

Administration -- table of contents

[6: Administer applications](#)

[6.1: Quick reference for administration](#)

Preparing the runtime

[6.2: Preparing to host applications](#)

[6.2.1: The default resources](#)

Packaging and installing applications

[6.4: Installing applications and setting classpaths](#)

[6.4.1: Setting classpaths](#)

[6.4.2: Installing application files](#)

Maintaining applications

[6.5: Maintaining and updating applications](#)

[6.5.2: Actions that require a restart](#)

Quick reference

[6.6: Tools and resources quick reference](#)

User assistance (help files)

[6.6.0: About user assistance](#)

Java administrative console

[6.6.0.1: Using the Java administrative console](#)

[6.6.0.1.1: Starting and stopping Java administrative consoles](#)

[Configuring new resources from the Topology tree](#)

[Configuring new resources from the Type tree](#)

[6.6.0.1.3: Configuring resources with the Java administrative console](#)

[Configuring new resources from the Topology tree](#)

- Configuring new resources from the Type tree
- 6.6.0.1.4: Actions on the right-click menus in the Java administrative console
- 6.6.0.1.6: Find the best place to start in the Java administrative console
- 6.6.0.1.7: The Topology and Type trees in the Java administrative console
- 6.6.0.1.8: Starting, stopping, and pinging resources with the Java administrative console
- 6.6.0.1.9: Removing resources with the Java administrative console
- 6.6.0.1.10: Editing resource properties
- 6.6.0.1.11: Buttons in the Java administrative console
- 6.6.0.1.12: Property recommendations by operating system
- 6.6.0.1.13: Java command line arguments reference
- 6.6.0.1.14: Actions on the right-click menus of resources
- 6.6.0.1a: Starting and stopping Java administrative consoles

Command line tools

- 6.6.0.2: Command line administration
 - 6.6.0.2.1: XMLConfig command line interface for XML configuration
 - XMLConfig - Command syntax
 - XMLConfig - Example of a full export
 - XMLConfig - Example of a partial export
 - XMLConfig grammar
 - XMLConfig - Using the tool programmatically
 - XMLConfig - Passwords and variable substitution
 - XMLConfig - User registry searches
 - wartoxmlconfig script
 - Troubleshooting XMLConfig
 - 6.6.0.2.2: WebSphere Control Program

Web administrative console

- 6.6.0.3: Web administrative console overview
 - 6.6.0.3.1: Tasks overview
 - Performing tasks
 - Properties for performing tasks
 - 6.6.0.3.2: Resources overview
 - Administering resources
 - Properties for configuring resources
 - 6.6.0.3.3: Create objects overview
 - Creating objects
 - Properties for creating objects
 - 6.6.0.3.4: Export to XML overview
 - Exporting the workspace to XML
 - 6.6.0.3.5: Submit and commit modifications overview
 - Submitting and committing modifications
 - Properties for submitting and committing modifications
- 6.6.0.3a: Starting and stopping the Web administrative console
- 6.6.0.3c: Buttons in the Web administrative console
- 6.6.0.3e: Using the Web administrative console

Editing property files by hand

6.6.0.4: Overview of editing property files by hand

6.6.0.4.1: Dynamic caching of Servlets and JSPs

dynacache.xml file

Quick reference - dynacache.xml file

servletcache.xml file

Removing entries from the cache

Caching examples

Enterprise applications

6.6.1: Administering applications (overview)

6.6.1.1: Administering enterprise applications with the Java administrative console

6.6.1.1.1: Configuring new enterprise applications with the Java administrative console

6.6.1.1.5: Editing the contents of enterprise applications

Nodes

6.6.2: Administering nodes (overview)

6.6.2.1: Administering nodes with the Java administrative console

6.6.2.1.1: Configuring new nodes with the Java administrative console

6.6.2.3: Administering nodes with the Web console

6.6.2.4: Property files pertaining to nodes

6.6.2.4.1: Multiple administrative servers on a node

Application servers

6.6.3: Administering application servers

6.6.3.0: Application server properties

6.6.3.1: Administering application servers with the Java administrative console

6.6.3.1.1: Configuring new application servers

6.6.3.1.2: Starting and stopping application servers with the Java administrative console

6.6.3.3: Administering application servers with the Web console

6.6.3.4: Property files pertaining to application servers

6.6.3.5: Invoking your own classes during application server startup and shutdown

6.6.7: Administering servlet engines

6.6.7.0: Servlet engine properties

6.6.7.1: Administering servlet engines with the Java administrative console

6.6.7.1.1: Configuring new servlet engines with the Java administrative console

6.6.7.3: Administering servlet engines with the Web console

6.6.7.4: Property files pertaining to servlet engines

Web applications

6.6.8: Administering Web applications (overview)

6.6.8.0: Web application properties

6.6.8.1: Administering Web applications with the Java administrative console

6.6.8.1.1: Configuring new Web applications with the Java administrative console

6.6.8.1.6: Converting WAR files with the Java administrative console

6.6.8.3: Administering Web applications with the Web console

Servlets

6.6.9: Administering servlets (overview)

6.6.9.0: Servlet properties

6.6.9.1: Administering servlets with the Java administrative console

6.6.9.1.1: Configuring new servlets with the Java administrative console

6.6.9.3: Administering servlets with the Web console

6.6.9.4: Property files pertaining to servlets

JSP files

6.6.10: Administering JSP files (overview)

6.6.10.0: JSP file properties

6.6.10.1: Administering JSP files with the Java administrative console

6.6.10.1.1: Configuring new JSP files with the Java administrative console

HTTP sessions

6.6.11: Administering HTTP session support (overview)

6.6.11.0: Session Manager properties

6.6.11.1: Administering session support with the Java administrative console

6.6.11.1.1: Configuring new Session Managers with the Java administrative console

User profiles

6.6.12: Administering user profile support (overview)

6.6.12.0: User Profile Manager properties

6.6.12.1: Administering user profile support with the Java administrative console

6.6.12.1.1: Configuring new User Profile Managers with the Java administrative console

Servlet redirectors

Database connections

- 6.6.14: Administering database connections (overview)
 - 6.6.14.0: Properties of JDBC drivers
 - 6.6.14.0.1: Properties of data sources
 - 6.6.14.1: Administering database connections with the Java console
 - 6.6.14.1.1: Configuring new database connections with the Java administrative console
 - 6.6.14.3: Administering database connections with the Web console
 - 6.6.14.4: Property files pertaining to database connections
 - 6.6.14.5: Additional administrative tasks for specific databases
 - 6.6.14.6: Notes about various databases
 - 6.6.14.7: Notes about InstantDB
 - 6.6.14.10: Failover support with HACMP
 - 6.6.14.11: Recovering from data source configuration problems using the XAResources file

Virtual hosts

- 6.6.16: Administering virtual hosts (overview)
 - 6.6.16.0: Properties of virtual hosts
 - 6.6.16.1: Administering virtual hosts with the Java administrative console
 - 6.6.16.1.1: Configuring new virtual hosts with the Java administrative console
 - 6.6.16.3: Administering virtual hosts with the Web console
 - 6.6.16.4: Property files pertaining to virtual hosts

Web resources

- 6.6.17: Administering Web resources (overview)
 - 6.6.17.0: Web resource properties
 - 6.6.17.1: Administering Web resources with the Java administrative console
 - 6.6.17.1.1: Configuring new Web resources with the Java administrative console

Security

- 6.6.18: Securing applications
 - 6.6.18.1: Securing applications
 - 6.6.18.1.1: Securing applications
 - 6.6.18.1.1a: Specifying global settings
 - 6.6.18.1.1b: Configuring application security
 - 6.6.18.1.1c: Configuring custom method groups
 - Viewing custom method groups
 - 6.6.18.1.1d: Configuring resource security
 - Default method groups
 - 6.6.18.1.1e: Configuring permissions
 - 6.6.18.1.4: Properties related to security
 - 6.6.18.1.4a: Properties for configuring global settings
 - General settings of the Configure Global Settings task

- Application Default settings of the Configure Global Settings task
- Authentication Mechanism settings
- User Registry settings of the Configure Global Settings task
 - Supported directory services
- 6.6.18.1.4b: Properties for configuring application security
- 6.6.18.1.4c: Properties for configuring method groups
- 6.6.18.1.4d: Properties for configuring resource security
- 6.6.18.1.4e: Properties for configuring permissions
- 6.6.18.1.4f: Properties for the security search dialog
- 6.6.18.1a: About setting global security settings
 - 6.6.18.1a01: About enabling security
 - 6.6.18.1a02: About setting application security defaults
 - 6.6.18.1a03: About specifying how to authenticate users
 - 6.6.18.1a04: About providing authentication mechanism details
- 6.6.18.1b: About configuring application security with the Java console
- 6.6.18.1c: About assigning method groups with the Java console
- 6.6.18.1d: About assigning methods to method groups with the Java console
- 6.6.18.1e: About assigning permissions
- 6.6.18.3: Administering security with the Web console
- 6.6.18.5: Managing security IDs for the application server and administrative accounts
- 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
- 6.6.18.11: SecureWay Directory Version 2.1

Messages, logs, and traces

- 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
 - 6.6.19.1: Administering the product messages, logs, and traces
 - 6.6.19.1.1: Administering messages with the Java administrative console
 - Filtering messages with the Java administrative console
 - Collecting serious events with the Java administrative console
 - 6.6.19.1.2: Viewing logs and messages
 - Viewing messages with the Java administrative console
 - Viewing serious events with the Java administrative console
 - Viewing logs
 - Viewing traces
 - 6.6.19.3: Administering server traces with the Web console

Transactions

- 6.6.20: Administering transactions (overview)
 - 6.6.20.0: Transaction properties

Models and clones

- 6.6.28: Administering IBM Distributed Debugger and OLT
 - 6.6.28.1: Installing Debugger and OLT code and documentation
 - 6.6.28.1.1: Installing Debugger and OLT clients (README)
 - 6.6.28.1.2: Example: Installing OLT and Debugger clients on Windows NT
 - 6.6.28.2: Using Debugger and OLT
 - 6.6.28.3: Prerequisites and limitations of IBM Distributed Debugger and OLT

ORBs

- 6.6.30: Administering Object Request Brokers (ORBs)
 - 6.6.30.5: Setting the ORB timeout value

Naming and location

- 6.6.32: Administering name service support (overview)

JVMs

- 6.6.36: Administering Java virtual machines (JVMs)
 - 6.6.36.0: Java command line arguments reference
 - 6.6.36.5: Using the JDK conversion assistant to switch Java 1.2.2 vendor implementations

Administrative domains

Plug-ins for Web servers

- 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

Administrative servers

- 6.6.46: Administering WebSphere administrative servers
 - 6.6.46.0: Administrative server configuration file properties

Generic servers

[6.6.47: Administering generic servers](#)

[6.6.47.0: Properties of generic servers](#)

[6.6.47.1: Administering generic servers with the Java administrative console](#)

[6.6.47.1.1: Adding generic servers with the Java administrative console](#)

Various resource types

[6.6.48: Administering ports](#)

[6.6.49: Administering National Language Support](#)

[6.6.50: Administering coexisting product versions and editions](#)

[6.6.51: Administering network configurations](#)

Starting and stopping

[6.6a: Starting and stopping servers](#)

[6.6a01: Running the product servers and consoles as non-root](#)

Tutorials

[6.7: Tutorials](#)

[Tutorial](#)

Overview

[6a: Administrative overview of Version 3.5](#)

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. Install the application files.** Install the application into the test environment.
 - Who: Java programmer, systems administrator
 - How: [As described in "Installing files and setting classpaths"](#)
- 3. Configure the runtime, supporting resources, and the application itself.** Create application servers and other resources that will support applications.
 - Who: Systems administrator
 - How: [Using the administrative console](#)
- 4. 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 -- see the [tutorial](#) for examples
- 5. Configure runtime security.** Configure and enable security in the application server runtime.
 - Who: Systems administrator
 - How: [Using the administrative console](#)
- 6. Test application security.** Test the application again, this time with security enabled.
 - Who: Tester, systems administrator
 - How: [Using the administrative console](#)
- 7. 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
- 8. 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
9. **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](#)

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 the Advanced and Standard Editions 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.

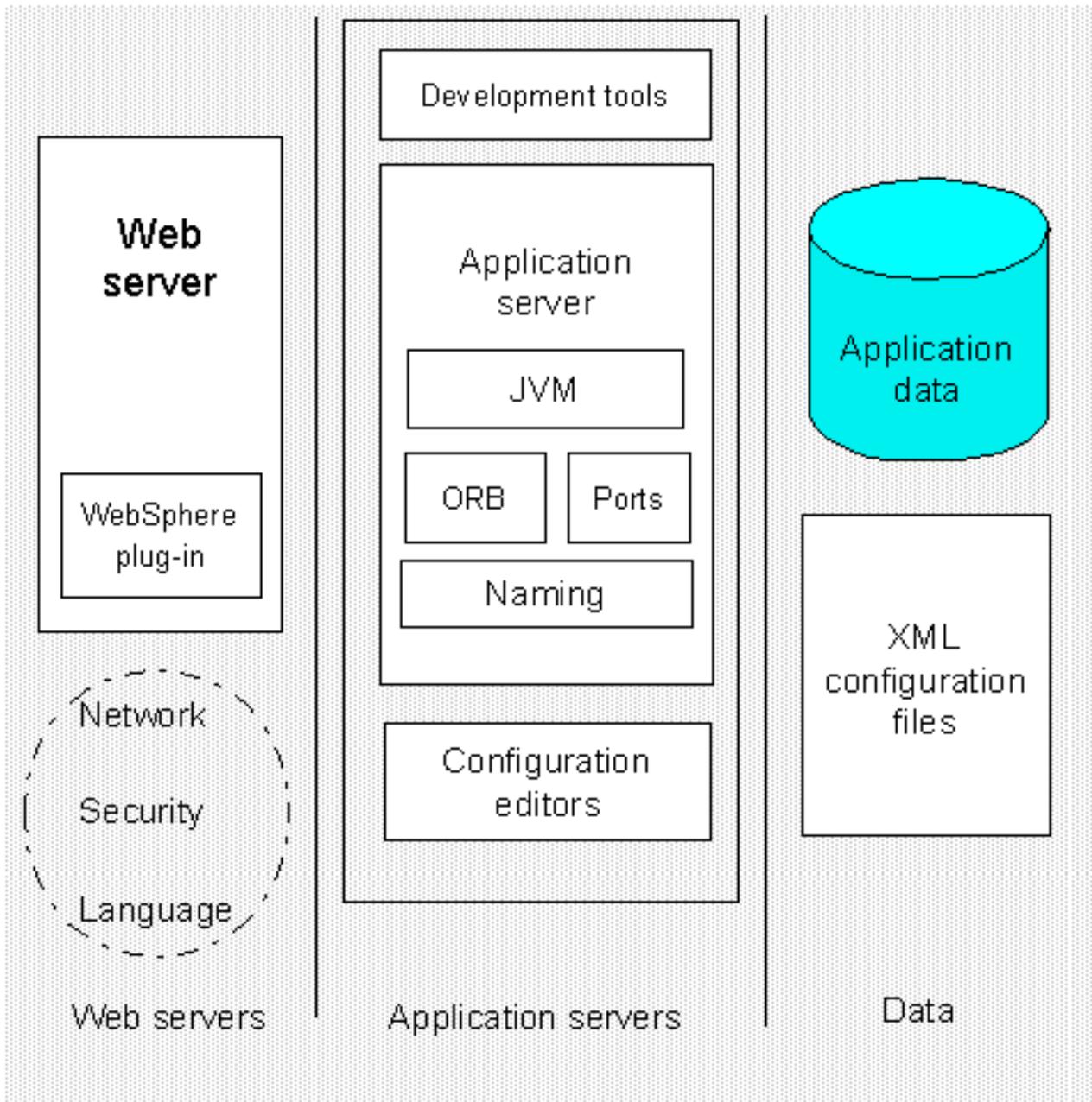
The administrative node contains an application server, plus several resources that transcend application servers (such as those related to JDBC support). The WebSphere plug-ins for the supported Web servers also require administration.

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

After the product is installed, the administrator needs to prepare the administrative resources and configurations necessary to support applications and their building blocks.

To configure the runtime and add the first application to it, do the following. Remember that your configuration must match your file placement as described in the article about [installing files and setting classpaths](#).

1. [Start the administrative server](#).
2. [Start the administrative console](#).
3. Verify that the resources for supporting an application exist. If this is not the first application, and the existing resources satisfy the administrator, the administrator can skip this step.

Use the [topology tree view in the console](#) to ensure that the following resources exist:

- A virtual host
- An application server
- A servlet engine (for servlets or JSP files)
- A Web application (for servlets or JSP files)

It is possible that the [default resources](#) are installed, in which case no further action is required to complete this step.

If you *do not* see the necessary resources, do one of two things:

- [Obtain the default resources](#) (the recommended approach if this is the first application)
- [Configure a new application server and its contents](#), and plan to use the [default virtual host](#)

It is also possible to [configure a new, custom virtual host](#).

4. The application support is in place. Now configure new administrative resources to represent the building blocks of the application:
 - For each JSP file, [configure a resource to represent the JSP file](#)
 - For each servlet, [review the options for configuring the servlet](#).

If you decide to explicitly configure the servlet, [follow the servlet configuration instructions](#). Otherwise, proceed to the next step.

5. Associate the servlets and JSP files into a logical unit by [configuring a Web application](#)
6. [Configure an enterprise application](#). It should contain the Web application -- perhaps multiple Web applications.
7. Verify that the following resources are running, allowing the application to be served:
 - The Web server
 - The application server containing the application
 - Any servlet engines supporting the application
 - The Web application
 - The application

Checklist for preparing runtime support for applications

Here is a summary for preparing the runtime environment for applications. Click each item in the table for instructions. To see these tasks as an end-to-end procedure applied to a sample application, perform the [application and configuration deployment tutorial](#).

To support ...	The administrative environment needs...
Anything	<ul style="list-style-type: none">✓ A virtual host✓ A node✓ An application server
Servlets, JSP files, and HTML	<ul style="list-style-type: none">✓ A servlet engine✓ A Web application
Data access	<ul style="list-style-type: none">✓ A JDBC driver✓ and data source
Personalization	Data access resources, plus: <ul style="list-style-type: none">✓ A session manager✓ A user profile manager

6.2.1: The default resources and configurations

IBM WebSphere Application Server provides default configurations, allowing administrators to get started quickly and easily:

- [Default administrative resources](#)

The product provides the option to install default versions of some resources that are necessary for getting started, such as application servers.

Without the default resources, more steps are required to set up a new administrative environment.

- [Default values for resource types](#)

Each resource has default values for some or all of its settings. The product provides original default values that are appropriate in many situations.

The administrator can change the original default values to customized default values, which will serve as a template each time the administrator configures a new resource.

- [The default and examples Web applications](#)

A default application (`default_app`) is provided for quickly migrating Web applications to Version 3.5 from Version 2.0x. Please note, the quick path is **not** the optimal path.

A second Web application named *examples* is provided to show the optimal way to configure Web applications and their components, and place Web application files in WebSphere directories, based on the Version 3.5 (and 3.0x) model.

The default administrative resources

The IBM WebSphere Application Server installation program provides an option to populate the administrative domain with some basic administrative resources, such as a default application server.

To support the first application in the WebSphere Application Server environment, the administrator must configure a few new resources. For example, an application containing servlets requires a Web application, servlet engine and application server for support.

With default administrative resources, the administrator does not need to configure the basic resources for supporting an application. Default resources even include a default application for reference.

Verifying that the default administrative resources exist

To verify whether the default resources are installed:

1. Start the WebSphere Administrative Console, the Java client. (The administrative server must also be running, prior to starting the client).
2. Display the Topology tree.
3. Locate resources in the tree, such as:
 - Default application server (Default Server)
 - Default servlet engine
 - Default application (Default App)

Note, the default virtual host exists in the Topology tree whether or not the default resources are installed.

Obtaining the default administrative resources

 The following actions cannot be reversed automatically. To get rid of the default resources, you must delete each default resource from the administrative domain. (Setting `install.initial.config` back to `false` will not remove the resources).

During installation:

To obtain the default resources while installing the product, take the Custom installation route. When offered various components to install, be sure to select the check box for installing a default configuration.

