that accesses Oracle with JTA set to true must be configured as follows:
  - In the deployment descriptor properties, Under Transactions, under the Remote tab, set the Transaction Attribute to TX\_BEAN\_MANAGED.
  - Under Isolation, under the Remote tab, set the Isolation Level to TRANSACTION\_READ\_COMMITTED.

## Using JTA drivers for Sybase on AIX

To enable JTA drivers for use with Sybase on the AIX operating system, follow these steps:

1. At a command prompt, enable the Data Transaction Manager (DTM) by issuing these commands (one per line):

```
isql -Usa -Ppassword -Sservername      sp_configure "enable DTM", 1      go
```

2. Stop the Sybase Adaptive Server database and start it again.

3. At a command prompt, grant the appropriate role authorization to the EJB user:

```
isql -Usa -Ppassword -Sservername      grant role dtm_tm_role to EJB      go
```


## Notes about Sybase JTA drivers

Do not use a Sybase JTA connection in an enterprise bean method with an unspecified transaction context. A Sybase JTA connection does not support the local transaction mode. The implication is that the Sybase JTA connection must be used in a global transaction context.



## 6.6.14.6: Notes about various databases

This article provides miscellaneous tips for using supported databases. See also the related links.

-  Always consult the [product prerequisites Web site](#) for a list of the database brands and versions that are supported by your particular Application Server version, edition, and fix pack.
- If using local DB2 databases for data access by session clients, in some cases, multiple connections for session clients cannot be established successfully. To avoid stale connections when there are large numbers of session clients, catalog the DB2 databases using a TCP/IP loopback.
  1. Set up a TCP/IP port in `/etc/services`, if a port for remote DB2 clients has not been established yet.
  2. Ensure that the TCP/IP communication protocol has been specified in the DB2COMM registry parameter.
    - To check the current setting of the DB2COMM parameter, enter `db2set DB2COMM`.
    - To update the DB2COMM registry variable to include TCP/IP, use the `db2set` command.

For example:

```
db2set DB2COMM=existing_protocol_names, tcpip
```

3. Update the SVCENAME database manager configuration parameter to the connection service name as defined in `/etc/services` (step 1). For example:

```
db2 update dbm cfg using svcename connection service name
```

4. Catalog the loopback node. For example:

```
db2 catalog tcpip node node_name remote 127.0.0.1 server connection_service_name
```

Reviewers: the above statement replaced the following statement, which was from V3.5.x. Can someone say whether the old stmt should be kept for V3.5.5 docs, or was incorrect and should be replaced with the V4.0 stmt?

```
db2 catalog tcpip node node_name remote 127
```

5. Catalog the database as follows:

```
db2 catalog db database_name as database_alias          db2 uncatalog db database_name
db2 catalog db database_alias as database_name at node node_name
```

This allows you to implement a TCP/IP loopback without needing to change the application to connect to the new alias and the USER and USING parameters.

6. Stop DB2 and start it again to refresh the directory cache.

- When using Sybase 11.x, you might encounter the following error when HttpSession persistence is enabled:

```
DBPortability W Could not create database table: "sessions" com.sybase.jdbc2.jdbc.SybSQLException:
The 'CREATE TABLE' command is not allowed within a multi-statement transaction in the 'database_name'
database
```

where 'database\_name' is the name of the database for holding sessions.

If you encounter the error, issue the following commands at the Sybase command line:

```
use database_name go
sp_dboption db, "ddl in tran ", true
```

- Sybase 12.0 does not support local transaction modes with a JTA enabled data source. To use a connection from a JTA enabled data source in a local transaction, Sybase patch EBF9422 must be installed.

## 6.6.14.8: Recreating database tables from the exported table DDL

To recreate database tables from the exported table DDL, you execute the exported table DDL file to create a table for CMP beans by hand. For instructions for exporting the table DDL, see the following:

- [Exporting DDLs of EJB modules](#)

Create the tables in the database of the data source configuration in the following order:

1. If you specified data source for the CMP bean, then use the data source for the bean.
2. If you specified data source for the EJB module of that CMP bean, then use the data source for the EJB module.
3. If neither of the above two conditions applies, use the default data source for the EJBContainer of the application server onto which the EJM module was installed.

The content of the table DDL file is the SQL strings needed to create a table. The content differs for different databases. The DDL file is essentially an SQL file, and each database has a different command to execute an SQL file.

Copy or paste the content of the table DDL file to a database command line, or use the command line option to take a SQL file as a parameter. The syntax of the command for supported databases is:

### For DB2:

```
db2 -tf table_DDL_file_name
```

### For Oracle:

```
sqlplus user_name/password
```

After a new window displays:

```
@table_DDL_file_name
```

### For Sybase:

```
isql -Uuser_name -Ppassword -Sserver_name -i table_DDL_file_name
```

## 6.6.14.9: Administering data source providers and data sources with the Application Client Resource Configuration Tool

Use the Application Client Resource Configuration Tool to edit the configurations of data source providers (such as JDBC providers) and data sources, which are used by your application clients to access data from databases.

Work with objects of this type by locating them in the tree that is displayed by the tool when you use it to open an EAR file. If the file with which you are working contains data source providers and data sources, its tree will contain one or more of the following:

**Resources** -> *application.jar* -> **Data Source Providers** -> *data\_source\_provider\_instance*

where *data\_source\_provider\_instance* is a particular data source provider.

If you expand the tree further, you will also see the **Data Sources** folders containing the data source instances for each data source provider instance.

## 6.6.14.9.1: Configuring new data source providers (JDBC drivers) with the Application Client Resource Configuration Tool

During this task, you will create new data source providers (also known as JDBC drivers) for your client application. Note, in a separate administrative task, the Java code for the required data source provider must be installed on the client machine on which the client application resides.

To configure a new data source provider:

1. [Start the tool and open the EAR file](#) for which you want to configure the new data source provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file in which you want to configure the new data source provider.
3. Expand the JAR file to view its contents.
4. Click the folder called **Data Source Providers**. Do one of the following:
  - Right-click the folder and select **New Provider**.
  - On the menu bar, click **Edit -> New**.
5. In the resulting property dialog, configure the [data source provider properties](#).
6. When finished, click **OK**.
7. On the menu bar, click **File -> Save** to save your changes.

## 6.6.14.9.1.1: Configuring new data sources with the Application Client Resource Configuration Tool

During this task, you will create new data sources for your client application.

1. In the tree, click the data source provider for which you want to create a data source.
  - [Configure a new data source provider.](#)
  - Or, click an existing data source provider.
2. Expand the data source provider to view its **Data Sources** folder.
3. Click the folder. Do one of the following:
  - Right-click the folder and select **New Factory**.
  - On the menu bar, click **Edit** -> **New**.
4. In the resulting property dialog, configure the [data source properties](#).
5. When finished, click **OK**.
6. On the menu bar, click **File** -> **Save** to save your changes.

## 6.6.14.9.3: Removing data source providers (JDBC drivers) and data sources with the Application Client Resource Configuration Tool

Please see "[Removing objects from EAR files with the Application Client Resource Configuration Tool](#)", as this task is similar for all object types supported by the tool.

## 6.6.14.9.4: Updating data source and data source provider configurations with the Application Client Resource Configuration Tool

During this task, you will modify (update) the configuration of an existing data source or data source provider.

1. [Start the tool and open the EAR file](#) containing the data source or data source provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file containing the data source or data source provider that you want to update.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular data source or data source provider that you want to update. When you find it, do one of the following:
  - Right-click the object and select **Properties**
  - On the menu bar, click **Edit -> Properties**
5. In the resulting property dialog, update the properties. For detailed field help, see:
  - [Data source provider properties](#)
  - [Data source properties](#)
6. When finished, click **OK**.
7. On the menu bar, click **File -> Save** to save your changes.

## 6.6.15: Administering custom services (overview)

Configuring a custom service provides the ability to plug into a WebSphere application server to define a hook point to be executed at server startup and shutdown. It allows configurable extensions to the application server runtime.

The application server runtime will load the classname specified in the [custom service properties](#) and call its `initialize()` method, passing in a `Properties` object that contains the configuration information for the service. The configuration information for the custom service can be arbitrary name-value pair properties, or information contained in an external file, such as an XML file on disk somewhere.



## 6.6.15.0: Properties of custom services

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Classname**



The classname of the service implementation. The class must implement the Services API of the product.

**Classpath**



Classpath for locating the ZIP and JAR files that contain the classes implementing this service

**Configuration File URL**



**or External Config URL**



The URL of a custom service configuration file

Each application server has a separate file containing the configuration properties for its custom service. Different application servers can run with the same custom service, but with different custom service configuration values. Specify the URL for an external file containing addition custom service configuration data that could be more complex than simple name-value config properties.

**Custom Properties**



**or Dynamic Properties**



Name-value pairs that will be passed to the init() method of the service

**Description**



An optional description for the custom service

**Display Name**



**or Name**



A logical name for the service

**Enable**



**or Enabled**



Whether the server will attempt to start and initialize the specified service

**External Config URL**



See Configuration File URL

**Name**



See Display Name

See also the other [application server properties](#).

## 6.6.15.3: Administering custom services with the Web console

Use the Web console to create and edit the configurations of custom services.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Custom Services**

Note, if a custom service is not *enabled* (using a setting in the custom service configuration), then all other custom service related properties will be ignored.

## 6.6.15.3.1: Configuring new custom services with the Web console

During this task, you will update the configuration of an existing application server to include custom services that you define. The custom services will start and stop in unison with your application server.

To configure a new custom service:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Custom Services**.
2. Click the **New** button.
3. Specify the custom service properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.15.3.3: Removing custom services with the Web console

During this task, you will remove an existing custom service so that it no longer starts and stops with your application server.

To remove a custom service:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Custom Services**.
2. Select the check box next to the name of the custom service to remove.
3. Click the **Delete** button.
4. Click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.15.3.4: Updating custom service configurations with the Web console

During this task, you will update the configuration of an existing application server to include custom services that you define. The custom services can start and stop in unison with your application server.

To update custom service configurations:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* -> **Custom Services**.
2. From the list of custom services, click the service that you want to configure.
3. Adjust the custom service properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.16: Administering virtual hosts (overview)

To define a virtual host, the administrator specifies information such as:

- An administrative name for the virtual host
- One or more domain name aliases for the virtual host
- Mime types and extensions to recognize
- Whether this is the default virtual host

### Ensuring there is a virtual host alias for each HTTP transport port

There must be a virtual host alias corresponding to each port being used by an HTTP transport. There is one HTTP transport in each Web container, with one Web container in each application server.

The following procedure describes how to find out (or set) the port for the HTTP transport, then create a corresponding virtual host alias. You will need to do so in the following cases:

- You are using the internal HTTP transport (described in [the transport administration overview](#)) with a port other than the default of 9080, or for some reason the virtual host does not contain the usual entry for port 9080

To discover or edit the transport port number for a given application server, and then create an alias corresponding to the port number:

1. [View or edit the transport properties](#). Note the value in the field named **Port** or **Transport Port**, such as 9082.
2. [Configure the virtual host](#) to contain an [alias](#) for the port number, such as \*:9082, if 9082 is port number in use by transport.
3. When you enter the URL for the application into a Web browser, include the port number in the URL, such as:

`http://localhost:9082/wlm/SimpleServlet`

using the port number from the previous step.

## 6.6.16.0: Properties of virtual hosts

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Aliases



The list of one or more DNS aliases by which the virtual host is known. For example: \*:9082

### Default MIME Entries



A collection of MIME type extension mappings defined for the virtual host. If MIME entries are not specified at the Web module level, these MIME entries will apply.

### Extension



or Extensions



A list of file extensions that are mapped to the MIME type

### Host Aliases



Configuration for aliases. An alias is the DNS host name and port number used by a client to form the URL request for a Web application resource (such as a servlet, JSP, or HTML page). For example, it is the "myhost:8080" portion of http://myhost:8080/servlet/snoop. When no port number is specified, the default port 80 is used.

### Host Name



The IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JSP, or HTML page). For example: www.VHTest.ibm.com

### MIME Entry



or MIME Types



Configuration for the mapping of MIME types to extensions

### MIME Types



or Type



A MIME type that is mapped to the specific extension. For example: text, image, or application

### Name



A logical name used for configuring Web applications to a particular host name. The default virtual host is typically suitable for most simple configurations.

### Port



A port value can be specified in conjunction with the host name to indicate the port for which the Web server has been configured to accept client requests.

### Type



See MIME type

## 6.6.16.3: Administering virtual hosts with the Web console

Use the Web console to work with the configurations of virtual hosts.

Work with objects of this type by locating them in the tree on the left side of the console:

Click the **Virtual Hosts** entry in the tree.



## 6.6.16.3.1: Configuring new virtual hosts with the Web console

During this task, you will define a new virtual host configuration.

To configure a new virtual host:

1. Click **Virtual Hosts** in the tree on the left side of the console. A list of virtual hosts will be displayed on the right side of the console.
2. Click the **New** button displayed on the right side of the console, above the list of installed applications.
3. Specify settings for the new virtual host.
4. When you are finished configuring the virtual host, click **OK**.
5. Verify that the virtual host has been added to the list of virtual hosts.
6. [Save your configuration.](#)
7. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.16.3.3: Removing virtual hosts with the Web console

During this task, you will remove a virtual host configuration.

To remove a virtual host:

1. Click **Virtual Hosts** in the tree on the left side of the console. A list of virtual hosts will be displayed on the right side of the console.
2. From the list, select the virtual host that you want to remove. Select it by clicking the check box to the left of the virtual host name.

You can delete more than one virtual host at a time. Be sure that no other virtual hosts are selected for removal unless you want to remove them.

3. Click the **Delete** button displayed on the right side of the console, above the list of installed applications.
4. Click **OK**.
5. Verify that the virtual host has been removed from the list of virtual hosts.
6. [Save your configuration.](#)
7. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.16.3.4: Updating virtual host configurations with the Web console

During this task, you will change the settings of a virtual host.

To update virtual host settings:

1. Click **Virtual Hosts** -> *virtual\_host\_name* in the tree on the left side of the console, where *virtual\_host\_name* is the name of the virtual host that you want to update. Its properties will be displayed on the right side of the console.
2. Modify the virtual host properties.
3. When you are finished, click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.18: Securing applications

For purposes of security, Application Server categorizes assets into two classes: resources and applications.

- *Resources* are individual components, such as servlets and enterprise beans.
- *Applications* are collections of related resources.

Security can be applied to applications and to individual resources. Setting up security involves the following general steps:

1. Setting global values for use by all applications.
2. Refining settings for individual applications.

Securing applications with IBM WebSphere ApplicationServer product security involves a series of tasks. Completing the tasks results in a set of policies defining *which* users have access to *which* methods or operations in *which* applications.

For example, the security administrator establishes policies specifying whether the user *Bob* is permitted to use the company's Inventory application to perform a write operation, such as changing the number of units of merchandise recorded in the company's inventory database.

The product security server works with the selected user registry or directory product to enforce the policies whenever a user tries to access a protected application. For example, *Bob* might be prompted for a digital certificate verifying his identity when he tries to use the Inventory application.

## 6.6.18.0: General security properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Cache Timeout or Security Cache Timeout

Time after which the authentication cache will be refreshed. Caching can improve performance with respect to authentication lookups.

Specify this value in seconds, with a minimum of 30.

### Default SSL Configuration or Use global SSL default configuration

Apply the default SSL configuration to the entire administrative domain.

For *Advanced Edition*, see [Configuring SSL support instructions](#).

### Enabled or Enable Security

Whether global security is enabled. When security is not enabled, all other security settings are not validated or used.

### Security Cache Timeout

See Cache Timeout

### Use Domain Qualified User Names

When the value of this setting is true, user names returned by calls such as `getUserPrincipal()` will be qualified with the security domain in which they reside

### Use global SSL default configuration

See the Default SSL Configuration field description



## 6.6.18.0.2: Properties for configuring security using local operating system

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool


### Authentication Mechanism



Select how to authenticate users that try to access applications.

- Against the local operating system user registry, or
- Against an LTPA based LDAP registry or custom registry

Note that the local operating system user registry is intended for single machine and single application server environments. *Advanced Single Server Edition* supports only the local operating system mechanism.

 When form-based login is used with local operating system authentication, the user information is stored in the HTTP session. Using an HTTP connection is not very secure, meaning the information can be obtained by others. Using SSL connections (HTTPS) between the browser and the Web server will improve security.

### Security Server ID



or Server ID



The user ID under which the server runs, for security purposes. This ID is not associated with the system process. This ID refers to the application security context within the WebSphere Application Server product.

If using local operating system authentication, the following conditions apply:

- On UNIX operating systems, the ID must be root or have root authority.
- On Windows operating systems, the account must be a member of the Administrators group and must have the rights to "Log on as a service" and "Act as part of the operating system." If the Windows machine is a member of an NT domain, then the ID must also be an administrator in the NT domain. Do not use an account whose name matches the name of your machine or Windows Domain.

If using LDAP or custom registry authentication (not available for *Advanced Single Server Edition*), the following conditions apply:

- The user should be a valid user in the LDAP or custom registry
- The user should *not* be a root DN or administrator DN because those users are not always in the directory in all LDAP implementations.

### Security Server Password



or Server Password



The password corresponding to the server ID

## 6.6.18.1a.7: Configuring SSL in WebSphere Application Server

- "What is Secure Socket Layer?" and related concepts
- Overview: WebSphere Application Server's use of SSL
- Configuring SSL for browsers
- Configuring SSL for Web servers
- Configuring SSL for IBM HTTP Server, specifically
- Configuring SSL for WebSphere plug-ins for Web servers
- Configuring SSL for WebSphere Application Server

### Overview: WebSphere Application Server's use of SSL

SSL (Secure Socket Layer) is used by several WebSphere Application Server components in order to provide secure communication. In particular, SSL is used by:

- HTTPS: the application server's built-in HTTPS transport.
- ORB: the application server's client and server ORB.
- LDAPS: the admin server's secure connection to the LDAP registry used for authentication. This is available only in WebSphere Application Server Advanced Edition.

The administrative model in WebSphere Application Server allows these various SSL components to be centrally managed by configuring the *default SSL Settings*. Furthermore, any of the *default settings* can be overridden by configuring the specific SSL settings for HTTPS, ORB, and LDAPS. This provides both central administration as well as individual configurability which may be required for the various uses of SSL.

### Configuring SSL for the browser


Configuring SSL for the browser is browser-specific. Consult your browser documentation for instructions.

Generally speaking, when the you type "https://..." instead of "http://...", the browser creates an SSL connection instead of a simple TCP connection to the Web server. The browser then typically either prompts the user or fails to connect if it was unable to validate the Web server or to agree upon the level of security options (the strength of the encryption algorithm to use).

### Configuring SSL for the Web server

Configuring SSL for the Web server depends on the type of Web server. Consult your Web server documentation for instructions.

Generally speaking, when SSL is enabled, an SSL key file is required. This key file should contain both the CA certificates (signer certificates) as well as any personal certificates. Client authentication can also be enabled; by default, it is disabled.

 In order for the client certificate (the certificate from the browser) to be forwarded by the WebSphere Web server plug-in to the WebSphere Application Server, client authentication must be enabled for the Web server. Enabling client authentication in WebSphere Application Server itself is not required unless you want to authenticate the WebSphere Web server plug-in (or any other clients connecting directly to the WebSphere Application Server over SSL).

### Configuring SSL for IBM HTTP Server, specifically

This section provides a brief example of configuring SSL for IBM HTTP Server. See the IBM HTTP Server documentation for the most recent and complete instructions. Note also that the *httpd.conf.sample* file of your Web server provides examples of all directives, including the SSL-related directives.

1. Create a keyfile using the IHS key management utility.
  1. Create a directory at a location such as "*product\_installation\_root*/myKeys"

This directory will be used to hold all of your SSL key files and certificates.

2. Start the Key Management Utility from the IBM HTTP Server start menu.

To start this utility on a Windows platform, click: **Start -> Programs -> IBM HTTP Server -> Start Key Management Utility**


3. Click the **Key Database File** menu and select **New**.

4. Specify settings and click **OK**:

- Key Database Type: CMS Key Database File
- File Name: WebServerKeys.kdb
- Location: The path to your "myKeys" directory

5. Enter a password for your SSL key file (twice for confirmation).

6. Check the "Stash the password to a file?" option. Click **OK**.

 This causes a file named "WebServerKeys.sst" to be created containing an encoded form of the password. Note that this encoding prevents a casual viewing of the password but is not highly secure. Therefore, operating system permissions should be used to prevent all access to this file by unauthorized persons.



7. When you see the list of default **Signer Certificates**, click the **Signer Certificates** menu and select **Personal Certificates**.

If you have a server certificate from a CA (for example, Verisign), you can click **Import** to import this certificate into your SSL key file. You will be prompted for the type and location of the file containing the server certificate.

If you do not have a valid server certificate from a CA, but want to test your system, click **New Self-Signed**.

You will be prompted minimally to enter a **Key Label** such as "Test" and **Organization**, such as "IBM". Choose to use the default values for other values.

8. Click the **Key Database File** menu and select **Close**.

2. Add the following lines to the bottom of your httpd.conf file:

```
LoadModule ibm_ssl_module modules/IBModuleSSL128.dll          Listen 443          SSLEnable
Keyfile "product_installation_root/myKeys/WebServerKeys.kdb"  # SSLClientAuth required
```

This causes the Web server to listen on port 443 (the default SSL port).

Uncomment the last line containing "SSLClientAuth required" if you want to enable client authentication. This will cause IHS to send a request for a certificate to the browser. Your browser may prompt you to choose a certificate to send to the Web server in order to perform client authentication.

3. Start your IBM HTTP Server.
4. Test your configuration from a browser by entering a URL such as:

`https://localhost`

If you are using a self-signed certificate, instead of a certificate issued by a CA such as Verisign, then your browser should prompt you to see if you want to trust the unknown signer of the server's certificate. Additionally, if you enabled client authentication, then your browser may prompt you to select a certificate to send to the Web server in order to perform client authentication. The page should then be displayed.

## Configuring SSL for WebSphere plug-ins for Web servers

After SSL is working between your browser and Web server, proceed to configure SSL between the Web server plug-in and the WebSphere Application Server product. This is not required if the link between the plug-in and application server is known to be secure or if your applications are not sensitive. If privacy of application data is a concern, however, this connection should be an SSL connection.

### Step 1: Creating an SSL key file for the WebSphere Web server plug-in

When configuring SSL, you must first create an SSL key file.

Note that if you are using the IBM HTTP Server, you *may* use the same SSL key file which the Web server is using; however, it is recommended that separate SSL key files be used because the trust policy for the connection to the web server will likely be different than the trust policy for the connection to the application server.

For example, we may want to allow many browsers to connect to the Web server's HTTPS port, whereas we only want to allow a small, well-known number of WebSphere plug-ins to connect directly to a WebSphere application server's HTTPS port. The following is an example of how to create an SSL key file for your WebSphere plug-in which will only allow the plug-in to connect to the application server on its SSL port.

1. Create the directory `product_installation_root\myKeys` if you have not already done so.

This directory will contain all of the SSL key files and extracted certificates that you will create.

2. Start the key management utility of GSKit.

GSKit is the SSL implementation used by the WebSphere plug-in, which is the same implementation used by the IBM HTTP Server.

The default path on Windows to this utility is `C:\Program Files\ibm\gsk5\bin\gsk5ikm.exe`.

3. Click the **Key Database File** pulldown and select **New**.
4. Specify settings and click **OK**:
  - **Key database type**: CMS Key Database File
  - **File name**: plug-inKeys.kdb
  - **Location**: your myKeys directory
5. Enter a password for your SSL key file (twice for confirmation).
6. Check the **Stash the password to a file?** option. Click **OK**.

This causes a file such as `"product_installation_root\myKeys\plug-inKeys.sth` to be created containing an encoded form of the password. This encoding prevents a casual viewing of the password but is not highly secure. Therefore, operating system permissions should be used to prevent all access to this file by unauthorized persons.

7. When you see the list of default **Signer Certificates**, select the first certificate and click **Delete**.
8. Repeat the previous step until all of the signer certificates have been deleted.
9. Create a self-signed certificate:
  1. Click the **Signer Certificates** menu and select **Personal Certificates**.
  2. Click **New Self-Signed**.
  3. Enter "plug-in" for the **Key Label** and "IBM" for the **Organization**.

4. Click **OK**.
10. Extract the certificate so that you can import it into the application server key file later.
  1. Click **Extract Certificate**.
  2. Specify settings:
    - **Base64-encoded ASCII data:** Data Type
    - **Certificate file name:** plug-in.arm
    - **Location:** path to your myKeys directory
  3. Click **OK**.
11. Click the **Key Database File** menu and select **Close**.

## Step 2: Modifying the WebSphere Web server's plug-in configuration file


Now that you have created the SSL key file for the plug-in, edit the [plug-in configuration file](#) so that it references your key file.

The following is an example of the plug-in configuration file. This configuration causes the plug-in to forward HTTP requests to the HTTP port of the application server, and to forward HTTPS requests to the HTTPS port of the application server.

The SSL configuration information is specified for *secureServer1*, which is the only member of the *secureServers* group. All HTTPS requests are forwarded to the *secureServers* group. (A server group is a concept that is supported only in *Advanced Edition*, not in *Advanced Single Server Edition*.)

The SSL key file is specified by the **keyring** property, and the stash file (which contains an encoded password) is specified by the **stashfile** property. Make sure that the path of this file is specified in your Web server configuration (for example, in "httpd.conf" for IHS).

```
<?xml version="1.0"?> <Config>          <Log LogLevel="Error"
Name=<"product_installation_root\logs\native.log"> <VirtualHostGroup Name="standardHost">
<VirtualHost Name="*:80"/>          </VirtualHostGroup>          <VirtualHostGroup Name="secureHost">
<VirtualHost Name="*:443"/>          </VirtualHostGroup>          <UriGroup Name="WebSphereURIs">          <Uri
Name="/servlet/snoop/*"/>          <Uri Name="/servlet/snoop"/>          <Uri
Name="/servlet/snoop2/*"/>          <Uri Name="/servlet/snoop2"/>          <Uri Name="/servlet/hello"/>
<Uri Name="/ErrorReporter"/>          <Uri Name="/servlet/*"/>          <Uri Name="/servlet"/>
<Uri Name="*.jsp"/>          <Uri Name="/j_security_check"/>          <Uri Name="/webapp/examples"/>
<Uri Name="/WebSphereSamples"/>          <Uri Name="/WebSphereSamples/SingleSamples"/>          <Uri
Name="/theme"/>          </UriGroup>          <ServerGroup Name="standardServers">          <Server
Name="standardServer1">          <Transport Hostname="localhost" Port="9080" Protocol="http"/>
</Server>          </ServerGroup>          <ServerGroup Name="secureServers">          <Server
Name="secureServer1">          <Transport Hostname="localhost" Port="9443" Protocol="https">
<Property name="keyring" value="product_installation_root\myKeys\plug-inKeys.kdb">
<Property name="stashfile" value="product_installation_root\myKeys\plug-inKeys.sth">
</Transport>          </Server>          </ServerGroup>          <Route VirtualHostGroup="standardHost"
UriGroup="WebSphereURIs" ServerGroup="standardServers"/>          <Route VirtualHostGroup="secureHost"
UriGroup="WebSphereURIs" ServerGroup="secureServers"/>          </Config>
```

 The XML implementation of the plug-in configuration file could change before this documentation is updated again. Consult the actual configuration file installed on your system with your current product version and fix pack level as the most current and correct version of the XML syntax.

## Configuring SSL for WebSphere Application Server

The [administrative console](#) provides the following access points to SSL settings.

Use the Default SSL Settings to centrally manage SSL settings for resources in the administrative domain. Any of the default settings can be overridden in the settings for an individual resource type -- the transport or ORB settings.

- Default SSL Settings

In the console tree view, click **Security -> Default SSL Settings**.

- HTTPS SSL settings for the HTTP transport of a Web container

[Display the transport properties](#). Click **SSL**.

- ORB SSL settings

[Display the ORB settings](#). Click **Secure Socket Layer Settings**.

The above settings that can be configured through any of these SSL settings is described by the:

- [SSL property reference](#)

In the SSL settings dialog, note the **Crypto Token** button for configuring settings for [supported cryptographic devices](#).

## Configuring SSL for the application server's HTTPS transport

In order to configure SSL, you must first create an SSL key file. The contents of this file depend on whom you want to allow to communicate *directly* with the application server over the HTTPS port (in other words, you are defining the HTTPS server security policy).

This article presents a restrictive security policy, in which only a well-defined set of clients (the WebSphere plug-ins for the Web server) are allowed to connect to the application server HTTPS port. The following procedure for creating an SSL key file without the default signer certificates follows that restrictive trend.

#### Step 1: Create an SSL key file without the default signer certificates

1. Start IKeyMan.

On Windows, start IKeyMan from the WebSphere Application Server entry on the Windows Start menu.

2. Create a new key database file.
  1. Click **Key Database File** and select **New**.
  2. Specify settings:
    - **Key database type:** JKS
    - **File Name:** appServerKeys.jks
    - **Location:** your myKeys directory, such as "*product\_installation\_root*\myKeys
  3. Click **OK**.
  4. Enter a password (twice for confirmation) and click **OK**.
3. Delete all of the signer certificates.
4. Click **Signer Certificates** and select **Personal Certificates**.
5. Add a new self-signed certificate.
  1. Click **New Self-Signed** to add a self-signed certificate.
  2. Specify settings.
    - **Key Label:** appServerTest
    - **Organization:** IBM
  3. Click **OK**.
6. Extract the certificate from this self-signed certificate so that it can be imported into the plug-in's SSL key file.
  1. Click **Extract Certificate**.
  2. Specify settings:
    - **Data Type:** Base64-encoded ASCII data
    - **Certificate file name:** appServer.arm
    - **Location:** the path to your myKeys directory
  3. Click **OK**.
7. Import the plug-in's certificate.
  1. Click **Personal Certificates** and select **Signer Certificates**.
  2. Click **Add**.
  3. Specify settings:
    - **Data Type:** Base64-encoded ASCII data
    - **Certificate file name:** appServer.arm
    - **Location:** the path to your myKeys directory
  4. Click **OK**.
8. Enter "plug-in" for the label and click **OK**.
9. Click **Key Database File**.
10. Select **Exit**.

#### Step 2: Add the signer certificate of the application server to the plug-in's SSL key file

1. Start the key management utility.
2. Click the **Key Database File** menu and select **Open**.
3. Select the file *product\_installation\_root*\myKeys\plug-inKeys.kdb.
4. Enter the associated password and click **OK**.
5. Click **Personal Certificates** and select **Signer Certificates**.
6. Click **Add**.
7. Specify settings.
  - **Data Type:** Base64-encoded ASCII data
  - **Certificate File Name:** appServer.arm
  - **Location:** the path to your myKeys directory.
8. Click **OK**.
9. Click **Key Database File** and select **Exit**.

#### Step 3: Reference the key file in WebSphere Application Server systems administration

Reference the appropriate SSL key file in the default SSL settings configuration panel or in the HTTPS SSL settings configuration panel. Here, we will use the default SSL settings panel.

1. [Start the administrative console.](#)
2. In the tree view, click **Security** -> **Default SSL Settings**.
3. Specify settings.
  - **Key File Name:** *product\_installation\_root*/myKeys/appServer.jks
  - **Key File Password:** *enter your password*
  - **Key File Format:** JKS
  - **Trust File Name:** (empty)
  - **Trust File Password:** (empty)
  - **Client Authentication:** selected
4. Click **OK**.
5. Save your server configuration.

#### Step 4: Stop the servers and start them again

The configuration is complete. In order to activate the configuration, stop and restart both the Web server and the application server.

## 6.6.18.3: Administering security with the Web console

Use the Web console to enable and disable global security, using the local operating system registry to authenticate users. After enabling security, access to this administrative console will be guarded by a login screen.

Work with security configurations by locating them in the tree on the left side of the console:

Click the **Security** entry in the tree.

## 6.6.18.3.1: Enabling global security with the Web console

To enable global security and configure default SSL settings:

1. In the tree on the left side of the console, click **Security**.
2. [Specify the user ID and password under which the application server will run.](#)
3. Specify the security settings. Be sure to select the check box next to **Security enabled**.
4. Click **OK**.
5. In the tree on the left side of the console, click **Security -> Default SSL Settings**.
6. Specify the security settings.
7. Click **OK**.
8. [Save your configuration.](#)
9. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.18.3.3: Removing global security with the Web console

To enable global security and configure default SSL settings:

1. In the tree on the left side of the console, click **Security**.
2. Specify the security settings. Be sure that the check box next to **Security enabled** is **not** selected.
3. Click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.18.3.6: Specifying user IDs for the server and administrator with the Web console

Before enabling security, specify an ID and password under which the application server will run:

1. In the tree on the left side of the console, click **Security** -> **Local OS Authentication** to display the authentication settings.
2. Click the **Local OS User Registry** link.
3. On the resulting panel, specify a valid **Server ID** and **Server Password**. The user ID-password pair must be defined already in the user registry of the local operating system, and must have "Act as Operating System" privileges (on Windows NT).
4. Click **OK**.

You can use the same ID and password to log in to the Web administrative console, or you can set up a different ID and password.

### Setting up another user ID for logging in to the Web console

To set it up so that you can log in to the administrative console using an ID and password other than the server ID and password:

1. Expand the tree on the left side of the console to display **Nodes** -> *hostname* -> **Enterprise Applications** -> **Server Administration Application**.
2. In the property sheet for the application, locate and click the **Mapping Roles to Users** task to display the panel for mapping roles to users.
3. For the Administrator role, specify values for the users, groups, and other settings. This is where you can specify additional IDs other than the server ID.
4. When finished changing the settings, click **Next** to display the confirmation page.
5. When finished confirming the settings, click **Finish**. The modifications will be saved to the EAR file for the application.
6. [Save your configuration.](#)
7. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)




## 6.6.18.6: Avoiding known security risks in the runtime environment

### Securing the properties files

WebSphere Application Server depends on several configuration files created during installation. These files contain password information and should be protected accordingly. Although the files are protected to a limited degree during installation, this basic level of protection is probably not sufficient for your site. You should ensure that these files are protected in compliance with the policies of your site.

The files are found in the bin and properties subdirectories in the WebSphere `<product_installation_root>`. The configuration files include:

- In the bin directory: admin.config
- In the properties directory:
  - sas.server.props
  - sas.client.props
  - sas.server.props.future

 Failure to adequately secure these files can lead to a breach of security in your WebSphere applications.

### Securing properties files on Windows NT

To secure the properties files on Windows NT, follow this procedure for each file:

1. Open the Windows Explorer for a view of the files and directories on the machine.
2. Locate and right-click the file to be protected.
3. On the resulting menu, click Properties.
4. On the resulting dialog, click the Security tab.
5. Click the Permissions button.
6. Remove the Everyone entry.
7. Remove any other users or groups who should *not* be granted access to the file.
8. Add the users who should be allowed to access the file. At minimum, add the identity under which the administrative server runs.

### Securing properties files on UNIX systems

This procedure applies only to the ordinary UNIX filesystem. If your site uses access-control lists, secure the files by using that mechanism.

For example, if your site's policy dictates that the only user who should have permission to read and write the properties files is the root user, to secure the properties files on a UNIX system follow this procedure for each file:

1. Go to the directory where the properties files reside.
2. Ensure that the desired user (in this case, root) owns each file and that the owner's permissions are appropriate (for example, rw-).
3. Delete any permissions given to the "group".

4. Delete any permissions given to the "world".

Any site-specific requirements can affect the desired owner, group and corresponding privileges.

## Risks illustrated by example applications

The level of security appropriate to a resource varies with the sensitivity of the resource. Information considered confidential or secret deserves a higher level of security than public information, and different enterprises will assess the same information differently. Therefore, a security system needs to be able to accommodate a widerange of needs. What is reasonable in one environment can be considered a breach of security in another.

The following describes some user practices and their potential risks. When applicable, it uses components of the example application to illustrate the point.

### Invoker Servlet

*Purpose:* The invoker servlet serves servlets by class name. For example, if you invoke `/servlet/com.test.HelloServlet`, the invoker will load the servlet class (if it is in its classpath) and execute the servlet.

*Security consideration:* By using this servlet, a user can access any other servlet in the application, without going through the proper channels. For example, if `/servlet/testHello` is a URI associated with `com.test.HelloServlet`, and if that URI is protected, user must be authenticated to invoke `/servlet/testHello`, but any user can invoke `/servlet/com.test.HelloServlet`, circumventing the security on the URI. This is a security exposure if you have secured servlets installed in the system.

*Solution:* Avoid installing this servlet in your configuration.

### An application's error page

*Purpose:* In case of application errors, users are redirected to an error page associated with the Web application. This can be any type of Web resource to which customers should be redirected in case of an error.

*Security consideration:* This page should be unprotected. If it is protected, the server cannot authenticate the user from the context and therefore cannot send the user to the error page when an error occurs.

*Solution:* Do not secure these resources.

### The web application "examples"

*Purpose:* This application is available as part of the default installation.

*Security consideration:* The servlets available in this application can export sensitive information, for example, the configuration of your server.

*Solution:* The "examples" Web application should not be installed in a production environment.

### Avoiding other known security risks

This file addresses specific problem areas. As always, periodically check the [product Web site Library page](#) for the latest information. See also the product [Release Notes](#).

- To avoid a security risk, ensure that the WebSphere Application Server document root and the Web server document root are different. Keep your JSP files in the WebSphere Application Server document

root or it will be possible for users to view the source code of the JSP files.

WebSphere Application Server checks browser requests against its list of virtual hosts. If the host header of the request does not match any host on the list, WebSphere Application Server lets the Web server serve the file. Suppose the requested file is a JSP file in the Web server document root -- the JSP file is served as a regular file.

This problem has been noticed in scenarios using Netscape Enterprise Server. Due to the nature of the problem, it is possible that other Web server brands are susceptible.

- **Microsoft Internet Information Server users:**

To use the Microsoft Internet Information Server with security enabled, in combination with IBM WebSphere Application Server security, you need to:

- Configure IIS authentication settings to Anonymous.
- Disable NTLM (Windows NT Challenge/Response) in the Microsoft Management Console
- Disable Basic Authentication on the Microsoft Management Console

Look for the setting on the Directory Security tab of the WWW Services properties.

Problems are common when Internet Information Server NTLM is enabled along with IBM WebSphere Application Server security. The above settings are recommended to avoid problems.

- **Users of Distinguish Names (DN) in LDAP:**

Make sure you use Distinguished Names (DNs) that your directory service product supports. Although WebSphere Application Server security supports valid LDAP DN's, some directory-service products support only a subset. For example, testing revealed that some directory services do not support all valid LDAP DN's. Specifically, a valid DN of the form `OID.9.2.x.y.z=foo` was rejected by one or more of the supported directory services.

Also, directory services vary in how they represent DN's, and DN's are both case- and space-sensitive. In some cases, this leads to situations in which a user enters a valid DN and is authenticated but is still refused access. This problem is often solved by using the Common Name (the short name) rather than the full Distinguished Name.

- **Users of digital certificates with European characters:**

If you use the iKeyman GUI tool to obtain manage certificates that contain European characters in names, the GUI will not display them. For example, a digital certificate contains the name of the company that owns the certificate and the name of the company that issued the certificate. In the US, there are companies that use symbols instead of letters in their names, like @Home and \$mart \$hopper. European characters in certificate names will not be displayed by the GUI.

## 6.6.18.7: Protecting individual application components and methods

### Protecting enterprise beans after redeployment

All methods in enterprise beans and Web applications are unprotected by default.

Security is not automatically updated when changes are made to a bean. It will be updated after the old application is stopped, the new application is deployed into the runtime, and the new application is started.

### Adding a method to a bean

If you add a method to a bean, you must use the Application Assembly Tool to associate the new method with a role.

### Modifying a method on a bean

If you modify a method on a bean, you must use the Application Assembly Tool to ensure that the method still has a role associated with it.

### Unprotecting resources

All methods in enterprise beans and Web applications are unprotected by default. If you have added a single method-to-role mapping to an enterprise-bean method, the user will be given an option to assign "DenyAllRole" role to all other unprotected methods during application installation. If the unprotected methods are assigned the "DenyAllRole" role, then these methods are protected; nobody is permitted to use them. If the unprotected methods are not assigned the "DenyAllRole" role, these methods are not protected and anyone can access those methods.

### Unprotecting an entire application

During application assembly, if you have assigned roles to methods within an application, you have protected those methods. To unprotect the methods, you can do either of the following:

- Use the Application Assembly Tool to remove the method-to-role mappings for every method in the application
- Assign the Everyone subject to all of the roles in the application, either during application installation or using the **Security Center** after installation

### Unprotecting a Method

The only way to unprotect a specific method is to use the Application Assembly Tool to edit the method-to-role mapping. Change the role associated with the method to a different role, one that is associated only with the Everyone subject.

## 6.6.18.9: Specifying authentication options in sas.client.props

You can use the sas.client.props file to direct WebSphere ApplicationServer to authenticate users by prompting or by using a user ID and password set in the properties file. The following steps describe the procedure:

1. Locate the sas.client.props file. By default, it is located in the properties directory under the *<product\_installation\_root>* of your WebSphere Application Server installation.
2. Edit the file to set up the authentication procedure:

- To authenticate by prompting, set the loginSource property to the value "prompt":

```
com.ibm.CORBA.loginSource=prompt
```


Note that when using prompt, a graphical panel is presented for the user for collecting the user ID and password. Pure Java clients must call the JDK API System.exit(0) at the end of the program in order to properly end the Java process. This is because the JDK starts a backward AWT thread that is not killed when the login prompt disappears. If you choose not to use a System.exit(0) call, pressing Ctrl-C ends the process.

- To authenticate by prompting on the console (stdout), set the loginSource property to the value "stdin". The user is then prompted for user ID and password by using a non-graphical console prompt. This is currently only supported by a pure Java client.
- To authenticate by the values configured in the file, set the loginSource property to the value "properties" and set the desired values for the loginUserId and loginPassword properties:

```
com.ibm.CORBA.loginSource=properties          com.ibm.CORBA.loginUserId=<user_ID>
com.ibm.CORBA.loginPassword=<password>
```

3. Save the file.

## 6.6.18.10: The demo keyring

 Do *not* use the demo keyring in production systems. It includes a self-signed certificate for testing purposes, and the private key for this certificate can be obtained easily, which puts the security of all certificates stored in the file at risk. This keyring is intended only for testing purposes.

## 6.6.18.12: Cryptographic token support

To understand how to make WebSphere Application Server (both the runtime and the IKeyMan key management utility) work correctly with any crypto hardware, you should become familiar with the JSSE documentation available from the Application Server product installation:

[product\\_installation\\_root/java/docs/jsse/readme.jsse.ibm.html](#)

Be sure to unzip the file:

[product\\_installation\\_root/java/docs/jsse/native-support.zip](#)

to the appropriate location; otherwise, link errors will occur.

Follow the documentation that accompanies your device in order to install your crypto hardware. Installation instructions for IBM crypto hardware devices can be found

at <http://www.ibm.com/security/cryptocards/html/library.shtml>

The product supports the use of the following cryptographic devices.

These can be used by an SSL client or server:

- IBM 4758-23
- nCipher nForce
- Rainbow Cryptoswift

These can be used by SSL clients:

- IBM Security Kit Smartcard
- GemPlus Smartcards
- Rainbow iKey 1000/2000 (USB "Smartcard" device)
- Eracom CSA800

IBM HTTP Server Version 1.3.19 supports the following cryptographic devices. [This information is provided for convenience. Consult the IBM HTTP Server Web site and documentation as the ultimate authority].

Cryptographic devices	Client or server	Interface	Operating system
Rainbow Cryptoswift	Client or server	BSAFE 3.0	Windows NT, Solaris, HP-UX
nCipher nFast	Client or server	BHAPI plugin under under BSAFE 4.0	Windows NT, Solaris
nCipher nForce accelerator mode	Client or server	BHAPI/BSAFE	Windows NT, Solaris
nCipher nForce - key storage mode	Client or server	PKCS11	Windows NT, Solaris, HP-UX, AIX, Linux
IBM4758	Client or server	PKCS11	Windows NT, AIX



Be sure to check the [WebSphere Application Server prerequisites Web site](#) for the currently supported version(s) of IBM HTTP Server.

## **6.6.19: Administering the product messages, logs, and traces (overview)**

Messages, logs, and traces provide information about the execution of IBM WebSphere Application Server components, such as the administrative server and clients, application servers, and other WebSphere processes in the environment.



## 6.6.19.0: Properties for tracing, logging, and messages

Please select a sub-topic to access settings pertaining to:

- Application server trace

## 6.6.19.0.3: Server trace properties

These are settings for tracing the internal components of the application server.

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Application Server

The application server whose internal components you are tracing

### Dynamic Properties

Name-value pairs for configuring additional properties beyond those available in the administrative interface

### Enable Trace File Overwrite

When a named file is specified for trace output, and overwrite is enabled, the named trace file will be truncated each time the application server starts. When overwrite is not enabled, trace data is appended instead to the trace file.

Click "View..." to view the contents of the trace file. If the server is not running, then "View" will display the trace file from the last time the server ran.

### Enable Trace String

Whether the specified trace string should be put into use

### Node

The administrative node on which you are performing tracing

### Trace File


The file to which to log trace events. Note that serious events are always logged to the standard output stream and to the specified trace file.

*For Advanced Edition:* On an administrative server, the property is called traceOutput.

Specify one of these values:

- ☐ stderr
- ☐ stdout
- ☐ *filename* (or Specify)

where *filename* is a file of your choice. If you do not specify the full path to the file (drive letter and directories), the file will be created in the working directory of the application server, as specified by the properties of the application server.

 For an application server, the literals `stderr` and `stdout` refer to files named `stderr.txt` and `stdout.txt` (respectively). For an administrative server (as in the case of Advanced Edition), they actually refer to the standard error and standard out streams, such as an error file or the screen.

It is recommended you trace to `stderr` or `stdout` instead of to a file. Tracing to `stderr` or `stdout` is more efficient than tracing to a specified file.

## Trace Host Name

The host name to be used for trace access. It is the host name of the machine on which the application server is running.

## Trace Output File

See Trace File

## Trace Port

The port provided by the application server for trace access.

*For Advanced Single Server Edition:* You might need to change the default value if the port is already in use by another application. Specify -1 to use automatic port selection. However, a value of -1 prevents you from performing the runtime tracing tasks, such as sending trace strings to a running server.

## Trace Specification or Trace String

Specifies the application server components to trace. On an administrative server (in the case of Advanced Edition), the property is called traceString.

For *Advanced Edition* (non-Single Server), when the application server is running, you can click the ellipses button ("...") located next to the **Trace Specification** text field to open a graphical tree interface for specifying trace.

For *Advanced Single Server Edition*, when the application server is running, you can use a graphical tree viewerto select components to trace. If the server is not running, then you will needto type a trace specification string into a text field, using the correct format.

The valid format is:

`<comp1>=<type>=[enabled|disabled],...:<comp2>=<type>=[enabled|disabled],...`

where:

### comp

Specifies the component name. The component can be a class, package, or group of classesor packages.

Specify the full name of the component, or use a wildcard ("\*") character. When you terminate a component name with a wild card character, the enable/disableaction applies to all components whose names begin with the specified prefix.

For example, if components named "a.b.c.d" and "a.b.c.e" are registered, then specify "a.b.c.d" to trace only the "a.b.c.d" component. Specify "a.b.c.\*" to trace both components.

### type

Specifies the type of tracing to perform:

- **debug** - Provides information for debugging purposes.
- **entry/exit** - Indicates that a process has entered or exited a method.
- **event** - Indicates that a significant event took place, such as a state change.
- **all** - Conducts all three types of tracing.

### enabled|disabled

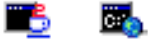
Specifies whether tracing is enabled or disabled for the component. Type either "enabled" or "disabled."

Here is an example of a valid trace specification or traceString. The statement:

```
com.ibm.ejs.container.*=all=enabled:com.ibm.ejs.jts.*=entry=enabled
```

enables all tracing on the container and entry/exit tracing on the JTS.

### Trace String



See Trace Specification

### View Log File



To view a log file, select the log file, determine which region of the file to view, and click "Refresh." The names of particular log files are as configured elsewhere in the server configuration. Note that it might be extremely slow, or impossible, to load the entire contents of a very large log file.

See also the other [application server properties](#).

## 6.6.19.3: Administering server traces with the Web console

Use the Web console to trace components of the application server runtime. You can learn more about the tracing tool, DrAdmin, in the problem determination section of the IBM WebSphere Application Server documentation (see the InfoCenter).

(If you are looking for information about tracing and debugging *applications* being hosted by the application server, see these topics instead: [instructions for starting application servers](#) and [configuring Distributed Debugger and OLT support](#)).

There are two basic scenarios for using the trace settings:

- **Configuration changes.** Like the majority of settings available in the Web console, you can change trace settings and save them to the server configuration file. They will take effect the next time you start the server.
- **Runtime tasks.** You can perform tasks against the trace service that is part of a running application server. For example, you can initiate a trace of specific components by sending a trace string to the running application server. These tasks yield immediate results. Saving the changes to the server configuration file so that they take effect the next time the server is started is optional.

Work with the trace service by locating it in the tree on the left side of the console:

**Nodes -> *hostname* -> Application Servers -> Trace Service**

There is also a related set of properties for defining Standard Error and Standard Out streams:

**Nodes -> *hostname* -> Application Servers -> IO Redirect**

## 6.6.19.3.1: Enabling, specifying, and editing trace strings with the Web console

During this task, you will specify or edit a trace string and enable tracing. These two actions are prerequisites to *starting* a trace. The trace string is a description of the application server component or components that you want to trace.

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Trace Service**. The trace service properties will be displayed on the right side of the console.
2. Specify a [trace string](#).

If the server is not running, then you must type a trace string with correct syntax into the trace specification field. Refer to the trace service field help for guidance.

If the application server is running, you can click the **Edit** button to display a tree view from which you can construct a trace string without knowing the trace syntax.

- Each node in the tree is a component that you can trace. The nodes are arranged in a hierarchy showing their relationships.
- To the right of the tree are the settings to disable, enable, or inherit tracing for the component that is currently selected in the tree.
- To begin with, each component in the tree is set for tracing according to the current trace string. If a component is not specified in the current trace string, then its tracing is off (not enabled).

Configure tracing for various components in the tree, then save the entire tree:

1. Modify a component in the tree of components. Use the settings on the right side of the work area to enable or disable tracing of the component, or have it inherit the setting from its parent node in the tree.
2. When you are finished specifying the settings for the selected component, click the **OK** button located directly below the settings.
3. Repeat the previous steps for as many components as you would like.
4. When finished, click the *other* **OK** button, located directly below the tree. This will save all of your changes to the tree, and exit the tree dialog.

The trace specification field will now contain the trace string corresponding to the way that you configured the various components in the tree view.

3. Click **Enable Trace String** so that the check box is checked. Otherwise, your trace string will be ignored.
4. Click **OK**.
5. At this point, you are ready to start tracing.
  - If the server is already running, you can send the trace settings to the server, to take effect immediately. For instructions, see the "Starting" link in the "Tasks..." links on the right.
  - If the server is not running, and you want to start tracing:
    1. [Save your configuration](#).
    2. To have the configuration take effect:
      1. [Stop the server](#)
      2. [Start the server again](#).

## 6.6.19.3.2: Starting (sending) server traces with the Web console

During this runtime task, you will send a trace string to the running application server. The trace string will take effect immediately. This task assumes that you have already enabled tracing and configured a trace specification.

To start tracing:

1. Ensure that the application server is running.
2. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Trace Service**. The trace service properties will be displayed on the right side of the console.
3. Verify that tracing is enabled.
4. Verify that you have specified the trace string that you want.
5. Click the **Send** button located below the trace string entry field.
6. **IF** you want to save the trace string to your server configuration:
  1. [Save your configuration.](#)
  2. (Optional) To have the configuration take effect:
    1. [Stop the server](#)
    2. [Start the server again.](#)

## 6.6.19.3.3: Disabling server traces with the Web console

During this task, you will turn tracing off, which will deactivate the trace specification while preserving the trace string contained in the trace specification field.

The server receives two pieces of information that instruct it how to trace --the trace string specification and the trace enablement flag. If tracing *is not* enabled, the server ignores the trace string and instead disables tracing for all components.

To disable tracing:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Trace Service**. The trace service properties will be displayed on the right side of the console.
2. Click **Enable Trace String** so that the check box is **not** checked.
3. Click **OK**.
4. If the server is already running, click **Send** to have the setting take effect immediately.

Otherwise, if the server is not running:

1. [Save your configuration.](#)
2. To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)



## 6.6.19.3.4: Retrieving trace strings from the server with the Web console

During this runtime task, you will retrieve the trace string that the running application server is using right now. This is useful when you are not sure which trace string is in use, or would like a copy of it in order to modify it and reapply it.

1. Ensure that the application server is running.
2. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Trace Service**. The trace service properties will be displayed on the right side of the console.
3. Click the **Retrieve** button. The trace specification field will be filled with the trace string under which the server is operating.

## 6.6.19.3.5: Specifying server trace logs with the Web console

During this task, you will specify where to send tracing.

1. Ensure that the application server is stopped.
2. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Trace Service**. The trace service properties will be displayed on the right side of the console.
3. In the trace service properties, specify whether to send tracing output to Standard Error (stderr), Standard Out (stdout), or to a file. (The stderr and stdout paths are those that were defined using the **IO Redirect** settings).
4. Click **OK**.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.19.3.6: Viewing trace logs with the Web console

During this task, you will view the trace log.

1. Ensure that the application server is running if you would like to view its current trace information. Otherwise, if the server is not running, you will view the traces that were generated the last time the server ran.
2. In the tree on the left side of the console, select one of two ways to display the dialog for viewing trace logs:
  - Click **Nodes** -> *hostname* -> **Application Servers** -> **Trace Service**. The trace service properties will be displayed on the right side of the console. Click the **View** button on the property sheet.
  - Click **Nodes** -> *hostname* -> **Application Servers** -> **IO Redirect**. Click the **View** button on the property sheet.
3. In the resulting panel, specify which log or logs to view (stderr, stdout, a trace file, and so on). Click **Select**.
4. Select which region of the log to view. Click **Refresh**. You might have to wait for a minute or less.
5. When the log loads into the viewing area, browse the log.

You can use the region selection settings and **Refresh** button to specify to view a different region of the log.

The settings for viewing traces persist for one administrative session. They cannot be saved to the server configuration file.

## 6.6.19.3.7: Specifying the trace host name and port with the Web console

During this task, you will specify the port of the application server that you want to trace, and the host name of the machine on which the application server resides.

Avoid changing these settings. However, sometimes you must do so. If you change the application server port value (suppose the port is already in use), then you must change the trace port value to match the application server port value. See the field help for more details.

1. Ensure that the application server is stopped.
2. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Trace Service**. The trace service properties will be displayed on the right side of the console.
3. Specify the values of the trace host name and port.
4. Click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.20: Administering transactions (overview)

Administrators do not usually need to intervene in server transactions. In fact, an administrator can introduce inconsistencies into server data by forcing a transaction to the wrong outcome.

Usually, it is sufficient simply to monitor the progress of transactions. The administrator should become familiar with transaction states and identifiers.

### Circumstances that could require intervention

Timeouts associated with transactions usually prevent any one transaction from holding resources at a server for too long.

For example, if two transactions are competing for the same resource (one holds a lock on a resource and the other is requesting that lock, and the lock modes conflict), timeouts will eventually abort one of the transactions: the idle timeout will abort a transaction that is inactive too long, and the operation timeout will abort an active transaction that is taking too long.

Nevertheless, a transaction can hang indefinitely if, for example, the transaction is prepared but the coordinator is unreachable.

If a hung transaction is interfering with the operation of a server (perhaps holding locks that other critical transactions are waiting for), the administrator might need to intervene.

An unprepared transaction can be aborted at any time. Unprepared transaction states include active and preparing.

Aborting an unprepared transaction does not affect the consistency of data -- the transaction's participants have not yet notified a coordinator of their readiness to commit. If the administrator aborts an unprepared transaction, any work that was completed on behalf of the transaction is rolled back.

With transactions that have already prepared, any action that the administrator takes is more significant. It is best to allow the system to resolve the transaction (allow normal processing to take place). However, the administrator will need to force an outcome if a transaction can never be completed otherwise.

## 6.6.20.0: Transaction properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

See also the other [application server properties](#).

**Client Inactivity Timeout**



**or Transaction Inactivity Timeout**



The number of milliseconds a transaction can remain inactive before it is aborted.

**Dynamic Properties**



Name-value pairs for configuring additional properties beyond those available in the administrative interface

**Enable**



Enable the transaction service. If the service is not enabled, all other transaction settings will be ignored.

**Total Tran Lifetime Timeout**



**or Transaction Timeout**



The number of seconds to allow a transaction to proceed before aborting it because it is taking too much time.

**Transaction Inactivity Timeout**



See Client Inactivity Timeout

**Transaction Log File**



The log file for transaction service logging. If specifying multiple log file names, separate them with commas. For example, specify:

"W:\WebSphere\AppServer\logs\tran1.log,W:\WebSphere4\_0\AppServer\logs\tran2.log"

If this property is blank, the server will use in-memory logging. In a production environment, be sure to set a value because transaction rollback capability requires transaction logging.

**Transaction Timeout**



See Total Tran Lifetime Timeout

## 6.6.20.3: Administering the transaction service with the Web console

Use the Web console to enable and edit the configurations of the transaction logging service.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes -> *hostname* -> Application Servers  
-> Transaction Service**

Note, if transaction support is not *enabled* (using a setting in the transaction service configuration), then all other transaction related properties will be ignored.

## 6.6.20.3.1: Enabling the transaction service with the Web console

During this task, you will turn on transaction logging, which will activate any transaction settings you have specified.

To enable the transaction service:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Transaction Service**. The transaction service properties will be displayed on the right side of the console.
2. Verify that you have specified the settings that you want.
3. Click **Enable** so that the check box is checked.
4. Click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).



## 6.6.20.3.3: Disabling the transaction service with the Web console

During this task, you will turn the transaction service off, which will deactivate any transaction logging settings you have specified.

To disable the transaction service:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Application Servers** -> **Transaction Service**. The transaction service properties will be displayed on the right side of the console.
2. Click **Enable** so that the check box is **not** checked.
3. Click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.24: Administering application client modules (overview)

Administration application client modules consists of the following.

Use the Application Assembly Tool to:

1. Creating the module
2. Setting deployment descriptor properties
3. Generating code for deployment

Use the Application Client Resource Configuration Tool to set additional configuration properties.

### Classpath considerations

An important classpath-related setting to note is the Module Visibility. This [application server setting](#) impacts the portability of applications and standalone modules from other WebSphere Application Server versions and editions. If your existing module does not run as-is when you transfer it to Version 4.0, you might need to reassemble an existing module or change the module visibility setting.

See the information on setting classpaths for a full discussion of classpath considerations.

## 6.6.24.0: Application client module properties

These are set using the Application Assembly Tool. Refer to the ["Assembly properties for client modules" property reference](#) to set values for these properties.

## 6.6.24.0.a: Assembly properties for application client modules

### File name (Required, String)

Specifies the file name of the application client module, relative to the top level of the EAR file. If this is a stand-alone module, the filename is the full pathname of the archive.

### Alternative DD

Specifies the file name for an alternative deployment descriptor file to use instead of the original deployment descriptor file in the module's JAR file. This file is the post-assembly version of the deployment descriptor file. (The original deployment descriptor file can be edited to resolve dependencies and security information. Directing the use of the alternative deployment descriptor allows you to keep the original deployment descriptor file intact). The value of the Alternative DD property must be the full path name of the deployment descriptor file relative to the module's root directory. By convention, the file is in the ALT-INF directory. If this property is not specified, the deployment descriptor file is read directly from the module's JAR file.

### Classpath

Specifies the full class path containing the dependent code that is not contained in the application client JAR file. Specify the values relative to the root of the EAR file and separate the values with spaces. Absolute values that reference files or directories on the hard drive are ignored. To specify classes that are not in JAR files but are in the root of the EAR file, use a period and forward slash (. /). Consider the following example directory structure in which the file myapp.ear contains an application client JAR file named client.jar. Additional classes reside in class1.jar and class2.zip. A class named xyz.class is not packaged in a JAR file but is in the root of the EAR file.

`myapp.ear/client.jar myapp.ear/class1.jar myapp.ear/class2.zip myapp.ear/xyz.class`

Specify `class1.jar class2.zip ./` as the value of the Classpath property. (Name only the directory for class files.)

### Display name (Required, String)

Specifies a short name that is intended to be displayed by GUIs.

### Small icon

Specifies a JPEG or GIF file containing a small image (16x16 pixels). The image is used as an icon to represent the application client in a GUI.

### Large icon

Specifies a JPEG or GIF file containing a large image (32x32 pixels). The image is used as an icon to represent the application client in a GUI.

### Description

Contains text describing the application client.

### Main class (Required, String)

Specifies the full path name of the main class for this application client.

## 6.6.24.5: Administering application clients with Application Assembly Tool

An application client is a standalone Java program (in contrast to a Webbrowser-based program). An application client module is used to assemble the files that make up the application client into a single unit. The Application Assembly Tool is used to create and edit modules, verify the archives, and generate deployment code. See the related topics for links to concepts, instructions for creating an application client module, and field help.

If the application client requires local resources, you must also use the Application Client Resource Configuration Tool. This tool allows you to define references to local resources other than enterprise beans (such as JDBC and JMS resources) on the machine where the application client resides.

## 6.6.24.5.1: Creating an application client

Application client modules can be created by using property dialog boxes or by using a wizard.

- [Using the property dialog boxes](#)
  - [Using the Create Application Client wizard](#)
- 

### Using the property dialog boxes

Creating a new application client consists of specifying the files that make up the client and then adding assembly properties. To create a new application client:

1. Click **File -> New -> Application Client**. The navigation pane displays a hierarchical structure used to build the contents of the module. The icons represent the components, assembly properties, and files for the module. A property dialog box containing general information about the application client is displayed in the property pane.
2. By default, the application client JAR file and the display name are the same. It is recommended that you change the display name in the property pane.
3. By default, the temporary location of the application client JAR file is `installation_directory/bin`. You must specify a new file name and location by clicking **File->Save**.
4. Enter the main class file name (required). Click **Browse** to locate the class file. By default, the root directory or archive is the current archive. If needed, browse the file system for the directory or archive where class files reside. Click **Select**. The archive's file structure is displayed in the window. Expand the structure and locate the files that you need. Select the file and click **OK**.
5. Enter values for other properties as needed. View the help for [6.6.24.0.a: Assembly properties for application client modules](#). Click **OK**.
6. Define assembly properties for the application client.
  - Right-click the EJB References icon and click **New**. A property dialog box for EJB References is displayed. View the help for [6.6.43.0.1: Assembly properties for EJB references](#). Enter values for each property and then click **OK**. Repeat to specify multiple EJB references. The top portion of the property pane lists each reference. Select the reference to view its corresponding property dialog box.
  - Right-click the Resource References and click **New**. A property dialog box for Resource References is displayed. View the help for [6.6.43.0.2 Assembly properties for resource references](#). Enter values for each property and then click **OK**. Repeat to specify multiple resource references.
  - Right-click the Environment Entries and click **New**. A property dialog box for Environment Entries is displayed. View the help for [6.6.34.0.a: Assembly properties for environment entries](#). Enter values for each property and then click **OK**. Repeat to define multiple environment variables.
7. Add files for the application client. In the navigation pane, right-click the Files icon and then choose **Add Files**. Use the file browser to locate files to add. First, browse for the root directory or archive where the files are located and click **Select**. If you are adding an entire archive, select the directory that contains the archive. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. Relative path names are maintained. The selected files are displayed in the Selected Files window. Click **OK**. The file names, extensions, modification dates, sizes, and path names are displayed in the property pane.

8. Review the contents of the module and make any desired changes.
  9. Click **File->Save** to save the module.
- 

## Using the Create Application Client wizard

Use this wizard to create an application client JAR file. During creation, you specify the files that make up the application client. You also specify information such as references to other (external) enterprise beans needed by the client.

Before you start the wizard, you must have the class files and other files belonging to the application client. When the wizard is completed, your application client JAR file resides in the location that you specified.

To create an application client, click the **Wizards** icon on the toolbar, and then click **Application Client**. Follow the instructions on each panel.

- [Specifying application client module properties](#)
- [Adding files](#)
- [Specifying additional application client module properties](#)
- [Choosing application client module icons](#)
- [Adding EJB references](#)
- [Adding resource references](#)
- [Adding environment entries](#)
- [Setting additional properties and saving the archive](#)

## Specifying application client module properties

On the **Specifying Application Client Module Properties** panel:

1. Indicate the application to which this module is to be added. If a parent application is not indicated, the module is created as a stand-alone application.
2. Specify a display name for the application client (required). The display name is used to identify your application in the Application Assembly Tool and can be used by other tools.
3. Specify a file name for the application client (required). The filename specifies a location on your system for the JAR file to be created.
4. Provide a short description of the application client (optional).
5. Click **Next**.

## Adding files

On the **Adding Files** panel, specify the files that are to be assembled for the application client. To add or remove files:

1. Click **Add**. Use the file browser to locate files to add. First, browse for the root directory or archive where the files are located and click **Select**. If you are adding an entire archive, select the directory that contains the archive. The directory structure is displayed in the left pane. Browse the directory structure. From the right pane, select one or more files to be added and click **Add**. If you select a directory and click **Add**, all files in the directory, including the directory, are added. Relative path names are maintained. The selected files are displayed in the Selected Files window. Click **OK**. The files are displayed in a table on the wizard panel.

2. If you want to remove a file, select the file in the table and then click **Remove**.
3. Continue to add or remove files until you have the correct set of files.
4. Click **Next**.

## Specifying additional application client module properties

On the **Specifying Additional Application Client Module Properties** panel:

1. Specify the classpath and main class for the application client. View the help for [6.6.24.0.a: Assembly properties for application client modules](#).
2. Click **Next**.

## Choosing application client module icons

On the **Choosing Application Client Module Icons** panel, specify icons for your module.

1. Specify the full path name of a file containing a small icon, or click **Browse** to locate and select the file. The icon must be a GIF or JPEG image 16x16 pixels in size.
2. Specify a full path name of a file containing a large icon, or click **Browse** to locate and select the file. The icon must be a GIF or JPEG image 32x32 pixels in size.
3. Click **Next**.

## Adding EJB references

On the **Adding EJB References** panel, specify the enterprise beans required by the application client.

1. Click **Add**. Enter the name of the enterprise bean to be used by the application client, the names of the bean's home and remote interfaces, and the bean type (required). View the help for [6.6.43.0.1: Assembly properties for EJB references](#). The EJB reference is displayed in a table on the wizard panel.
2. If the referenced bean is located in the module being created (or in the encompassing application), enter a name in the Link field (optional). If the bean is external to the module, leave the Link field blank. You can specify JNDI binding information later (by using the property dialog boxes or by using the administrative console).
3. To remove a reference, select the entry in the table and then click **Remove**.
4. Continue to add and remove references as needed.
5. Click **Next**.

## Adding resource references

On the **Adding Resource References** panel, enter references to connection factory objects for resource managers.

- Click **Add**. Enter values for the name, type, and authorization mode of the resource reference, then click **OK**. View the help for [6.6.43.0.2 Assembly properties for resource references](#). The resource reference information is displayed in a table.
- To remove a reference, select the reference in the table and click **Remove**.
- Continue adding or removing references as needed.
- Click **Next**.

## Adding environment entries



On the **Adding Environment Entries** panel, add environment entries for the application client.

- Click **Add**. Enter the name and type of the entry, then click **OK**. View the help for [6.6.34.0.a: Assembly properties for environment entries](#).
- To remove an environment entry, select the entry in the table and then click **Remove**.
- Continue adding or removing environment entries as needed.
- Click **Next**.

## Setting additional properties and saving the archive

Click **Finish** to complete the wizard. To change settings for properties, click **Back** to return to the appropriate panel. Make any needed changes, and then click **Finish**.

After you click **Finish**, the contents of the archive are displayed in the main window. You can continue adding or modifying properties as needed. For example, you can add binding information. When you are finished editing the archive, click **File->Save**. Specify a name for the archive and click **Save**.

## 6.6.25: Administering resource providers (overview)

The administrative console provides settings for administering several J2EE resource provider types:

- [JDBC providers](#) - provide data access
- [JavaMail sessions](#) - provide JavaMail session support
- [URL providers](#) - provide URL support
- [JMS providers](#) - provide Java messaging support

In addition to configuring instances of each provider type, you can configure some settings that apply to all of the above-listed provider types.

## 6.6.25.0: J2EE resource provider properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Description



An optional description of the resource provider

### Name



The required display name for the resource provider

### Property Set



Name-value pairs for configuring additional properties beyond those available in the interface

## 6.6.25.3: Administering resource providers with the Web console

Use the Web console to configure resource providers. See also the topics for Administering particular types of resource providers, such as JDBC Providers.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes -> Resources**

## 6.6.26: Administering application server process definitions (overview)

The process definition is an *Advanced Single Server Edition* concept that encompasses the runtime attributes of the application server process, such the working directory, startup attempts, file permissions, and process priority on the operating system. It is a sub-component of the application server.

In *Advanced Edition* (non-Single Server), the "process definition" is not part of the formal model, but the application server has many of the same properties. For this reason, the property reference for process definitions is shared among *Advanced Edition* and *Advanced Single Server Edition*.

For the many other application server properties, see the [application server property reference](#).

## 6.6.26.0: Process definition properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

See also the other [application server properties](#).



*Important note for Advanced Single Server Edition users*

### Environment



Name-value pairs for configuring environment variable values for the running process

### Executable Name



The executable name of the process

### File Permissions



Read, Write, and Execute permissions for the Owner, User, and Group security roles

### Group ID



The name of the operating system user under which to run the server

Note that the operating system user must exist on the machine where the server is to run before the server is started. This user must be assigned the necessary operating system privileges for performing operations such as creating output files on the local file system.

### JVM Settings



[Java virtual machine configuration settings](#) for the application server's process

### Maximum Startup Attempts



The maximum number of times to attempt to start the application server before giving up

### Ping Initial Timeout



The maximum time in seconds for the server to finish initialization. After this time elapses, the administrative server attempts to restart the server.

### Ping Interval



The frequency of communication attempts between the parent process, such as a process manager, and the process it is trying to spawn. Adjust this value based on your requirements for restarting failed servers. Decreasing the value detects failures sooner; increasing the value reduces the frequency of pings, reducing system overhead.

### Ping Timeout



When a parent process is spawning a child process, such as when a process manager spawns a server, the parent process pings the child process to see whether the child was spawned successfully. This value specifies the number of milliseconds that the parent process should wait (after pinging the child process) before assuming that the child process failed.

### **Process Priority**

Operating system priority for the process. Only root users can change this value.

### **Run As User**

Runs the process as a specific user

### **Run As Group**

Runs the process as a member of a specific group

### **Standard Input**

The file from which the standard input stream will be read

### **Standard Output** **or Stdout File Name**

The file to which the standard output stream will be directed. The file name can include a symbolic path name defined in the path map entries.

*For the Advanced Single Server Edition:* Click "View..." to view the contents of the stdout file.

### **Standard Error** **or Stderr File Name**

The file to which the standard error stream will be directed. The file name can include a symbolic path name defined in the path map entries.

*For the Advanced Single Server Edition:* Click "View..." to view the contents of the stderr file.

### **Stderr File Name**

See Standard Error

### **Stdout File Name**

See Standard Output

### **Umask**

The user mask that the process runs under (the file-mode permission mask)

### **User ID**

The name of the operating system user under which to run the server

Note that the operating system user must exist on the machine where the server is to run before the server is started. This user must be assigned the necessary operating system privileges for performing operations such as creating output files on the local file system.

### **Working directory**

The local directory in which to run the server

This directory is used to determine the locations of input and output files with relative path names.

After you start a server, it is recommended that you do not change the server's working directory.

Passivated enterprise beans are placed in the current working directory of the application server on which the beans are running. Make sure the working directory is a known directory under the root directory of the Application Server product.



## 6.6.26.3: Administering application server process definitions with the Web console

Use the Web console to configure the application server process definition, including the stdout and stderr logs to be used, the JVM settings, classpath, and the identity under which the server will run.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Process Definition**

Expand the tree further to see the categories of process definition settings.

## 6.6.26.3.4: Updating process definition configurations with the Web console

During this task, you will update the application server process definition.

1. In the tree on the left side of the console, click

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Process Definition**. Its properties will be displayed on the right side of the console.

2. Modify the process definition properties.
3. When finished, click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  - [Stop the server](#)
  - [Start the server again](#).

## 6.6.27: Administering path maps (overview)

Please see the related information links.

## 6.6.27.0: Path map properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Description



An optional description for your administrative records

### Path



The absolute path that the symbolic name represents

For example, the absolute path "C:\WebSphere\AppServer\Config" on a Windows system could be represented by the symbolic name "CONFIG\_ROOT."

### Symbolic Name



The symbolic name representing a physical path or URL root

This name will be substituted into path names configured in the system. For example, "CONFIG\_ROOT" could be the symbolic name representing the path "C:\WebSphere\AppServer\Config" on a Windows system.

## 6.6.27.3: Administering path maps with the Web console

Use the Web console to create and edit entries in the path map, which specifies symbolic names for physical filesystem locations.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Path Map** -> **Entries**

## 6.6.27.3.1: Configuring new path maps with the Web console

During this task, you will specify a new path map configuration that defines a symbolic name for a file system root.

To configure a new path map:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Path Maps** -> **Entries**. A list of existing path maps will be displayed on the right side of the console.
2. Click the **New** button displayed above the list.
3. Specify the new path map settings.
4. Click **OK**.
5. Verify that the new path map is displayed in the list of path maps.
6. [Save your configuration.](#)
7. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)

## 6.6.27.3.3: Removing path maps with the Web console

During this task, you will remove a path map configuration that defines a symbolic name for a file system root.

To remove a path map:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Path Maps** -> **Entries**. A list of existing path maps will be displayed on the right side of the console.
2. Locate one or more path map entries that you want to delete. Ensure that the checkbox to the left of each entry is selected.

Also ensure that no check boxes are selected for path maps you do **not** want to delete.