With product already installed:

If the product is already installed, edit the file `installation_root/bin/admin.config`. Set:

```
install.initial.config=true.
```

 For Advanced Edition environments involving application servers on multiple machines, install the default resources on only one machine in the cluster. You can clone the default resources to other machines in the cluster.

Default properties for resource types

The administrator should be aware that properties can have default values, custom default values and custom values:

- Each resource begins with the default values for its resource type
- The administrator can customize the default values of a resource type so that all new resources of that type begin with the "customized default" values
- The administrator can customize the values of an individual resource

The "default" and "examples" Web applications

The default Web application is designed to facilitate servlet and Web application migration from a Version 2 installation. The migration steps are described in the Related information below.

Things to note about the `default_app` include:

- The application Web path is `/`
- The application classpaths are:
`installation_root\hosts\default_host\default_app\servlets`
and
`installation_root\servlets`
- The application document root is the Web server document root.

Because the application Web path for the `default_app` is `/`, you can put the HTML resources for that application in the Web server document root.

- Servlets, such as `SnoopServlet`, have been added to the application. The servlet URIs (Web resources) are:

```
/servlet/servlet_name
```

where `servlet_name` is a user-defined name for the servlet.

For example, SnoopServlet has two servlet URIs:

- /servlet/snoop
- /servlet/snoop2
- The following WebSphere internal servlets (see Related information below) have been configured for the default_app:
 - InvokerServlet
 - PageCompileServlet
 - DefaultErrorReporter

6.4: Installing applications and setting classpaths

Configuring an application (using the administrative console and other interfaces) is closely related to the above tasks. The configuration must match the file placement. An administrator can either:

- Configure an application, then place the application files on the file system according to the application settings, which can include classpath information
- Place the files on the file system, then configure the application settings and the classpath to match the file locations

Refer to [article 6.4.1](#) for an understanding of the classpaths used by applications running in a WebSphere Application Server environment.

Refer to [article 6.4.2](#) for a description of typical and recommended locations for various file types.

6.4.1: Setting classpaths

The IBM WebSphere Application Server environment has these classpath components:

1. [The Application Server \(JVM\) Classpath](#)
2. [The Web Application Classpath](#)

You can also learn about the:

- [Administrative server classpath](#)
- [managedServerClasspath](#) property

Application server (JVM)classpath

Where to set	<p>There is one application server classpath for each applicationserver in the IBM WebSphere Application Server environment. Each applicationserver corresponds to a JVM.</p> <p>Set the classpath in one of two ways:</p> <ul style="list-style-type: none">● Use an administrative client to set the Command Line Arguments field ofthe application server. Specify the classpath with the flag -classpath.● Use an administrative client to set the application server Environment propertyto include a CLASSPATH variable and value.
Scope	This classpath is visible to all servlets and JSP files contained by an application server.
Behavior	The classes in this classpath are loaded by the JVM primordial ClassLoader. Hence, after the application server is started,any changes to this classpath will not take effect until the application server is stopped and started again.
Reloadable?	Classes loaded from this classpath will not be reloaded if theyare changed while the application server is running.
Typical contents	<ul style="list-style-type: none">● Classes referenced from servlets whose objects are added to sessions, suchobjects that are serialized and whose classes must not be reloaded.● Classes that call Java Native Interface (JNI) methods. Those classes andany imported classes must be placed in the application server classpathto prevent loading errors● For AS/400, any servlet classes and helper classes on which you are goingto run the AS/400 system debugger or IBM Distributed Debugger. <p>Note that classes which are put in this classpath should not referenceother classes that cannot be found in this classpath.</p>
ClassLoader	JVM primordial ClassLoader
Trace component	<p>To see the contents of this classpath, enable the trace component:</p> <pre>com.ibm.ejs.sm.active.ActiveEJBServerProcess=all=enabled</pre> <p>for the WebSphere administrative server.</p> <p>Note that this trace componentshould be enabled for the administrative server because that server is responsible for constructing the application server command linebefore starting the application server process.</p> <p>Enable the trace before starting the application server.</p>

Web application classpath

Where to set	Use an administrative client to specify the Classpath setting of a Web application.
Scope	The classpath is visible to all servlets and JSP files in the corresponding Web application.
Behavior	<p>This classpath is monitored and all components (JAR or class files) are reloaded whenever it is automatically detected that a component has been updated.</p> <p>A new JAR file is automatically loaded upon detection in any directory already contained in this classpath. This means that it is not necessary to explicitly specify a new JAR file in this classpath. Rather, it suffices to just put the JAR file in a directory that is already present in the classpath.</p> <p>Automatic reloading at the Web application level keeps all of the application components synchronized and conserves system resources compared to the Version 2.0x reloading scheme.</p> <p>Version 3.5 does not support remote servlet loading (that is, loading servlets across a network). All application components must be on the machine containing the application server hosting the application.</p>
Reloadable?	Yes
Typical contents	<ul style="list-style-type: none"> ● Directories or JAR files with Servlet classes. ● Directories or JAR files with helper classes that are not included in the servlet JAR file and that are expected to be reloadable. ● Directories or JAR files with Access Bean classes that are referenced from servlet classes.
ClassLoader	PowerClassLoader. There is one for each Web application.
Trace component	<p>Trace the Web application classpath by tracing the component <code>com.ibm.servlet.classloader.*=all=enabled</code> on a running application server.</p> <p>Enable the trace before invoking a servlet or JSP file in the Web application from a Web browser.</p>

 For improved performance, set the automatic reloading property to false in the properties of Web applications contained by production application servers.

Administrative server classpath

One last classpath you should be aware of is the WebSphere administrative server classpath, though it is recommended that you do **not** modify the classpath for the use of particular applications.

The administrative server classpath is set automatically when you install the product. The default classpath setting contains all of the IBM WebSphere Application Server APIs.

The administrative server classpath value depends on how you start the administrative server.

If you use one of these methods to start the administrative server:

- The `startupServer.sh` script on UNIX platforms
- The Windows NT Services panel

then the administrative classpath is set based on the value of the property:

```
com.ibm.ejs.sm.adminserver.classpath
```

in the file:

[product_installation_root/bin/admin.config](#)

If you use the `adminserver.[bat|sh]` script to start the administrative server, the classpath is set based on its value as specified by the `adminserver.[bat|sh]` script.

The managedServerClassPath

By default, the administrative server classpath is appended to each application server JVM classpath. If you would like to append a *different* classpath to each application server JVM classpath, define the different classpath in the property:

```
com.ibm.ejs.sm.adminServer.managedServerClassPath
```

in the file:

[product_installation_root/bin/admin.config](#)

The `managedServerClassPath` will be appended instead of the `administrativeserver` classpath.

6.4.2: Installing application files

This article describes how to place the files comprising an application onto the physical system containing the WebSphere Application Server product.

Enterprise applications containing various resources are deployed on application servers. For successful deployment, the WebSphere administrator must specify where application servers can find the files belonging to enterprise applications:

Java content

Use the application server classpath to specify its location. This includes JAR and class files for servlets

Static content

Use the Web application document root to specify its location. This includes JSP files, HTML files, and graphics

When you deploy servlets, Web applications, and Enterprise JavaBean applications, ensure that the component files are in the correct directories, typically relative to the IBM WebSphere Application Server [product_installation_root](#). Use this quick reference table as a guide.

File description	File Extension	Directory path
HTML documents and related static files	.html, .shtml, .jhtml, .gif, .au, and so on	These can be either served by the Web server, or placed in the Web application document root with the WebSphere file servlet enabled.
JavaServer Pages files	.jsp	Web application document root
Servlets that are to be reloaded	.class or .jar	Web application classpath If the servlets are in a package, mirror the package structure as subdirectories under the Web application classpath.
Servlet that are <i>not</i> to be reloaded	.class or .jar	Application server classpath
Servlet configuration file	.servlet	Directory that contains the servlet
JavaBean (not an enterprise bean) or other object to be reloaded	.ser or .jar	Web application classpath
JavaBean (not an enterprise bean) or other object <i>not</i> to be reloaded, such as serialized objects and servlets that use Java Native Interface methods	.ser or .jar	Application server classpath
Java objects added to a session	.class, .jar, or .ser	Application server classpath
Objects passed as arguments for remote calls		Application server classpath

Placing Web application files in product directories

Managing a Web application involves placing the Web application files in the WebSphere directories and configuring the Web application so that it can find the files.

Select one of two ways to accomplish the task:

- [Simple procedure: Default file placement](#)

- [Advanced procedure: Custom file placement](#)

Simple procedure: Default file placement

1. [Configure a new Web application](#). Accept the default values for its document root and classpath.
2. Change directory to `<was_installation_root>/hosts`.
3. Create a subdirectory with the logical name of the Web application, such as `myWebApplication`.

The logical name is the name by which the administrator specified, or will specify, to manage the Web application.

4. Change directory to the subdirectory created in the previous step.
5. Create subdirectories named "servlets" and "web."
6. Place the servlet class and JAR files in the servlets directory.
7. Place all other supporting files in the web directory:
 - HTML
 - XML
 - graphics

Advanced procedure: Custom file placement

Use this procedure instead if the Web application files must reside somewhere other than the default location.

1. Place the Web application files in one or more directories of choice.
2. When [configuring the Web application](#), ensure that:
 - The classpath contains all directories in which servlet class and JAR files reside
 - The document root contains all directories in which other supporting files reside, such as HTML, XML, and graphics

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).

Please see the sub-topics of this article.

6.5.2: Actions that require a restart

The table summarizes actions that require stopping a server or other process and starting it again. The actions are usually configuration changes to one or more administrative resources.

If you change ...	Stop this and start it again...
General, authentication mechanism, security server ID, or user registry settings in the Configure Global Settings security task	Any WebSphere administrative servers in the administrative domain to which the resource belongs
Properties in the Configure Application Security task	Any application servers on which the secured application runs
Properties in the Configure Resource Security task while the Web server is running	Any WebSphere administrative servers in the administrative domain to which the resource belongs
Servlet engine queue type or transport properties	Any application servers containing the servlet engine
Web application properties or contents	The Web application whose properties you changed (perform a "Restart Web App")
Servlet properties	Any Web application containing the servlet
Database-related properties for a given Session Manager	The Session Manager Specifically: <ul style="list-style-type: none"> ● Changes on the Tuning and Persistence tabbed pages take effect the next time the application server containing the Session Manager is started. ● Changes on the Persistence and Interval tabbed pages take effect immediately. ● Changes on the Enable tab vary. See the individual property descriptions.
User profile properties	Any application servers containing the user profile
Virtual host properties	Any application servers that contain the resources associated with the virtual host, if they were running when the virtual host properties were changed. See below for an example.

Example of virtual host changes requiring a restart

Suppose the administrator decides to add an alias *newalias* to a virtual host that has three servlets. Two of the servlets are running on *application_server_1* and the other servlet is running on *application_server_2*.

The administrator adds the new alias to the virtual host properties while *application_server_1* and *application_server_2* are running. Users will not be able to access the servlets by the URL http://newalias/servlet_name until the administrator stops the two application servers and starts them again.

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

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.

Configuring the overall administrative environment

Description	Administrative overview	Settings
Nodes	Administering nodes	
Administrative servers	Administering WebSphere administrative servers	Administrative server properties
Web resources	Administering Web resources	Web resource properties
Virtual hosts	Administering virtual hosts	Virtual host properties

Configuring specific application servers

Description	Administrative overview	Settings
Application servers	Administering application servers	Application server properties
Servlet engines	Administering servlet engines	Servlet engine properties

Configuring application components installed in a runtime

Description	Administrative overview	Settings
Applications	Administering applications	
Web applications	Administering Web applications	Web application properties
Servlets	Administering servlets	Servlet properties
JSP files	Administering JSP files	

Configuring resource providers for data access and other functions

Description	Administrative overview	Settings
Data access	Administering database connections	JDBC driver 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	No console settings
User Profile Managers	Administering User Profile Managers	
Tracing	Administering the product trace	Trace properties
Transactions	Administering transactions	Transaction properties
Generic servers	Administering generic servers	Generic server properties
OLT and Distributed Debugger	Administering OLT and Distributed Debugger	

Configuring resources related to routing and distributing application requests

Description	Administrative overview	Settings
Plug-ins for Web servers	Administering WebSphere plug-ins for Web servers	Plug-in properties

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 \(consoles and tools\)](#)
- [Comparison of administrative consoles](#)
- [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
 Administrative console (Java based)	Graphical interface for configuring and operating application servers and other resources that support applications in a runtime environment
 WSCP	Command line interface for operational tasks, such as starting and stopping application servers
 XMLConfig	Command line interface for configuration tasks, including the import and export of human readable XML files describing resource configurations
 Administrative console (Web based)	Graphical interface for configuring and operating application servers and other resources that support applications in a runtime environment
Configuration using property files	Ability to manually edit various property files in the product directory structure for miscellaneous configuration purposes



Comparison of administrative consoles

The following table summarizes the traits or features of the administrative interfaces that are alternative options to one another.

Trait	Java console?	Web console?	WSCP?	XMLConfig?
Graphical interface (requires human presence for use)	✓	✓		
Command line (can be used programmatically)			✓	✓
Lets you save configuration changes and apply them later		✓	✓	✓
Produces or allows work with properties files that can be read and edited by hand	✓	✓	✓	✓
Full administrative functionality	✓			✓
Can be used remotely with respect to administrativeserver machine	✓	✓		
Supports variable substitution			✓	

Scope of administrative capabilities

Administrators that have access to an administrative client canuse the full capabilities of the client. At this time, it is **not**possible to allow administrators to perform some operations (such as starting andstopping 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.

6.6.0.1: Using the Java administrative console

The WebSphere Administrative Console is a graphical, Java-based administrative client to the IBM WebSphere Application Server administrative server. This administrative console supports the full range of product administrative activities.

The console features Topology and Type tree views for surveying and manipulating resources in the administrative domain. In addition, task wizards are provided to lead administrators through the creation and protection of new resources, such as servlets, applications and application servers.

The console has three main areas: the navigation area, the content or work area and the message area.

The navigation area

The navigation area on the left side of the console controls the content displayed on the right side of the console.

The navigation area has two tree views, which the administrator can toggle between:

- The Topology tree for surveying and managing existing components
- The Type tree for specifying defaults and taking inventory

The content or work area

The content or work area on the right of the console displays information based on the selected items in the navigation pane.

- When the administrator clicks a folder in the Type tree, the content area displays a table listing information for all existing instances of that type.
- When the administrator clicks an item in the Topology tree, the content area displays the properties for the item.

The messages area

The Console Messages window at the bottom of the console provides a high-level view of important events, such as successful completions and fatal errors.

The administrator can resize the window to see as many messages at a time as desired.

6.6.0.1.1: Starting and stopping Java administrative consoles

Preconditions

The WebSphere administrative server must be running before you start the Java administrative console. By default, the server does not start automatically when you start the machine.

The Java administrative console (also known as the WebSphere Administrative Console) is a Java process. As such, it must be able to locate [a JDK supported by IBM WebSphere Application Server, as specified on the prerequisites Web page](#).

If you are operating the Java console on a machine that does not contain the WebSphere runtime and its JDK, you might need to set JAVA_HOME or other environment variables used to help Java processes find the JDK.

Also, ensure that the Java console and WebSphere administrative server with which it is communicating are using the same supported JDK (and ORB) level.

Instructions for AIX, HP-UX, Linux, and Solaris

1. Start the administrative server, or ensure it is already running.
2. Start the administrative console:
 - a. Change directory to the bin directory under the product installation root.
`cd product_installation_root/bin`
 - b. Ensure that the DISPLAY variable is set in the shell from which the administrative client script will be run:
`$ export DISPLAY=localhostName:0.0`
 - c. Run the administrative client script:
`./adminclient.sh`

To stop the console, use the Console -> Exit option on the console menu bar.

Instructions for Windows NT and 2000

From the Start menu, select Programs -> IBM WebSphere -> Application Server 3.5 -> Administrator's Console.

To stop the console, use the Console -> Exit option on the console menu bar.

6.6.0.1.1.1: Configuring new resources from the Topology tree

1. In the tree, locate the parent resource for which you would like to configure a new child resource.
2. Right-click the parent resource.
3. From the resulting menu, select Create -> *Child*, where *Child* is the type of child resource you want to configure. The properties dialog is displayed for naming and configuring the new child resource.
4. Complete the properties dialog and click OK.

Example:

1. To configure a new servlet to reside as part of the default Web application (default_app), locate default_app in the Topology tree.
2. Right-click the default_app.
3. Click Create -> Servlet on the right-click menu of the default_app.
4. Complete the servlet properties dialog and click OK.

6.6.0.1.1.2: Configuring new resources from the Type tree

1. In the tree, locate the type of resource to configure.
2. Right-click the resource type.
3. From the resulting menu, select Create. The properties dialog is displayed for configuring the new resource.
4. Complete the properties dialog and click OK.

Example:

1. To configure a new servlet, locate Servlets in the Type tree.
2. Right-click Servlets.
3. On the resulting menu, select Create to display a properties dialog for configuring the new servlet.
4. Specify the servlet properties and click OK.

6.6.0.1.3: Configuring resources with the Java administrative console

The administrator can use task wizards to configure new instances of most resource types. Alternatively, the administrator can use the Topology or Type tree views to introduce new resources into the administrative domain.

Using the right-click menu of a parent resource in the Topology tree, the administrator can configure a new child resource that will reside on the parent resource.

Using the right-click menu of a resource type on the Type tab, the administrator can configure a new instance of that resource type.

The procedure for configuring a new resource from the tree views is basically the same for each resource type. The Related information links to instructions.

6.6.0.3.1.1: Configuring new resources from the Topology tree

1. In the tree, locate the parent resource for which you would like to configure a new child resource.
2. Right-click the parent resource.
3. From the resulting menu, select Create -> *Child*, where *Child* is the type of child resource you want to configure. The properties dialog is displayed for naming and configuring the new child resource.
4. Complete the properties dialog and click OK.

Example:

1. To configure a new servlet to reside as part of the default Web application (default_app), locate default_app in the Topology tree.
2. Right-click the default_app.
3. Click Create -> Servlet on the right-click menu of the default_app.
4. Complete the servlet properties dialog and click OK.

6.6.0.3.1.2: Configuring new resources from the Type tree

1. In the tree, locate the type of resource to configure.
2. Right-click the resource type.
3. From the resulting menu, select Create. The properties dialog is displayed for configuring the new resource.
4. Complete the properties dialog and click OK.

Example:

1. To configure a new servlet, locate Servlets in the Type tree.
2. Right-click Servlets.
3. On the resulting menu, select Create to display a properties dialog for configuring the new servlet.
4. Specify the servlet properties and click OK.

6.6.0.1.4: Actions on the right-click menus in the Java administrative console

The table summarizes choices available on the right-click menus of resources in the Topology or Type trees. To view the right-click menu of a resource, locate and right-click the resource in the tree.

Some actions are not available for certain resources, or can become temporarily unavailable due to the state of a resource. (For example, if a server is already stopped, its Stop action will be unavailable).

Action	Result
Create	<p>Creates a new instance, child, or model of the resource.</p> <ul style="list-style-type: none"> For resources in the Type tree, the action displays a properties dialog for creating a new instance of the resource. For resources in the Topology tree, the action is part of a menu sequence for creating either children or models of the resource. <p>For example, the right-click menu of a servlet engine in the Topology tree contains options for creating a Web application, User Profile or Session Manager, or model of the servlet engine.</p>
Default Properties	<p>Displays a properties dialog for specifying the default properties of the resource type.</p> <p>For example, to specify a set of properties that each new servlet engine should have after its initial creation, select Default Properties from the right-click menu of the Servlet Engines resource in the Type tree.</p>
Enabled	<p>Specify whether to make the resource available to users, or temporarily remove it from service.</p>
Force Stop	<p>Stops the resource when the regular Stop action is ineffective. Use the option when the administrative server or other resource is in a strange state and cannot be stopped using the ordinary Stop action.</p>
Performance	<p>Displays the EPM dialog for specifying which components of the resource to monitor for performance data. See the related information to learn more about performance monitoring.</p>
Ping	<p>Pings the resource to see whether the resource is responding. The administrator can use the ping action as a basic check of the "health" of the resource.</p> <p>For information about the ping operation, consult the documentation for the operating system in use.</p>
Properties	<p>Displays a properties dialog for configuring the resource.</p>
Remove	<p>Permanently removes the resource and its child resources from the administrative domain.</p>
Restart	<p>Stops a resource, then returns the resource to the state it was in before it (or its parent resource) was stopped.</p> <p>For example, suppose you performed Stop for Restart (or just Restart) on a running administrative server node containing application server "A" in the stopped state and application server "B" in the running state. The Restart command would cause application server A to remain stopped. It would return application server B to the running state, along with the administrative server.</p> <p>(For some resources, this action is listed as Restart <i>resource type</i>).</p>
Start	<p>Changes the resource to running state.</p>

Stop	<p>Stops the resource.</p> <p>For an administrative server node, this action stops the administrative server running on the node. It does not shut down the operating system.</p>
Stop for Restart	<p>Stops a resource and any resources contained by the resource, in preparation for the Restart command.</p> <p>It can be useful for stopping and restarting an administrative server that contains multiple application servers, some of which are running and others of which are stopped. Using a combination of Stop for Restart and then Restart will return the application servers to their former states (stopped or running) when the administrative server is restarted.</p>
Transactions	<p>Displays the transactions dialog for viewing and controlling transactions occurring on the resource. See the Related information to learn more about administering transactions.</p>
Trace	<p>Displays the trace administration dialog. See the Related information below to learn more about tracing.</p>

6.6.0.1.6: Find the best place to start in the Java console

Chances are, you would like to start performing administrative tasks right away. This table suggests the best place to start when you have an administrative task.

If you want to...	Start here...
Configure an application server	On the console menu bar, click Console -> Task -> Configure an application server.
Support Java components that access databases	On the console menu bar, click Console -> Task -> Create data source.
Introduce new resources, such as servlets, into the administrative domain	<p>To configure new servlets, Web pages and JSP files, use the console menu bar selections:</p> <ul style="list-style-type: none"> ● Console -> Tasks -> Add a servlet. ● Console -> Tasks -> Add a JSP file or Web resource. <p>To configure new enterprise beans to use an existing application server and container (such as the default server and container):</p> <ol style="list-style-type: none"> 1. On the console menu bar, click View -> Topology. 2. Locate and right-click the container to use. 3. Click Create -> Enterprise Beans. <p>To configure new enterprise beans when you do not yet have an application server or container, click Console -> Tasks -> Configure an application server.</p>
Combine servlets, JSP files, or Web pages into a "Web" application	On the console menu bar, click Console -> Tasks -> Configure a Web application.
Combine enterprise beans, Web applications, JSP files or Web pages into an enterprise application	On the console menu bar, click Console -> Tasks -> Create enterprise application.
Alter the contents of an application	On the console menu bar, click -> Edit enterprise application.
Fine-tune the configurations of resources	<ol style="list-style-type: none"> 1. On the console menu bar, click View -> Topology. 2. Locate and click the resource to display and edit its properties in the right side of the console.
Start, stop and ping resources	<ol style="list-style-type: none"> 1. On the console menu bar, click View -> Topology. 2. Right-click a resource for a menu of actions you can perform on it, such as Start, Stop and Ping.
Specify default configurations for new resources to inherit	<ol style="list-style-type: none"> 1. On the console menu bar, click View -> Type 2. Locate and click a resource type to configure default properties for the resource type.

Add or remove resources	<ol style="list-style-type: none">1. On the console menu bar, click View -> Topology.2. Locate and right-click the resource.3. Click Remove.
Set up security	Click Console -> Tasks. The security tasks include the Configure Global Settings task and the tasks below it.

6.6.0.1.7: The Topology and Type trees

Using the console View menu, the administrator can toggle the Java console navigation area between two tree views for managing resources.

The **Topology tree** displays all managed nodes in the administrative domain. For each node, this tree displays a hierarchy of existing resources associated with the node. The administrator can:

Action	Result
Click an instance	Display its properties (settings) on the right side of the console
Right-click an instance	Display a pop-up menu for performing common actions, such as starting, stopping, pinging, and deleting the resource

The **Type tree** displays a hierarchical or tree view of the potential resources that can exist on a node in the administrative domain. Each resource type, such as a Servlet or an Application Server, is represented by a folder icon. The Type tree shows the administrator:

Information	Details
Configurable kinds of items in the administrative domain, and how many of each kind already exist	For example, the tab contains folders labeled "Application Servers". When the administrator clicks the Application Server folder, a table listing information about the existing application servers is displayed on the right side of the console.
The default properties for each type of resource	The administrator can use the Type tree to configure default properties for applications, servlets and other resource types. When resources are created, these instances will inherit the default properties.
The relationships among resource types in the administrative domain	The resource types are displayed according to their hierarchical, parent-child relationships.

6.6.0.1.8: Starting, stopping and pinging resources with the Java administrative console

Each resource in the Topology tree has a set of valid actions the administrator can perform, such as starting, stopping or pinging the resource.

1. To view the set of valid actions, locate and right click the resource in the Topology tree to display its menu.
2. To perform an action, select the action from the menu.

If you are unsure of what an action does, see [Related information](#) for references to available actions.

6.6.0.1.9: Removing resources with the Java administrative console

To remove a resource:

1. Click View -> Topology on the console menu bar.
2. Locate the resource in the Topology tree view.
3. Right click the resource to display a menu.
4. Click Remove.

If a resource does not have a Remove menu option, it is not valid for removal.

6.6.0.1.10: Editing resource properties

To edit the properties of a resource in the administrative domain:

1. Locate the resource in the Topology tree.
2. Click the resource to display its properties on the right side of the console.
3. Modify the properties and click the appropriate button on the property sheet (such as OK or Finish) to save the property changes.

6.6.0.1.11: Buttons in the Java administrative console

- **Abort.** Aborts a transaction that is not yet in the prepared state. All operations that the transaction completed are undone.
- **Add.** Adds the selected or typed item to a list, or produces a dialog for adding an item to a list.
- **Apply.** Saves your changes to tabbed pages in a dialog box, without exiting the dialog box.
- **Back.** Displays the previous page or item in a sequence.
- **Browse.** Lets you look for a file on your system.
- **Cancel.** Exits the current dialog box, discarding unsaved changes.
- **Change.** Lets you search the user registry for an application user ID to run under in the context of security.

Lets you change the container data source in the context of container properties.

- **Change JAR.** Lets you browse for a JAR file.
- **Clear.** Clears your changes to tabbed pages in a dialog box and restores the most recently saved values.
- **Clear Selections.** Clears selected cells in the tables on this tabbed page.
- **Close.** Exits the dialog box.
- **Commit.** Releases all locks held by a prepared transaction and forces the transaction to commit.
- **Create.** Saves your changes to tabbed pages in a dialog box and exits the dialog box.
- **Details.** Shows transaction details.
- **Done.** Saves your changes to tabbed pages in a dialog box and exits the dialog box.
- **Down.** Moves downward through a list.
- **Dump.** Activates a traced application server dump.
- **Edit.** Lets you edit the selected item in a list, or produces a dialog box for editing the item.
- **Filter.** Produces a dialog box for specifying the resources to view in the tables on this tabbed page.
- **Finish.** Forces a transaction to finish, regardless of whether its outcome has been reported to all participating applications.
- **First.** Displays the first record in a series of records.
- **Last.** Displays the last record in a series of records.
- **Hide Details.** Changes the display to show only the most important data.
- **Next.** Displays the next page or item in a sequence.
- **OK.** Saves your changes to tabbed pages in a dialog box and exits the dialog box.
- **Quit.** Exits a dialog box, discarding any unsaved changes.
- **Refresh.** Refreshes the view of data for instances currently listed on this tabbed page.
- **Regen Plugin Config.** Forces a regeneration of the Web server plug-in configuration on every node (physical machine) in the administrative domain.
- **Remove.** Deletes the selected item.
- **Reset.** Clears your changes to tabbed pages in a dialog box and restores the most recently saved values.
- **Retrieve new.** Retrieves a new record.
- **Save.** Saves your changes and exits the current dialog box.
- **Select.** Lets you select a scope in which to monitor resources for resource analysis.
- **Set.** Saves your changes to settings in a dialog box.

- **Settings.** Produces a dialog box for editing servlet-related resource settings.
- **Settings in use.** Produces a dialog box showing the current settings.
- **Start.** Starts collecting data for the tables on this tabbed page.
- **Stop.** Stops collecting data for the tables on this tabbed page.
- **Up.** Moves upward through a list.
- **Update Resource List.** Updates the data on a table. Discovers and adds new instances to the table.

6.6.0.1.12: Property recommendations by operating system

This file lists recommended property settings with respect to specific operating systems.

Unless noted, property settings are treated uniformly across all operating systems.

For the properties listed in this file, only the operating system-specific information is provided. For a general description of a property, see the help file that describes the resource possessing the property.

The properties currently listed in this file apply to server resources, such as application servers and generic servers (managed processes).

To learn the operating systems to which a particular version of the WebSphere Application Server product was ported, consult the documentation accompanying that version. The mention of an operating system in this file does not imply or guarantee that the current product version was or will be ported to that operating system.

Property behavior on UNIX systems

Process priority

Specifies the operating system process priority under which to run the server. The lower the number, the greater the importance of the process.

Stderr

If this attribute is set to the null string(""), the stream is set to the null device (/dev/null).

Stdin

If this attribute is set to the null string(""), the stream is set to the null device (/dev/null).

Stdout

If this attribute is set to the null string(""), the stream is set to the null device (/dev/null).

Property behavior on Windows NT

Group ID

This attribute is ignored.

Process priority

This attribute is ignored.

Stderr

If this attribute is set to the null string(""), the stream is set to the null device (nul).

Stdin

If this attribute is set to the null string(""), the stream is set to the null device (nul).

Stdout

If this attribute is set to the null string(""), the stream is set to the null device (nul).

User ID

This attribute is ignored.

Property behavior on AS/400

Group ID

This attribute is ignored.

Process priority

This attribute is ignored.

Stderr

If this attribute is set to the null string(""), the stream is set to the null device (QPRINT).

Stdin

If this attribute is set to the null string(""), the stream is unable to accept input.

Stdout

If this attribute is set to the null string(""), the stream is set to the null device (QPRINT).

Umask

This attribute is ignored.

6.6.0.1.13: 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" 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	-mx	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	-ms	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	-nojit	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	<i>true/false</i>	If true, the message is printed
Prevent asynchronous garbage collection	-noasyncgc	None	None
Disable class garbage collection	-noclassgc	None	None
Specify whether to print a message each time the JVM loads a class	-verbose	<i>true/false</i>	If true, the message is printed

6.6.0.1.14: Actions on the right-click menus of resources

The table summarizes choices available on the right-click menus of resources in the Topology or Type trees. To view the right-click menu of a resource, locate and right-click the resource in the tree.

Some actions are not available for certain resources, or can become temporarily unavailable due to the state of a resource. (For example, if a server is already stopped, its Stop action will be unavailable).

Action	Result
Create	<p>Creates a new instance, child, or model of the resource.</p> <ul style="list-style-type: none"> For resources in the Type tree, the action displays a properties dialog for creating a new instance of the resource. For resources in the Topology tree, the action is part of a menu sequence for creating either children or models of the resource. <p>For example, the right-click menu of a servlet engine in the Topology tree contains options for creating a Web application, User Profile or Session Manager, or model of the servlet engine.</p>
Default Properties	<p>Displays a properties dialog for specifying the default properties of the resource type.</p> <p>For example, to specify a set of properties that each new servlet engine should have after its initial creation, select Default Properties from the right-click menu of the Servlet Engines resource in the Type tree.</p>
Enabled	Specify whether to make the resource available to users, or temporarily remove it from service.
Force Stop	Stops the resource when the regular Stop action is ineffective. Use the option when the administrative server or other resource is in a strange state and cannot be stopped using the ordinary Stop action.
Performance	Displays the EPM dialog for specifying which components of the resource to monitor for performance data. See the Related information to learn more about performance monitoring.
Ping	<p>Pings the resource to see whether the resource is responding. The administrator can use the ping action as a basic check of the "health" of the resource.</p> <p>For information about the ping operation, consult the documentation for the operating system in use.</p>
Properties	Displays a properties dialog for configuring the resource.
Remove	Permanently removes the resource and its child resources from the administrative domain.
Restart	<p>Stops a resource, then returns the resource to the state it was in before it (or its parent resource) was stopped.</p> <p>For example, suppose you performed Stop for Restart (or just Restart) on a running administrative server node containing application server "A" in the stopped state and application server "B" in the running state. The Restart command would cause application server A to remain stopped. It would return application server B to the running state, along with the administrative server.</p> <p>(For some resources, this action is listed as Restart <i>resource type</i>).</p>
Start	Changes the resource to running state.

Stop	<p>Stops the resource.</p> <p>For an administrative server node, this action stops the administrative server running on the node. It does not shut down the operating system.</p>
Stop for Restart	<p>Stops a resource and any resources contained by the resource, in preparation for the Restart command.</p> <p>It can be useful for stopping and restarting an administrative server that contains multiple application servers, some of which are running and others of which are stopped. Using a combination of Stop for Restart and then Restart will return the application servers to their former states (stopped or running) when the administrative server is restarted.</p>
Transactions	<p>Displays the transactions dialog for viewing and controlling transactions occurring on the resource. See the Related information to learn more about administering transactions.</p>
Trace	<p>Displays the trace administration dialog. See the Related information below to learn more about tracing.</p>

6.6.0.1a: Starting and stopping Java administrative consoles

Preconditions

The WebSphere administrative server must be running before you start the Java administrative console. By default, the server does not start automatically when you start the machine.

The Java administrative console (also known as the WebSphere Administrative Console) is a Java process. As such, it must be able to locate [a JDK supported by IBM WebSphere Application Server, as specified on the prerequisites Web page](#).

When you install the Java console, the JDK is **not** installed with it. If operating the Java console on a machine that does not contain the WebSphere runtime and its JDK, you might need to set JAVA_HOME or other environment variables used to help Java processes find the JDK.

Also, ensure that the Java console and WebSphere administrative server with which it is communicating are using the same supported JDK (and ORB) level.

Instructions for AIX, HP-UX, and Solaris

1. Start the administrative server, or ensure it is already running.
2. Start the administrative console:
 1. Change directory to the bin directory under the product installation root.
`cd product_installation_root/bin`
 2. Run the administrative client script:
`./adminclient.sh`

To stop the console, use the Console -> Exit option on the console menu bar.

Instructions for Windows NT

From the Start menu, select Programs -> IBM WebSphere -> Application Server 3.5 4.0 -> Administrator's Console.

To stop the console, use the Console -> Exit option on the console menu bar.

6.6.0.2: Command line administration

The following command line administrative tools are available for interacting with the WebSphere administrative server.

- Use **wscp** for operational and configuration tasks such as starting, stopping, and configuring application servers and other object types.
- Use **XMLConfig** for configuration tasks. This tool provides a way to work with the administrative domain as represented in WebSphere XML.

6.6.0.2.1: XMLConfig command line interface for XML configuration

Use the *XMLConfig* tool to import and export configuration data to and from the WebSphere Application Server administration repository. This XML-based approach complements the administration you can perform through the WebSphere administrative console.

You may use this tool to perform multiple changes to the WebSphere Application Server administration repository at a single time without having to go through the repetitive routines on the Administration Console. You may also use this tool to extract the repository information from one server and import it onto a cloned server.

The XMLConfig tool provides three fundamental features:

- **full export**

Generates an XML document describing the configuration of the entire administrative domain. In effect, the full export takes a "snapshot" of the administrative repository contents. Click [here](#) for an example of a full export, including a discussion of the various parts of the resulting XML file.

- **partial export**

Provides an XML document specifying administrative objects to export from the administrative domain. The objects and their children are exported to an XML document that you specify. Click [here](#) for an example of a partial export.

- **import**

Imports a newly created XML document or a document you previously exported and modified. In the XML document, you can add, modify, or remove administrative objects such as servlets and virtual hosts. Your XML document can replace the administrative repository contents partially or entirely.

6.6.0.2.1.1: XMLConfig - Command syntax

This section describes the command line syntax for the XMLConfig tool.

Because setting the classpath appropriately is vital to the tool's success, WebSphere Application Server Version 3.5 contains an XMLConfig.bat (Windows NT) or XMLConfig.sh (*IX) file for starting the tool. The file is located in the bin directory of the product installation root, uses the com.ibm.websphere.xmlconfig.XMLConfig java class, and has the following command line syntax:

```
{ ( -import <xml data file> ) || [ ( -export <xml output file> [-partial <xml data file>] ) ] } -adminNodeName <primary node name> [ -nameServiceHost <host name> [ -nameServicePort <port number> ] ] [-traceString <trace spec> [-traceFile <file name>]] [-substitute <"key1=value1[;key2=value2;[...]]">]] In input xml file, the key(s) should appear as $key$ for substitution.
```

The arguments include:

-adminNodeName

Required argument specifies the node containing the administrative server to which you are connecting. The argument value must match the node name given in the topology tree on the Topology tab of the WebSphere Administrative Console.

-import || -export || -export -partial

Required argument specifies the operation to perform -- an import or export. Unless you also specify the parameter -partial, the export will be treated as a full export.

-nameServiceHost, -nameServicePort

Optional arguments specify the hostname of the machine containing the name service, and the port by which to communicate with the name service. The default value of -nameServicePort is 900.

-traceString

Optional argument specifies the WebSphere Application Server internal code to trace.

For more information, see the [traceString section of the trace help](#).

-substitute

Optional argument specifies the variables to be substituted. For example:

```
-substitute "NODE_NAME=admin_node;APP_SERVER=default_server"
```

This argument substitutes any occurrence of \$NODE_NAME\$ with admin_node and any occurrence of \$APP_SERVER\$ with default_server that is found in the input XML file.

If the substitution string contains semicolons, use \$semiColon\$ to separate it from the ";" delimiter. On UNIX platforms, be sure to add an escape character to each dollar sign (\$) within the substitution string (for example, \\$semiColon\\$).

The following examples demonstrate correct syntax. 'Node1' is the name by which the node containing the administrative server is administered.

Import operation:

```
XMLConfig -adminNodeName Node1 -import import.xml
```

Full export operation:

```
XMLConfig -adminNodeName Node1 -export export.xml
```

Partial export operation:

```
XMLConfig -adminNodeName Node1 -export export.xml -partial input.xml
```

6.6.0.2.1.1.1: XMLConfig - Example of a full export

The following example export shows the XML elements for each object type in the WebSphere administrative domain. It is a full export of the administrative repository of a Standard Edition installation featuring the default administrative configuration.

To produce a similar export, you would issue the command:

```
XMLConfig -adminNodeName buccaneer -export export.xml
```

where the hostname of the machine containing the administrative server is "buccaneer" and the output will be directed to a file named export.xml.

i When you perform an export, the XML output file will not contain blank lines or gaps. In contrast, the following export has been broken into segments, each of which is briefly discussed.

Also, this example was obtained from a system that used the default configuration, and might vary slightly from actual output due to changes on your particular system. It is recommended you try the export command yourself to see exactly the output that is produced.

```
<?xml version="1.0"?><!DOCTYPE websphere-sa-config SYSTEM
"$server_root$$dsep$bin$dsep$xmlconfig.dtd" >
<websphere-sa-config>
```

The above tags mark the beginning of the export. The following part of the export contains a tag for the default **virtual host**, `default_host`, in the administrative domain.

The default host recognizes several MIME types, which are listed as part of a MIME table in the virtual-host tag. In fact, there are so many MIME types that some are omitted from the example code below.