3. Click the **Delete** button displayed above the list.
4. Click **OK**.
5. Verify that the deleted path map is removed from the list of path maps.
6. [Save your configuration](#).
7. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.27.3.4: Updating path maps with the Web console

During this task, you will modify a path map configuration that defines a symbolic name for a file system root.

To modify a path map:

1. In the tree on the left side of the console, click **Nodes** -> *hostname* -> **Path Maps** -> **Entries**. A list of existing path maps will be displayed on the right side of the console.
2. Locate and click the path map that you want to update.
3. Modify the path map properties.
4. Click **OK**.
5. [Save your configuration.](#)
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again.](#)



## 6.6.28: Administering Object Level Trace and Distributed Debugger support (overview)

Complimentary copies of IBM Object Level Trace and IBM Distributed Debugger software are provided for tracing and debugging the applications running on the application server.

See the related topics for instructions for enabling and configuring OLT and Debugger support through the WebSphere Application Server administrative console.

See also the following OLT and Debugger documentation, included with the InfoCenter:

- [IBM Object Level Trace documentation](#)
- [IBM Distributed Debugger documentation](#)

## 6.6.28.0: Object Level Trace and Distributed Debugger properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Dynamic Properties



Name-value pairs for configuring properties beyond those available in the administrative interface

### Enable IBM Distributed Debugger



Whether to enable the IBM Distributed Debugger support. If you do not, all other settings related to the Distributed Debugger will be ignored.

Enabling IBM Distributed Debugger makes the execution of your application enabled for the debugger. With this selection, you can debug your application in standalone mode, or through Object Level Trace.

This is a necessary step for using the Object Level Trace to trace code running on the application server. Note, you must also enable Object Level Trace.

This field and the **Enable Debug Mode** field in [JVM properties](#) cannot be enabled at the same time.

See the InfoCenter documentation for more information about using OLT and Debugger.

### Enable Object Level Trace



### or Object Level Trace Enabled



Whether to enable support for IBM Object Level Trace. If you enable OLT, also enable the Distributed Debugger.

This is a necessary step for using the Object Level Trace to trace code running on the application server. You use the [JVM Settings properties](#) to configure debugging support. See the InfoCenter documentation for more information about using OLT and Debugger.

### Host Name



### or OLT Server Host



The fully qualified host name of the machine on which the OLT trace viewer is located. Trace results will be directed to the machine.

### Object Level Trace Enabled



See Enable Object Level Trace

### OLT Server Host



See Host Name

### OLT Server Port



### or Port



The port number corresponding to the OLT Host Name. The default is 2102.

**Port** 

See OLT Server Port

**Source Path** 

The root directory from which to obtain the source code for debugging purposes. If the directory is already set in your CLASSPATH environment variable, then you do not need to set it here.

## 6.6.28.3: Administering IBM Object Level Trace and Distributed Debugger with the Web console

Use the Web console to manage tracing and step-through debugging of the applications running in your application server runtime. These configurations specify settings for using IBM Object Level Trace (OLT) and IBM Distributed Debugger (DD).

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **IBM Debug and OLT**

Follow this procedure to enable and configure OLT and Debugger support in the server runtime:

1. [Open the OLT properties dialog.](#)
2. Use the [OLT properties](#) to:
  - Enable the IBM Distributed Debugger.
  - Specify the "source path" if you know that the source files of your application reside in a different directory than your class files. The source path is the list of directories that contain the source files of your application.
  - Enable Object Level Tracing.
  - Provide the hostname of the machine when the OLT and debugger user interface will be running.
  - Set the OLT server port to the default value of 2102.
3. Save the changes.
4. [Start the application server.](#)

## 6.6.28.3.4: Updating IBM Object Level Trace and Distributed Debugger configurations with the Web console

During this task, you will update an existing Object Level Trace and Distributed Debugger configuration.

To update a configuration:

1. Click **Nodes** -> *hostname* -> **Application Servers** -> *application\_server\_name* where *application\_server\_name* is the name of the existing application server.
2. Click **IBM Debug and OLT**. Its properties will be displayed on the rightside of the console.
3. Modify the properties.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## **6.6.29: Administering Location Service Daemons (overview)**

Please see the related information links.

## 6.6.29.0: Location Service Daemon properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

See also the other [application server properties](#).

### Dynamic Properties



Name-value pairs for configuring properties beyond those available in the administrative interface

### Enable



Whether the location service daemon is enabled

### Host Name



The host name or IP address of the location service daemon

### Port



The port number for the location service daemon

## 6.6.29.3: Administering Location Service Dameons with the Web console

Use the Web console to configure the Location Service Daemon (LSD) that the application server uses to route client requests for applications and their components.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Location Service Daemon**



## 6.6.29.3.4: Updating Location Server Daemon configurations with the Web console

During this task, you will update the configuration of the Location Server Daemon.

1. In the tree on the left side of the console, click

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Location Service Daemon**. Its properties will be displayed on the right side of the console.

2. Modify the daemon properties.
3. When finished, click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.30: Administering Object Request Brokers (overview)

The WebSphere Application Server uses the ORB to manage communication between client applications and server applications, as well as, communication between WebSphere components, such as the application server . During WebSphere Application Server installation, default values are established, called ORB properties, which are used when WebSphere is started and the ORB is initialized. These properties control the run-time behavior of the ORB and can also affect the behavior of WebSphere components which are tightly integrated with the ORB, such as security.

It might be necessary to modify some of the ORB properties under certain circumstances. For example, it may be necessary to modify the default timeout delay the ORB uses when waiting for a response to a request, or to modify the maximum number of active connections the ORB caches for better performance, and so on.

In every request/response exchange, there is a client-side ORB and a server-side ORB, depending on which side initiated the exchange. It is important that the ORB properties be set for either the client-side or server-side as necessary because each ORB is, in general, independent of the other.

### Do not use multiple ORB instances

IBM WebSphere Application Server security does not support the programming model of using multiple ORB instances.

To obtain its ORB instance, every application should always make calls to:

```
com.ibm.ejs.oa.EJSORB.getORBinstance()
```

After the ORB instance has been established with a process, the process should not change ORB-related properties because property changes will trigger a new ORB instance to be created. A multiple ORB scenario will occur, which is not supported.

### Finding information about the Request Interceptor interface

For information about the Request Interceptor Interface of the IBM Java ORB, see the IBM Web site:

[http://www.ibm.com/software/webservers/appserv/request\\_interceptors.html](http://www.ibm.com/software/webservers/appserv/request_interceptors.html)

## 6.6.30.0s: Object Request Broker and ORB thread pool properties for Advanced Single Server Edition

### Bootstrap Host

The DNS hostname or IP address of the machine on which initial server contact for this client resides.

### Bootstrap Port

The port to which the ORB will connect for bootstrapping. In other words, the port of the machine on which the initial server contact for this client is listening. The default is 900.

### Can Be Grown

Whether to allow the number of threads to increase beyond the maximum size configured for the thread pool, if there is a demand for more threads

### Dynamic Properties

Use dynamic properties to configure additional name-value pairs beyond the settings available in the administrative interface. The descriptions below include both name-value pairs and settings available from fields in particular interfaces.

#### com.ibm.CORBA.ForceTunnel

Controls how the client ORB attempts to use HTTP tunneling. The values of this optional property can be:

- **always** - use HTTP tunneling immediately, without trying TCP connections first
- **never** - disable HTTP tunneling. If a TCP connection fails, a CORBA system exception COMM\_FAILURE is thrown
- **whenrequired** - use HTTP tunneling if TCP connections fail

#### com.ibm.CORBA.ListenerPort

The port on which this server will listen for incoming requests. The default is the next available system assigned port number. The range for this optional property is from 0 to 2147483647 (maximum Java Int).

#### com.ibm.CORBA.requestRetriesCount

The number of times the ORB will attempt to send an indirect-IOR request in case a server failed. If a server has been determined to have failed, the ORB will give the Location Service Daemon an opportunity to issue a new IOR. The default value of this property is 1. The range is from 0 to 2147483647 (maximum Java Int).

#### com.ibm.CORBA.requestRetriesDelay

The amount of delay time (in milliseconds) between request retries. The default value of this optional property is 0. The range is from 0 to 2147483647 (maximum Java Int).

#### com.ibm.CORBA.requestTimeout

The number of seconds to wait before timing out on a request message. The default value of this optional property is 180. The range is from 0 to 2147483647 (maximum Java Int).

## **com.ibm.CORBA.ServerSocketQueueDepth=*value***

The property can be specified to change the queue depth of a server.

For an application server, the argument can be specified using the [command line arguments](#) of the application server JVM properties.

## **com.ibm.CORBA.TunnelAgentURL**



The URL of the servlet used to support HTTP tunneling. It must be a properly formed URL, such as "http://w3.mycorp.com:81/servlet/com.ibm.CORBA.services.IIOPTunnelServlet" or, for applets, "http://applethost:port/servlet/com.ibm.CORBA.services.IIOPTunnelServlet." There is no default value. This field is required if com.ibm.CORBA.ForceTunnel is set.

## **Enable**



Whether to enable the ORB within the application server. Use this setting to disable EJB and CORBA when running only Web applications.

## **Inactivity Timeout**



Socket timeout value for the port that the ORB is using. The default is 0, meaning no timeout (wait infinitely).

## **Maximum Size**



The maximum amount of connections in use kept in the connection cache table at any one time.

## **Minimum Size**



The number of connections that must be in the connectioncache in order for the ORB to clean up any connections that are not busy.

For example, if the default value is 100, then when the cache size reaches 101, the ORB will begin shutting any connections that are open but not busy. It will stop when there are 100 or fewer connections in the cache.

## **Secure Socket Layer settings**



See the [Secure Socket Layer property help](#)

## 6.6.30.3: Administering Object Request Brokers (ORBs) with the Web console

Use the Web console to configure the Object Request Broker (ORB) that the application server uses.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **ORB Settings**

Expand the node to view additional settings, including settings for configuring:

- Secure Socket Layer (SSL) settings for the client and server
- Thread pool settings to adjust the pool of threads used by the ORB for handling IIOP connections

## 6.6.30.5: Setting the ORB timeout value

If the network goes down when a Java application is attempting to contact a server, it can take too long for the ORB in the Java application to get an exception. This causes the Web browser originating the request to time out before the Java application can respond to the connection failure (network down). Setting the ORB timeout value to be less than the browser timeout value will allow the Java application to handle the exception before the browser times out. This provides the opportunity for a more graceful, user-friendly response to down servers.

The setting should only be used if you cannot set or adjust a timeout at the JDK level. At the time of this writing, the JDK does not have such a setting. Also, you should be fairly sure of the problem before attempting this solution. Note the considerations.

### Considerations for setting the ORB timeout

It is recommended that you do not adjust the ORB timeout unless experiencing a problem, because configuring a value that is inappropriate for the environment can create a problem. If you set the value, experimentation will be needed to find the correct value for the particular environment. Configuring an incorrect value can produce results worse than the original problem.

That said, if Web clients accessing Java applications running in the WebSphere environment are consistently experiencing problems with their requests, and the problem cannot be traced to other sources and addressed through other solutions, consider setting an ORB timeout value and adjusting it for your environment.

Web browsers vary in their language for indicating that they have timed out. Usually, the problem is announced as a "connection failure" or "no path to server" message.

### Tuning the ORB timeout

When tuning this parameter, aim to set the ORB timeout value to *less than* the time after which a Web client eventually times out. Because it can be difficult to tell how long Web clients will wait before timing out, setting the ORB timeout requires some experimentation. Another difficulty is that the ideal testing environment will feature some simulated network failures for testing the proposed setting value.

Empirical results from limited testing indicate that 30 seconds is a reasonable starting value. Mainly, you need to ensure that the setting is not too low. To fine tune the setting, find a value that is not too low. Then gradually decrease the setting until reaching the threshold at which the value becomes too low. Set the value a little above the threshold.

When the ORB timeout is set too low, the symptom is numerous CORBA 'NO\_RESPONSE' exceptions, which occur even for some requests that should have been valid. If requests that should have been successful (for example, the server is not down) are being lost or refused, then the value is likely too low.

### How to set the ORB timeout

To set the ORB timeout, add a command line argument to the Java application that the Web clients are trying to access. The Java application could be your own application, or a Java process internal to WebSphere Application Server, such as an application server (and its related servlet engine).

On the Java command line for the application, specify:

```
-Dcom.ibm.CORBA.requestTimeout=30
```

where 30 is a value that you think will be appropriate for the environment.

## 6.6.32: Administering name service support (overview)

### Domain Name Service (DNS) support

The application server can act as a Domain Name Service (DNS) client. In some cases, DNS access is required. For applications using enterprise beans, a name service is also provided.

### WebSphere Application Server clients and DNS resolution

By default, a WebSphere administrative server or application server uses the IP address of the local host on which it is running in its object references. Thus, clients of these servers do not usually need to have DNS enabled in order to access these object references, nor do they need to be on the same network as the server.

However, in cases where DNS is disabled on the client machine, additional steps must be performed for certain clients as follows:

- If the product edition you are using provides a Java-based administrative console (WebSphere Administrative Console), it must be run with the `-primaryNode` option:  
`-primaryNode <name of administrative server to which to connect>`
- If the product edition you are using supports clients that rely on the workload management features of WebSphere Application Server, the clients must be run with the system property `com.ibm.ejs.wlm.BootstrapNode`:  
`-Dcom.ibm.ejs.wlm.BootstrapNode=<name of administrative server to which to connect>`

By default, the administrative server name is the short name of the host on which it is running. As shown above, use an argument to specify the administrative server host name, which is required because the clients do name service lookups for names that are qualified by the administrative server name. Usually, if DNS is enabled, they can derive the administrative server name by doing a DNS reverse translation; however, if DNS is disabled, then they have to be explicitly provided with the administrative server name.

In some situations, it might be necessary to override the default value for the host information in the object references generated by WebSphere administrative or application servers. To do this, set the system property `com.ibm.CORBA.LocalHost` as follows:

```
-Dcom.ibm.CORBA.LocalHost=<value>
```

where `<value>` can be a host name (long or short) or an IP address. Set this property on a per server basis.

Some possible reasons for overriding this value are as follows:

- The host machine has multiple IP addresses.  
  
Given the default behavior, either address could be selected, possibly arbitrarily. The `LocalHost` property should be set to specify a single IP address to be used in all object references.
- You want to place the host name, rather than the IP address, in the object references.

This might be necessary for situations in which external clients exist outside a firewall, and these clients cannot access the internal network directly. In this case, the host name can be translated to a gateway machine, which can then translate and forward to the internal network using the real IP address.

### Java Remote Method Invocation (RMI) support

#### Applications can invoke RMI servers, but should not be RMI servers

WebSphere Application Server supports servlets that invoke Remote Method Invocation (RMI) servers, but the servlets are not permitted to be RMI servers themselves. They can be RMI clients. The same applies to enterprise beans. If this guideline is violated, a `java.rmi.RMISecurityException` will result.

#### Do not set security managers in RMI clients

Correct use of the Java RMI services requires that a security manager (for example, class `java.rmi.RMISecurityManager`) be set within the RMI server. Typically, a security manager will **not** be set in the RMI client program.

As such, a servlet acting as an RMI client should not set a security manager. The same applies to enterprise beans acting as RMI clients. Setting a security manager within a servlet (that is acting as an RMI client) is not only incorrect usage of RMI, but will cause problems for the server. The security manager will be global to the server and will affect server operations.

If one of your applications currently sets a security manager when it should not be doing so, stop the application server containing the application, remove the statements that set the securitymanager, and start the application server again.



## 6.6.34: Administering environment entries

Environment entries are defined during application assembly. Please see the related information topics.

## 6.6.34.0: Properties of environment entries

Environment entries are configured during application assembly. Refer to the ["Assembly properties for environment entries"](#).

## 6.6.34.0.aa: Assembly properties for environment entries

Environment entries define variables used to customize the business logic of an application at run-time. This eliminates the need to access or change the application's source code. The container makes the application's environment entries available in a JNDI naming context (`java:comp/env`). An example environment entry is `maxExceptions`, describing the maximum number of tax exemptions that are allowed. The environment entry's expected type is `java.lang.Integer` and its value is 15.

### **Name (Required, String)**

Specifies the name of the environment entry, relative to the `java:comp/env` context.

### **Value**

Contains the value of the environment entry. The value must be a string that is valid for the constructor of the specified type that takes a single string parameter.

### **Type (Required, String)**

Specifies the fully qualified Java type of the environment entry value that is expected by the module's code. The following are the legal values for this field: `java.lang.Boolean`, `java.lang.String`, `java.lang.Integer`, `java.lang.Double`, `java.lang.Byte`, `java.lang.Short`, `java.lang.Long`, and `java.lang.Float`.

### **Description**

Contains text describing the environment entries.

## 6.6.35: Administering generation of deployment code

The Application Assembly Tool contains settings for administering how deployment code will be generated. Please see the sub-topics for details.

## 6.6.35.0: Assembly properties for generating deployment code

### Deployed module location (Required, String)

Specifies the name and location where the deployed archive will be saved. By default, this is the directory where you saved the archive file or the directory from which you launched the Application Assembly Tool. The deployed archive file is given the same name as the undeployed archive, prefixed with `deployed_`.

### Working directory (Required, String)

Specifies the full path name of the working directory. This directory is used to save temporary files while creating the deployed archive. On UNIX systems, if there is insufficient space in the working directory, a segmentation violation error can occur. Make sure that there is sufficient space in the working directory. Required space varies depending on the application. By default, the working directory is `/tmp` on UNIX systems. You can specify a different working directory by using this field.

### Dependent classpath

Specifies the class path where dependent class files can be found, if needed by the module being deployed. List only additional JAR files referred to by the JAR file being deployed. On Windows NT systems, separate each path by a semicolon (;). On UNIX systems, separate each path by a colon (:). Use one entry for each directory or JAR file to be searched. Files without the extension `.jar` or `.zip` are ignored.

### Code generation only

Specifies that only stub and skeleton files are to be generated. The RMIC command and javac compiler are not to be run. The default is `false` (that is, all steps are executed).

### Verify archive

Specifies whether verification is to take place during deployment. The default is `false`. Verification checks the completeness of the archive and ensures that the classes in the EJB JAR files are in compliance with the EJB 1.1 specification. Note that verifying archive files is important for successful generation of deployment code for the application. If an archive file is not valid, code generation can fail.

### RMIC options

Specifies additional options to be passed to the RMIC command. These additional options are appended to the following options, which are always passed to the RMIC command and cannot be overridden: `-keep` (keeps intermediate generated files), `-iiop` (generates stubs for IIOP), `-ddirectory_name` (writes the files to the specified directory), `-sourcepath directory_name` (locates the source files in the specified directory), and `-nowrite` (compiled classes are not written to the file system).

### Database type

Specifies the database type, operating system, and version number. This information is used when generating the SQL code for creating database tables for entity beans with container-managed persistence (CMP).

### Database name

Specifies the name of the database. This information is used when generating the SQL code for creating database tables for entity beans with container-managed persistence (CMP). The database name is used in the generated DDL file.

### Schema name

Specifies the name of the database schema. This information is used when generating the SQL code for creating database tables for entity beans with container-managed persistence (CMP). The schema name is used in the generated DDL file and in the generated persistence code.

## **6.6.36: Administering Java Virtual Machine settings (overview)**

Please see the related information links.

## 6.6.36.0: JVM properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool



[Important note for Advanced Single Server Edition users](#)

### Boot Classpath



Some JVMs contain an option to specify bootstrap classes and resources.

*For the Advanced Single Server Edition:* This property can contain multiple paths separated by colons (":") or semicolons (";") depending on the operating system of the node.

*For the Advanced Edition:* Use the three fields for specifying:

- Total replacement of the default bootclasspath
- A string to *prepend* to the beginning of the bootclasspath
- A string to *append* to the end of the bootclasspath

### Classpaths



or Classpath



The standard classpath in which the Java virtual machine looks for classes.

*For the Advanced Single Server Edition:* Enter the classpath into a single text field, separating multiple paths by colons (":") or semicolons (";") depending on the operating system of the node.

*For the Advanced Edition:* Enter each classpath entry into a table row; you do not need to add the colon or semicolon at the end of each entry.

### Command line arguments



or Generic Command line arguments



The command line arguments to pass to the Java virtual machine (JVM) that starts the application server process.

You can find the most updated information about the command line arguments by entering at a command prompt `java` for the standard options and `java -X` for the non-standard options. In case that is not convenient, check this [summary list of the arguments](#).

### Debug Arguments



or Debug String



When Debug Mode for the Java virtual machine is enabled, this property allows additional debug arguments to be passed to the Java virtual machine.

### Debug Mode



or Enable Debug Mode



Enables the JVM debug output. When the property is set, the application server will start with the `java_g -debug` argument, which is necessary to allow the Distributed Debugger, or any Java debugger, to attach to the application server.

This field and the **Enable IBM Distributed Debugger** field cannot be enabled at the same time.

## Disable JIT



Disables the Just In Time (JIT) compiler option of the JVM

## Enable Class Verbose Mode or Verbose Mode

Enables verbose debugging output for class loading

## Enable Debug Mode

See Debug Mode

## Enable IBM Distributed Debugger

See the [OLT and Debugger property reference](#)

## Generic Command line arguments

See Command Line Arguments

## Enable Garbage Collection Verbose Mode or Verbose Mode Garbage Collection

Enables verbose debug output for garbage collection

## Enable JNI Verbose Mode or Verbose Mode JNI

Enables verbose debugging output for native method invocation

## Generated Command Line Arguments (read only)

Displays the commands that will be added to the Java command line, based on your configuration of the JVM settings (including the Advanced JVM Settings).

## Hprof Arguments

Additional profiler arguments to use when Run HProfs is used

## Initial Heap Size or Initial Java Heap Size

The initial heap size available to the JVM, in megabytes

## Initial Java Heap Size

See Initial Heap Size

## Maximum Heap Size or Maximum Java Heap Size

The maximum heap size available to the JVM, in megabytes

## Maximum Java Heap Size

See Maximum Heap Size

## Run HProfs

Enable HProf profiler support

*For the Advanced Edition:* Use the checkbox to enable or disable HProf options. Use the text field to



specify the options.

*For the Advanced Single Server Edition:* To use a different profiler, use the command line property to specify settings for the custom profiler.

### Source Path

See the [OLT and Debugger property reference](#)

### System Properties

A list of name-value pairs that can be passed into the virtual machine for initializing it. *-Dname=value* pairs are created from the values entered.

### Verbose Mode

See Enable Class Verbose Mode

### Verbose Mode Garbage Collection

See Enable Garbage Collection Verbose Mode

### Verbose Mode JNI

See Enable JNI Verbose Mode

See also the other [application server properties](#).

## 6.6.36.0.1: Java command line arguments reference

This section provides a reference of Java Virtual Machine (JVM) commandline arguments related to performance and debugging. The WebSphereadministrator can configure application servers and other managed Java processes to start with these parameters as command line arguments.

Additional command line arguments are valid, beyond those listed in this section. For a list of the command line arguments currently available on your operating system with your Java Development Kit (JDK) level, type `java -X` at a system command prompt.

### Command line arguments related to performance

Goal	Argument	Values	Notes
Specify the maximum heap size the Java interpreter will use for dynamically allocated objects and arrays	-Xmx	Specify the value in bytes, with a value greater than 1000	On AIX, the default is 32M
Specify how much memory is allocated for the heap when the JVM starts	-Xms	Specify the value in bytes, with a value greater than 1000	On AIX, the default is 1M
Specify the size of each thread Java code stack	-oss	Specify the value in bytes, with a value greater than 1000	On AIX, the default is 400K
Specify the size of each thread native code stack	-ss	Specify the value in bytes, with a value greater than 1000	On AIX, the default value is 256K

### Command line arguments related to debugging

Goal	Argument	Values	Notes
Disable the JIT compiler	-Djava.compile=none	None	Not available on AIX or Solaris
Specify whether to run the byte-code verifier on all loaded classes	-verify	<i>true false</i>	None
Verify classes read in over the network	-verifyremote	None	None
Avoid verifying any classes	-noverify	None	None
Specify whether to print a message whenever the garbage collector frees memory	-verbosegc	None	None
Prevent asynchronous garbage collection	-noasyncgc	None	None
Disable class garbage collection	-Xnoclassgc	None	None
Specify whether to print a message each time the JVM loads a class	-verbose	None	None

## 6.6.36.3: Administering JVM settings with the Web console

Use the Web console to configure the JVM settings that the application server uses.

Work with objects of this type by locating them in the tree on the left side of the console:

**Nodes** -> *hostname* -> **Application Servers**

-> *application\_server\_name* -> **Process Definition** -> **JVM Settings**

## 6.6.37: Administering mail providers and mail sessions (overview)

WebSphere Application Server's implementation of JavaMail does not provide mail servers. Even if your application components can communicate with mail servers, you still must configure separate IMAP and SMTP servers to enable the mail functions.

Only the SMTP and IMAP service providers are shipped with WebSphere Application Server. To use other protocols, install the appropriate service providers for those protocols.

When you configure your mail servers, use fully qualified internet host names.


For information on how to install a service provider, see Chapter 5, *The Mail Session*, in the [Java Mail API design specification](#).

The parameters used to configure a mail session resource can be divided into two groups:

1. mail send (transport) and
2. mail store access

If your enterprise application references a mail session resource, your application will use one of these functions:

- only mail-send
- both mail-send and mail-store access
- only mail-store access

 Use of the *only mail store-access* option is rare. As such, the store access part of the configuration is treated by System Management as optional while the *mail-send* function is treated as mandatory.

## 6.6.37.0: Properties of JavaMail providers

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Class Path



The path to the JAR file containing the implementation class for the JavaMail provider

### Custom Properties



Name-value pairs for setting additional properties beyond those available in the administrative interface

### Description



Optional description for your administrative records

### Name



Administrative name of the JavaMail provider

## 6.6.37.0.1: Properties of JavaMail sessions

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Category



Optional category for classifying this resource for your administrative records

### Custom Properties



Name-value pairs for setting additional properties

### Confirm Password



### or Re-Enter Password



Confirm the password that you entered in the preceding field

### Description



Optional description for your administrative records

### Enable Store



Enable JavaMail sessions

### J2EE Resource Provider



The JavaMail provider with which this JavaMail session is associated

### JNDI Binding Path



### or JNDI Name



The JNDI name for the resource, including any naming subcontexts. This name is used as the linkage between the platform's binding information for resources defined in the client applications deployment descriptor and actual resources bound into JNDI by the platform

### JNDI Name



See JNDI Binding Path

### Name



Administrative name of the JavaMail session object

### Mail From



### or Outgoing Mail Originator



Value of replyTo field in mail messages sent through the mail transport host. The Internet email address that by default will be displayed in the received message as either the "From" or the "Reply-To" address. The recipient's reply will come to the specified address.

### Mail Store Access Host



### or Mail Store Host



The server to which to connect when reading mail. This property combines with the mail store user ID

and password to represent a valid mail account. For example, if the mail account is john\_william@my.company.com, enter my.company.com.

### **Mail Store Access Password** **or Mail Store Password**

The password to use when connecting to the mail store host. This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is john\_william@my.company.com, enter the password corresponding to john\_william.

### **Mail Store Access Protocol** **or Mail Store Protocol**

The protocol to be used when reading mail. Should be IMAP.

### **Mail Store Access User Name** **or Mail Store User**

The user ID to use when connecting to the mail store host. This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is john\_william@my.company.com, enter john\_william.

### **Mail Store Host**

See Mail Store Access Host

### **Mail Store Password**

See Mail Store Access Password

### **Mail Store Protocol**

See Mail Store Access Protocol

### **Mail Store User**

See Mail Store Access User

### **Mail Transport Host** **or Outgoing Mail Server**

The server to which to connect when sending mail. Specify the fully qualified Internet host name of the mail server, also known as the SMTP server

### **Mail Transport Password** **or Outgoing Mail Password**

The password to provide when connecting to the mail transport host. This property is rarely used for the default SMTP protocol. You can leave this field blank unless you use a transport protocol that requires a user ID and password.

### **Mail Transport Protocol** **or Outgoing Mail Protocol**

The transport or protocol to use when sending mail, such as "POP3", "IMAP4", or "SMTP"

The default is set to SMTP, and usually you should not change it. To use a different protocol, first install the required service provider, and then enter the protocol name in this field.

### **Mail Transport User** **or Outgoing Mail User Name**

The user ID to provide when connecting to the mail transport host. This property is rarely used for the default SMTP protocol. You can leave this field blank unless you use a transport protocol that requires a user ID and password.

**Outgoing Mail Originator** 

See Mail From

**Outgoing Mail Server** 

See Mail Transport Host

**Outgoing Mail Password** 

See Mail Transport Password

**Outgoing Mail Protocol** 

See Mail Transport Protocol

**Outgoing Mail User Name** 

See Mail Transport User

**Re-Enter Password** 

See Confirm Password



## 6.6.37.3: Administering mail providers and mail sessions with the Web console

Use the Web console to edit the configurations of mail providers and mail sessions.

Work with objects of this type by locating them in the tree on the left side of the console:

**Resources -> Mail Providers**

Expand the tree further to see the **Mail Sessions** associated with each **Mail Provider** instance.

## 6.6.37.3.1: Configuring new mail providers with the Web console

During this task, you will configure a new mail provider.

1. Click **Resources** -> **Mail Providers** in the tree on the left side of the console.
2. When the mail providers view is displayed on the right side of the console, click the **New** button.
3. Specify the [properties](#) for the new mail provider.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.37.3.3: Updating mail provider configurations with the Web console

During this task, you will update an existing mail provider configuration.

1. In the tree on the left side of the console, click **Resources** -> **Mail Providers** to see a list of available mail providers.
2. Click the mail provider that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the mail provider properties.
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.37.3.4: Updating mail session configurations with the Web console

During this task, you will update an existing mail session configuration.

1. In the tree on the left side of the console, click **Resources** -> **Mail Providers** -> *mail\_provider\_name* -> **Mail Sessions**.
2. Click the mail session that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the [mail session properties](#).
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.37.3.5: Removing mail provider configurations with the Web console

During this task, you will remove an existing mail provider configuration.

1. Click **Resources** -> **Mail Providers** in the tree on the left side of the console.
2. From the mail provider view, select the mail provider to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.37.3.6: Removing mail session configurations with the Web console

During this task, you will remove an existing mail session configuration.

1. Click **Resources** -> **Mail Providers** -> *provider\_name* -> **Mail Sessions** in the tree on the left side of the console.
2. From the mail session view, select the mail session to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.37.9: Administering JavaMail providers and sessions with the Application Client Resource Configuration Tool

Use the Application Client Resource Configuration Tool to edit the configurations of JavaMail sessions to be used by your application clients.

Work with objects of this type by locating them in the tree that is displayed by the tool when you use it to open an EAR file. If file with which you are working contains JavaMail sessions, its tree will contain one or more of the following:

**Resources** -> *application.jar* -> **JavaMail Providers** -> **Mail Provider** (a default mail provider) -> **JavaMail Sessions**

Inside the **JavaMail Sessions** folder will be JavaMail session instances.

## 6.6.37.9.1: Configuring new JavaMail sessions with the Application Client Resource Configuration Tool

During this task, you will configure new mail sessions for your application client. The mail sessions will be associated with the preconfigured default mail provider supplied by the product.

To configure a new JavaMail Session:

1. [Start the tool and open the EAR file](#) for which you want to configure the new JavaMail session. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file in which you want to configure the new JavaMail session.
3. Expand the JAR file to view its contents.
4. Click **JavaMail Providers** -> **MailProvider** -> **JavaMail Sessions**. Do one of the following:
  - Right-click the **JavaMail Sessions** folder and select **New Factory**.
  - On the menu bar, click **Edit** -> **New**.
5. In the resulting property dialog, configure the [JavaMail session properties](#).
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.



## 6.6.37.9.3: Removing JavaMail sessions with the Application Client Resource Configuration Tool

Please see "[Removing objects from EAR files with the Application Client Resource Configuration Tool](#)", as this task is similar for all object types supported by the tool.

## 6.6.37.9.4: Updating JavaMail session configurations with the Application Client Resource Configuration Tool

During this task, you will modify (update) the configuration of an existing JavaMail session. Note, you cannot update the default JavaMail provider, but you can view its properties by performing similar steps.

1. [Start the tool and open the EAR file](#) containing the JavaMail session. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file containing the JavaMail session that you want to update.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular JavaMail session that you want to update. When you find it, do one of the following:
  - Right-click the object and select **Properties**
  - On the menu bar, click **Edit -> Properties**
5. In the resulting property dialog, update the properties. For detailed field help, see:
  - [JavaMail session properties](#)
6. When finished, click **OK**.
7. On the menu bar, click **File -> Save** to save your changes.

## **6.6.38: Administering URL providers and URLs (overview)**

A Default URL Provider is included in the initial product configuration. It utilizes the URL support provided by the JDK. Any URL resources with protocols supported by the JDK (such as HTTP, FTP, FILE) can use the Default URL Provider.

Please see the related information links for tasks and settings related to URL support.

## 6.6.38.0: Properties of URL providers

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Classpath** or **Class Path** or **Server Classpath**

The path to the JAR file containing the implementation classes for the URL provider

**Custom Properties**

Name-value pairs for setting additional properties

**Description**

Optional description of the URL Provider, for your administrative records

**Name**

Administrative name for the URL Provider

**Node**

The node with which the URL Provider is associated

If using the Java-based administrative console, use the buttons on the node panel to access dialogs for installing drivers on specific nodes, and for uninstalling drivers.

**Protocol**

The protocol supported by this stream handler.

*For Advanced Single Server Edition:* This field is required

*For the Application Client Resource Configuration Tool:* This refers to a *custom* protocol. If you are going to use a standard protocol, such as "nntp," "smtp," or "ftp," then leave this field blank

**Server Classpath**

See Classpath

**Stream Handler Class** or **Stream Handler Class Name**

Fully qualified name of Java class that implements the stream handler for the protocol specified by the **Protocol** property

## 6.6.38.0.1: Properties of URLs

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Category



Administrative category for the URL

### Custom Properties



Name-value pairs for setting additional properties

### Description



Optional description of the URL, for your administrative records

### JNDI Binding Path



or JNDI Name



The JNDI name for the resource, including any naming subcontexts. This name is used to link the platform's binding information. The binding associates the resources defined in the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

### JNDI Name



See JNDI Binding Name

### Name



Administrative name for the URL

### Spec



or URL



The string from which to form a URL

### URL



See Spec

### URL Provider



The URL Provider that implements the protocol for this URL

## 6.6.38.3: Administering URL providers with the Web console

Use the Web console to create and edit the configurations of URL providers and URLs.

Work with objects of this type by locating them in the tree on the left side of the console:

**Resources -> URL Providers**

Expand the tree further to see the **URLs** associated with each **URL Provider** instance.

## 6.6.38.3.1: Configuring new URL providers with the Web console

During this task, you will configure a new URL driver.

1. Click **Resources** -> **URL Providers** in the tree on the left side of the console.
2. When the URL providers view is displayed on the right side of the console, click the **New** button.
3. Specify the [properties](#) for the new URL provider.
4. When you are finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.38.3.3: Updating URL provider configurations with the Web console

During this task, you will update the configuration of an existing URL provider.

1. In the tree on the left side of the console, click **Resources** -> **URL Providers** to see a list of available URL providers.
2. Click the URL provider that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the [URL provider properties](#).
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).



## 6.6.38.3.4: Updating URL configurations with the Web console

During this task, you will update existing URL configurations.

1. In the tree on the left side of the console, click **Resources** -> **URL Providers** -> *url\_provider\_name* -> **URLs**
2. Click the URL that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the [URL properties](#).
4. When finished, click **Apply**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.38.3.5: Removing URL provider configurations with the Web console

During this task, you will remove an existing URL provider configuration.

1. Click **Resources** -> **URL Providers** in the tree on the left side of the console.
2. From the URL provider view, select the URL provider to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.38.3.6: Removing URL configurations with the Web console

During this task, you will remove an existing URL configuration.

1. Click **Resources** -> **URL Providers** -> *provider\_name* -> **URLs** in the tree on the left side of the console.
2. From the URL view, select the URL to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.38.9: Administering URL providers and URLs with the Application Client Resource Configuration Tool

Use the Application Client Resource Configuration Tool to edit the configurations of URL providers and URLs to be used by your application clients.

Work with objects of this type by locating them in the tree that is displayed by the tool when you use it to open an EAR file. If the file with which you are working contains URL providers and URLs, its tree will contain one or more of the following:

**Resources** -> *application.jar* -> **URL Providers** -> *url\_provider\_instance*

where *url\_provider\_instance* is a particular URL provider.

If you expand the tree further, you will also see the **URLs** folders containing the URL instances for each URL provider instance.

## 6.6.38.9.1: Configuring new URL providers and URLs with the Application Client Resource Configuration Tool

During this task, you will create URL providers and URLs for your client application. Note, in a separate administrative task, the Java code for the required URL provider must be installed on the client machine on which the client application resides.

To configure a new URL provider:

1. [Start the tool and open the EAR file](#) for which you want to configure the new URL provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file in which you want to configure the new URL provider.
3. Expand the JAR file to view its contents.
4. Click the folder called **URL Providers**. Do one of the following:
  - Right-click the folder and select **New Provider**.
  - On the menu bar, click **Edit -> New**.
5. In the resulting property dialog, configure the [URL provider properties](#).
6. When finished, click **OK**.
7. On the menu bar, click **File -> Save** to save your changes.

## 6.6.38.9.1.1: Configuring new URLs with the Application Client Resource Configuration Tool

During this task, you will create URLs for your client application.

1. In the tree, click the URL provider for which you want to create a URL.
  - [Configure a new URL provider.](#)
  - Or, click an existing URL provider.
2. Expand the URL provider to view its **URLs** folder.
3. Click the folder. Do one of the following:
  - Right-click the folder and select **New Factory**.
  - On the menu bar, click **Edit** -> **New**.
4. In the resulting property dialog, configure the [URL properties](#).
5. When finished, click **OK**.
6. On the menu bar, click **File** -> **Save** to save your changes.

## 6.6.38.9.3: Removing URL providers and URLs with the Application Client Resource Configuration Tool

Please see ["Removing objects from EAR files with the Application Client Resource Configuration Tool"](#), as this task is similar for all object types supported by the tool.

## 6.6.38.9.4: Updating URL and URL provider configurations with the Application Client Resource Configuration Tool

During this task, you will modify (update) the configuration of an existing URL or URL provider.

1. [Start the tool and open the EAR file](#) containing the URL or URL provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file containing the URL or URL provider that you want to update.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular URL or URL provider that you want to update. When you find it, do one of the following:
  - Right-click the object and select **Properties**
  - On the menu bar, click **Edit -> Properties**
5. In the resulting property dialog, update the properties. For detailed field help, see:
  - [URL provider properties](#)
  - [URL properties](#)
6. When finished, click **OK**.
7. On the menu bar, click **File -> Save** to save your changes.



## **6.6.39: Administering messaging and JMS providers (overview)**

Please see the related information links.

## 6.6.39.0: Properties of JMS providers

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Binding Classname** or **Binding Class** or **JNDI Binding Mechanism**

Java classname to be used for namespace binding. This value is required only for providers with non-standard binding requirements.

**Classpath** or **Class Path** or **Server Class Path**

The class path that identifies the location of the driver classes (the JMS initial context factory)

**Context Factory Class** or **Context Factory Classname**

The Java class name of the JMS providers initial context factory

**Custom Properties**

Name-value pairs for setting additional properties

**Description**

Optional description for your administrative records

**External Initial Context Factory**

Java classname of the initial context factory of a provider

**External Provider URL** or **Provider URL**

JMS provider URL for external JNDI lookups

**JNDI Binding Mechanism**

See Binding Classname

**Name**

Administrative name for this provider

**Nodes**

The nodes with which this provider is associated. See also [node properties](#).

If using the Java-based console, use the buttons on the node panel to access dialogs for installing providers on specific nodes, and for uninstalling providers.

**Provider URL**

See External Provider URL

## Server Class Path



See Classpath

## 6.6.39.0.1: Properties of JMS connection factories

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Category



Optional category for your administrative records

### Connection Type



Whether the JMS Destination is a queue or topic. Queues are used for point-to-point messaging. Topics are used for publish-and-subscribe messaging.

### Custom Properties



Name-value pairs for setting additional properties

### Description



Optional description for your administrative records

### External JNDI Name



### or External JNDI Path



Namespace location of JMS created connection factory

### External JNDI Path



See External JNDI Name

### JMS Provider



The JMS provider with which this connection factory is associated

### JNDI Name



Namespace location of JMS created connection factory, including any naming subcontexts.

The name is used to link the platform binding information. The binding associates the resources defined the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

### Name



Administrative name for this JMS connection factory

## 6.6.39.0.2: Properties of JMS destinations

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

### Category



Optional category for your administrative records

### Custom Properties



Name-value pairs for setting additional properties

### Description



Optional description for your administrative records

### Destination Type



Whether the JMS Destination is a queue or topic. Queues are used for point-to-point messaging. Topics are used for publish-and-subscribe messaging.

### External JNDI Name



### or External JNDI Path



Namespace location of JMS created destination

### External JNDI Path



See External JNDI Name

### JMS Provider



The JMS provider with which this connection factory is associated

### JNDI Name



Namespace location of JMS created connection factory, including any naming subcontexts.

The name is used to link the platform binding information. The binding associates the resources defined the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

### Name



Administrative name for this JMS destination

## 6.6.39.3: Administering JMS providers with the Web console

Use the Web console to create and edit the configurations of JMS providers.

Work with objects of this type by locating them in the tree on the left side of the console:

**Resources -> JMS Providers**

## 6.6.39.3.1: Configuring new JMS providers with the Web console

During this task, you will configure a new JMS provider to support applications that utilize Java messaging.

1. In the tree on the left side of the console, click **Resources** -> **JMS Providers** to see a list of available JMS providers.
2. Click the **New** button located above the list.
3. Specify [settings](#) for the new JMS provider.
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.3.2: Configuring new JMS connection factories with the Web console

During this task, you will configure a new JMS connection factory.

1. In the tree on the left side of the console, click **Resources** -> **JMS Providers** -> *provider\_name* -> **JMS Destinations** to display the connection factory view.
2. Click the **New** button located above the list.
3. Specify [settings](#) for the new JMS connection factory.
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).



## 6.6.39.3.3: Configuring new JMS destinations with the Web console

During this task, you will configure a new JMS destinations.

1. In the tree on the left side of the console, click **Resources** -> **JMS Providers** -> *provider\_name* -> **JMS Destinations** to display the connection factory view.
2. Click the **New** button located above the list.
3. Specify [settings](#) for the new JMS provider.
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.3.4: Updating JMS provider configurations with the Web console

During this task, you will update the configurations of resources to support applications that utilize Java messaging.

1. In the tree on the left side of the console, click **Resources** -> **JMS Providers** to see a list of available JMS providers.
2. Click the JMS provider that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the JMS provider properties.
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.3.5: Updating JMS connection factory configurations with the Web console

During this task, you will update the configurations of an existing JMS connection factory.

1. In the tree on the left side of the console, click **Resources** -> **JMS Providers** -> *provider\_name* -> **JMS Connection Factories** to see a list of available JMS connection factories.
2. Click the JMS connection factory that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the [JMS connection factory properties](#).
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.3.6: Updating JMS destination configurations with the Web console

During this task, you will update the configuration of an existing JMS destination.

1. In the tree on the left side of the console, click **Resources** -> **JMS Providers** -> *provider\_name* -> **JMS Destinations** to see a list of available JMS destinations.
2. Click the JMS destination that you want to modify. Its properties will be displayed on the right side of the console.
3. Modify the **JMS destination** properties.
4. When finished, click **OK**.
5. [Save your configuration](#).
6. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.3.7: Removing JMS provider configurations with the Web console

During this task, you will remove an existing JMS provider configuration.

1. Click **Resources** -> **JMS Providers** in the tree on the left side of the console.
2. From the JMS provider view, select the JMS provider to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.3.8: Removing JMS connection factory configurations with the Web console

During this task, you will remove an existing JMS connection factory configuration.

1. Click **Resources** -> **JMS Providers** -> *provider\_name* -> **JMS Connection Factories** in the tree on the left side of the console.
2. From the JMS Connection Factory view, select the JMS connection factory to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.3.10: Removing JMS destination configurations with the Web console

During this task, you will remove an existing JMS destination configuration.

1. Click **Resources** -> **JMS Providers** -> *provider\_name* -> **JMS Destinations** in the tree on the left side of the console.
2. From the JMS Destination view, select the JMS destination to delete. Click **Remove**.
3. [Save your configuration](#).
4. (Optional) To have the configuration take effect:
  1. [Stop the server](#).
  2. [Start it again](#).

## 6.6.39.9: Administering JMS providers, connection factories, and destinations with the Application Client Resource Configuration Tool

Use the Application Client Resource Configuration Tool to edit the configurations of JMS providers, JMS connection factories, and JMS destinations to be used by your application clients.

Work with objects of this type by locating them in the tree that is displayed by the tool when you use it to open an EAR file. If the file with which you are working contains JMS providers, JMS connection factories, and JMS destinations, its tree will contain one or more of the following:

**Resources** -> *application.jar* -> **JMS Providers** -> *jms\_provider\_instance*

where *jms\_provider\_instance* is a particular JMS provider.

If you expand the tree further, you will also see the **JMS Connection Factories** and **JMS Destinations** folders containing the connection factory and destination instances for each JMS provider instance.



## 6.6.39.9.1: Configuring new JMS providers with the Application Client Resource Configuration Tool

During this task, you will create new JMS provider configurations for your applicationclient. The client application can make use of a messaging service through the Java Message Service APIs. A JMS provider provides two kinds of J2EE factories. One is a JMS Connection factory, and the other is aJMS destination factory.

Note, in a separate administrative task, the JMS client must be installed on the client machine where the client application resides. The messaging product vendor must provide an implementation of the JMS client. For more information, see your messaging product documentation.

To configure a new JMS provider:

1. [Start the tool and open the EAR file](#) for which you want to configure the new JMS provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file in which you wantto configure the new JMS provider.
3. Expand the JAR file to view its contents.
4. Click the folder called **JMS Providers**. Do one of the following:
  - Right-click the folder and select **New Provider**.
  - On the menu bar, click **Edit** -> **New**.
5. In the resulting property dialog, configure the [JMS provider properties](#).
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.

## 6.6.39.9.1.1: Configuring new connection factories with the Application Client Resource Configuration Tool

During this task, you will create a new JMS connection factory configuration for your applicationclient.

To configure a new connection factory:

1. In the tree, click the JMS provider for which you want to create a connection factory.
  - [Configure a new JMS provider.](#)
  - Or, click an existing JMS provider.
2. Expand the JMS provider to view its **JMS Connection Factories** folder.
3. Click the folder. Do one of the following:
  - Right-click the folder and select **New Factory**.
  - On the menu bar, click **Edit** -> **New**.
4. In the resulting property dialog, configure the [JMS connection factory properties](#).
5. When finished, click **OK**.
6. On the menu bar, click **File** -> **Save** to save your changes.

## 6.6.39.9.1.2: Configuring new JMS destinations with the Application Client Resource Configuration Tool

During this task, you will create new JMS destination configuration for your applicationclient.

To configure a new destination:

1. In the tree, click the JMS provider for which you want to create a destination.
  - [Configure a new JMS provider](#).
  - Or, click an existing JMS provider.
2. Expand the JMS provider to view its **JMS Destinations** folder.
3. Click the folder. Do one of the following:
  - Right-click the folder and select **New Factory**.
  - On the menu bar, click **Edit** -> **New**.
4. In the resulting property dialog, configure the [JMS destination properties](#).
5. When finished, click **OK**.
6. On the menu bar, click **File** -> **Save** to save your changes.

## 6.6.39.9.3: Removing JMS providers, connection factories, and destinations with the Application Client Resource Configuration Tool

Please see ["Removing objects from EAR files with the Application Client Resource Configuration Tool"](#), as this task is similar for all object types supported by the tool.

## 6.6.39.9.4: Updating JMS provider, connection factory, and destination configurations with the Application Client Resource Configuration Tool

During this task, you will modify (update) the configuration of an existing JMS provider, connection factory, or destination.

1. [Start the tool and open the EAR file](#) containing the JMS provider, connection factory, or destination. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file containing the JMS provider, connection factory, or destination that you want to update.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular JMS provider, connection factory, or destination that you want to update. When you find it, do one of the following:
  - Right-click the object and select **Properties**
  - On the menu bar, click **Edit -> Properties**
5. In the resulting property dialog, update the properties. For detailed field help, see:
  - [JMS provider properties](#)
  - [JMS connection factory properties](#)
  - [JMS destination properties](#)
6. When finished, click **OK**.
7. On the menu bar, click **File -> Save** to save your changes.

## **6.6.41: Administering WebSphere administrative domains (overview)**

Please see the related information links.

## 6.6.41.0: Administrative domain properties

Key:



Applies to Java administrative console of Advanced Edition Version 4.0



Applies to Web administrative console of Advanced Single Server Edition Version 4.0



Applies to Application Client Resource Configuration Tool

**Name**



A logical name for the WebSphere administrative domain

## 6.6.41.3: Administering administrative domains with the Web console

Use the Web console to edit the configuration of the administrative domain that contains your application server, its contents, and the supporting resources.

Work with the domain by clicking it in the tree on the left side of the console:

**WebSphere Administrative Domain**



## 6.6.41.3.4: Updating administrative domain configurations with the Web console

During this task, you will update the configurations the administrativedomain that contains your application server, all of its contents, and supporting resources.

1. In the tree on the left side of the console, click **Administrative Domain**. Its properties will be displayed on the right side of the console.
2. Modify the administrative domain properties.
3. When finished, click **OK**.
4. [Save your configuration](#).
5. (Optional) To have the configuration take effect:
  1. [Stop the server](#)
  2. [Start the server again](#).

## 6.6.41.5: Establishing multiple administrative domains on a machine

This procedure describes how to install multiple WebSphere 4.0 domains on the same physical machine and allow them to run concurrently. This configuration is useful for allowing developers or testers to isolate their WebSphere domain from other WebSphere administrative domains on the same machine. It also provides the ability to make changes to the overall domain without affecting other domains.

The procedure describes how to set up this configuration using examples from Solaris 8 and Windows NT. The Solaris examples can be extrapolated to other UNIX operating systems supported by IBM WebSphere Application Server. It is intended for experienced administrators who need to setup and maintain multiple WebSphere instances on multiprocessor servers.

The UNIX configuration used in this document is:

- Hardware: Sun Ultra 80 4x300MHz with 1024 MB of RAM
- Operating system: Sun Solaris 8
- Database: IBM DB2 Version 7.1.0.43
- Web Server: IBM HTTP Server 1.3.19.0

The Windows NT configuration used in this document is:

- Hardware: Dual 400 MHz Pentium II Netfinity with 650 MB of RAM
- Operating system: Windows NT 4.0 Service Pack 6a
- Database: IBM DB2 Version 7.1.0.43
- Web Server: IBM HTTP Server 1.3.19.0

Overview:

1. Install and configure separate HTTP servers and their WebSphere plug-ins
2. Run the WebSphere installation twice with the appropriate configuration information.
3. Change the default ports for the bootstrap service, LSD service, and Trace Service.
4. Adjust the port values to eliminate conflicts.
5. Start the servers in the configuration.
6. Test each Web server and application server combination.

### Install the first Web server instance

Install the Web server as you usually would, making note of:

- The port on which the Web server is listening. Look for the port line in the httpd.conf file.
- The path in which the Web server is installed

Note, IBM HTTP Server is available during IBM WebSphere Application Server installation, which is the next step.

### Install the first IBM WebSphere Application Server instance

See the InfoCenter [PDF view](#) for case-specific installation documents, and [section 2](#) for more information.

Make note of:

- The path in which IBM WebSphere Application Server is installed, such as:  
AIX:                    /opt/WebSphere/AppServer40a  
Windows NT:           C:\WebSphere\AppServer40a

### Test the single instance setup

1. Start the Web server. Ensure it is working.
2. [Start the WebSphere Application Server product.](#)
3. [Start the administrative console.](#)
4. [Start the application server.](#)
5. [View the HTTP transport properties.](#)

Record the port numbers on which the Web container HTTP transport is listening, for future reference.

6. Try to access the "snoop servlet" for verification:

`http://HTTP_Server_hostname/servlet/snoop`

This confirms that the "one instance" setup has basic functionality.

7. Stop the Web server.
8. Stop the application server.

### Install the second Web server instance

To install the second instance of the Web server, you need either to create a new Web server instance using the Web server configuration application, as used by Netscape Enterprise Server, or to make copies of the necessary configuration files and hand edit them inserting appropriate values, as is necessary when using IBM Web server or Apache Server.

In this example, we will hand edit the configuration files, because we are using IBM Web server.


1. **NT** Perform this step if using Windows NT. Otherwise, skip this step.

To install a second copy of the IBM HTTP Server on Windows NT, first you must remove an entry in the Windows Registry so the installprogram does not detect the already installed version.

1. Start regedit
2. Carefully, remove just this one key:  
My Computer\HKEY\_LOCAL\_MACHINE\SOFTWARE\IBM\HTTP Server
3. Copy the httpd.conf and apachectl files in their corresponding directories, such as:  
httpd.conf copy to httpd2.confapachectl copy to apachectl2
4. Edit the httpd2.conf file. Change the values slightly (such as adding 2 to their values).
  - o Change the port to an unused port, such as: Port 81
  - o Change the error log, such as (for Solaris): ErrorLog /opt/IBMHTTPD/logs/error\_log2
  - o Change the custom log, such as (for Solaris): CustomLog /opt/IBMHTTPD/logs/access\_log2 common
  - o Change the pid file, such as (for Solaris): PidFile /opt/IBMHTTPD/logs/httpd2.pid
  - o Change the score board file, such as (for Solaris): ScoreBoardFile /opt/IBMHTTPD/logs/httpd2.scoreboard
5. Save the httpd2.conf file.
6. Open the apachectl2 file in a text editor and make similar changes. For example, for Solaris:
  - o PIDFILE=/opt/IBMHTTPD/logs/httpd2.pid
  - o HTTPD=/opt/IBMHTTPD/bin/httpd -f /opt/IBMHTTPD/conf/httpd2.conf(Note the single quotation marks in the HTTPD command).
7. Save the apachectl2 file.

## Install the second WebSphere Application Server instance

Install the second WebSphere Application Server, with these notes:

- Use the Custom installation option.
- Skip the migration assistant if it is present.
- Do not "Backup and Uninstall" the previous version.
- When the installion notifies you that it has found another copy ofWebSphere 4.0, simply click **OK** to continue.
- Do not install the Web server or the database.
- Install the appropriate Web server plug-in.
- For *Advanced Edition* (non-Single Server), name the database something unique, such as "was40b" if the first installation's database was "was40a."
- For the Destination Directory, enter a unique directory for this instance,such as (for Solaris): /opt/WebSphere/AppServer40b
-  The installation program will onlyaccept a file named httpd.conf when it asks the location of the httpd.conf file. To get past this correctly, move thehttpd.conf file to httpd.confA and move httpd2.conf to httpd.conf.

After the installation program updates thehttpd.conf file, move the files back to their original names, httpd.conf andhttpd2.conf.

Test the second instance installation, just as you did with the first instance to verify that each set of HTTP/AppServer functions correctly alone.(Note, you might need to restart the Web server to have it read in the plugin-cfg.xml file.)

## Adjust the port values to eliminate conflicts

With the configuration as it stands, you can run each HTTPServer and Application Server combination separately, but you cannot run bothinstances concurrently due to conflicting ports.

To allow both instances to run at the same time, edit the port settings.

1. Edit the server.cfg.xml file with a text editor. Change the ports to unusedvalues. For example:
  - o Search for orbSettings and change the bootstrapPort attribute to 901.
  - o Search for locationServiceDaemon and change the port attribute to 9001.
  - o Search for traceService and change the diagThreadPort attribute to 7001.
2. Append a new port number to all of the virtual host settings of thisinstance of WebSphere. This is the value that the corresponding HTTPServer will listen on.

Change the default ports of the host aliases:

From \*:80 to \*:81From \*:9080 to \*:9081

[Editing virtual host settings](#)

3. Edit the server.cfg.xml file with a text editor.
  - o Search for VirtualHosts and increment all of the port attributes of the host aliases by 1. (For example, change 9080 to 9081.

By default there are two VirtualHosts. Be sure to change the ports in both sections. Make sure that you set the values to ports that are free and valid.

4. Verify that each Web Container port does not conflict with any otherWeb Container port, both on this instance of WebSphere Application Server and otherinstances.

Edit the server.cfg.xml file with a text editor.

- o Search for webContainer and increment all of the transports' port attributes by 1, such as changing 9080 to 9081.

Make sure this port is free and is valid.

5. **NT** For Windows NT, change the HTTP server port numberfrom 80 to another valid port number, such as 81, in httpd.conf.

## Start the servers in the configuration

To run this configuration, you will need to use a command prompt to start each Web server and each WebSphere Application Server.

1. **NT** For Windows NT, stop all of the running Webserver and application server services. Set each of the services to **manual** so that they will not start automatically. The services to set are:  
IBM HTTP Administration          IBM HTTP Server          IBM WS AdminServer
2. Start each Web server.
3. Start each application server by changing directory to the bin directory of each of the two installations and running the startupServer script.

**Test each Web server and application server combination**

To test that each Web server and application server pair has basic functionality, start the default application server in each WebSphere ApplicationServer instance. Try the following URLs to verify that basic functionality exists:  
`http://HTTP_server_hostname/servlet/snoop``http://HTTP_server_hostname:81/servlet/snoop``http://HTTP_server_hostname/webapp/examples/HitCount``http://HTTP_server_hostname:81/webapp/examples/HitCount`

**Additional usage notes**

Whenever you add a new Web Container to either instance of WebSphere Application Server, you must make sure that the port number of the WebContainer Service does not conflict with another Web Container Service on either instance of WebSphere Application Server.

## 6.6.43: Administering references

References are logical names used to locate external resources for enterprise applications. References include EJB references and resource references (for external resources such as databases and messaging systems).

References are defined in the application's deployment descriptor file by using the ApplicationAssembly Tool. At deployment, the references are bound to the physical location (JNDI name) of the resource in the target operational environment.

## 6.6.43.0: Properties related to references

References are defined in an application's deployment descriptor by using the Application Assembly Tool. Refer to the assembly properties listed in the related information.

## 6.6.43.0.1: Assembly properties for EJB references

An EJB reference is a logical name used to locate the home interface of an enterprise bean used by the application. At deployment, the EJB reference is bound to the enterprise bean's home in the target operational environment. The container makes the application's EJB references available in a JNDI naming context. It is recommended that references to enterprise beans be organized in the `ejb` subcontext of the application's environment (in `java:comp/env/ejb`).

### **Name (Required, String)**

Specifies the JNDI name of the enterprise bean's home, relative to the `java:comp/env` context. For example, if `ejb/EmplRecord` is specified, the referencing code looks up the enterprise bean's home at `java:comp/env/ejb/EmplRecord`. This JNDI name is a JNDI name alias used by the code (the actual JNDI name is specified on the Binding tab).

### **Link**

Used to link an EJB reference to an enterprise bean in the current module (the same module as the one making the reference) or in another module within the same encompassing J2EE application. The value of this property is the name of the target enterprise bean. (The target enterprise bean can be in any EJB module in the same J2EE application as the referencing module.) To avoid having to rename enterprise beans to have unique names within an entire J2EE application, specify the path name of the EJB JAR file containing the referenced enterprise bean and append the target bean's name, separated by a `#` symbol, for example, `...products/product.jar#ProductEJB`. The path name is relative to the referencing module's JAR file. If a link is not specified, the reference must be resolved to a JNDI name during installation.

### **Home (Required, String)**

Specifies the fully qualified name of the enterprise bean's home interface. An example is `com.ibm.ejbs.EmplRecordHome`.

### **Remote (Required, String)**

Specifies the fully qualified name of the enterprise bean's remote interface. An example is `com.ibm.ejbs.EmplRecord`.

### **Type (Required, String)**

Specifies the expected type of the referenced enterprise bean. The value must be either `Entity` or `Session`.

### **Description**

Contains text describing the EJB reference.

### **JNDI Name**

Binding information is used by the run-time environment to resolve the location of a resource. For EJB references, the JNDI name must match the JNDI name of the enterprise bean as specified on the Binding tab in the EJB module containing the bean.

## 6.6.43.0.2 Assembly properties for resource references

A resource reference declares a logical name used to locate a connectionfactory object. These objects define connections to external resources such as databases and messaging systems. The container binds these references to actual resource manager connection factories in the target operational environment. It is recommended that resource references be organized in the subcontexts of the application's naming environment, using a different subcontext for each resource manager type. For example, all JDBC DataSource references can be declared in the `java:comp/env/jdbc` context. All JMS connection factories can be declared in the `java:comp/env/jms` subcontext. All JavaMail connection factories can be declared in the `java:comp/env/mail` subcontext. All URL connection factories can be declared in the `java:comp/env/url` subcontext. All J2C connection factories can be declared in the `java:comp/env/eis` subcontext.

### **Name (Required, String)**

Specifies the JNDI name used to retrieve a connection factory for a resource manager (relative to the `java:comp/env` context). For example, if the name specified is `jdbc/EmployeeAppDB`, the referencing code looks up the factory under the name `java:comp/env/jdbc/EmployeeAppDB`.

### **Type (Required, String)**

Specifies the Java programming language type of the connection factory object. For obtaining JDBC API connections, use `javax.sql.DataSource`. For obtaining JMS connections, use `javax.jms.QueueConnectionFactory` or `javax.jms.TopicConnectionFactory`. For obtaining JavaMail connections, use `javax.mail.Session`. For obtaining URL connections, use `java.net.URL`.

### **Authentication (Required, String)**

Specifies whether the enterprise bean (or servlet) code signs on programmatically to the resource manager, or whether the container signs on to the resource manager on behalf of the bean (or servlet). In the latter case, the container uses information that is supplied by the deployer. The value of this field must be one of the following: `Application` or `Container` (for enterprise beans) and `Servlet` or `Container` (for Web applications). Note for J2C (Connector Architecture for WebSphere Application Server) resources: This property is valid only for session beans. There is no CMP support yet. WebSphere Application Server supports only component-managed sign-on (Option C in the J2EE/Connector Architecture specification) in this release. As a result, the value for this property is ignored and processed as `Application`. This means that either the session bean needs to pass the user ID and password credentials on the `getConnection` call or the J2C Connection Factory needs to have its user ID and password fields filled in (this is done by using the WebSphere administrative console).

### **Description**

Contains text describing the connection factory object.

### **JNDI name**

Specifies the name of the connection factory in the global JNDI namespace.



## 6.6.43.0.3: Assembly properties for security role references

### **Role name (Required, String)**

Specifies the name of a security role reference used in the applicationcode, for example, "boss." The AccountBean can make a decision based onwhether the user executing a method is granted the role of a "boss."

### **Link**

Specifies the name of a security role defined in the encompassingapplication. The role reference will be linked to this name. Forexample, the AccountBean code uses a role named "boss." The AccountBean is a part of an enterprise application, FinanceApp, that has a role named"Manager." If the link specifies "Manager," then when the bean makes acall to isCallerInRole("boss"), the result will be true if and only if theuser, who invoked the method, has been granted the FinanceApp's Managerrole. The security role reference is the name used by an applicationcomponent (module), and the link name is the name defined in the deploymentdescriptor of the encompassing application. The link maps the name usedin the component to a corresponding name in the application.

### **Description**

Contains text describing the security role.

## 6.6.45: Administering WebSphere plug-ins for Web servers

The WebSphere Application Server works with a Web server to handle requests for Web applications. The Web server and application server communicate using the WebSphere plug-in for the Web server.

When you install WebSphere Application Server, it modifies the Web server configuration file automatically to establish the plug-in. The exception is for installations using Domino Server, in which you need to perform manual steps to update the Web server configuration file after installing WebSphere Application Server. For more information, see "Modifications to Web server configurations during product installation."

### Administering Web servers

Refer to your Web server documentation as the ultimate source of information about administering your Web server. See also the subtopics of this article.

Ensure that Web servers are configured to perform operations required by Web applications, such as GET and POST. Typically, this involves setting a directive in the Web server configuration file (such as httpd.conf for IBM HTTP Server). Refer to the Web server documentation for instructions. If an operation is not enabled when a servlet or JSP page requiring the operation is accessed, an error message is displayed, such as this one from IBM HTTP Server:

```
HTTP method POST is not supported by this URL.
```

### The internal HTTP transport and the Web server plug-in

A Web server and WebSphere plug-in for the Web server are required for use in a production environment, for performance reasons. But strictly speaking, they are not required in order to start the application server or the administrative console. In a test or development environment, you might want to use the [internal HTTP transport described in the transport administration overview](#), instead of the Web server plug-in and Web server.

The remainder of this article and its sub-topics (6.6.45.\*) discuss the Web server plug-in.

### Administering WebSphere plug-ins for Web servers

Version 4.x introduces the HTTP transport plug-ins as the preferred method of communication between the Web server and the application server. The HTTP plug-in introduces the following advantages:

- XML-based configuration file
- Standard protocol recognized by firewall products
- Security using HTTPS, replacing proprietary OSE over SSL

The remainder of this article discusses the main plug-in tasks and when to perform them:

- [Manually triggering a regeneration of the plug-in configuration](#)
- [Manually editing the plug-in configuration](#)

### Manually triggering a regeneration of the plug-in configuration

The WebSphere administrative console supplies a manual way to force plug-in configurations to update (regenerate) themselves. For a summary of the occasions on which to update the plug-in configuration, see the [instructions for regenerating the plug-in](#).

## Time required for plug-in regeneration

Regenerating the configuration might take a while to complete. After it is finished, all objects in the administrative domain will use their newest settings, of which the Web server is now aware.

Whether triggered manually or occurring automatically, plug-in regeneration requires about 30 - 60 seconds to complete, when the application server product is on the same physical machine (node) as the Web server. In other cases, it takes more time.

The delay is important because it determines how soon the new plug-in configuration will take effect. Suppose you add a new served path for a servlet, then regenerate the plug-in configurations. The regeneration requires 40 seconds, after which a user should be able to access the servlet by the new served path. You can see how dynamic plug-in configuration enables you to make newly configured resources available to users right away.

For an HTTP plug-in, the length of the delay is determined by the Refresh Interval attribute of the Config element in the plugin-cfg.xml file. The plug-in polls the disk at this interval to see whether the configuration has changed. The default interval is 60 seconds.

To regenerate the plug-in configurations requires twice the refresh interval.

In a development environment in which you are frequently changing settings in the administrative console, it is recommended you set the refresh interval to 3 to 5 seconds.

In a production environment, set a longer refresh interval, perhaps as long as 30 minutes, depending on the frequency of changes.

## Example of regenerating the plug-in configuration

After using the administrative console to make configuration changes that involve the served paths of Web applications, manually trigger the regeneration of the plug-in configuration (or manually edit the file if that is what you have been doing).

When you trigger the regeneration of the plug-in configuration:

1. The Web container containing the Web application registers the configuration changes to those objects.
2. The Web container unloads the Web application, then loads it again, causing the Web application to adopt the new settings.

Note that the state and data of any servlets running in the Web application will be lost, unless the Web application usually persists data (for example, it saves servlet-generated session and user profile data in a database).

3. The plug-in configuration is regenerated. The Web server is aware of the new Web application configuration. If a user requests the servlet using the path specified by the new Web resource, the request should be successful.

## **Manually editing the plug-in configuration**

The [Web server plug-in property reference](#) describes the location of the plug-in configuration files.

Use care when you are hand-editing the HTTP plug-in configuration, because your changes will be overwritten if the plug-in configuration is regenerated. However, regenerating the plug-in configuration cannot guarantee a correct configuration file for advanced configuration scenarios. If you see strange behavior after triggering a regeneration of the plug-in configuration, consider manually editing the configuration.

## 6.6.45.0: Properties of WebSphere plug-ins for Web servers

- [Finding the plug-in configuration files](#)
- [HTTP plug-in properties](#)
- [See also Transport Properties](#)

### Finding the plug-in configuration files

The working or active versions of the HTTP plugin-cfg.xml file resides in the directory:

`product_installation_root/config`

### HTTP plug-ins

The plugin-cfg.xml file includes the following elements and attributes:

Config (exactly one)

This element starts the WebSphere HTTP plug-in configuration file. It contains all other elements and attributes of the configuration.

Refresh interval (exactly one)

The time interval (in seconds) at which the plug-in should check the configuration file to see if updates or changes have occurred. The plug-in checks the file for any modifications that have occurred since the last time the plug-in configuration was loaded.

In a development environment in which changes are frequent, a lower setting than the default setting of 60 is recommended. In production, a higher value than the default is a good idea since updates to the configuration will not occur as often. If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in config reload is successful. If you are not seeing the changes you made to your plug-in configuration, check the plug-in log file for indications of the problem.

Log (zero or one for each Config)

The log describes the location and level of log messages that should be written by the plug-in. If a Log is not specified within the Config, then in some cases log messages will be written to the Web server error log.

Name (exactly one)

The fully qualified path to the log file to which the plug-in will write error messages.

If the file does not exist then it will be created. If the file already exists then it will be opened in append mode and the previous plug-in log messages will remain.

LogLevel (zero or one for each Log)

The level of detail of the log messages that the plug-in should write to the log. Values for this attribute are one of the following:

- Trace
- Warn
- Error

If a LogLevel is not specified with the Log, then the default value is Error.

Trace allows you to see the steps in the request process in detail. Warn and Error means that only information about abnormal request processing will be logged.

Be careful when setting the level to Trace. A lot of error messages are logged at this level which can cause the disk space to fill up very quickly. A Trace setting should never be used in a normally functioning environment as it affects performance.

```
<ServerGroup name="Servers">      <Server name="Server1">          <Transport Hostname="localhost"
Port="9080" Protocol="HTTP"/>      <Transport Hostname="localhost" Port="9443"
Protocol="HTTPS">                  <Property name="Keyring"
value="c:/WebSphere/AppServer/keys/keyring.kdb"/>          <Property name="Stashfile"
value="c:/WebSphere/AppServer/keys/keyring.sth"/>      </Server></ServerGroup>
```

ServerGroup (one or more for each Route)

A group of servers that are generally configured to service the same types of requests.

In the simplest case, the server group will contain only one server definition. In the case in which more than one server is defined, the plug-in will load balance across the defined servers using either a Round Robin or a Random algorithm. The default is Round Robin.

Name (exactly one for each ServerGroup)

The logical or administrative name to be used for this group of servers.

LoadBalance (one for each ServerGroup)

The default load balancing type is Round Robin

The Round Robin implementation has a random starting point. This means that the first server picked will be done so randomly and then round robin will be used from that point forward. This is so that in multiple process based Web servers all of the processes don't

start up by sending the first request to the same application server.

**RetryInterval (one for each ServerGroup)**

An integer specifying the length of time that should elapse from the time that a server is marked down to the time that the plug-in will retry a connection. The default is 60 seconds.

**RemoveSpecialHeaders (one for each ServerGroup)**

The plug-in adds special headers to the request before it is forwarded to the application server. These headers store information about the request that will need to be used by the application. By default the plug-in will remove these headers from incoming requests before adding the headers it is supposed to add.

The value can be true or false.

Setting the attribute to false introduces a potential security exposure by not removing headers from incoming requests.

**Server (one or more for each ServerGroup)**

A WebSphere Application Server instance that is configured to handle requests routed to it given the routing rules of the plug-in configuration. The Server should correspond to an application server running on either the local machine or a remote machine.

**Name (exactly one for each Server)**

The administrative or logical name for the server.

**CloneID (zero or one for each Server)**

If this unique ID is present in the HTTP Cookie header of a request (or the URL if using URL rewriting), the plug-in routes the request to this particular server, provided all other routing rules are met. If a CloneID is not specified in the Server, then Session Affinity is not enabled for this server.

Used in conjunction with Session Affinity. When this attribute is set the plug-in will check the incoming cookie header or URL for jsessionid. If the jsessionid is found then the plug-in will look for a clone ID or clone IDs. If clone IDs are found and a match is made to this attribute then the request will be sent to this server rather than being load balanced across the server group.

If you are not using Session Affinity then it is best to remove these clone IDs from the configuration because there is added request processing in the plug-in when these are set. If clone IDs are not in the plug-in then it is assumed that session affinity is not on and the request is load balanced across the server group.

**WaitForContinue (one for each Server)**

This true or false attribute allows you to specify whether to use the HTTP 1.1 100 Continue support before sending the request content to the application server. The default is false.

This function is necessary when configuring the plug-in to work with certain types of proxy firewalls. The plug-in by default does not wait for the 100 Continue response from the application server before sending the request content because it is a performance hit.

**Transport (one or more for each Server)**

The transport for reading and writing requests to a particular WebSphere application server instance. The transport provides the information needed to determine the location of the application server to which the request will be sent. If the Server has multiple transports defined to use the same protocol, the first one will be used.

It is possible to configure the Server to have one non-secure transport and one that uses SSL. In this configuration, a match of the incoming request protocol will be performed to determine the appropriate transport to use to send the request to the application server.

**Hostname (exactly one for each Transport)**

The hostname or IP address of the machine on which the WebSphere application server instance is running.

**Port (exactly one for each Transport)**

The port on which the WebSphere application server instance is listening.

**Protocol (exactly one for each Transport)**

The protocol to use when communicating over this transport -- either HTTP or HTTPS.

**Property (zero, one, or more for each Transport)**

When the Protocol of the Transport is set to HTTPS, use this element to supply the various initialization parameters, such as keyfile and stashfile.

**Name (exactly one for each Property)**

The name of the Property being defined. Supported names recognized by the transport are keyring, stashfile, and password.

**Value (exactly one for each Property)**

The value of the Property being defined.

```
<VirtualHostGroup name="Hosts"> <VirtualHost name="www.x.com"/> <VirtualHost name="www.x.com:443"/>
<VirtualHost name="*:8080"/> <VirtualHost name="www.x.com:*/> <VirtualHost
name="*:*/></VirtualHostGroup>
```

**VirtualHostGroup (zero, one, or more for each Config)**

A group of virtual host names that will be specified in the HTTP Host header. Enables you to group virtual host definitions together that are configured to handle similar types of requests.

**Name (exactly one for each VirtualHostGroup)**

The logical or administrative name to be used for this group of virtual hosts.