The MIME types are followed by the URIs (also known as **Web resources**) hosted by `default_host`:

- [Skip ahead](#)
- [View object syntax](#)

```
<virtual-host name="default_host" action="update"> <mime-table> <mime type="audio/x-wav">
<ext>wav</ext> </mime> <mime type="application/x-sv4cpio"> <ext>sv4cpio</ext>
</mime> <mime type="text/x-ssi-html"> <ext>htmls</ext> <ext>shtml</ext>
</mime> ... <mime type="application/x-netcdf"> <ext>nc</ext>
<ext>cdf</ext> </mime> <mime type="video/x-motion-jpeg"> <ext>mjpg</ext>
</mime> </mime-table> <alias-list> <alias>localhost</alias> <alias>127.0.0.1</alias>
<alias>buccaneer.raleigh.ibm.com</alias> <alias>buccaneer</alias>
<alias>9.67.127.58</alias> </alias-list> <uri name="/" rootURI="/" action="create"/> <uri
name="/servlet/snoop" rootURI="/" action="create"/> <uri name="/servlet/snoop2" rootURI="/"
action="create"/> <uri name="/servlet/hello" rootURI="/" action="create"/> <uri
name="/ErrorReporter" rootURI="/" action="create"/> <uri name="/servlet" rootURI="/"
action="create"/> <uri name="/*.jsp" rootURI="/" action="create"/> <uri name="/*.jsw"
rootURI="/" action="create"/> <uri name="/*.jsw" rootURI="/" action="create"/> <uri
name="/admin" rootURI="/admin" action="create"/> <uri name="/admin/install" rootURI="/admin"
action="create"/> <uri name="/admin/*.jsp" rootURI="/admin" action="create"/> <uri
name="/admin/*.jsw" rootURI="/admin" action="create"/> <uri name="/admin/*.jsw" rootURI="/admin"
action="create"/> <uri name="/admin/" rootURI="/admin" action="create"/> <uri
name="/admin/servlet" rootURI="/admin" action="create"/> <uri name="/admin/ErrorReporter"
rootURI="/admin" action="create"/> <uri name="/webapp/examples" rootURI="/webapp/examples"
action="create"/> <uri name="/webapp/examples/simpleJSP.servlet" rootURI="/webapp/examples"
action="create"/> <uri name="/webapp/examples/simpleJSP" rootURI="/webapp/examples"
action="create"/> <uri name="/webapp/examples/Servlet" rootURI="/webapp/examples"
action="create"/> <uri name="/webapp/examples/ping" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/SourceCodeViewer" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/showCfg" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/ShowCfg" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/showConfig" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/ShowConfig" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/HitCount" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/verify" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/*.jsp" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/*.jsw" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/*.jsw" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/HelloPervasive" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/StockQuote" rootURI="/webapp/examples" action="create"/> <uri
name="/webapp/examples/BeenThere" rootURI="/webapp/examples" action="create"/> </virtual-host>
```

The next section contains the database driver and data source information :

- [Skip ahead](#)
- [View object syntax](#)

```

<jdbc-driver name="Admin DB Driver" action="update">
<implementation-class>jdbc.idbDriver</implementation-class>    <url-prefix>jdbc:idb</url-prefix>
<jta-enabled>>false</jta-enabled>    <install-info>    <node-name>buccaneer</node-name>
<jdbc-zipfile-location>/usr/WebSphere/AppServer/lib/idb.jar</jdbc-zipfile-location>
</install-info> </jdbc-driver> <jdbc-driver name="AccountDB2Driver" action="update">
<implementation-class>com.ibm.db2.jdbc.app.DB2Driver</implementation-class>
<url-prefix>jdbc:db2</url-prefix>    <jta-enabled>>false</jta-enabled>    <install-info>
<node-name>buccaneer</node-name>
<jdbc-zipfile-location>/home/db2as/sqlllib/java/db2java.zip</jdbc-zipfile-location>
</install-info> </jdbc-driver> <data-source name="Sample Datasource" action="update">
<database-name>/usr/WebSphere/AppServer/bin/myidb.prp</database-name>    <jdbc-driver-name>Admin DB
Driver</jdbc-driver-name>    <minimum-pool-size>1</minimum-pool-size>
<maximum-pool-size>10</maximum-pool-size>    <connection-timeout>120000</connection-timeout>
<idle-timeout>180000</idle-timeout>    <orphan-timeout>1800000</orphan-timeout>

```

The following section contains the [node](#), or physical machine, in the administrative domain. Its hostname is buccaneer:

- [Skip ahead](#)
- [View object syntax](#)

```

<node name="buccaneer" action="update">
<deployed-jar-directory>/usr/WebSphere/AppServer/deployedEJBs</deployed-jar-directory>
<dependent-classpath></dependent-classpath>

```

Next there is an [application server](#). In this case, it is the default application server, "Default Server":

- [Skip ahead](#)
- [View object syntax](#)

```

<application-server name="Default Server" action="update">    <executable>java</executable>
<command-line-arguments/>    <environment/>    <user-id></user-id>    <group-id></group-id>
<working-directory></working-directory>    <umask>18</umask>    <stdin></stdin>
<stdout>/usr/WebSphere/AppServer/logs/default_server_stdout.log</stdout>
<stderr>/usr/WebSphere/AppServer/logs/default_server_stderr.log</stderr>
<process-priority>20</process-priority>    <maximum-startup-attempts>2</maximum-startup-attempts>
<ping-interval>60</ping-interval>    <ping-timeout>200</ping-timeout>
<ping-initial-timeout>300</ping-initial-timeout>
<selection-policy>roundrobinpreferlocal</selection-policy>
<trace-specification></trace-specification>    <trace-output></trace-output>
<transaction-log-file></transaction-log-file>    <system-properties/>
<debug-enabled>>false</debug-enabled>    <transaction-timeout>120</transaction-timeout>
<transaction-inactivity-timeout>60000</transaction-inactivity-timeout>
<thread-pool-size>20</thread-pool-size>    <security-enabled>>false</security-enabled>

```

The application server also contains a [servlet engine](#)

- [Skip ahead](#)
- [View object syntax](#)

```

<servlet-engine name="Default Servlet Engine" action="update">
<maximum-connections>25</maximum-connections>    <transport-port>-1</transport-port>
<servlet-mode>1</servlet-mode>    <transport-type name="ose">    <ose-transport>
<link-type>local</link-type>    <log-file-mask trace="false" inform="false" warning="false"
error="true"/>    <queue-name>ibmoselink</queue-name>
<clone-index>1</clone-index>    <native-log-file>native.log</native-log-file>
</ose-transport>    </transport-type>    <isclone>>false</isclone>

```

In turn, the servlet engine contains a few [Web applications](#). Two are omitted from this example. The third (shown here) is named "examples":

- [Skip ahead](#)
- [View object syntax](#)

```

<web-application name="examples" action="update">    <description>Useful Example
Servlets</description>
<document-root>/usr/WebSphere/AppServer/hosts/default_host/examples/web</document-root>
<classpath>    <path value="/usr/WebSphere/AppServer/hosts/default_host/examples/servlets"/>
</classpath>    <error-page>/debug_error.jsp</error-page>    <session-config>
    <session-timeout>0</session-timeout>
</session-config>    <mime-mapping>
    <extension>*.gam</extension>
    <mime-type>nixon</mime-type>
</mime-mapping>
<welcome-file-list>
    <welcome-file>slappy.html</welcome-file>
</welcome-file-list>
<error-page-j2ee>

```

```

        <exception-type>anException</exception-type>
        <location>except.html</location>
    </error-page-j2ee>
</error-page-j2ee>
    <error-code>333</error-code>
    <location>horrid.html</location>
</error-page-j2ee>
<taglib>
    <taglib-uri>/WEB-INF/app.tld</taglib-uri>
    <taglib-location>/WEB-INF/app.tld</taglib-location>
</taglib>
    <filter-list/>          <group-attributes/>          <auto-reload>true</auto-reload>
<reload-interval>9000</reload-interval>          <enabled>true</enabled>
<root-uri>default_host/webapp/examples</root-uri>          <shared-context>false</shared-context>
<shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name>
<isclone>false</isclone>

```

The example app contains several servlets. The first one is administered by the name "simpleJSP":

- [Skip ahead](#)
- [View object syntax](#)

```

        <servlet name="simpleJSP" action="update">          <description>Simple JSP
Servlet</description>          <code>SimpleJSPServlet</code>          <init-parameters>
<parameter name=" " value=" " />          </init-parameters>
<load-at-startup>false</load-at-startup>          <debug-mode>false</debug-mode>
<uri-paths>          <uri value="/simpleJSP.servlet"/>          <uri value="/simpleJSP"/>
</uri-paths>          <enabled>true</enabled>          <isclone>false</isclone>
</servlet>

```

The Web application contains some other servlets that are not listed here. The Web application ends:

```

</web-application>

```

Now, a [session manager](#) follows. Recall, the tag for the servlet engine is still open, indicating that all of these objects are contained by the servlet engine:

- [Skip ahead](#)
- [View object syntax](#)

```

    <session-manager name="Session Manager" action="update">
<enable-sessions>true</enable-sessions>          <enable-url-rewriting>false</enable-url-rewriting>
<enable-cookies>true</enable-cookies>
<enable-protocol-switch-rewriting>false</enable-protocol-switch-rewriting>          <cookie
name="sesessionid">          <comment>servlet session support</comment>
<domain></domain>          <maximum>-1</maximum>          <path></path>
<secure>false</secure>          </cookie>
<interval-invalidation-time>1800</interval-invalidation-time>
<persistent-sessions>false</persistent-sessions>
<persistence-type>directodb</persistence-type>          <database location="jdbc:db2:was">
<driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>          <user-id></user-id>
<password></password>          <number-of-connections>30</number-of-connections>
</database>          <enable-stat-collection>true</enable-stat-collection>
<using-cache>false</using-cache>          <using-multi-row>false</using-multi-row>
<using-manual-update>false</using-manual-update>
<using-native-access>false</using-native-access>          <base-memory-size>1000</base-memory-size>
<allow-overflow>true</allow-overflow>          <data-source name=" " />          </session-manager>

```

Next, the servlet engine contains a [user profile manager](#):

- [Skip ahead](#)
- [View object syntax](#)

```

    <user-profile-manager name="User Profile Manager" action="update">
<enable-user-profile>false</enable-user-profile>
<data-wrapper>com.ibm.servlet.personalization.userprofile.UserProfile</data-wrapper>
<remote-interface-ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnly</remote-interface-ro>
<remote-interface-rw>com.ibm.servlet.personalization.userprofile.UP_ReadWrite</remote-interface-rw>
<home-interface-ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnlyHome</home-interface-ro>
<home-interface-rw>com.ibm.servlet.personalization.userprofile.UP_ReadWriteHome</home-interface-rw>
<jndi-name-ro>UP_ReadOnlyHome</jndi-name-ro>          <jndi-name-rw>UP_ReadWriteHome</jndi-name-rw>
</user-profile-manager>

```

The user profile manager is the last object in the servlet engine and the application server. The end tags for each are displayed:

```

</servlet-engine>          </application-server>

```

Then the end tag for the node is placed, indicating that all the elements pertaining to the node have been addressed:

</node>

Several enterprise application entries come after this (only the first is shown here):

- [Skip ahead](#)
- [View object syntax](#)

```
<enterprise-application name="AdminApplication" action="create">    <uri name="/admin"/>
<web-application name="admin"/> </enterprise-application>
```

Followed by several URI security entries (only one is shown):

- [Skip ahead](#)
- [View object syntax](#)

```
<uri-security>    <uri name="/admin"/>    <method-group-mapping method="HTTP_GET"
method-group="ReadMethods"/>    <method-group-mapping method="HTTP_POST"
method-group="ReadMethods"/>    <method-group-mapping method="HTTP_PUT"
method-group="WriteMethods"/>    <method-group-mapping method="HTTP_DELETE"
method-group="RemoveMethods"/> </uri-security>
```

Followed by the server security configuration:

- [Skip ahead](#)
- [View object syntax](#)

```
<security-config security-enabled="false" security-cache-timeout="600">    <app-security-defaults>
<realm-name>raleigh.ibm.com</realm-name>    <challenge-type ssl-enabled="false">
<basic-challenge/>    </challenge-type>    </app-security-defaults>    <auth-mechanism>
<localos>    <user-id>root</user-id>    <password>$server-password$</password>
</localos>    </auth-mechanism> </security-config>
```

The last set of tags defines the method groups:

- [Skip ahead](#)
- [View object syntax](#)

```
<method-group>ReadMethods</method-group>    <method-group>WriteMethods</method-group>
<method-group>RemoveMethods</method-group>    <method-group>CreateMethods</method-group>
<method-group>ExecuteMethods</method-group>    <method-group>FinderMethods</method-group>
```

The last item is the end tag for export itself:

```
</websphere-sa-config>
```

6.6.0.2.1.1.2: XMLConfig - Example of a partial export

To do a partial export of your Websphere Administrative Domain configuration into an XML file, you need to create an XML file specifying the resources you would like to export. This partial file is then used as an input parameter in the XMLConfig export command line.

The partial XML file always begins with the following two header lines:

```
<?xml version="1.0"?><!DOCTYPE websphere-sa-config SYSTEM
"$server_root$$$dsep$bin$$$dsep$xmlconfig.dtd" >
```

The contents of the Websphere Administrative Domain that you wish to extract into an XML file start with <websphere-sa-config> and end with </websphere-sa-config>. What goes in between these tags depends on what you want to export. This is an example of a partial XML file you would create to export the entire contents of an Application Server on a node named "mynode":

```
<?xml version="1.0"?><!DOCTYPE websphere-sa-config SYSTEM
"$server_root$$$dsep$bin$$$dsep$xmlconfig.dtd"><websphere-sa-config <node name="mynode"
action="locate"> <application-server name="Default Server" action="export">
</application-server> </node></websphere-sa-config>
```

The next example is of an XML file you would create to export a single Web application from the same Application Server on the same node:

```
<?xml version="1.0"?><!DOCTYPE websphere-sa-config SYSTEM
"$server_root$$$dsep$bin$$$dsep$xmlconfig.dtd" ><websphere-sa-config <node name="mynode"
action="locate"> <application-server name="Default Server" action="locate">
<servlet-engine name="Default Servlet Engine" action="locate"> <web-application
name="default_app" action="export"> </web-application> </servlet-engine>
</application-server> </node></websphere-sa-config>
```

As an example of how you would do a partial configuration export into an XML file, we can name the second file above (the one that would export a single Web application from the node "mynode") *PartialFile.xml*, place it in the appserver/bin directory, and run the following command from that directory:

```
XMLConfig -export NewExport.xml -adminNodeName mynode -partial PartialFile.xml
```

An XML file named *NewExport.xml* will then be created in the appserver/bin directory with the following output:

```
<?xml version="1.0"?><!DOCTYPE websphere-sa-config SYSTEM
"$XMLConfigDTDLocation$$$dsep$xmlconfig.dtd" ><websphere-sa-config <node name="mynode"
action="locate"> <application-server name="Default Server" action="locate"> <servlet-engine
name="Default Servlet Engine" action="locate"> <web-application name="default_app"
action="update"> <description>Default Application</description>
<document-root>d:\IBM HTTP Server\htdocs</document-root> <classpath> <path
value="d:\WebSphere\AppServer\hosts\default_host\default_app\servlets"/> <path
value="d:\WebSphere\AppServer\servlets"/> </classpath>
<error-page>/ErrorReporter</error-page> <session-config>
<session-timeout>0</session-timeout> </session-config> <welcome-file-list/>
<filter-list/> <group-attributes/> <auto-reload>true</auto-reload>
<reload-interval>9</reload-interval> <enabled>true</enabled>
<root-uri>default_host/</root-uri> <shared-context>false</shared-context>
<shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name> <servlet name="snoop"
action="update"> <description>Snoop servlet</description>
<code>SnoopServlet</code> <init-parameters> <parameter name="param1"
value="test-value1"/> </init-parameters>
<load-at-startup>false</load-at-startup> <debug-mode>false</debug-mode>
<uri-paths> <uri value="/servlet/snoop/*"/> <uri
value="/servlet/snoop2/*"/> </uri-paths> <enabled>true</enabled>
</servlet> <servlet name="hello" action="update"> <description>Simple Hello
servlet</description> <code>HelloWorldServlet</code> <init-parameters/>
<load-at-startup>false</load-at-startup> <debug-mode>false</debug-mode>
<uri-paths> <uri value="/servlet/hello"/> </uri-paths>
<enabled>true</enabled> </servlet> <servlet name="ErrorReporter" action="update">
<description>Default error reporter servlet</description>
<code>com.ibm.servlet.engine.webapp.DefaultErrorReporter</code> <init-parameters/>
<load-at-startup>true</load-at-startup> <debug-mode>false</debug-mode>
<uri-paths> <uri value="/ErrorReporter"/> </uri-paths>
<enabled>true</enabled> </servlet> <servlet name="invoker" action="update">
<description>Enables Serving Servlets By Classname</description>
<code>com.ibm.servlet.engine.webapp.InvokerServlet</code> <init-parameters/>
<load-at-startup>true</load-at-startup> <debug-mode>false</debug-mode>
<uri-paths> <uri value="/servlet/*"/> </uri-paths>
<enabled>true</enabled> </servlet> <servlet name="jsp10" action="update">
<description>JSP 1.0 support servlet</description>
<code>com.sun.jsp.runtime.JspServlet</code> <init-parameters/>
<load-at-startup>true</load-at-startup> <debug-mode>false</debug-mode>
<uri-paths> <uri value="*.jsp"/> <uri value="*.jsw"/> <uri
value="*.jsw"/> </uri-paths> <enabled>true</enabled> </servlet>
</web-application> </servlet-engine> </application-server> </node></websphere-sa-config>
```

6.6.0.2.1.2: XMLConfig grammar

This section discusses the general structure of an XML element. For detailed information about each object, see the [package summary file](#). See also the full export [example](#) page.

Each object tag in the XML document contains:

- An object type, such as "servlet" or "virtual-host"
- A **name** attribute that identifies the particular resource on which to perform the action
- An **action** attribute that controls the behavior of the import or partial export operation

For example, a servlet engine named MyServletEngine has the object tag:

```
<servlet-engine name="MyServletEngine" action="update" >
```

In this case, the action is "update." The list below describes all of the available actions.

Unless otherwise stated, the actions apply only to the import operation.

create

Adds the specified resource to the administrative domain. If the resource already exists, the create action is treated as an update action. When using this action, you must specify all required attributes for the object type.

update

Updates the properties of a specified resource. You need only specify the properties you want to update.

However, if the resource does not exist, the update action becomes a create. In such a case, if some of the properties are not specified, an error will occur.

delete

Removes the specified resource and its children (recursive delete).

locate

Locates the specified resource. Use this action to provide the containment path to specific child resources. Applies to the partial export operation as well as the import operation.

export

Exports the configuration of the specified resource and its children to the output XML document. Applies only to the partial export operation.

start

Starts the specified resource (if applicable).

stop

Stops the specified resource, (if applicable).

stopforrestart

Stops the specified resource, and restarts it.(if applicable). Node only.

restart

Stops the specified resource and starts it again (if applicable).

enable

Makes the specified resource available for user requests (currently applies only to servlets and Web applications).

disable

Makes the specified resource unavailable for user requests (currently applies only to servlets and Web applications).

createclone

create a clone

associateclone

associate a clone to a model

disassociateclone

disassociate a clone from a model

 Some of the operational attributes, such as start and stop, do not apply to all object types. In general, if the operation is supported in the WebSphere Administrative Console, it is supported in the XML import operation.

6.6.0.2.1.3: XMLConfig - Using the tool programmatically

The XMLConfig class is structured so that you can use it programmatically to retrieve information as a TXDocument or Element. The import/export facility can thus be included in a Java program, as well as being operated from a command line.

Creating platform-neutral configurations

For import and partial export operations, a variable substitution operation is performed on the input XML document, allowing you to create platform-neutral XML documents. These variables are available:

\$server_root\$

Replace with the WebSphere Application Server installation directory, such as C:\WebSphere\AppServer on Windows NT.

\$psep\$

Replace with the path separator as specified by the operating system JDK.

- On Windows NT, it is ; (semicolon)
- On AIX and Solaris, it is : (colon)

\$dsep\$

Replace with the directory separator as specified by the operating system JDK.

- On Windows NT, it is \ (backward slash)
- On AIX and Solaris, it is / (forward slash)

Security XML configurations

When configuring security with XML, you should be familiar with the following two factors:

1. [Passwords and variable substitution](#)
2. [Searches of the user registry](#)

Javadoc for the tool

It is recommended that you refer to the Javadoc for the latest programmatic use of XMLConfig, and refer to the exported xml for the sample xml for repository objects.