VirtualHost (one or more for each VirtualHost)

The name used for a virtual or real machine used to determine if incoming requests should be handled by WebSphere Application Server or not. Use this element to specify hostnames that will be in the HTTP Host header that should be seen for requests that need to be handled by the application server. You can specify specific hostnames and ports that incoming requests will have or specify a \* for either the hostname, port, or both.

Name (exactly one for each VirtualHost)

The actual name that should be specified in the HTTP Host header in order to match successfully with this VirtualHost.

The value is a hostname or IP address and port combination, separated by a colon.

The plug-in can be configured to route requests to the application server based on the incoming HTTP Host header and port for the request. Name allows you to specify what those combinations are.

Wildcarding is available for this attribute. The only acceptable solutions are either a \* for the hostname, a \* for the port, or a \* for both. A \* for both means that any request will match this rule. If no port is specified in the definition the default HTTP port of 80 is assumed.

```
<UriGroup name="Uris"> <Uri name="/servlet/snoop"/> <Uri name="/webapp/*"/> <Uri name="*.jsp"/></UriGroup>
```

UriGroup (one or more for each Config)

A group of uris that will be specified in the HTTP request line. They will be able to be handled by the same application server. The route will compare the incoming uri with the uris in the group to determine if the application server will handle the request.

Name (exactly one for each UriGroup)

The logical or administrative name for this group of uris.

Uri (one or more for each UriGroup)

The virtual path to the resource that will be serviced by WebSphere Application Server. Each Uri specifies the incoming URLs that need to be handled by the application server. Wildcarding is available in these definitions.

Name (exactly one for each Uri)

The actual string that should be specified in the HTTP request line in order to match successfully with this Uri. Wildcarding within the uri definition is acceptable. You can specify rules such as \*.jsp or /servlet/\* to be handled by WebSphere Application Server.

```
<Route VirtualHostGroup="Hosts" UriGroup="Uris" ServerGroup="servers"/>
```

Route (one or more for each Config)

A request routing rule by which the plug-in will determine if an incoming request should be handled by a WebSphere application server.

The route definition is the central element of the plug-in configuration. It specifies how the plug-in will handle requests based on certain characteristics of the request. The route definition contains the other main elements: a required ServerGroup, and either a VirtualHostGroup, UriGroup, or both.

Using the information that is defined in the VirtualHostGroup and the UriGroup for the route, the plug-in determines if the incoming request to the Web server should be sent on to the ServerGroup defined in this route.

VirtualHostGroup (zero or one for each Route)

The group of virtual hosts that should be used in route determination. The incoming host header and server port are matched to determine if this request should be handled by the application server.

It is possible to omit this from the route definition. If it is not present then every request will match during the virtual host match portion of route determination.

UriGroup (zero or one for each Route)

The group of uris to use for determining the route. The incoming uri for the request are matched to the defined uris in this group to determine if this request should be handled by the application server.

It is possible to omit this from the route definition. If it is not present then every request will match during the uri match portion of route determination.

ServerGroup (exactly one for each Route)

The server group to which to send request that successfully match the route.

The server group that should be used to handle this request. If both the uri and the virtual host matching is successful for this route then the request will be sent to one of the servers defined within this server group.

## 6.6.45.0.1: Modifications to Web server configuration files during product installation

Based on the plug-ins selected, the following modifications are made to the Web server configuration files during the installation process. For the exact Web server brands, products, and versions supported by your version and edition of WebSphere Application Server, see the [product prerequisites Web site](#).

The purpose of this article is to help you identify the additions in the event that you need to restore the configuration file to its original form, and do not have a backup copy. Refer also to the updated configuration file itself, as it could contain slightly different and more current modifications as this documentation ages and product fix packs are released.

In the actual configuration file, paths shown in the example below, such as:

```
C:/WebSphere/AppServer/bin/mod_app_server_http.dll
```

should reflect the [product\\_installation\\_root](#) for your particular installation and operating system. The default Windows installation root is used in the example.

If installing on a UNIX-based system and you find that few of the modifications were made to the Web server configuration file, try logging on directly as root to perform the installation. Installation problems of this nature have been reported when using su from another account, rather than the root account.

### Apache (httpd.conf)

For the HTTP plug-in:

```
LoadModule app_server_module C:/WebSphere/AppServer/bin/mod_app_server_http.dll
```

### Apache (srm.conf)

For the HTTP plug-in:

```
Alias /IBMWebAS/samples/ "C:/WebSphere/AppServer/samples/"Alias /IBMWebAS/  
"C:/WebSphere/AppServer/web/"NcfAppServerConfig BootFile  
C:\WebSphere\AppServer\config\plugin-cfg.xml
```

### Domino (httpd.cnf)

For the HTTP plug-in, updating the configuration file is a manual process as described in "Manually updating the Domino Web server configuration file."

### IBM HTTP Server (httpd.cnf)

For the HTTP plug-in:

```
Alias /IBMWebAS/samples/ "C:/WebSphere/AppServer/samples/"Alias /IBMWebAS/  
"C:/WebSphere/AppServer/web/"LoadModule ibm_app_server_module  
C:/WebSphere/AppServer/bin/mod_app_server_http.dllNcfAppServerConfig BootFile  
C:\WebSphere\AppServer\config\plugin-cfg.xml
```

### Netscape or iPlanet (obj.conf)

For the HTTP plug-in:

```
Init fn="load-modules" funcs="as_init, as_handler,  
as_term"shlib="C:/WebSphere/AppServer/bin/ns41_http.dll"Init fn="init_exit"  
bootstrap.properties="C:/WebSphere/AppServer/config/plugin-cfg.xml"Service fn ="as_handler"
```

## 6.6.45.5: Controlling where the WebSphere plug-ins for Web servers are installed

Sometimes, you might want to install the WebSphere plug-ins for Web servers to locations other than their default locations.

### Installing Web server plug-ins to non-default locations

Depending on the operating system, when you select to install a Web server plug-in, the plug-in may be installed silently if the Web server is installed in a standard location for that Web server brand. In such a case, the WebSphere Application Server installation does not prompt you for the configuration file location.

If you have installed two Web servers on the same machine, the "standard" place for installing the plug-in might be okay for one of the servers, but will not succeed in installing the plug-in for the server that is in the non-standard location.

For example, suppose IBM HTTP Server "installation 1" is installed in the standard directory in which the Web server is installed on Solaris, `/opt/IBMHTTPD`. When the WebSphere plug-in for this first Web server is installed, it will silently modify the `httpd.conf` file in `/opt/IBMHTTPD/conf`, which is okay.

But now suppose that you installed IBM HTTP Server "installation 2" in `/opt/IHS2`. When the Web server plug-in for this second Web server is installed, it too will be installed in `httpd.conf` in `/opt/IBMHTTPD/conf`.

One way around this is to manually edit the `httpd.conf` file of the second IBM HTTP Server installation, rather than installing the WebSphere plug-in. You could copy the modified configuration file from the first Web server and make it the configuration file of the second server. Of course, this approach assumes that you have not customized the configuration file of the second server, and find it acceptable for its configuration to match that of the first server.

Another way is to install both Web servers in some non-standard place, such as `/opt/IHS1` and `/opt/IHS2`. When the WebSphere Application Server installation program cannot find either `httpd.conf` file, you will be prompted for which one to edit. Specify the location of one of the Web server files. Then repeat the plug-in installation, specifying the other location this time.

### Installing WebSphere plug-ins on non-default IIS servers

When the product installs the WebSphere plug-in for Microsoft Internet Information Server (IIS), it assumes the user wants to attach the plug-in to the default IIS server. The following instructions explain how to attach the plug-in to an IIS server *other than* the default.

The procedure might be necessary for a user implementing multiple Web sites that separate the pages they serve along some logical boundary, such as security level. Follow the steps to allow a newly defined site (using an IIS instance other than the default) to exercise servlets in conjunction with an existing site or sites.

The instructions apply to IIS Versions 4.x and 5.x.

1. Use the Internet Service Manager (from Microsoft IIS) to create a new site with a name, port number, and base subdirectory.
2. [Open the WebSphere Application Server administrative console](#).
3. [Configure the virtual host](#) to contain an [alias](#) for the port number used by the site.
4. Create a virtual directory for the new site.
  1. Open the Internet Service Manager for IIS.



2. Select the new site in the Tree View.
3. Right-click to display a menu. Select New -> Virtual Directory.
4. Ensure the values are set appropriately:
  - The name of the virtual directory should be set to SEPLUGINS (using all capital letters, as shown).
  - The physical path should be set to the WebSphere bin directory (such as c:\WebSphere\AppServer\bin on Windows NT).

Note, the directory must have the EXECUTE permission set, but setting any other permissions (or allowing it to inherit other permissions) is a security risk.

5. Start the new site.
6. Issue a browser request to verify that the configuration works, such as:


http://mymachine:8080/servlet/snoop

7. The administrator must ensure that the SEPLUGINS retains its EXECUTE permission.

It is possible for virtual directories under a site to inherit properties from the site. The administrator must ensure that the SEPLUGINS virtual directory does not inherit permissions from changes to the site, if those changes involve withdrawing the EXECUTE permission.

## 6.6.45.6: Regenerating the Web server plug-in configuration

Sometimes you might want to trigger (manually) the WebSphere plug-in to regenerate its configuration. For example, you might have installed or removed an enterprise application, or added or removed servlets and mappings from a particular application. Failure to regenerate the plug-in after introducing a new application will likely result in a "404 File Not Found" error when a user tries to access the new Web application.

 This task can overwrite manual configuration changes that you might want to preserve instead. Before performing this task, understand its implications as described in the article about [administering Web server plug-ins](#).

You can run the following script:

```
product\_installation\_root/bin/GenPluginCfg.sh | bat
```

You can also use the administrative node to manually trigger an update of the configuration for the WebSphere plug-in for the Web server you are using:

1. [Display the application server properties](#).
2. Modify the **Web Server Plug-in Configuration**.
3. Save your changes.

## 6.6.45.7: What to do after changing Web server ports

Suppose you change a Web server port. You will need change the WebSphere virtual host configurations associated with the Webserver.

Specifically, modify the virtual host aliases for that Web server to reflect the new port number. After doing so, stop the application servers associated with the affected virtual hosts and start them again.

For example, if you change the Web server port to 9082:

1. For each virtual host whose configuration contains [aliases](#) for the Web server,
2. use the administrative console to [display virtual host properties](#).
  1. Locate the alias settings.
  2. Append :1111 to every alias pertaining to the Web server with the changed port.
  3. Save the changes.
3. Identify the application servers associated with the virtual host.
4. [Stop the application servers](#) and [start them again](#).
5. [Regenerate the HTTP plug-in configuration](#).

A virtual host configuration assumes the Web server port to be 80, unless otherwise specified. If you change the port from some other number to 80, either append :80 to the virtual host aliases, or make sure you remove the previous port number from any aliases.

## 6.6.45.8: Checking your IBM HTTP Server version

1. Change directory to the installation root of the Web server. For example, this is /opt/IBMHTTPD on a Solaris machine.
2. Locate the file named version.signature.
3. At a system command prompt in that directory, enter:  
`more version.signature`

The version will be reported as IBM HTTP Server 1.3.12.xx .

## 6.6.45.9: Manually updating the Domino Web server configuration file

At the time of this writing, you must update the Domino Web server configuration manually, after installing IBM WebSphere Application Server, and before running the product. See "Administering WebSphere plug-ins for Web servers" for more information about configuration file updates.

Note, the following third party product instructions are subject to change, without notice to IBM. Refer to your Web server documentation for the latest and most detailed instructions.

1. Start the Domino server.
2. Start the Domino administrative interface.
3. In the Domino administrative interface, edit the server.
  1. Click the **Configuration** tab.
  2. On the left side, click **Server -> All Server Documents**.
  3. When a list of servers is displayed on the right side, double-click the Domino server whose configuration file you need to update.
  4. When the **Edit Server** button is displayed, click it to display the server properties.
  5. Click the **Internet Protocols** tab.
  6. Place your cursor between the brackets next to **DSAPI filter file names**. Type the full path to the WebSphere DSAPI plug-in, such as:  

`product_installation_root/bin/domino5_http.dll`
  7. Click the **Save and Close** button.
4. Close the Domino administrative interface.

## 6.6.48: Administering ports

The product uses the following ports. See below for port assignment guidelines, particularly notes about the effects of port changes on other WebSphere components.

### Bootstrap port

- One for each application server
- Default value is 900
- To change it, use the administrative console to modify [ORB properties](#) or update the orbSettings object in the [application server configuration file](#)

### Location Service Daemon (LSD) port

- One for each application server
- Default value is 9000
- To change it, use the administrative console to modify [LSD properties](#) or update the location service object in the service configuration in the [application server configuration file](#)

### ORB listener port for RMI/IIOP

- One for each application server
- Value is assigned randomly
- To make the value static, add **-Dcom.ibm.CORBA.ListenerPort=xxxx** where xxxx is a valid TCP port (see guidelines below).
  - For application servers, add the argument to the Command Line Arguments setting of the application server properties.

### Port for [internal HTTP transport](#)

- One for each application server
- Default value is 9080

### Secure port for internal HTTP transport

- One for each application server
- Default value is 9443

### Administrative application port

- One for each product installation
- Default value is 9090
- To change it, modify the [application server configuration file](#)

### DrAdmin port

- One for each application server
- Value is 7000
- To change it, use the administrative console to modify the [trace properties](#) or modify the [application server configuration file](#) directly

## Object Level Trace and Debugger port

- One for each application server
- Value is 2012
- To change it, use the administrative console to modify [OLT and Debugger properties](#)

## Guidelines for assigning ports

Though WebSphere Application Server provides default values, the above port numbers can be modified by the system or network administrator. Here are some guidelines:

- Ports can range from 1024 to 64000. Choose a port that does not conflict with existing ports in use. To check ports in use:
  - Use the netstat command (netstat -a)
  - View the /etc/services file on UNIX
  - View the *drive\winnt\system32\drivers\etc\services* file on Windows NT
- Ports must be unique in the scope of each physical host. That is, two servers on the same machine cannot have the same port values.
- The same port numbers *can* be used for servers running on different physical hosts.
- Remember to configure firewalls to allow traffic to pass for each assigned port. For security, try to minimize ports.

## 6.6.49: Administering National Language Support

IBM WebSphere Application Server supports several Asian and European language locales, as described in the table below.

This article provides information about the supported locales.

### Supported locales

WebSphere Application Server supports the locales listed in the table below. The support is dependent on the availability of the TrueType fonts listed for the respective languages.

Language	Windows NT/2000	Solaris	AIX	HP-UX
Brazilian Portuguese	IBM-850	pt_BR	pt_BR.ISO8859-1	No Brazilian Portuguese locale
French	IBM-850 IBM-858	fr fr.ISO8859-15	fr_FR.ISO8859-1 fr_FR.ISO8859-1@euro	fr_FR.ISO8859-1 fr_FR.ISO8859-15@euro
German	IBM-850 IBM-858	de de.ISO8859-15	de_DE.ISO8859-1 de_DE.ISO8859-1@euro	de_DE.ISO8859-1 de_DE.ISO8859-15@euro
Italian	IBM-850 IBM-858	it it @ISO8859-15	it_IT.ISO8859-1 it_IT.ISO8859-1@euro	it_IT.ISO8859-1 it_IT.ISO8859-15@euro
Japanese	IBM-943	ja ja_PC.PCK	ja_JP.IBM.eucJP Ja_JP.IBM-942	ja_JP.SJIS *
Korean	IBM-1363	ko	ko_KR.IBM.eucKR	ko_KR.eucKR *
Simplified Chinese	IBM-1386	zh	zh_CN.IBM.eucCN	zh_CN.hp15CN *
Spanish	IBM-850 IBM-858	es es @ISO8859-15	es_ES.ISO8859-1 es_ES.ISO8859-1@euro	es_ES.ISO8859-1 es_ES.ISO8859-15@euro
Traditional Chinese	IBM-950	zh_TW.big5	zh_TW.big5	zh_TW.big5 *



## 6.6.51: Administering network configurations

This article discusses topics related to network configurations in a WebSphere environment. Network configurations can affect your WebSphere configuration.

### Multiple networks

WebSphere Application Server tolerates the presence of more than one network on a single platform as long as the environment used to run WebSphere Application Server is configured such that the following conditions are met:

- The primary hostname refers to the default - primary - network interface.
- The Domain Name Server's response to a query of the platform's hostname returns the address of the primary network interface.
- The IP for the platform resolves to the hostname and vice versa for both formats of hostname: fully qualified and not fully qualified.

Here is a good example of a multi-network interface card Windows NT platform that conforms to the above specifications. The hostnames, IP addresses, and adapter IDs have been genericized to *my.server.net*, patterns such as *aaa.bbb.ccc.ddd*, and variables of the form *adapter\_ID\_n*, respectively:

```
d:\work>ipconfig
Windows NT IP Configuration
Ethernet adapter adapter_ID_1: IP Address. . . . . : aaa.bbb.ccc.ddd (Primary IP)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 208.180.235.1
Ethernet adapter adapter_ID_2: IP Address. . . . . : eee.fff.ggg.hhh
Subnet Mask . . . . . : 255.255.254.0
Default Gateway . . . . . : 129.17.32.1
d:\work>nslookup
aaa.bbb.ccc.ddd
Server: my.server.net
Address: mmm.nnn.ooo.ppp
Name: cdm-235-122-pflu.cox-internet.com
(Resolves to hostname)
Address: aaa.bbb.ccc.ddd
d:\work>hostname cdm-235-122-pflu (hostname is set correctly)
d:\work>nslookup cdm-235-122-pflu (and resolves to the same)
Server: my.server.net
Address: mmm.nnn.ooo.ppp
Name: cdm-235-122-pflu.cox-internet.com
Address: aaa.bbb.ccc.ddd (IP when NOT fully qualified)
d:\work>nslookup cdm-235-122-pflu.cox-internet.com (also resolves to the same)
Server: my.server.net
Address: mmm.nnn.ooo.ppp
Name: cdm-235-122-pflu.cox-internet.com
Address: aaa.bbb.ccc.ddd (IP when fully qualified)
```

## 6.6a: Starting and stopping servers

**Starting the server.** Use one of these methods. See below for details:

- By typing "startServer" at a system command prompt
- Using the **Services** window [NT](#)
- By selecting **IBM WebSphere -> Application Server V4.0 -> Start Application Server** from the Start menu [NT](#)
- From the product **First Steps** dialog, which is displayed at the end of the product installation
- From the **Servers** page of the administrative console
- By generating a startup script, possibly modifying its parameters, and running the script
- By running the basic startup script, provided for backward compatability

**Stopping the server.** Use one of these methods. See below for details:

- By typing "stopServer" at a system command prompt
- Using the **Services** window [NT](#)
- From the product **First Steps** dialog, which is displayed at the end of the product installation
- From the **Servers** page of the administrative console
- Using the "DrAdmin" script

### [Starting and stopping the administrative console](#)

### Starting and stopping servers from the Servers page of the administrative console

1. [Open the administrative console.](#)
2. Access the **Servers** page by...
3. Click the **Start** or **Stop** button, depending on whether you want to start or stop the server.

The **Servers** page supports one server at a time. It can be used to start multiple servers; however, to do so the appropriate server configurations must be individually loaded and individually launched.

The **Servers** page cannot be used to launch the server configuration containing the administrative application. (See [the article about server configuration files](#) for details about that configuration.)

### Starting and stopping servers as services from the Windows NT Services window

You can use the **Services** window of your Windows NT or 2000 operating system to start and stop the IBM WebSphere Application Server service. Note, you can also use the command line options or **Servers** page of the administrative console to stop a server that you started as a service.

### Starting and stopping servers from the First Steps dialog

The **First Steps** dialog uses the startServer and stopServer script (see the description of those scripts). It keeps the system command window open so that you can view the server launch output until you close the window. Closing the window will not terminate the server process.

## Starting and stopping servers from the command line

There are a few ways to start the server from the command line:

- Using `"startServer"` (recommended)
- Using `"startServer -script"` to generate a startup script, then running the script
- Using `"startServerBasic,"` the basic startup script

... and a couple of ways to stop the server from the command line:

- Using `"stopServer"` (recommended)
- Using `"DrAdmin"`

### startServer script


The recommended way to start the *Advanced Single Server Edition* from a commandline is to run the script:

```
product_installation_root/bin/startServer.bat | sh
```

from a system command prompt.


When you use startServer, the server obtains its process definition from a [server configuration file](#). The logs are written as defined in the ioRedirect element of the server configuration file.

The startServer script waits for the server that it launched to report status back to it. Therefore, the script might take several minutes to complete.

 Stopping the startServer script will not necessarily cause the server process to be terminated. See below to learn [how to stop the server](#).

### startServer -script option

Add the -script option to the startServer command to generate a startup script file. You can view and edit the startup parameters, then launch the startup script.

 Note that on UNIX operating systems, the generated script will not be marked as an executable switch. You must correct this manually.

Stopping the generated script will stop the server.


### startServerBasic option

You can also start the server by running the script:

```
product_installation_root/bin/startServerBasic.bat | sh
```

The script is provided for backward compatibility, and is *not* the recommended means of launching the server.

Stopping the startServerBasic script *will* cause the server process to be terminated (unlike with the startServer script).

 The basic startup script does not pick up any of the [process definition settings](#), which include the [JVM settings](#). For example, if you set the classpath in the JVM settings, it will be disregarded.

It also does not make use of the configured log files. Server output is written to the command prompt from which the server was started.

### **Stopping the server using stopServer**

The recommended way to stop the *Advanced Single Server Edition* from a commandline is to run the script:

```
product_installation_root/bin/stopServer.bat | sh
```

from a system command prompt.

Note, the stopServer script submits a request to stop the server, but does not tell if the server was completely stopped. Examine the server process to tell whether the server has been stopped successfully.

### **Stopping the server using DrAdmin**

The DrAdmin script can be stopped from the command line using the DrAdmin script. As with the stopServer script, an alternate configuration file can be specified. The DrAdmin script provides additional options that are beyond the scope of this article.

Like the stopServer script, the DrAdmin script does not provide complete shutdown information. Examine the server process to tell if the server has been successfully stopped.

Both DrAdmin and the stopServer script operate by accessing server runtime functions through a runtime utility port configured as the 'diagThreadPort' and 'diagThreadHostname'. These values can be specified directly to DrAdmin to avoid reading the configuration file.

## **Advanced tasks pertaining to starting and stopping the server**

The remainder of this article provides instructions for tasks closely related to starting the server.

### **Specifying a different configuration file**

By default, the startServer script starts an application server, using configuration file:

```
product_installation_root/config/server-cfg.xml
```

You can override the default configuration by specifying your own configuration file, using the -configFile <full path to XML configuration file> parameter.

See [the server configuration file topic](#) for information about alternative configuration files.

On Windows-based systems, a start Application Server entry is added to the program folder for WebSphere Application Server. If you want to run with a server configuration other than server-cfg.xml, update the program folder entry to specify the desired configuration file.

### **Obtaining feedback from the server**

The start process launches the application server as a background process. All application server output (stdout and stderr) is routed to files as configured in the server configuration file (such as [product\\_installation\\_root/logs/default\\_server\\_stdout.txt](#)).

The startServer process will monitor the progress of the startup of the server and log messages to the command shell window indicating the progress. It will output the message:

```
WSPL0057I: the server is open for e-business
```

when the started application server is ready for requests.

## **Enabling the server to be started directly from the command shell**

The characteristics of the application server process are defined under the Process Definition object in the server configuration. In the case that you wish to have more control over how the application server is launched, you can specify the `-script` parameter with the `startServer` command. This will generate a script that can be used to start the application server directly from the command shell.

## **Finding usage information for the startup script**

To see the complete command line usage for `startServer`, enter:

```
startServer -usage
```

Additional information is available by entering:

```
startServer -help
```

## **Ability to start and stop a remote server**

The script must be run on the same machine on which you want to start the server. Launching a remote application server is not supported.

Stopping a server remotely is only supported if the runtime utility port is configured to allow remote access. You can perform a remote shutdown of a server using the `DrAdmin` command with `diagThreadHostname` configured to contain a hostname other than `localhost`.

## 6.6.a.1: Running the product servers and consoles as non-root

The application server can be run using a non-root ID. By default, WebSphere servers use a root ID. Use these instructions to change the ID.

1. Change permissions to the product installation directories to allow access to the administrative server when it "runs as" a non-root ID. Do **one** of the following:
  - Change the ownership of all files and directories under the [product\\_installation\\_root](#) to the user and group that you want the administrative server to "run as."
  - Change the ownership of these specific files and directories to the user and group that you want the administrative server to "run as."
    - [product\\_installation\\_root/bin/\\*](#)
    - [product\\_installation\\_root/config/\\*](#)
    - [product\\_installation\\_root/etc/\\*](#)
    - [product\\_installation\\_root/installedApps/\\*](#)
    - [product\\_installation\\_root/logs/\\*](#)
    - [product\\_installation\\_root/properties/\\*](#)
    - [product\\_installation\\_root/tranlog/\\*](#)
    - [product\\_installation\\_root/temp/\\*](#)

Make sure that the user has write and execute privileges.
2. Ensure that all ports have a value greater than or equal to 1024. The default configuration file, server-cfg.xml, must have its orbSetting's bootstrapPort changed:
  - a. Open the [server configuration file](#) in a text editor.
  - b. Change the bootstrapPort of the orbSettings:

```
<orbSettings xmi:id="ORBConfig_1" enable="true" bootstrapHost="localhost" bootstrapPort="2222">
```

where 2222 is just an example of a new port that you might use.

Changing the bootstrap port affects the clients that connect to the server. See the [port administration overview](#) for details.
3. Start the administrative server, using the new ID.

## 6.7: Tutorials

The tutorials help you learn about various aspects of application assembly, configuration, deployment, and special topics, such as SOAP support and debugging. The available tutorials are listed in the related information to the right.

The tutorials refer to example code that is shipped with the product or the documentation set, allowing you to practice the tasks firsthand.

### Example code for "main" tutorials

The "main" set of tutorials, 6.7.1 - 6.7.6, comprise an end-to-end view of the administrative procedure. The main tutorials are not included in the **Back** and **Forward** button sequence. Instead, each one provides a link to the next tutorial in the sequence.

Save the following code to a directory of your choice:  
[tutorial.zip](#)

For the tutorials, you will want to extract the files (preserving directory structure) so that you have the following directory:

[product\\_installation\\_root](#)/temp/tutorial/

with subdirectories under "tutorial" matching the subdirectories in the tutorial.zip file (cmp, security, and assembly).

### Tasks covered in "main" tutorials

The following table summarizes the tasks covered by the main set of tutorials.

Segment of main tutorial ->	6.7.1: Assembly	6.7.2: Deployment	6.7.3: Test	6.7.4: Security	6.7.5: Advanced (CMP)	6.7.6: Cleanup
Start assembly tool	✓			✓	✓	
Convert EJB 1.0 JAR to EJB 1.1 JAR	✓				✓	
Assemble EJB module (Session bean)	✓				✓	
Assemble Web module	✓					
Assemble application client module	✓				✓	

#### Related topics

[Home \(Getting started page\)](#)

#### Tutorials

[6.7.1: Application assembly tutorial](#)

[6.7.2: Application deployment tutorial](#)

[6.7.3: Application testing tutorial](#)

[6.7.4: Application security tutorial](#)

[6.7.5: Advanced application assembly and deployment tutorial -- CMP bean application](#)

[6.7.6: Application cleanup and removal tutorial](#)

[6.7.soap: Deploying a Java class as a Web service, using SOAP](#)

[6.7.sq: StockQuote tutorial: Using Debugger and OLT](#)

Assemble J2EE application	✓				✓	
Generate code for deployment	✓				✓	
Exit assembly tool	✓			✓	✓	
Start application server		✓		✓	✓	
Open administrative console		✓		✓	✓	
Deploy (install) J2EE application		✓		✓	✓	
Regenerate Web server plug-in		✓		✓	✓	
Stop and restart application server		✓		✓	✓	
Test Web client			✓			
Test Java client			✓		✓	
Enable security during assembly				✓		
Enable security in application server				✓		
Deploy secured application				✓		
Test secured Web client				✓		
Test secured Java client				✓		
Create database tables					✓	
Configure data source					✓	
Uninstall application						✓

6.7.hc:  
HitCount  
tutorial: Using  
Debugger and  
OLT

### Peer topics

6.1: Quick  
reference for  
administration

6.2: Preparing  
to host  
applications

6.3:  
Assembling  
applications  
and generating  
deployment  
code

6.4: Installing  
applications  
and setting  
classpaths

6.5:  
Maintaining  
and updating  
applications

6.6: Tools and  
resources quick  
reference

6.6.a.1:  
Running the  
product servers  
and consoles as  
non-root

6.6a: Starting  
and stopping  
servers

6.10: Backing  
up and  
restoring  
administrative  
configurations

**InfoCenter**



To launch the full documentation set in a separate browser window, click:

[Display InfoCenter](#)

### **PDF library**

To browse the PDF library for this product, containing this article and others, click:

 [PDF versions](#)

### **Using this documentation**

Become an InfoCenter super user! To find out more about navigation, numbering, search, downloads, and more, click:

[Using this documentation](#)

 [Back](#)

[Forward](#) 

## 6.7.1: Application assembly tutorial

During this tutorial, you will assemble a J2EE application .ear file from an EJB JAR file, some Web application components, and a Java application client JAR file. Assembly is a prerequisite to deploying your application on the application server. Put simply, assembly involves identifying the various code artifacts as a single unit (application), and configuring the deployment descriptor of the application.

The application used for this tutorial is a Session bean application, but the assembly steps are similar for a BMP bean application. For information about CMP bean applications, see the [Advanced tutorial](#), which assumes knowledge of this assembly tutorial and others.



### Prerequisites

There are no prerequisites for performing this tutorial.

### Overview of steps (requires 30 to 50 minutes)

1. [Obtain the tutorial application](#)
2. [Start the Application Assembly Tool](#)
3. [Assemble an EJB module, converting EJB 1.0 JAR to EJB 1.1](#)
4. [Assemble a Web module](#)
5. [Assemble an application client module](#)
6. [Assemble a J2EE application](#)
7. [Generate code for deployment](#)
8. [Exit the Application Assembly Tool](#)

### Paths through the tutorial

- **Quick path:** If your objective is to practice these steps as quickly as possible, skip the explanations marked with the  graphic.
- **Maximum learning path:** If your objective is to understand and explore the product capabilities, read the  notes and browse the links they provide to additional documentation.

### Obtain the tutorial application

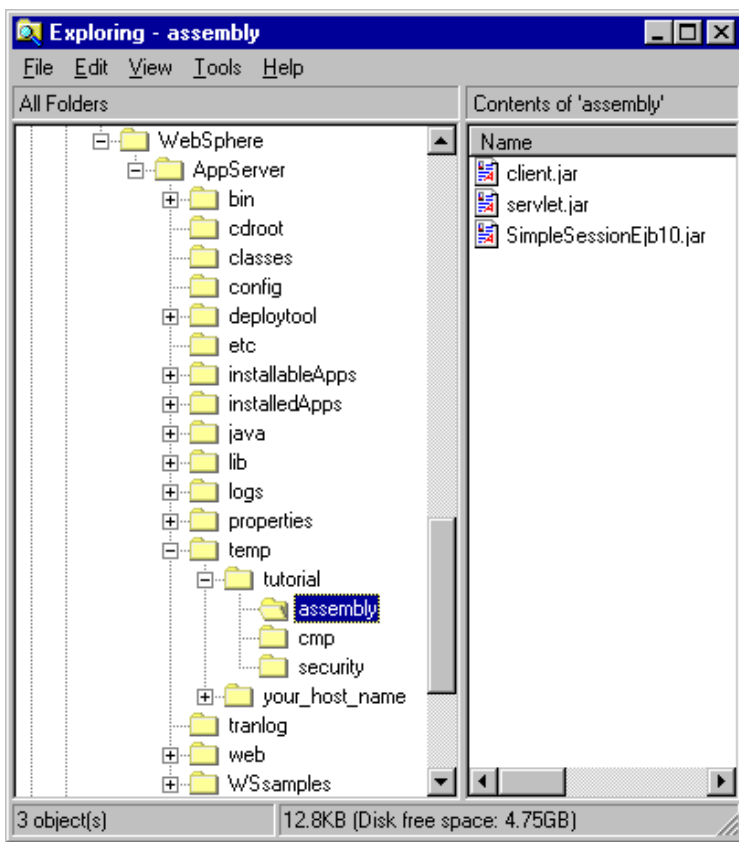
1. Create a directory named "tutorial" under the path:

`product_installation_root/temp`

2. [Click here](#) to access the .zip file containing the tutorial application components.
3. Save the .zip file to

`product_installation_root/temp/tutorial`

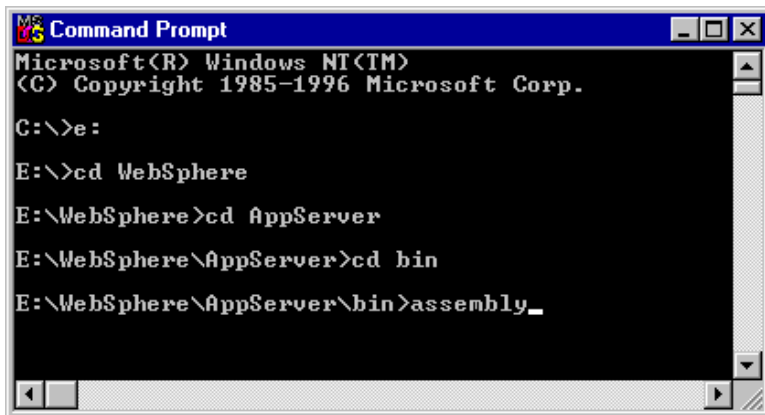
4. Use your favorite .zip or .jar utility to extract the tutorial.zip content into the tutorial directory.



## Start the Application Assembly Tool

Start the [Application Assembly Tool \(AAT\)](#) by:

1. Open a system command prompt.
2. Change directory to [product\\_installation\\_root/bin](#)
3. Type assembly.

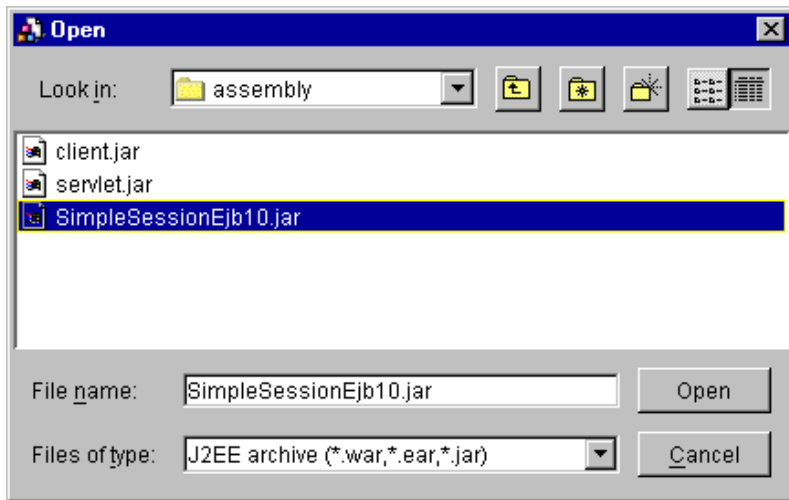


**NT** If using Windows NT or 2000, and you do not see the graphical interface of the tool right away, check for the minimized tool on the Task Bar.

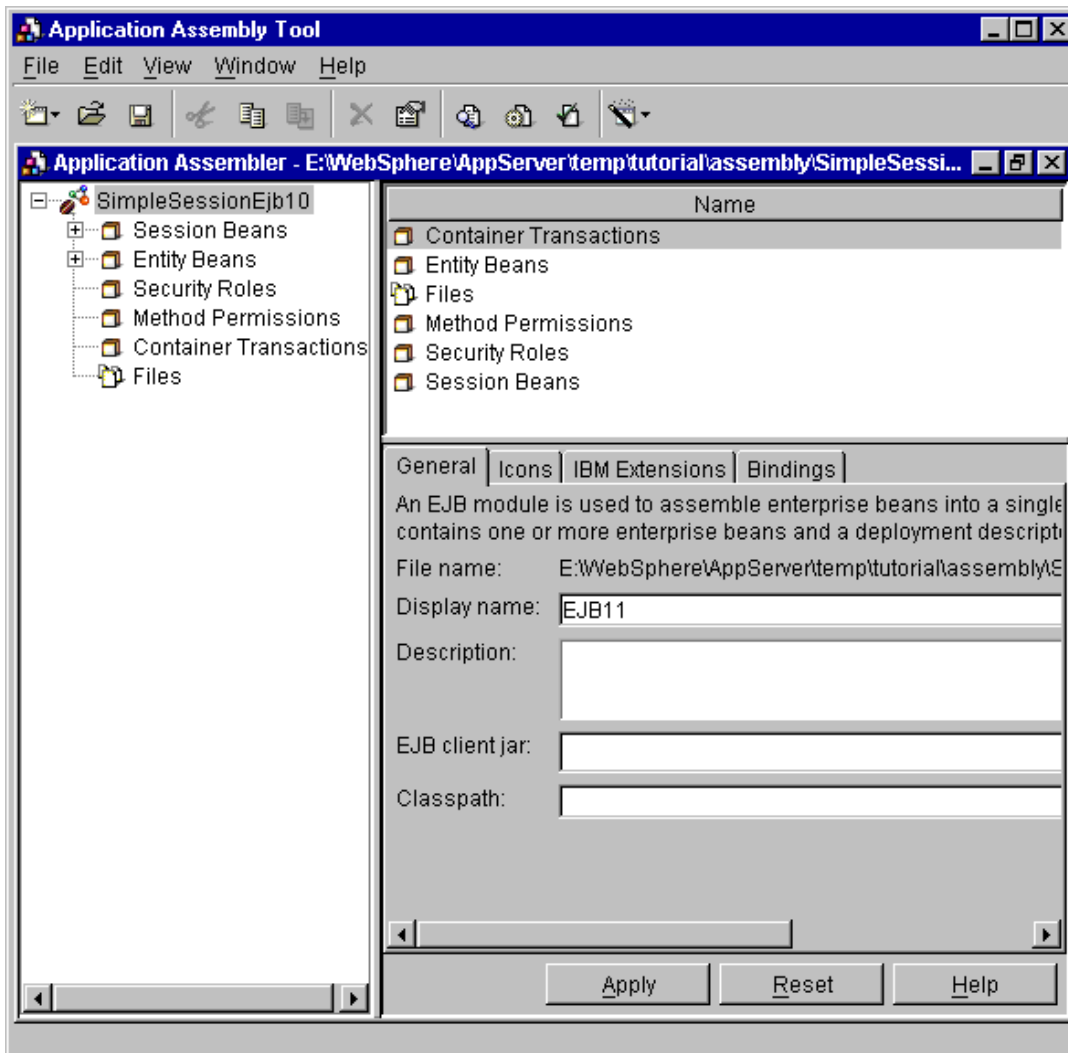
## Assemble an EJB module, converting EJB 1.0 JAR to EJB 1.1

1. Click **Cancel** at the **Welcome to Application Assembly Tool** panel.
2. Load the EJB jar file into the tool:
  - a. Click **File** -> **Open**.
  - b. Navigate to the **SimpleSessionEjb10.jar** and select **open**.
  - c. Click **OK** at the "... please specify the dependent classpath ..." dialog.
3. [Optional] Edit the deployment descriptor.
  - a. Select the jar (the top entry in the assembly tool tree view) and click it to display its properties.
  - b. Enter the following:
    1. Display Name: **EJB11**
  - c. Click **Apply**

4. Save your changes using **File->Save As**. Save the file as **Ejb11.jar** in the path:  
`product_installation_root/temp/tutorial/assembly/Ejb11.jar`

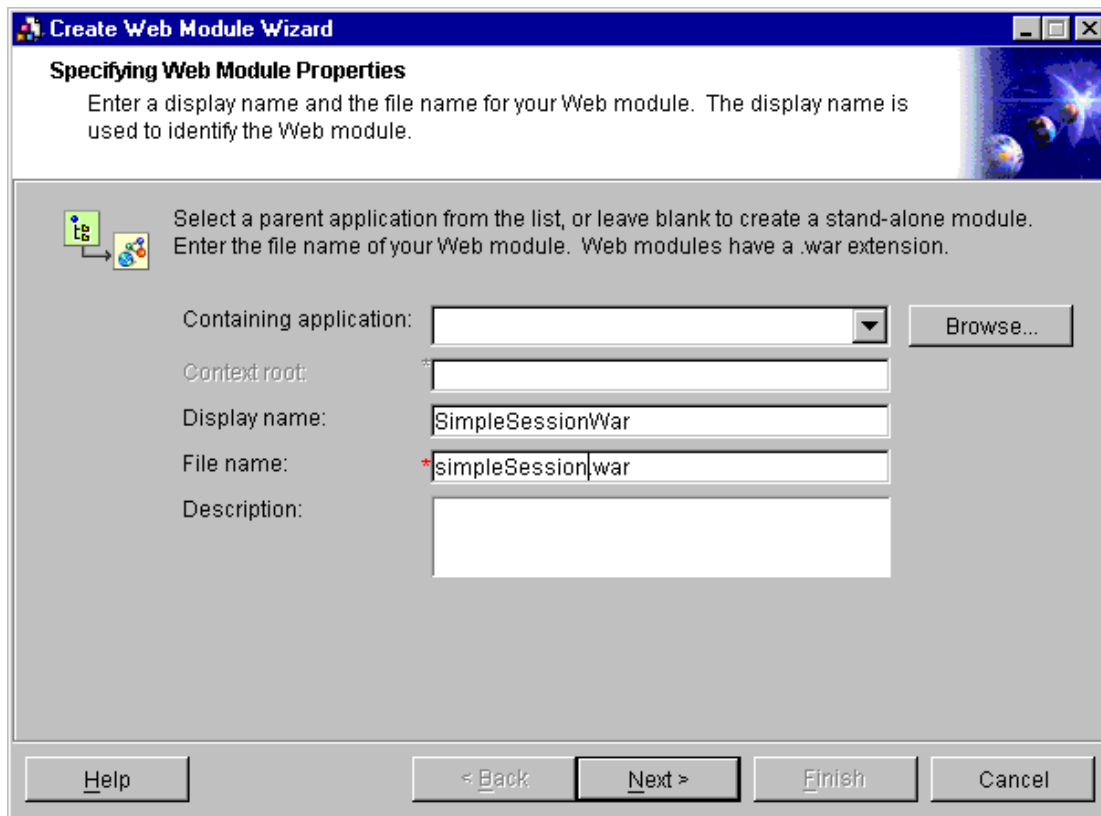


You now have a view of the EJB module in the Application Assembly Tool tree view.



## Assemble a Web Module

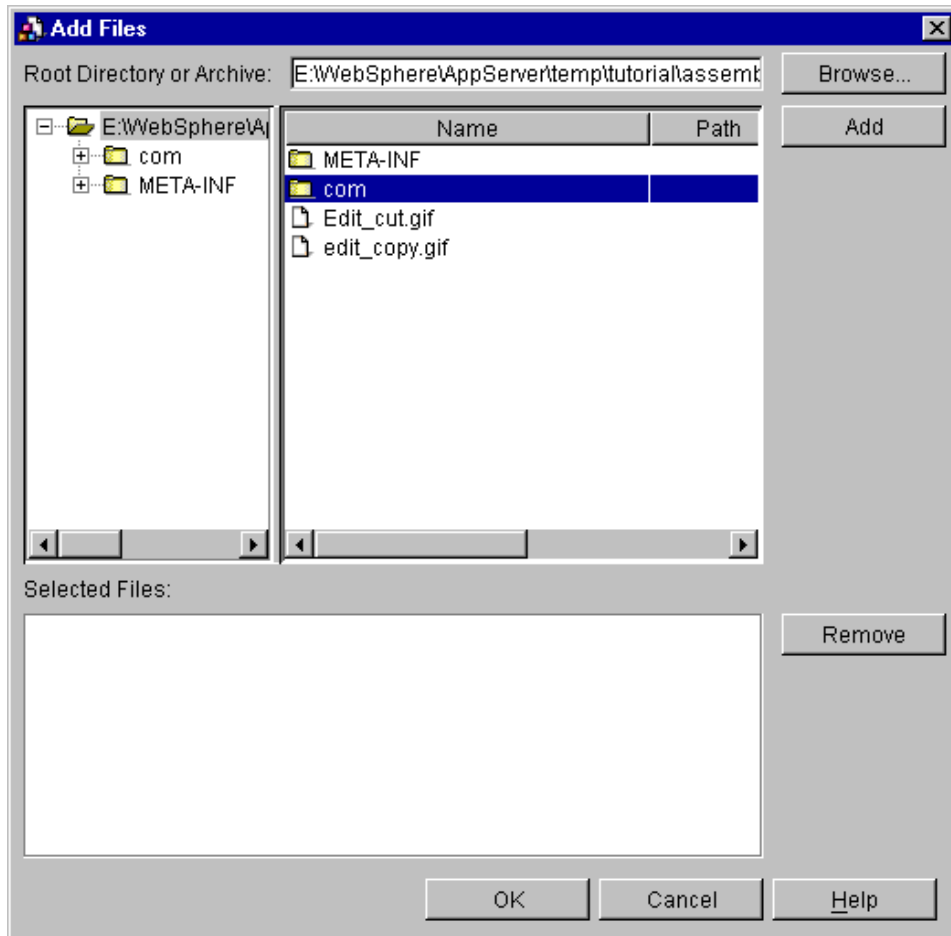
1. Create a new .war file by clicking **File -> Wizards -> Create Web Module Wizard**.
2. On the **Specifying Web Module Properties** panel, enter the following:
  - a. Display Name: **SimpleSessionWar**
  - b. File Name: **simpleSession.war**



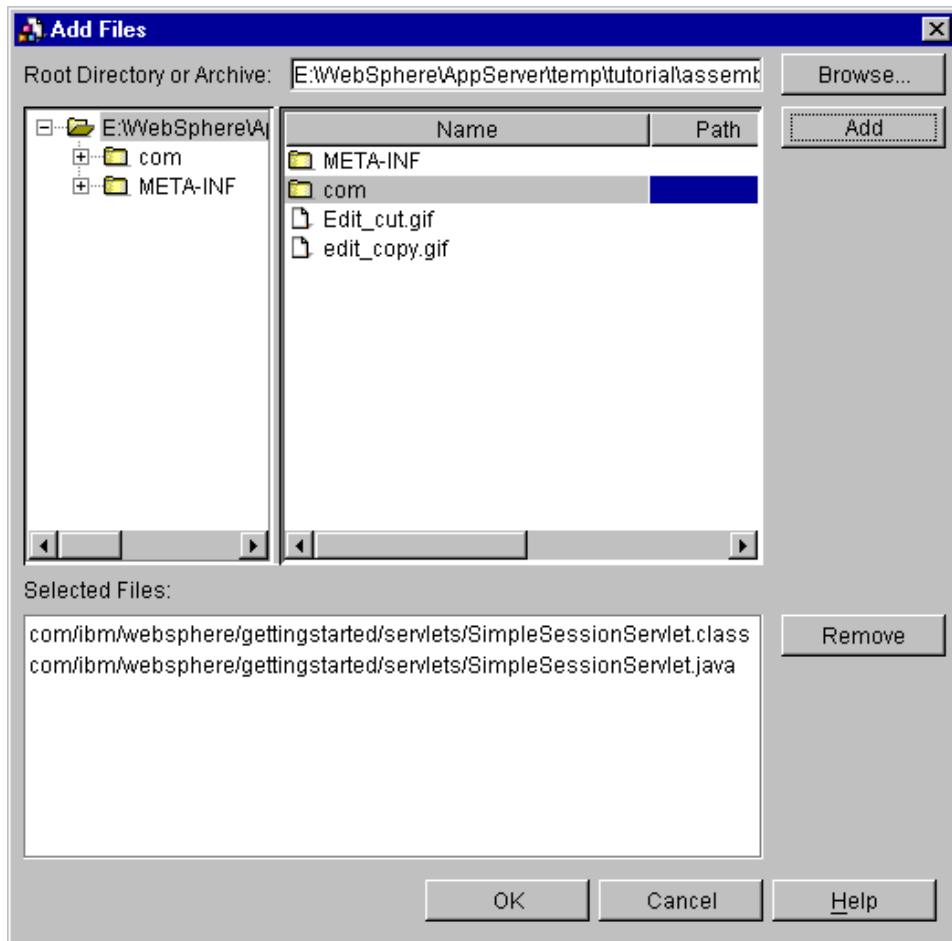
c. Click **Next**.

3. Click **Add Class Files...** on the **Adding Files** panel:

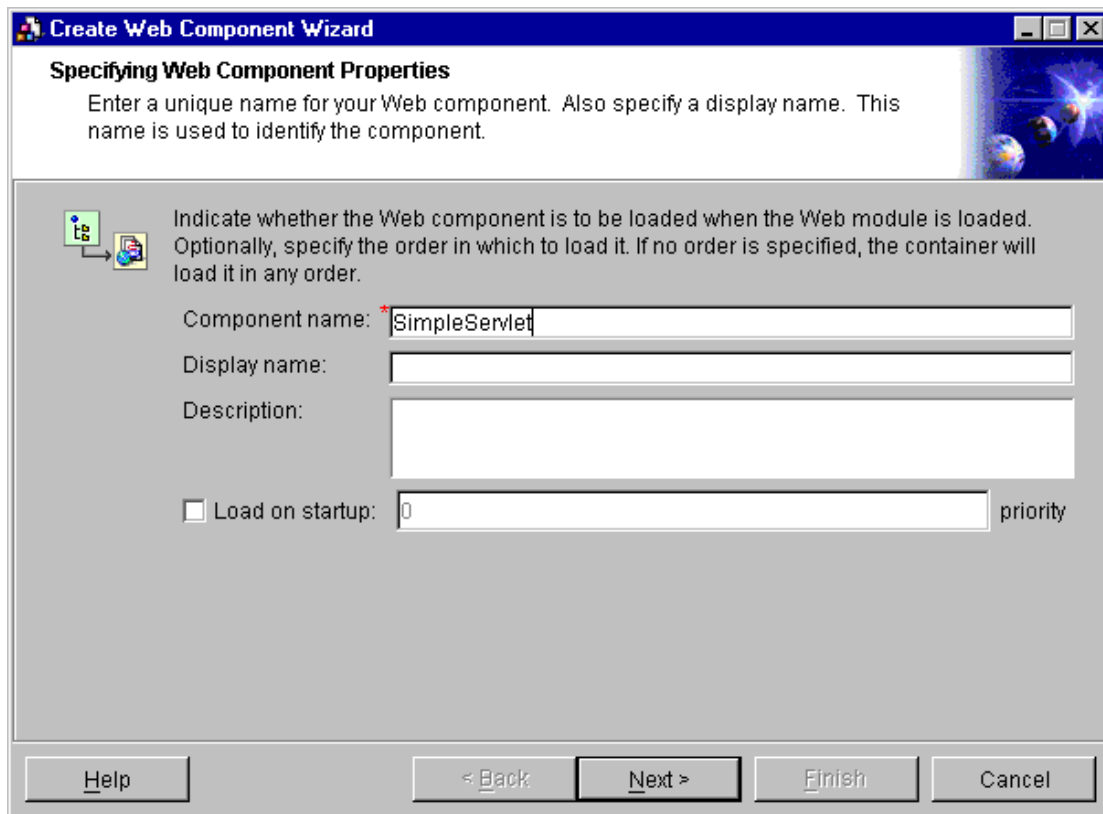
- Select **Browse**
- Navigate to the **servlet.jar** file, highlight the file and click **Select**.
- Select the **com** folder (in the right hand pane).



d. Click **Add** to display the files (in the com folder) in the lower part of the dialog.



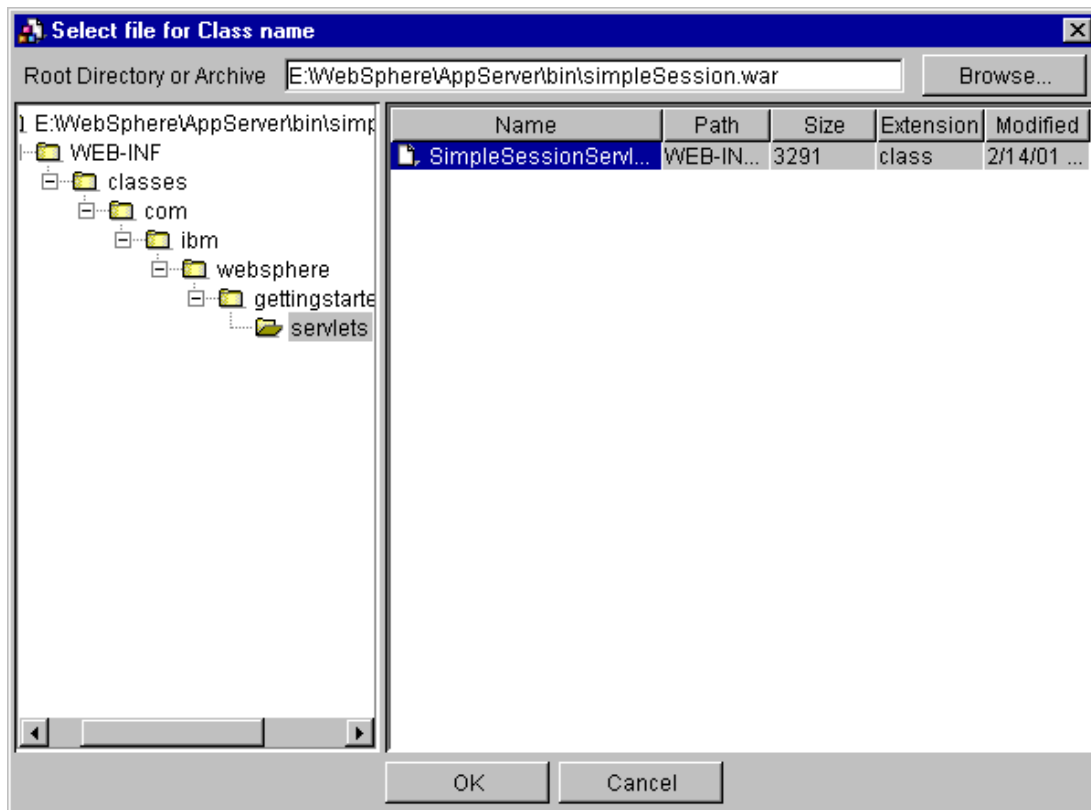
- e. Click **OK**.
- f. Click **Next**.
- 4. Click **Next** on the **Specifying Optional Web Module Properties** settings panel.
- 5. Click **Next** on the **Choosing Web Module Icons** settings panel.
- 6. Click **New...** on the **Adding Web Components** panel to start the **Create Web Component Wizard**:
  - a. On the **Specifying Web Component Properties** panel:
    - 1. Enter the Component Name: **SimpleServlet**



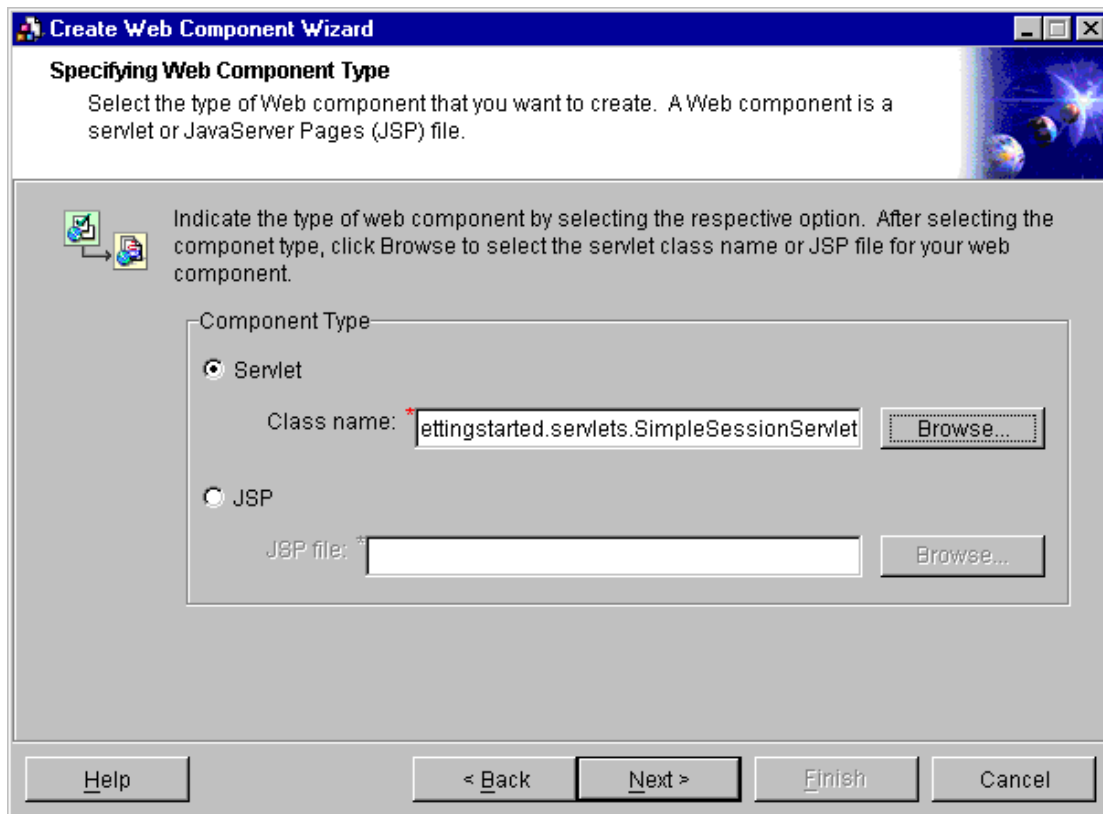
b. Select **Next**.

c. On the **Specifying Web Component Type** panel:

1. Set the **Component Type** to Servlet.
2. Click **Browse**.
3. Expand the tree view as far as it will go and select **servlets**.
4. Select **SimpleSessionServlet.class** in the right pane.



5. Click **OK**.

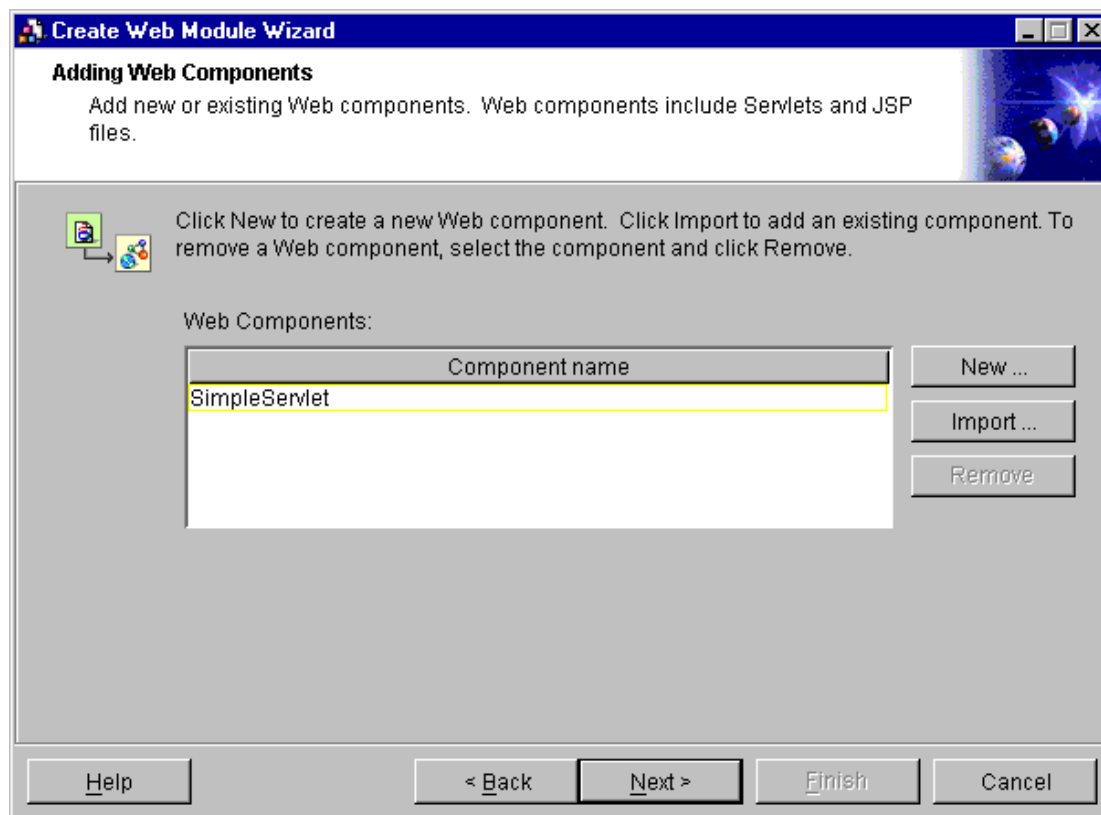


6. Click **Next**

d. Click **Next** on the **Choosing Web Components Icon** panel.

e. Click **Next** on the **Adding Security Role References** panel.

f. Click **Finish** on the **Adding Initialization Parameters** panel. This will return you to the **Create Web Module** wizard.



g. Click **Next**.

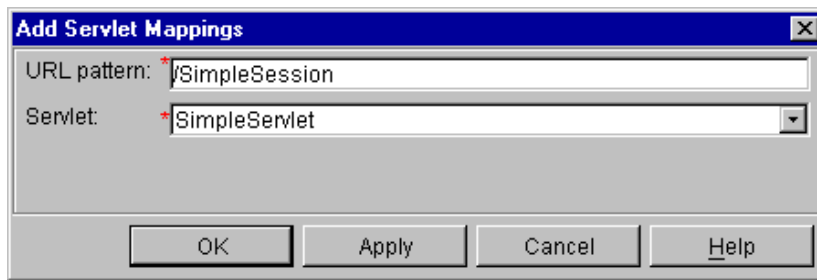
7. Click **Next** on the **Adding Security Roles** panel.

8. Click **Add** on the **Adding Servlet Mappings** panel.

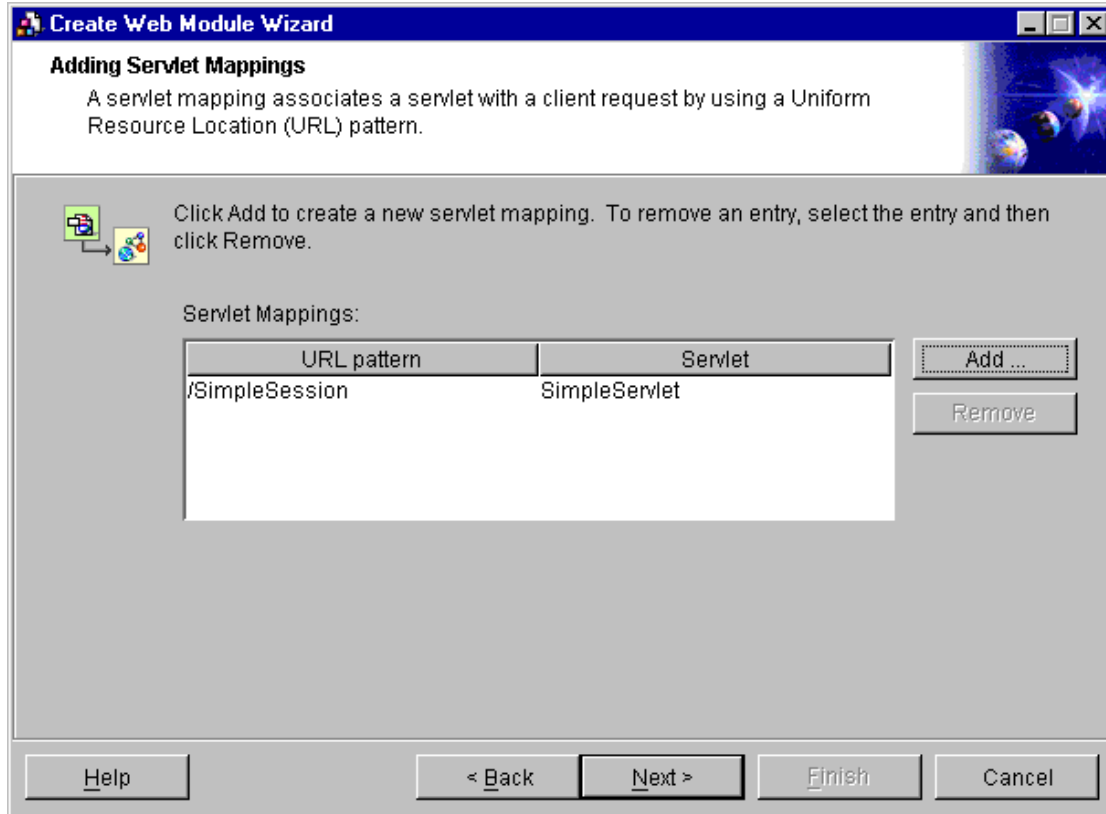
a. Specify a URL Pattern: **/SimpleSession**

b. Ensure that **SimpleServlet** is selected as the **Servlet**.





c. Click **OK**.



d. Click **Next**.

9. Click **Next** at the **Adding Resource References** panel.
10. Click **Next** at the **Adding Context Parameters** panel.
11. Click **Next** at the **Adding Error Pages** panel.
12. Click **Next** at the **Adding MIME Mappings** panel.
13. Click **Next** at the **Adding Tag Libraries** panel.
14. Click **Next** at the **Adding Welcome Files** panel.
15. Click **Add** at the **Adding EJB References** panel:
  - a. Enter the following:
    - Name: **ejb/ABean**
    - Home: **com.ibm.websphere.gettingstarted.ejbs.SimpleSessionHome**
    - Remote: **com.ibm.websphere.gettingstarted.ejbs.SimpleSession**
  - b. Ensure **Session** is selected as the **Type**.

**Add EJB References**

Name: \*

Description:

Link:

Home: \*

Remote: \*

Type: \*

c. Click **OK**.

**Create Web Module Wizard**

**Adding EJB References**

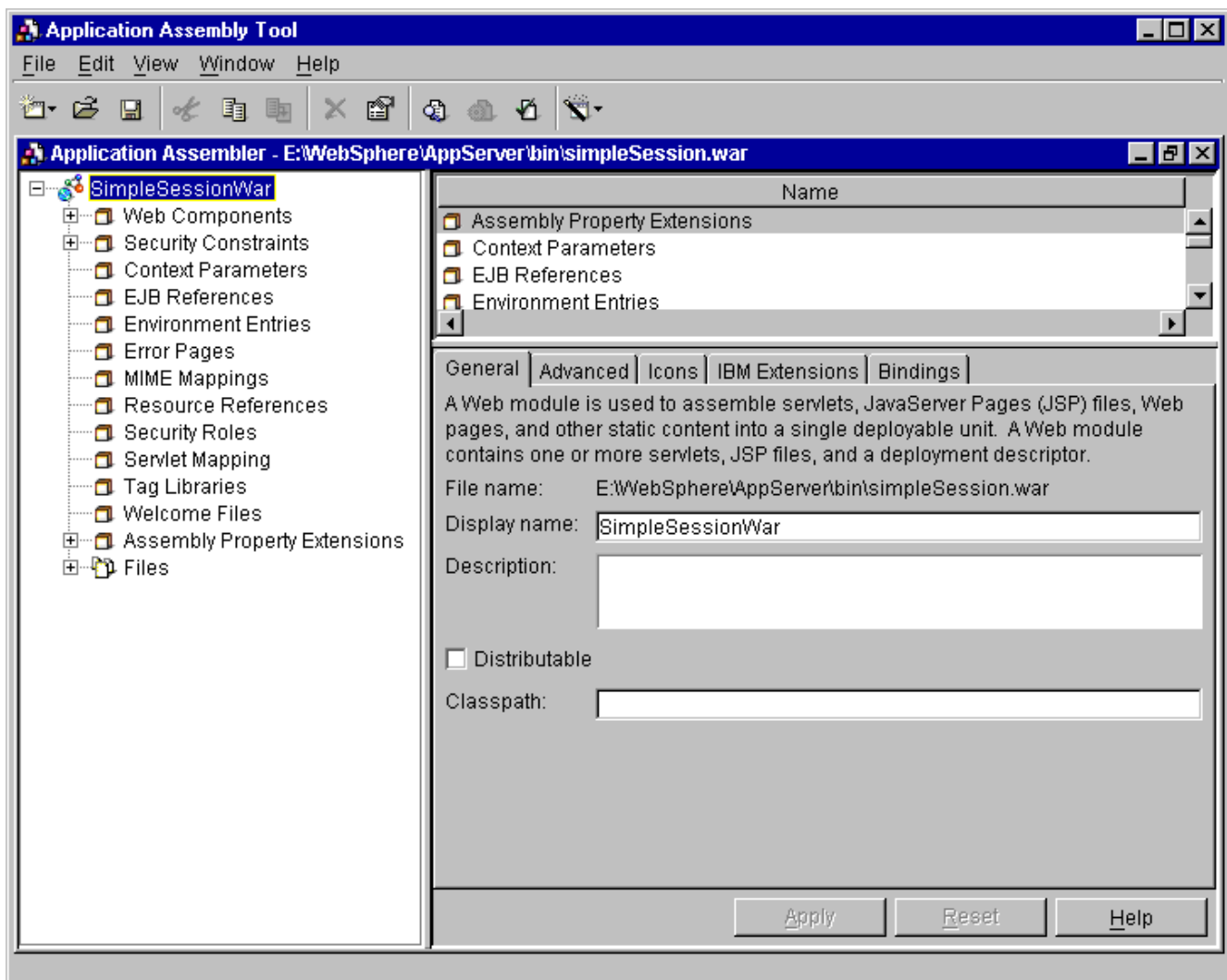
An EJB reference is a logical name used to locate the home interface of an enterprise bean used by the application.

Click Add to create a new EJB reference. To remove an entry, select the entry and then click Remove.

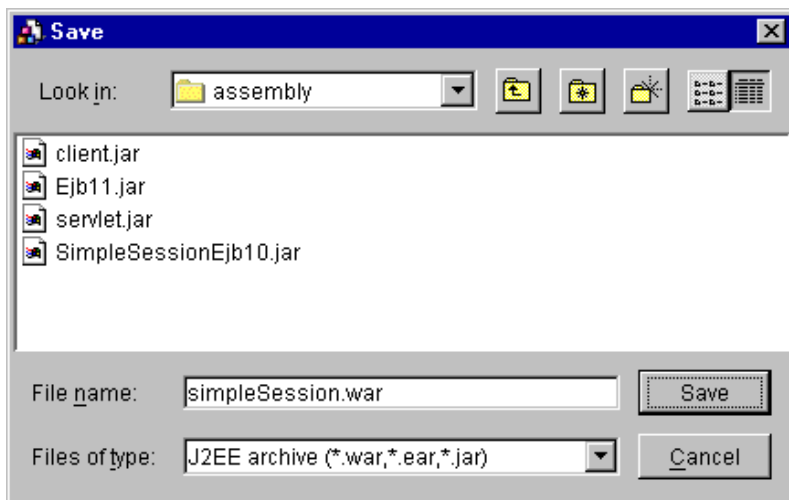
EJB References:

Name	Home	Link	Description
ejb/ABean	com.ibm.websphere.gettin...		

16. Click **Finish**. The Web module is displayed in the assembly tool interface.



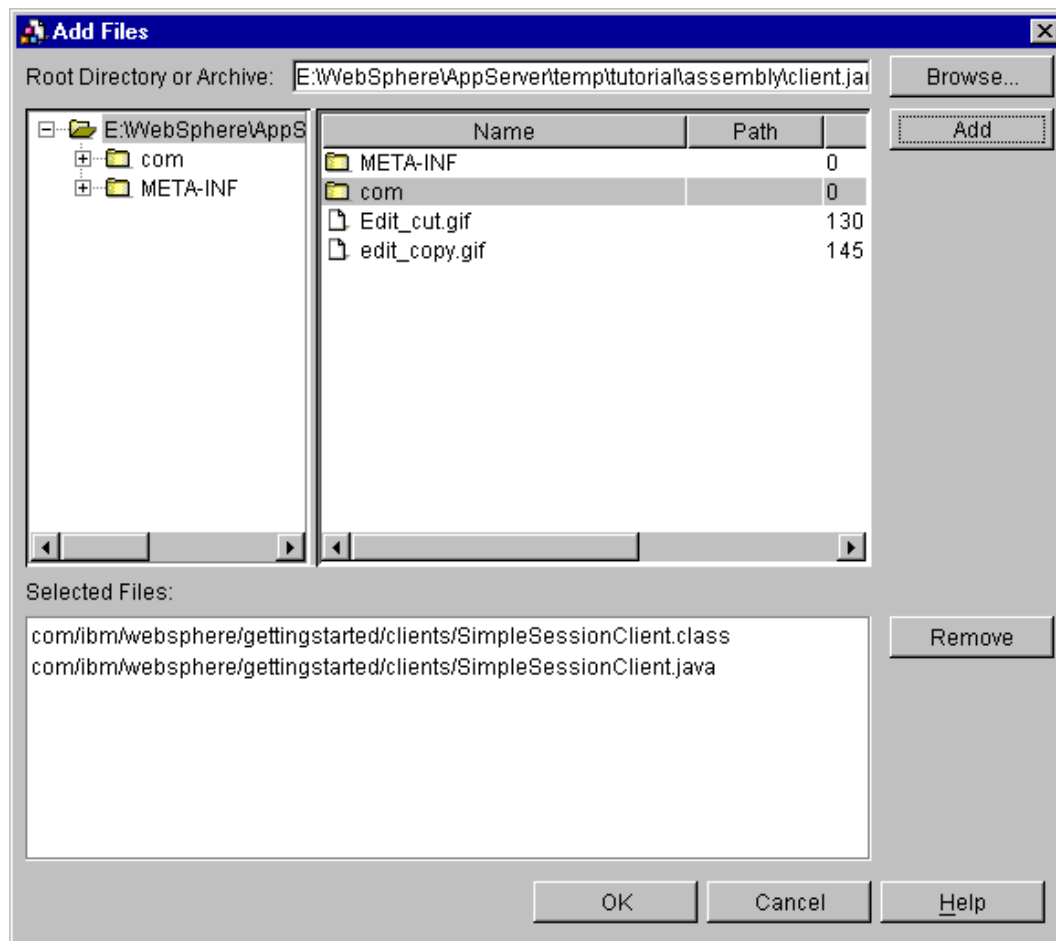
17. Save the .war file using **File -> Save As**. Name the file **simpleSession.war**.



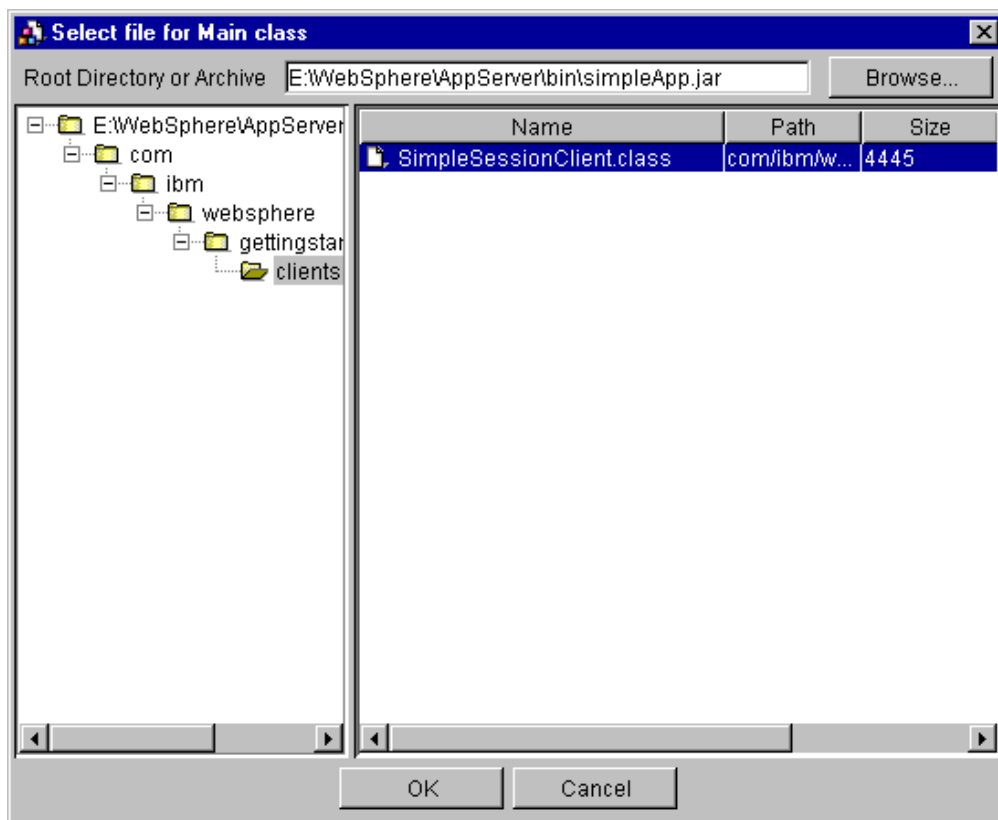
## Assemble an application client module

1. Create a new application client using **File -> Wizards -> Create Application Client Wizard**.
2. On the **Specifying Application Client Module Properties** panel, enter the following:
  - a. Display Name: **SimpleSessionClient**
  - b. File Name: **simpleApp.jar**
  - c. Click **Next**.
3. Click **Add...** on the **Adding Files** panel.
  - a. Select **Browse**.
  - b. Navigate to the **client.jar** file, highlight the file and click **Select**.

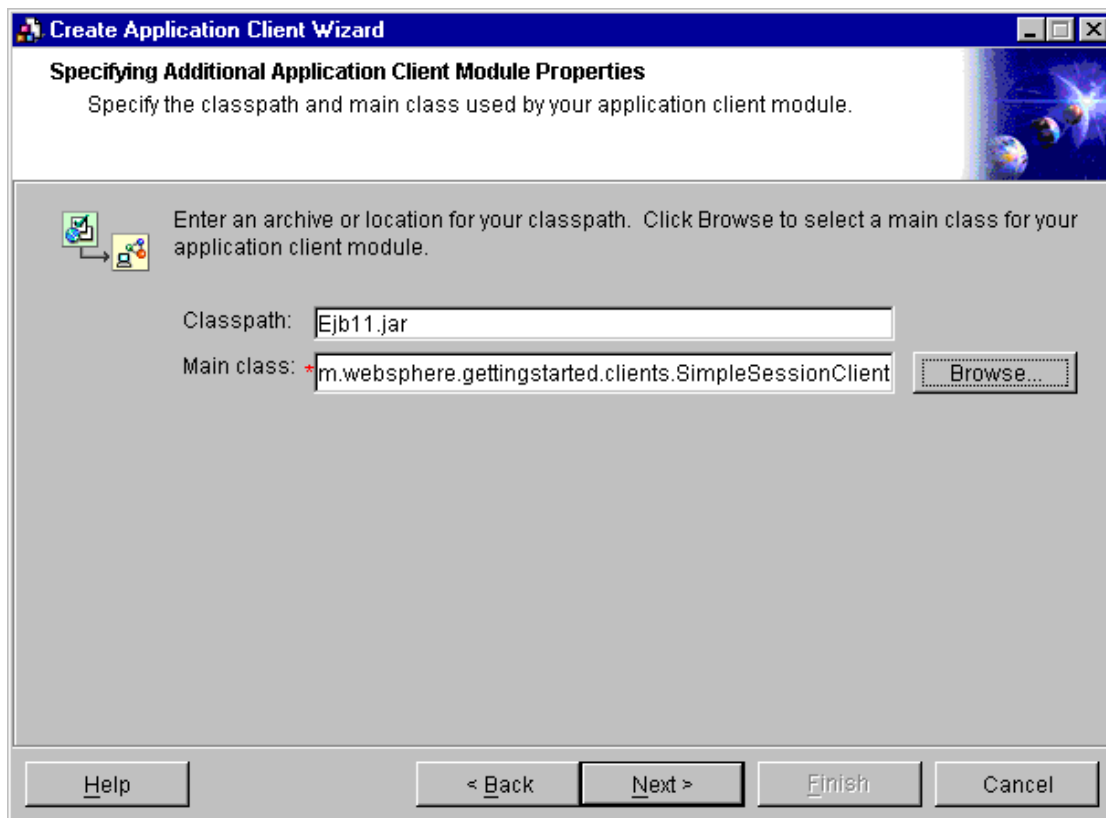
- c. Select the **com** folder (in the right hand pane).
- d. Click **Add**.



- e. Click **OK**.
  - f. Click **Next**.
4. On the **Specifying Additional Application Client Modules Properties** panel:
- a. Enter **Ejb11.jar** for **Classpath**.
  - b. Select **Browse** next to **Main Class** field.
  - c. Expand the tree view in the left pane as far as it goes and select **clients**.
  - d. Select **SimpleSessionClient.class** in the right panel.



e. Click **OK**.



f. Click **Next**.

5. Click **Next** on the **Choosing Application Client Modules** panel.

6. Click **Add** on the **Adding EJB References** panel.

a. Enter the following:

- Name: **ejb/SimpleSession**
- Home: **com.ibm.websphere.gettingstarted.ejbs.SimpleSessionHome**
- Remote: **com.ibm.websphere.gettingstarted.ejbs.SimpleSession**

**Add EJB References**

Name: \*ejb/SimpleSession

Description:

Link:

Home: \*here.gettingstarted.ejbs.SimpleSessionHome

Remote: \*websphere.gettingstarted.ejbs.SimpleSession

Type: \*Session

- b. Click **OK**.
- c. Verify that your EJB reference is displayed in the EJB References list.
7. Click **Next**.
8. Click **Next** on the **Adding Resource References** panel.
9. Click the **Finish** on the **Specifying Environment Entries** panel. The application client module is displayed in the Application Assembly Tool.

**Application Assembly Tool**

File Edit View Window Help

Application Assembler - E:\WebSphere\AppServer\bin\simpleApp.jar

SimpleSessionClient

- EJB References
- Resource References
- Environment Entries
- Files

EJB References

Environment Entries

Files

Resource References

General | Icons

An application client is a standalone Java program (as opposed to a Web browser program).

File name: E:\WebSphere\AppServer\bin\simpleApp.jar

Display name: \*SimpleSessionClient

Description:

Classpath: Ejb11.jar

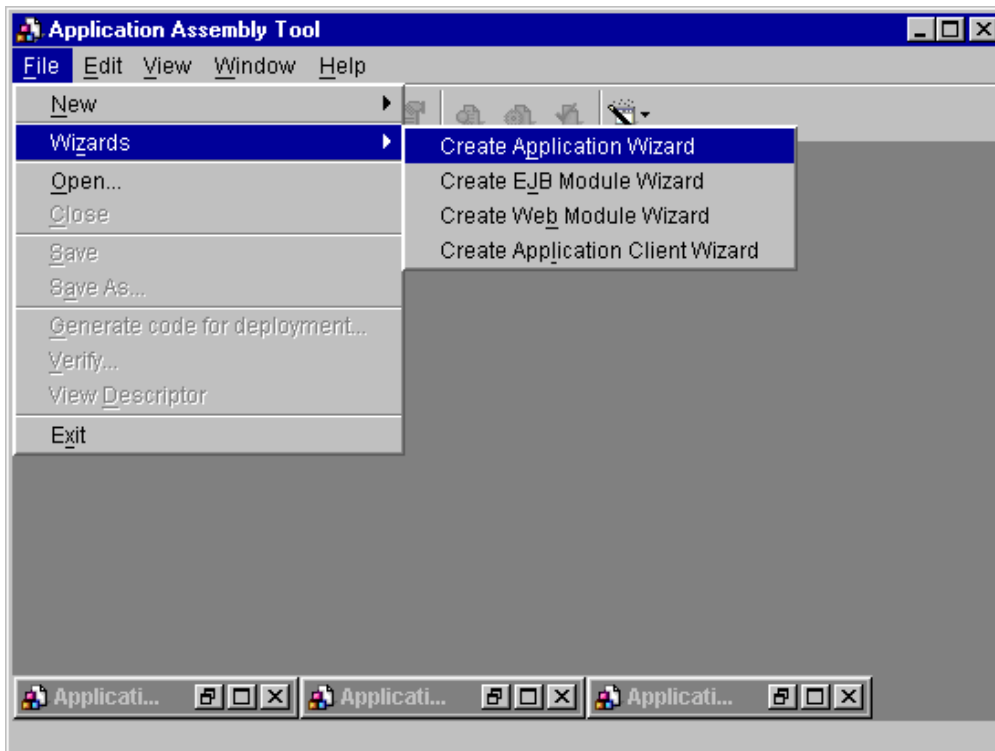
Main class: \*h.websphere.gettingstarted.clients.SimpleSessionClient



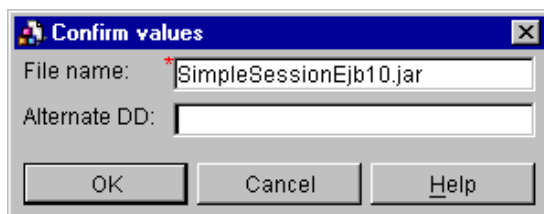
10. Save the application client jar file by clicking **File -> Save As**. Name it **simpleApp.jar**.

## Assemble a J2EE application

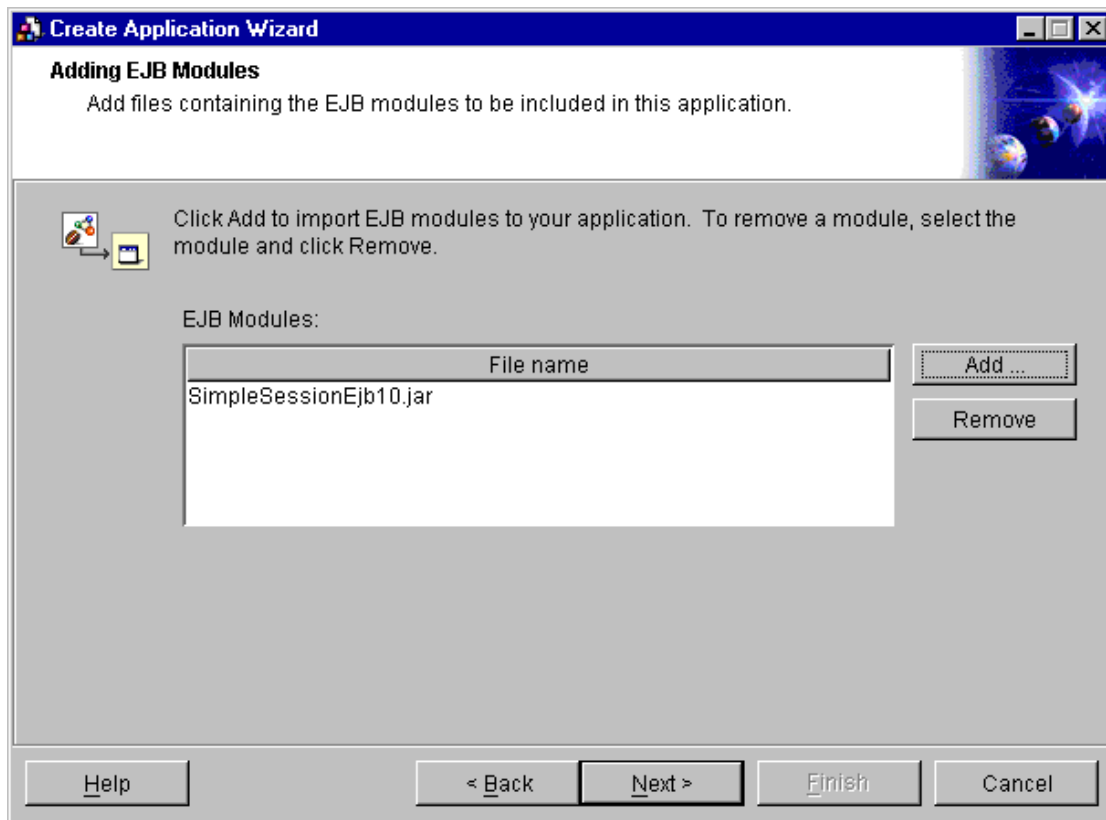
1. Using AAT, create an .ear by clicking **File -> Wizards -> Create Application Wizard**.



2. On the **Specifying Application Properties** panel:
  - a. Enter the following:
    - Display Name: **SimpleSessionApp**
    - File Name: **simpleSession.ear**
  - b. Click **Next**
3. Click **Next** on the **Adding Supplementary Files** panel.
4. Click **Next** on the **Choosing Application Icons** panel.
5. Click **Add...** on the **Adding EJB Modules** panel.
  - a. Navigate to your **Ejb11.jar** file, highlight the file and click **Open**.
  - b. Click **OK** on the resulting **Confirm values** dialog.



The EJB module is now listed on the **Adding EJB Modules** panel.



c. Click **Next**.

6. Click **Add...** on the **Adding Web Modules** panel.

a. Navigate to your **simpleSession.war** file, highlight the file and click **Open**.

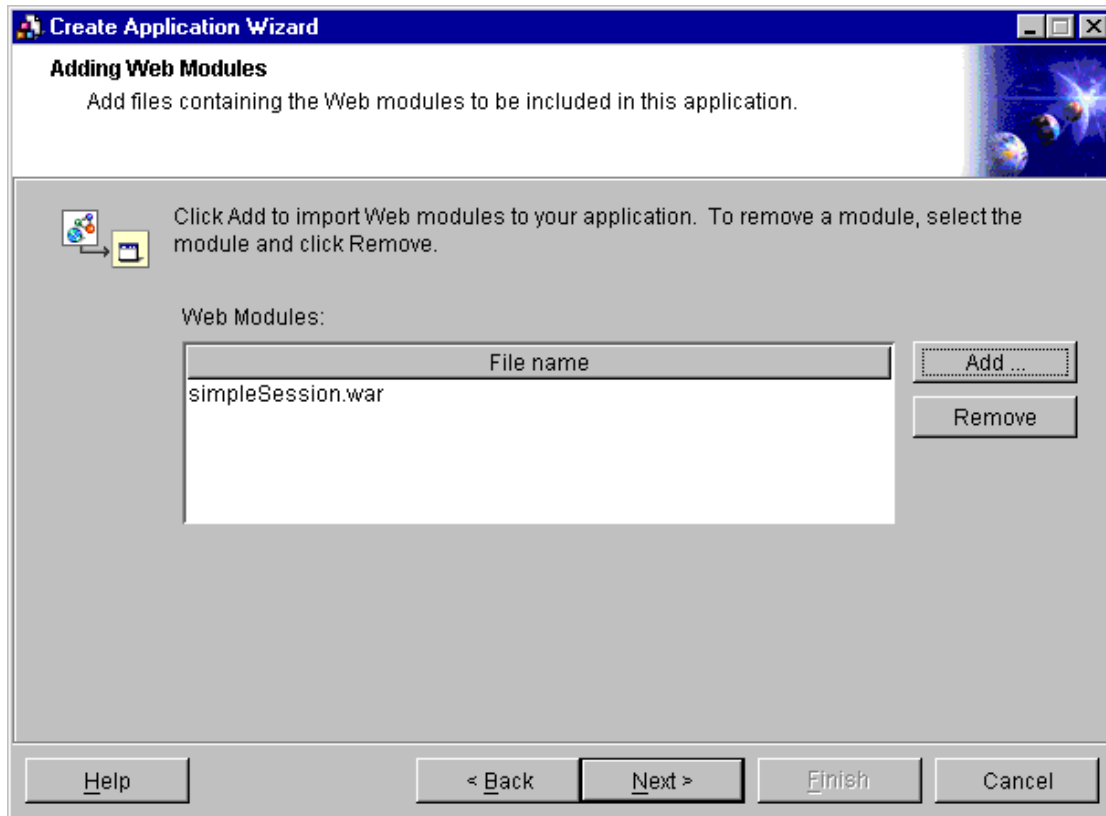
b. In the resulting dialog, enter a Context Root: **/gettingstarted3**



c. Click **OK**.

The Web module is now listed on the **Adding Web Modules** panel.

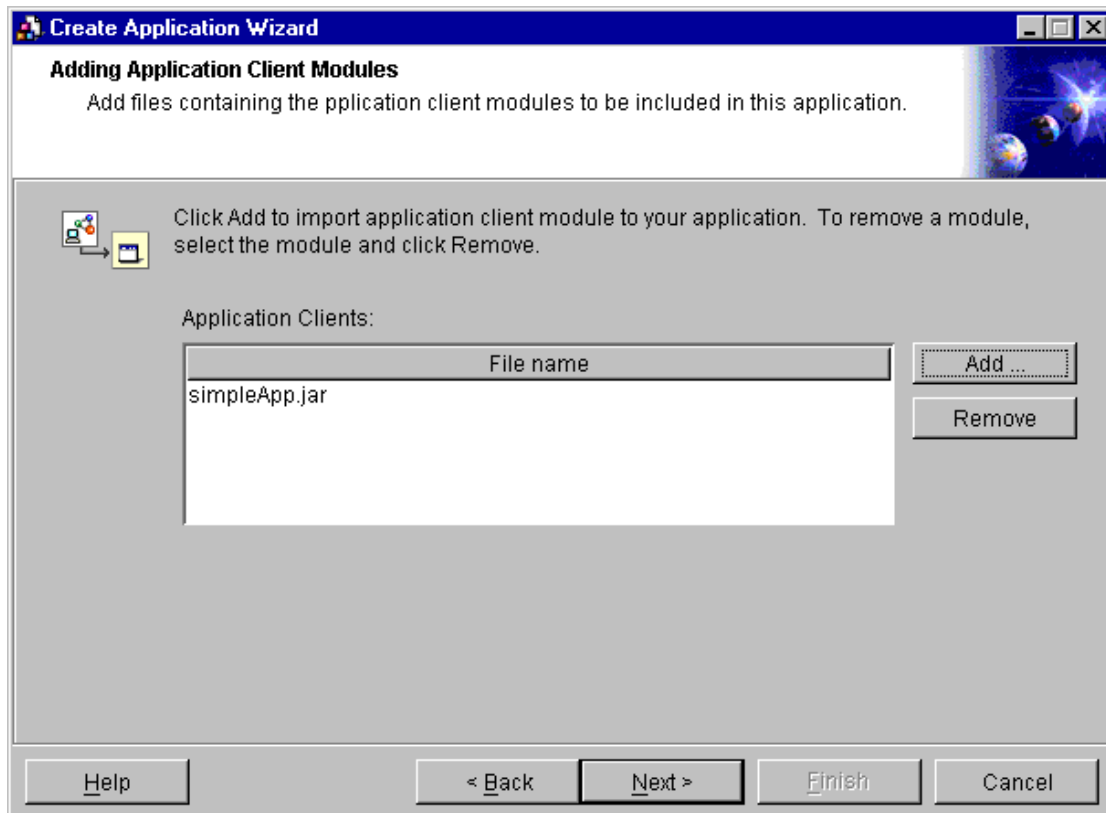




d. Click **Next**.

7. Click **Add...** on the **Adding Application Client Modules** panel.
  - a. Navigate to your **simpleApp.jar** file, highlight the file and click **Open**.
  - b. Click **OK** on the resulting dialog.

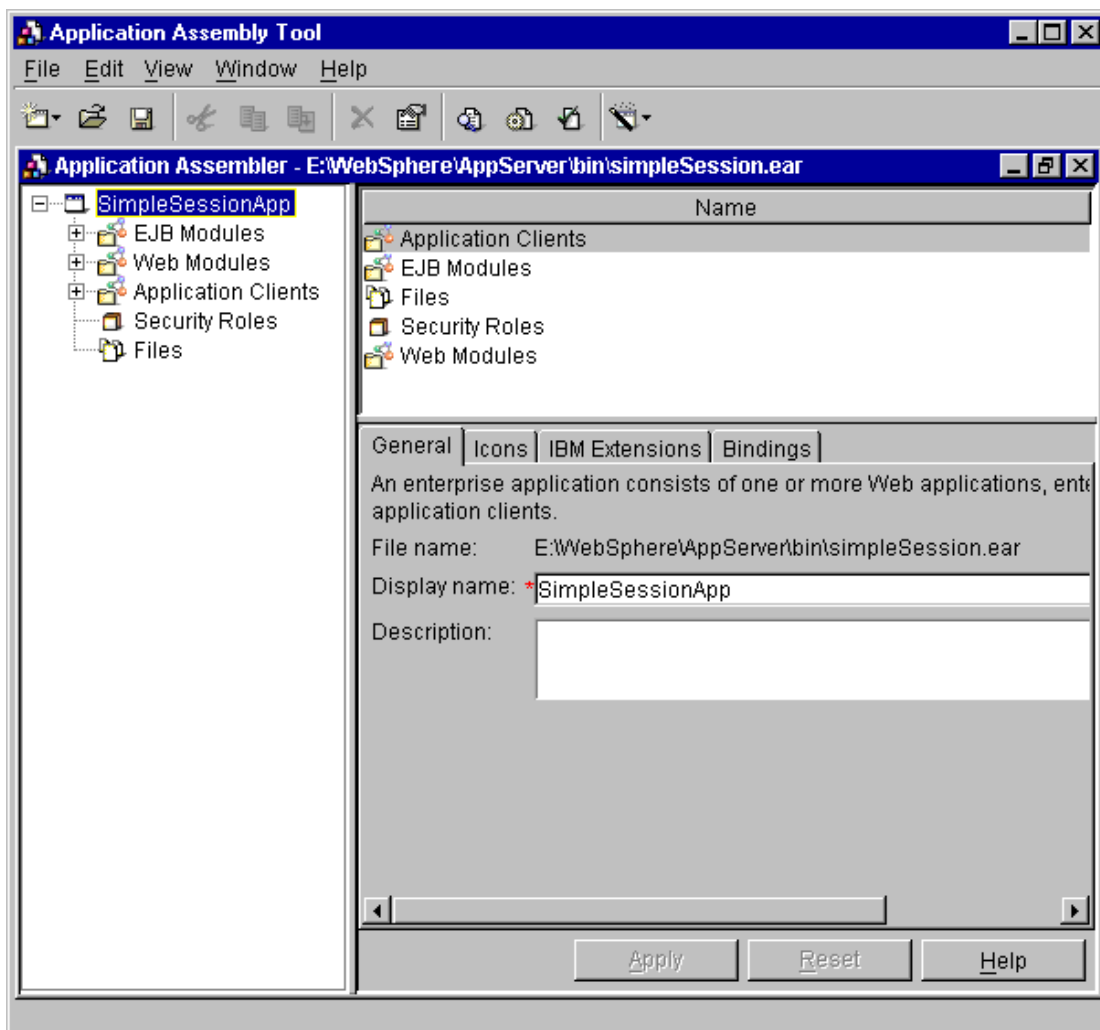
The application client module is now listed on the **Adding Application Client Modules** file.



c. Click **Next**.

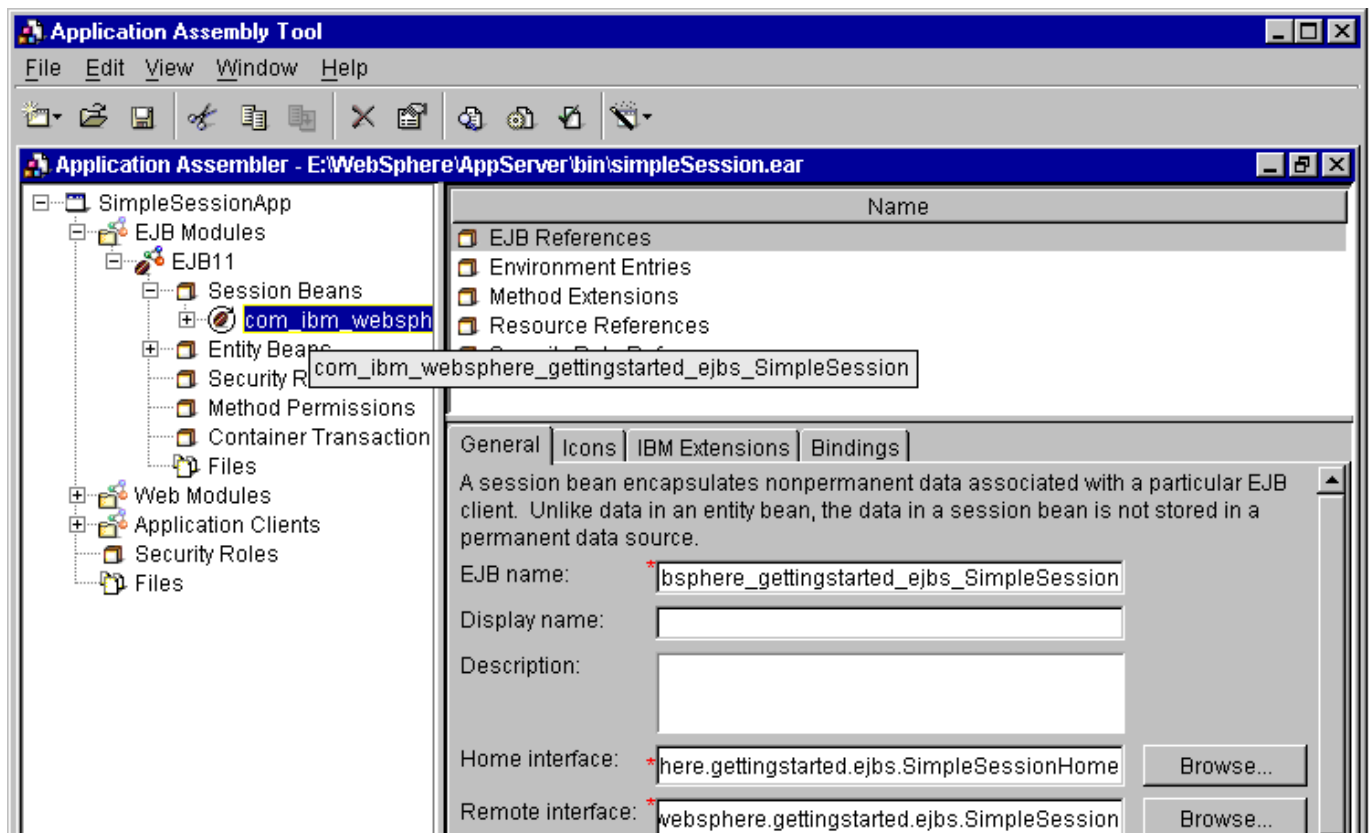
8. Click **Finish** on the **Adding Security Roles** panel.

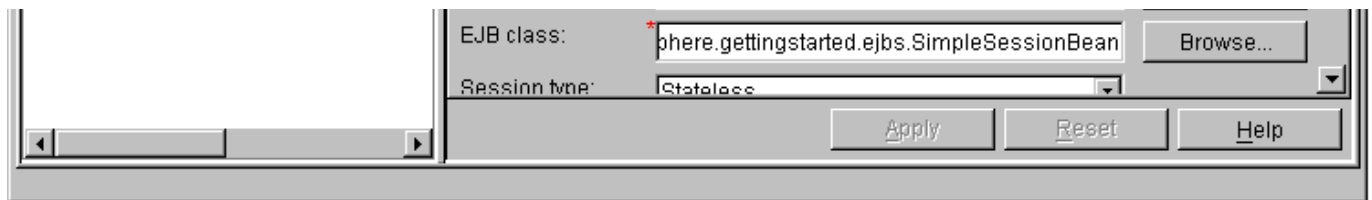
The application is now displayed in the Application Assembly Tool.



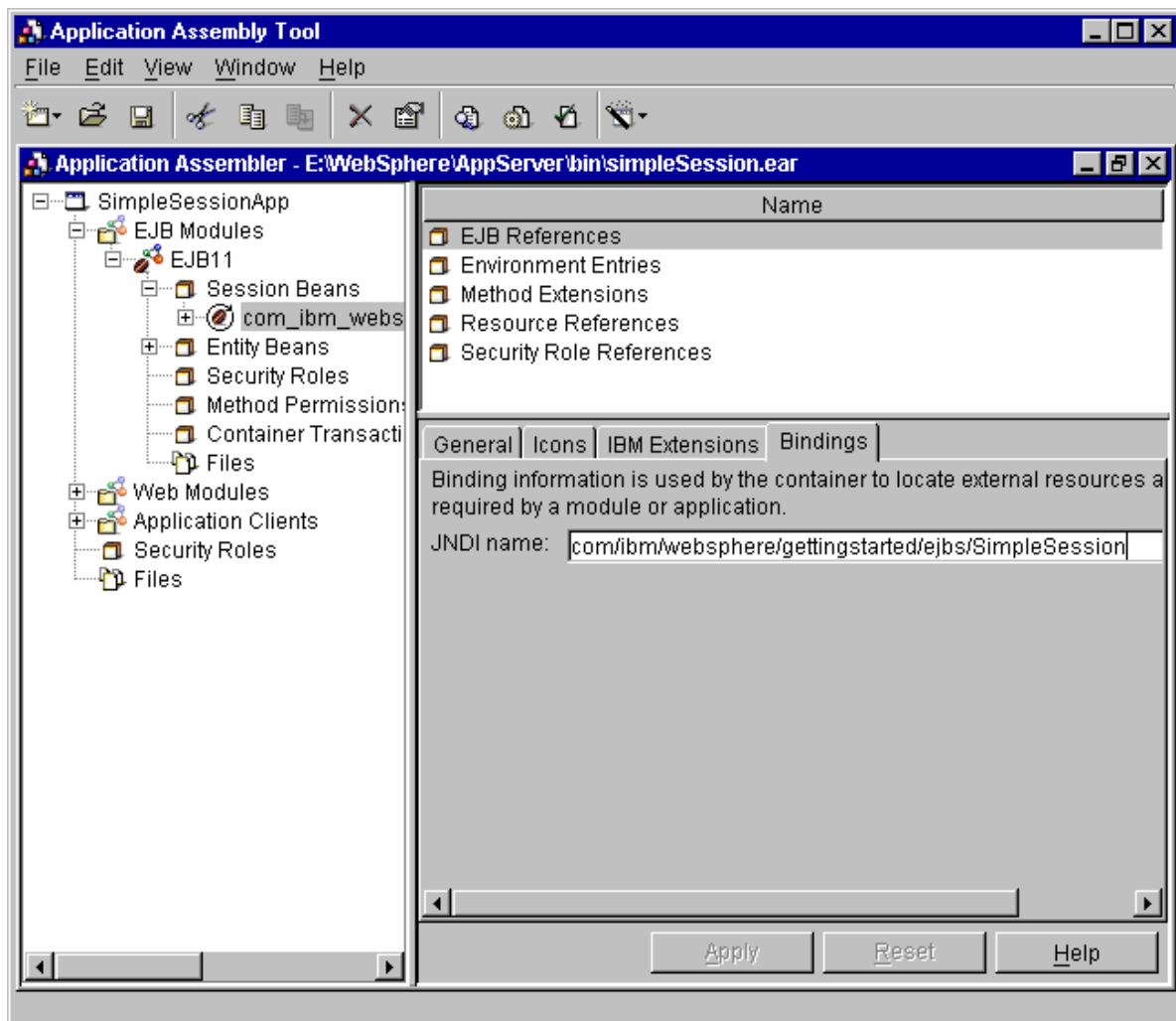
9. Bind the EJB and references to it:

- In the assembly tool tree view, expand **EJB Modules** -> **EJB11** -> **Session Beans** -> **com\_ibm...ejbs\_SimpleSession**.
- Click it to display its properties.

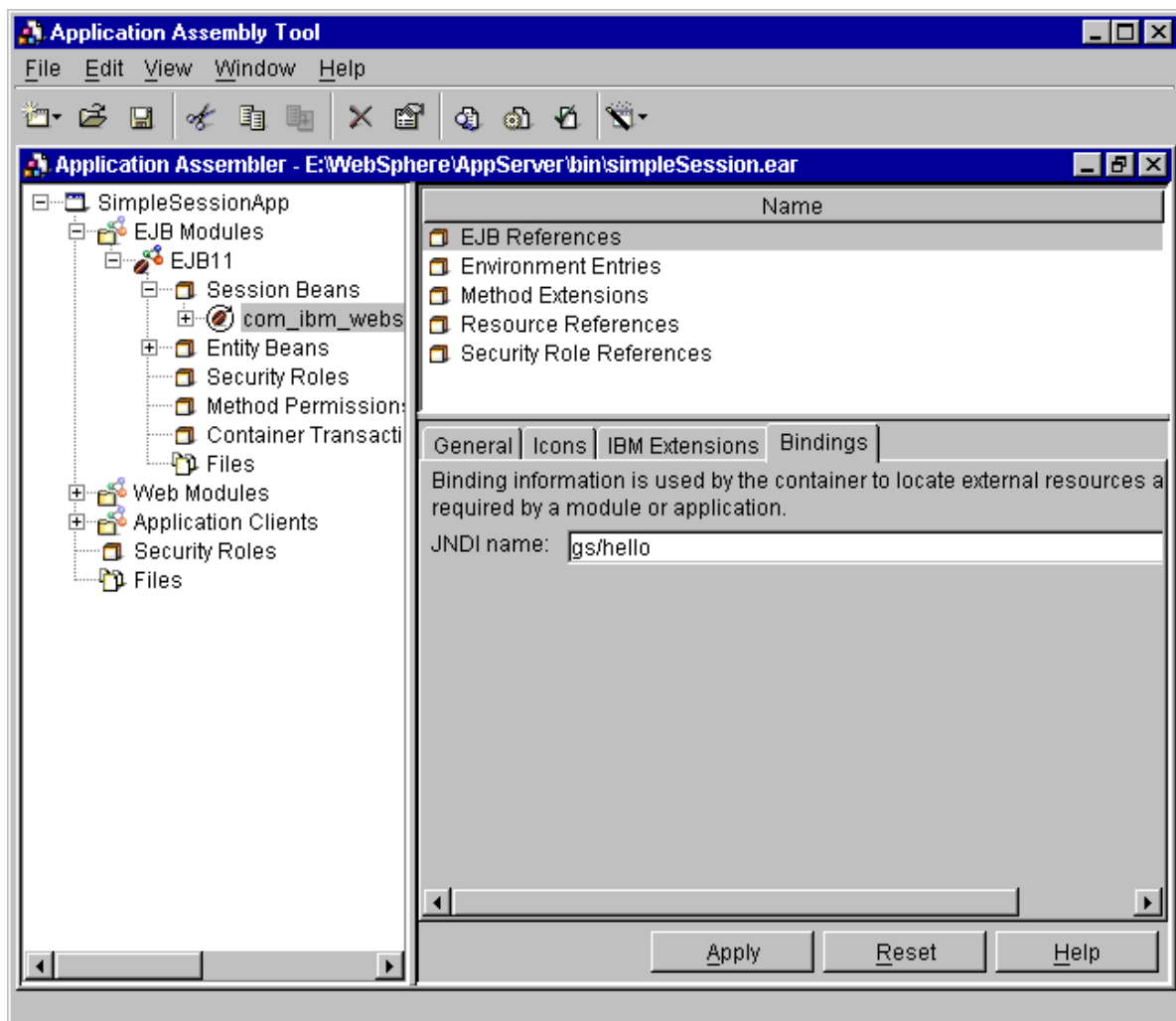




c. Select the **Bindings** tab.



d. Replace the existing text string with **gs/hello** as the global JNDI name of the session bean.



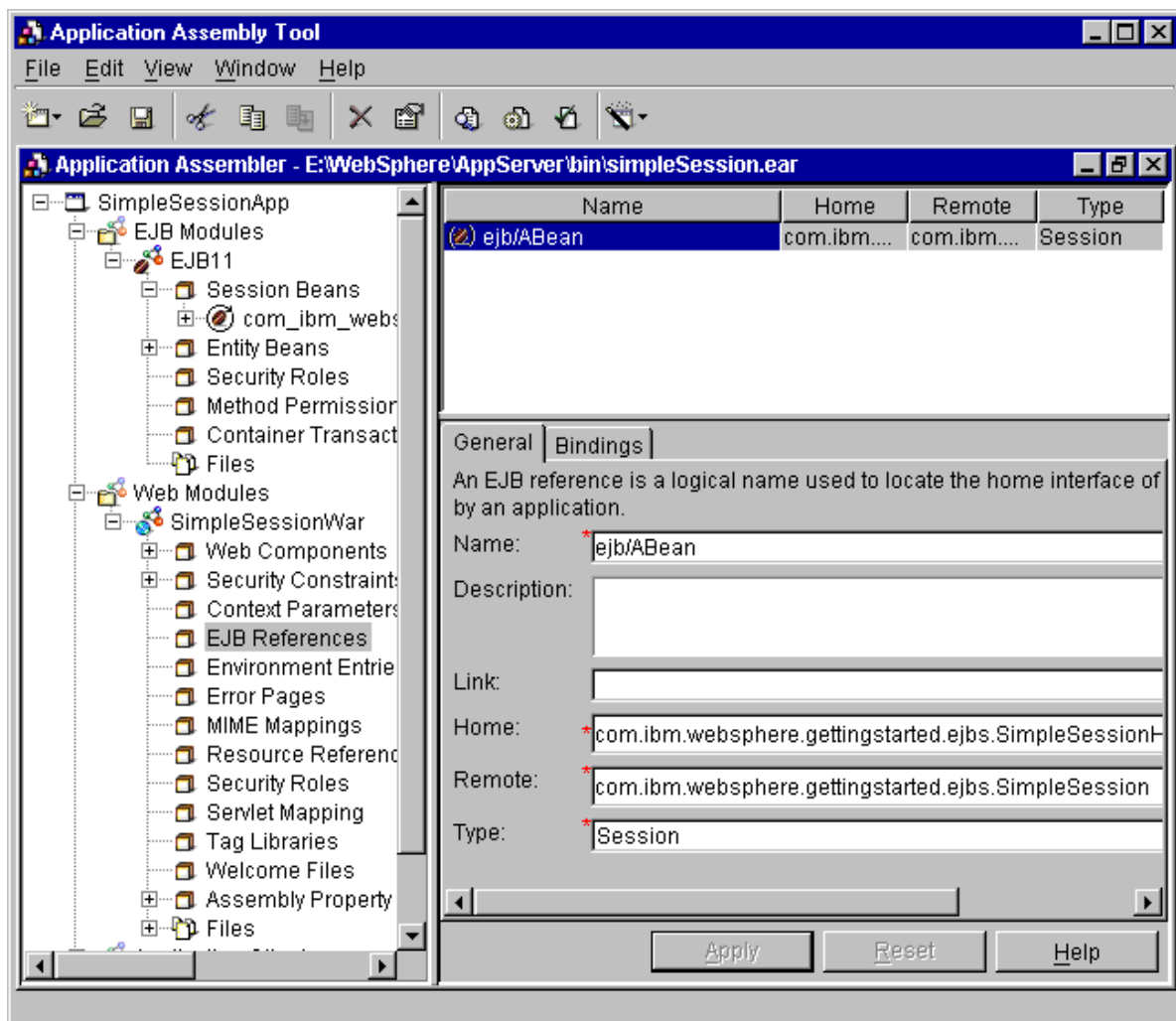
Although this example uses a short name for the session bean, consider using longer names in actual practice. A long name will help ensure that the name is unique in the global name space. An example of a suitable long name is:

`com/ibm/websphere/gettingstarted/ejbs/SimpleSessionHome`

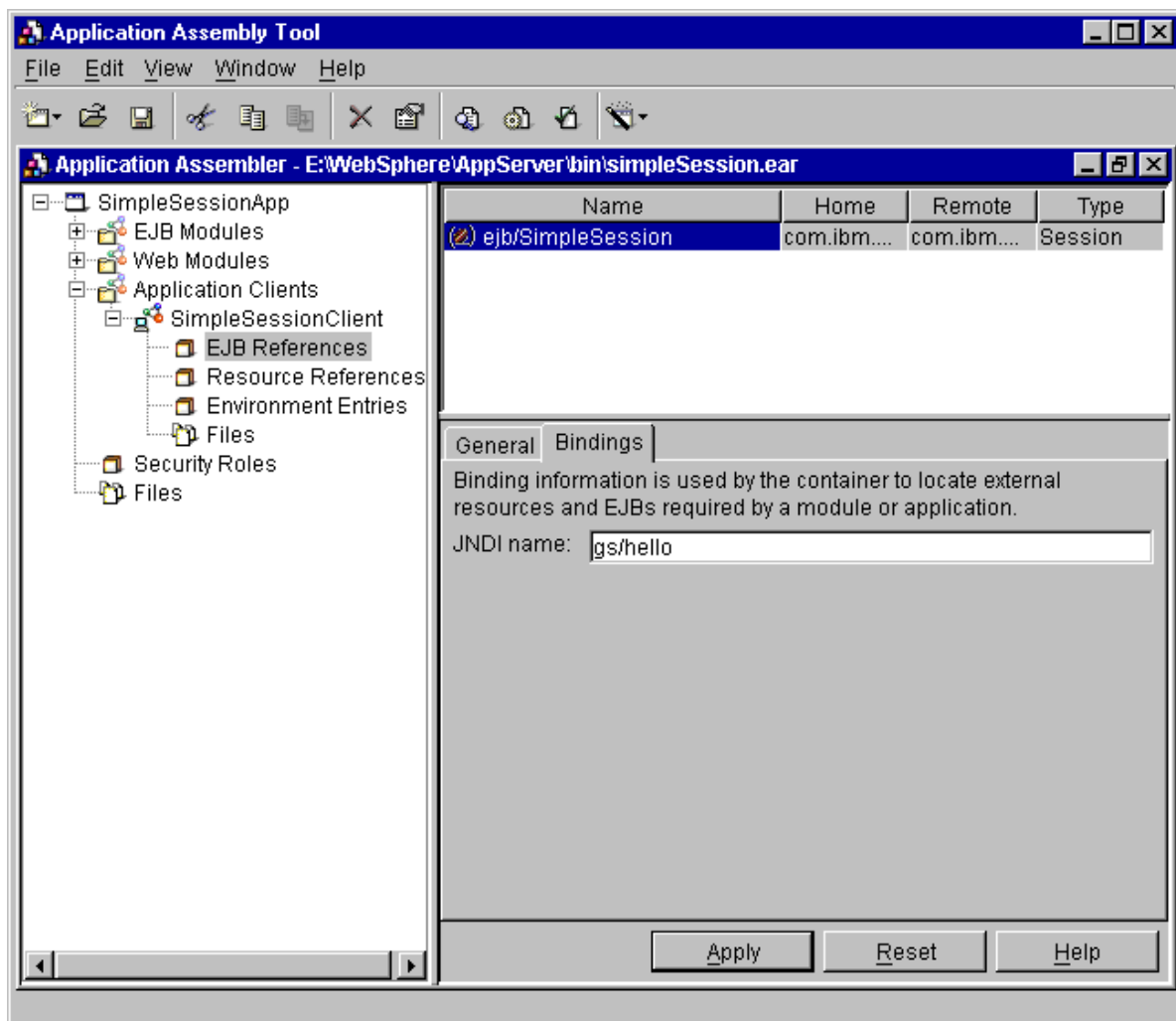
e. Click **Apply**.

10. Bind the EJB reference (java:comp/env/ejb/ABean) to the global JNDI name of the bean:

- a. Expand **Web Modules** -> **SimpleSessionWar**.
- b. Select **EJB References**.
- c. Select in the right pane the particular reference called **ejb/ABean**.

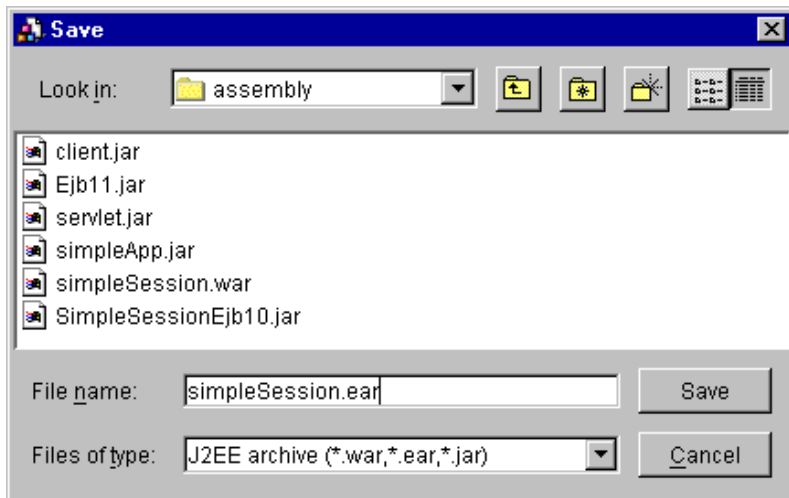


- d. Select the **Bindings** tab.
  - e. Replace the existing text string with **gs/hello**. This is the JNDI Name that you assigned to the enterprise bean in an earlier step.
  - f. Click **Apply**.
11. Bind the EJB reference:
- a. Expand **Application Clients** -> **SimpleSessionClient**.
  - b. Select **EJB References**.
  - c. Select in the right pane the particular reference to your session bean: **ejb/SimpleSession**
  - d. Select the **Bindings** tab.
  - e. Replace the existing text string with **gs/hello**. This is the JNDI Name that you assigned to the enterprise bean in an earlier step.



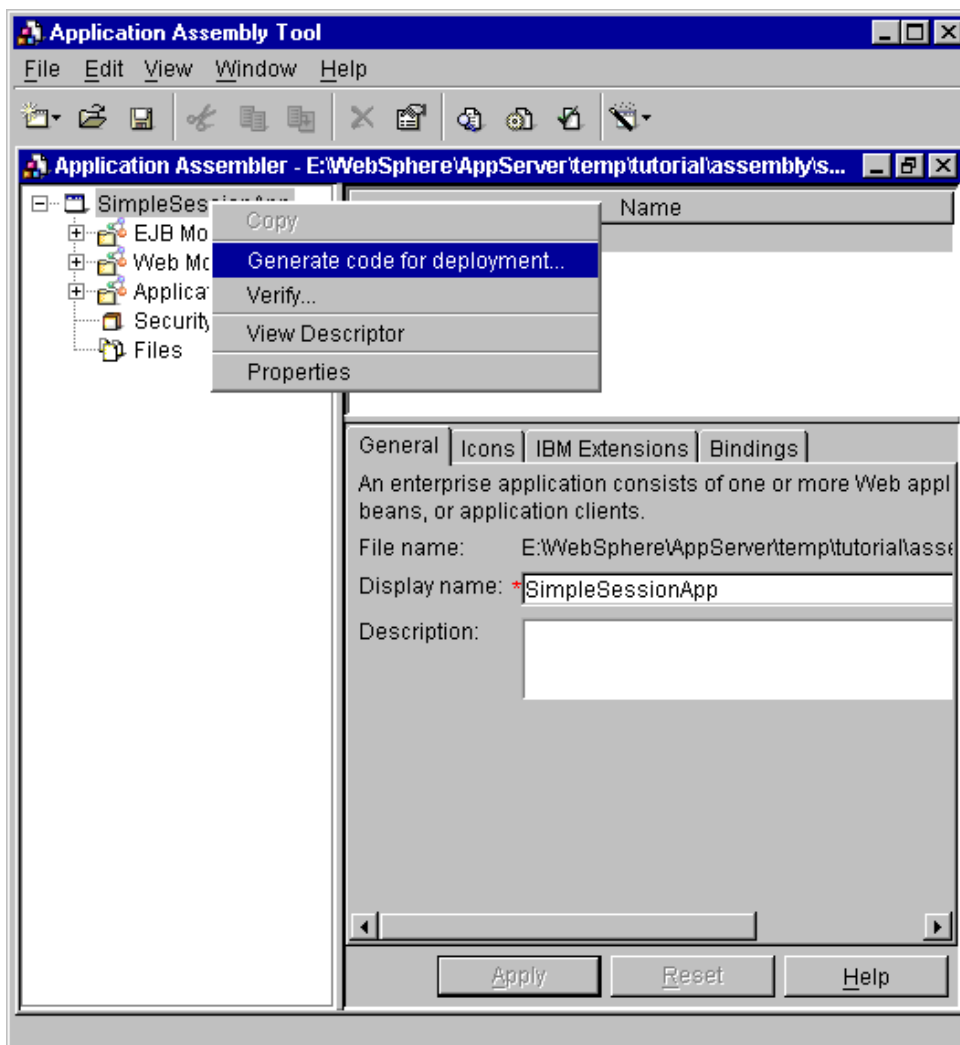
f. Click **Apply**.

12. Save the .ear file by clicking **File -> Save As**. Name it **simpleSession.ear**.

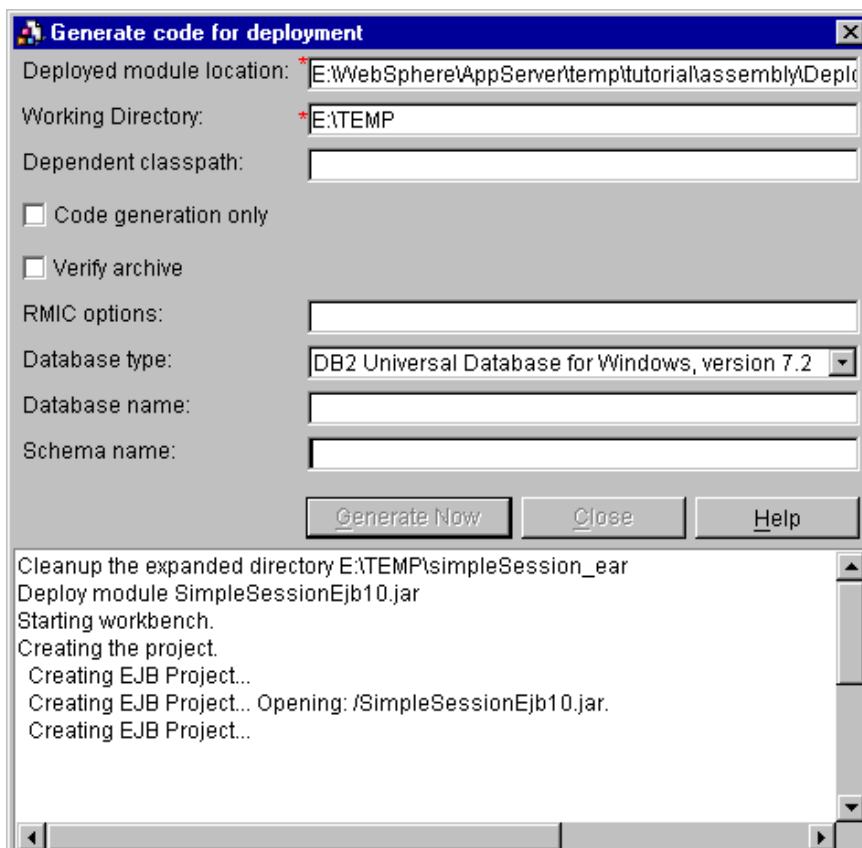


## Generate code for deployment

- Right-click the .ear file by selecting the top entry in the tree view of the assembly tool.
- Select **Generate Code for Deployment**.




- c. Select appropriate **Database Type**.
- d. Click **Generate Now**.



---

e. When deployment is complete, close this window.

 Another way to generate code for deployment is to use the [Deployment Tool](#) directly. The Application Assembly Tool calls the command line Deployment Tool for you when you use the **Generate Now** button and accompanying dialog.

## Exit the Application Assembly Tool

Exit the assembly tool using the **File** -> **Exit** option.

## What's next?

Now that you have your application assembled, you can deploy it on the application server, which includes "installing" it. The next steps are outlined by the [Application deployment tutorial](#).



## 6.7.2: Application deployment tutorial

During this tutorial, you will deploy the J2EE application that you assembled during the [Application assembly tutorial](#). Deployment involves configuring the server and supporting resources (you will use the existing default server), and installing the application. Installing the application means configuring the settings pertaining to this application in the server runtime. The runtime settings include choices such as whether to automatically load servlets, or to precompile JSP files.

### Prerequisites



Before performing this tutorial, either:

- Complete the [Application assembly tutorial](#)
- Plan to use the already assembled "shortcut\_simpleSession.ear" file described later in this tutorial

### Overview of steps (requires 15 to 30 minutes)

1. [Obtain the tutorial application](#)
2. [Start the application server](#)
3. [Open the administrative console](#)
4. [Prepare the application server and needed resources](#)
5. [Install the application](#)
6. [Regenerate the Web server plug-in and save the server configuration](#)
7. [Stop the server and start it again](#)

### Paths through the tutorial

- **Quick path:** If your objective is to practice these steps as quickly as possible, skip the explanations marked with the  graphic.
- **Maximum learning path:** If your objective is to understand and explore the product capabilities, read the  notes and browse the links they provide to additional documentation.

### Obtain the tutorial application

If you completed the [Application assembly tutorial](#), you simply need to know the location of the simpleSession.ear file that you assembled, including generating code for deployment. It should be located in the directory:

```
product_installation_root/temp/tutorial/assembly/Deployed_simpleSession.ear
```

If you did not perform the assembly tutorial, and thus plan to use the already assembled .ear file:

1. Create a directory named "tutorial" under the path:

```
product_installation_root/temp
```

2. [Click here](#) to access the .zip file containing the tutorial application components.
3. Save the .zip file to

```
product_installation_root/temp/tutorial
```

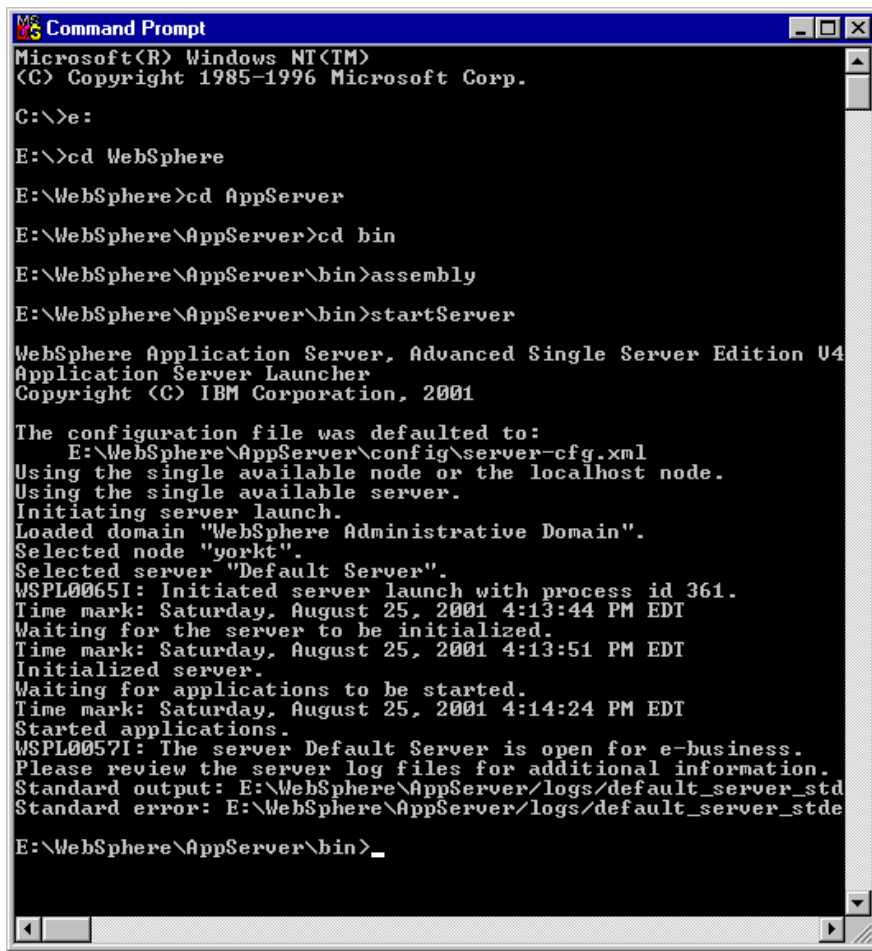
4. Use your favorite .zip or .jar utility to extract the tutorial.zip content into the tutorial directory.
5. Copy the shortcut\_Deployed\_simpleSession.ear file. Name the copy Deployed\_simpleSession.ear.

### Start the application server

First, start the product using:

```
<product_installation_root>/startServer
```

Leave the command window open, as you will use it again later to start the server.



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>e:

E:\>cd WebSphere

E:\WebSphere>cd AppServer

E:\WebSphere\AppServer>cd bin


E:\WebSphere\AppServer\bin>assembly


E:\WebSphere\AppServer\bin>startServer


WebSphere Application Server, Advanced Single Server Edition V4
Application Server Launcher
Copyright (C) IBM Corporation, 2001

The configuration file was defaulted to:
  E:\WebSphere\AppServer\config\server-cfg.xml
Using the single available node or the localhost node.
Using the single available server.
Initiating server launch.
Loaded domain "WebSphere Administrative Domain".
Selected node "yorkt".
Selected server "Default Server".
WSPL00651: Initiated server launch with process id 361.
Time mark: Saturday, August 25, 2001 4:13:44 PM EDT
Waiting for the server to be initialized.
Time mark: Saturday, August 25, 2001 4:13:51 PM EDT
Initialized server.
Waiting for applications to be started.
Time mark: Saturday, August 25, 2001 4:14:24 PM EDT
Started applications.
WSPL00571: The server Default Server is open for e-business.
Please review the server log files for additional information.
Standard output: E:\WebSphere\AppServer\logs/default_server_std
Standard error: E:\WebSphere\AppServer\logs/default_server_stde