Javadoc for com.ibm.websphere.xmlconfig class and all of the related object classes resides in the apidocs directory:

```
<installation_root>\web\apidocs\package and class name
```

See the [package summary file](#) for a list of the class names, such as ApplicationServerConfig. The Javadoc is labeled by the class name preceded by the package name, com.ibm.websphere.

6.6.0.2.1.3.1: XMLConfig - Passwords and variable substitution

Passwords are required for the WebSphere security configuration, but exposing passwords during an XML import or export by putting an unencrypted password in the XML text file would create an unacceptable security risk. Thus, WebSphere Application Server uses special password tags in the exported XML file that signify XML variables that replace the real passwords during an import. Consequently, the passwords are only part of the XMLConfig command line, which you can clear after invoking the utility.

Three of the password variables have constant names while the others depend on the name to what they are related. The three constant variable names are:

server-password

This is the password for the ID that has all permissions and rights to access the administrative server. It is the password that is entered during the installation process, and the one that appears on the User Registry tab of the Configure Global Security Settings task.

ltpa-password

This is the password for generating LTPA keys. It is the password entered on the LTPA Password dialog when changing the Authentication Mechanism to LTPA for the first time.

ldap-bindpwd

This is the password that the security server will use to bind to an LDAP directory during searches. It is the password entered on the User Registry tab of the Configure Global Security Settings task when LTPA is the Authentication Mechanism.

<enterprise app name>_AppSecurityPwd [*where <enterprise app name> is the name of an enterprise application with a defined application identity*]

This is the password associated with the identity defined as the application identity. Application identities are set on the last panel of the Configure Application Security task and are limited to users defined in the configured user registry.

All password variables must be substituted on the XMLConfig command line using the -substitute parameter. Multiple substitutions are separated by a semicolon. For example, the server-password and LTPA-password variables may be substituted with the following command line:

```
XMLConfig -import was.xml -adminNodeName myhost -substitute  
"server-password=pwd1;LTPA-password=pwd2"
```

6.6.0.2.1.3.2: XMLConfig - User registry searches

The permission element accepts access-id-search children that allow you to specify user registry searches instead of explicit user registry entries. The values of these elements are substituted into the appropriate search filters for the specified types before the searches execute. This gives you an easy way to specify user registry entries for permissions, as opposed to knowing the exact and often complicated unique ID for each user registry entry.

There is a condition for searches that must be met for a permission import to succeed. Since an XML import is not interactive and the XML import utility has no way of choosing the intended match from multiple matches, user registry searches must return one and only one match. A permission import will fail if more than one match is returned from a search.

Not all types may be supported by a user registry even though the type attribute of an access-id-search element allows "user", "group", and "role". Therefore, importing a permission element will fail when the type specified is not supported by the user registry.

6.6.0.2.1.4: wartoxmlconfig script

The wartoxmlconfig.bat/sh scripts performs from the command line the same function as the [Convert WAR File task](#) on the console. The command line syntax is shown below:

```
wartoxmlconfig [war filename] [webapp destination] [admin node] [node] [server] [servlet engine]
[virtual host] [webapp path] [webapp name]
```

- **war filename** - full path to the WAR file
- **webapp destination** - directory where web application will be rooted (a subdirectory will be created that matches the specified web app name)
- **admin node** -name of the node containing the admin server you are connecting to (as shown in the Admin GUI)
- **node** - name of the node you are installing the WAR on to (as shown in the Admin GUI)
- **server** - name of the server you are installing the WAR on to (as shown in the Admin GUI)
- **servlet engine** - name of the servlet engine you are installing the WAR on to (as shown in the Admin GUI)
- **virtual host** - name of the virtual host you wish this application to be accessible from (as shown in the Admin GUI)
- **webapp path** - the context path of the web application
- **webapp name** - the name of the web application to be shown in the Admin GUI

Example:

```
wartoxmlconfig c:\temp\servlet-tests.war c:\websphere\appserver\hosts\default_host mynode mynode
"Default Server" "Default Servlet Engine" default_host /servlet-tests Servlet_Tests
```

6.6.0.2.1.5: Troubleshooting XMLConfig

This article describes what to do if XMLConfig fails and displays the following message at its command line.

Unable to create Initial Context. Please check name service settings:

```
org.omg.CORBA.COMM_FAILURE:  minor code: 3  completed: Nojava com.ibm.websphere.xmlconfig.XMLConfig
{ ( -import <xml data file> ) ||          [ ( -export <xml output file> [-partial <xml data file>] ) ] }
-adminNodeName <primary node name>      [ -nameServiceHost <host name> [ -nameServicePort <port
number> ] ]          [-traceString <trace spec> [-traceFile <file name>]]          [-substitute
<"key1=value1[;key2=value2[...]]">]}      In input xml file, the key(s)  should appear as $key$
for substitution.
```

One or more of the following conditions is likely to have caused the problem:

- The admin server didn't start properly.
- XMLConfig is being run remotely from the machine on which the product administrative server is installed, and the -nameServiceHost parameter has not been set to the hostname of the machine containing the administrative server.
- The naming service port was changed from the default 900 on the application server and the -nameServicePort parameter was not set to the changed port value
- The -nameServicePort parameter was defined, but the -nameServiceHost parameter was not.
- If the fully qualified host name (hostname + domain) is specified for the -adminNodeName parameter, instead of the node name (short host name), expect this error:

```
001.553 22f45b XMLConfig      X Unabled to export Virtual Host Data: {0}
javax.naming.NameNotFoundException:
```

To resolve the problem:

1. Check whether the administrative server started successfully, as described in the documentation for [starting the administrative server](#).
2. If the XMLConfig tool is being run remotely with respect to the machine on which the IBM Websphere administrative server is running, ensure that the -nameServiceHost (as well as the -adminNodeName) parameter are set to the host name of the remote machine.
3. If the naming service port has been changed from the default 900 on the machine on which the administrative server is running, look in the administrative server configuration file for the parameter:

```
com.ibm.ejs.sm.adminServer.bootstrapPort
```

Set the XMLConfig -nameServicePort parameter to this port value.

4. If you use the -nameServicePort parameter, you must also use the -nameServiceHost parameter even if you are running XMLConfig on the same machine as the administrative server.
5. Modify the -adminNodeName parameter to use the node name, which can be found as your hostname in your TCP/IP networking configuration and also in the Websphere administrative console under "Websphere Administrative Domain."

6.6.0.2.2: WebSphere Control Program (wscp)

The WebSphere Control Program (wscp) is a command line client for the administrativeserver. It can be used to administer application servers, enterprise applications, and other types of WebSphere Application Server objects.

6.6.0.3: Web administrative console overview

The Web administrative console is a light client that runs in a Web browser. You can use it to work with a subset of the resources in the administrative domain. It provides the opportunity to work with property files encoded in eXtensible Markup Language (XML).

The table describes what you can use the Web administrative console to do. Click an activity for a detailed overview of the activity and links to help with the related tasks and settings.

Activity	Brief description
Perform tasks	Perform two main kinds of tasks: (1) create objects, and (2) export the workspace to XML (described separately below).
Create resources (objects)	Create new resources of a variety of types. This console supports the most popular resources in the administrative domain.
Configure resources	View and configure existing resources, including resources you created during this session, and previously existing resources of the supported types.
Submit modifications	Accumulate changes in the local memory (workspace) as you create and modify resources. Modify a property sheet, then use the button at the bottom to save the changes in the workspace.
Commit all modifications	When ready, you can commit the changes you have accumulated in the local workspace. Committing the changes to the administrative database makes the administrative domain aware of the changes. If you do not commit them, the changes will be discarded when you close the browser.
Export workspace to XML	Instead of, or in addition to, committing the modifications, you can save them to an XML file. You can later use another administrative client to import the file into the administrative domain.

Learn more about the [administrative model](#).

6.6.0.3.1: Tasks overview

The Tasks part of the navigation tree provides forms for creating new objects and for exporting administrative changes to an XML file.

6.6.0.3.1.1: Performing tasks

Use the Tasks to create new objects (administrative resources):

1. Select a "Create" task to display a form.
2. Specify properties in the form.
3. To save the form changes locally, Submit the form.
4. To apply the changes to the administrative domain, Commit All Modifications.

You can also export the local workspace changes to an XML file. You can later use another administrative client to import the file into the administrative domain and apply the changes.

If you do not either commit or export your changes, they will be discarded when you end the administrative session (close the browser).

6.6.0.3.1.2: Properties for performing tasks

The table provides links to the property help for various tasks. Most tasks allow you to create a new resource.

Please note, the property help is combined for all WebSphere administrative clients. Some properties listed in the property help might not be available from the Web console, or might differ slightly in name.

Properties for Create Virtual Host task
Properties for Create Server task
Properties for Create Servlet Engine task
Properties for Create Web Application task
Properties for Create JDBC Driver task
Properties for Create Data Source task
Properties for Create Servlet task
Properties for Export Workspace to XML task (button help)

6.6.0.3.2: Resources overview

The Resources part of the navigation tree displays the existing resources that you can administer with the Web administrative console. Use the table to find descriptions and task help for each resource type.

Resource Description	Documentation for administering resource
0.2: What are nodes?	6.6.2: Administering nodes
0.3: What are application servers?	6.6.3: Administering application servers
0.7: What are servlet engines?	6.6.7: Administering servlet engines
0.8: What are Web applications?	6.6.8: Administering Web applications
0.9: What are servlets?	6.6.9: Administering servlets
0.10: What are JSP files?	6.6.10: Administering JSP files
0.14: What is data access?	6.6.14: Administering database connections
0.16: What are virtual hosts?	6.6.16: Administering virtual hosts
0.17: What are Web resources?	6.6.17: Administering Web resources

 The Resources tree does not necessarily show all of the resources in the WebSphere administrative domain:

- The Web administrative console displays only the *subset* of resource types that it supports. Additional resources types can be administered with other WebSphere administrative clients.
- The Resources tree displays only *existing* resources. If no resources of a supported type have been created, then no resources of that type will be displayed.

6.6.0.3.2.1: Administering resources

Use the Resources part of the navigation area to view and configure existing resources. The console displays resources you created during the current session, as well as previously existing resources of the types supported by the console.

To edit resource properties:

1. Select a resource to display a form with buttons at the bottom.
2. Edit the resource properties.
3. Click the Submit Modifications button to save the changes locally.

Depending on the resource type, you might also be able to:

- Delete the resource
- Create another resource with the same properties
- Install the resource on a node (pertains to JDBC driver)

6.6.0.3.2.2: Properties for configuring resources

The table provides links to the property help for various resources in the administrative domain.

Please note, the property help is combined for all WebSphere administrative clients. Some properties listed in the property help might not be available from the Web console, or might differ slightly in name.

Virtual host properties	Application server properties
Servlet engine properties	Web application properties
JDBC driver properties	Data source properties
Servlet properties	Button help

6.6.0.3.3: Create objects overview

Use the tasks under Create Objects to create new resources for the WebSphere administrative domain. The console supports a subset of the available resources.

Certain resources cannot be created until other resources exist. For example, when you create a servlet engine, you must specify an existing application server on which to install it. It is recommended that you follow the creation order presented in the navigation tree (virtualhost first, servlet last).

6.6.0.3.3.1: Creating objects

To create an object:

1. Expand Tasks.
2. Expand Create Objects.
3. Select a "Create" task to display a form.
4. Specify properties in the form.
5. To save the form changes locally, Submit the form.
6. To apply the changes to the administrative domain, Commit All Modifications.

6.6.0.3.3.2: Properties for creating objects

The table provides links to the property help for various tasks. Most tasks allow you to create a new resource.

Please note, the property help is combined for all WebSphere administrative clients. Some properties listed in the property help might not be available from the Web console, or might differ slightly in name.

Properties for Create Virtual Host task
Properties for Create Server task
Properties for Create Servlet Engine task
Properties for Create Web Application task
Properties for Create JDBC Driver task
Properties for Create Data Source task
Properties for Create Servlet task

6.6.0.3.4: Export to XML overview

Using the Web console, you can preserve administrative work in an XML file. The Submit button available on the property forms for creating and configuring various resources saves your administrative changes to an XML file.

The XML file contains XML-based configurations of resources in the WebSphere administrative domain. Using XMLConfig, the commandline client for manipulating XML configuration files, you can import the file into the WebSphere administrative domain, applying the changes.

The export to XML feature can be useful for saving administrative changes for later. It also allows hand-editing of XML configuration files, instead of performing the work through a graphical interface.

6.6.0.3.4.1: Exporting the workspace to XML

1. In the navigation tree, expand Tasks and select Export Workspace to XML.
2. In the resulting form on the right side of the console, specify the details, such as where to save the file.
3. Import the XML file into the WebSphere administrative domain at a later time, if you like. See section 6.6.0.2 for instructions.

6.6.0.3.5: Submit and commit modifications overview

You can preserve administrative work in an XML file by submitting the work. When you finish modifying a resource, click the Submit button to submit the changes before leaving the property form. If you leave a property form without submitting your changes, the changes will be lost when you end the administrative session (close the browser).

You can apply the work to the WebSphere administrative repository by submitting the work, then committing the work.

Prior to being committed, submitted administrative work is added to a queue of actions to perform against the WebSphere administrative domain. The queue of actions is similar to a list of transactions waiting either to be committed to a database, or rolled back.

Several submitted changes can be applied to the administrative repository as a batch or set. Updating the administrative domain from a Web administrative console on a machine remoteto the administrative server and database can be a time consuming operation that involves initiating connections. In such an environment, sending a collection of updates is more efficient.

Alternatively, you have the option to Commit individual changes as soon as you submit them. If you end the current Web-based administrative session without committing a change, the change will not be applied to the administrative domain.

6.6.0.3.5.1: Submitting and committing modifications

To submit modifications:

1. Click a task or resource in the navigation tree to display its property form.
2. When finished modifying the property form, click the Submit button at the bottom of the form.

To commit modifications:

1. Select Commit All Modifications to view the submitted changes.
2. Use the Select All and Clear All buttons to select some or all of the changes to commit.
3. When ready, click the Commit button.

6.6.0.3.5.2: Properties for submitting and committing modifications

To submit and commit modifications, you do not need to specify any settings. Simply use the Submit, Select All, Clear All, and Commit buttons as described in the [Web administrative console button help](#) and [task help](#).

6.6.0.3a: Starting and stopping the Web administrative console

The Web console is available on machines containing WebSphere administrative servers. You can use a Web browser to access it remotely from machines that do not contain an administrative server.

- [Starting the console](#)
- [Stopping the console](#)

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 the default application server is also running.
3. In a Web browser, type:

```
http://your.server.name/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. From the HTML page displayed, select XML Web Administration Tool.
5. Wait for the console to load into the browser.

Stopping the console

To stop the console, simply close the browser. Make sure you submitted your changes and either exported them to XML or committed them. Otherwise, they will be lost when you close the browser. See 6.6.0.3 for more information.

6.6.0.3c: Buttons in the Web administrative console

Submit Modifications

Submit any changes you made to the task or resource with which you are currently working. Submitting changes saves them to the workspace (local memory).

Submitted changes are added to a queue of changes you can consider whether to commit. Use Commit All Modifications to view the queue.

Delete

Delete the resource with which you are currently working.

Create Like...

Create a resource just like the current resource. The new resource will be of the same type, and have the same values for its properties.

You will be prompted to specify values for the properties that must have unique values for the resource, such as the name of the new resource.

Select All

Select all modifications, allowing you to commit them all simultaneously by clicking the Commit button.

Clear All

Deselect all modifications.

Commit

Commit the selected modifications to the administrative database.

A committed change does not necessarily take effect immediately. Committing a change simply means that the administrative domain is now aware of the change and will begin to handle it as it would any such change.

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.

The work area buttons are for submitting property changes or performing other actions relevant to the task or resource with which you are currently working. See the [Related information](#) for details.

6.6.0.4: Overview of editing property files by hand

Few resources can be edited through product property files (such as admin.config) and the [dynamic caching](#) feature of the servlet engine.

The following articles describe methods in which you edit a propertyfile that was shipped with the product, and is known by some component of the product. A prime example of such a property file is the [administrative server configuration file](#). Without additional actions by you, your changes to the property file are read and used by the component.

6.6.2.4: Administering nodes
6.6.3.4: Administering application servers
6.6.7.4: Administering servlet engines
6.6.7.4: Administering servlet engines
6.6.9.4: Administering servlets
6.6.14.4: Administering database connections
6.6.16.4: Administering virtual hosts

6.6.0.4.1: Dynamic caching of Servlets and JSPs

Use the *dynamic caching* feature to enhance WebSphere Application Server performance. This in-memory cache uses hooks in WebSphere Application Server's servlet engine to intercept calls to a servlet's service method, checking whether requests can be served from the cache.

After a servlet is executed once, a cache entry is generated containing:

- The servlet's output
- Results of the servlet's execution, including calls to other servlets and JSP files
- Metadata about the servlet's entry in the cache, including timeout values and entry priority information

Each entry in the cache is unique, distinguished by an id string object generated from the *HttpServletRequest* object. A different id string object is generated for each invocation of the servlet. With this process, a servlet can be cached based on:

- Request parameters
- The URI used to invoke the servlet
- Session information
- Other servlet properties

To enable dynamic caching, administrators and developers must build two XML configuration files:

- *dynacache.xml* - the global caching administration file
- *servletcache.xml* - the individual servlet caching configuration file

Both XML files use UTF-8 (the 8-bit Universal Character Set transformation format) character encoding. Therefore any short Unicode string can be used as a string variable.

6.6.0.4.1.1: Description of the dynacache.xml file

The servlet cache can be enabled or disabled by the presence of the *dynacache.xml* file. If the file is not present, the servlet engine functions in the usual manner. However, caching is enabled if the *dynacache.xml* file is present in directory:

```
\WebSphere\AppServer\properties
```

Use the *dynacache.xml* file to configure the size of the cache and to register external caches to be used by applications.

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 (document type definition) for the *dynacache.xml* file is specified in the *dynacache.dtd* file which ships with WebSphere Application Server and is located in directory:

```
\WebSphere\AppServer\properties
```

The *dynacache.dtd* file defines the root element, `<cacheUnit>`.

The *dynacache.dtd* file is included in the xml file through a **DOCTYPE** declaration, as for example:

```
<!DOCTYPE cacheUnit SYSTEM "dynacache.dtd">
```

Therefore, the beginning of the *dynacache.xml* file looks like:

```
<?xml version="1.0"?>
<!DOCTYPE cacheUnit SYSTEM "dynacache.dtd">
```

Configuring the global cache operation

The root `<cacheUnit>` element should contain one `<cache>` element that provides global settings for all Application Servers on a node. The format of the `<cache>` element is:

```
<cache size="entries" priority="default_priority"/>
```

In this example, the variables, *entries* and *default_priority*, are defined as:

- *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 value.

Note: The size of the cache is not limited by a memory cap (e.g., 20 megabytes), but by the number of distinct elements that can be held in the cache. 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 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 the *servletcache.xml* file. The recommended value for priority is 1.

Note: 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. For more information on the priority parameter, see the [priority documentation](#) in the *servletcache.xml* file description.

So to declare a cache storing at most 10,000 entries and giving all entries a default priority of one, you would enter:

```
<cache size="10000" priority="1"/>
```

Controlling external caches

WebSphere Application Server can control external caches. You can define different external caches, each of which can have 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 with the `<externalCacheGroups>` element. You can also use the `<externalCacheGroups>` element in the *servlecache.xml* file. Each group is further defined with the `<group>` element. Group elements can contain `<member>` elements.

The format of a `<group>` element is:

```
<group id="group_name">...</group>
```

The variable, *group_name*, is a string identifier for a group of external caches. For a servlet to be externally cacheable, a *group_name* must be used to identify the external group when the servlet is configured in the *servlecache.xml* file.

The format of the `<member>` element is:

```
<member address="address" adapterBeanName="adapter_class"/>
```

The variable, *address*, is the hostname or IP address of the external cache location.

The variable, *adapter_class*, is the package and class name of the adapter for the external cache. The *adapter_class* string is passed to the adapter bean's `setAddress` method. Check the adapter bean documentation for the format of the string.

The format of an `<externalCacheGroups>` stanza is:

```
<externalCacheGroups> <group id="edgeservers"> <member address="edgeone"
adapterBeanName="my.package.EdgeAdapter"/> <member address="edgetwo"
adapterBeanName="other.package.OtherAdapter"/> </group> </externalCacheGroups>
```

6.6.0.4.1.1.1: Quick reference - dynacache.xml file

The **dynacache.xml** file, located in directory, \WebSphere\AppServer\properties, consists of the following elements:

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="sever name" adapterBeanName="some.package.Adapter"/>`
8. `</group >`
9. `</externalCacheGroups>`
10. `</cacheUnit>`

A completed **dynacache.xml** file may 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>
```

6.6.0.4.1.2: Description of the servletcache.xml file

Administrators identify the servlets to be cached in the *servletcache.xml* file. The *servletcache.xml* file is located in directory:

```
\WebSphere\AppServer\properties
```

Within the <servlet> element, you specify parameters that:

- Identify the servlets to be cached
- Determine how entry ids for servlets are created
- Determine how servlet entries are deleted.