E:\WebSphere\AppServer\bin>
```

 Plan to use the default [server configuration file](#) for this tutorial. Other configurations are available, including those that you customize.

 Starting the server from a system command prompt is a method available on all supported operating systems. There are [other ways to start the server](#), as well as variations of the **startServer** command.

 This tutorial assumes that the commands to start the product and tooling can be issued at a system command window, opened anywhere on your machine. If you have any trouble, change directory to:

[product\\_installation\\_root](#)/bin

and try the command again.

## Open the administrative console


To open the [administrative console](#):

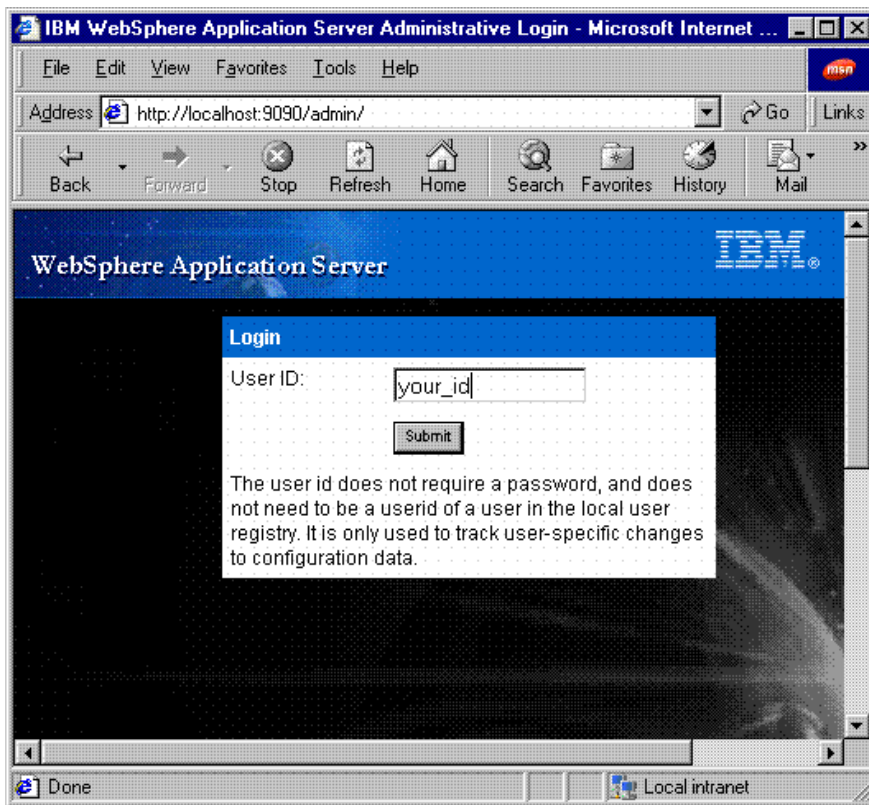
1. Ensure that the Web server is running on the machine containing the console.
2. Ensure that the product (application server) is also running. Recall, you started it a while ago.
3. In a Web browser, enter the URL:

`http://your_host_name:9090/admin`

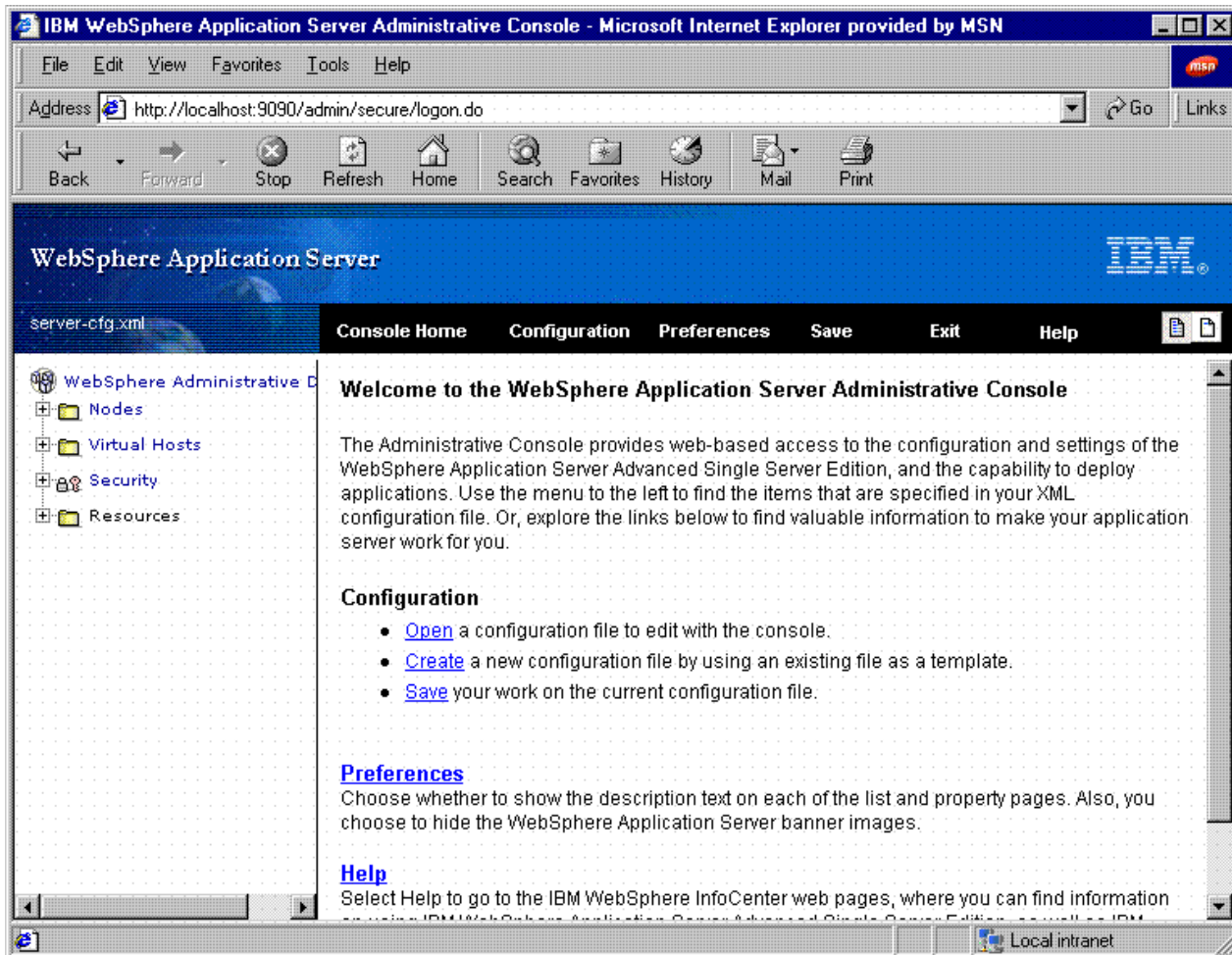
where *your\_host\_name* is localhost if the Web console is on the local machine. On Windows 2000, it has been found that localhost is not always recognized. In such a case, use the actual host name.

4. Enter an arbitrary login ID.

 Even if you have not secured the console yet, you will be prompted to log in. The login area explains this somewhat. For more information about the use of this ID, see [the help for starting and logging into the console](#). It will also contain any additions, corrections, and details pertaining to the above instructions.



5. Wait for the console to load into the browser.

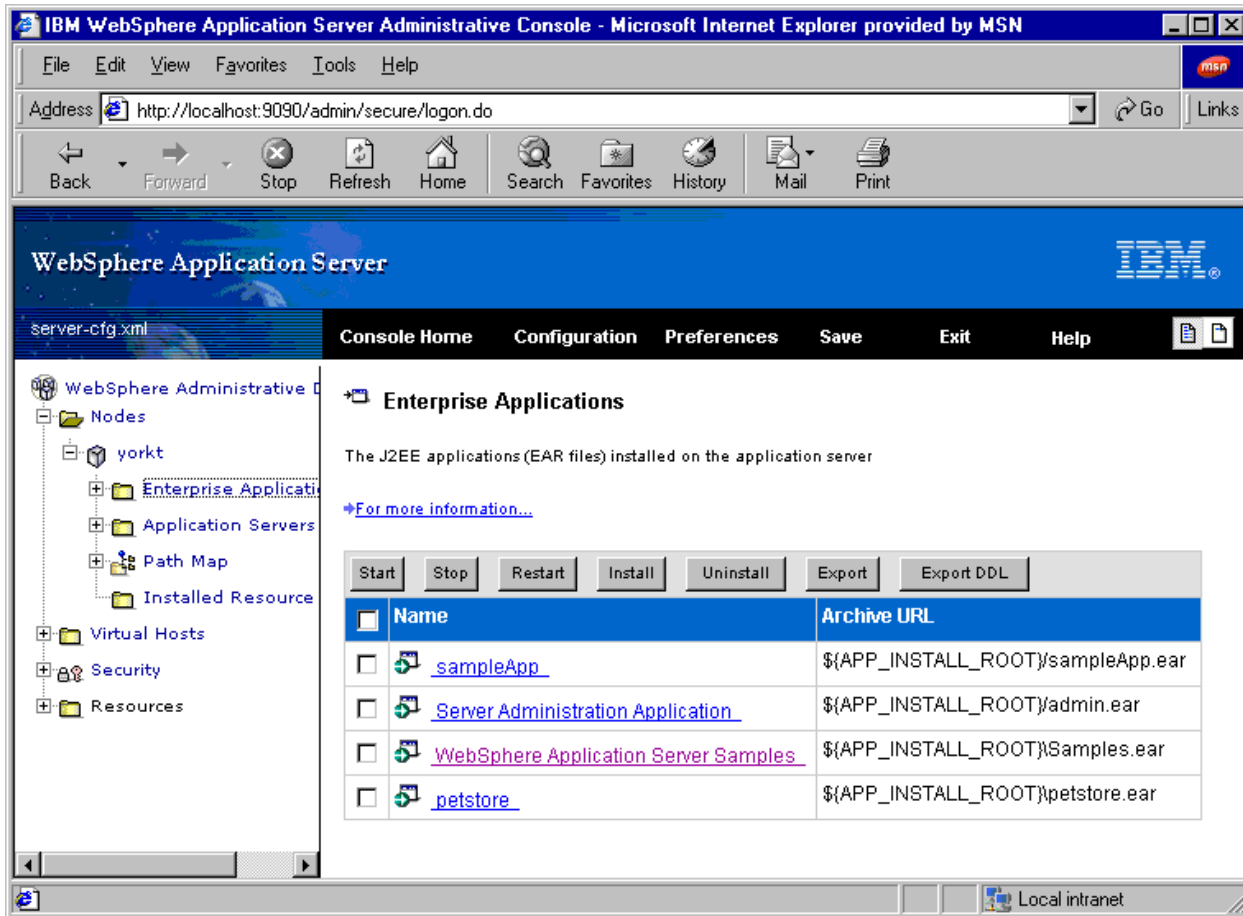


## Prepare the application server and needed resources

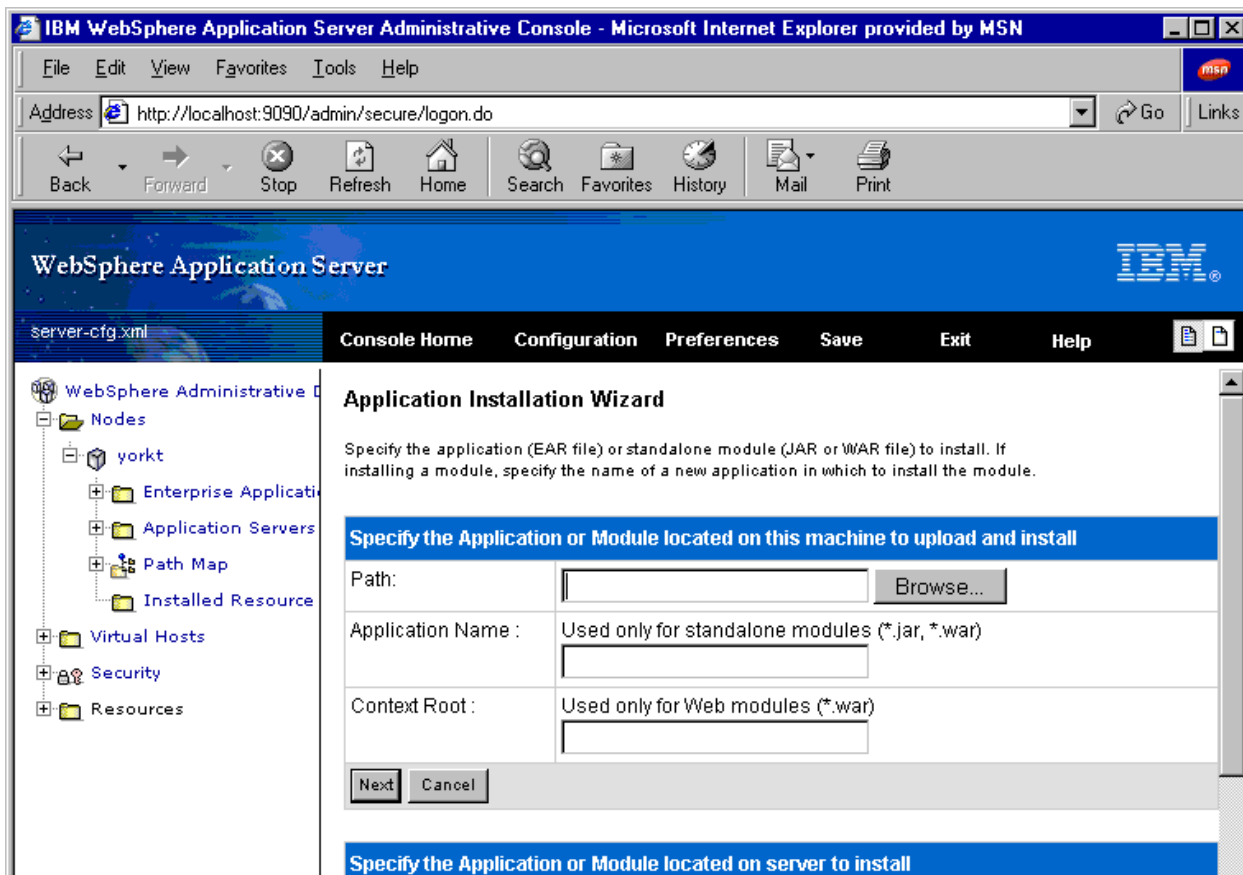
For the purposes of the tutorial, use the WebSphere runtime environment "as is." The default application server and other resources are everything you need to deploy the simpleSession application.

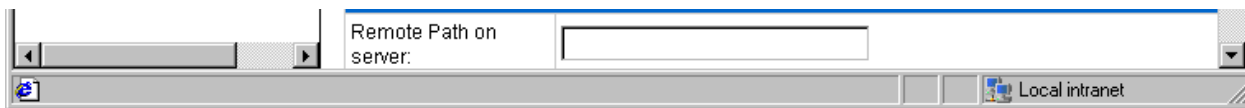
## Install the application

1. In the tree view on the left side of the console, expand **Nodes** -> **localhost**
2. Click **Enterprise Applications** to display a list of applications.
3. Click the **Install** button in the right pane.

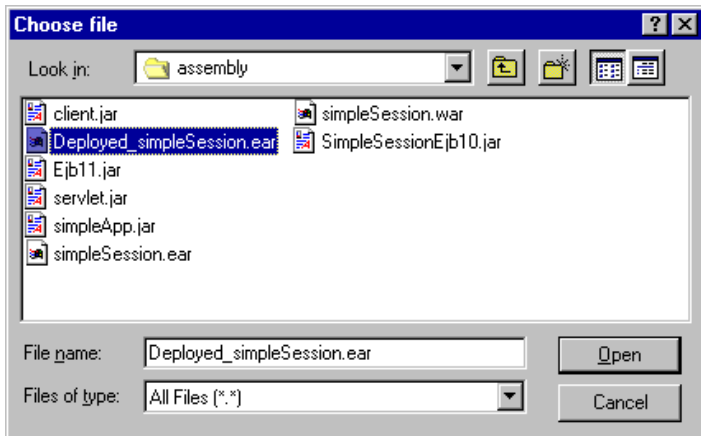


4. Click **Browse**.





5. Navigate to your deployed .ear file (**Deployed\_simpleSession.ear**), select it and click **Open**.



6. Click **Next**.

7. On the **Binding Enterprise Beans to JNDI Names** panel:

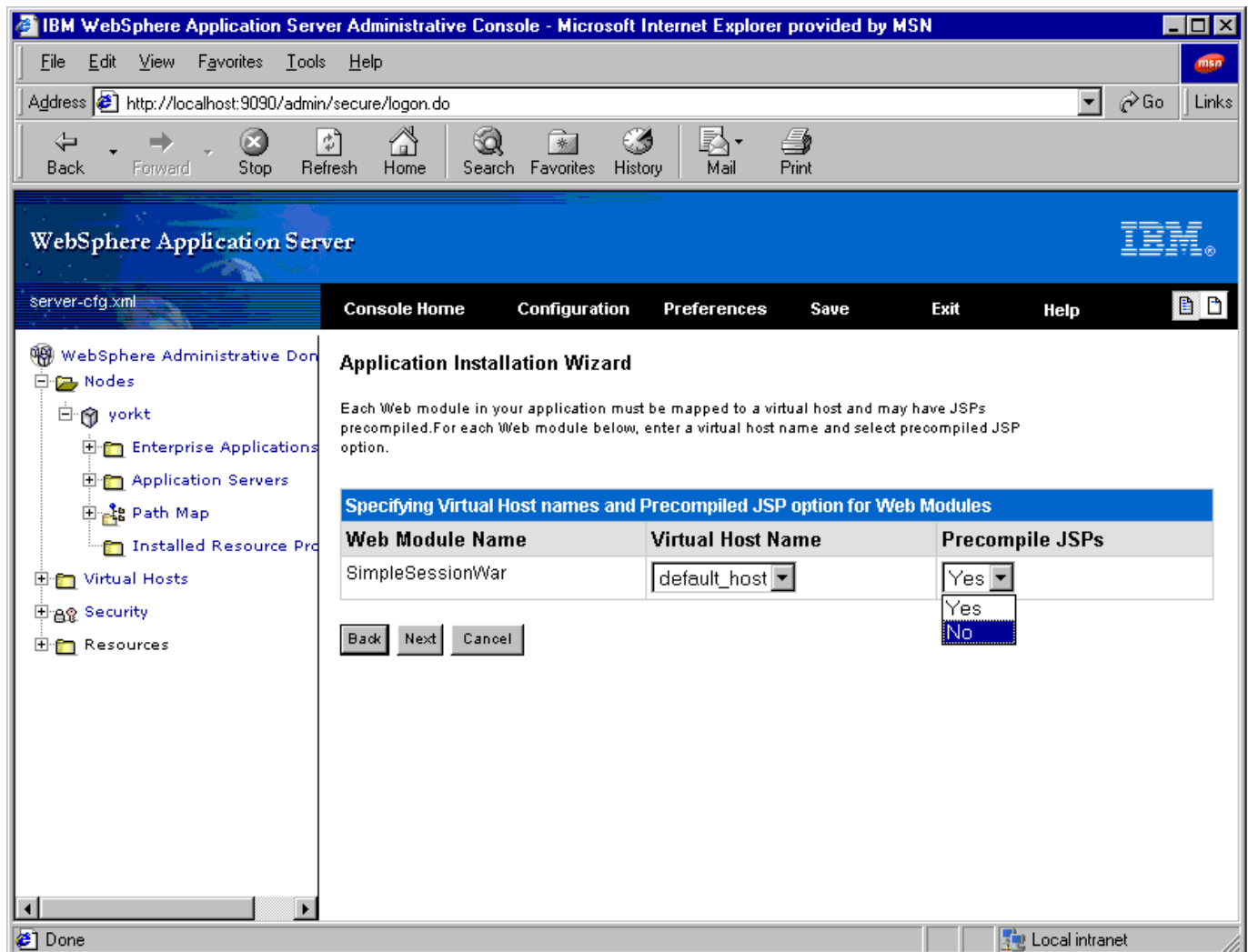
1. Verify the JNDI Name is set to gs/hello and select **Next**

8. On the **Mapping EJB References to Enterprise Beans** panel:

- a. Verify the Enterprise Bean is set to gs/hello and select **Next**.

9. On the **Specifying Virtual Host names... for Web Modules** panel:

- a. Set the **Virtual Host** to default\_host
- b. Set the **Precompile JSPs** to No



c. Click **Next**.

10. On the **EJB Deploy** page, *deselect* the check box for Re-Deploy option.

Because we deployed the .ear file from the application assembly tool, there is no need to re-deploy the EJB jar now. We could have skipped this step in the assembly tool and deployed the EJB JAR using the administrative console.

11. Click **Next**.

### Application Installation Wizard

Confirm the information that you entered for installing the application. The application or module will be installed with the settings you provided. To change the settings, click the **Back** button. To finish installing the application, click the **Finish** button.

**Confirm the following:**

Application Name =

Application File Pathname =

E:\WebSphere\AppServer\temp\yorkt\Default\_Server\Server\_Adminon\admin.war\Deployed\_simpleSession.ear

RoleAssignments:

Run as Mapping for Roles:

EJB Reference Bindings:

EJB Reference = ejb/ABean JNDI Name = gs/hello

Back

Finish

Cancel

12. Click **Finish**.

The application is now displayed in the list of installed applications.

IBM WebSphere Application Server Administrative Console - Microsoft Internet Explorer provided by MSN

File Edit View Favorites Tools Help

Address http://localhost:9090/admin/secure/logon.do

Back Forward Stop Refresh Home Search Favorites History Mail Print

WebSphere Application Server

server-cfg.xml Console Home Configuration Preferences Save Exit Help

WebSphere Administrative Domain

Nodes

yorkt

Enterprise Applications

Application Servers

Path Map

Installed Resource Providers

Virtual Hosts

Security

Resources

Configuration needs to be saved

Plug-in configuration needs to be regenerated.

Enterprise Applications

The J2EE applications (EAR files) installed on the application server

For more information...

Start Stop Restart Install Uninstall Export Export DDL

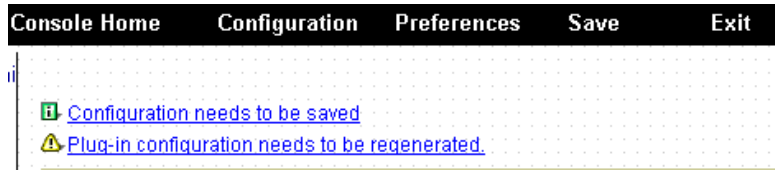
<input type="checkbox"/>	Name	Archive URL
<input type="checkbox"/>	<a href="#">sampleApp</a>	\${APP_INSTALL_ROOT}\sampleApp.ear
<input type="checkbox"/>	<a href="#">Server Administration Application</a>	\${APP_INSTALL_ROOT}\admin.ear
<input type="checkbox"/>	<a href="#">WebSphere Application Server Samples</a>	\${APP_INSTALL_ROOT}\Samples.ear
<input type="checkbox"/>	<a href="#">petstore</a>	\${APP_INSTALL_ROOT}\petstore.ear
<input type="checkbox"/>	<a href="#">SimpleSessionApp</a>	\${APP_INSTALL_ROOT}\Deployed_simpleSession.ear

13. In the topology tree, expand **Nodes** -> **hostname** -> **Application Servers**
14. Select **Default Server**
15. Click **OK** at the bottom of the pane.

## Saving the server configuration and regenerating the plug-in configuration

**I** The administrative console is good at prompting you to take action when you need to do so. In this case, it will now remind you to save your [server configuration file](#) and [regenerate the configuration](#) of the WebSphere plug-in for the Web server you are using. The prompts are displayed directly below the console menu bar.

1. Click the first prompt, "Configuration needs to be saved."



2. On the **Save Configuration** panel, click **OK**.
3. Return to the **Enterprise Applications** collections view where the two prompts were displayed. You should be able to do so by clicking your browser's **Back** button twice.

If that does not work, in the tree view of the console, click:

**Nodes -> your\_host\_name -> Enterprise Applications**

4. Now click the second prompt, "Plug-in configuration needs to be regenerated."
5. On the **Web Server Plug-In Configuration** panel, click **Generate**.
6. When returned to the **Enterprise Applications** page, confirm that the prompt now says: New plug-in configuration has been generated.

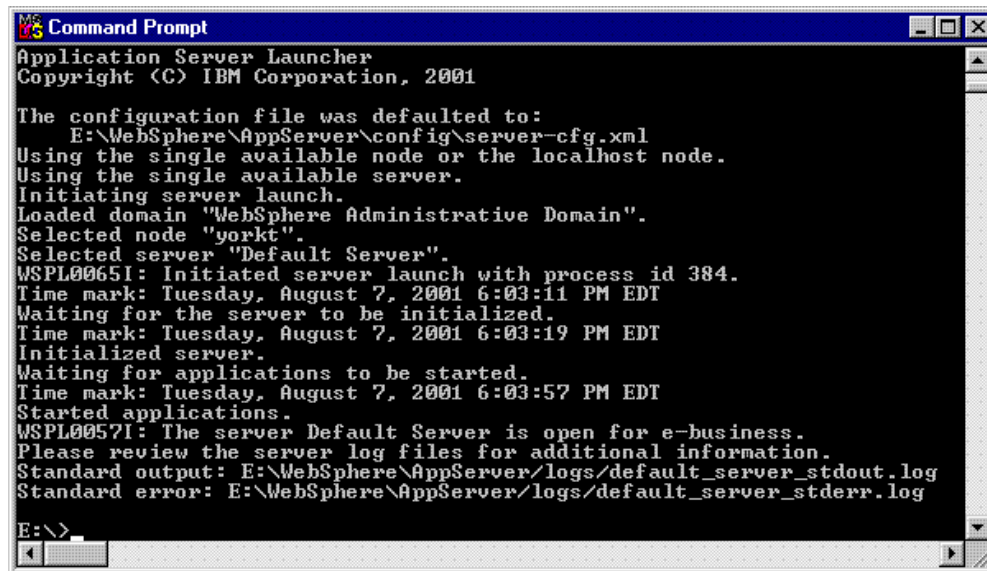
## Stopping the application server and starting it again

1. In the tree view of the console, click:  
**Nodes -> your\_host\_name -> Application Servers** to display the **Application Servers** collections view, including the Default Server.



2. Click **Stop**.
3. Click **OK** when the screen discussing the implications of stopping the server is displayed.
4. Close the Web browser.
5. Start the server, using the startServer script, as described earlier in this tutorial.

The server will issue a confirmation message in the launch window when it has started successfully.



**I** Because adding a new application requires a change to the server configuration file, the server must be stopped and started again to pick up the change. However, there are some types of changes you can make to an installed application that do not require stopping the server and starting it again. See [the dynamic reloading and hot deployment documentation](#) for more information.

## What's next?

Now that you have your application deployed, it is time to verify that users will be able to access it, using either a Web client or Java client. The next steps are described in the [Application testing tutorial](#).



# 6.7.3: Application testing tutorial

During this tutorial, you will test the application that you deployed during the [Application deployment tutorial](#). This involves trying the Web and Java clients of the application to ensure the application can be accessed by users.



## Prerequisites

You need to have performed the [Application deployment tutorial](#) successfully so that you have a deployed application to test.

## Overview of steps (requires 10 minutes)

- 1. [Ensure that the application, application server, and Web server are running](#)
- 2. [Test the Web client](#)
- 3. [Test the Java application client](#)

## Paths through the tutorial

- **Quick path:** If your objective is to practice these steps as quickly as possible, skip the explanations marked with the  graphic.
- **Maximum learning path:** If your objective is to understand and explore the product capabilities, read the  notes and browse the links they provide to additional documentation.

## Ensure that the application, application server, and Web server are running

For successful use of the client:

- The application must be running
- The application server hosting the application must be running
- In the case of the Web client, the Web server must be running






To verify whether the application and application server are running:

- 1. Open the administrative console, [as discussed in the deployment tutorial](#).
- 2. Click **Nodes** -> *localhost* -> **Enterprise Applications** in the console tree view.
- 3. Verify that your application is running (by noting the graphic next to it).

### Enterprise Applications

The J2EE applications (EAR files) installed on the application server

[For more information...](#)

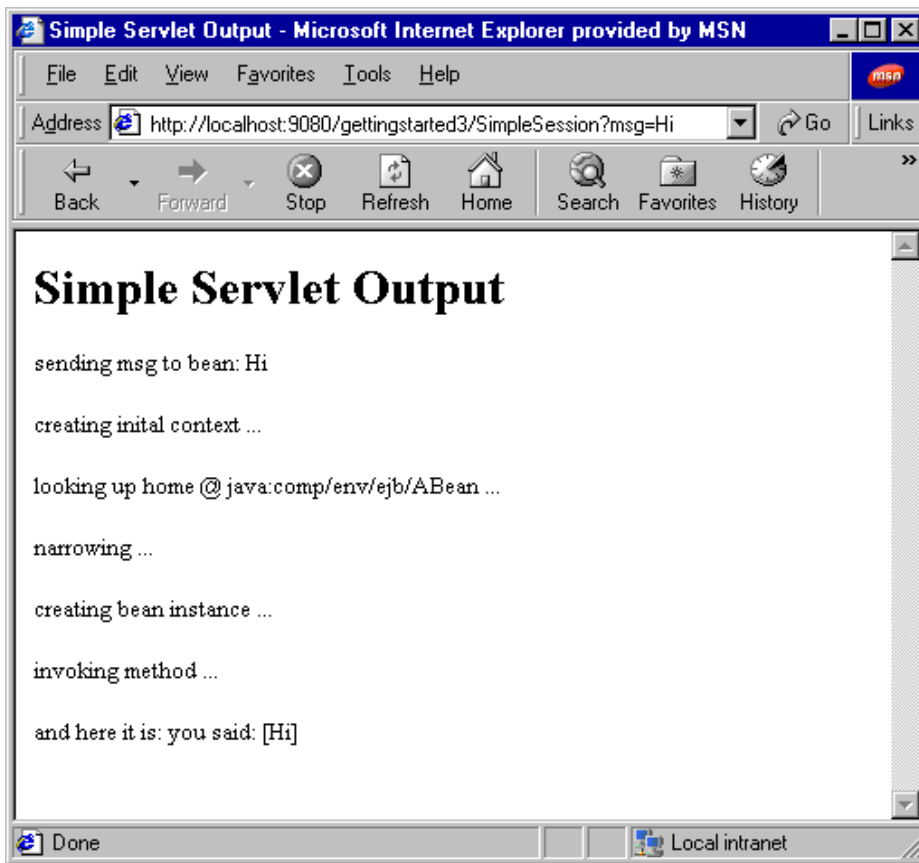
<div>StartStopRestartInstallUninstallExportExport DDL</div>		
<input type="checkbox"/>	Name	Archive URL
<input type="checkbox"/>	 <a href="#">sampleApp</a>	E:\WebSphere\AppServer\installedApps\sampleApp.ear
<input type="checkbox"/>	 <a href="#">Server Administration Application</a>	E:\WebSphere\AppServer\installedApps\admin.ear
<input type="checkbox"/>	 <a href="#">WebSphere Application Server Samples</a>	E:\WebSphere\AppServer\installedApps\Samples.ear
<input type="checkbox"/>	 <a href="#">petstore</a>	E:\WebSphere\AppServer\installedApps\petstore.ear
<input type="checkbox"/>	 <a href="#">SimpleSessionApp</a>	E:\WebSphere\AppServer\installedApps\Deployed_simpleSession.ear

## Test the Web client

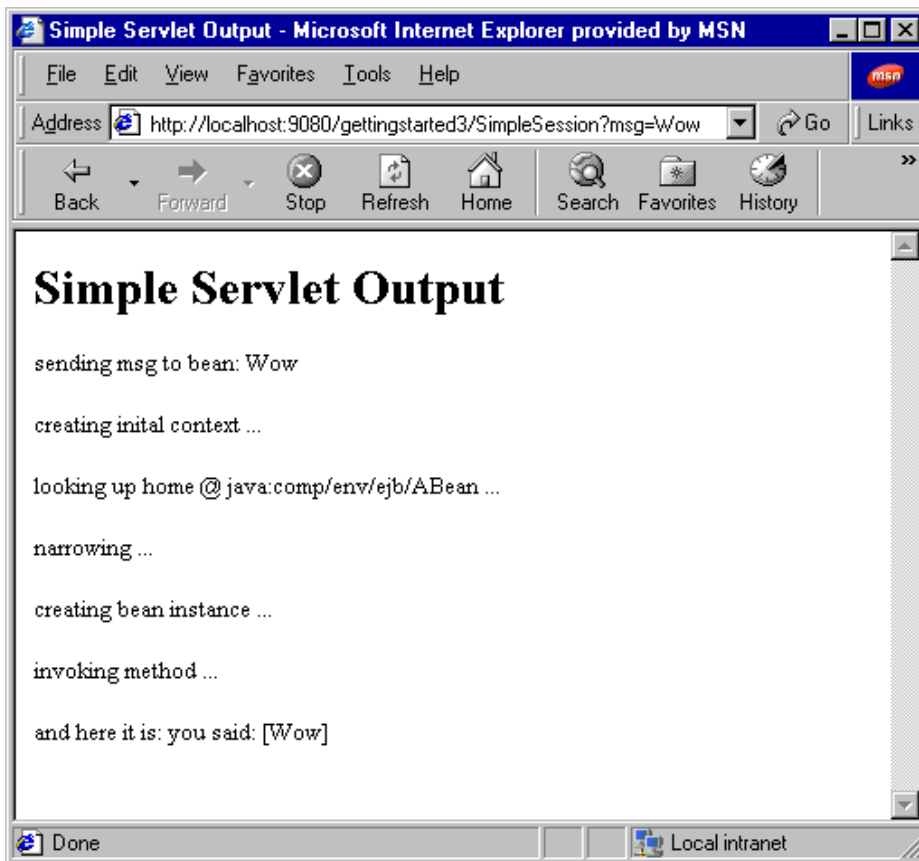
Test the Web client of simpleSession by browsing the URL:

<http://hostname:9080/gettingstarted3/SimpleSession?msg=Hi>





The client will return your original message of "Hi" (notice the end of the URL: ?msg=Hi). You can send other messages and have them echoed back to you, such as ?msg=Wow



## Test the Java application client

Test the Java application client by launching your client application in the client container.

1. Open a system command prompt.

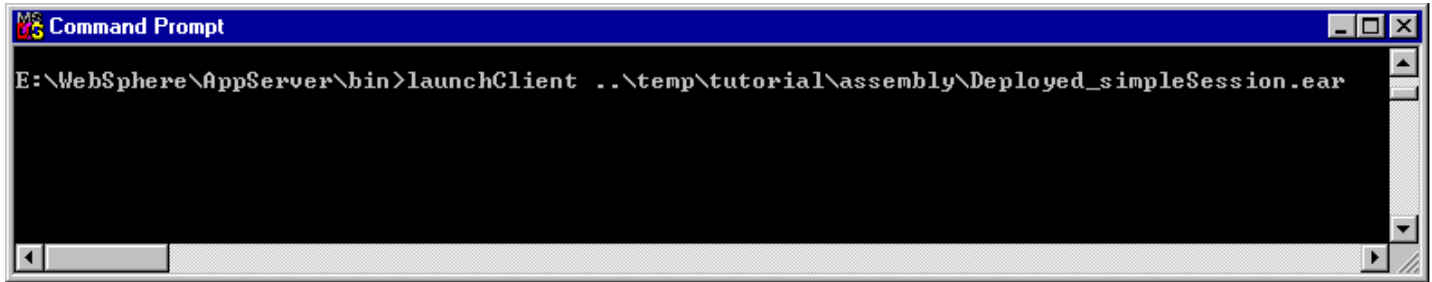
2. Change directory to:

`product_installation_root/bin`

3. Enter the command:

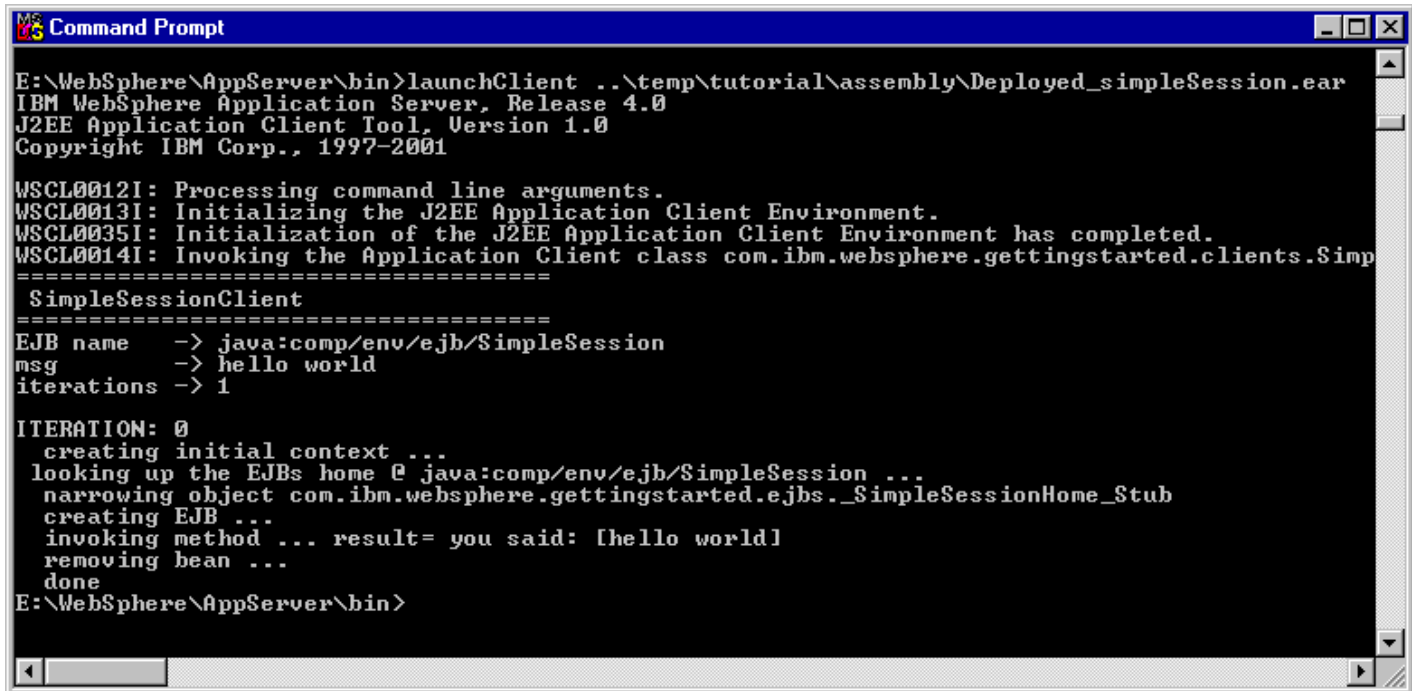
```
launchClient ../temp/tutorial/assembly/Deployed_simpleSession.ear
```

(Remember, on Windows systems, the forward slashes should be backslashes).



```
Command Prompt
E:\WebSphere\AppServer\bin>launchClient ..\temp\tutorial\assembly\Deployed_simpleSession.ear
```


The client will contact the enterprise bean and return a "hello world" message.



```
Command Prompt
E:\WebSphere\AppServer\bin>launchClient ..\temp\tutorial\assembly\Deployed_simpleSession.ear
IBM WebSphere Application Server, Release 4.0
J2EE Application Client Tool, Version 1.0
Copyright IBM Corp., 1997-2001

WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class com.ibm.websphere.gettingstarted.clients.SimpleSessionClient
=====
SimpleSessionClient
=====
EJB name    -> java:comp/env/ejb/SimpleSession
msg         -> hello world
iterations  -> 1

ITERATION: 0
  creating initial context ...
  looking up the EJBs home @ java:comp/env/ejb/SimpleSession ...
  narrowing object com.ibm.websphere.gettingstarted.ejbs._SimpleSessionHome_Stub
  creating EJB ...
  invoking method ... result= you said: [hello world]
  removing bean ...
  done
E:\WebSphere\AppServer\bin>
```

 The above command is issued from the bin directory because the argument providing the path to the Deployed\_simpleSession.ear is relative to the bin directory. Actually, you should be able to issue the command from any directory, as long as you correctly specify the path to the .ear file whose application client you want to launch. More information is available about the [launchClient](#) command.

## What's next?

Now that you have assembled and deployed an unsecured application, and tested your results, you might be interested in securing your application. The [Application security tutorial](#) describes how to do so, using local operating system authentication and login forms.

## 6.7.4: Application security tutorial

During this tutorial, you will enable security in all three containers of your simpleSession application:

- Web
- EJB
- Client

You will use the Application Assembly Tool (AAT) to declare and define J2EE security roles, as well as to control authorization on various J2EE modules. You will also enable security in the application server runtime and test your settings.

Authentication will be performed using the local operating system user registry. This example only uses declarative security. It does not illustrate any of the programmatic methods supported by the J2EE programming model.



### Prerequisites

You need to have performed the Application deployment tutorial and Application testing tutorial successfully in order to perform this tutorial successfully. If you ran into trouble testing your application, you can still perform this tutorial to practice the steps involved, but the testing phase of this tutorial will be unsuccessful.

### Overview of steps (requires 45 to 60 minutes)

1. [Enable security in your application](#)
2. [Enable security in the application server runtime](#)
3. [Install the secured application](#)
4. [Regenerate the Web server plug-in and save the server configuration](#)
5. [Stop the server and start it again](#)
6. [Ensure the application, application server, and Web server are running](#)
7. [Test the Web client](#)
8. [Test the Java client](#)
9. [Disable security in the application server runtime](#)

### Paths through the tutorial

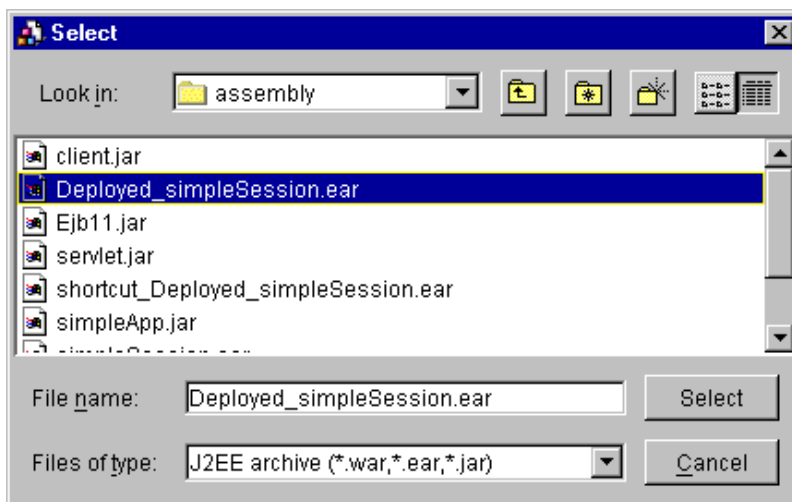
- **Quick path:** If your objective is to practice these steps as quickly as possible, skip the explanations marked with the  graphic.
- **Maximum learning path:** If your objective is to understand and explore the product capabilities, read the  notes and browse the links they provide to additional documentation.

### Enable security in your application

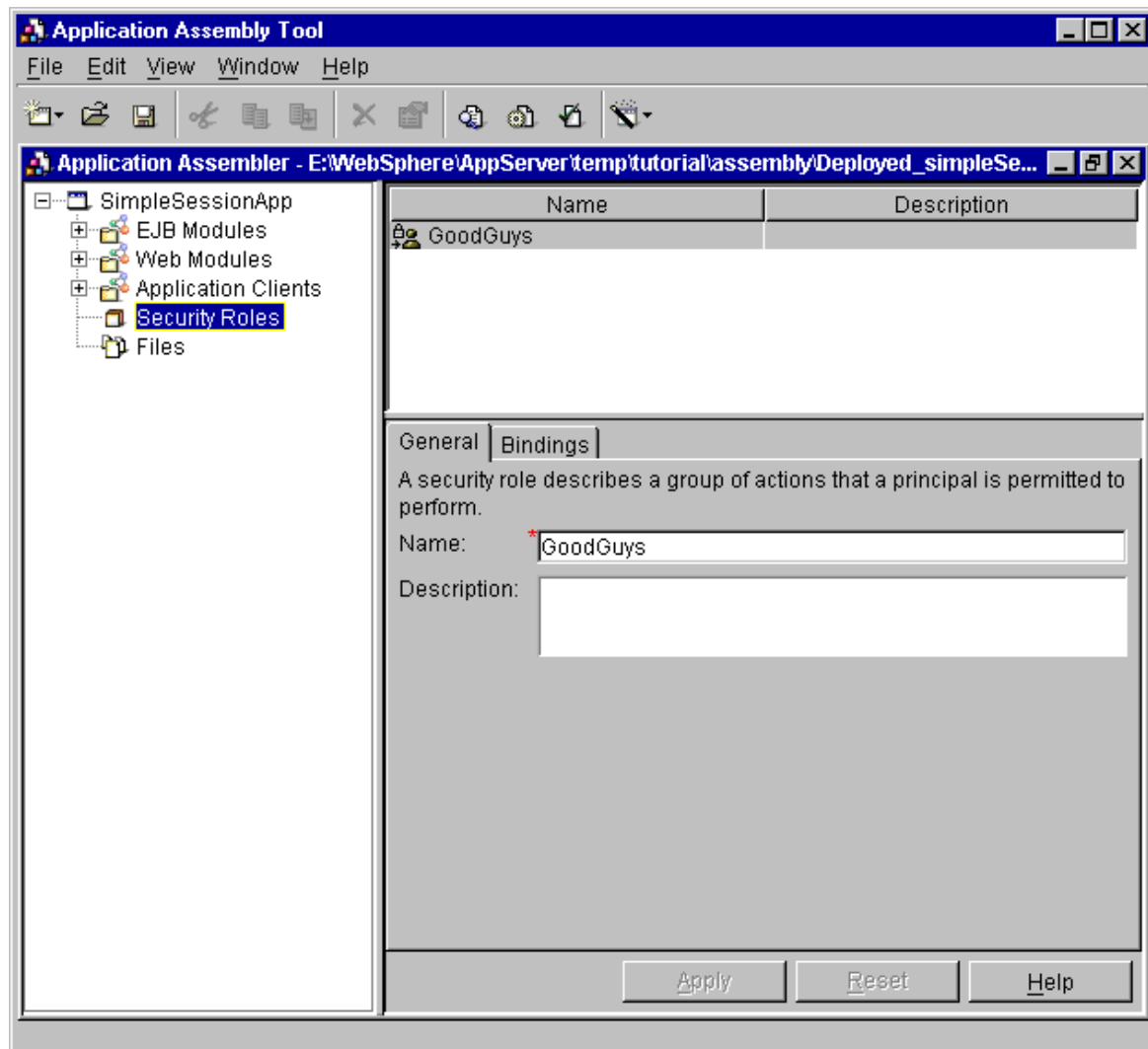
1. Start the Application Assembly Tool, [as discussed in the Application assembly tutorial](#).
2. Edit the Deployed\_simpleSession.ear application.

On the **Welcome to Application Assembly Tool** panel:

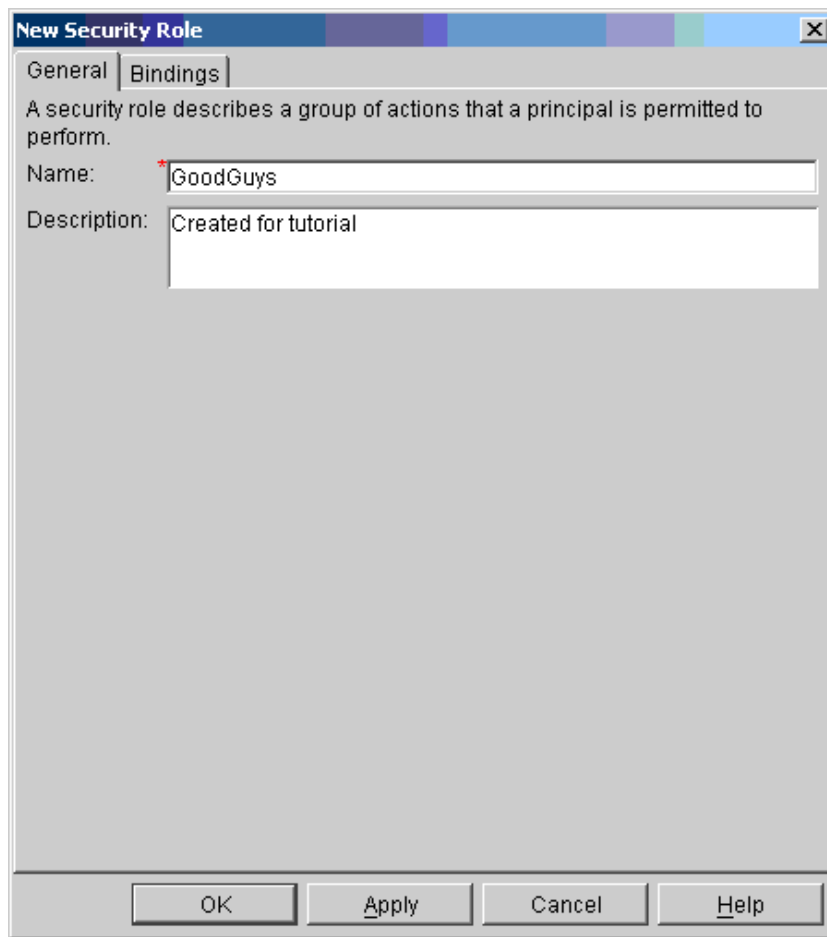
1. Click the **Existing** tab.
2. Click **Browse** next to the File name field.



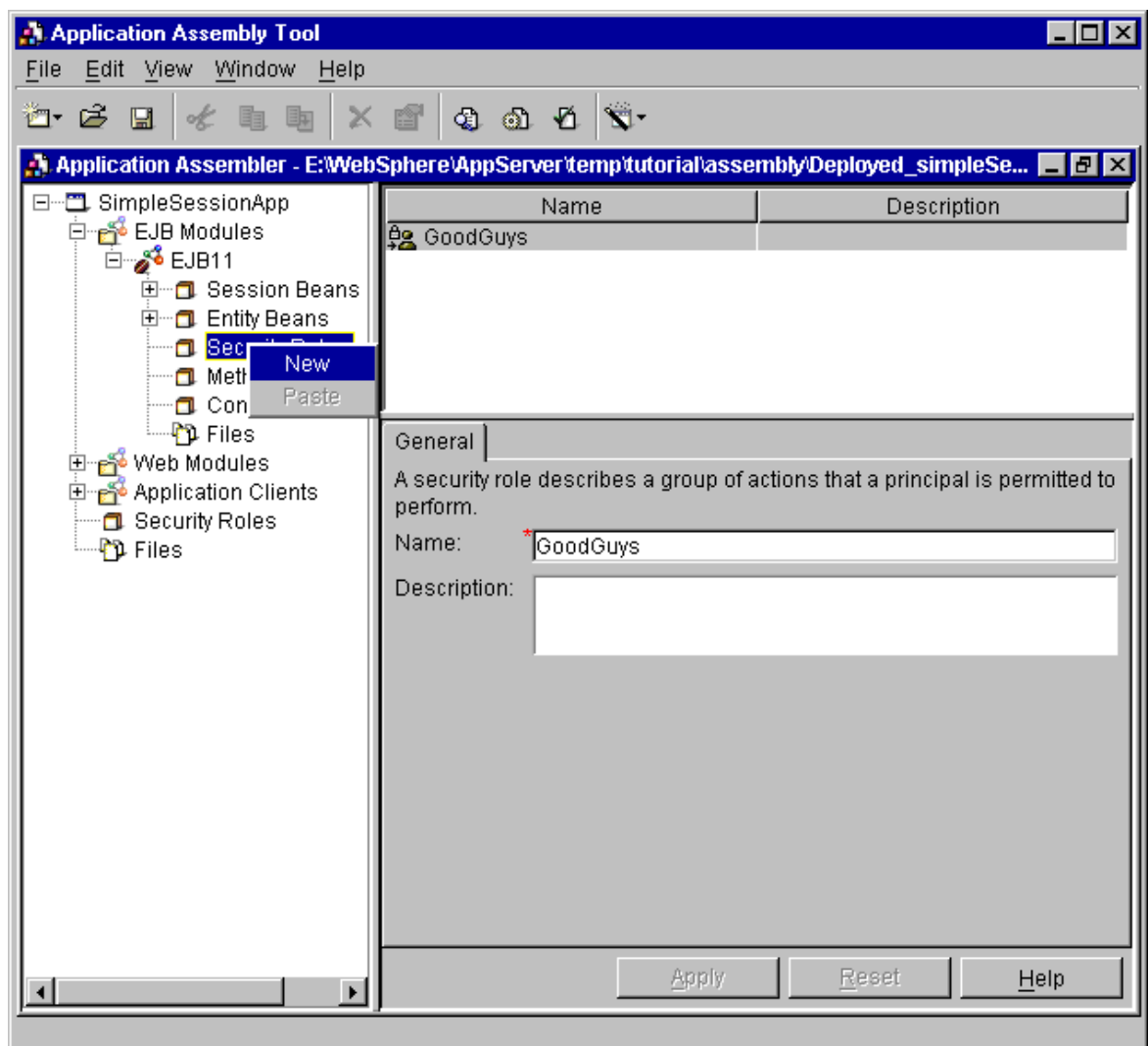
3. Navigate to the **Deployed\_simpleSession.ear** that was assembled in the Simple Session Bean tutorial and select **Select**.
4. Select **OK**.
3. Create a security role to which you will be granting authorization to the session bean and servlet in the application:
  - a. In the .ear file, select the **Security Roles** folder.



- b. Right-click it and select **New**.
- c. On the **General** tab, enter **GoodGuys** for the **Name**.
- d. Optionally, enter a **Description**.



- e. Select the **Bindings** tab.
- f. Associate your (native) user ID with this security role:
  - 1. Click **Add...** (next to the **Users:** area).
  - 2. Specify your user ID in the **Name** field.
  - 3. Click **OK**.
- g. Click **OK**.
- 4. Authorize methods on the enterprise bean. You will authorize all home and remote methods to security role GoodGuys:
  - a. In the tree view, expand **EJB Modules**.
  - b. Expand the **EJB11**.
  - c. Create a new security role:
    - 1. Select **Security Roles**.
    - 2. Right-click it and select **New**.



3. Enter **GoodGuys** as the Name.
4. Optionally, enter a description.

**New Security Role**

General

A security role describes a group of actions that a principal is permitted to perform.

Name:

Description:

OK Apply Cancel Help

5. Click **OK**.

5. Create a new method permission:

- In the tree view, select the **Method Permissions** under EJB11.
- Right-click it and select **New**.
- Add the methods:
  - For the **Name**, specify myMethodPerm.
  - In the **Methods** area, click **Add...**

**New Method Permission**

General

Method permissions map security roles to the methods that a member of that role can invoke.

Method permission name:

Description:

Methods:

Name	Enterprise B...	Type	Parameters

Add ... Remove

Roles:

Role name

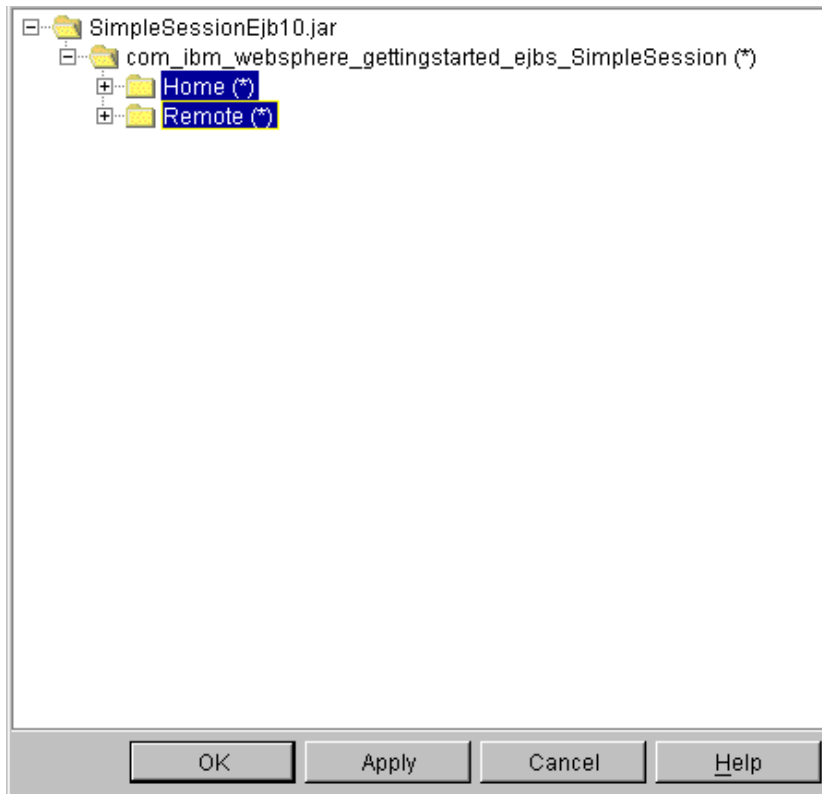
Add ... Remove

OK Apply Cancel Help

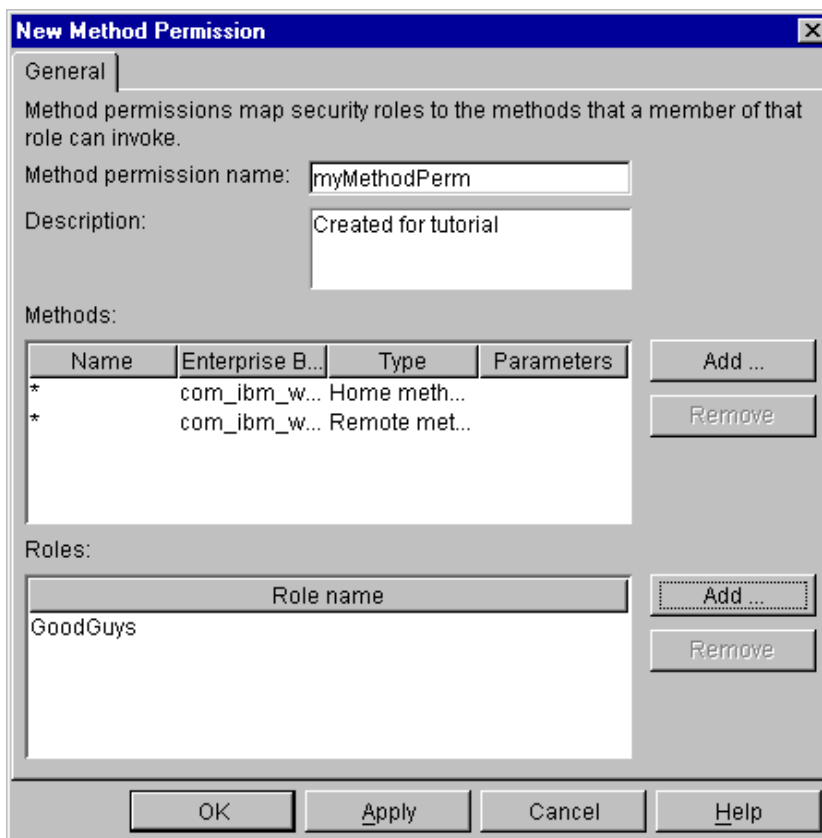
3. In the **Add Methods** dialog, expand the Ejb11.jar and the beans path:

com\_ibm\_websphere\_gettingstarted\_ejbs\_SimpleSession

4. Select the **Home** and **Remote** interface folders (hold down the Shift key to select both at once).



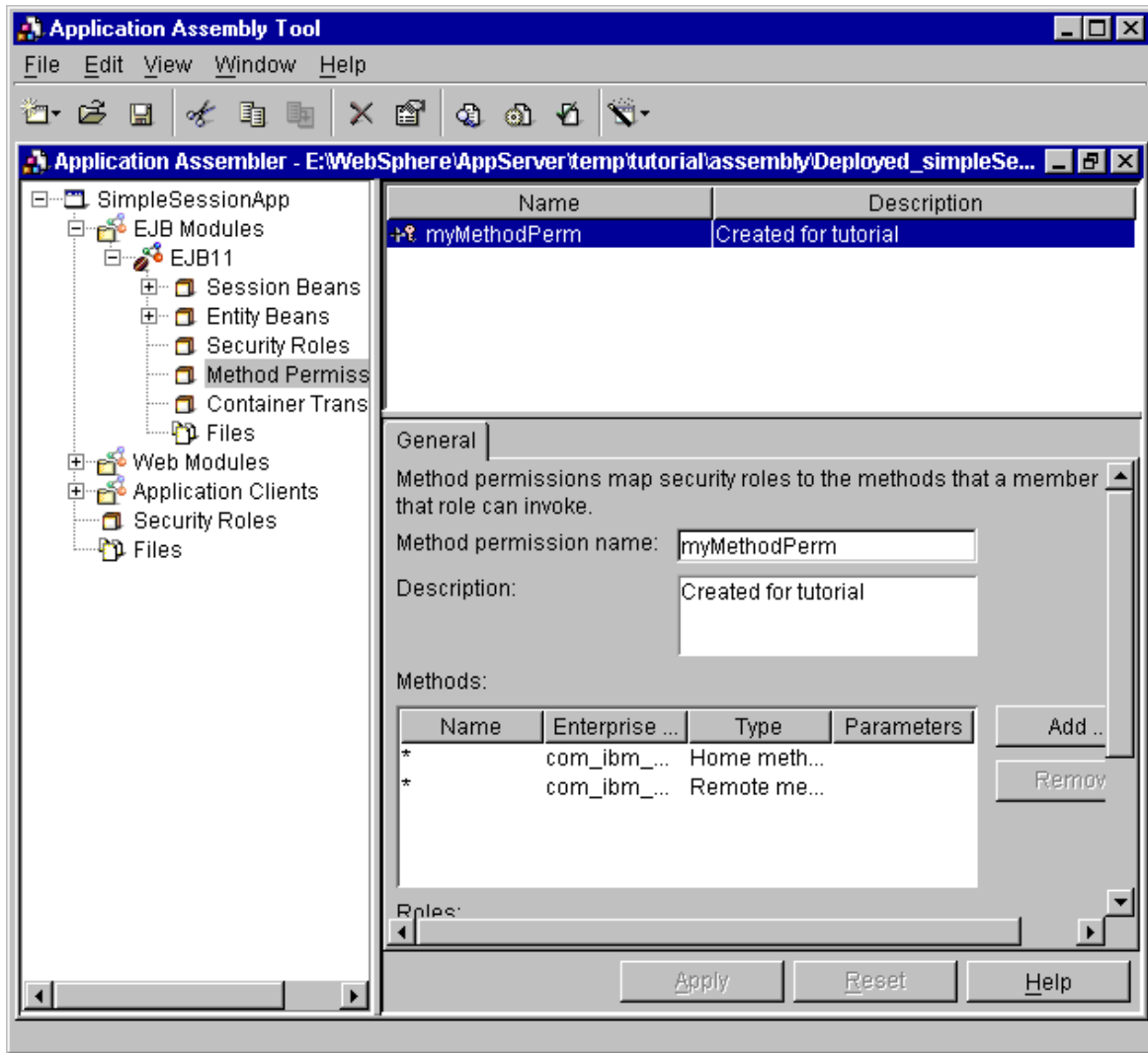
5. Click **OK**.
6. In the **Roles** area, click **Add...**
7. Select security role **GoodGuys**.
8. Click **OK**.



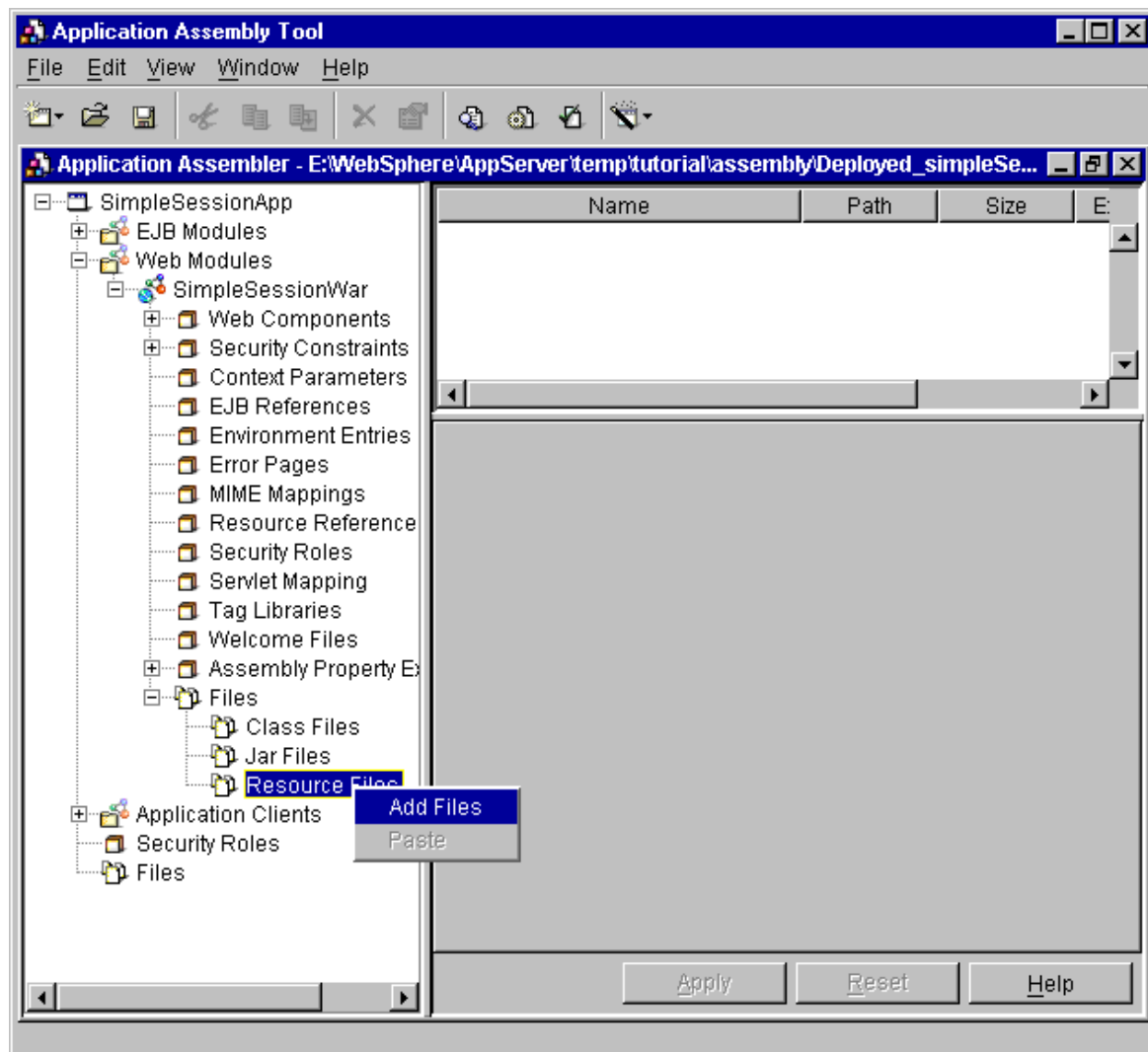
- d. Click **OK** to close the New Method Permissions window.

The





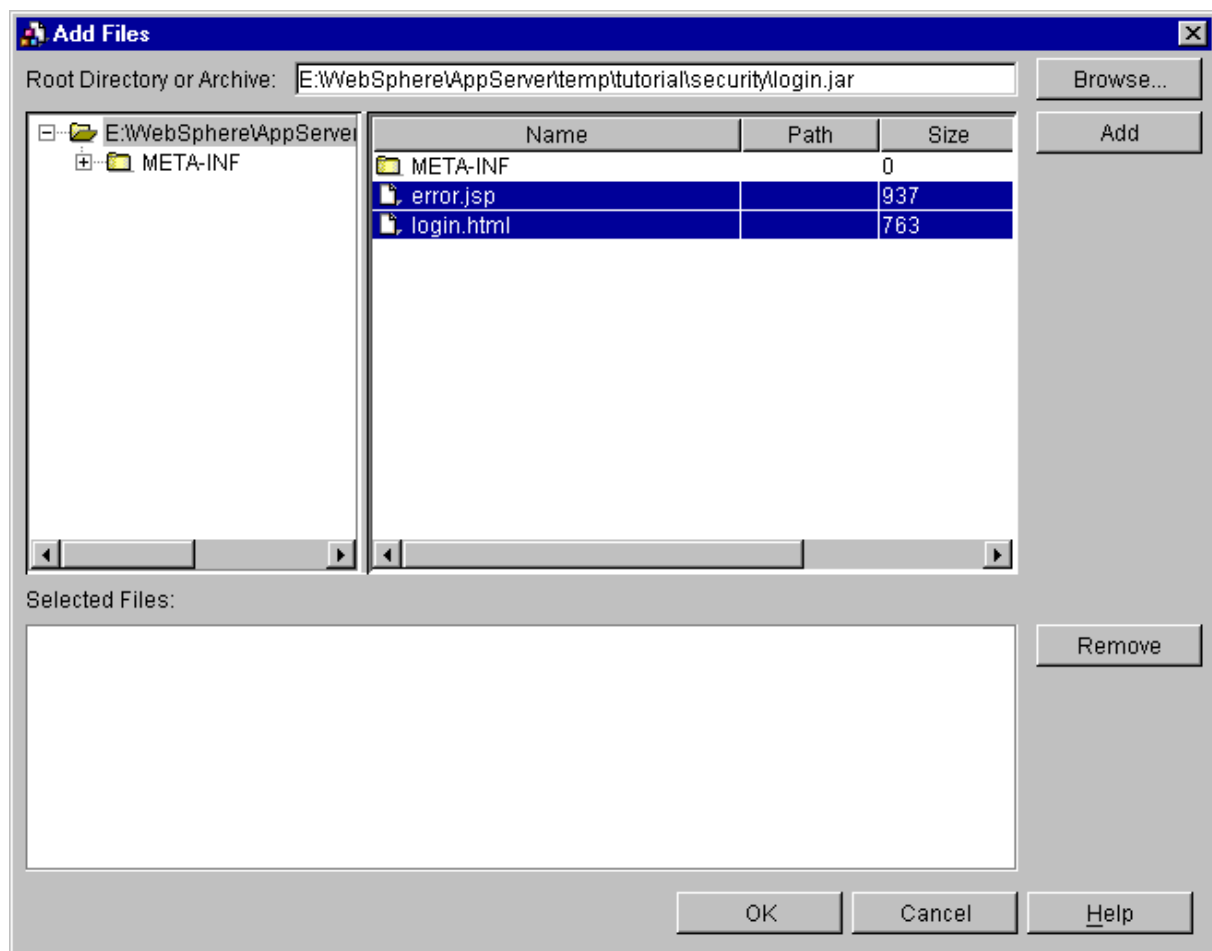
6. Authorize a servlet. Add a login form to be used for authentication:
  - a. In the tree view, expand **Web Modules** -> **SimpleSessionWar** -> **Files**
  - b. Select **Resource Files**.
  - c. Right-click it and select **Add Files**.



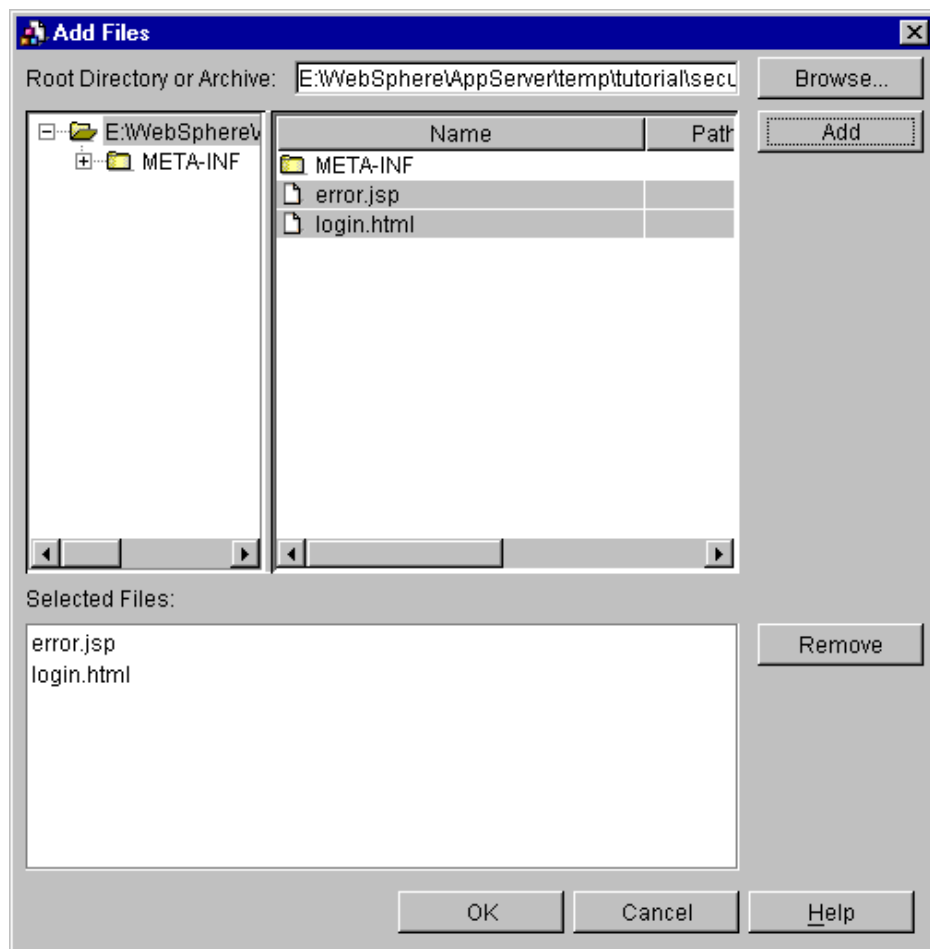
- d. Click **Browse** and navigate to the **login.jar** that you previously downloaded.



- e. Click **OK**.
- f. Select **login.html** and **error.jsp** (hold down the Shift key to select both at once).

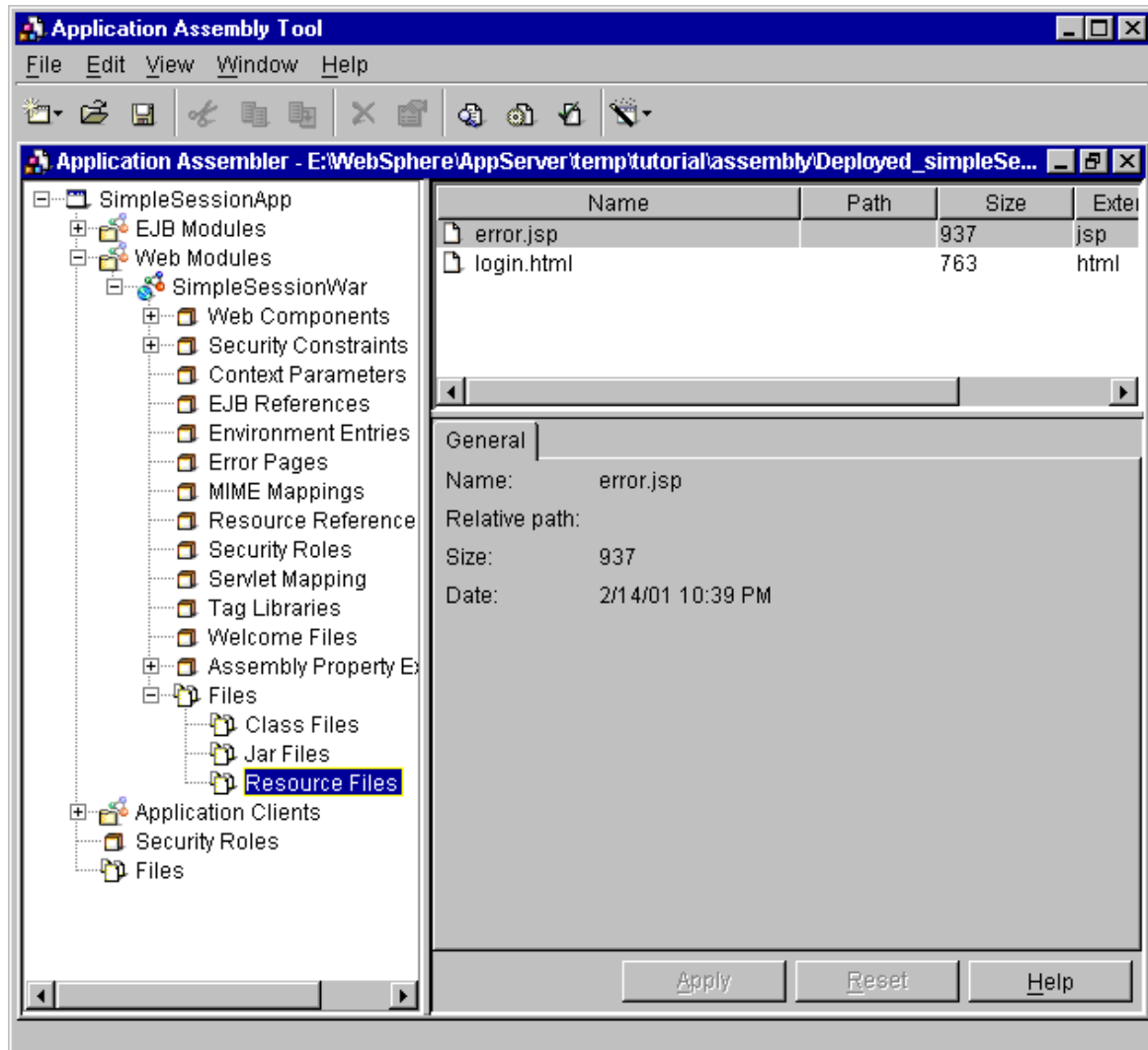


g. Click **Add**.



h. Click **OK**.

The files are displayed in the Application Assembly Tool, in the **Resource Files** folder of the Web module.



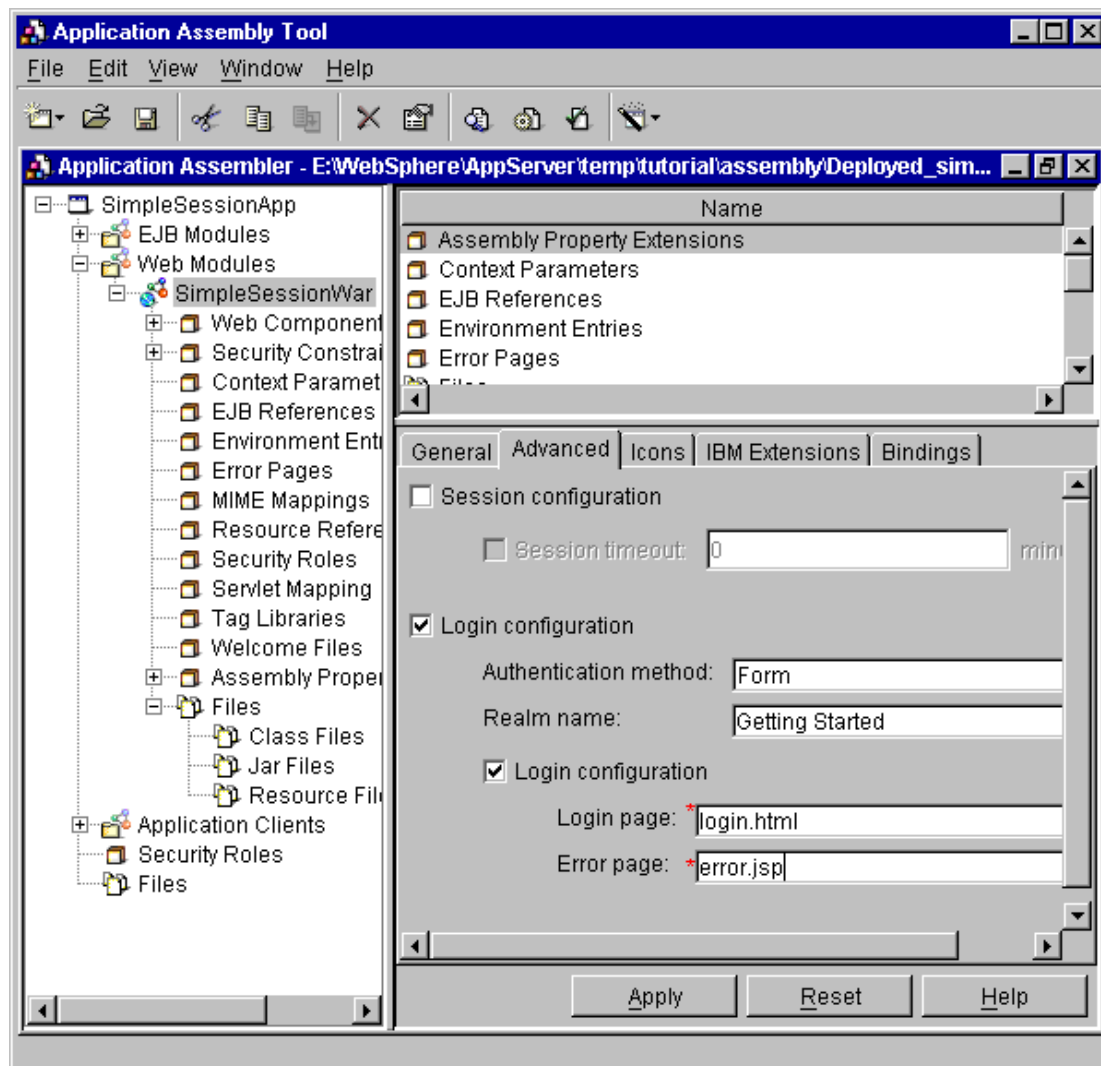
i. In the tree view, click the .war file named **SimpleSessionWar** to display its properties.

j. Click the **Advanced Tab**.

k. Check the **Login Configuration** box.

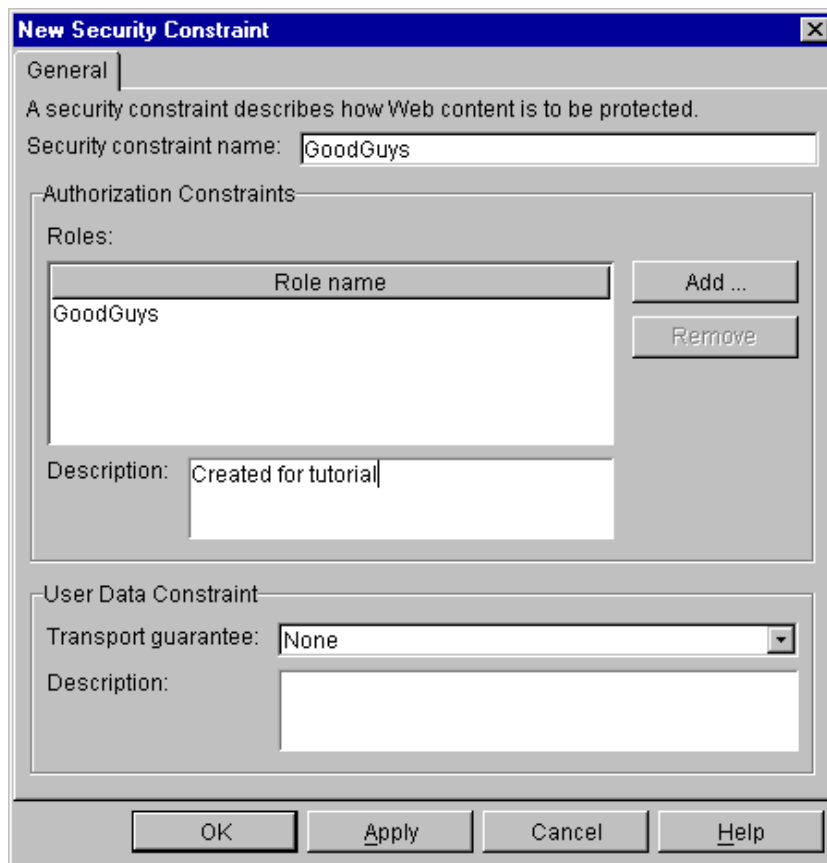
l. Do the following:

1. Set Authentication Method to **FORM**
2. Enter Realm name : **Getting Started**
3. Enter Login Page: **login.html**
4. Enter Error Page: **error.jsp**
5. Click **Apply** when finished.



7. Add a security constraint:

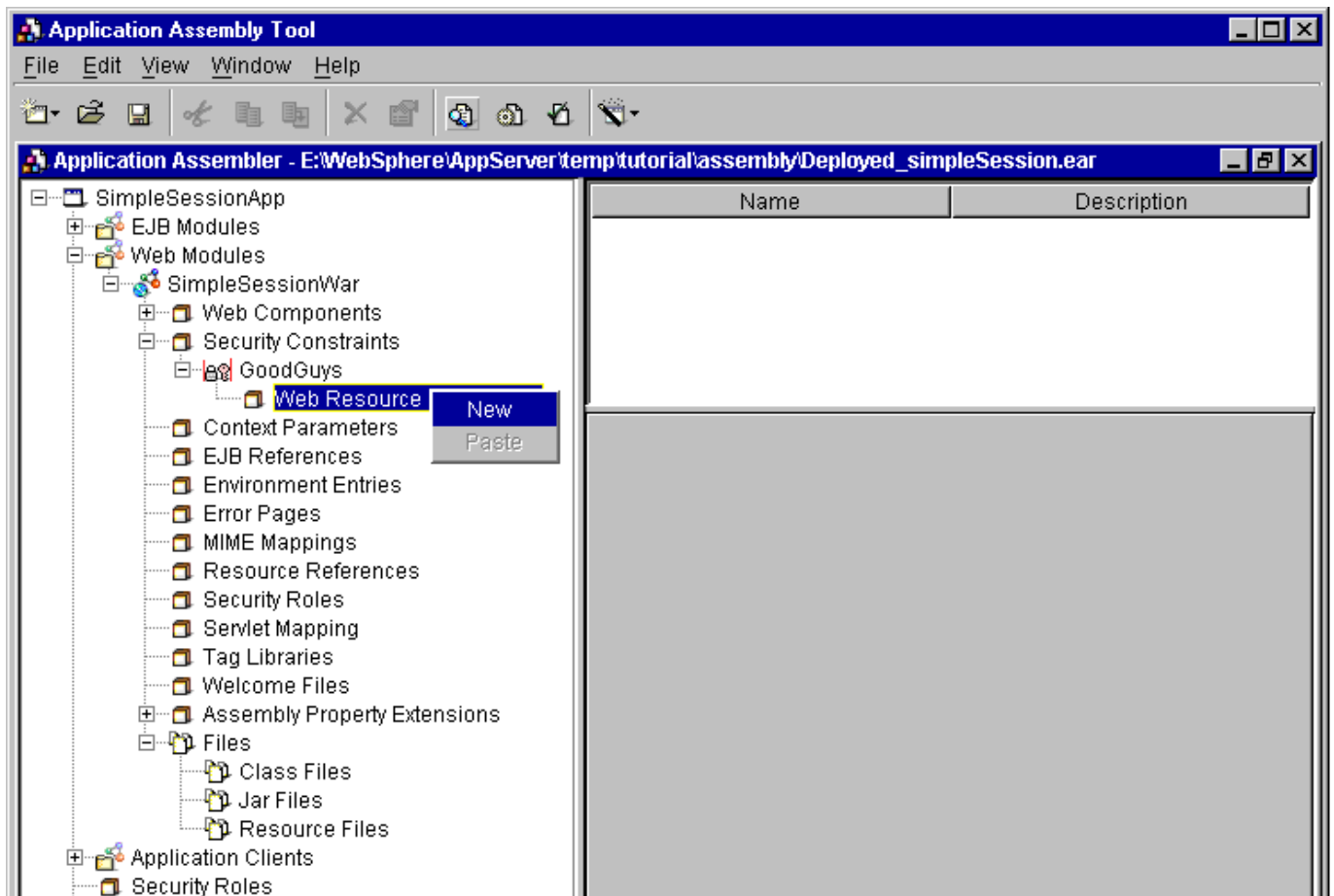
- a. Expand **SimpleSessionWar**.
- b. Select **Security Constraints**.
- c. Right-click it.
- d. Click **New**.
- e. Add a new security constraint:
  1. Enter the Security Constraint Name: **GoodGuys**
  2. Click **Add...** next to the Roles area.
  3. Select the Security Role of **GoodGuys**.
  4. Select **OK**.
  5. Set **Transport Guarantee** to **NONE**.



6. Click **OK**.

10. Add a resource collection to the constraint:

- Expand **SimpleSessionWar** -> **Security Constraints** -> **GoodGuys**
- Select **Web Resource Collections**.
- Right-click it and select **New**.



Files

Apply Reset Help

d. Add a new web resource collection:

1. Enter the Web Resource Name: **SecureMe**.
2. Click **Add...** in the **HTTP Methods** section.
3. Change HTTP Method to **POST**.
4. Click **OK**.
5. Click **Add...** in the **HTTP Method** section.
6. Change HTTP Method to **GET**.
7. Click **OK**.
8. Click **Add...** in the **URLs** section.
9. Enter **/SimpleSession** for the **URL Pattern**.

**New Web Resource Collection**

General

A Web resource collection defines a set of URL patterns (resources) and HTTP methods belonging to the resource.

Web Resource Name: \*SecureMe

Description:

HTTP methods:

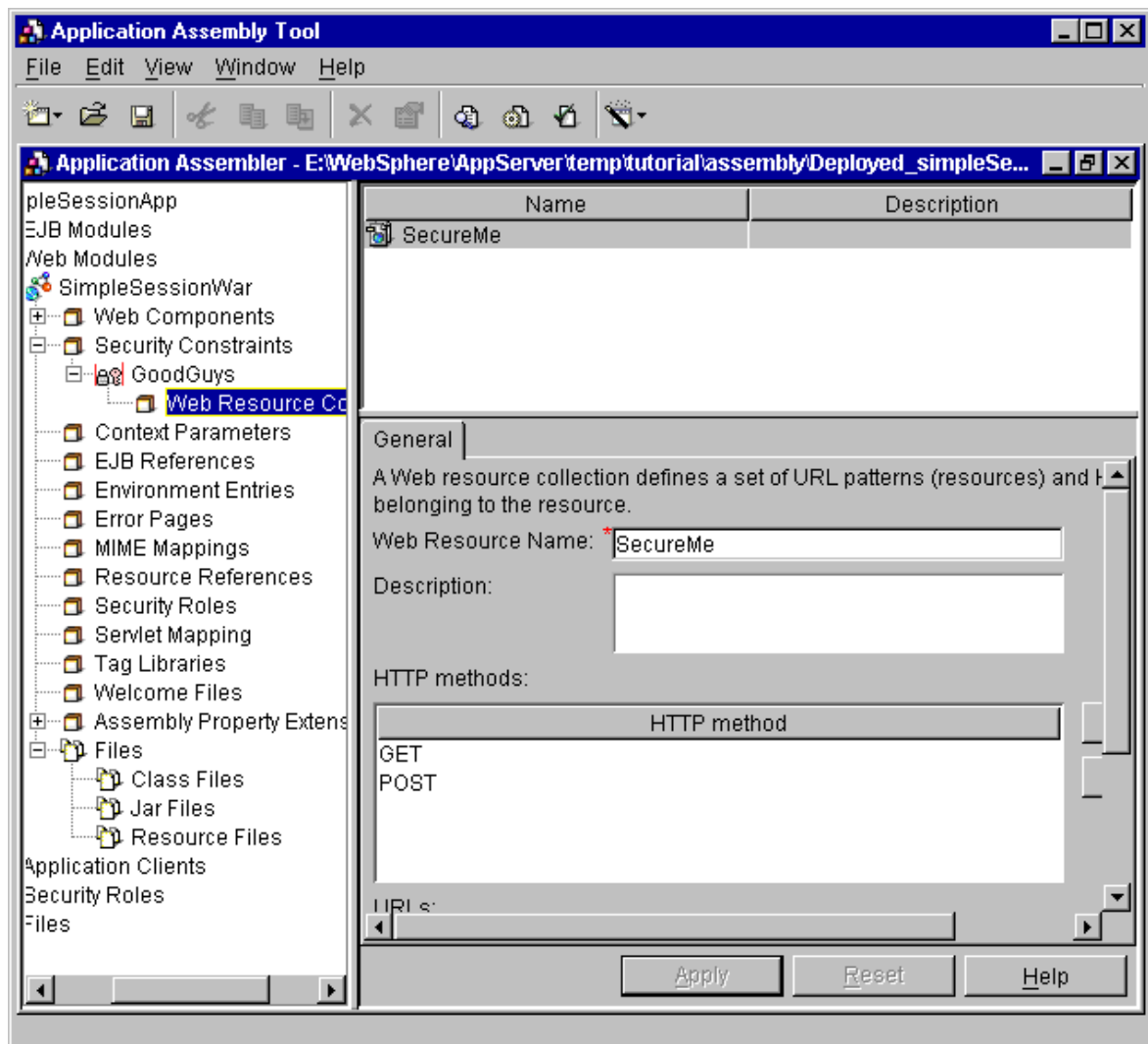
HTTP method	Add ...	Remove
GET		
POST		

URLs:

URL pattern	Add ...	Remove
/SimpleSession		

OK Apply Cancel Help

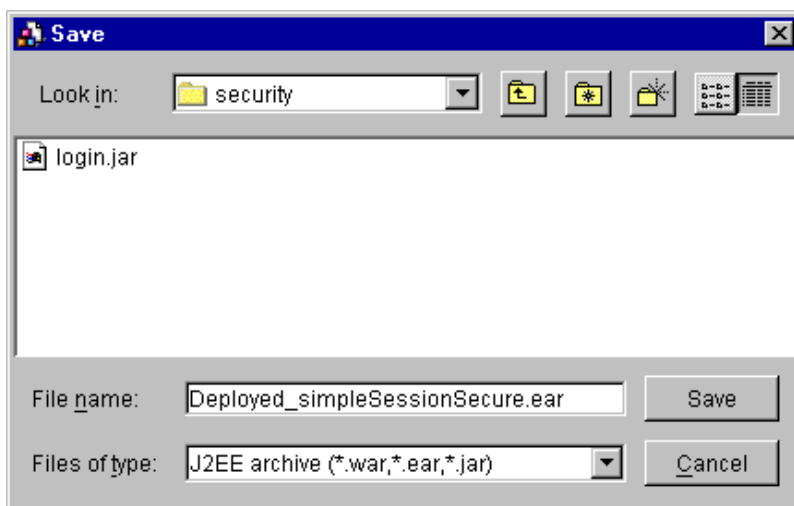
10. Click **OK**.



e. Click **OK**.

11. Save your .ear file using **File -> Save As** and save as

`product_installation_root/temp/tutorial/security/Deployed_simpleSessionSecure.ear`



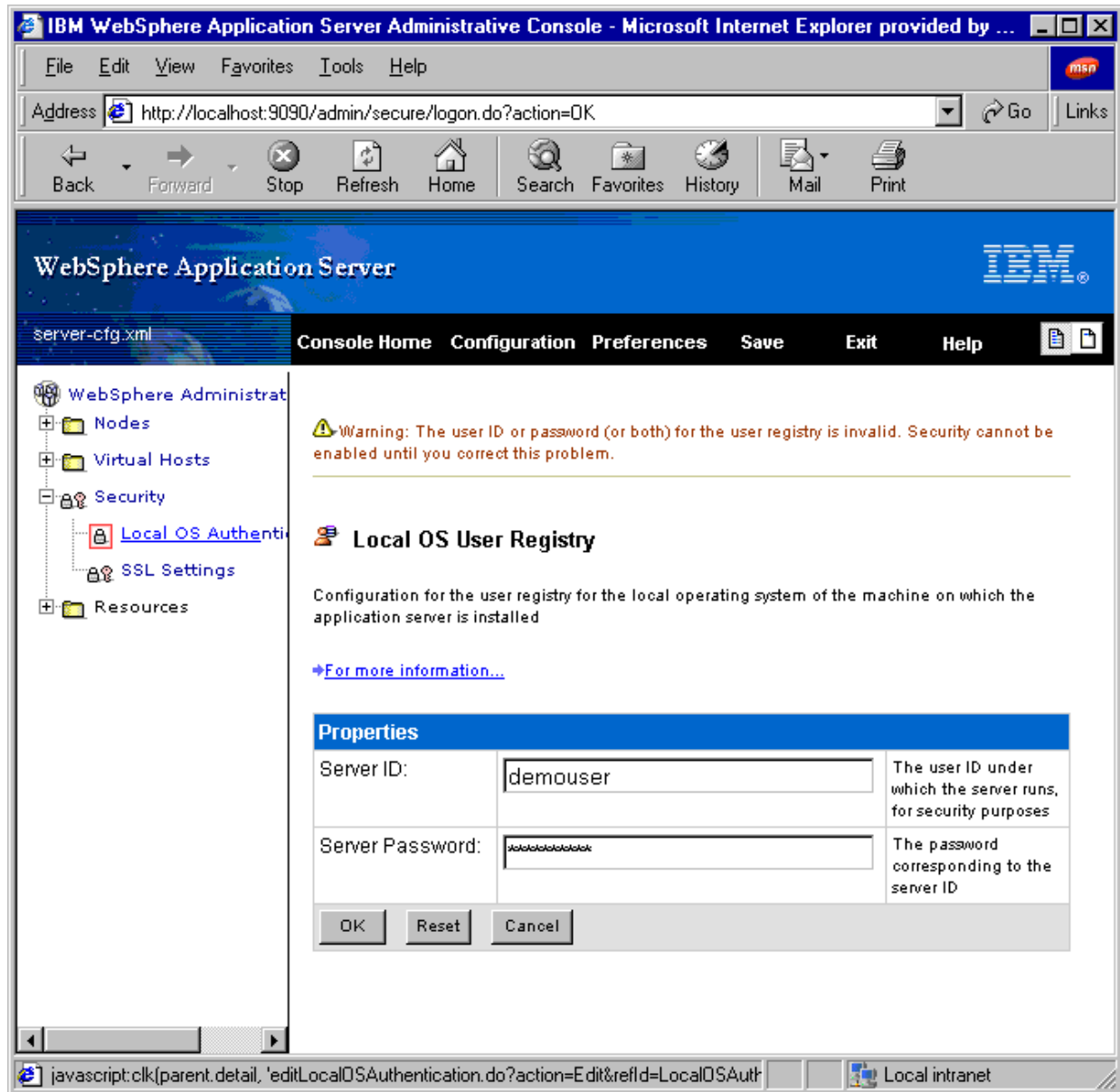
12. Exit the AAT.

## Enable security in the server runtime

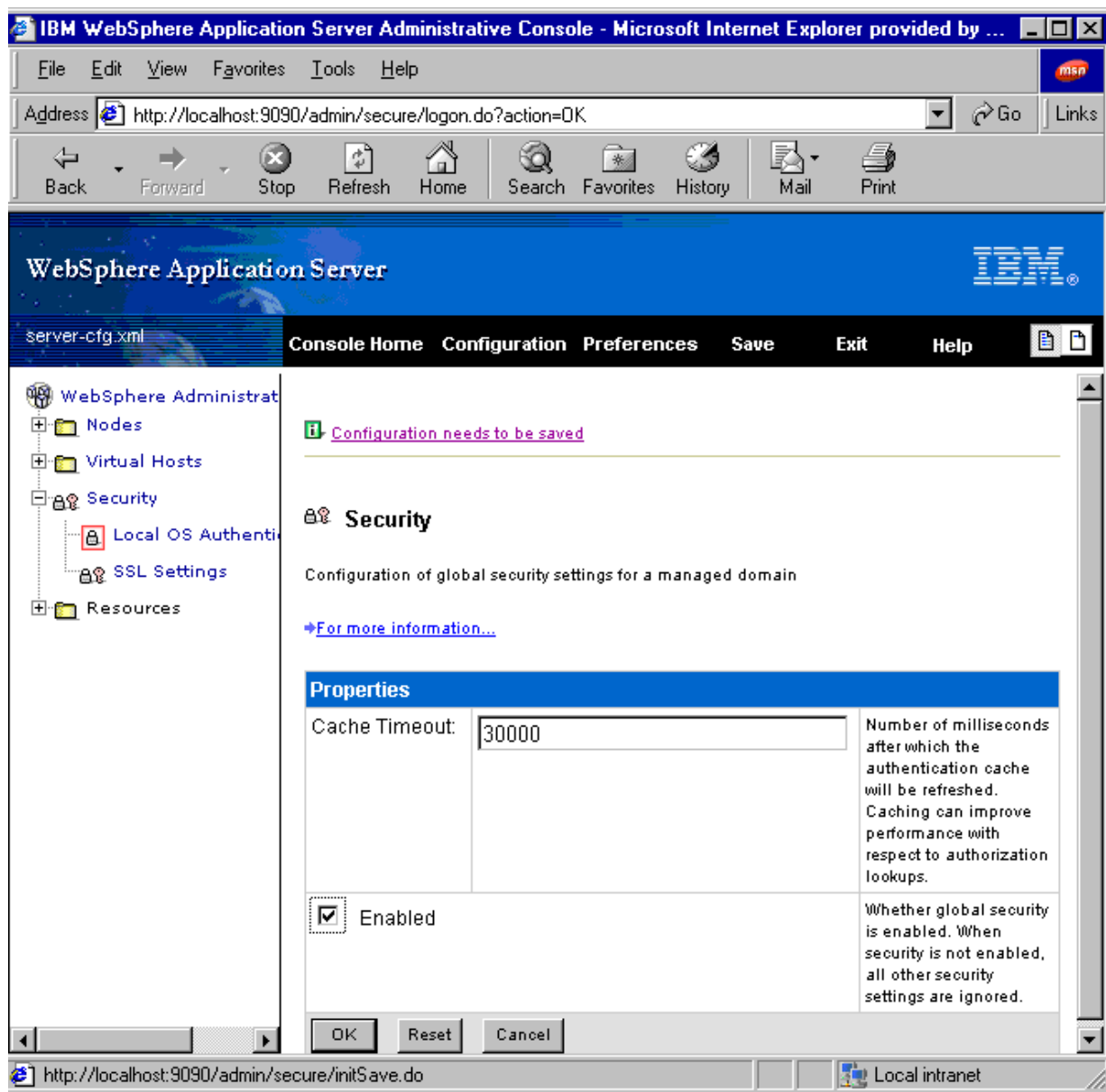
1. Start the application server, [as discussed in the Application deployment tutorial](#).
2. Open the administrative console
3. Enable security:
  1. In the tree view, expand **Security**.



2. Select **Local OS Authentication**.



3. Select **Local OS User Registry** in the right pane.
4. Change the Server ID: <operating system userid>
5. Change the Server Password: <operating system userid password>
6. Click **OK**.
7. In the topology tree, select **Security**.
8. Check **Enabled** in the right pane.



9. Click **OK**.
10. Save the configuration by clicking **Configuration needs to be saved** link at the top of the **Security** page.

## Install the secured application

The deployment tutorial discussed how to install an application using the administrative console installation wizard. This tutorial shows an alternate method, the application installer command line tool. A key benefit, as described in the documentation for the [application installer tool](#), is that the tool can be used with the application server stopped.

1. Open a system command prompt.
2. Change directory to:

`product_installation_root/bin`

3. Enter the command on a single line:

```
SEAppInstall -install ../temp/tutorial/security/Deployed_simpleSessionSecure.ear
             -ejbdeploy false
```

(Remember, on Windows systems, the forward slashes should be backslashes).

4. Answer the questions when prompted (the prompts are summarized here, but not verbatim):

**Remove the application and reinstall it?**

Yes

**precompile all JSP files**

No

**precompile individual Web Applications**

No

**default data source for the EJB JAR**

Push **Enter**

#### JNDI names

Push **Enter**, which indicates the default value: gs/hello

#### Subject Name for the Security Roles

Verify the default value is U:userid, where *userid* is your native operating system user ID. Accept the default value by pushing **Enter**.

#### Virtual Host name

Verify the default is set to **default\_host** and press **Enter**.

#### Denying all unprotected methods

Yes



```
MC Command Prompt
E:\WebSphere\AppServer\bin>SEAppInstall -install ..\temp\tutorial\security\Depl
IBM WebSphere Application Server Release 4, AEs
J2EE Application Installation Tool, Version 1.0
Copyright IBM Corp., 1997-2001

The -configFile option was not specified. Using E:\WebSphere\AppServer\config\s
You have chosen to install this application in interactive mode.
When prompted for information, pressing ENTER without entering
any information will cause the default value <shown in []>
to be used for that property.
When prompted for information, entering ? as the value will cause
the current value to be erased. This can be used to unset current
information.

Loading Server Configuration from E:\WebSphere\AppServer\config\server-cfg.xml
Server Configuration Loaded Successfully
Loading E:\WebSphere\AppServer\temp\tutorial\security\Deployed_simpleSessionSec
Getting Expansion Directory for EAR File
Expanding EAR File to E:\WebSphere\AppServer\installedApps\Deployed_simpleSessi
This EAR is already installed on this Server/Node. You can remove the applicati
Do you want remove and re-install the application <[Y/N]? y
Removed EAR From Server
Do you wish to precompile all JSPs in this application <[Y/N]? n
Do you wish to precompile individual Web Applications <[Y/N]? n

Please provide the following EJB Jar Binding Information
-----
EJB Jar: SimpleSessionEjb10.jar
-----
Default Datasource JNDI Name <optional> []:
Enterprise Bean: com.ibm.websphere.gettingstarted.ejbs.SimpleSessionBean
JNDI Name [gs/hello]:

No Resource References Defined as part of the Application

The following EJBs are available on this node.
Format: n) <Bean Name> [<JNDI Name>]
-----
1) com.ibm.websphere.examples.Inc.IncBean [IncBean]
2) WebSphereSamples.YourCo.Timeout.HistoryBean [History]
3) WebSphereSamples.YourCo.Timeout.AuditBean [Audit]
4) WebSphereSamples.YourCo.Timeout.LeaveBean [Leave]
5) WebSphereSamples.YourCo.Timeout.AccessBean [Access]
6) WebSphereSamples.Increment.IncrementBean [WSsamples/IncrementHome]
7) Hello Session EJB [WSsamples/HelloEJBHome]
8) Transfer Session EJB [WSsamples/TransferHome]
9) Account CMP EJB [WSsamples/AccountHome]
```

## Regenerate the Web server plug-in and save the server configuration

Installing an application requires regenerating the configuration for the WebSphere plug-in for the Web server. This task was discussed in the [Application deployment tutorial](#), but the method used there required starting the server and opening the administrative console.

Try an alternate method that does not require the server to be running:

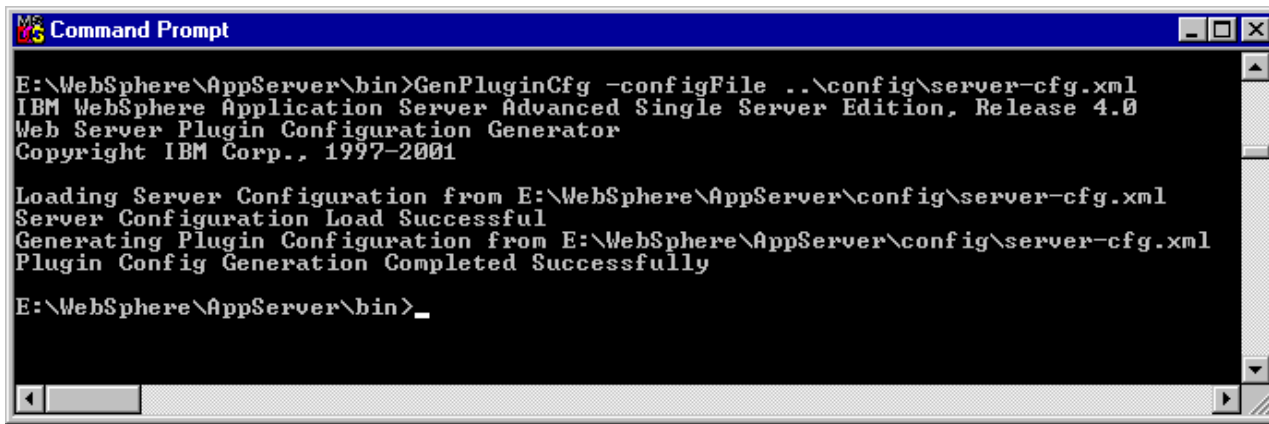
1. Open a system command prompt.
2. Run the GenPluginCfg script, pointing it to the default server XML configuration file.

The script is located at:

`product_installation_root/bin/GenPluginCfg.sh|bat`

(Run the script as you usually would run a script or bat file on your particular operating system). Use the argument:

```
-configFile ../config/server-cfg.xml
```



```
MS-DOS Command Prompt
E:\WebSphere\AppServer\bin>GenPluginCfg -configFile ../config/server-cfg.xml
IBM WebSphere Application Server Advanced Single Server Edition, Release 4.0
Web Server Plugin Configuration Generator
Copyright IBM Corp., 1997-2001

Loading Server Configuration from E:\WebSphere\AppServer\config\server-cfg.xml
Server Configuration Load Successful
Generating Plugin Configuration from E:\WebSphere\AppServer\config\server-cfg.xml
Plugin Config Generation Completed Successfully

E:\WebSphere\AppServer\bin>
```

## Stop the server and start it again

In this case, the server should be stopped already. Confirm this.

Start the server again, [as you learned to do in the Application deployment tutorial](#).

## Ensure the application, application server, and Web server are running

Recall learning this step in the [Application testing tutorial](#).

## Test the Web client

1. Test the Web client the same way you tested the Web client for the unsecured simpleSession application, [as discussed in the Application testing tutorial](#). That is, in a Web browser, type the URL:

<http://hostname:9080/gettingstarted3/SimpleSession?msg=Hi>

2. When the login screen is displayed, enter your native operating system user ID and password.

Notice that your login credentials flowed to the EJB container as well because the above procedure established authorization of the EJB methods, as well as of the servlet.

## Test the Java client

This, too, is similar to the way you did it in the Application testing tutorial, except you will reference the secured .ear file in the LaunchClient command.

1. Open a system command prompt.
2. Change directory to:

[product\\_installation\\_root](#)/bin

3. Enter the command:

```
launchClient ../temp/tutorial/security/Deployed_simpleSessionSecure.ear
```

(Remember, on Windows systems, the forward slashes should be backslashes).

The client will proceed to the point of looking the EJB home before it prompts the user to log in.

```
MS Command Prompt - launchClient ..\temp\tutorial\security\Deployed_simpleSessionSecure.ear
E:\WebSphere\AppServer\bin>launchClient ..\temp\tutorial\security\Deployed_
IBM WebSphere Application Server, Release 4.0
J2EE Application Client Tool, Version 1.0
Copyright IBM Corp., 1997-2001

WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has co
WSCL0014I: Invoking the Application Client class com.ibm.websphere.gettings
=====
SimpleSessionClient
=====
EJB name    -> java:comp/env/ejb/SimpleSession
msg         -> hello world
iterations -> 1

ITERATION: 0
  creating initial context ...
  looking up the EJBs home @ java:comp/env/ejb/SimpleSession ...
```

4. When the login pop-up is displayed, enter your native operating system user ID and password.

```
MS Command Prompt
E:\WebSphere\AppServer\bin>GenPluginCfg -configFile ..\config\server-cfg.xml
IBM WebSphere Application Server Advanced Single Server Edition, Release 4.0
Web Server Plugin Configuration Generator
Copyright IBM Corp., 1997-2001

Loading Server Configuration from E:\WebSphere\AppServer\config\server-cfg.xml
Server Configuration Load Successful
Generating Plugin Configuration from E:\WebSphere\AppServer\config\server-cfg.xml
Plugin Config Generation Completed Successfully

E:\WebSphere\AppServer\bin>_
```

## Disable security in the application server runtime

1. Start the application server, [as discussed previously](#).
2. Open the administrative console, [as discussed in the Application deployment tutorial](#).
3. Disable security.
  1. In the console tree view, select **Security**.
  2. Deselect the **Enabled** check box.
  3. Click **OK**.
4. Save the server configuration, using the dialog displayed by clicking the prompt "Configuration needs to be saved" at the top of the Security page.
5. Stop the application server and start it again, [as discussed previously](#).

Do not forget to go back into the console and disable security **if** you are sharing the console with a colleague who will not know the correct ID and password the next time he or she tries to open the console!

## What's next?

Now that you have assembled, deployed, and secured a Session bean application, you might want to try your hand at a CMP application. The [Advanced application assembly and deployment tutorial](#) describes how to do so.

## 6.7.5: Advanced application assembly and deployment tutorial -- CMP bean application

During this tutorial, you will assemble and deploy a CMP bean application. This simple J2EE application consists of a single CMP entity bean and a Java application client. The client creates an EJB, invokes a remote method, then destroys it. We will start with an EJB 1.0 jar file.

### Prerequisites

This tutorial assumes knowledge of the Application assembly tutorial and Application deployment tutorial, which focused on a Session bean application, but are also applicable for BMP bean applications. Indeed, many steps of this tutorial are abbreviated, with links to the steps in the assembly and deployment tutorials in case you need to refresh your memory.



This tutorial puts more attention on the new steps for CMP bean support, namely, creating database tables and configuring data sources in the application server runtime.

You will need a **working, supported DB2 or Oracle database** to verify your results using the tutorial steps as they are written. Of course, you can always use this tutorial to practice the steps, knowing that your data access is not set up yet.

### Overview of steps (60 minutes)

1. [Obtain the example code](#)
2. [Start the Application Assembly Tool](#)
3. [Assemble an EJB module, converting from EJB 1.0 JAR to EJB 1.1](#)
4. [Assemble an application client module](#)
5. [Assemble a J2EE application](#)
6. [Exit the Application Assembly Tool](#)
7. [Start the server](#)
8. [Open the administrative console](#)
9. [Configure a data source](#)
10. [Install the application](#)
11. [Regenerate the Web server plug-in configuration](#)
12. [Stop the application server and start it again](#)
13. [Create database table or tablespace](#)
14. [Ensure that the application, application server, and Web server are running](#)
15. [Test the Java application client](#)

### Paths through the tutorial

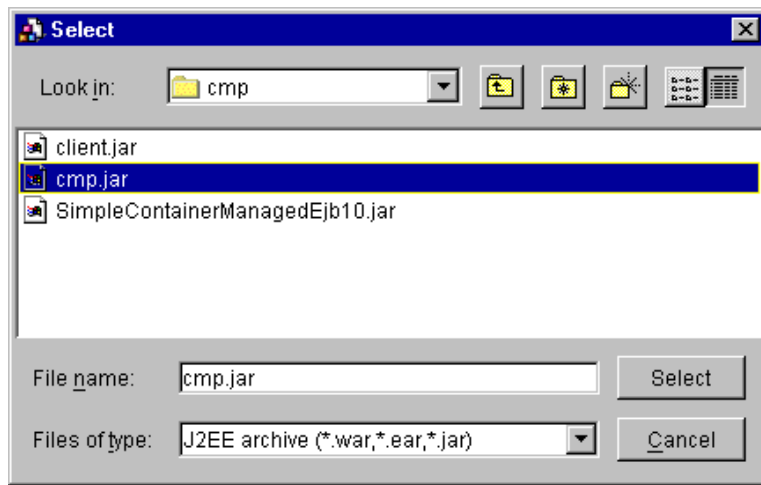
- **Quick path:** If your objective is to practice these steps as quickly as possible, skip the explanations marked with the  graphic.
- **Maximum learning path:** If your objective is to understand and explore the product capabilities, read the  notes and browse the links they provide to additional documentation.

### Start the Application Assembly Tool

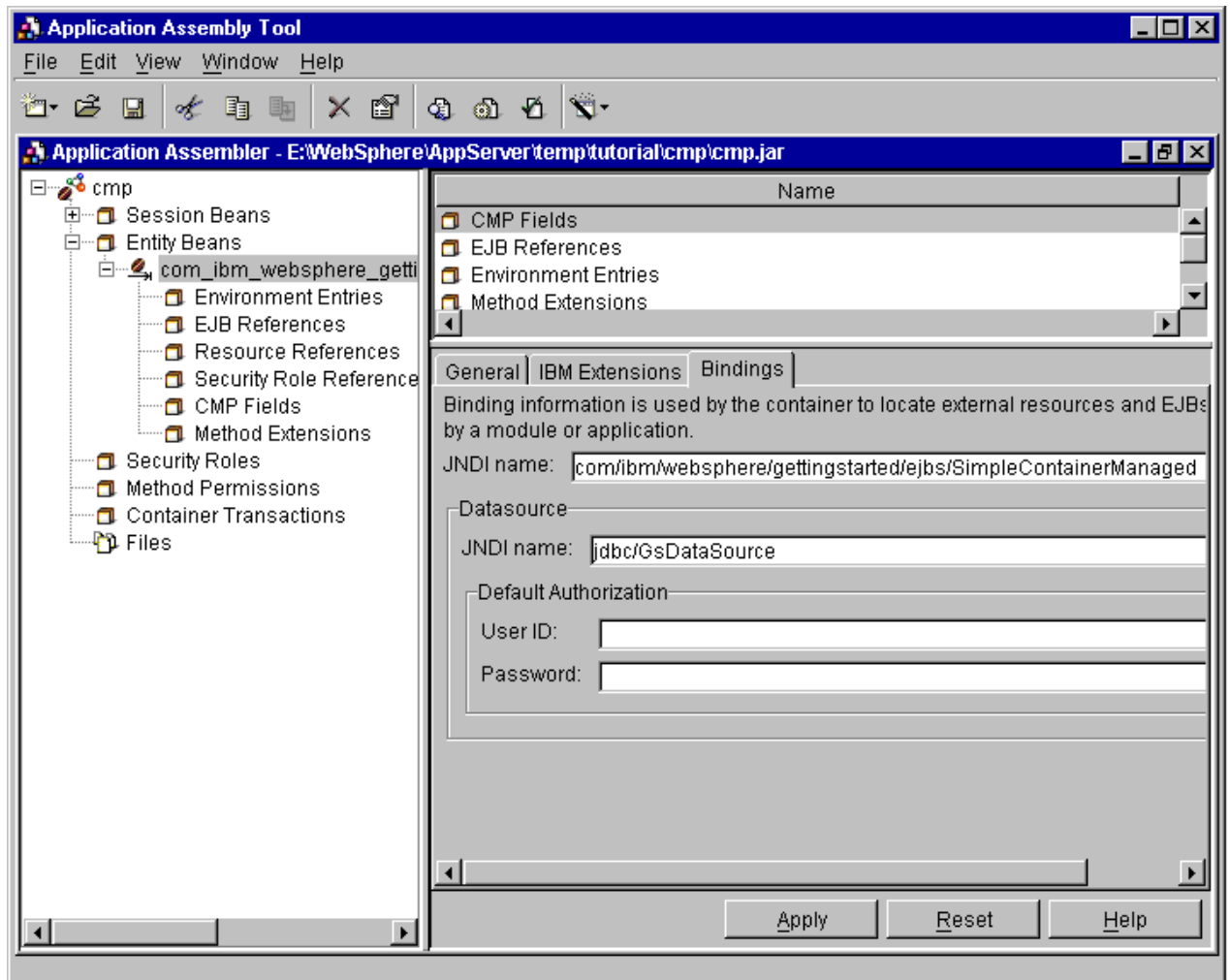
Start the Application Assembly Tool, [as discussed in the Application assembly tutorial](#).

### Assemble an EJB module, converting from EJB 1.0 JAR to EJB 1.1

1. On the **Welcome to Application Assembly Tool** panel:
  - a. Click the **Existing** tab.
  - b. Click **Browse** next to the File name field.
  - c. Navigate to the **cmp.jar** file that you previously downloaded and select **Select**.



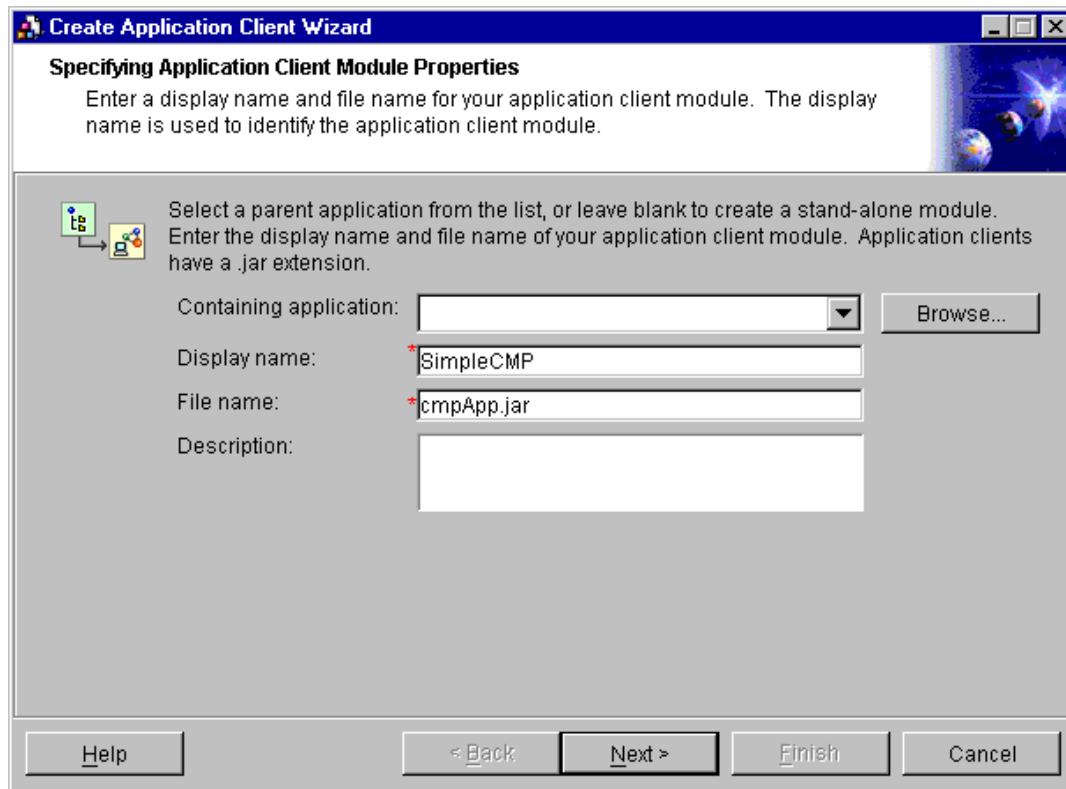
- d. Select **OK**.
2. Click **OK** at the "...please specify the dependent classpath..." dialog.  
This will convert the EJB 1.0 JAR file into an EJB 1.1 JAR file.
3. Set the global JNDI name of the bean, and bind the data source:
  - a. In the topology tree, expand the **Entity Beans** folder.
  - b. Select the bean labeled:  
**com\_ibm\_websphere\_gettingstarted\_ejbs\_SimpleContainerManaged**
  - c. Click it to display its properties.
  - d. Select the **Bindings** tab.
    1. Set the **JNDI Name** to:  
**com/ibm/websphere/gettingstarted/ejbs/SimpleContainerManagedHome**  
You can use a shorter name as long as you guarantee that it is unique among all installed applications. Using the fully qualified package name of the bean will accomplish this, usually.
    2. Enter Datasource JNDI Name: **jdbc/GsDataSource**



3. If using an Oracle database, enter the Oracle admin userid and password under **Default Authorizations** (this is not required for DB2).
4. Click **Apply**.
5. Save the JAR file using **File -> Save As...** and name it **cmp11.jar**.

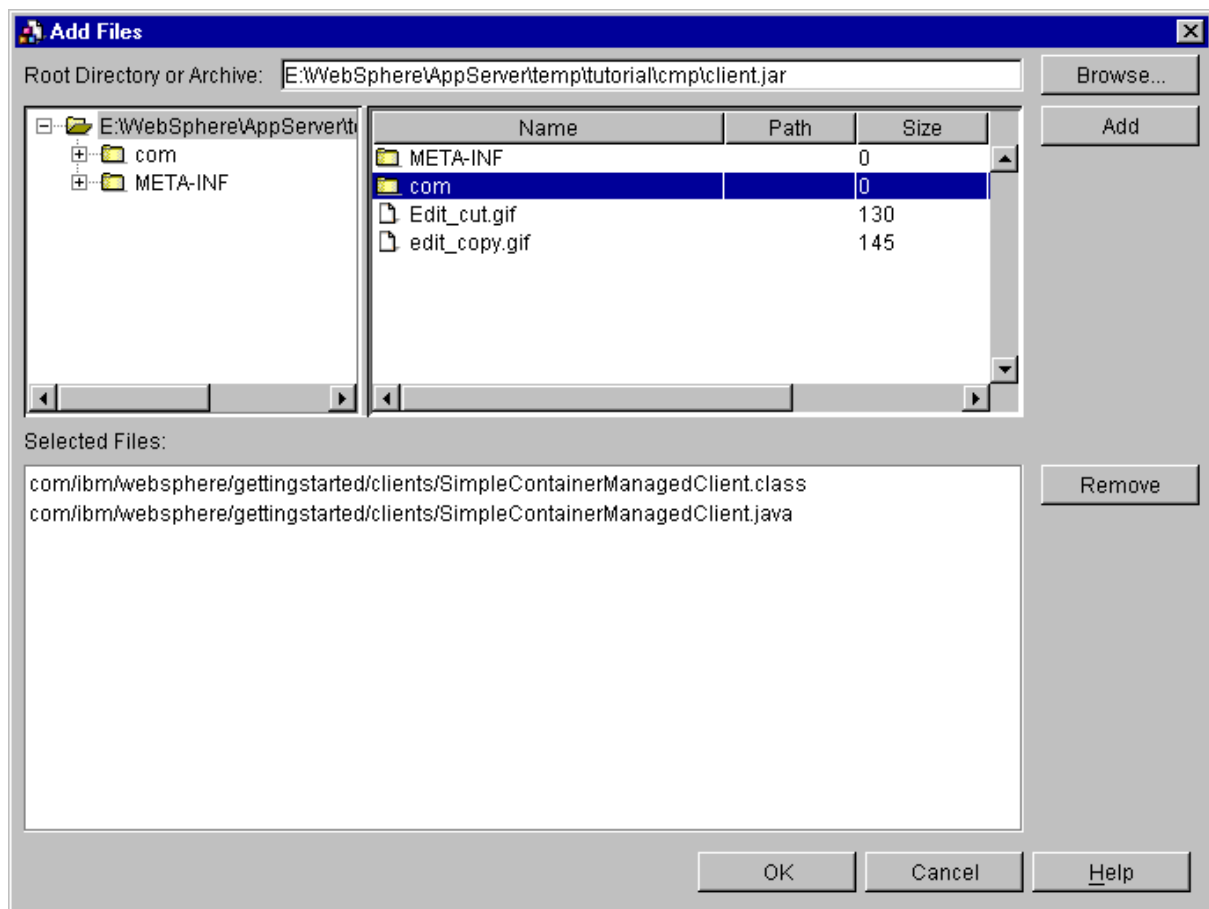
## Assemble an application client module

1. Create a new application client using **File -> Wizards -> Create Application Client Wizard**.
2. On the **Specifying Application Client Module Properties** panel:
  - a. Enter the following:
    - Display Name: **SimpleCMP**
    - File Name: **cmpApp.jar**

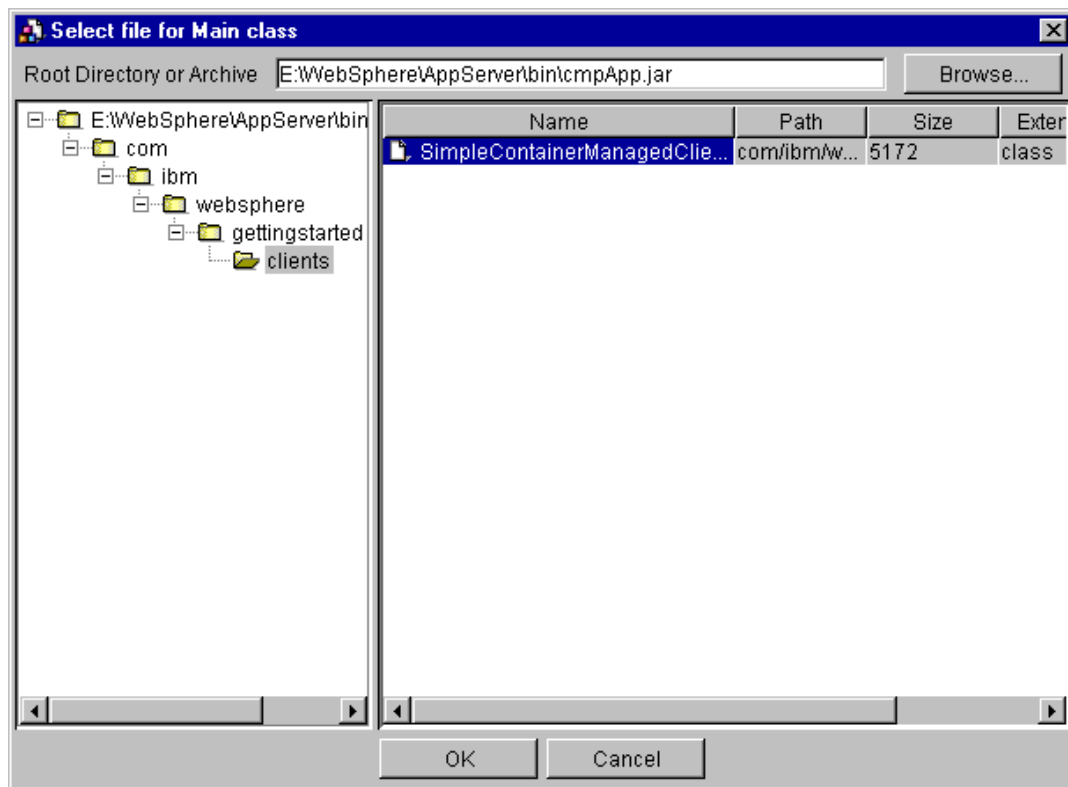


- b. Click **Next**.
3. Click **Add...** on the **Adding Files** panel.
  - a. Click **Browse**.
  - b. Navigate to the **client.jar** that was previously downloaded, highlight the file and click **Select**.
  - c. Select the **com** folder (in the right hand pane).
  - d. Click **Add**.



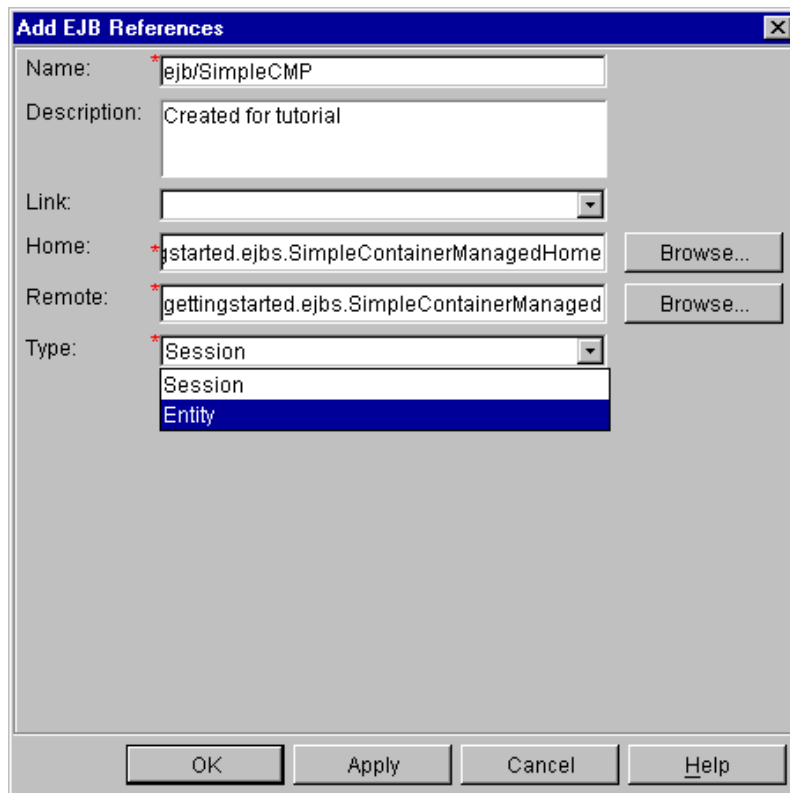


- e. Click **OK**.
- f. Click **Next**.
4. On the **Specifying Additional Application Client Module Properties** panel:
  - a. Enter **cmp11.jar** for **Classpath**.
  - b. Click **Browse** next to **Main Class** field.
  - c. Expand the topology tree in the left pane as far as it goes and select **clients**.
  - d. Select **SimpleContainerManagedClient.class** in the right pane.



- e. Click **OK**.

- f. Click **Next**.
5. Click **Next** on the **Choosing Application Client Module Icons** panel.
6. Click **Add** on the **Adding EJB References** panel:
- Enter the following:
    - Name: **ejb/SimpleCMP**
    - Home: **com.ibm.websphere.gettingstarted.ejbs.SimpleContainerManagedHome**
    - Remote: **com.ibm.websphere.gettingstarted.ejbs.SimpleContainerManaged**
  - Set Type: **Entity**

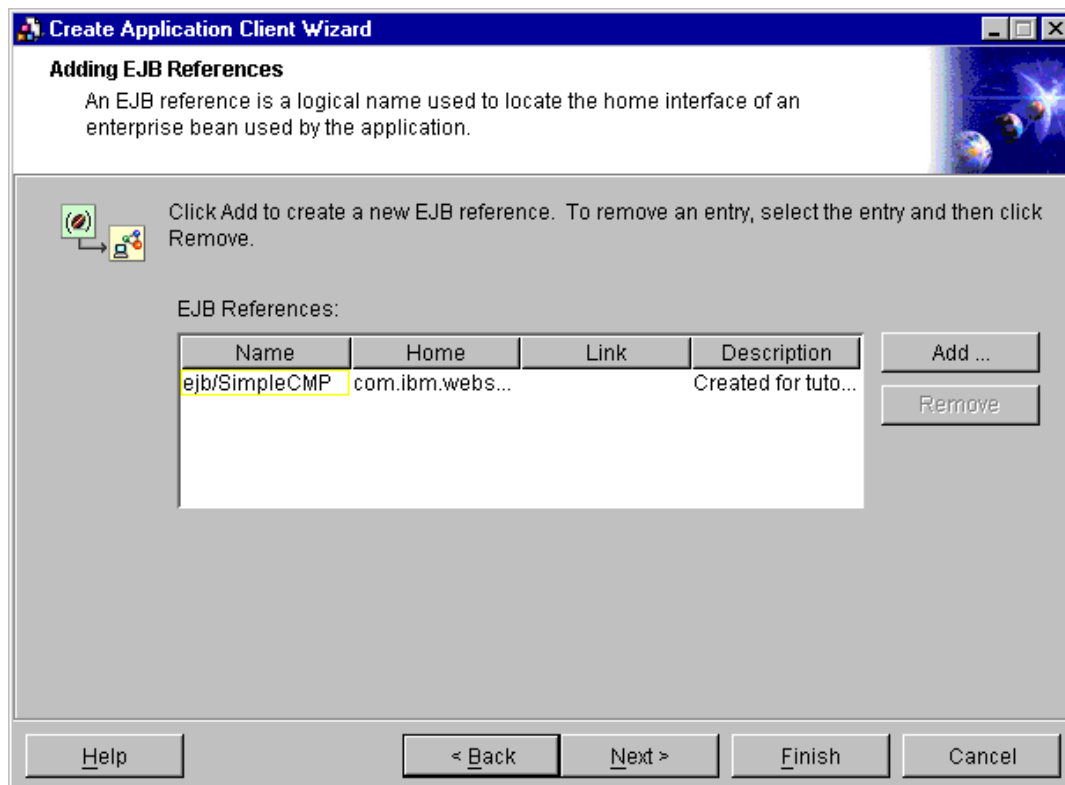


The 'Add EJB References' dialog box contains the following fields and controls:

- Name:** Text box with 'ejb/SimpleCMP' entered.
- Description:** Text box with 'Created for tutorial' entered.
- Link:** Dropdown menu (empty).
- Home:** Text box with 'com.ibm.websphere.gettingstarted.ejbs.SimpleContainerManagedHome' entered, and a 'Browse...' button.
- Remote:** Text box with 'com.ibm.websphere.gettingstarted.ejbs.SimpleContainerManaged' entered, and a 'Browse...' button.
- Type:** Dropdown menu with 'Session' selected, and a list showing 'Session' and 'Entity' (where 'Entity' is highlighted).

At the bottom are buttons for **OK**, **Apply**, **Cancel**, and **Help**.

- c. Click **OK**.
- d. Verify that your EJB reference is displayed in the EJB References list.



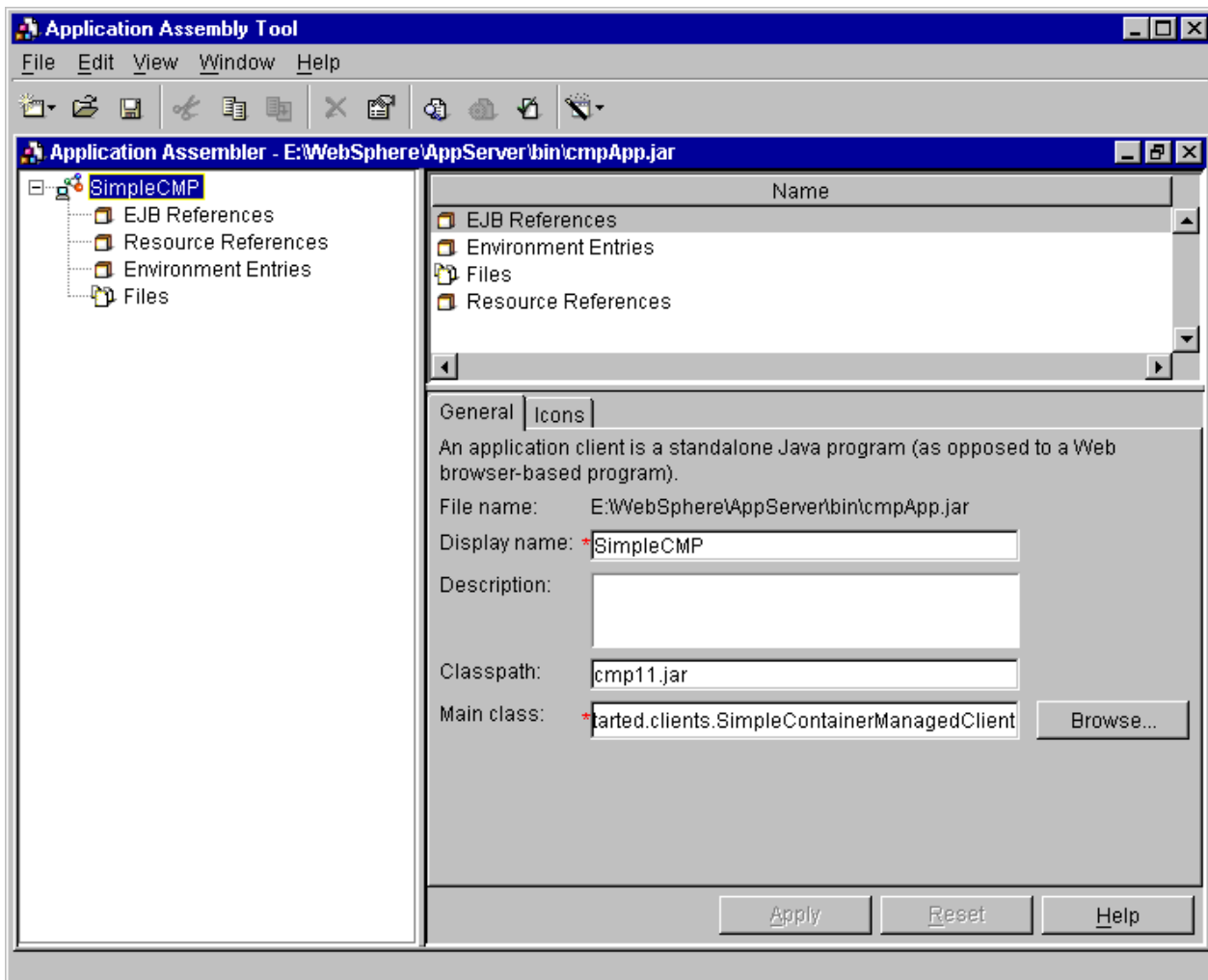
The 'Create Application Client Wizard' screen for 'Adding EJB References' includes:

- Header:** 'Create Application Client Wizard' with standard window controls.
- Section:** 'Adding EJB References'.
- Text:** 'An EJB reference is a logical name used to locate the home interface of an enterprise bean used by the application.'
- Instructions:** 'Click Add to create a new EJB reference. To remove an entry, select the entry and then click Remove.'
- EJB References Table:**

Name	Home	Link	Description
ejb/SimpleCMP	com.ibm.webs...		Created for tuto...
- Buttons:** 'Add ...' and 'Remove' buttons to the right of the table.
- Footer:** 'Help', '< Back', 'Next >', 'Finish', and 'Cancel' buttons.

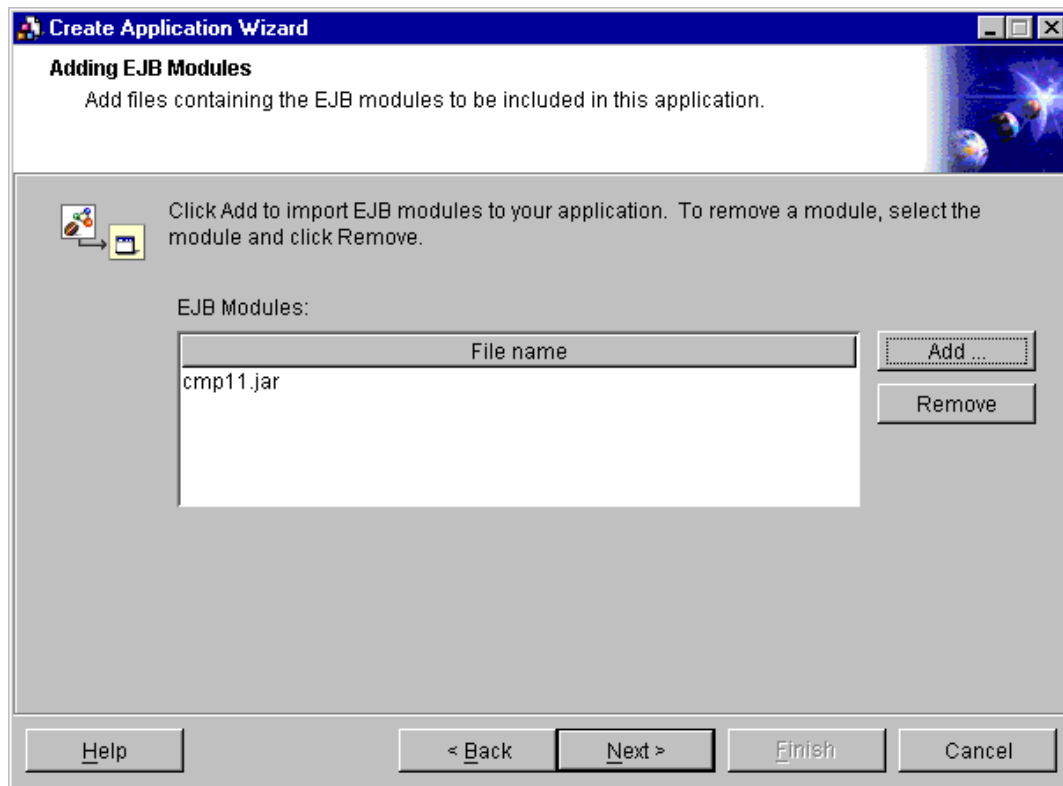
- e. Click **Next**.

7. Click **Next** on the **Adding Resource References** panel.
8. Click **Finish** on the **Adding Environment Entries** panel.
9. Save the application client jar file using **File -> Save As**. Name it **cmpApp.jar**.

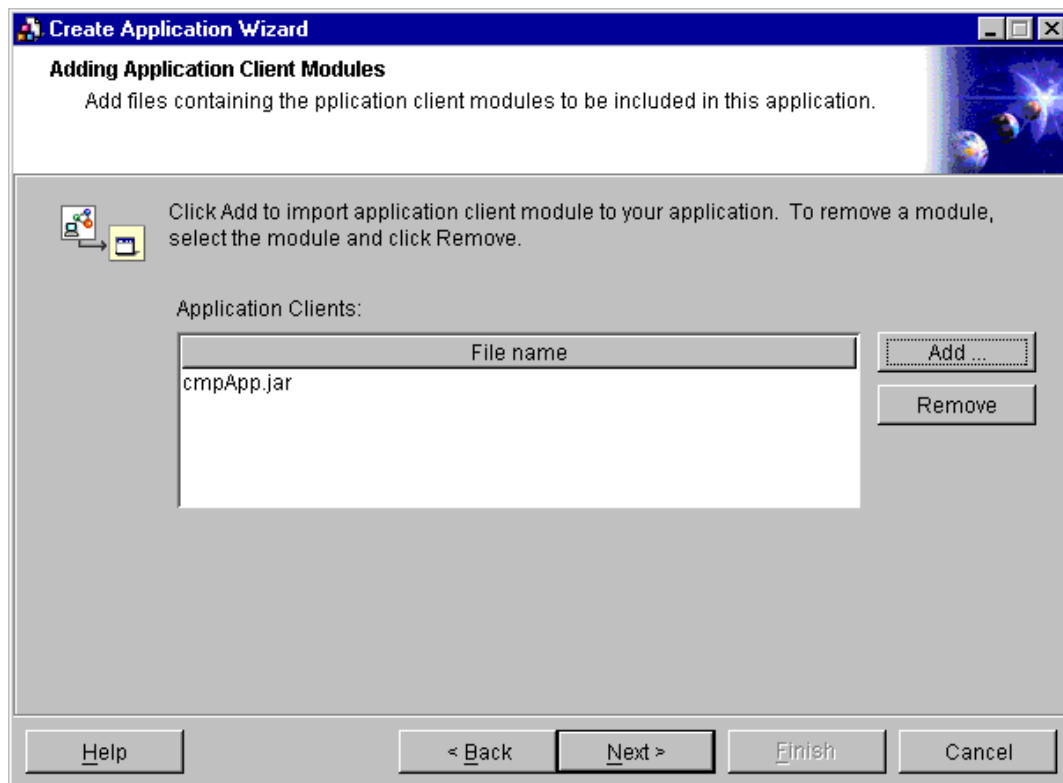


## Assemble a J2EE application

1. Using AAT, create an .ear by clicking **File -> Wizards -> Create Application Wizard**.
2. On the **Specifying Application Properties** panel:
  - a. Enter the following:
    - Display Name: **CmpApp**
    - File Name: **cmp.ear**
  - b. Click **Next**.
3. Click **Next** on the **Adding Supplementary Files** panel.
4. Click **Next** on the **Choosing Application Icons** panel.
5. Click **Add...** on the **Adding EJB Modules** panel:
  - a. Navigate to your **cmp11.jar** file, highlight the file and click **Open**.
  - b. Click **OK** in the resulting dialog.

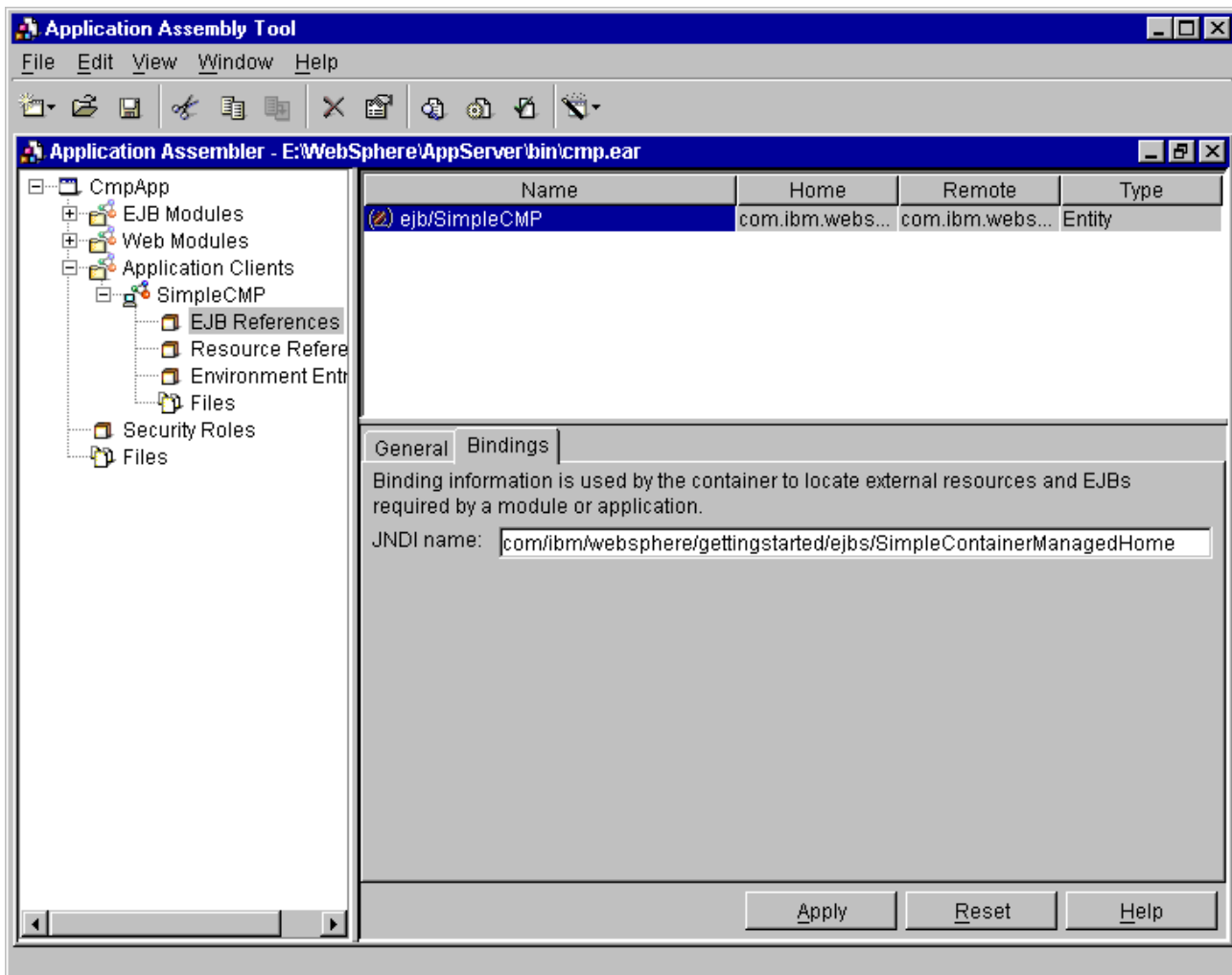


- c. Click **Next**.
- 6. Click **Next** on the **Adding Web Modules** panel.
- 7. Click **Add...** on the **Adding Application Client Module** panel:
  - a. Navigate to your **cmpApp.jar** file, highlight the file and click **Open**.
  - b. Click **OK** in the resulting dialog.

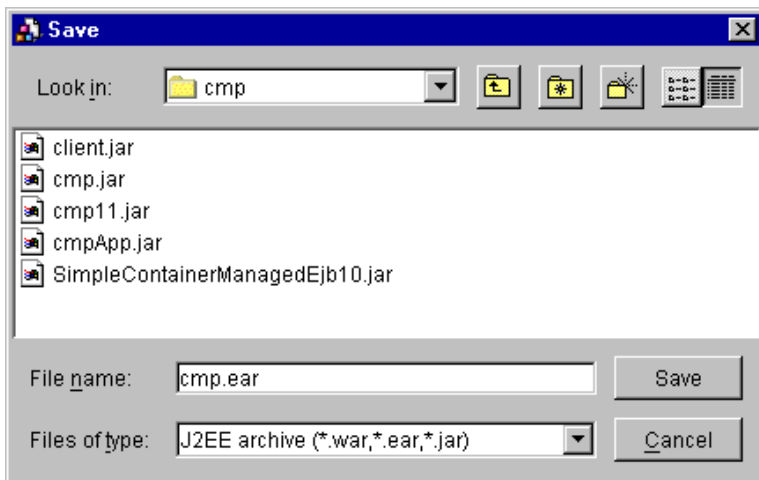


- c. Click **Next**.
- 8. Click **Finish** on the **Adding Security Roles** panel.
- 9. Bind the EJB reference:
  - a. Expand **Application Clients**
  - b. Expand **SimpleCMP**
  - c. Select **EJB References**.
  - d. Select in the right pane the particular reference to your session bean called **ejb/SimpleCMP**.

- e. Select the **Bindings** tab.
- f. Enter the following for the JNDI name:  
**com/ibm/websphere/gettingstarted/ejbs/SimpleContainerManagedHome**



10. Click **Apply**.
11. Save the .ear file by clicking **File -> Save As**. Name it **cmp.ear**.



12. Generate code for deployment of the .ear file:
  - a. Select the .ear file by selecting the top entry in the topology tree (named CmpApp).
  - b. Right-click it.
  - c. Select **Generate Code for Deployment**.
  - d. In the Generate Code for Deployment panel:
    1. Accept the default for the Deployed module location.
    2. Select the appropriate database type from the list.
  - e. Click **Generate Now**.
  - f. When the deployment is complete, close this window.

The file containing the code generated for deployment will be created called **Deployed\_cmp.ear**.

It might take a while to generate the code for deployment. You can proceed to configure your data source, as described in the next few steps.

**i** The terminology is potentially confusing here. The file containing the code generated for deployment on the application server is called the `Deployed_filename`, although it has not been deployed on the server yet. At this point in the application assembly and deployment procedure, the name "Deployed\_filename" is more anticipatory than accurate. However, the name will be suitable shortly. After you deploy the "Deployed" file, the name will be useful for distinguishing the file from the pre-deployed version (the one for which you have neither generated code for deployment nor installed on the application server).

## Exit the Application Assembly Tool

Recall this step from the [Application assembly tutorial](#).

## Start the server

Recall this step from the [Application deployment tutorial](#).

## Open the administrative console

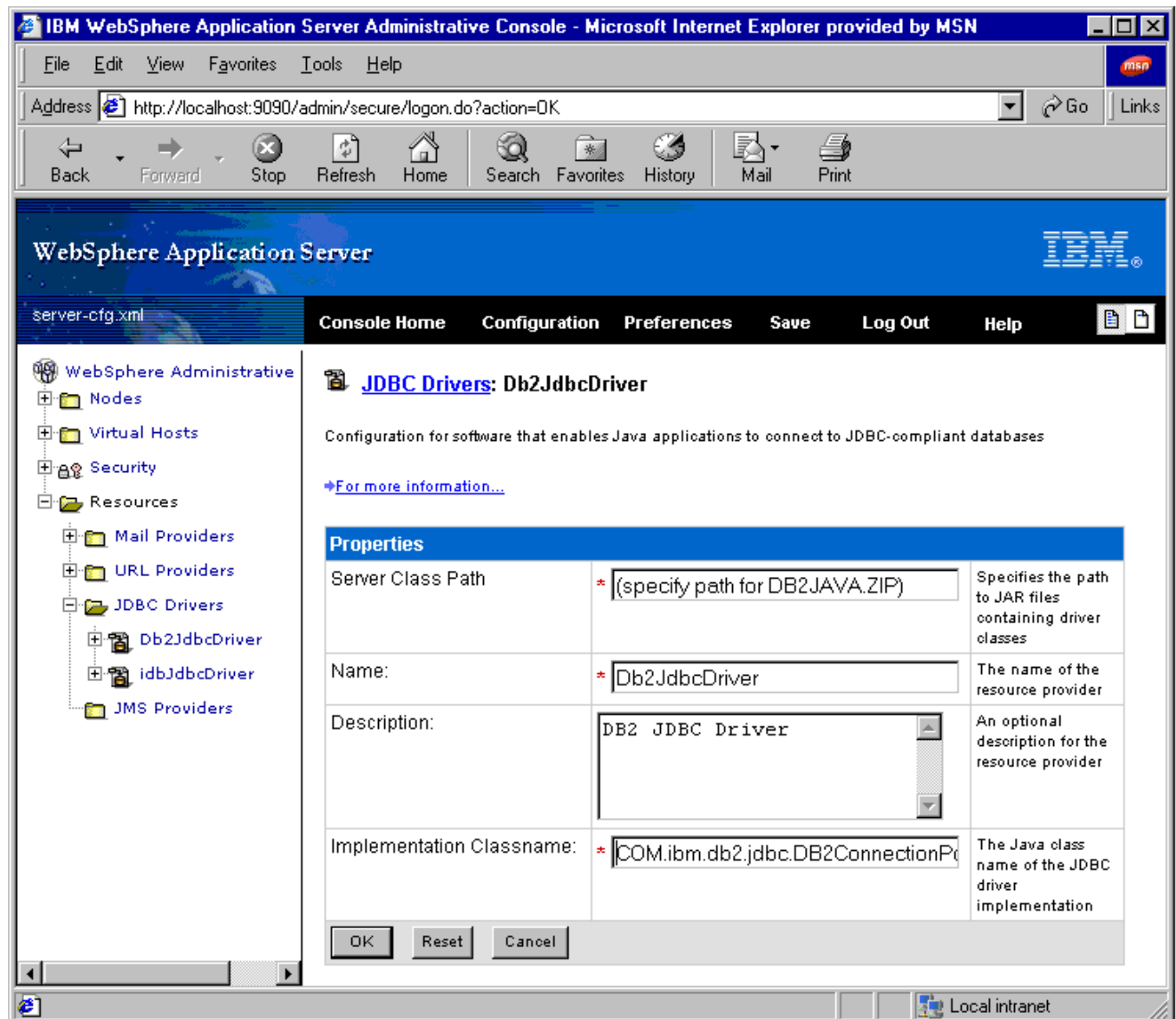
Recall this step from the [Application deployment tutorial](#).

## Configure a data source

The instructions differ according to whether you are using a DB2 database or [Oracle database](#).

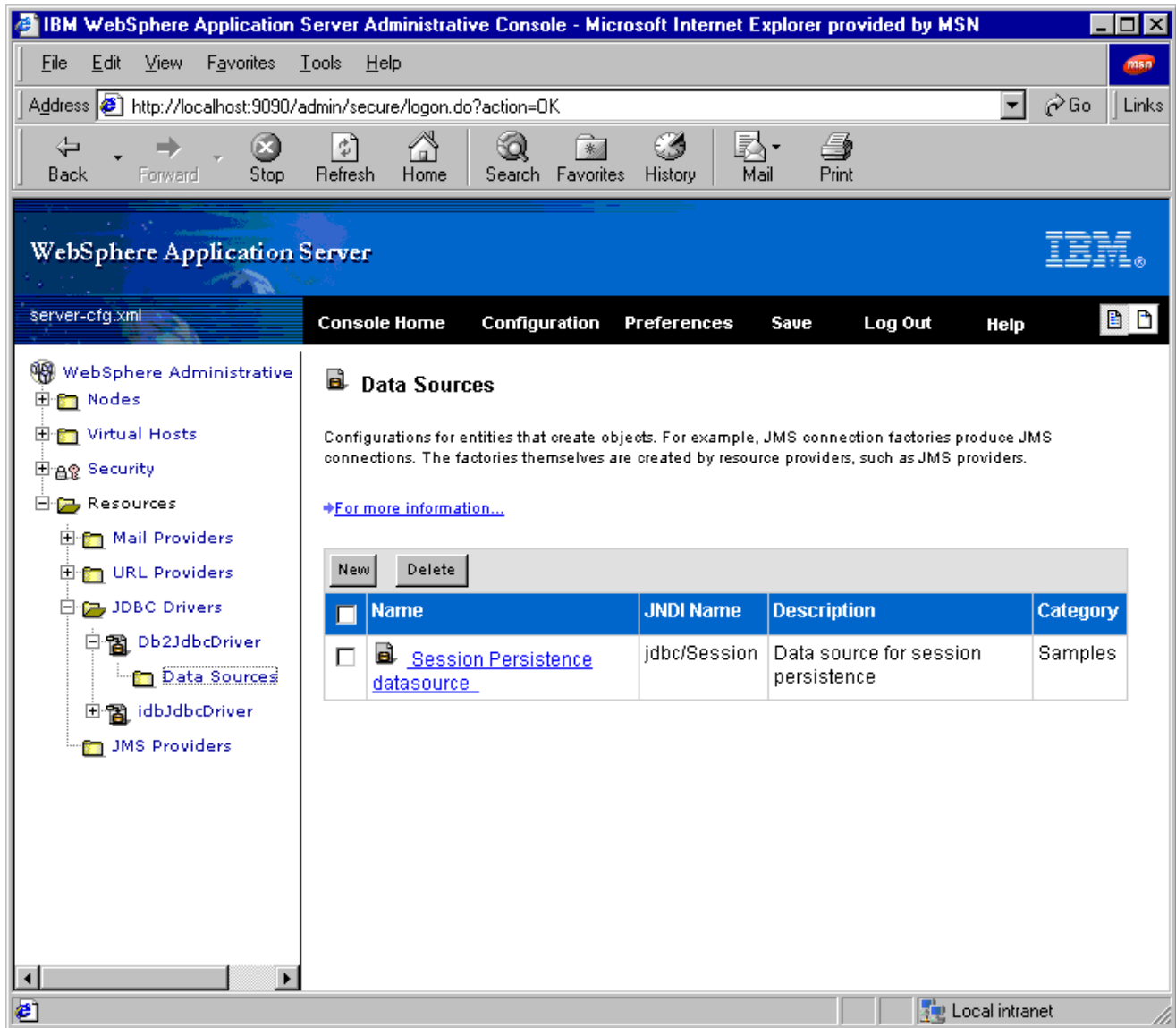
### DB2 Database:

1. In the administrative console, update the Db2JdbcDriver:
  - a. In the topology tree, expand **Resources -> JDBC Drivers**.



- b. Select **Db2JdbcDriver**.
- c. Modify Server Class Path to show the location of the JAR file containing the DB2 JDBC driver classes (db2java.zip) on your system.

- **NT or Windows 2000:** db2java.zip file is located in the **sqllib\java** directory
  - **Unix:** db2java.zip file is located in the **sqllib/java12** directory
- d. Click **OK**.
- e. Save the configuration by clicking **Save**.
2. In the administrative console, configure a new data source:
- In the tree view, expand **Resources -> JDBC Drivers -> Db2JdbcDriver**.
  - Select **Data Sources**.



- c. Select **New**.
- d. Enter the following values. (If a field value is not specified, leave the field blank).
- Name: **GettingStartedDataSource**
  - JNDI Name: **jdbc/GsDataSource**

IBM WebSphere Application Server Administrative Console - Microsoft Internet Explorer provided by MSN

File Edit View Favorites Tools Help

Address <http://localhost:9090/admin/secure/logon.do?action=OK> Go Links

Back Forward Stop Refresh Home Search Favorites History Mail Print

## WebSphere Application Server

server-cfg.xml

Console Home Configuration Preferences Save Log Out Help

WebSphere Administrative

- Nodes
- Virtual Hosts
- Security
- Resources
  - Mail Providers
  - URL Providers
  - JDBC Drivers
    - Db2JdbcDriver
      - Data Sources
    - idbJdbcDriver
  - JMS Providers

### Data Sources: Data Source

Configuration for a data source, which is a pool of managed database connections. All of the Persistence attributes are added as string from podata association class names from the WebSphere XMLConfig DTD.

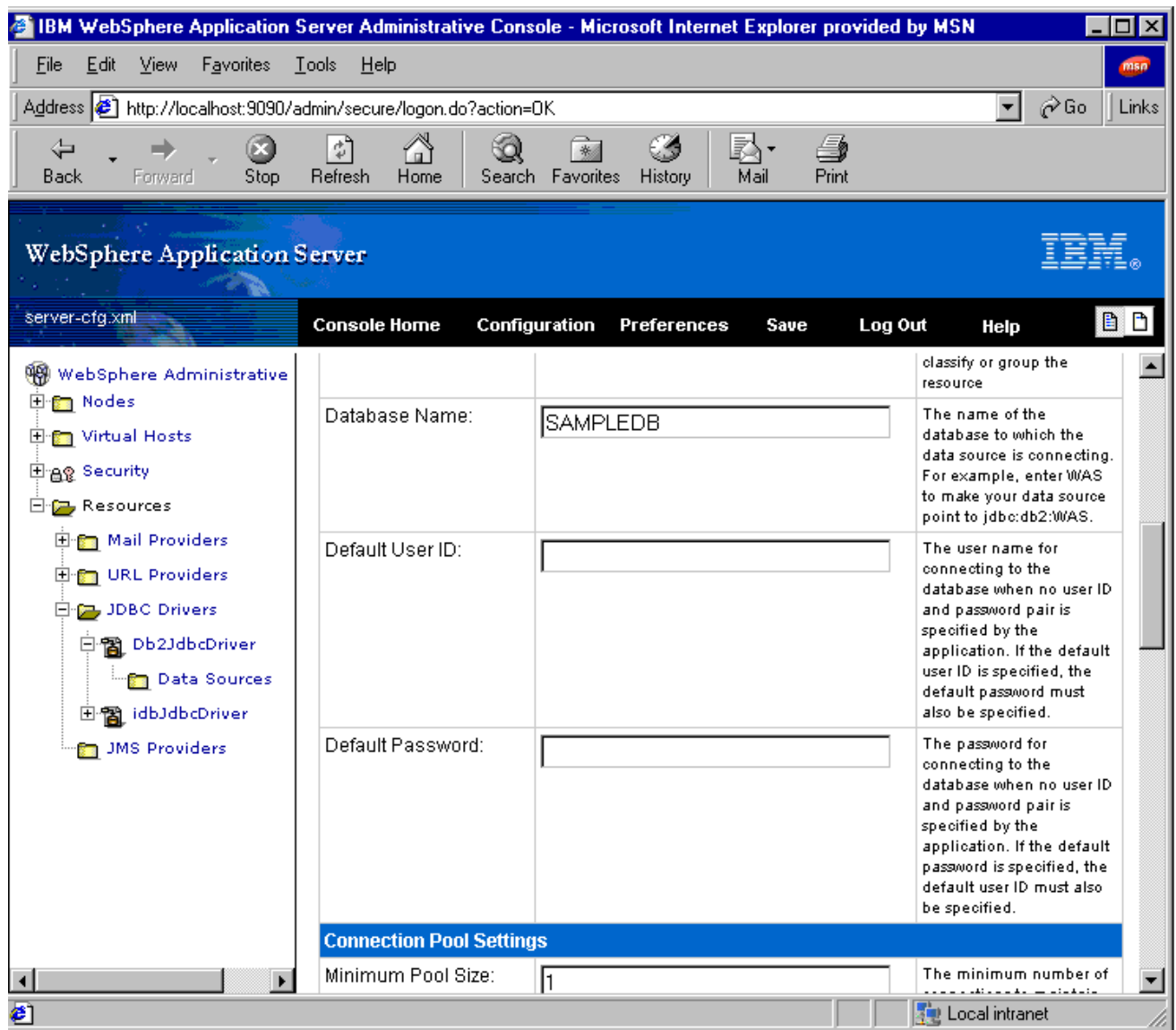
[For more information...](#)

Properties		
Name:	* GettingStartedDataSource	The required display name for the resource
JNDI Name:	* jdbc/GsDataSource	The required JNDI name for the resource, including any naming subcontexts. The name is used to link the platform binding information. The binding associates the resources defined the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.
Description:	New DB2 Data source	An optional description of the resource

Done Local intranet

- Database Name: SAMPLEDB





- e. Click **OK**.
- f. Save the configuration by clicking **Save**. Save it under **server-cfg.xml** and click **OK**.

#### Oracle Database:

1. In the administrative console, create the OracleJdbcDriver:
  - a. In the topology tree, expand **Resources**
  - b. Select **JDBC Drivers**.
  - c. Click **New**.
  - d. Enter the following:
    - Server Class Path: **<oracle JDBC classes12.zip location>**
    - Name: **OracleJdbcDriver**
    - Description: **Oracle JDBC Driver**
    - Implementation Class: **oracle.jdbc.pool.OracleConnectionPoolDataSource**
  - e. Click **OK**.
  - f. Save the configuration by clicking **Save**.
2. In the administrative console, **configure a new data source**:
  - a. In the topology tree, expand **Resources -> JDBC Drivers -> OracleJdbcDriver**.
  - a. Select **Data Sources**.
  - b. Click **New**.
  - c. Enter the following values. (If a field value is not specified, leave the field blank).
    - Name: **GettingStartedDataSource**
    - JNDI Name: **jdbc/GsDataSource**
    - Default User ID: **<Oracle DB userid>**
    - Default Password: **<Oracle DB password>**
  - d. Click **OK**.
  - e. In the topology tree, expand **Resources -> JDBC Drivers -> OracleJdbcDriver -> DataSources**.

- f. Select **GettingStartedDataSource**.
- g. Select **Property Set** at bottom of screen.
- h. Select **Resource Properties**.
- i. Click **New**.
- j. Enter the following:
  - Name: **URL**
  - Type: **java.lang.String**
  - Value: **jdbc:oracle:thin:@<hostname>:1521:<your\_Oracle\_instance\_name>**
- k. Click **OK**.
- l. Click **OK**.
- m. Click **OK**.
- n. Save the configuration by clicking **Save**. Save it under **server-cfg.xml** and click **OK**.

## Install the application

Ensure you have finished generating code for deployment, before beginning this step.

1. Install the application. Use the [application installer tool](#).

On Windows NT, use the command:

```
SEAppInstall -install path/Deployed_cmp.ear -ejbdeploy false
```

On UNIX systems:

1. cd [product\\_installation\\_root](#)/bin
2. Invoke SEAppInstall.sh using the syntax for your operating system. Use the arguments:

```
-install path/Deployed_cmp.ear -ejbdeploy false
```

In either case, *path* is the fully qualified path to the .ear file.

2. When prompted for the Default Datasource JNDI Name for the EJB jar, enter **jdbc/GsDataSource**
3. When prompted for a database User ID and password:
  - DB2 database: Press enter
  - Oracle database: Enter oracle db userid and password
4. Verify and press **Enter** if the JNDI Name for the Enterprise Bean is set to:  
**com/ibm/websphere/gettingstarted/ejbs/SimpleContainerManagedHome**
5. Verify and press **Enter** if the Datasource JNDI name is set to:  
**jdbc/GsDataSource**
6. When prompted for a database User ID and password:
  - DB2 database: Press enter
  - Oracle database: Enter oracle db userid and password

See the [Application deployment tutorial](#) for an example of installation using the Web-based administrative console instead. You can also view the [administrative console instructions](#).

## Regenerate the plug-in configuration

Recall this step from the [basic Application deployment tutorial](#).

## Stop the server and start it again

Recall this step from the [basic Application deployment tutorial](#).

## Create database table or tablespace

The instructions differ according to whether you are using a DB2 database or an [Oracle database](#).

DB2 Database:

1. Open a command window.
2. Type **db2cmd** (NT/Windows 2000) or **db2** (Unix) to start a DB2 command window.
3. **If** you do not already have a database named SAMPLEDB, you must create one:
  - a. **db2 create database SAMPLEDB**
4. Extract the DDL from the deployed jar file:
  1. In the WebSphere administrative console tree view, select **Nodes -> hostname -> Application Servers -> Default Server -> EJB Containers -> Installed EJB modules**
  2. Select the EJB module from list of EJB modules by selecting its check box.
  3. Click the **Export DDL** button.
  4. Click the file from the list of Table.DDL files to download it.
  5. Click **OK**.
5. Create the table:
  - a. **db2 connect to SAMPLEDB**

b. **db2 -t -f META-INF\Table.ddl**

A table named COM\_IBM\_WEBSPHERE\_GETTINGSTAR1 will be added to the SAMPLEDB database.

6. Exit the db2 command window.

Oracle Database:

1. Open a command window.

2. Extract the DDL from the deployed jar file:

a. **jar -xvf %was\_home%\installedApps\Deployed\_cmp.ear\cmp11.jar META-INF\Table.ddl**

3. Create the tablespaces:

a. **svrmgrl**

b. **connect <oracle db userid> / <oracle db password>**

c. **@META-INF\Table.ddl**

A tablespace named COM\_IBM\_WEBSPHERE\_GETTINGSTAR1 will be added database.

4. Exit the oracle command window.

## Ensure that the application, application server, and Web server are running

Recall learning this step in the [Application testing tutorial](#).

## Test the Java application client

1. Open a system command prompt.

2. Change directory to:

[product\\_installation\\_root](#)/bin

3. Enter the command:

```
launchClient ../temp/tutorial/cmp/Deployed_cmp.ear
```

(Remember, on Windows systems, the forward slashes should be backslashes).

## What's next?

Proceed to the [Application removal and cleanup tutorial](#) if you would like to uninstall the applications that you introduced into the server runtime during the tutorial.

## 6.7.6: Application removal and cleanup tutorial

During this tutorial, you will use the administrative console to uninstall (remove) the simpleSession application (or perhaps simpleSessionSecure application if you performed the Application security tutorial).

### Prerequisites

This tutorial assumes that you have installed the simpleSession application (unsecured or secured) from a previous tutorial, and are okay with uninstalling it now. You can extrapolate the instructions to remove the applications pertaining to the SOAP tutorial and advanced CMP tutorial also.

### Overview of steps (5 minutes)

1. Start the application server.
2. Open the administrative console.
3. Uninstall the application.

You learned most of the above steps already, in the [Application deployment tutorial](#).

### Uninstall the application

1. In the tree view of the console, expand **Nodes** -> *localhost*.
2. Select **Enterprise Applications** to display a list of applications.
3. Click the check box next to SimpleApp so that it is selected.
4. Click **Uninstall**.
5. Click **Uninstall** again to start the installation wizard.
6. Save the server configuration using the dialog obtained as a result of clicking the "Configuration needs to be saved" prompt at the top of the **Enterprise Applications** page.



# Tutorial -- Deploying a Java class as a Web service using SOAP

During this tutorial, you will deploy a Java class as a Web service in WebSphere Application Server. You will use the Stock Quote sample, which is one of the SOAP samples included with the product. Specifically, you will assemble a J2EE application .ear file containing the Stock Quote JavaBean component. Then you will SOAP-enable the .ear file and deploy it in the server runtime. Finally, you will test the enterprise application.

The value of Web services and SOAP technology is that you can take code components that were not previously Web applications -- such as enterprise beans or your Stock Quote sample -- and expose them as services available to HTTP requests (Apache SOAP Remote Procedure Call uses HTTP as its transport). You can do so without rewriting the code, in most cases. For exceptions, see "Creating Type Mappings" in the Apache SOAP 2.2 documentation on the [Apache 'Organization Web site](#). In other words, SOAP can be applied as a transparent wrapper to Web-enable the components.

Note, SOAP message services *do* have to be aware of SOAP. An option is to write your own SOAP provider to get the message and translate it to match the method expected by the code component (object) that you are calling. For example, this practice could be used for communication between SOAP Messaging and the JMS layer. For more information, see the Apache site.


## Paths through the tutorial

- **Quick path:** If your objective is to practice these steps as quickly as possible, skip the explanations marked with the  graphic.
- **Maximum learning path:** If your objective is to understand and explore the product capabilities, read the  notes and browse the links they provide to additional documentation.

## Overview of steps (requires 30 - 60 minutes)


- [Background](#)
- 1. [Expanding the enterprise application EAR file \(soapsamples.ear\)](#)
- 2. [Assembling the JavaBean into a JAR file](#)
- 3. [Starting the Application Assembly Tool](#)
- 4. [Assembling the EAR file containing the JavaBean JAR](#)
- 5. [Creating an Apache SOAP deployment descriptor](#)
- 6. [Exiting the Application Assembly Tool, saving your EAR file](#)
- 7. [SOAP-enabling the services in the EAR file](#)
- 8. [Starting the server](#)
- 9. [Starting the administrative console](#)
- 10. [Installing the EAR file into the server runtime](#)
- 11. [Saving the server configuration and regenerating the plug-in configuration](#)
- 12. [Stopping the application server and starting it again](#)
- 13. [Testing the Stock Quote service](#)

## Background

 Use these links to expand your J2EE vocabulary. These terms are used throughout the tutorial.

- [What are Web services?](#)
- [What are enterprise applications?](#)
- [What are Enterprise Archive \(EAR\) files?](#)
- [What are EJB modules?](#)
- [What are Web modules?](#)
- [What are Web Archive \(WAR\) files?](#)
- [What are application client modules?](#)
- [More concepts and terminology](#)

See also the [SOAP coverage](#) in the InfoCenter.

 You might notice small discrepancies between the graphics in this tutorial and the actual product screens as this documentation ages and product fix packs are issued.

## Expanding the enterprise application EAR file (soapsamples.ear)

Use the following steps to obtain the SOAP sample server and client code from the soapsamples.ear file provided with the product. Expand the contents of the .ear file into a temporary directory as described here.

1. Create a soapsamples subdirectory in the path:

```
product_installation_root\temp
```

2. Change directory to:

```
product_installation_root\bin
```

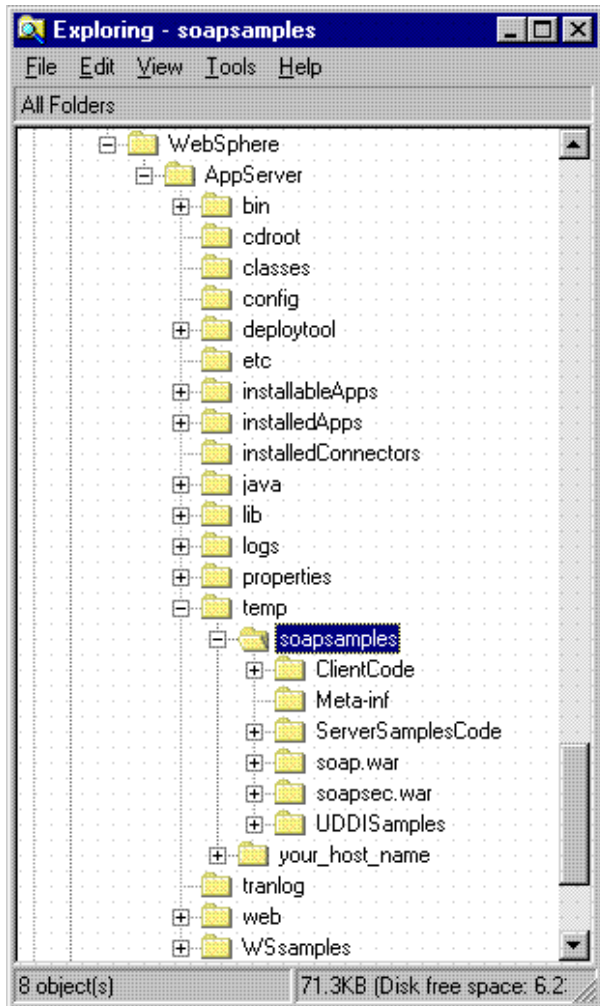
**i** In this case, it is important to issue the command from bin because the arguments for the tool will be specified relative to the bin directory. On other occasions, you can use the tool from any directory, provided you specify the file paths in the arguments relative to that directory.

3. Expand the EAR file, using the [EAR expander tool](#).

1. Open a system command prompt.
2. Issue the command (on a single line, using the appropriate forward or backward slashes for your system):

```
earexpander -ear  
..\installableApps\soapsamples.ear -expandDir  
..\temp\soapsamples -operation expand -expansionFlags war
```

When you finish, your temp directory will contain the contents of the expanded .ear file.



**i** To learn more about the tool syntax, type `earexpander` at a command prompt, with no arguments. The `-expansionFlags war` argument is notable. This tells the tool to expand the .war files, but not the .jar files. If you see a directory named `samples.jar` in your `temp/soapsamples` directory, delete the `soapsamples` directory and issue the command again. Pay special attention to issuing the `-expansionFlags war` correctly.

## Assembling the JavaBean into a JAR file

For the example used in the tutorial, you can skip this step.

**i** At this point, the code artifact (in this case, the Stock Quote JavaBean component) is typically compiled and then packaged into a JAR file. Because the Stock Quote sample is already packaged into `samples.jar` in your `temp/soapsamples` directory, you can skip this step. The `samples.jar` file is part of `soapsamples.ear`, as you will notice by viewing the contents of the expanded .ear file from the previous step.

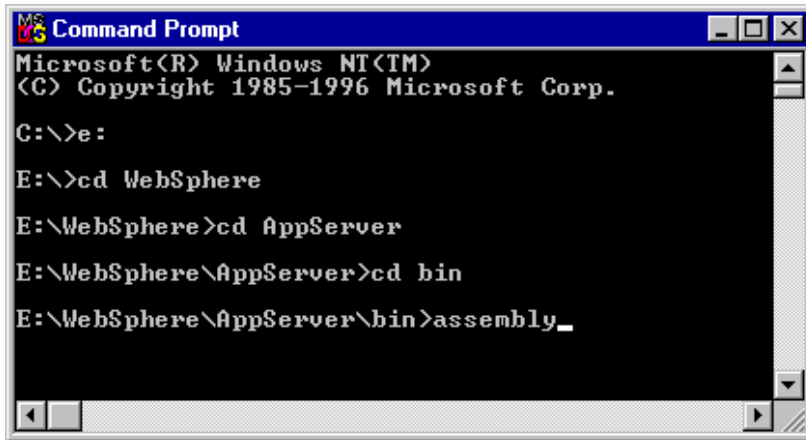
## Starting the Application Assembly Tool

The next step is assemble an EAR file (enterprise application) containing the JAR file for the Stock Quote JavaBean component. To do so, you must start the [Application Assembly Tool](#).

1. Open a system command prompt.
2. Change directory to:

`product_installation_root/bin`

3. Enter: assembly\_

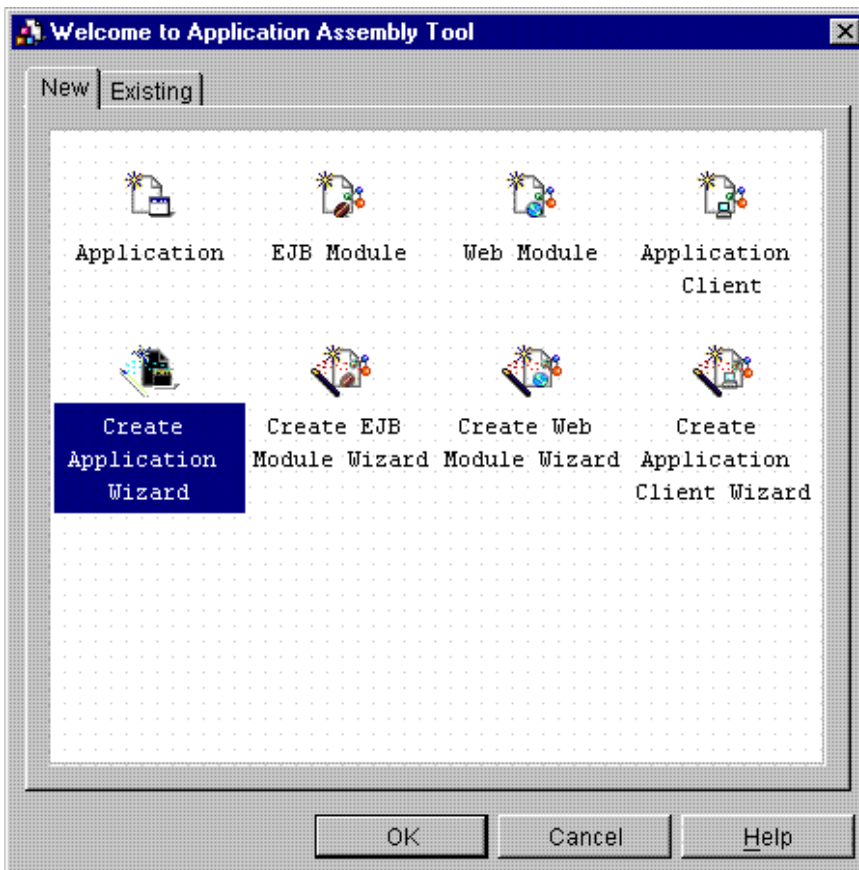


**NT** If using Windows NT or 2000, and you do not see the graphical interface of the tool right away, check for the minimized tool on the Task Bar.

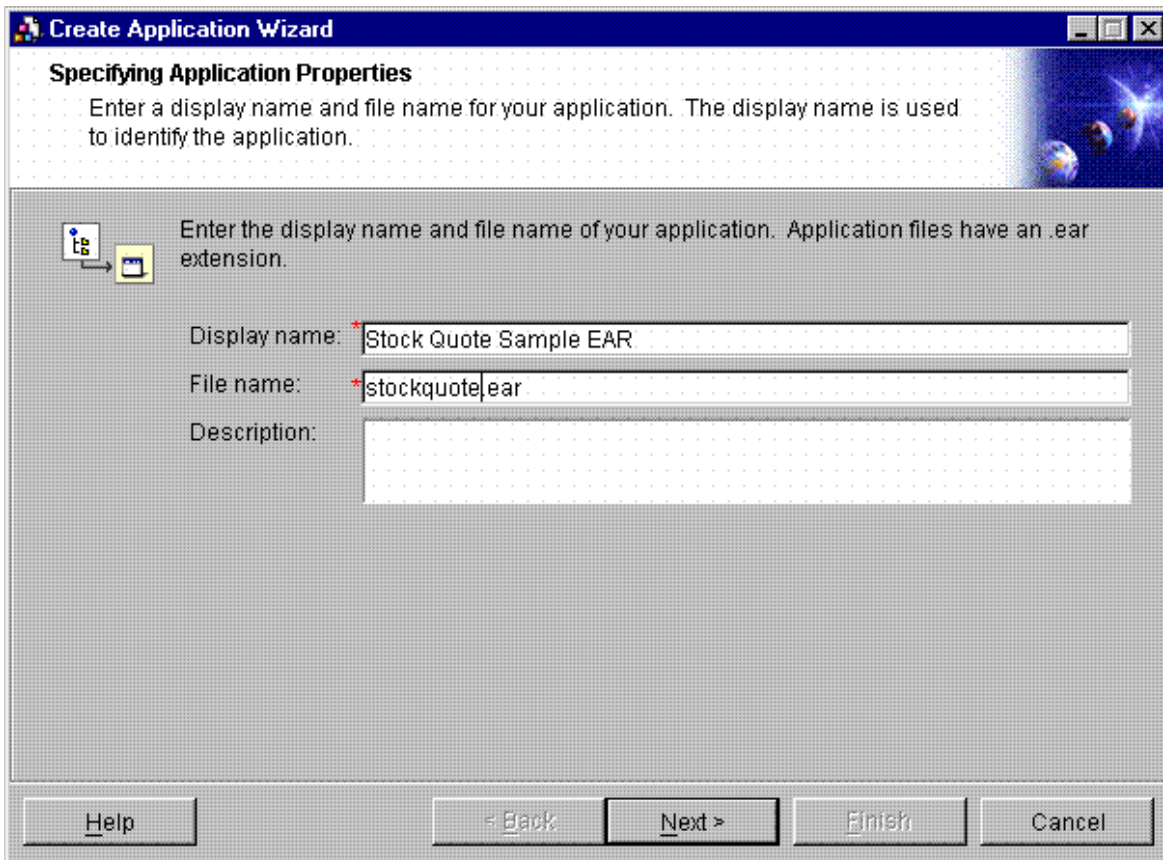
## Assembling the EAR file containing the JavaBean JAR

Now, create the enterprise application archive (EAR) with the Application Assembly Tool.

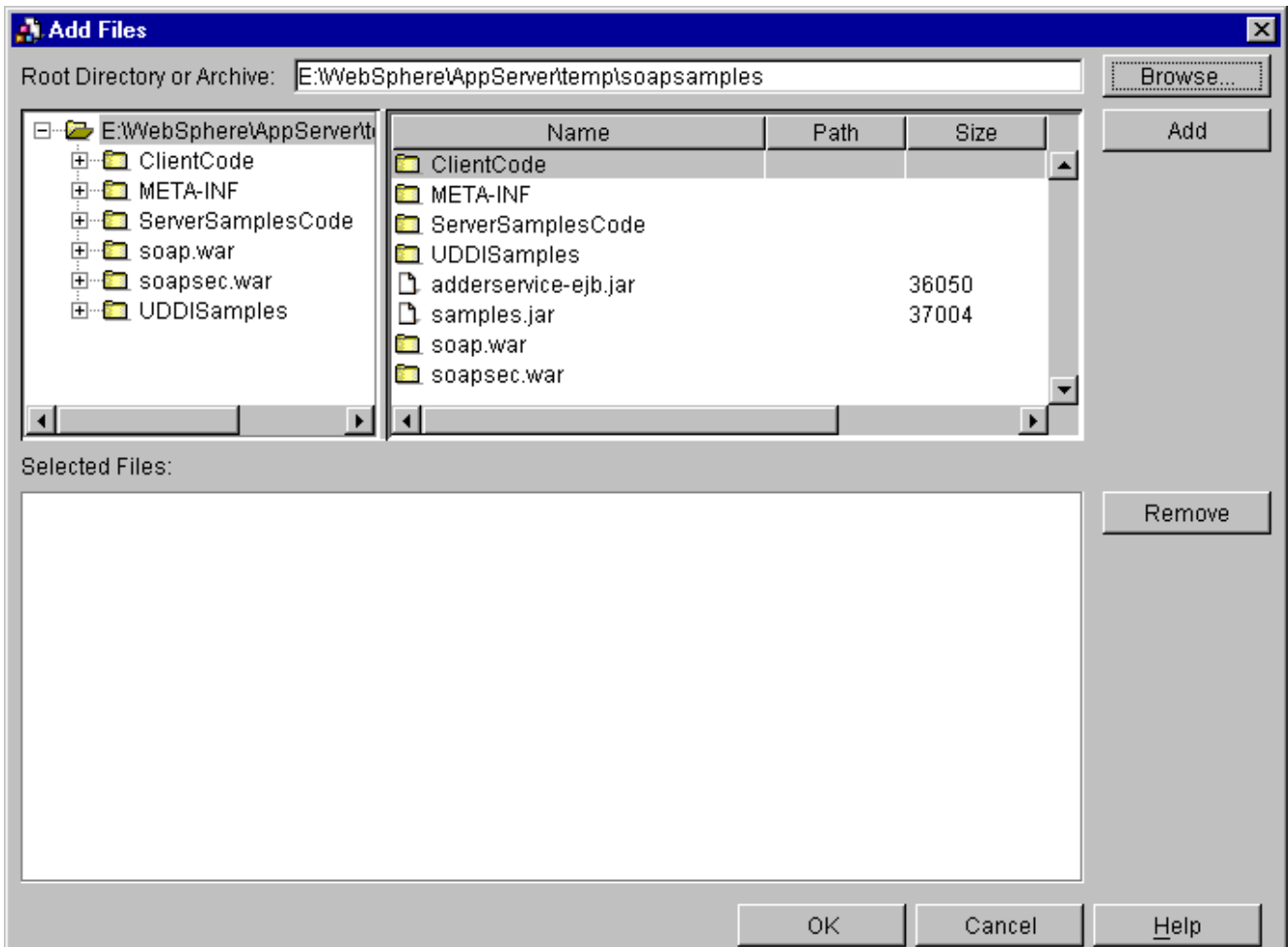
1. Create a new enterprise application archive (.ear file) containing the sample.jar.
  1. Select the **Create Application Wizard**.



2. In the **Specifying Application Properties** panel of the wizard, specify information:
  - Display name
    - Stock Quote Sample EAR
  - File name
    - stockquote.ear

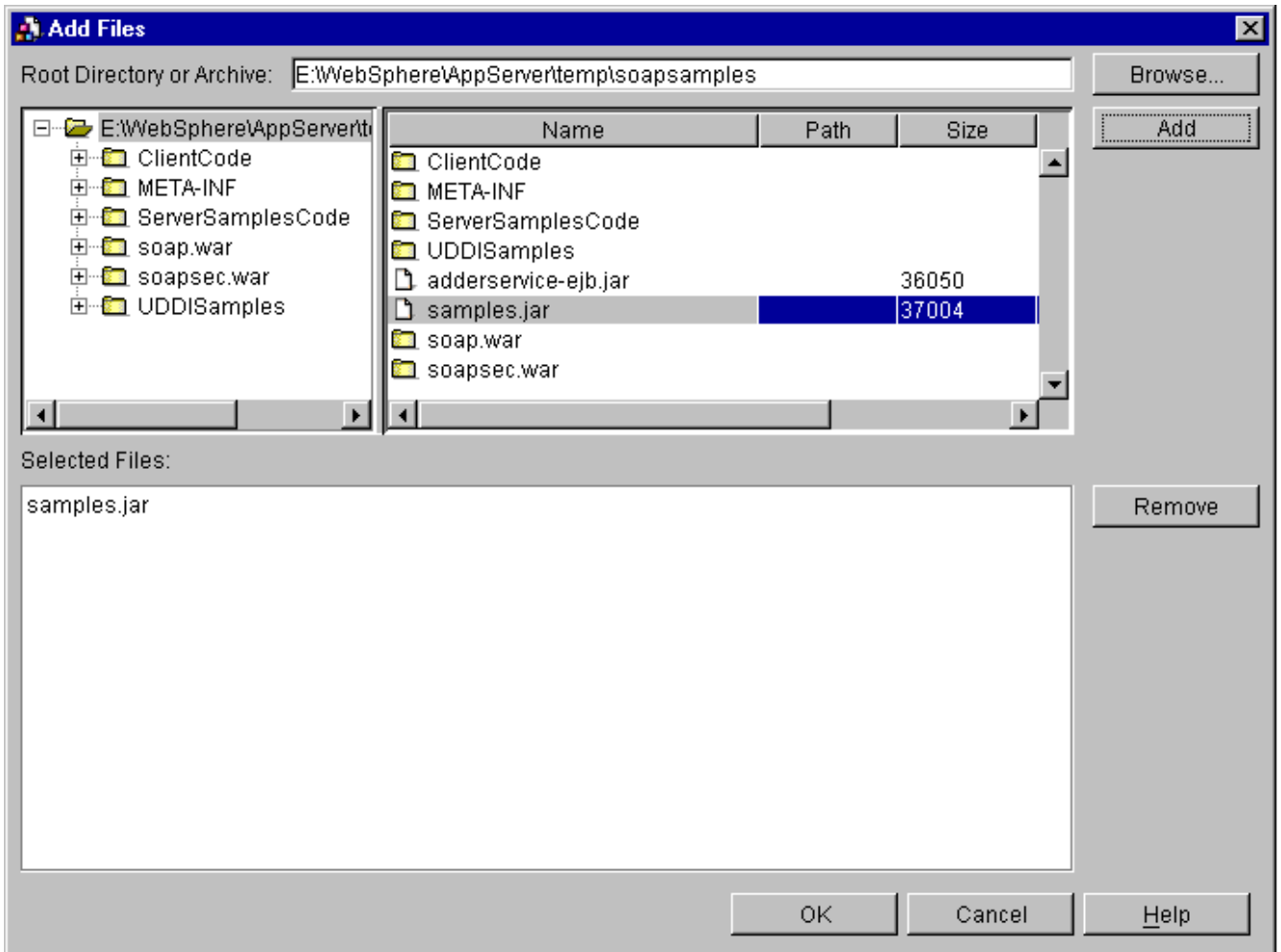


3. Click **Next**.
4. On the **Adding Supplementary Files** panel of the wizard, click **Add**.
5. Browse for and select **the directory** containing the samples.jar file -- that is, select the temp/soapsamples directory.

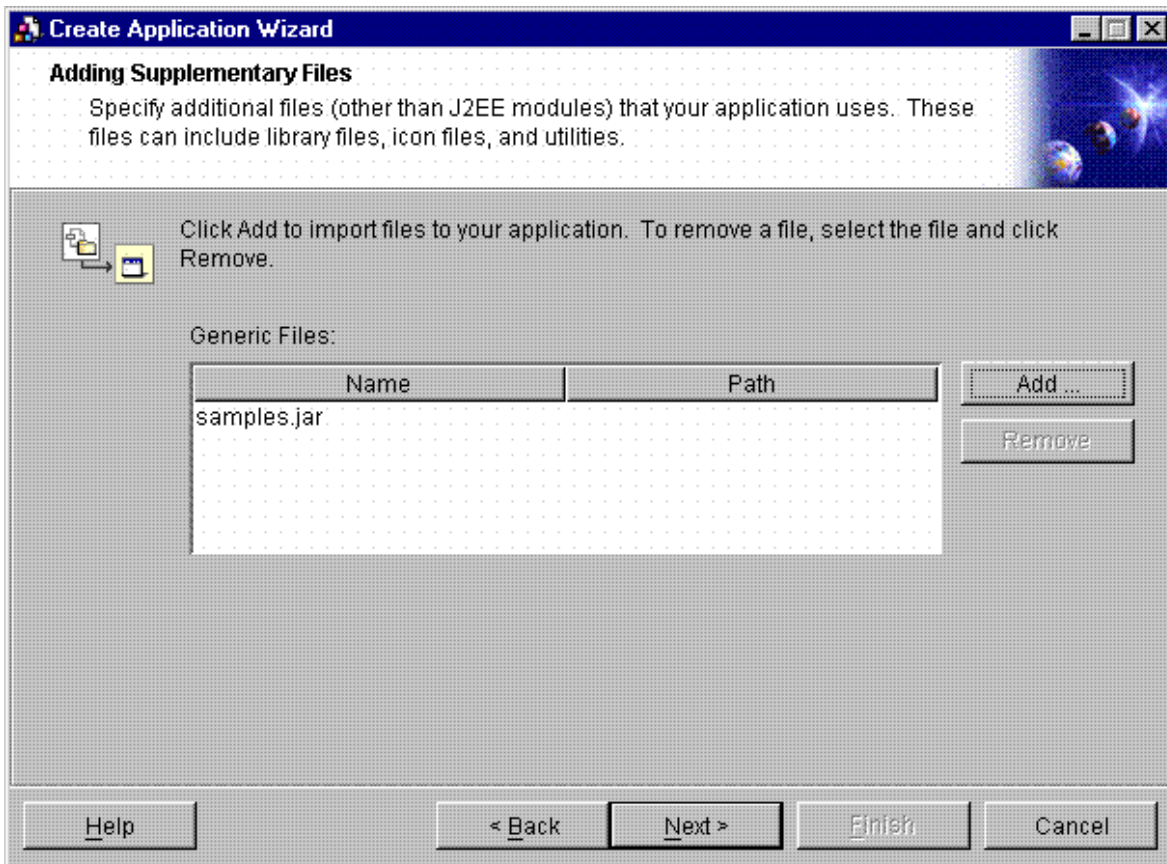




6. When you return to the **Add Files** dialog window:
- Select the `samples.jar` file in the list in the upper right part of the screen.
  - Click **Add** to display the `samples.jar` in the lower half of the screen.



7. Click **OK** to exit the **Add Files** dialog window and return to the main wizard.



8. Click **Next** until you reach a panel on which you can click **Finish**. Do so, to exit the wizard.

**i** Advancing through each panel will give you a good idea of the types of application characteristics you can configure with this wizard. To learn more about the wizard, read the [help file for the wizard](#).

2. As a formality, create a .war file for the enterprise application.

The main assembly tool window will display a tree view containing the Stock Quote Sample EAR application.

1. Expand the tree view until you can see the **Web Modules** folder.
2. Right-click the folder and select **New** from the resulting menu.
3. Specify the general Web module properties.

File name

stockquote.war

Context root

/stockquote

Display name

Stock Quote Sample

Note, the context root value is arbitrary.

4. Click **OK**.

You will now be able to see the Stock Quote Sample if you expand the assembly tool tree view to show the contents of the **Web Modules** folder.

**i** Why create a .war file (Web module archive) if you are not enabling a servlet or JSP file? The Application Assembly Tool requires the EAR file to contain at least one J2EE archive, meaning a .war file or an EJB or application client .jar file. In this case, you will create an empty .war file (Web module) to satisfy the requirement.


## Creating an Apache SOAP deployment descriptor

For the example used in the tutorial, you can skip this step.

**i** Before running the SOAP EAR Enabler tool, typically you must create an Apache SOAP Deployment Descriptor for each service to be enabled. For the Stock Quote sample, the deployment descriptor was created for you. To view it:

1. Right-click the **Stock Quote Sample EAR** application in the assembly tool tree view.
2. Select **View Descriptor** from the resulting menu.
3. When finished viewing the descriptor, click **Close**.

See also the [documentation about Apache SOAP deployment descriptors](#).

 What are descriptors? Where do they originate? Currently, you can use a text editor to create the descriptor, based on IBM documentation of Apache SOAP descriptors. Basically, the descriptor is a service definition file containing the information the SOAP runtime needs to understand how to invoke the service. In a nutshell, it contains binding information such as:

- The name by which the service can be accessed, which is generally related to the JNDI name of an enterprise bean
- The methods available on the service for calling
- What 'class' implements the service
- What SOAP provider handles this service -- for example, the generic Java provider, an EJB provider, a Bean Scripting Framework (BSF) provider, and so on.
- Any special data type mappings so it can properly serialize/deserialize the objects a method receives and returns (In the case of a BSF script, the entire script can be embedded in the descriptor, hence the reason the tool asks you for classpath requirements. For a BSF script, it is possible to have no classpath requirements
- Anything else optional that the service may need. In the case of enterprise beans, the JNDI name of the bean and what the full home interface name

## Exiting the Application Assembly Tool, saving your EAR file

Save your file and exit the assembly tool.

1. Click **File** -> **Save As** on the menu bar of the tool.
2. Save the stockquote.ear file to:

`product_installation_root/temp`


3. Click **File** -> **Close** on the menu bar of the tool.
4. If prompted, specify to save your modifications. Then click **OK** when you receive the message that the "Archive was saved successfully."
5. Click **File** -> **Exit** to exit the Application Assembly Tool.

## SOAP-enabling the services in the EAR file

WebSphere Application Server provides a Java tool for enabling an interface in an EAR file. This [Soap EAR Enabler tool](#) takes as input the Apache SOAP Deployment Descriptor and the EAR file from the previous step.

1. Open a system command prompt.
2. Change directory to

`product_installation_root\bin`

 In this case, it is important to issue the command from bin because the arguments for the tool will be specified relative to the bin directory. On other occasions, you can use the tool from any directory, provided you specify the file paths in the arguments relative to that directory.

3. Start the SOAP EAR Enabler tool by typing soapearenabler.
4. When the tool prompts you for the name of your .ear file, specify (using the correct forward or backward slashes for your system):

`..\temp\stockquote.ear`

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>e:

E:\>cd websphere

E:\WebSphere>cd appserver

E:\WebSphere\AppServer>cd bin

E:\WebSphere\AppServer\bin>soapearenabler

IBM WebSphere Application Server Release 4
SOAP Enterprise Archive enabler tool.
Copyright IBM Corp., 1997-2001

Please enter the name of your ear file: ..\temp\stockquote.ear

*** Backing up EAR file to: ..\temp\stockquote.ear~

How many services would you like your application to contain (1...n)?1

Now prompting for info for service 1:
    Please enter the file name of the SOAP deployment descriptor xml file:
```

**H** What if the tool replies that you should specify a file that exists? Then your stockquote.ear is not in the temp directory. Exit the tool, move the file, and try again.

5. Answer the rest of the questions as you are prompted to do so. For readability, the following responses are shown on a separate line from the questions. When using the tool, you must enter your answer directly at the end of the question (on same line, with no spaces preceding your answer).

How many services would you like your application to contain (1...n)?

1

Now prompting for info for service #1: Please enter the file name of the SOAP deployment descriptor xml file:

..\temp\soapsamples\ServerSamplesCode\src\stockquote\DeploymentDescriptor\stockquote.xml

Is this service an EJB (y= yes/n= no)?

n

How many jar files are required for this service (0...n)?

1

Classpath requirement #1: Please choose a file ([1] samples.jar, [2] stockquote.war):

1

Should this service be secured (y/n)?

n

Please enter a context root for your non-secured services (e.g. /soap):

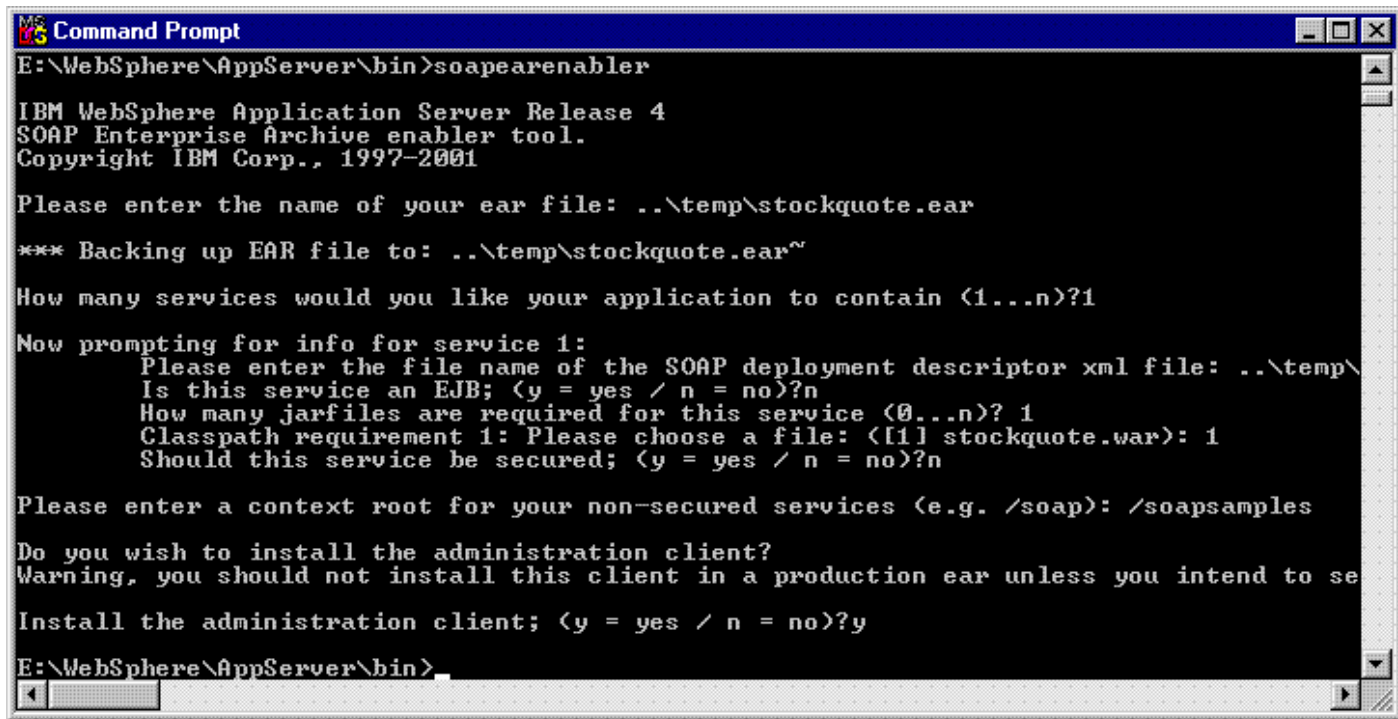
/soapsamples

**H** The context root is important when invoking the router servlet from the client.

Do you wish to install the administration client? Warning! You should not install this client in a production ear unless you intend to secure the URI to it. Install the administration client (y= yes/n= no)?

y

When the tool has received your answers for all of its questions, it will exit, and the system command prompt will return.



```
Command Prompt
E:\WebSphere\AppServer\bin>soapearenabler

IBM WebSphere Application Server Release 4
SOAP Enterprise Archive enabler tool.
Copyright IBM Corp., 1997-2001

Please enter the name of your ear file: ..\temp\stockquote.ear

*** Backing up EAR file to: ..\temp\stockquote.ear~

How many services would you like your application to contain (1...n)?1

Now prompting for info for service 1:
  Please enter the file name of the SOAP deployment descriptor xml file: ..\temp\
  Is this service an EJB; (y = yes / n = no)?n
  How many jarfiles are required for this service (0...n)? 1
  Classpath requirement 1: Please choose a file: ([1] stockquote.war): 1
  Should this service be secured; (y = yes / n = no)?n

Please enter a context root for your non-secured services (e.g. /soap): /soapsamples

Do you wish to install the administration client?
Warning, you should not install this client in a production ear unless you intend to se

Install the administration client; (y = yes / n = no)?y

E:\WebSphere\AppServer\bin>
```

**E** A little bit about the classpath requirement prompt -- the reason for selecting samples.jar is to ensure that the file will be put on the classpath that the SOAP runtime will see. The samples.jar has to be at the root of the .ear file in order to show up in the classpath requirements list presented by the EAR Enabler. During the assembly phase, you might have wondered why, if you had to create an empty .war file to satisfy the assembly tool requirements, that you did not just put the samples.jar inside the .war file. The samples.jar has to be included outside of a Web module, EJB module, or application client module in the .ear file, in order to be added to the classpath for the SOAP runtime.

## Starting the server

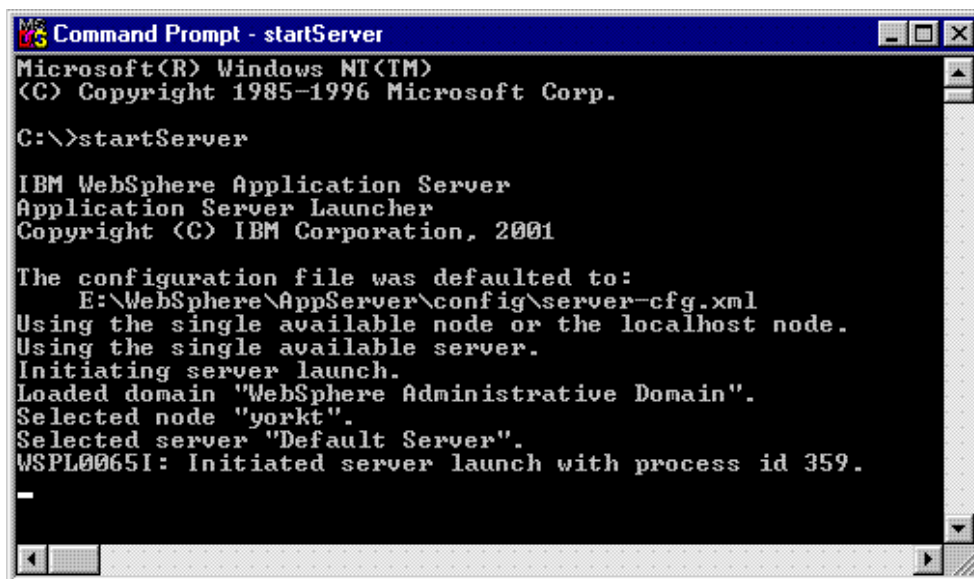
Start the application server.

1. Open a system command prompt.
2. Change directory to:

`product_installation_root/bin`

3. Enter: startServer

Leave the command window open, as you will use it again later to start the server.



```
Command Prompt - startServer
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>startServer

IBM WebSphere Application Server
Application Server Launcher
Copyright (C) IBM Corporation, 2001

The configuration file was defaulted to:
  E:\WebSphere\AppServer\config\server-cfg.xml
Using the single available node or the localhost node.
Using the single available server.
Initiating server launch.
Loaded domain "WebSphere Administrative Domain".
Selected node "yorkt".
Selected server "Default Server".
WSPL0065I: Initiated server launch with process id 359.
```

**E** Plan to use the default [server configuration file](#) for this tutorial. Other configurations are available, including those that you customize.

**E** Starting the server from a system command prompt is a method available on all supported operating systems. There are [other ways to start the server](#), as well as variations of the **startServer** command.

## Starting the administrative console

Now it is time to deploy (install) the application in the server runtime. To do so, you will use the [administrative console](#).

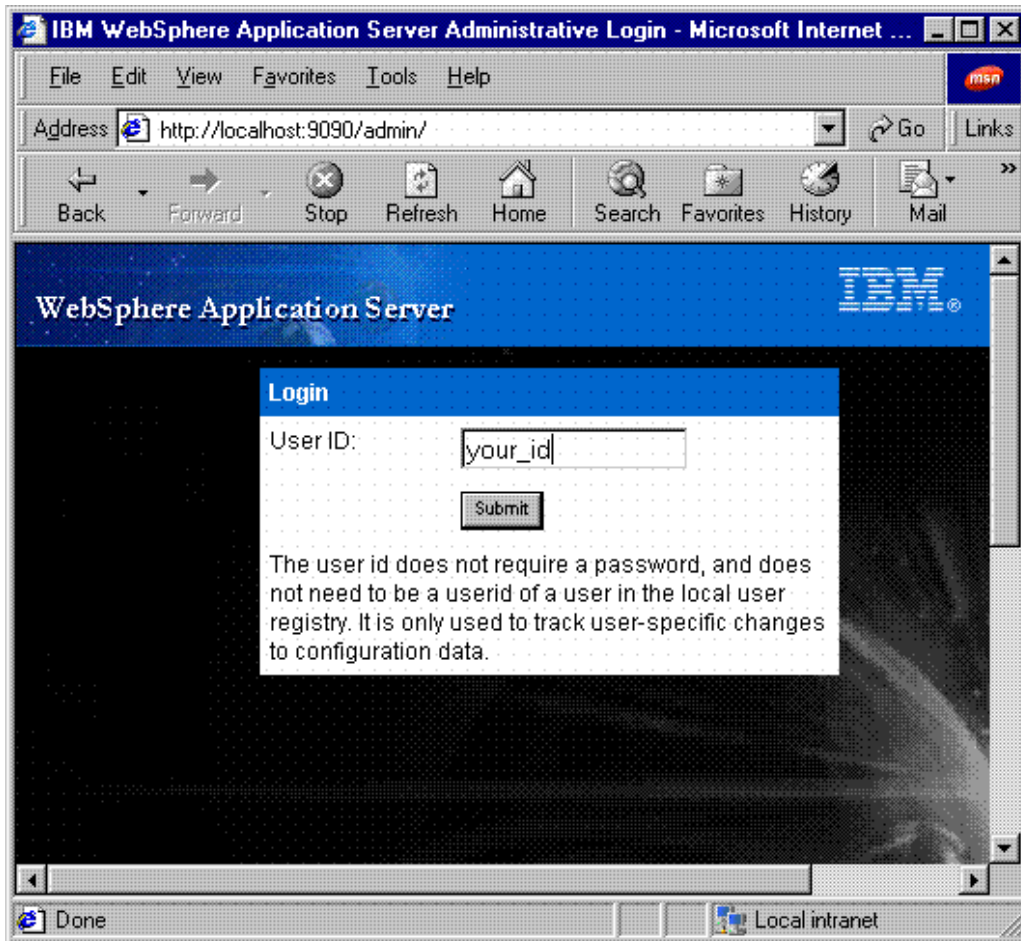
1. Ensure that the Web server is running on the machine containing the console.
2. Ensure that the product (application server) is also running. Recall, you started it a while ago.
3. In a Web browser, enter the URL:

`http://your_host_name:9090/admin`

where *your\_host\_name* is localhost if the Web console is on the local machine. On Windows 2000, it has been found that localhost is not always recognized. In such a case, use the actual host name.

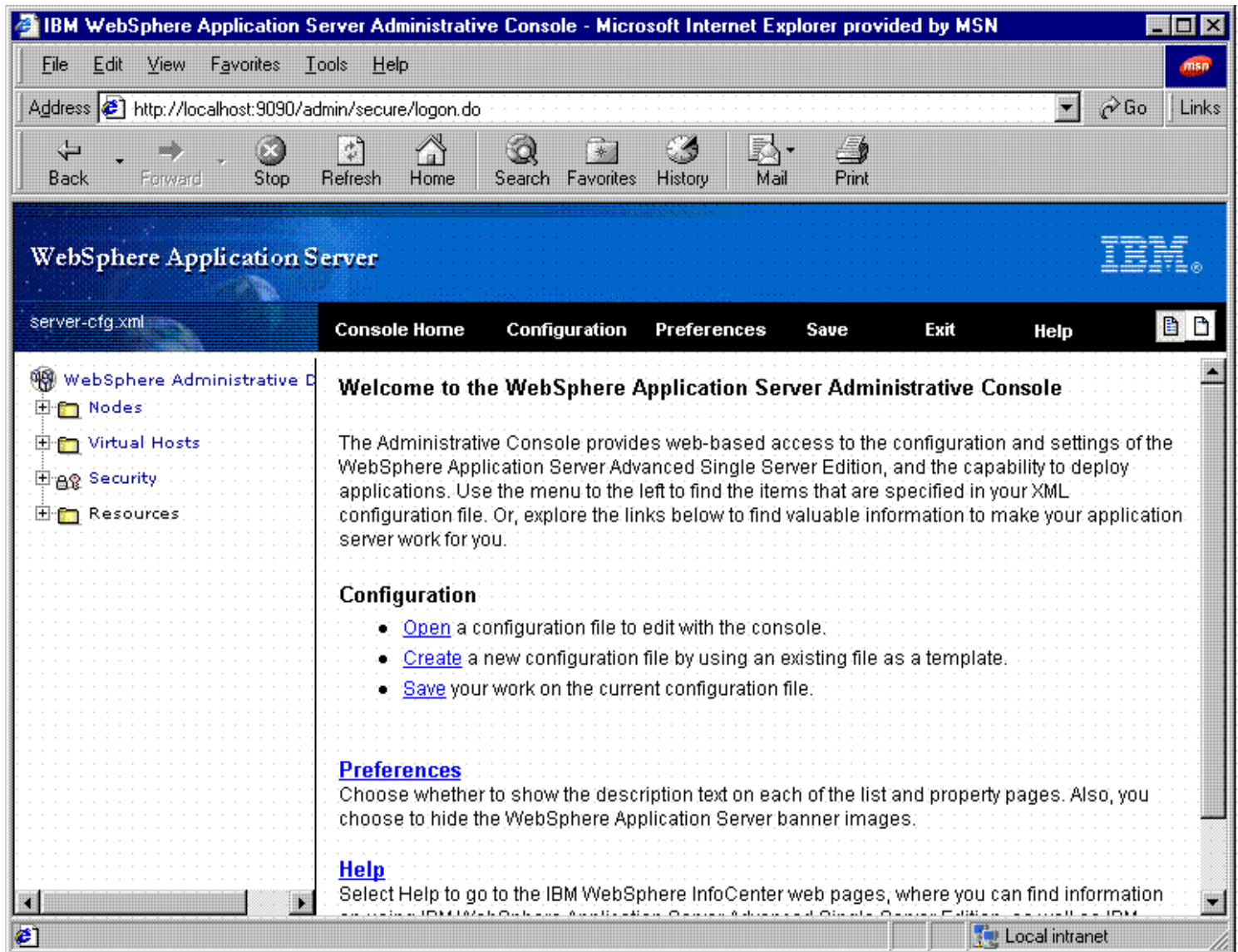
4. Enter an arbitrary login ID.

**i** Even if you have not secured the console yet, you will be prompted to log in. The login area explains this somewhat. For more information about the use of this ID, see [the help for starting and logging into the console](#). It will also contain any additions, corrections, and details pertaining to the above instructions.



5. Wait for the console to load into the browser.





## Installing the EAR file into the server runtime

1. In the tree view of the console, click:  
**Nodes -> your\_host\_name -> Enterprise Applications**
2. When the application collections view is displayed, click **Install**.

### Enterprise Applications

The J2EE applications (EAR files) installed on the application server

➔[For more information...](#)

<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Restart"/> <input type="button" value="Install"/> <input type="button" value="Uninstall"/> <input type="button" value="Export"/> <input type="button" value="Export DDL"/>		
<input type="checkbox"/>	Name	Archive URL
<input type="checkbox"/>	<a href="#">sampleApp</a>	\${APP_INSTALL_ROOT}/sampleApp.ear
<input type="checkbox"/>	<a href="#">Server Administration Application</a>	\${APP_INSTALL_ROOT}/admin.ear
<input type="checkbox"/>	<a href="#">WebSphere Application Server Samples</a>	\${APP_INSTALL_ROOT}/Samples.ear
<input type="checkbox"/>	<a href="#">petstore</a>	\${APP_INSTALL_ROOT}/petstore.ear

3. Browse for stockquote.ear on your local machine.

## Application Installation Wizard

Specify the application (EAR file) or standalone module (JAR or WAR file) to install. If installing a module, specify the name of a new application in which to install the module.

Specify the Application or Module located on this machine to upload and install	
Path:	E:\WebSphere\AppServer\tem <input type="button" value="Browse..."/>
Application Name :	Used only for standalone modules (*.jar, *.war) <input type="text"/>
Context Root :	Used only for Web modules (*.war) <input type="text"/>
<input type="button" value="Next"/> <input type="button" value="Cancel"/>	

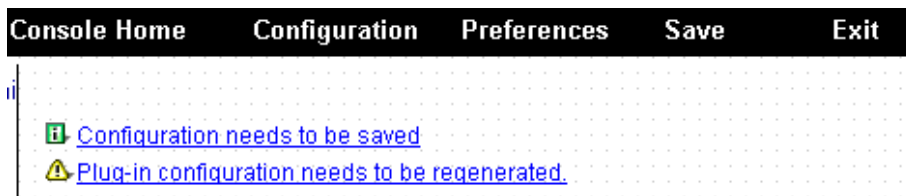
4. When the path to the .ear file is displayed in the **Path** field, click **Next**.
5. On the **Specifying Virtual Host names...** panel, keep the default values.
6. Click **Next**.
7. Confirm the settings and click **Finish** to deploy (install) the application.

**i** Another method of installing an application is to use the command line application installer tool. If you try it instead, press **Enter** when prompted whether to use the default host.

## Saving the server configuration and regenerating the plug-in configuration

**i** The administrative console is good at prompting you to take action when you need to do so. In this case, it will now remind you to save your [server configuration file](#) and [regenerate the configuration](#) of the WebSphere plug-in for the Web server you are using. The prompts are displayed directly below the console menu bar.

1. Click the first prompt, "Configuration needs to be saved."



2. On the **Save Configuration** panel, click **OK**.
3. Return to the **Enterprise Applications** collections view where the two prompts were displayed. You should be able to do so by clicking your browser's **Back** button twice.

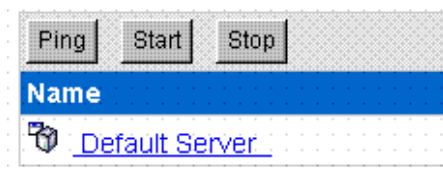
If that does not work, in the tree view of the console, click:

**Nodes -> your\_host\_name -> Enterprise Applications**

4. Now click the second prompt, "Plug-in configuration needs to be regenerated."
5. On the **Web Server Plug-In Configuration** panel, click **Generate**.
6. When returned to the **Enterprise Applications** page, confirm that the prompt now says: New plug-in configuration has been generated.

## Stopping the application server and starting it again

1. In the tree view of the console, click:  
**Nodes -> your\_host\_name -> Application Servers** to display the **Application Servers** collections view, including the Default Server.

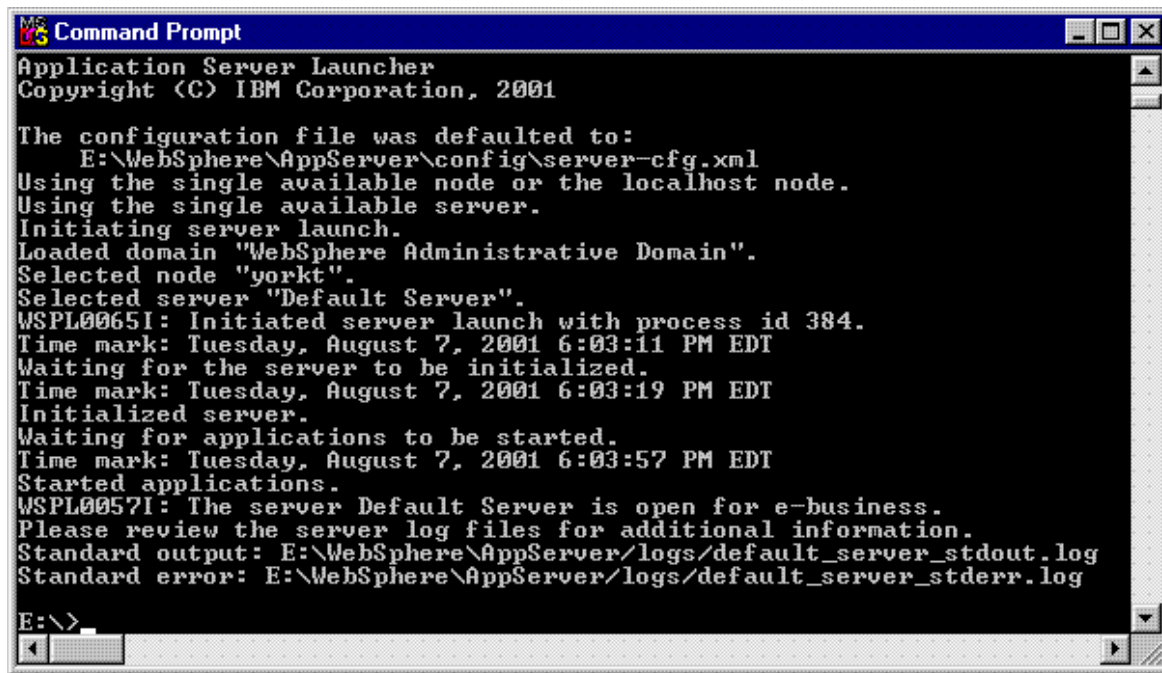


2. Click **Stop**.
3. Click **OK** when the screen discussing the implications of stopping the server is displayed.
4. Close the Web browser.



5. Start the server, using the `startServer` script as described earlier in this tutorial.

The server will issue a confirmation message in the launch window when it has started successfully.



```
Command Prompt
Application Server Launcher
Copyright (C) IBM Corporation, 2001

The configuration file was defaulted to:
E:\WebSphere\AppServer\config\server-cfg.xml
Using the single available node or the localhost node.
Using the single available server.
Initiating server launch.
Loaded domain "WebSphere Administrative Domain".
Selected node "yorkt".
Selected server "Default Server".
WSPL0065I: Initiated server launch with process id 384.
Time mark: Tuesday, August 7, 2001 6:03:11 PM EDT
Waiting for the server to be initialized.
Time mark: Tuesday, August 7, 2001 6:03:19 PM EDT
Initialized server.
Waiting for applications to be started.
Time mark: Tuesday, August 7, 2001 6:03:57 PM EDT
Started applications.
WSPL0057I: The server Default Server is open for e-business.
Please review the server log files for additional information.
Standard output: E:\WebSphere\AppServer\logs\default_server_stdout.log
Standard error: E:\WebSphere\AppServer\logs\default_server_stderr.log

E:\>
```

**E** Because adding a new application requires a change to the server configuration, the server must be stopped and started again to pick up the change. However, there are some types of changes you can make to an installed application that do not require stopping the server and starting it again. See [the dynamic reloading and hot deployment documentation](#) for more information.

## Testing the Stock Quote service

The SOAP samples come with a script to help you test the sample. The script is preconfigured with the URI of the SOAP router servlet. This URI is based on your specification of the context root when you ran the SOAP Ear Enabler tool (/soapsamples). If you specified a context root other than the default, then you must update the script.

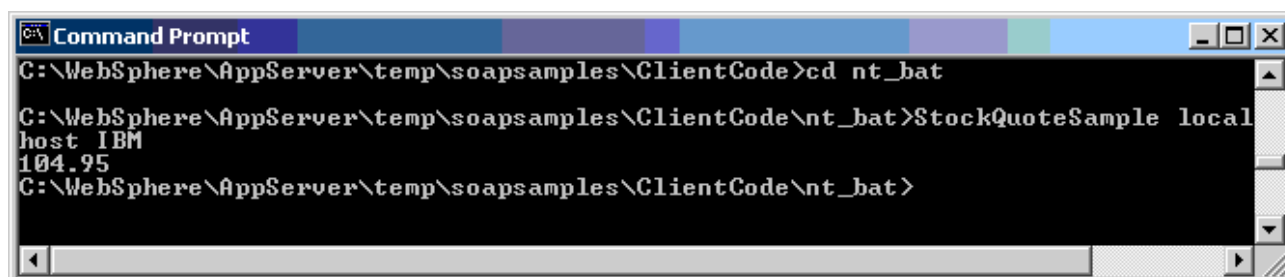
1. Open a system command prompt.
2. Change directory to:  
  
`product_installation_root\temp\soapsamples\ClientCode\`
3. Change directory to where the scripts are located, for your operating system.
  - o **UNIX** `unix_scripts`
  - o **NT** `nt_bat`
4. Invoke the script:

```
StockQuoteSample.bat | sh
with the arguments "localhost IBM."
```

For example, on Windows NT, this would be:

```
StockQuoteSample localhost IBM
```

You should receive a current stock quote for IBM stock.



```
Command Prompt
C:\WebSphere\AppServer\temp\soapsamples\ClientCode>cd nt_bat
C:\WebSphere\AppServer\temp\soapsamples\ClientCode\nt_bat>StockQuoteSample localhost IBM
104.95
C:\WebSphere\AppServer\temp\soapsamples\ClientCode\nt_bat>
```

If you receive a connection refused error or connection timed out error, then there could be a problem with your external access to the internet.

## Workaround

If you experience a connection problem, it might have to do with your network's firewall. Try the following:

1. Start the product.
2. Open the administrative console.
3. Access the JVM command line settings.

In the tree view, click **Nodes** -> *your\_host\_name* -> **Application Servers** -> *your\_server\_name* -> **JVM Settings**.

4. At the *front* of the field for entering command line arguments, set the arguments for the socks or HTTP proxy hosts and ports. Enter the value in the **Command line arguments** field.

If using a socks proxy server, add:

```
-DsocksProxyHost=The name of your socks proxy host  
and
```


```
-DsocksProxyPort=The number of your socks proxy port
```

If using an HTTP proxy server, add:

```
-DproxyHost=The name of your HTTP proxy host  
and
```

```
-DproxyPort=The number of your HTTP proxy port
```

5. Save the server configuration.

 After saving the configuration, you can check what the command line arguments will be, before you start the server. Run the startServer command with the -script argument. This will generate a startup script, from which you can check the accuracy of the command line arguments to ensure they are what you intended. See [instructions for starting the server](#) for more information about the -script option and others.

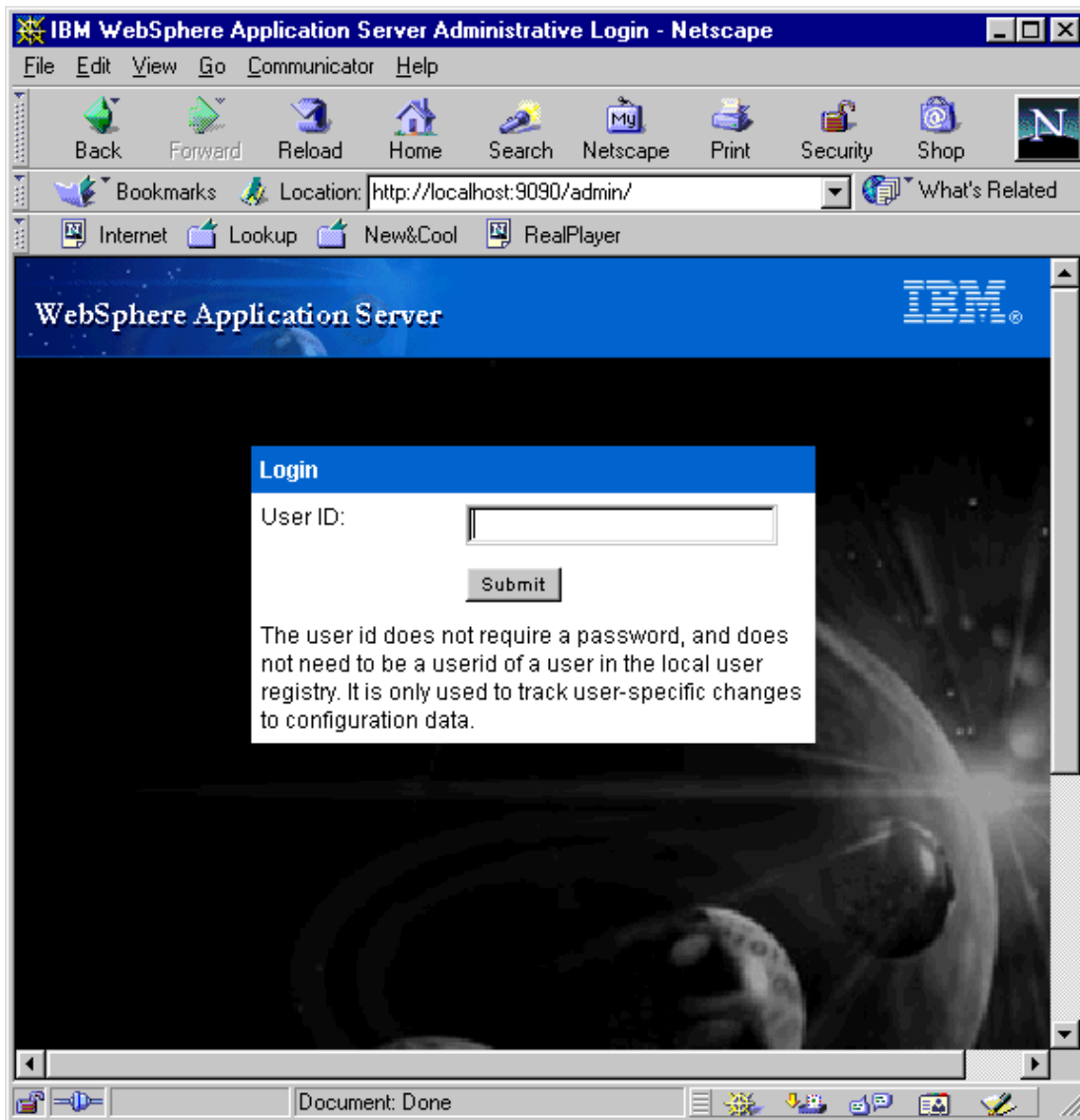
6. Stop the application server and start it again, as described previously in this tutorial.

# HitCount tutorial: Using Debugger and OLT

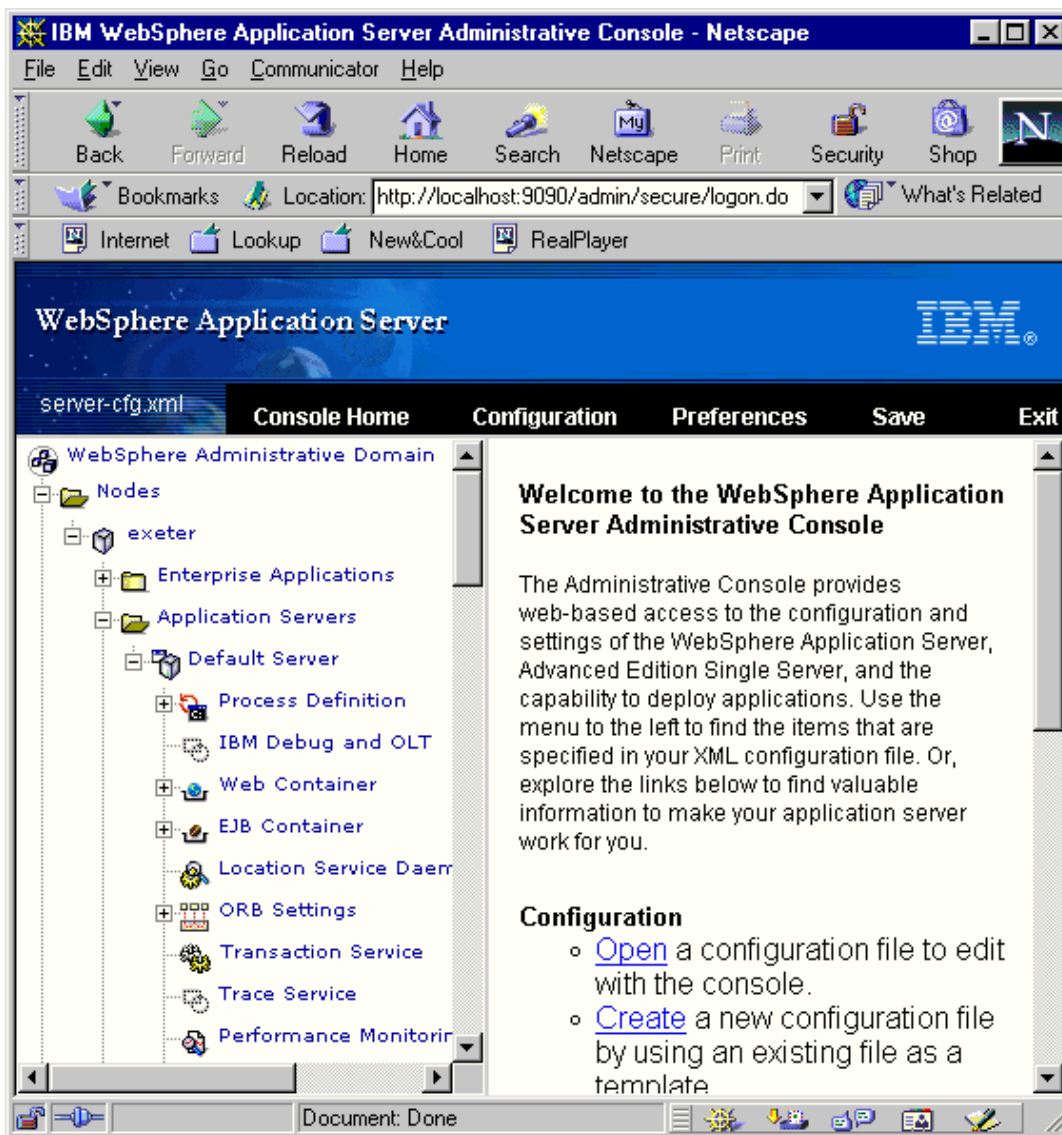
HitCount is an example servlet that installs by default with IBM WebSphere Application Server. This tutorial demonstrates how to use the IBM Distributed Debugger and Object Level Trace (OLT) on the HitCount servlet, referring to screen captures from the administrative console (WebSphere Administrative Console) used with the Application Server Advanced Single Server Edition.