Typically, the *servletcache.xml* file contains multiple <servlet> elements.

Note: The *servletcache.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 (document type definition) for the *servletcache.xml* file is specified in the *servletcache.dtd* file that ships with WebSphere Application Server and is located in directory:

```
\WebSphere\AppServer\properties
```

The *servletcache.dtd* file defines the root element, <servletcache>.

The *servletcache.dtd* file is included in the xml file through a **DOCTYPE** declaration, as for example:

```
<!DOCTYPE servletcache SYSTEM "servletcache.dtd">
```

Therefore, the beginning of the *servletcache.xml* file looks like:

```
<?xml version="1.0"?>
<!DOCTYPE servletcache SYSTEM "servletcache.dtd">
```

Identifying what to cache

The cache parses the *servletcache.xml* file when WebSphere Application Server is started, and extracts a set of configuration parameters from each <servlet> element. Then each time a new servlet is initialized, the cache attempts to match the servlet with a servlet element to find the configuration information for that servlet.

You can specify the servlets to cache in two ways:

1. [Provide the class name of the servlet](#)
2. [Provide the full URI for the servlet.](#)

Note: The way you specify the servlet URI in the *servletcache.xml* file is different from the WebSphere Application Server convention where URIs are relative to the Web applications in which they are defined. In the *servletcache.xml* file, the URI begins with the hostname or IP address.

● Servletimpl -

Use this tag to cache a servlet class. The format of the <servletimpl> tag is:

```
<servletimpl class="ClassName" />
```

The variable *ClassName* identifies the class to be cached. This class must extend **HttpServlet**.

When a servlet class is initialized, dynacache matches the servlet with the set of config parameters declared in the <servlet> element. This comparison is made by matching strings. Subclasses of the specified servlet will not match this declaration.

● Path -

Use this tag to cache a servlet by its URI. The format of the <path> tag is:

```
<path uri="web_path" />
```

The variable *web_path* identifies the full URI of the servlet, beginning with the hostname. It does not include any CGI variables. The web application's web path must be a part of this parameter. So to access a cacheable servlet with URL, *http://servername.ibm.com/webappname/servlet/MyServlet?arg1=1&* that servlet's path element is:

```
<path uri="/webappname/servlet/MyServlet" />
```

You can specify different path elements for the same servlet. Also, the same path can appear in two different <servlet> elements. When configuring a servlet invoked by path, dynacache uses the configuration from the first valid, matching <servlet> element.

The following table describes the differences between caching a servlet by class name versus caching a servlet or JSP by web path:

Caching a servlet by class name	Caching a servlet by web path

Tags	<pre><servlet> <servletimpl class= "CalcServlet"/> (other configuration information) </servlet></pre>	<pre><servlet> <path uri="/tools/Calc"/> <path uri="/tools/servlet/CalcServlet"/> <path uri="/tools/Calculator.jsp"/> (other configuration information) </servlet></pre>
Behaviors	<p>Whether defined through the administration client or loaded with <code>InvokerServlet</code>, this configuration would match any servlet of class, CalcServlet. However, even if class ScientificCalculatorServlet extended CalcServlet, this configuration would not match ScientificCalculatorServlet.</p>	<p>This configuration yields a different behavior.</p> <p>The first path catches any request for the <code>/tools/Calc</code> URI, regardless of the servlet's class.</p> <p>The second path, <code>/tools/servlet/CalcServlet</code>, matches any calls to the <code>InvokerServlet</code> for the CalcServlet class.</p> <p>The third path, <code>/tools/Calculator.jsp</code>, enables caching on a JSP implementation of the calculator.</p>
Results	<p>This general setup caches every servlet of class CalcServlet, across the entire server, regardless of the URI.</p>	<p>This specific setup caches the defined servlet Calc (which may or may not be part of the CalcServlet class), and only caches CalcServlet if it is specifically invoked.</p>

Description of other configuration parameters in `servletcache.xml`

Within the `<servlet>` stanza, the administrator must define other elements that are used to build entries in the cache, or are required to remove entries.

The next two elements invalidate entries in the cache. See file, [6.6.0.4.1.3: Removing entries from a cache](#), for more information on clearing the cache.

- **Timeout** -

Determines when an entry is invalidated. If the `<timeout>` element is not present, then the `<servlet>` stanza that contains it, is invalidated and is not cached. The format of the `<timeout>` element is:

```
<timeout seconds="time_in_cache" />
```

The variable, `time_in_cache`, is the length of time in seconds that indicates when an entry, after it is created, should be removed from the cache. This value is required. If this value is zero or a negative number, the entry does not timeout and is only removed when the cache is full or programmatically, from within an application.

- **Priority** -

Determines how long an entry stays in the cache. The format of the `<priority>` element is:

```
<priority val="priority" />
```

The variable, `priority`, is a zero or positive integer designating the length of time an entry stays in the cache before being eligible for removal. If this element is not present, then the `default_value`, defined in the `<priority>` element in the `dynacache.xml` file, is used as the value for `priority`.

The next set of elements determines how requests are served.

- **Externalcache** -

Identifies external cache groups. The format of the `<externalcache>` element is:

```
<externalcache id="group_name" />
```

The variable, `group_name` is the name of an external cache group defined in the `dynacache.xml` file.

When serving external requests (that is, not when building servlet or JSP fragments), the generated page is copied to the external cache groups declared in the `dynacache.xml` file. This results in two parallel entries: one in WebSphere Application Server and another in the external caches. The external entry uses the full URI of the page as an id, and is timed out or invalidated at the same time as the local entry. To recap, when a page is cached, a copy of it is pushed to the external cache group identified by `group_name`.

- **MetaDataGenerator** -

Identifies a "user written" interface for handling invalidation, priority and external cache group. When a `<metadatagenerator>` is declared in the `servletcache.xml` file, it replaces the `<timeout>`, `<priority>`, and `<externalcache>` elements for a servlet.

The format of the `<metadatagenerator>` tag is:

```
<metadatagenerator class="class_name" />
```

The variable, `class_name`, is the full package and class name of a class extending `com.ibm.servlet.dynacache.MetaDataGenerator`. See the [programming documentation](#) for more information on creating the `MetaDataGenerator` interface.

The next set of elements determines how cache entries are created.

- **Description of how cache entries are created** -

Each time a servlet is called, WebSphere Application Server generates a corresponding `HttpServletRequest` object. The cache uses information in that object to build an id string to represent the call.

The value of the id string varies, depending on the tag information provided in the `servletcache.xml` file. It ranges from minimal default information (as for example the URI of the requested servlet), to user supplied combinations of request and session variables.

- **Using request and session variables**

When a servlet is called, the servlet receives variable information from three main sources:

1. [request parameters](#)
2. [request attributes](#)
3. [session parameters](#)

An administrator defines request and session variables, and then the cache uses the values of these variables to create ids for the entries. The request and session variables can also be used to group cache entries. The values of the variables are combined with id strings to generate data ids. Entries with specific data ids can join other, like entries in a group. Entries with specific data ids can also invalidate all existing members of that group. Using the `<invalidate>` tag, you can define an entry that does no caching, and only invalidates groups.

- **Request parameters and attributes -**

A client browser can set request parameters with CGI, when submitting a form. Servlets can set attributes on an **HttpServletRequest** object, and then forward or include that request to another servlet.

The format of the `<request>`, `<parameter>`, and `<attribute>` elements is:

```
<request> <parameter id="parameter_name" data_id="group_identifier"
invalidate="group_identifier" required="true|false"/> <attribute id="attribute_name"
method="method_name" data_id="group_identifier" required="true|false"/> </request>
```

The variable, *parameter_name*, is the name of the request parameter. The value of *parameter_name* is used to generate cache entry ids.

The variable, *attribute_name*, is the name of the request attribute. The output of one or more of this object's methods is used to generate cache entry ids.

The variable, *method*, is the name of a method in this request attribute. The output of this method is used to generate cache entry ids. If this value is not specified, **toString** is assumed. This method does not take any arguments. Do not include parenthesis in the method name when defining this variable.

The variable, *group_identifier*, is a string that, combined with the value of the request variable, generates a group name for the cache entry. If *group_identifier* is specified on the `<data_id>` element, the cache entry is placed in that group. If *group_identifier* is used in an `<invalidate>` entry, the group is invalidated, and the entry is placed in the cache. If a *group_identifier* is not defined, these actions will not occur.

The element, `<required>`, indicates whether a value must be present in the request. If `<required>` is set to *true* and either the `<parameter>` or `<attribute>` element is not present, or the *method* is missing in an `<attribute>` definition, the request is not cached. The `<required>` value defaults to false when not set.

Note: You can define unlimited `<parameter>` and `<attribute>` elements within a `<request>` element, but you can only define **one** `<request>` element per a `<servlet>` element.

- **Session parameters -**

The cache handles session parameters in the same way it handles attributes.

The format of the `<session>` element is:

```
<session id="session_parameter_name" method="method_name"
data_id="group_identifier" invalidate="group_identifier" required="true|false"/>
```

The variable, *session_parameter_name*, is the name of the session parameter. The value of *session_parameter_name* is used to generate cache entry ids.

The variable, *method*, is the name of a method in this session parameter. The output of this method is used to generate cache entry ids. If this value is not specified, **toString** is assumed. This method does not take any arguments. Do not include parenthesis in the method name when defining this variable.

The variable, *group_identifier*, is a string that, combined with the value of the session variable, generates a group name for the cache entry. If *group_identifier* is specified on the `<data_id>` element, the cache entry is placed in that group. If *group_identifier* is used in an `<invalidate>` entry, the group is invalidated, and the entry is placed in the cache. If a *group_identifier* is not defined, these actions will not occur.

The element, `<required>`, indicates whether a value must be present in the session. If `<required>` is set to *true* and either the `<session parameter>` element is not present or does not contain the specified *method*, the request is not cached. The `<required>` value defaults to false when not set.

Note: You can define unlimited `<session parameter>` elements within a `<servlet>` element.

- **Invalidateonly -**

When this tag is used, no caching is performed for the servlet. However, invalidations triggered by this servlet will take effect. This prevents caching of control servlets (which would prevent the servlets from being executed).

The format of the `<invalidateonly>` element is:

```
<invalidateonly />
```

- **IdGenerator -**

Identifies a "user written" interface for handling parameters, attributes and sessions. When an `<idgenerator>` element is declared in the *servletcache.xml* file, it replaces the `<parameter>`, `<attribute>`, and `<session>` elements for a servlet.

The format of the `<idgenerator>` tag is:

```
<idgenerator class="class_name" />
```

The variable, *class_name*, is the full package and class name of a class extending com.ibm.servlet.dynacache.IdGeneratorClass. See the [programming documentation](#) for more information on creating the IdGenerator interface.

6.6.0.4.1.3: Removing entries from the cache

Once you decide which servlets to cache, you must build unique entries for different requests. You must also remove entries from the cache when necessary. You can remove entries using the *servletcache.xml* file or by writing your own class to handle invalidation.

The cache removes entries in three circumstances:

1. One of the cache's invalidation methods was called inside the servlet code.
(See the [programming documentation](#) for more information on cache invalidation methods.)
2. The timeout for the entry expired.
3. The cache is full and a new entry must replace an old one.

In the first case, removing the entry is done programmatically inside an application. In the other cases, entry invalidations are configured with the `<timeout>` and `<priority>` tags.

Both tags are located in the *servletcache.xml* file.

- **Timeout** determines when an entry is invalidated. If the `<timeout>` element is not present, then the `<servlet>` stanza that contains it, is invalidated and will not be cached. The format of the `<timeout>` element is:

```
<timeout seconds="time_in_cache" />
```

The variable, *time_in_cache*, is the length of time in seconds that indicates when an entry, after it is created, should be removed from the cache. This value is required. If this value is zero or a negative number, the entry will not timeout and can only be removed when the cache is full or programmatically, from within an application.

If the *time_in_cache* is a positive value, the timeout is added to the current time to determine when the entry will be invalidated. When that time occurs, if the entry is still in the cache, the cache will force its invalidation.

- **Priority** determines how long an entry stays in the cache. The format of the `<priority>` element is:

```
<priority val="priority" />
```

The variable, *priority*, is a zero or positive integer designating the length of time an entry stays in the cache before being eligible for removal. If this element is not present, then the *default_priority*, defined in the `<priority>` element in the *dynacache.xml* file, is used.

Note: the *default_priority* is an integer that defines the default priority for cacheable servlets defined in the *servletcache.xml* file. The recommended value for priority is 1.

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. On each cycle of the algorithm, the priority of an entry is decremented. Once it 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.

In summary, a higher priority provides a longer availability for an entry. Frequently requested entries stay in the cache longer. But regardless of the priority value and number of requests, an entry will be invalidated when `<timeout>` occurs.

Performance considerations

In a high volume application where space in the cache is at a premium, administrators should consider increasing the priority of a servlet or JSP under these conditions:

- when it is difficult to calculate the output of the servlet or JSP

- when the servlet or JSP is executed more often than average.

Note: Priority values should be low, as higher values will not yield much improvement, but will use extra LRU cycles. Therefore declaring a servlet with priority 3 and another with priority 4 generates the same invalidation behavior as servlets declared with priority 1 and 2, only marginally slower. When dealing with caches with thousands of entries, that slowdown becomes significant.

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. You can tailor the cache's operation using the priority variable.

6.6.0.4.1.4: Caching examples

1. The first caching example describes a servlet that receives request parameters from CGI variables provided through either an HTML form or an applet.

Servlet description

The **Calculator Servlet** is defined within a tools Web application. It takes 2 arguments and one operation:

- arg1
- arg2
- operation

The values for these variables are provided through CGI variables from an external user, (also known as **request parameters** from a browser or applet).

So to calculate 2+3, the servlet is invoked as:

```
URI/tools/Calc?arg1="2"&arg2="3"&operation="+"
```

Caching requirements

The output of the servlet must be cached so that each set of results is distinguished from the previous set.

Therefore, the results are distinguished by the **request parameters**.

The request parameters of **2,3,+** must have one cache entry.

The request parameters of **4,5,*** must have another cache entry.

Definition in *servletcache.xml*

```
<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>
```

Externally caching the output

Since this servlet relies on request parameter variables (which are set externally by users), it is possible to externally cache the output of this servlet.

Assume an external cache group named *extcache* is defined in the *dynacache.xml* file.

You can now assign this servlet's output to the external cache using the `<externalcache>` 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" />
  <externalcache id="extcache" />
</servlet>
```

```
</servlet>
```

2. The next caching example describes how to group and invalidate cache entries depending on the value of a variable.

Servlet description

The news servlet of class **CoastalNewsServlet** displays either west coast news or east coast news depending on the user's location.

This servlet has the following attributes:

- A session object named "location" of class **LocationBean**
- **LocationBean** has a method `getCoast()` that returns "east" or "west"

If the session object is not present on a session, the servlet will return the news for both coasts.

Caching requirements

All output from the servlet must be cached regardless if the session object, "location" is present on the session.

Therefore, the output to be cached is:

- east coast news if the `getCoast()` method returns "east"
- west coast news if the `getCoast()` method returns "west"
- news from both coasts if "location" is not present on the session

Also, the entries **cannot time out**.

Definition in `servletcache.xml`

In this example, since the **required** parameter is not defined, it defaults to **false** which means "location" is not required, and the request still is cached.

```
<servlet>
<servletimpl class="CoastalNewsServlet" />
<session id="location" method="getCoast" />
<timeout seconds="-1" />
</servlet>
```

Grouping and invalidating cache entries

Consider the design of the application when you want to group cache entries by coast, or invalidate all cache entries for a coast when the location bean is updated by a control servlet.

If a variable is not available to a servlet at execution (i.e., the request/session variable has not been set), then, even if the servlet is cacheable, no group id is generated.

Remember the behavior of the **CoastalNewsServlet**, where the missing session object, "location," causes the servlet to display the news for both coasts? In that case, you want the cache entry to belong to both the east and west coast groups. However, this will **not** occur because the cache does not handle data ids when variables are missing.

To resolve this problem, mark 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. The location bean must be present to place entries into the correct groups.

File, `servletcache.xml`, therefore, requires the following changes:

1. The **CoastalNewsServlet** entry must be modified to put entries into **groups**
2. A new entry for the control servlet must be added to allow invalidation of groups.
(The control servlet updates the location bean.)

Now when the news servlet is invoked, the cache takes the **data_id** "coast" and appends "=" and the value of **location.getCoast()** to create a group name to identify the entry.

In the update servlet, a **new_coast** string is expected as a request attribute, and is used in the same way to build group names for removal from the cache.

The new entries are **highlighted**.