Let's begin.

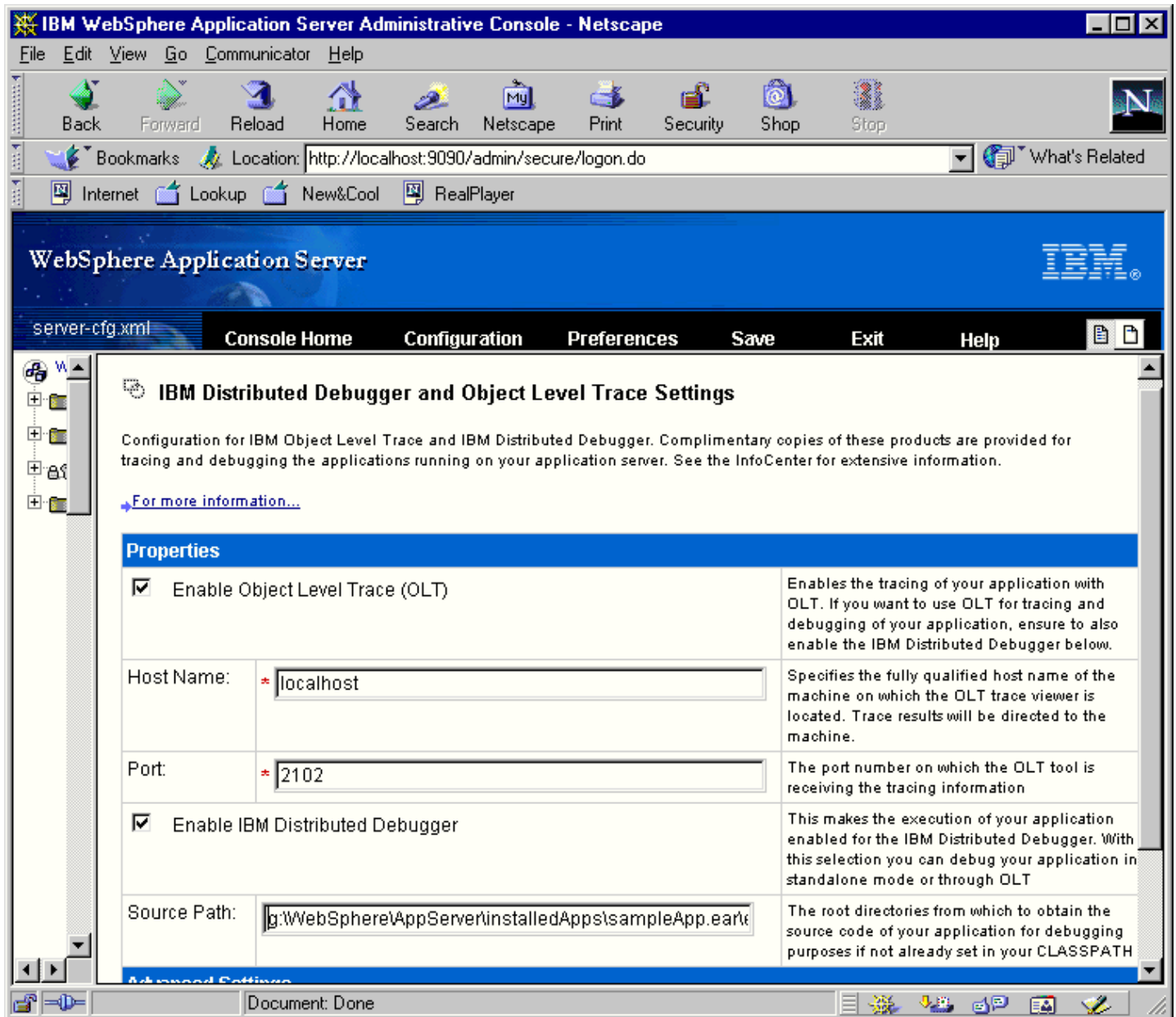
1. Start the Administrative Console:



2. Select **IBM Debug and OLT** in the topology view on the left side of the console for the default server on your node:



3. This will cause the **IBM Distributed Debugger and Object Level Trace Settings** page to display in the right-hand frame of the browser:

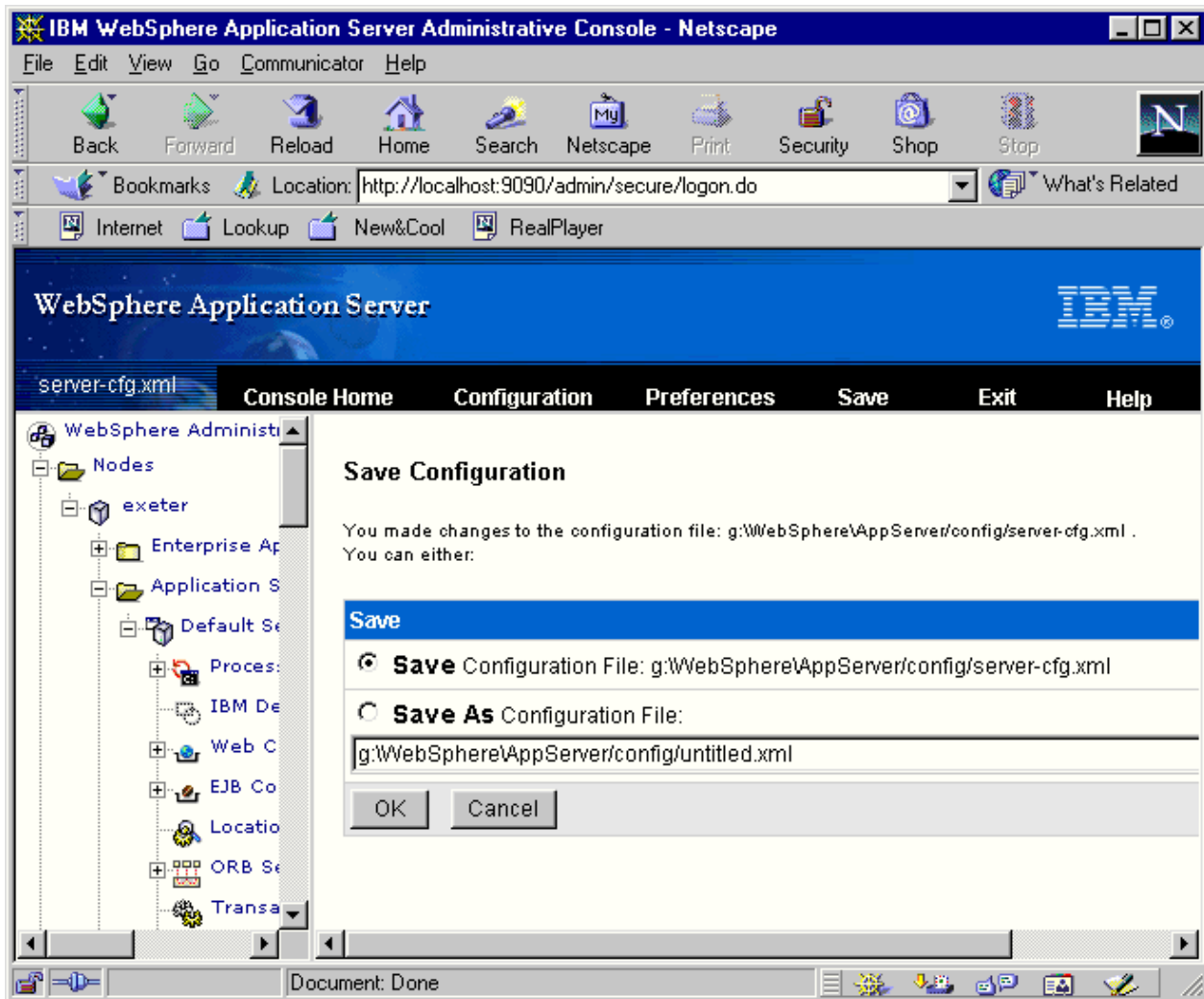


Select the **Enable Object Level Trace** and **Enable IBM Distributed Debugger** checkboxes. In the **Host Name** field, specify the OLT Host Name by entering the name of the machine where you will be running the OLT tool. The OLT Port should not be changed from its default of 2102, unless there is a port conflict (see the [Note](#) below). The entry in the **Source Path** field indicates to the Debugger where servlet source files, EJB source files, and JSP files can be found. Use semicolons to separate all entries on the Source Path. For the example, add the following to the source path:

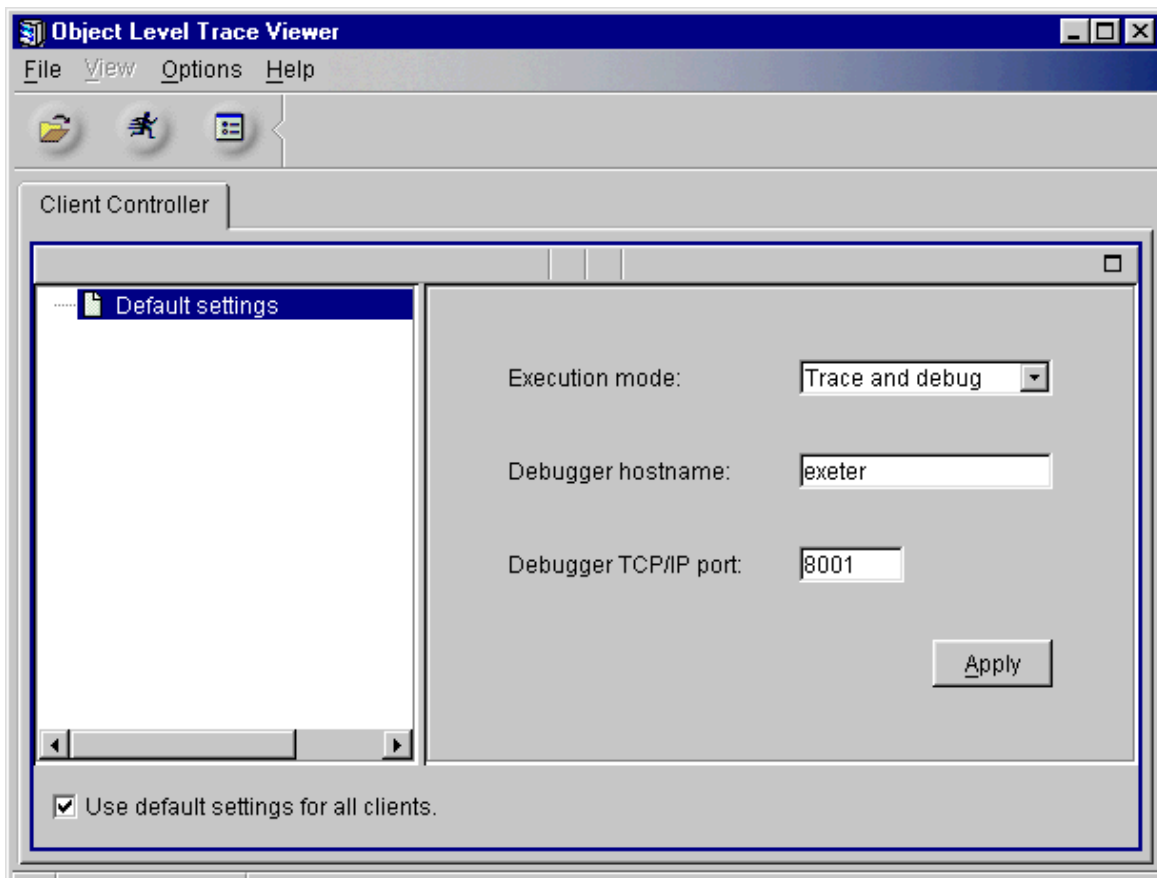
```
<WAS_ROOT>\installedApps\sampleApp.ear\examples.war;<WAS_ROOT>\installedApps\sampleApp.ear\examples.war\WEB-INF\classes
```

where WAS\_ROOT is where WebSphere Application Server is installed (eg. c:\WebSphere\AppServer). Click **OK** when you are finished.

4. To save your configuration, click **Save** and then click **OK** in the **Save Configuration** frame. At this point, you must restart the application server in order to pick up your changes:



- Once you have restarted the application server, start OLT by issuing the `olt` command at a command prompt. You will then see the following screen:

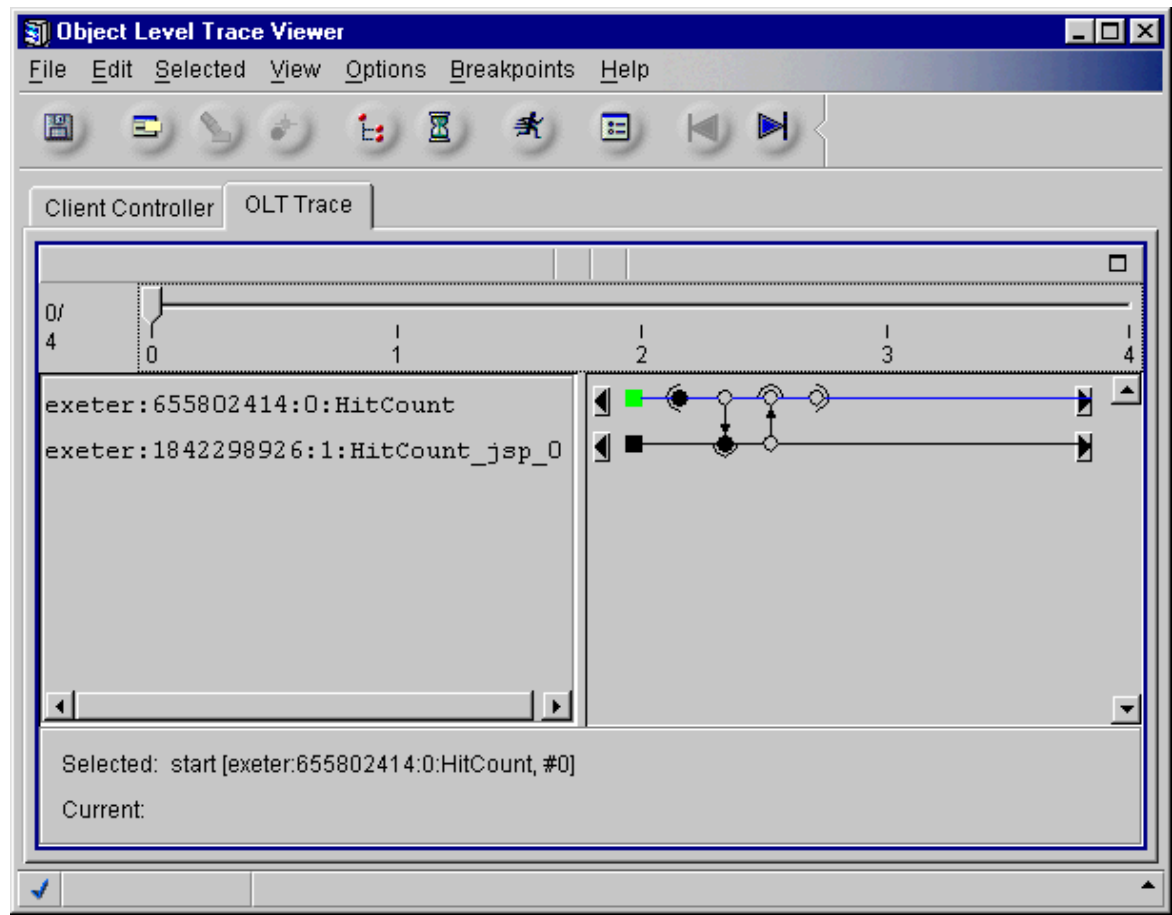


Ensure that the **Execution mode** is set to **Trace and debug**. If it is not, click the drop-down list and select it. In the **Debugger hostname** field, specify the name of the machine where the debugger tool is located. Assuming for this example that the Debugger is installed locally, enter the host name of the machine on which you are working. Keep the Debugger TCP/IP port set to 8001 (change it only if port 8001 is already in use on your machine). Click **Apply**.

6. Invoke the HitCount servlet by typing its URL in a Web browser:

http://<your\_machine>/webapp/examples/HitCount

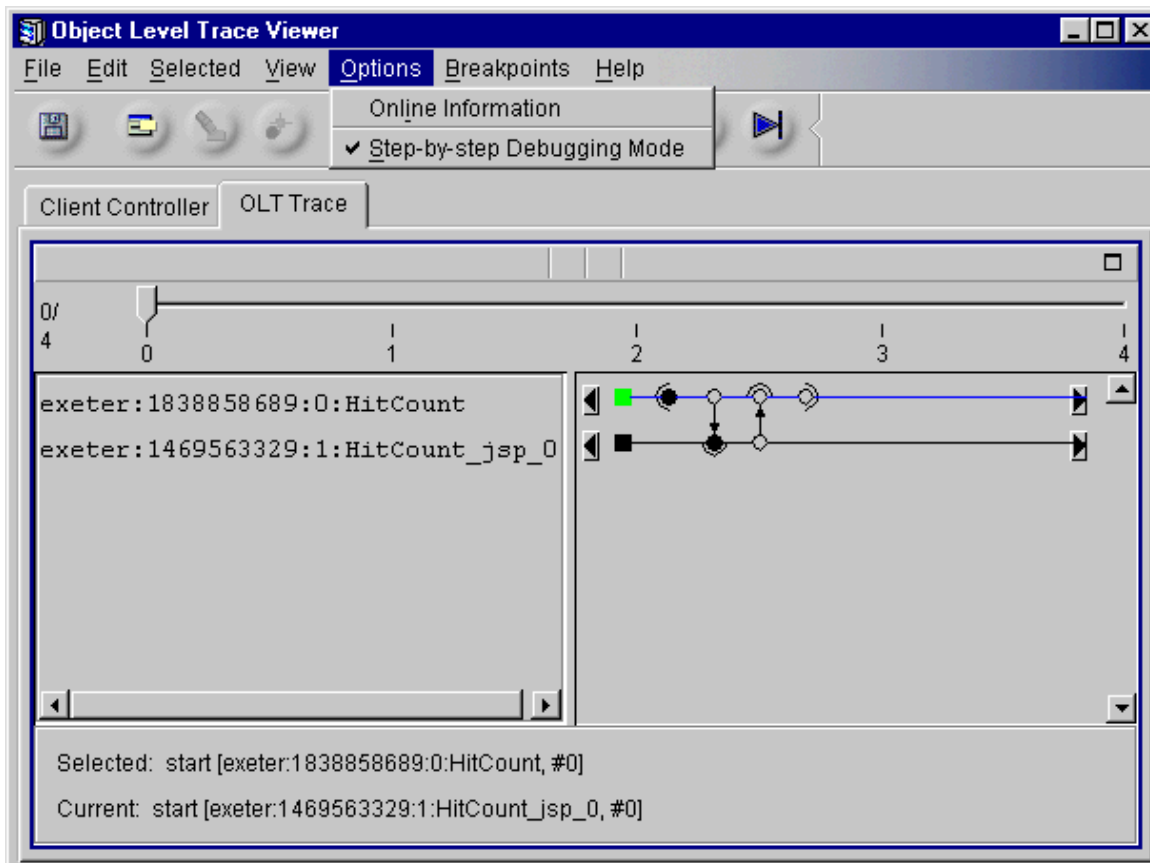
OLT will trace the initial calls to the HitCount servlet and JSP:



Your browser should now be displaying the following:



7. Let's enable step-by-step debugging now.

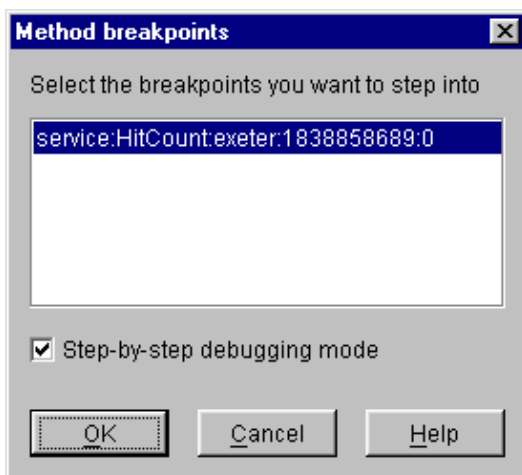




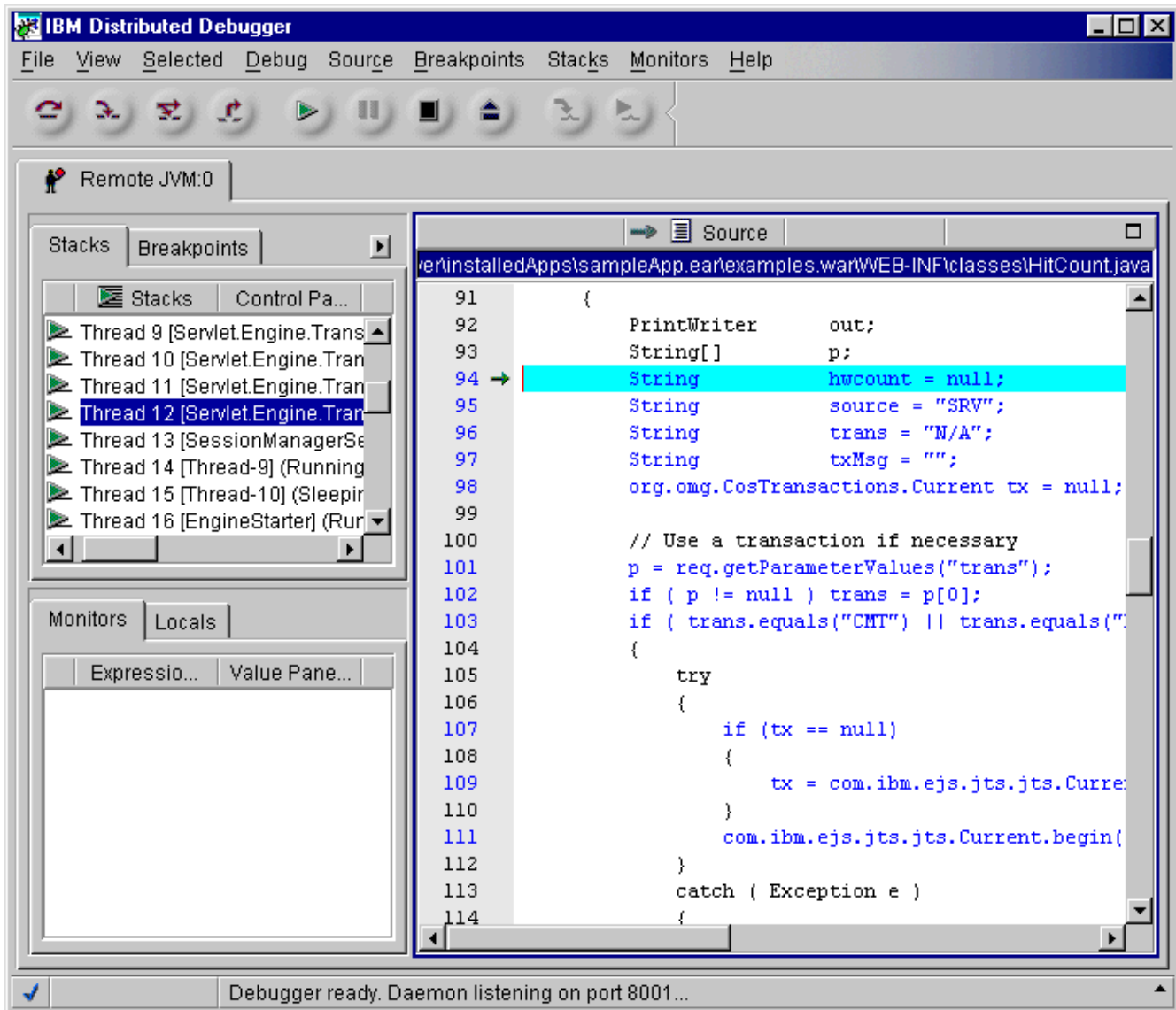
8. In the browser, choose the **Enterprise Java Bean** radio button and Transaction Type **None**. Click the **Increment** button, and OLT will start tracing the events taking place:



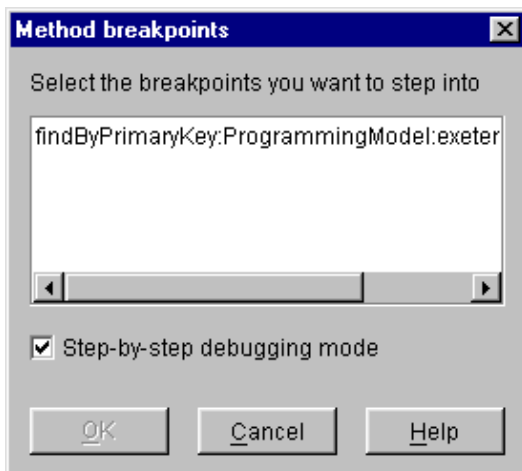
Because you selected step-by-step debugging mode, you will be asked if you want to step into every method that can be debugged. The first method is the service method of HitCount. The following dialog box will be displayed:



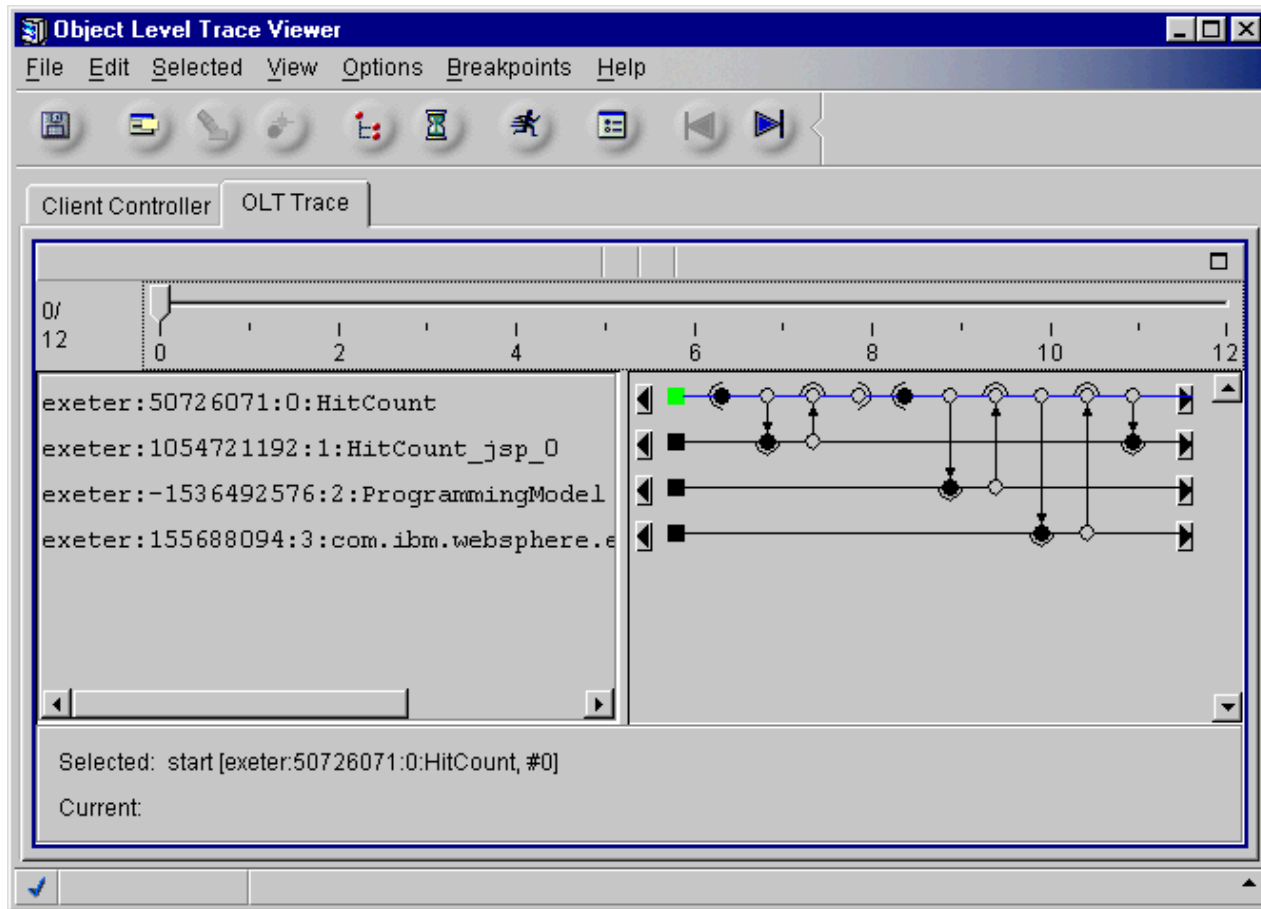
9. Highlight the service method and Click **OK**. This will cause the debugger to attach and suspend in the first executable line of the HitCount service method:



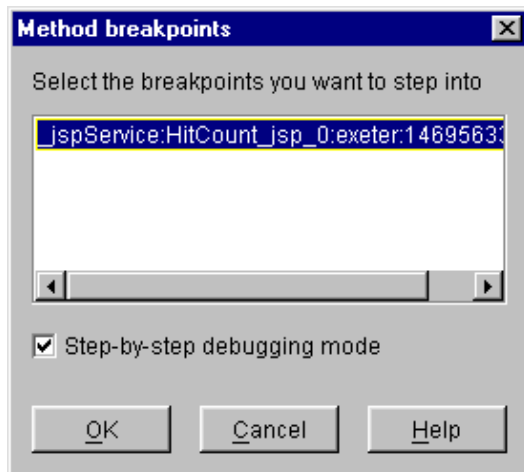
10. You can now choose to step over methods, step into others, or run the program. Click **Run** in the Debugger toolbar. Because OLT is in Step-by-Step mode, when it encounters the next method that can be debugged, it will ask you again (via the **Method breakpoints** dialog box) if you would like to step into the method. Indicate that you would like to skip breaking into the enterprise bean. An enterprise bean is displayed as two distinct objects in the trace: the enterprise bean object itself, and an object called the `ProgrammingModel`. The `ProgrammingModel` is an EJB Factory instance that is beyond application logic and is not intended to be debugged.
11. To skip breaking into a method, click **Cancel** in the **Method breakpoints** dialog:



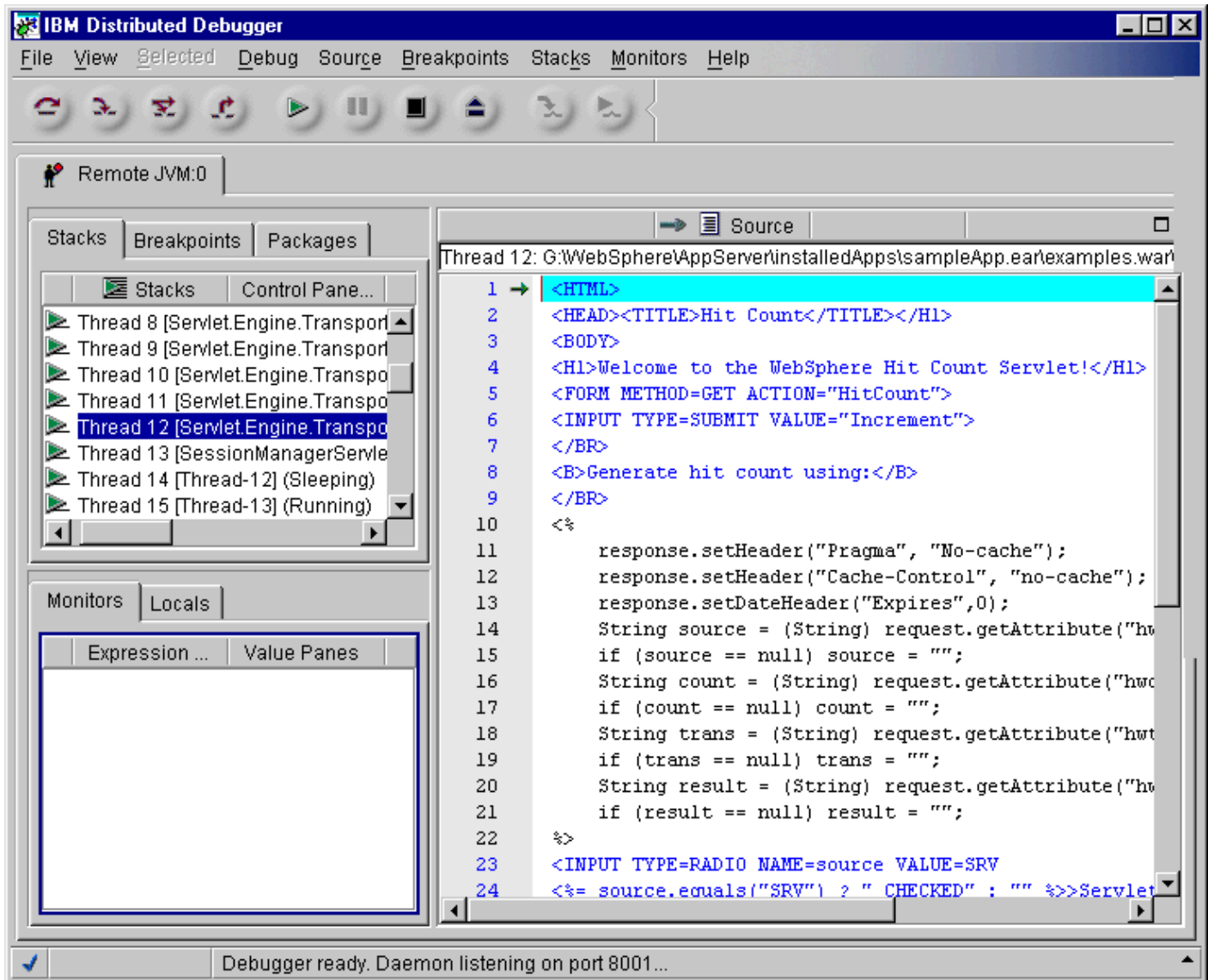
The EJB trace lines in the trace are `IncBean` and `ProgrammingModel`, as shown:



12. To break into the JSP, step into the `_jspService` method:



The debugger will enter the JSP code, displaying the following:



13. You can now choose to step over, step debug, run, and so on. If you click **Run**, then HitCount should run to completion. Note, your browser may have timed out by this time. The Troubleshooting section of the OLT/Debugger documentation describes how to increase the browser timeout.

**Note:**

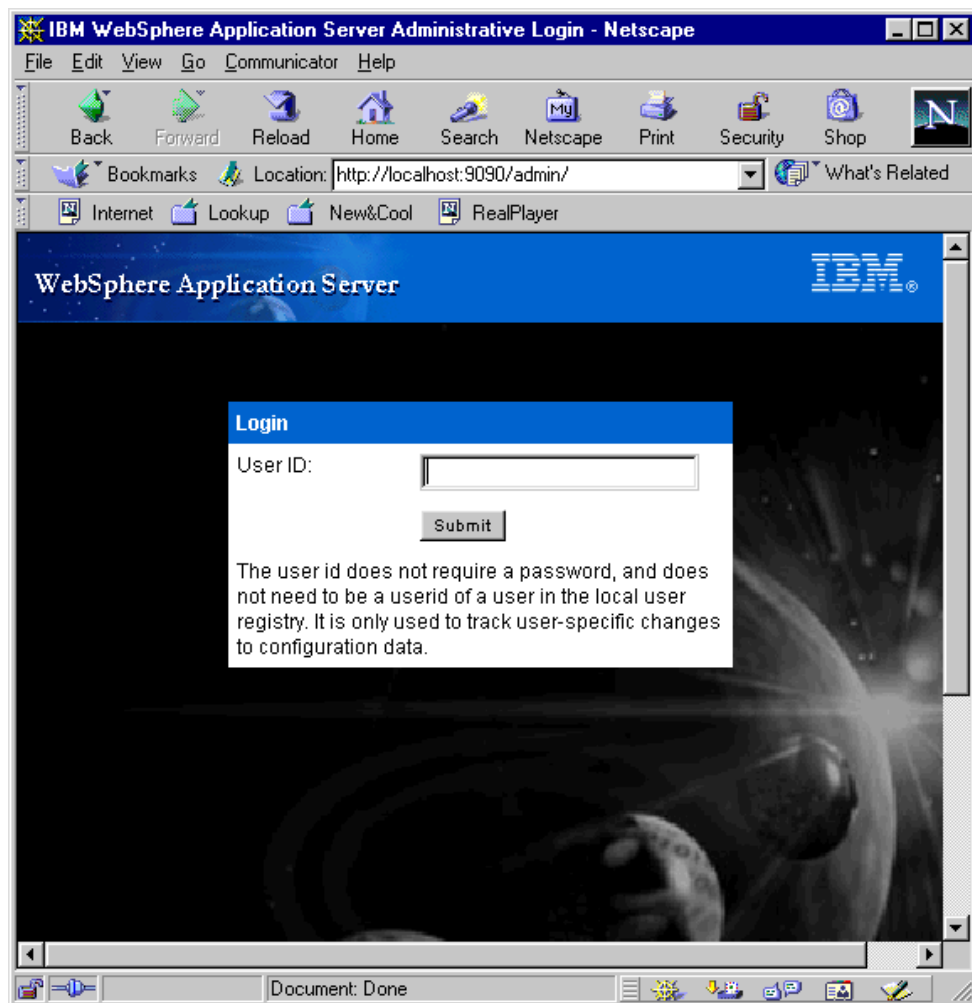
If you modify the Port in the **IBM Distributed Debugger and Object Level Trace settings** frame to a port other than 2102, you must start OLT and modify the **OLT Server TCP/IP port** accordingly. The OLT Server TCP/IP port is set in the **Browser Preferences** dialog box (accessed by selecting **File > Preferences** from the OLT menu bar and then selecting the **OLT** node). If there is a port conflict and you are unable to start OLT, go to the %userprofile%\DbgProf directory and modify the OLT\_TRC\_SRVAPP\_PORT in the dertrenv.dat file accordingly.

# StockQuote tutorial: Using Debugger and OLT

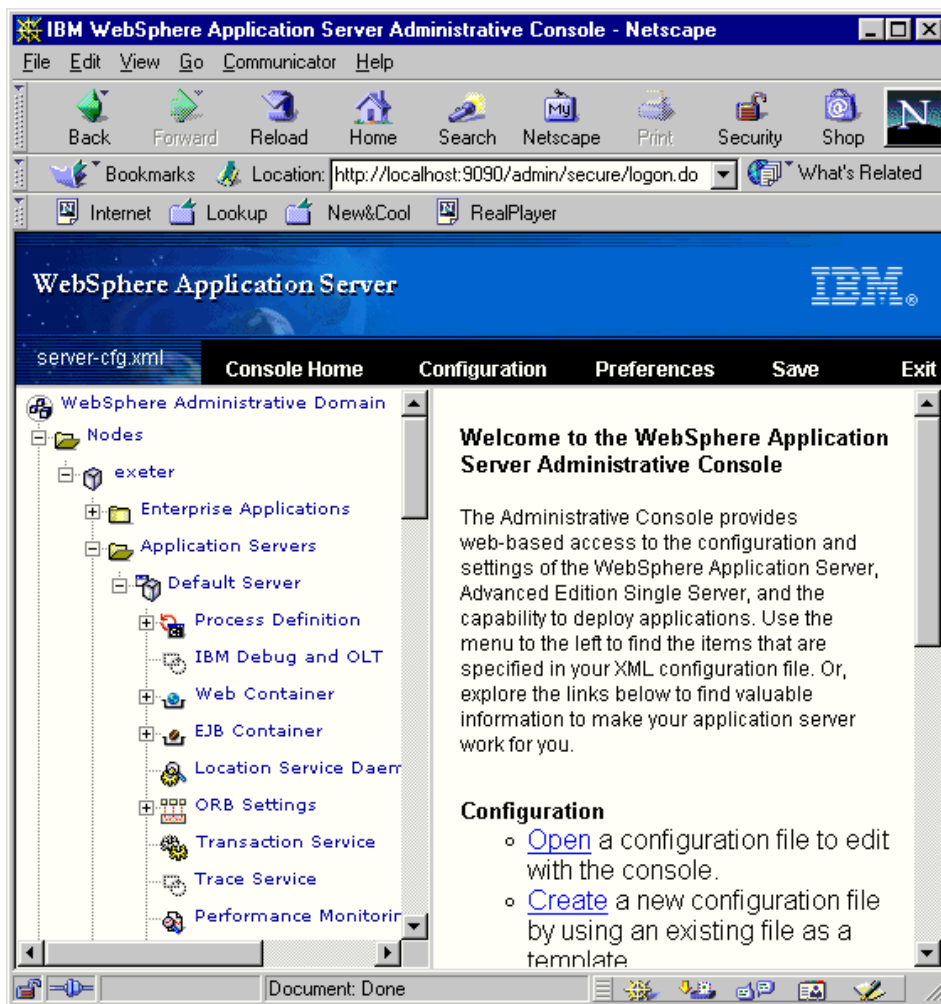
StockQuote is one of the samples that ships with IBM WebSphere Application Server. It is installed if, during WebSphere Application Server installation, you choose to install the Samples. This tutorial demonstrates how to use the IBM Distributed Debugger and Object Level Trace (OLT) on the StockQuote sample, referring to screen captures from the administrative console (WebSphere Administrative Console) used with the Application Server Advanced Single Server Edition.

Let's begin.

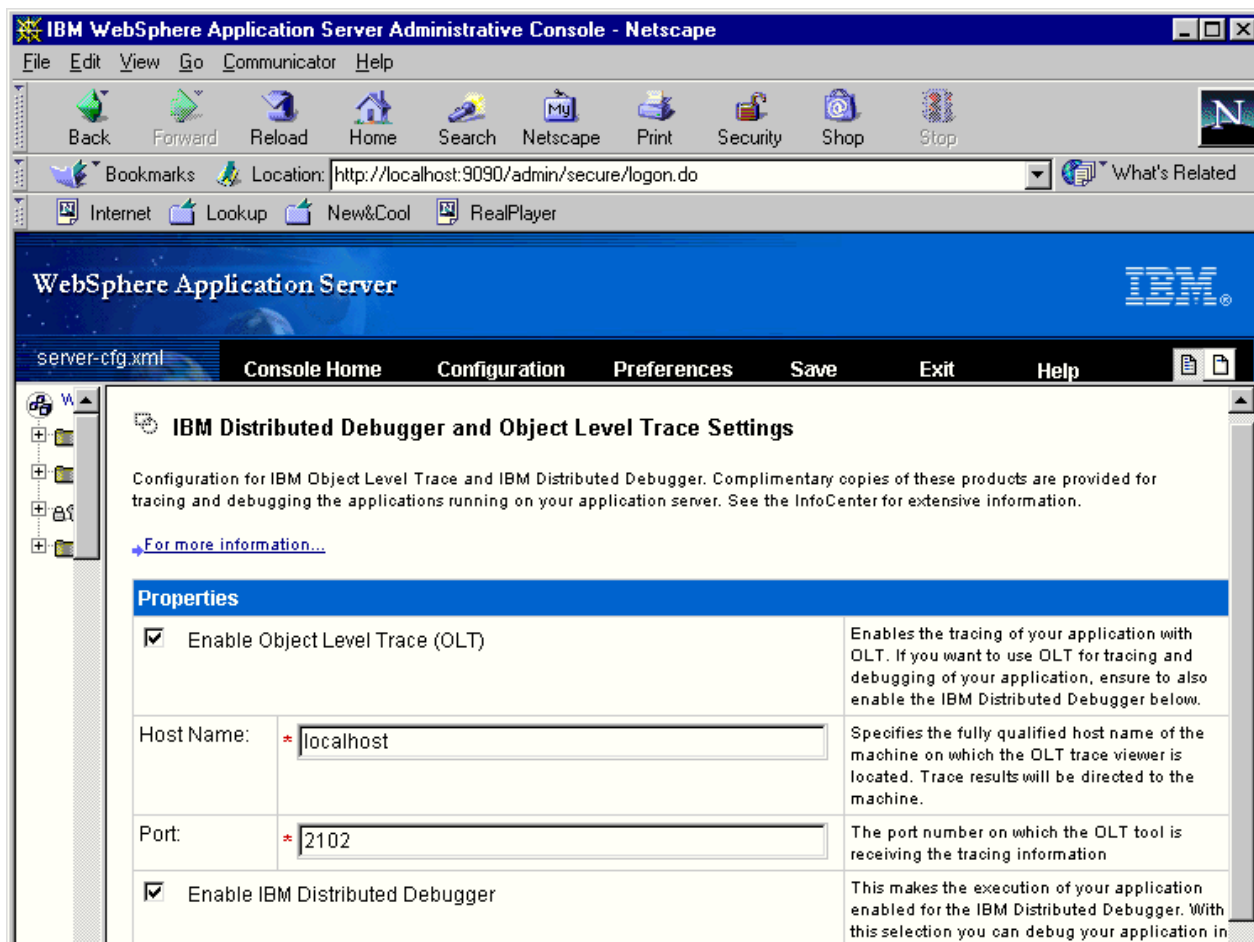
1. Start the Administrative Console:

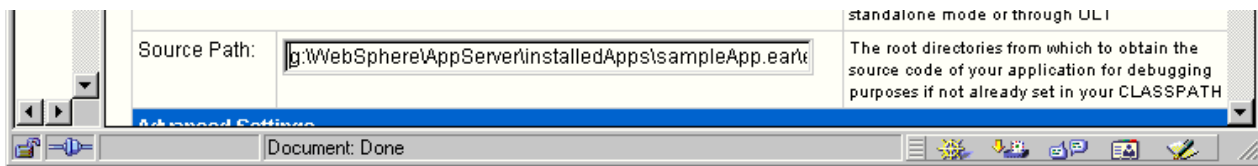


2. Select **IBM Debug and OLT** in the topology view on the left side of the console for the default server on your node:



3. This will cause the **IBM Distributed Debugger and Object Level Trace Settings** page to display in the right-hand frame of the browser:



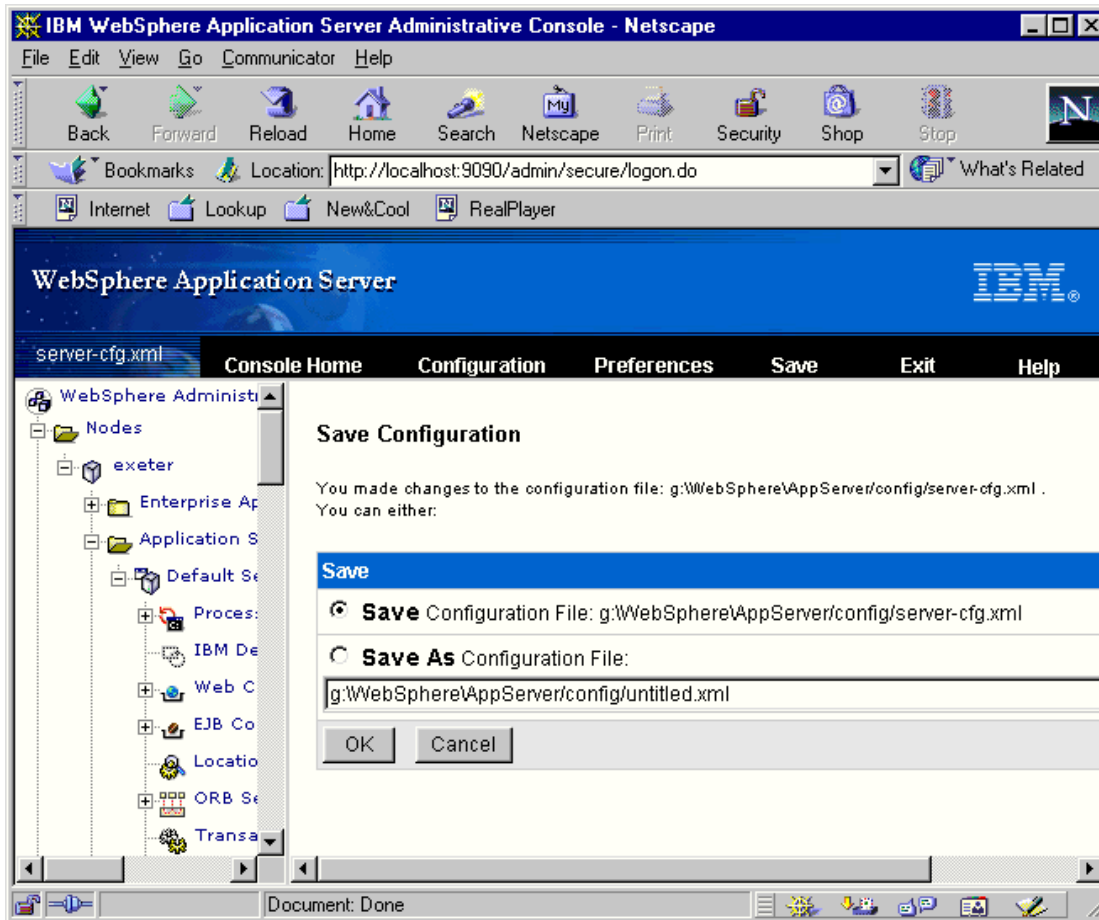


Select the **Enable Object Level Trace** and **Enable IBM Distributed Debugger** checkboxes. In the **Host Name** field, specify the OLT Host Name by entering the name of the machine where you will be running the OLT tool. The OLT Port should not be changed from its default of 2102, unless there is a port conflict (see the [Note](#) below). The entry in the **Source Path** field indicates to the Debugger where servlet source files, EJB source files, and JSP files can be found. Use semicolons to separate all entries on the Source Path. For this sample, add the following to the source path:

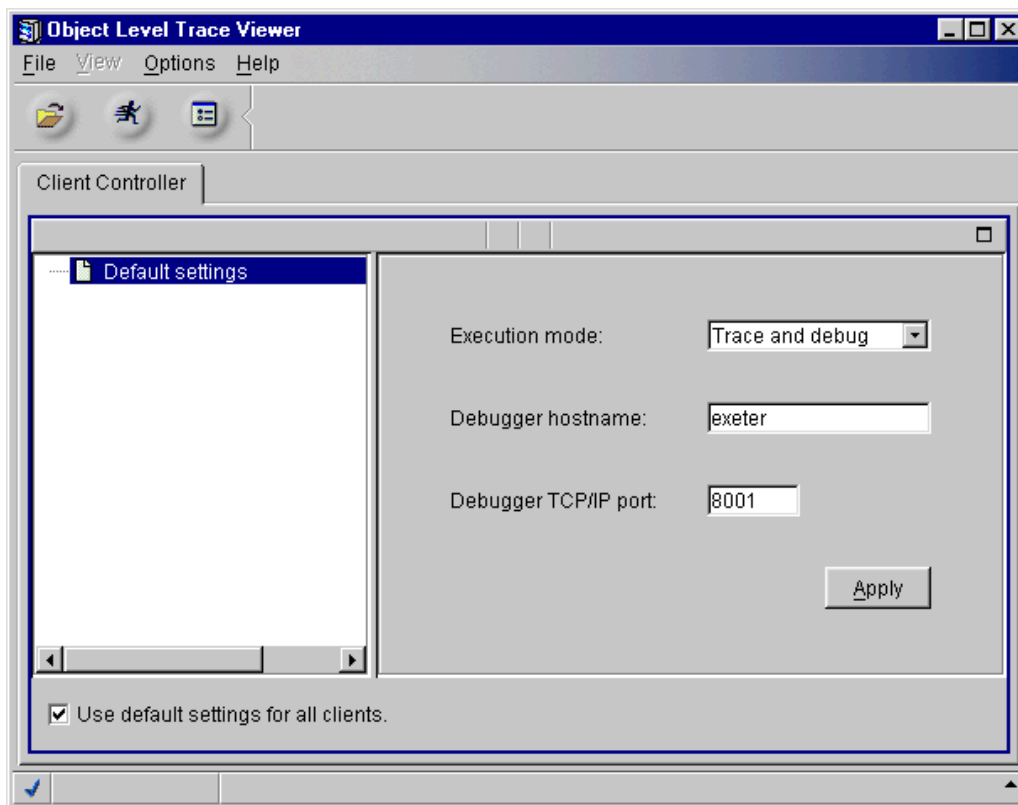
`<WAS_ROOT>\installedApps\Samples.ear\Samples.war\StockQuote;<WAS_ROOT>\installedApps\Samples.ear\Samples.war\WEB-INF\classes\WebSphereSamples\StockQuote`

where WAS\_ROOT is where WebSphere Application Server is installed (eg. `c:\WebSphere\AppServer`). Click **OK** when you are finished.

4. To save your configuration, click **Save** and then click **OK** in the **Save Configuration** frame. At this point, you must restart the application server in order to pick up your changes:



5. Once you have restarted the application server, start OLT by issuing the `olt` command at a command prompt. You will then see the following screen:



Ensure that the **Execution mode** is set to **Trace and debug**. If it is not, click the drop-down list and select it. In the **Debugger hostname** field, specify the name of the machine where the debugger tool is located. Assuming for this example that the Debugger is installed locally, enter the host name of the machine on which you are working. Keep the Debugger TCP/IP port set to 8001 (change it only if port 8001 is already in use on your machine). Click **Apply**.

6. Invoke the IBM WebSphere Application Server Samples and select the **StockQuote** sample. You will see the following in your web browser:

**IBM WebSphere Samples Gallery - Microsoft Internet Explorer**

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print

Address <http://monaco/WSsamples/index.html> Go Links

# IBM WebSphere Samples

[Getting Started](#)  
[Database Config](#)  
[Trade Setup](#)

*Single Samples:*

[Bean Scripting](#)  
[StockQuote](#)  
[XML](#)  
[HelloEJB](#)  
[Increment](#)  
[Account](#)  
[Transfer](#)

[Trade](#)

*Integrated Samples*

[YourCo](#)

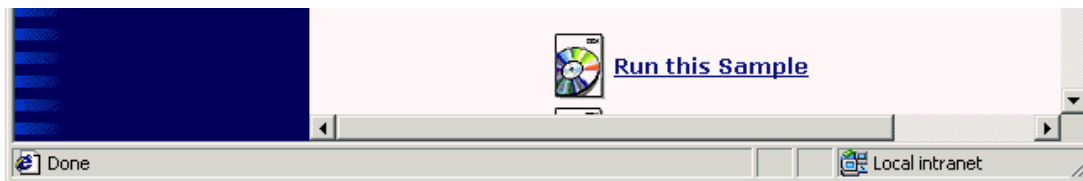
## Stock Quote Sample

This sample demonstrates a Pervasive Computing solution using WebSphere Application Server. It can be called from any of the following types of clients:

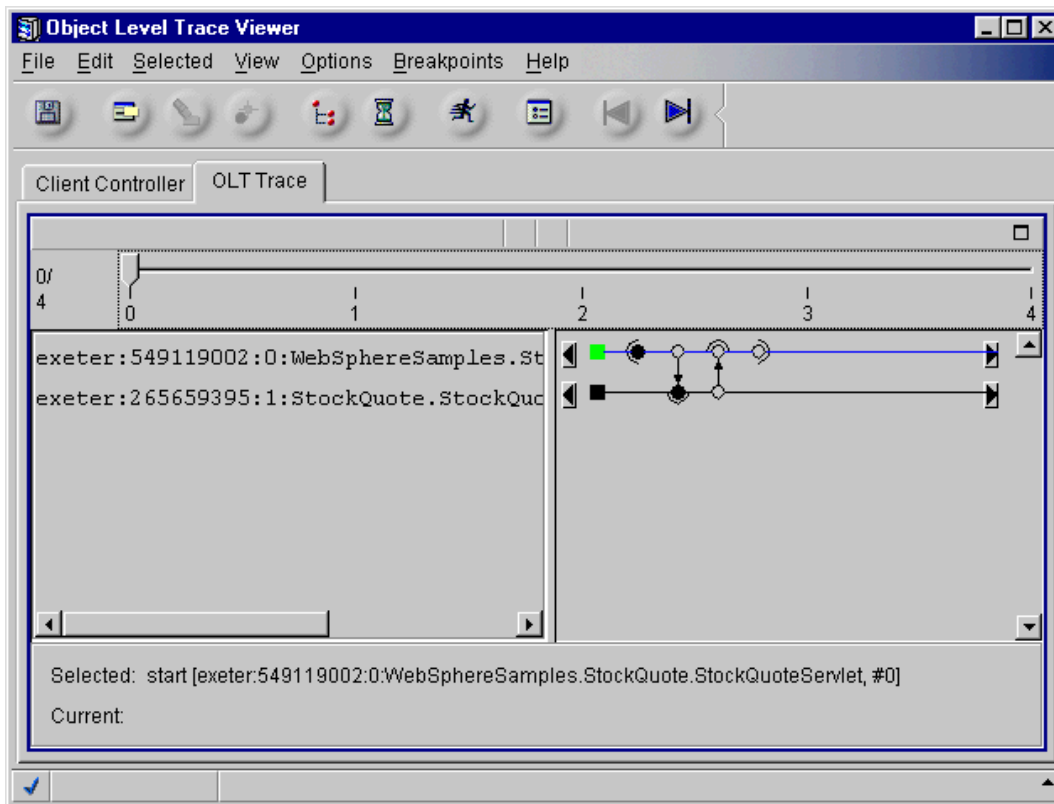
Client Type	Data Type Returned
HTML	HTML
Speech	VXML
Wireless	WML

**Requirements:** The machine executing this servlet must be connected to the Internet, and have access through any firewalls, in order to retrieve the stock information requested.

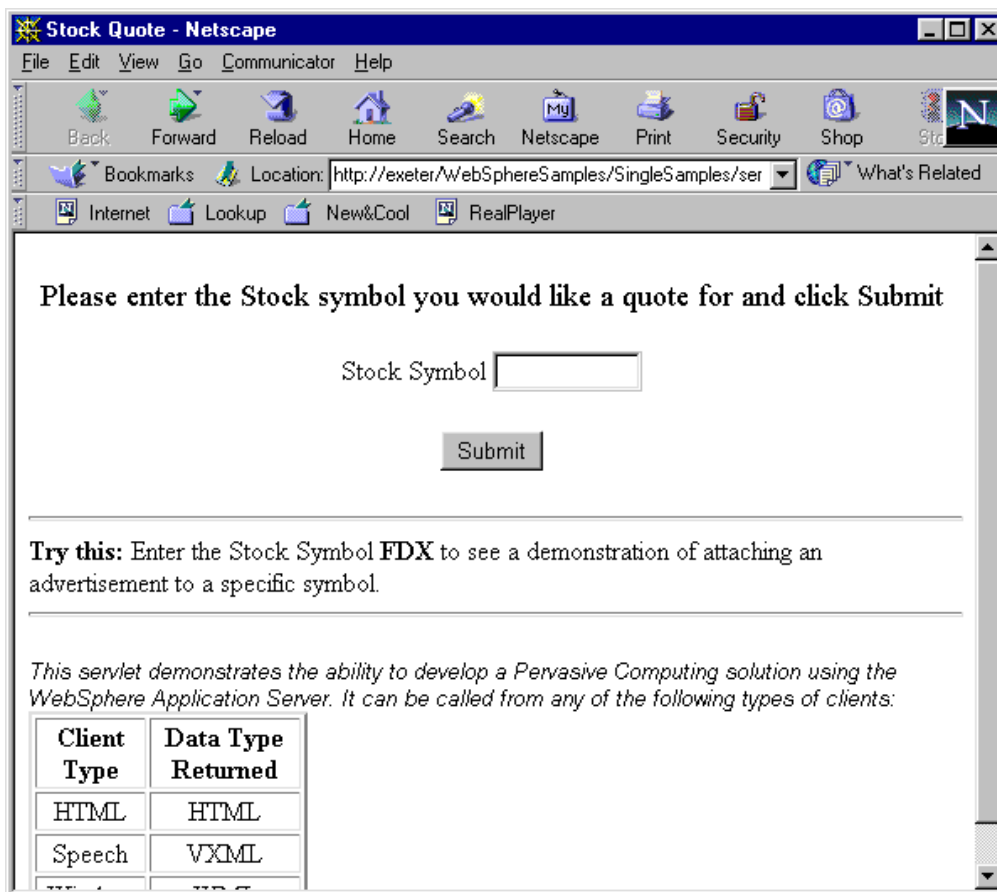




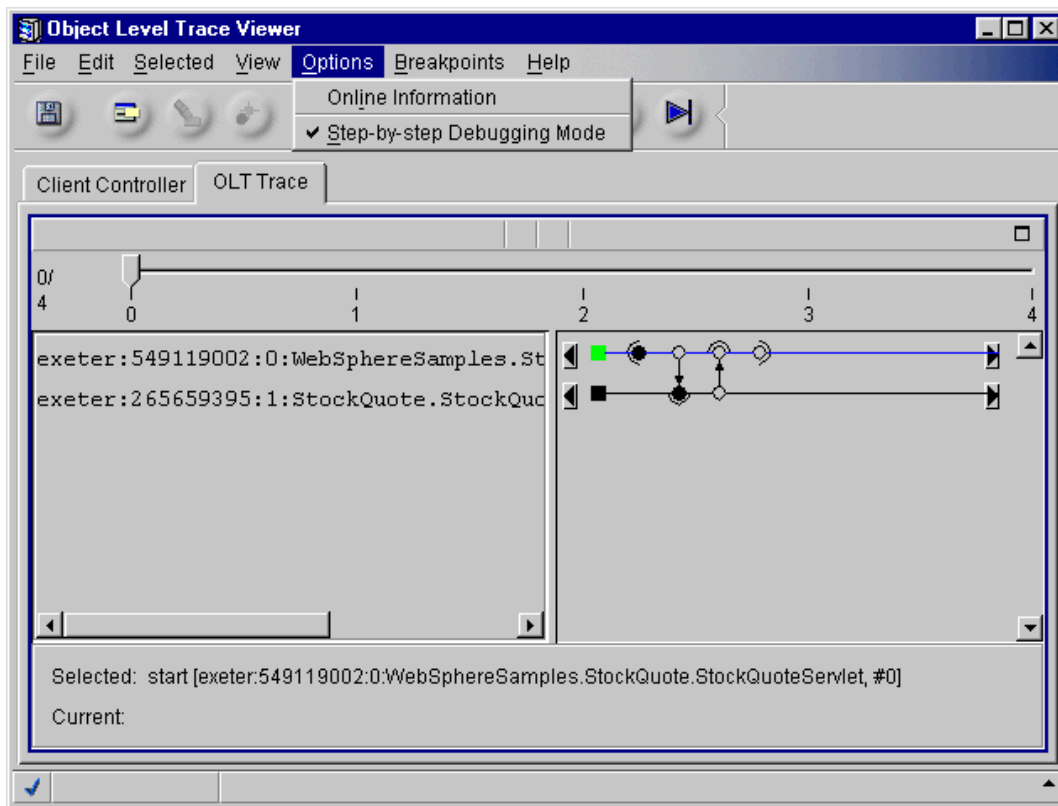
- Click **Run this Sample**. OLT will trace the initial calls to the StockQuote servlet and JSP:



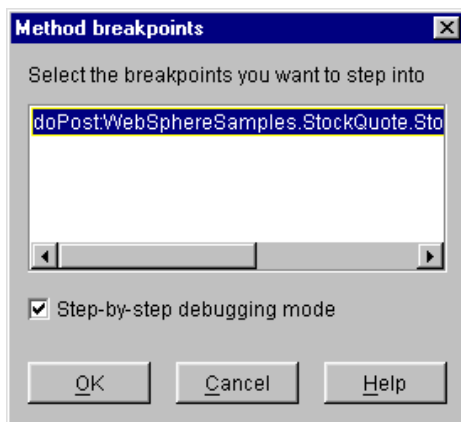
- Your browser should then display the following:



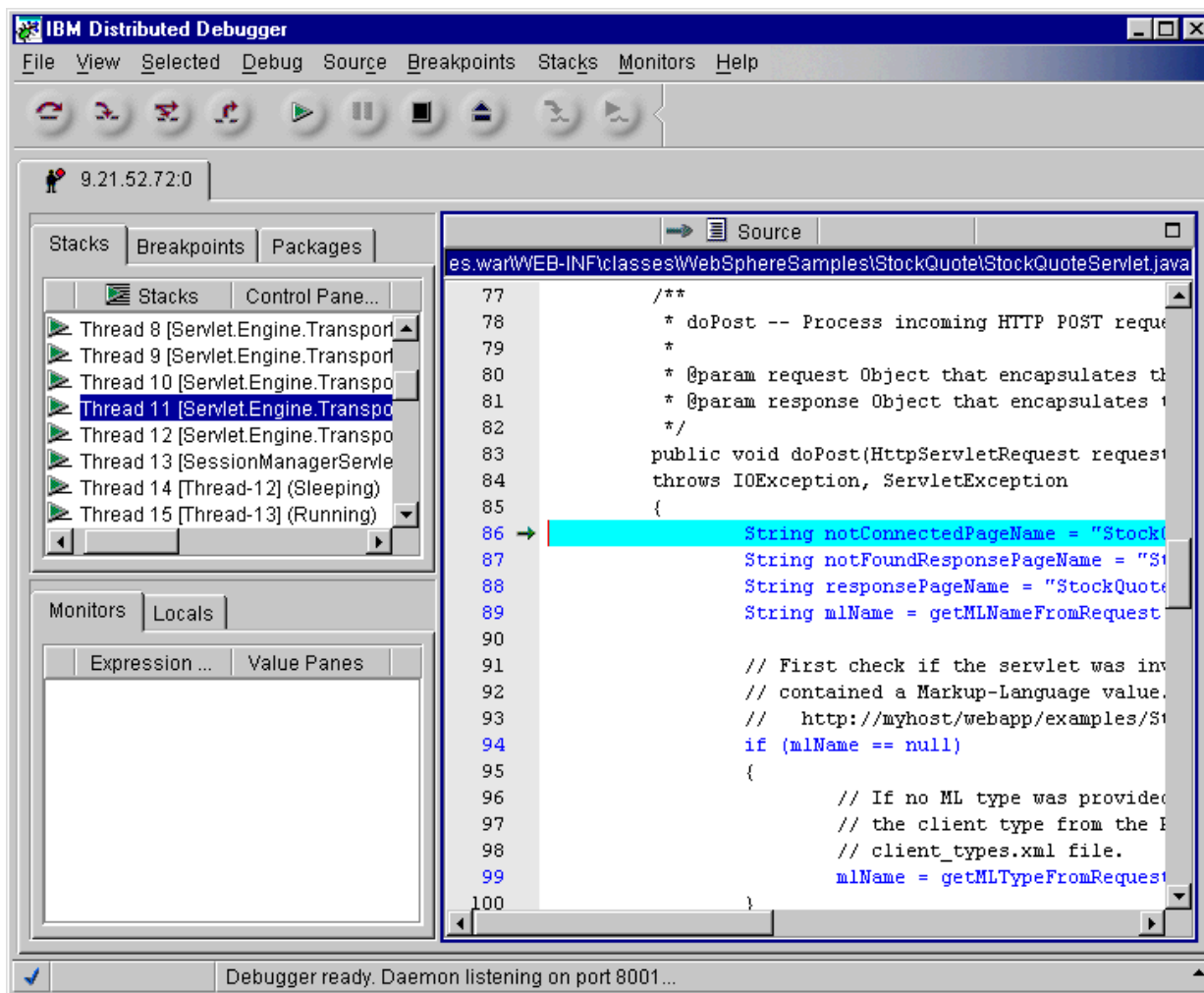
9. Let's enable step-by-step debugging now.



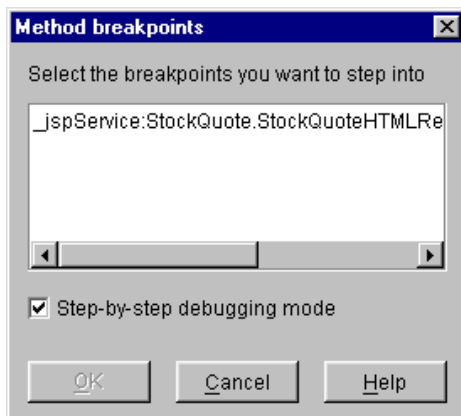
10. In the browser, enter **IBM** in the **Stock Symbol** field and then click **Submit**. Because you selected step-by-step debugging mode, you will be asked if you want to step into every method that can be debugged. The following dialog box will be displayed:



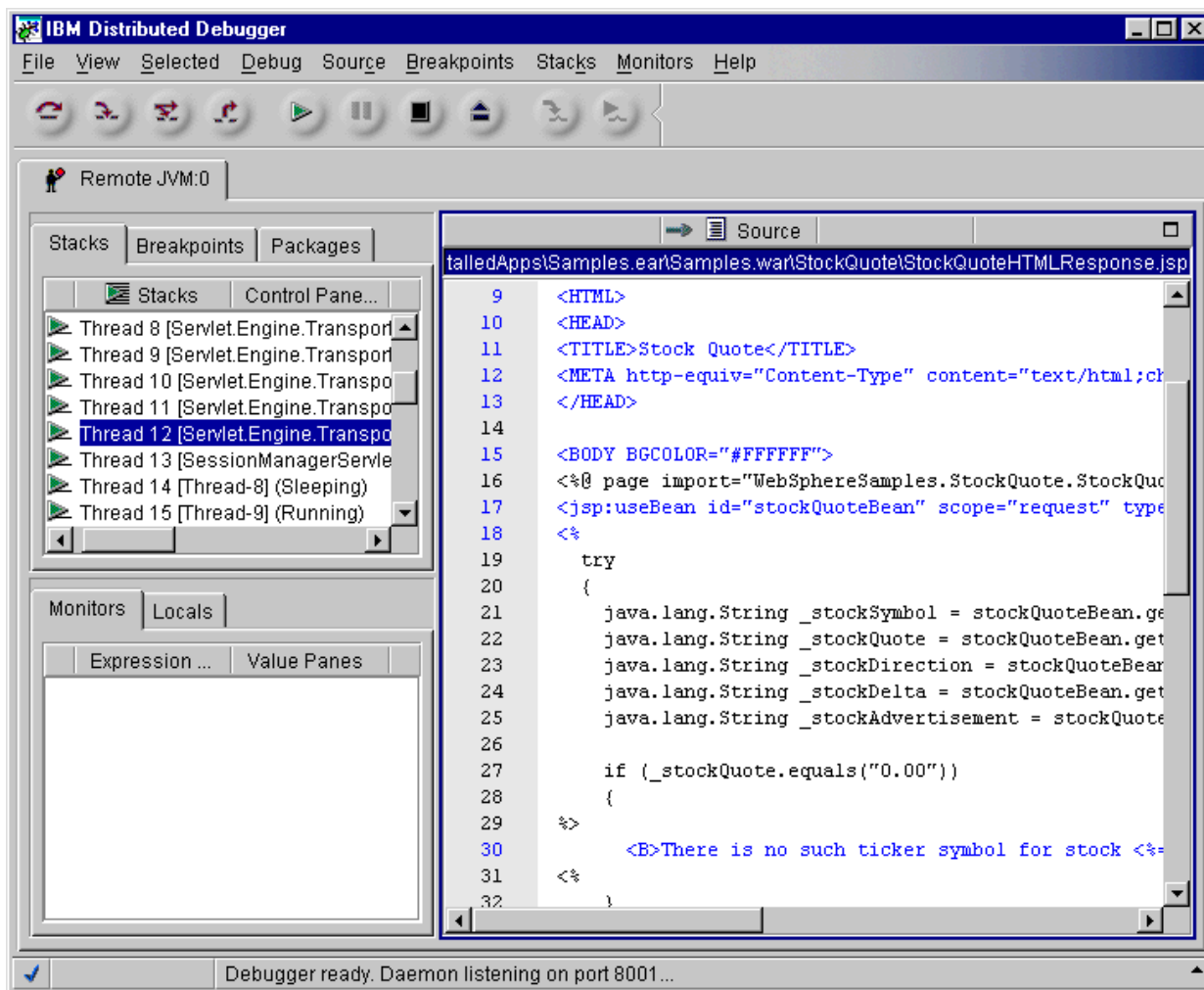
11. Highlight the doPost method and Click **OK**. This will cause the debugger to attach and suspend in the first executable line of the StockQuoteServlet doPost method:



12. You can now choose to step over methods, step into others, or run the program. Click **Run** in the Debugger toolbar. Because OLT is in Step-by-Step mode, when it encounters the next method that can be debugged, it will ask you again (via the **Method breakpoints** dialog box) if you would like to step into the method:



13. Let's step into the JSP, highlight the `_jspService` method in the **Method Breakpoints** dialog box, and click **OK**. Or, if you wanted to skip breaking into this method, click **Cancel** in the **Method Breakpoints** dialog box.
14. The debugger will enter the JSP code, displaying the following:



15. You can now choose to step over, step debug, run, and so on. If you click **Run**, then StockQuote should run to completion. Note, your browser may have timed out by this time. The Troubleshooting section of the OLT/Debugger documentation describes how to increase the browser timeout.

**Note:**

If you modify the Port in the **IBM Distributed Debugger and Object Level Trace settings** frame to a port other than 2102, you must start OLT and modify the **OLT Server TCP/IP port** accordingly. The OLT Server TCP/IP port is set in the **Browser Preferences** dialog box (accessed by selecting **File > Preferences** from the OLT menu bar and then selecting the **OLT** node). If there is a port conflict and you are unable to start OLT, go to the %userprofile%\DbgProf directory and modify the OLT\_TRC\_SRVAPP\_PORT in the dertenv.dat file accordingly.

## 6.10: Backing up and restoring administrative configurations

To start with, the product provides multiple [application server configuration files](#). It is recommended that you make backup copies of the files before modifying them, in case you want to restore your initial configuration at a later time.

Because the configuration file holds the complete product configuration, and you can make backup copies of a configuration file at any time, consider doing so periodically. It will allow you to return to a particular configuration should something go wrong.