```
<servlet>
<servletimpl class="CoastalNewsServlet" />
```

```
<session id="location" method="getCoast"/>
<timeout seconds="-1" />
</servlet>

<servlet>
<invalidateonly/>
<servletimpl class="LocationUpdateServlet"/>
<request>
<attribute id="new_coast" invalidate="coast"/>
</request>
</servlet>
```

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 if you use the variable to define a group id.
- Data_id and invalidate tags that are within the same <servlet> element should have different values. 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 occurs in different <servlet> elements.

6.6.1: Administering applications (overview)

Formally, an enterprise application can contain the following resources:

- Web applications
- Web resources

Therefore, the WebSphere administrator must represent servlets, JSP files, and HTML files as "Web applications" and "Web resources" in order to add them to enterprise applications:

- Web resources specify served paths for HTML files served by a Web server, allowing you to secure the paths
- Web applications are groups of Web resources, JSP files, and servlets sharing a servlet context

When Web applications alone are sufficient

Suppose a developer application contains only a servlet and an HTML file, or perhaps just a JSP file. The WebSphere administrator has gathered these resources into a Web application.

The administrator might then wonder if it is necessary to add such a Web application to an enterprise application. Because the Web application seems complete by itself, why perform the extra step of configuring an enterprise application that contains only the Web application?

The WebSphere administrator should include a Web application, or any resource, in an enterprise application when he or she needs to control access to the Web application as a whole, in addition to controlling access to its individual resources.

Imagine a developer application comprised of multiple Web applications. In such a case, an enterprise application is required to give an identity to the group of Web applications. Without its own administrative identity, the enterprise application cannot be secured as a whole.

The enterprise application also allows the group of resources in it to be managed (for example, to be made available to users, or stopped) as a unit.

Enterprise applications transcend hosts

Unlike the resources they contain, enterprise applications transcend particular application servers and host machines.

The same resource can belong to multiple enterprise applications. It can also exist independently of an enterprise application. For example, by including the Web application in the enterprise application, **MyServlet** can exist alone, as part of a Web application *and* as part of an enterprise application.

In any case, a resource must already exist in the administrative domain before being added to an enterprise application. For most resource types, when adding the resource to the administrative domain, the administrator must associate the resource with a particular application server and host machine.

The administrative task for creating an enterprise application lets you build applications from resources existing before you started the task. You can also add new resources to the administrative domain as you create the enterprise application to which the resources will belong.

To perform the above activities, the administrator can choose from among a few WebSphere Application Server administrative clients. Though the clients differ somewhat in terminology and the scope of activities they support, the concepts and procedures are the same as described above. See the Related information for considerations and instructions specific to each client.

6.6.1.1: Administering enterprise applications with the Java administrative console

The Type tree contains an Enterprise Applications folder object.

The Topology tree can contain one or more existing applications (the WebSphere "AdminApplication" is there by default). Their names vary; they are supplied by the administrator.

Use the View menu on the console menu bar to toggle between tree views.

Any task wizards for manipulating this resource?

On the console menu bar:

Console -> Task -> Create enterprise application

Console -> Task -> Edit enterprise application

6.6.1.1.1: Configuring new enterprise applications

To configure a new enterprise application, click Console -> Tasks -> Create Enterprise Application. The Console menu is located on the WebSphereAdministrative Console (Java console) menu bar.

The task is straightforward. After naming the new application, the administrator can select which resources will belong to the application.

In an enterprise application, some resource types are represented indirectly by Web resources. See the Related information for details.

6.6.1.1.5: Editing the contents of enterprise applications

To edit the contents of an enterprise application, click Console -> Tasks ->Edit Enterprise Application. The Console menu is located on the WebSphereAdministrative Console (Java console) menu bar.

The task is straightforward. After selecting the application to edit, the administrator can specify to add contents to, or delete contents from, the application.

6.6.2: Administering nodes (overview)

Administrative server node configurations provide information about machines (physical hosts) involved in the IBM WebSphere Application Server environment.

For each administrative server node, the administrator specifies properties such as:

- Administrative name, host name, and host system type
- WebSphere Application Server product installation root
- Process ID of the administrative server on the node

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).

A node can be in the "running" state, which indicates not only that the physical machine and operating system are running, but that the WebSphere administrative server on the machine is also running.

The administrative server must be running in order for the node to be active and operational in the administrative domain.

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-ins is a task associated with the administrative node.

6.6.2.1: Administering nodes with the Java administrative console

This article extends article 6.6.2 (the overview of administering nodes) with information specific to the Java console.

The table answers the most basic questions. See the [Related information](#) for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	The Type tree contains a Nodes folder object. The Topology tree can contain one or more existing nodes. Their names vary; they correspond to machine host names. Use the View menu on the console menu bar to toggle between tree views.
Any task wizards for manipulating this resource?	No

6.6.2.1.1: Configuring new nodes with the Java administrative console

There is no administrative task for configuring a new node. The product automatically displays a node in the Topology tree when:

- An IBM WebSphere Application Server administrative server is installed on the node
- The administrative server is configured to use the same administrative database as other nodes in the domain displayed by the Topology tree

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:

Select the node from the **Resources** section of the tree.

 Creations and changes made with this console are not applied to the administrative domain until you Commit them. See section 6.6.0.3.5 for details.

6.6.2.4: Property files pertaining to nodes

Two files are associated with node properties:

- *admin.config*
- *SERunner.properties*

The *admin.config* file is located in directory,

<WebSphere/Appserver>/bin

and the *SERunner.properties* file is located in directory,

<WebSphere/Appserver>/properties

The following entries in the [admin.config](#) file apply to nodes:

com.ibm.ejs.sm.adminServer.classpath	location of WebSphere Application Server libraries and files
com.ibm.ejs.sm.util.process.Nanny.path	location of executables in WebSphere Application Server, HTTP server, and database server
com.ibm.ejs.sm.adminServer.bootstrapPort	starts the Bootstrap Server on default port 900
server.root	main path of WebSphere Application Server
com.ibm.ws.jdk.path	location of JDK™ used by WebSphere Application Server

Also review these two entries in *SERunner.properties* file:

lsdPort	Admin Server location service daemon port (default port is 9000)
httpPort	HTTP Server login port

See the Related information for a description of nodes and their properties.

6.6.2.4.1: Configuring multiple administrative servers on a node

On the AS/400 system, multiple WebSphere administrative servers can be run on a single node. (Such a configuration might be possible on other operating systems, but is not formally tested or supported).

The following administrative server properties are provided in the `administrativeserver` configuration file to support running multiple administrative servers on the same machine. The `admin.config` file is located in the `bin` directory of the WebSphere Application Server installation root.

In the `admin.config` file, add the properties and their values as arguments on the `com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs` property. Add them using standard Java command line syntax:

```
-D property_name=property_value
```

For example:

```
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-ms32m -mx128m -D server.root=property_value
```

where *property_value* is the value you would like `server.root` to have.

server.root

The default product installation directory. Used to locate properties files, servlets, and other WebSphere resources.

com.ibm.websphere.servlet.admin.config.file

Override to specify the full path, including filename, to the `admin.config` file, instead of using `server.root/bin`

com.ibm.websphere.servlet.initialEJSSetup.file

Override to specify the full path, including filename, to the `initialEJSSetup.xml` file, instead of using the `server.root/properties` directory. Used to create the initial system configuration.

com.ibm.websphere.servlet.default.servlet.engine.file

Override to specify the full path, including filename, to the `default.servlet_engine` file, instead of using the `server.root/properties` directory. Used to create the initial system configuration.

com.ibm.websphere.servlet.webapp.root.directory

Override to specify the directory containing the webapp configuration tree, instead of using `server.root/hosts`. Used to create the initial system configuration.

com.ibm.websphere.servlet.admin.bootstrap.file

Override to specify the full path, including filename, to the `bootstrap.properties` file, instead of using the `server.root/properties` directory. The load order for the plug-in configuration used by the admin server is now:

1. This property
2. The root of the classloader (for example, a path on the classpath)
3. The `server.root/properties` directory

All configuration for the plug-in used in the administrative server is acquired from `list` file and has no dependency on `server.root`.

The `queues`, `rules`, and `vhosts` property files are written to the `ose.tmp.dir` location specified in `bootstrap.properties`.

An additional property can be set from the WebSphere Administrative Console. Add it as a command line argument of the target application server, again using the standard Java command line argument notation `-D property_name=property_value`:

default.client.encoding

The default language encoding to be used by the servlet engine for writing client data.

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

Application Server Name

Specifies a name for the application server. The name must be unique within the node (physical machine) containing the application server.

Command line arguments

Specifies 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 typing
java

at a command prompt. In case that is not convenient, check this [summary list of the arguments](#).

You might also add a [ServiceInitializer parameter](#) or [set the server socket queue depth using com.ibm.CORBA.ServerSocketQueueDepth](#), as described in the administrative server properties.

Current state

Indicates the state the application server is currently in. The next time it is started, it will try to change to its desired state setting.

Debug enabled

Specifies whether the server is running in debugging mode.

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.

Selecting this setting is necessary, but not sufficient, for using the Debugger to debug code running on this application server. You must also select the "Object Level Trace enabled" setting and perform some other steps. See the full InfoCenter for more information.

Desired state

Indicates the state the application server should have the next time it is started.

Environment

Specifies environment variables, and their values, to be used by the application server.

To set variables, click the Environment field to display a dialog box. In the box, enter variable names and values, clicking the Add button after each one.

The value of this property is a list of strings of the form "name=value"

EPM Specification

Executable in use

Indicates the executable (Java class) representing the application server.

Group ID

Specifies the name of the operating system group under which to run the server.

Note that the operating system group must exist on the machine where the server is to run before the server is started. This group must be assigned the necessary operating system privileges for performing operations such as creating output files on the local file system. [Additional information specific to operating system](#)

Group ID in use

Indicates the group ID now in use by the server. [Additional information specific to operating system](#)

Maximum startup attempts

Specifies the number of times to try to start the server before discontinuing attempts.

- Legal Values: Any positive integer

Name

Indicates a name for the application server. The name must be unique within the node (physical machine) containing the application server.

Node

Indicates the administrative node on which the application server resides.

Object Level Tracing enabled

Specifies whether Object Level Trace (OLT) and the Distributed Debugger are enabled.

If this option is deselected, this server will not send any trace or debug information to the OLT and the Debugger client interfaces.

Selecting this setting is necessary, but not sufficient, for using OLT/the Debugger to trace or debug code running on this application server.

Parent

Specifies the node on which the application server will reside. Though it is not indicated as such, this property is required.

Ping initial timeout

Specifies the maximum time in seconds after the server starts before a successful ping must occur. After this time elapses, the administrative server attempts to restart the server.

Ping interval

Specifies the frequency of communication attempts between the server and the administrative server to ensure that the server is running.

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 and thus reduces system overhead.

Ping timeout

Specifies the maximum amount of time in seconds that can elapse after the last successful ping before the administrative server assumes the server has failed.

Adjust this value based on your requirements for restarting failed servers. Decreasing the value shortens the length of time that a server can be down before any attempt to restart it.

Process ID

Indicates the operating system process ID of the server.

Process priority

Specifies the operating system process priority under which to run the server. The lower the number, the greater the importance of the process. [Additional information specific to operating system](#)

- Legal Values: Any positive integer

Process priority in use

Indicates the current process priority of the server. [Additional information specific to operating system](#)

Security enabled

Specifies whether to enable WebSphere Application Server security for the application server.

Security enabled in use

Specifies whether WebSphere Application Server security is currently enabled for the application server.

Standard error

Specifies the standard error stream for the operating system.

If the value of this property is set to the null string (""), the stream is set to the null device.

If this property is set to a relative path name, the path is relative to the server's working directory. Any class of trace output can be redirected to this file. By default, the output of the fatal, error, and audit trace classes is sent to this file.

[Additional information specific to operating system](#)

- Default: The file stderr.txt in the server's working directory

Standard error in use

Indicates the standard error stream now in use by the server.

[Additional information specific to operating system](#)

Standard input

Specifies the standard input stream for the operating system.

If this property is set to the null string (""), the stream is set to the null device.

If this property is set to a relative path name, the path is relative to the server's working directory.

[Additional information specific to operating system](#)

Standard input in use

Indicates the standard input stream now in use by the server.

[Additional information specific to operating system](#)

Standard output

Specifies the standard output stream for the operating system.

If this property is set to the null string (""), the stream is set to the null device.

If this property is set to a relative path name, the path is relative to the server's working directory.

[Additional information specific to operating system](#)

- Default: The file stdout.txt in the server's working directory.

Standard output in use

Indicates the standard output stream now in use by the server.

[Additional information specific to operating system](#)

Start Time

Indicates the time at which the server was most recently started.

State

Indicates the state the application server is currently in.

Thread Pool Size

Specifies the starting size of the thread pool for the application server.

Each application server has its own thread pool or cache from which it uses threads to process remote method invocations.

The size of a server's thread pool varies throughout the server's lifetime. Threads are created when needed and destroyed when there are too many idle threads.

Trace specification

Specifies the initial trace mask value to use for the server.

A trace mask defines which components are to be traced and allows you control how much trace information is generated.

The format of this property (and tracing in general) is described in the [trace properties help](#).

Trace specification in use

Indicates the trace specification now in use by the application server.

Trace output file

Specifies the file to which to write tracing information.

Trace output file in use

Indicates the trace output file currently in use.

Transaction inactivity timeout

Specifies the number of milliseconds a transaction can remain inactive before it is aborted.

Transaction timeout

Specifies the number of seconds to allow a transaction to proceed before aborting it because it is taking too much time.

Umask

Specifies an octal value that sets the operating system's file creation mask for the server.

The file creation mask specifies permissions that cannot initially be granted for new files. When a new file is created, the system refuses to grant the permissions specified in the file creation mask.

For example, a mask of 022 prevents the granting of the write permission to members of the group that own the file and to all other users who do not own the file.

A mask of 022 allows the owner to be granted all permissions; in other words, it leaves the owner's permissions the way the system specifies them.

If the system would otherwise create a file with privilege values of 777 (read, write, and execute permissions for owner, group, and other), a file creation mask of 022 causes it to create the file with 755 (all permissions for the owner, but only read and execute permissions for group and other).

- Legal Values: An integer in the octal range 0 through 0777

Umask in use

Indicates the mask now in use by the server.

User ID

Specifies 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.

- Legal Values: A string of 8 or fewer characters

- Default: The ID used by the administrative server

User ID in use

The user ID now in use by the server.

Working directory

Specifies 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.

6.6.3.1: Administering application servers with the Java administrative console

This article extends article 6.6.3 (the overview of administering application servers) with information specific to the Java console.

The table answers the most basic questions. See the [Related information](#) for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains an Application Servers folder object.</p> <p>The Topology tree can contain zero or more existing application servers. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	<p>On the console menu bar:</p> <p>Console -> Task -> Create application server</p>

6.6.3.1.1: Configuring new application servers

The product offers several ways to configure new application servers:

- By clicking Console -> Tasks -> Create Application Server from the console menu bar.
- By clicking Create Application Server from the drop-down list on the Wizards toolbar button.
- Using menus on resources in the Topology and Type trees (see Related information)

The first two methods lead to the Create Application Server task wizard, for which detailed help is provided here.

1. Follow the wizard instructions. On the first page, specify whether the application server will support Web applications.

If you do not select Web applications, the wizard will lead you through only the configuration of the application server.

2. Click Next to proceed. Specify application server properties. Only the name by which you will administer the server is required.
3. Click Next to proceed. Specify whether to start the server automatically after you finish this configuration wizard.
4. Click Next to proceed. Specify the node (physical machine) on which the application server will reside.
5. Click Next to proceed to the virtual host page of the wizard. Specify the virtual host for this application server's files, or specify a new virtual host to be configured.

[View virtual host properties help](#)

6. If you specified that this application server will support servlets, you will now begin specifying configuration information about the servlet support.

Click "Next" to proceed through the wizard pages as you specify a servlet engine and other servlet engine properties that are optional because they will use the default values if you do not customize them.

[View servlet engine properties help](#)

[View servlet properties help](#)

7. Click Next to proceed to Web application properties.
 - Specify a name by which to administer the Web application.
 - Optionally, describe the Web application.
 - Specify the virtual host part of the Web application's served path. That is, what host name (or its aliases) will users specify when they access the Web application from a Web browser?
 - Specify the Web Path for the Web application. That is, what should users type in after the host name when requesting this Web application?

For example, if you would like users to type

```
http://default_host_alias/webapp/mywebapp
```

to access the application (where *default_host_alias* is any valid alias for the default virtual host), specify:

- Virtual Host = "default_host"
- Web Application Web Path = "/webapp/mywebapp"
- Specify the document root for the Web application. This is the fully qualified path to where the

HTML and JSP files for the Web application will be found.

- Specify the classpath, adding either a directory for servlets or specifying servlets individually. Also specify any other resources the Web application needs to know about in order to operate correctly.

Note that both the document root and the classpath contain default values. You can accept the default values and then move your files there after finishing the task. Alternatively, you can change the default values to point to your files in their present locations, or a location to which you plan to move them.

- Specify other Web application properties or accept the default values for them.
- Enable the file servlet if you plan to drop servlets "anonymously" into a specified servlet directory in the Web application classpath.

Alternatively, you can configure each servlet so that it is known explicitly by name in the administrative domain.

- Specify to serve servlets by classname if you plan to drop servlets into a servlet directory without configuring them explicitly in the domain.
- Specify the JavaServer Pages (JSP) specification level to use. If you anticipate adding JSP files to the Web application later, make sure you select the anticipated JSP level. If your Web application will never contain JSP files, this value does not matter.

[View Web application properties help](#)

8. Click Finished.

6.6.3.1.2: Starting and stopping application servers with the Java administrative console

1. Make sure the administrative server is running.
2. [Open the Java administrative console](#).
3. From the console menu bar, click View -> Topology.
4. In the topology tree, expand the **WebSphere Administrative Domain** tree node to display a list of host names.
5. Expand the host name corresponding to the machine containing the application server that you want to start.
6. Click the application server to display its properties on the right side of the console.

If its current state is *Stopped*, right-click the application server. On the resulting menu, click Start.

If the administrative console appears to stop, there may be a problem with data access that the administrative console cannot properly return back to you. This happens when the archive file that contains database exception classes has not been added to the classpath in `adminclient.bat/.sh`. If you experience this behavior, make sure the archive file has been added to the classpath setting. (For example, the name of the file that contains DB2 exception classes is `db2java.zip`.)

7. After an information dialog is displayed, stating that the application server is running, click OK on the dialog.

After the application server starts, it is marked in the configuration database that it should be running. If it stops, or if you reboot the machine, the administrative server will automatically restart it. Even if the administrative server fails, the application server will continue to run.

8. Test the application server. Start the Web server if it is not running. Then, open a browser and go to:

```
http://localhost/servlet/snoop
```

Snoop is a standard sample servlet installed by default. In the browser, you should see information on `/servlet/snoop`, including Init Parameters and Request information.

To stop the application server, right-click it to display a menu. From the menu, select Stop.

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.

Work with objects of this type by locating them in the tree on the left side of the console:

Click **Tasks** -> **Create Objects** -> **Create Server**.

Also, existing application servers in the administrative domain are displayed in the **Resources** section of the navigation tree.

 Creations and changes made with this console are not applied to the administrative domain until you Commit them. Refer to section 6.6.0.3.5 for details.

6.6.3.4: Property files pertaining to application servers

The application server properties are in file:

- *sas.server.props*

This file is located in directory:

<WebSphere/Appserver>/properties

The following entries in the *sas.server.props* file are used to administer application servers:

com.ibm.CORBA.loginUserId	associates the application server with an authorized system user ID
com.ibm.CORBA.loginPassword	password for authorized system user ID Note: Ensure this file is located on a secure, firewall-protected server, to prevent unauthorized access.
com.ibm.CORBA.principalName	associates application server with node name and system user ID
com.ibm.CORBA.SSLKeyRing	enables security for the server
com.ibm.CORBA.SSLV3SessionTimeout	defines the secured session timeout
com.ibm.CORBA.LoginTimeout	defines a non-secured session timeout
com.ibm.CORBA.LoginSource	specifies the location of the runtime environment and policies

See the Related information for a description of applicationservers and their properties.

6.6.3.5: Invoking your own classes during application server startup and shutdown

The `ServiceInitializer` enables you to specify user defined classes that are invoked as the last action (or actions) during application server startup and shutdown.

Creating classes for use with `ServiceInitializer`

`ServiceInitializer` classes must have a no-argument constructor (for `newInstance()`) and must implement the following interface:

```
import javax.naming.Context; public interface ServiceInitializer { public void initialize(Context initialNamingContext) throws Exception; public void terminate(Context initialNamingContext) throws Exception; }
```

Adding `ServiceInitializer` classes to an application server

Specify your classes as part of the `ServiceInitializer` [command line argument for an application server](#), such as:

```
-Dcom.ibm.ejs.sm.server.ServiceInitializer=<class>[,<class>]...
```

where each `<class>` is one of your own classes.

`ServiceInitializer` classes are initialized in the order listed in the property, and are terminated in reverse order.

6.6.7: Administering servlet engines (overview)

A servlet engine configuration provides information about the applicationserver component that handles servlet requests forwarded bythe Web server. The administratorspecifies servlet engine properties including:

- Application server on which the servlet engine runs
- Number and type of connections between the Web server and servlet engine
- Port on which the servlet engine listens

6.6.7.0: Servlet engine properties

Application Server

Specifies the application server with which to associate the servlet engine.

Current State

Indicates the state the servlet engine is currently in. The next time the servlet engine is started, it will try to change to its desired state setting.

Desired state

Indicates the state the servlet engine should have the next time it is started.

Max Connections

Specifies the maximum number of concurrent resource requests to allow.

Max Connections in use

Specifies the Max Connections value currently in use.

Port

Specifies the port the servlet engine will listen on for servlet requests from the Web server.

Port in use

Specifies Port value currently in use.

Queue Type (Transport Type)

Specifies the connectivity type for communication between Web servers and application servers to obtain servlet requests:

OSE	For routing requests locally.
HTTP	Not recommended at this time
None	

If you specify OSE, specify these properties related to Queue Type.

Clone Index

Specifies a unique numerical identifier for this servlet engine instance.

Native Log File

Specifies the log file that will be considered "standard out" for tracing and debugging of the native code of the product. Specify either:

- A file name, with [product_installation_root](#)/logs assumed to be the directory
- A fully qualified path to a log file

Queue Name

Specifies the name of the queue for holding requests to be processed by the servlet engine.

Select Log File Mask

Specifies one or more levels of messages to log -- error, warning, informational, or trace.

Transport Type

Specifies the communication protocol type to use with the OSE transport:

- Local pipes
- INET sockets
- JAVA TCP/IP (not currently supported)

See the servlet engine tuning section of the [Tuning Guide](#) for suggested values, typically based on the operating system.

Queue Type in use

Indicates the queue type currently in use (see Queue Type description above).

Servlet Engine Mode ^{FP} 3.5.2

Specifies how servlets will be supported (in terms of how the specification levels are enforced). Consider the implications carefully before changing this setting. See [article 3.3.2a](#) for a discussion and details.

If you switch the servlet engine to full compliance mode, adjust the [Servlet Web Path Lists](#) of servlets running in this servlet engine, to keep your Web applications from breaking. Add a /* to the end of each Web path.

For example, if the path for a servlet is:

```
default_host/WebSphereSamples/servlet
```

then change it to:

```
default_host/WebSphereSamples/servlet/*
```

Servlet Engine Name

Specifies a servlet engine name. The name must be unique in the scope of the application server. In other words, you can create two servlet engines with the same name as long as each servlet engine is associated with a different application server.

Start Time

Indicates the time at which the servlet engine was most recently started. A value of "--" indicates the servlet engine has not been started since the administrative server started.

6.6.7.1: Administering servlet engines with the Java administrative console

This article extends article 6.6.7 (the overview of administering servlet engines) with information specific to the Java console.

The table answers the most basic questions. See the [Related information](#) for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains a Servlet Engines folder object.</p> <p>The Topology tree can contain zero or more existing servlet engines. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	<p>On the console menu bar:</p> <p>Console -> Task -> Create a servlet engine</p>

6.6.7.1.1: Configuring new servlet engines with the Java administrative console

The product offers several ways to configure new servlet engines:

- By clicking Console -> Tasks -> Create a servlet engine from the console menu bar.
- By clicking Create a servlet engine from the drop-down list on the Wizards toolbar button.
- Using menus on resources in the Topology and Type trees (see Related information)

The first two methods lead to the Create a servlet engine task wizard, for which detailed help is provided here.

1. Follow the wizard instructions.
 - Specify a name by which to manage the servlet engine.
 - Specify the application server to contain the servlet engine.

Click Next to proceed.

2. Specify servlet engine properties.
3. Click Finish.

6.6.7.3: Administering servlet engines with the Web console

Use the Web console to edit the configurations of servlet engines, which are responsible for providing needed services to running Web modules and their contained servlets and JSP files. Each application server runtime has one logical servlet engine, 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:

Click **Tasks** -> **Create Objects** -> **Create Servlet Engine**

When creating a servlet engine, you must specify an existing application server to contain it. Existing servlet engines and application servers in the administrative domain are displayed in the **Resources** section of the navigation tree.

 Creations and changes made with this console are not applied to the administrative domain until you Commit them. Refer to section 6.6.0.3.5 for details.

6.6.7.4: Property files pertaining to servlet engines

The servlet engine properties are in file:

- *servlet_engine.properties*

This file is located in directory:

`<WebSphere/Appserver>/properties`

6.6.8: Administering Web applications (overview)

The servlets and JavaServer Pages (JSP) files in a Web application share a servletcontext, meaning they share data and information about the execution environment, including a Web application classpath.

Approaches to configuring Web applications

There are two basic approaches to configuring Web applications. The administrator can configure a Web application:

- From the bottom up
- From the top down

To configure a Web application from the **bottom up**, the administrator can first explicitly configure the individual servlets that will eventually comprise the Web applications. When configuring a servlet, the administrator specifies the name and location of the servlet class file, and other information necessary for enabling the administrator to manage the servlet.

The administrator can combine one or more explicitly-defined servlets and Web resources into a Web application, allowing them to be managed as a logical unit (the Web application).

Because they are explicitly configured, the servlets can also be managed individually. For example, the administrator can unload a servlet from the Web application without causing the rest of the application to become unavailable to users.

The administrator can also configure a Web application from the **top down**. This technique might be familiar to an administrator who has used Web server products or IBM WebSphere Application Server Version 2.

Instead of explicitly defining each component (servlet, Web page, and so on), the administrator specifies the directories in which he or she plans to place the components of each type.

In the simplest case, each Web application has one directory for servlets and another for Web resources. Any servlet placed in the designated servlet directory becomes part of the Web application, and similarly any Web pages and JavaServer Pages (JSP) files are picked up from the designated Web resource directory.

Because the servlets are not explicitly defined, they cannot be managed or monitored individually.

Web applications inside enterprise applications

A Web application can be part of an enterprise application (an "application" for short). In the simplest case, an enterprise application is simply a "wrapper" for a Web application -- the files that comprise the application are exactly the same files that comprise the *Web* application.

In such a case, why bother to add a Web application to an enterprise application? An [enterprise application](#) help file discusses the benefits.

In a more complex case, an application might contain multiple Web applications and (in the case of IBM WebSphere Application Server *Advanced Edition*) some enterprise beans as well.

Configuring Web applications directly in WebSphere systems administration

The administrator should understand a few main settings as he or she configures Web applications:

- Classpath

Specifies where to find the servlets that belong to the application.

The classpath can specify a *directory* containing servlets, or can specify each servlet explicitly.

It can also specify the location of other files supporting the Web application.

- Document root

Specifies where to find the Web pages and JSP files belonging to the Web application.

- Web path

Combined with the virtual host, specifies what users will type in a Web browser to access the Web application.

The administrator can also specify properties such as:

- Servlet filtering parameters
- Affiliation with a virtual host
- Whether to reload servlets whose class files have changed
- Whether to temporarily suspend the Web application from use
- Servlet context attributes

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
```

FP 3.5.2 Version 3.5.2 introduces a Welcome Files setting, as described by the Servlet 2.2 specification. See the [Web application properties](#) for details.

FP 3.5.2 Converting WAR files

Version 3.5.2 (Fix Pack 2 applied to Version 3.5 base) introduces a new way to introduce Web applications into the WebSphere environment. The product now consumes and [converts WAR files](#) into WebSphere configurations.

Alternatively, you can continue to configure Web applications directly in WebSphere Application Server systems administration. The latter allows you to add WebSphereservlets to your Web applications to extend their functionality.

You can use either the [Java console \(WebSphere Administrative Console\)](#) or [command line programs](#) to convert WAR files.

Utilizing servlets available from WebSphere

See section 4.2.1.2.3 for information about adding complimentary WebSphere servlets to Web applications to provide functions such as JSP enablement, errorreporting, file serving, and the ability to invokeservlets by classname.

6.6.8.0: Web application properties

Attributes

Specifies servlet context attributes for the entire Web application.

- property Name - A servlet parameter of your choice
- property Value - The value associated with the property name

Note, the servlet context established by this property differs from the Shared Context, which pertains to clustering situations.

See the [JSP administration overview](#) for a description of attributes related to JSP reloading, available starting with Version 3.5.2.

Auto Reload

Specify whether to automatically reload servlets in the Web application when their classfiles change.

After specifying to Auto Reload, use the Reload Interval property to specify how often to check for updates.

Classpath

Specifies the classpath for the Web application.

Classpath in use

Specifies the classpath currently in use for the Web application.

Current State

Indicates the state the Web application is currently in. The next time it is started, it will try to change to its desired state setting.

Desired State

Indicates the state the Web application is in, according to the administrative server.

Description

Specifies a description of the Web application.

Document Root

Specifies the document root of the Web application.

Enabled

Indicates whether the servlet group (Web application) is available to handle requests.

Error Page ^{FP} 3.5.2 (changed)

Specifies mappings between error codes or exception types and the paths of resources in the Web application. Basically, defines what to display to the user in the event of a specific error.

Consists of:

- Status Code or Exception - An HTTP error code (such as 404) or fully qualified classname of a Java exception type
- Location - Location (in the Web application) of the error page to display when that status code or exception occurs

Example values:

- Status Code: 404

- Exception: java.lang.NullPointerException
- Location: /webapp/myapp/my404ErrorPage.jsp

The location is a "Web path," to use the terminology of the WebSphere Administrative Console.

Full Web Path in use

Specifies the URI by which the Web application can currently be located.

MIME Table ^{FP} 3.5.2

Specifies mappings between extensions and MIME types. Consists of:

- Extension - Text string describing an extension, such as .txt
- Type - The defined MIME type associated with the extension, such as text/plain

You can also specify MIME table parameters at the virtual host level, but the MIME table parameters you specify for a Web application take precedence (local scope).

Reload Interval

Specifies the interval between reloads of the web application.

Specify the value in *seconds*

6.6.8.1: Administering Web applications with the Java administrative console

This article extends article 6.6.8 (the overview of administering Web applications) with information specific to the Java console.

The table answers the most basic questions. See the Related information for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains a Web Applications folder object.</p> <p>The Topology tree can contain zero or more existing Web applications. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	<p>On the console menu bar:</p> <p>Console -> Task -> Configure a Web application</p> <p>There are also subtasks:</p> <ul style="list-style-type: none">● Add a servlet● Add a JSP file or Web resource● Add a JSP enabler

6.6.8.1.1: Configuring new Web applications

The product offers several ways to configure new Web applications:

- By clicking Console -> Tasks -> Configure a Web application from the console menu bar.
- By clicking Configure a Web application from the drop-down list on the Wizards toolbar button.
- Using menus on resources in the Topology and Type trees (see Related information)

The first two methods lead to the Configure a Web application task wizard, for which detailed help is provided here.

1. Follow the wizard instructions. On the first page, name the Web application and specify whether to add WebSphere "internal" servlets to the Web application to perform certain functions:
 - File servlet
 - Enable Serving Servlets by Classname (adds invoker servlet)
 - JSP enabler (adds the JSP processor servlets)
 - Chainer servlet

See [article 4.2.1.2.3](#) for a detailed description of each internal servlet.

2. Click Next to proceed. Specify the servlet engine on which the Web application should reside.
3. Click Next to proceed. Now:
 - Specify a name by which to administer the Web application.
 - Optionally, describe the Web application.
 - Specify the virtual host part of the Web application's served path. That is, what host name (or its aliases) will users specify when they access the Web application from a Web browser?
 - Specify the Web Path for the Web application. That is, what should users type in after the host name when requesting this Web application?

For example, if you would like users to type

```
http://default_host_alias/webapp/mywebapp
```

to access the application (where *default_host_alias* is any valid alias for the default virtual host), specify:

- Virtual Host = default_host_alias
- Web Application Web Path = /webapp/mywebapp

4. Click Next to proceed:
 - Specify the document root for the Web application. This is the fully qualified path to where the HTML and JSP files for the Web application will be found.
 - Specify the classpath, adding either a directory for servlets or specifying servlets individually. Also specify any other resources the Web application needs to know about in order to operate correctly.

Note that both the document root and the classpath contain default values. You can accept the default values and then move your files there after finishing the task. Alternatively, you can change the default values to point to your files in their present locations, or a location to which you plan to move them.

- Specify other Web application properties or accept the default values for them.
5. Click Finish.

6.6.8.1.6: Converting WAR files with the Java administrative console

To convert WAR files (see article 0.8.2 for a description) using the Javaconsole:

1. Select the Convert WAR File task from the console Tasks menu.
2. Follow the instructions in the task wizard.

You will need to specify the following information:

- The servlet engine where the Web application will reside
- A name for the Web application
- A Web Path for the Web application
- The path to the WAR file

6.6.8.3: Administering Web applications with the Web console

Use the Web console to edit the configurations of Web applications.

Work with objects of this type by locating them in the tree on the left side of the console:

Click **Tasks** -> **Create Objects** -> **Create Web Application**

When creating a Web application, you must specify an existing servlet engine to contain it. Existing Web applications and application servers in the administrative domain are displayed in the Resources section of the navigation tree.

 Creations and changes made with this console are not applied to the administrative domain until you Commit them. Refer to section 6.6.0.3.5 for details.

6.6.9: Administering servlets (overview)

The servlet developer will create a servlet class file and either provide the file to the administrator or notify the administrator of the file's location on a machine in the WebSphere administrative domain.

For an "explicitly configured servlet," the administrator specifies properties such as:

- The administrative name and actual class name
- The Web application to which the servlet belongs
- Servlet initialization parameters
- Whether to start the servlet in Java debug mode
- Web paths by which the servlet can be requested
- Whether to load the servlet when the application server starts, or wait until the servlet is requested

An "implicitly configured servlet" resides in a directory specified to hold servlets for a Web application, and can be invoked only if the Web application is enabled to serve by classname. This is accomplished by adding a certain [WebSphere internal servlet](#) to the Web application.

Approaches to administering servlets

The administrator can represent the servlet a few ways in the administrative domain:

- As an explicit servlet resource belonging to a Web application

The administrator can "explicitly" configure a servlet resource to represent and manage the servlet class file in the administrative domain.

The servlet must be associated with a Web application. The Web application classpath tells where to find the servlet class file in the WebSphere Application Server product directories.

- As an implicit part of a Web application

As previously stated, the Web application classpath contains the servlet class file location.

The classpath does not have to reference the exact servlet class file name. Instead, it can reference a servlet directory for the Web application. The servlet directory can contain several servlets whose particular class names are unknown to the administrative domain. The servlets need not exist as explicitly configured servlet resources.

To use this approach, you need to add a particular [WebSphere internal servlet](#) to the Web application, for serving servlets by their classnames.

- As part of an enterprise application

After a servlet is added to the administrative domain one way or the other, the administrator can add it to an enterprise application.

A servlet resource cannot be added to an enterprise application directly, but can be added as part of a Web application.

An explicitly configured servlet can be monitored, but an implicitly configured servlet cannot. If the administrator requires statistics such as requests and execution time, he or she should explicitly configure the servlet.

WebSphere Studio configuration files

Unless creating your servlets in the Studio product, disregard this section. It discusses the handling of .servlet files generated by the IBM WebSphereStudio product.

Whenever WebSphere Application Server Version 3 loads a servlet, it looks for an accompanying .servlet file specifying initialization parameters and configuration information for the servlet.

To be found, the .servlet file must reside in the same directory as the servlet. No further action is required to use .servlet files in WebSphere Application Server Version 3.

The WebSphere Administrative Console does not display .servlet files, even if they are present. However, it *will* display the servlet and its configuration if you define the servlet in the administrative domain.

If using the console to specify servlet settings, make sure they do not conflict with those defined in the .servlet file. In such a case, it is likely, but not definite, that the console settings will override the .servlet file settings.

6.6.9.0: Servlet properties

Current State

Indicates the state the servlet is currently in. The next time it is started, it will try to change to its desired state setting.

Debug Mode

Specifies whether to run the servlet code in debug mode.

Debug Mode in use

Indicates whether the servlet is now running in debug mode.

Description

Specifies a description of the servlet.

Desired State

Indicates the state the servlet is in, according to the administrative server.

Enabled

Indicates whether the servlet is available to handle requests.

Init Parameters

Specifies values for the parameters the servlet uses when it is initially loaded. The parameter names specified here must match the parameters known to the servlet class code.

Instead of a constructor, a servlet uses an `init()` method and a `ServletConfig` interface that specifies its initialization parameters. The parameter-value pairs you specify in Init Parameters are added to the servlet's `ServletConfig` object.

Init Parameters in use

The initial parameters (see above) now in use by the servlet.

Load at Startup

Specifies whether to load the servlet when the Web application containing it is started. If this value is false, the servlet will not be loaded until it is first requested.

Selecting to load at startup is a performance decision. It transfers the burden of the servlet load time to the Web application load time, instead of making the user wait for the servlet to load the first time it is requested.

 If one servlet in a Web application fails to load at startup, the entire Web application is considered invalid and cannot be accessed by users. The symptom is "Error 503: Application is currently unavailable for service."

Follow good practices to prevent the above condition:

- If a servlet is known to throw exceptions at startup, make sure it has a try-catch statement in its `init()` method to prevent its failure.
- If the servlet loads successfully, but you need to change its class file later, overwrite the class file. If you instead delete the class file and then add a new one, the Web application classloader will consider the servlet removed and declare the Web application invalid.

Load at Startup in use

Indicates whether the servlet is currently specified to load when the administrative server is started (see Load at Startup).

Servlet Class Name

Specifies the servlet class name. Specify the package, but do not include the *.class* extension. For example, `com.ibm.server.MyClassName` demonstrates a valid format.

Servlet Class Name in use

Indicates the name of the servlet class (see above) currently in use.

Servlet Name

Specifies a name for the servlet.

Servlet Web Path List

Specifies one or more Uniform Resource Identifiers (URIs) by which this servlet can be located and invoked.

This setting is affected by the servlet engine mode setting introduced with Version 3.5.2. See [the Servlet Engine Mode field help description](#) for details.

Servlet Web Path List in use

Indicates the servlet URI list (see above) now in use.

Start Time

Indicates the time at which the servlet was most recently loaded. A value of "--" indicates the servlet has not yet been loaded since the application server containing the servlet was started.

Web Application

Specifies the Web application with which the servlet is associated.

6.6.9.1: Administering servlets with the Java administrative console

This article extends article 6.6.9 (the overview of administering servlets) with information specific to the Java console.

The table answers the most basic questions. See the [Related information](#) for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains a Servlets folder object.</p> <p>The Topology tree can contain zero or more existing servlets. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	<p>On the console menu bar, there is a task for adding a servlet to a Web application:</p> <p>Console -> Task -> Add a Servlet</p>

6.6.9.1.1: Configuring new servlets with the Java administrative console

The product offers several ways to configure new servlets:

- By clicking Console -> Tasks -> Add a Servlet from the console menu bar.
- By clicking Add a Servlet from the drop-down list on the Wizards toolbar button.
- Using menus on resources in the Topology and Type trees (see Related information)

The first two methods lead to the Add a Servlet task wizard, for which detailed help is provided here.

1. Follow the wizard instructions. On the first page, specify whether you want to browse for servlet files or configure a servlet without really having its class files in place yet.

If you are adding a [WebSphere internal servlet](#) to a Web application, then specify that you do not want to browse for files.

2. Click Next to proceed. On the second page, specify the Web application to which the servlet will belong.
3. Click Next to proceed. On the third page, specify the type of servlet to configure.

If this is your own servlet code, specify Create User-Defined Servlet. Otherwise, specify to add one of the "internal servlets" that ship with WebSphere Application Server to support various functions:

- File servlet
- Enable Serving Servlets by Classname (adds invoker servlet)
- JSP enabler (adds the JSP processor servlets)
- Chainer servlet

See [article 4.2.1.2.3](#) for a detailed description of each internal servlet.

4. If you selected an internal servlet other than the Chainer servlet, you can finish the wizard now. Otherwise, click Next to proceed and specify:
 - A name by which to administer the servlet.
 - The Web application to which it will belong.
 - An optional description of the servlet.
 - The class name of the servlet.
 - The Web Paths by which the servlet can be accessed. For example, if the servlet can be accessed by `http://myhostname/myServlet`, specify `/myServlet`.

If you are configuring a Chainer servlet, name it, supply any initial parameters, and specify the Web paths by which it can be accessed (see above).

5. Click Next to proceed. Specify additional servlet properties or accept the default values.
6. Click Finish.

6.6.9.3: Administering servlets with the Web console

Use the Web console to edit the configurations of servlets.

Work with objects of this type by locating them in the tree on the left side of the console:

Click **Tasks** -> **Create Objects** -> **Create Servlet**

When creating a servlet, you must specify an existing Web application to contain it. Existing servlets and applicationservers in the administrative domain are displayed in the **Resources** section of the navigation tree.

 Creations and changes made with this console are not applied to the administrative domain until you Commit them. Refer to section 6.6.0.3.5 for details.

6.6.9.4: Property files pertaining to servlets

The servlets properties are in file:

- *servlet_engine.properties*

This file is located in directory:

<WebSphere/Appserver>/properties

See the Related information for a description of servlets and their properties.

6.6.10: Administering JSP files (overview)

The IBM WebSphere Application Server administrator should know the following about administering JSP files.

Enable JSP handling at the Web application level

- The ability of the product to serve JSP files is controlled at the Web application level. It is quite simple: If a Web application contains a JSP enabler servlet, the Web application can handle requests for JSP files.
- WebSphere Application Server provides the JSP enabler servlets. There is one for each supported JSP specification level. The "full" InfoCenter provides additional documentation about them.
- The administrator is responsible for permanently adding JSP enablers to Web applications requiring the ability to handle JSP requests:
 - A Web application can contain zero or one JSP enablers.
 - A Web application **cannot** contain more than one JSP enabler.
 - A Web application that does not need to serve JSP files can contain zero JSP enablers.

JSP-enabled Web applications look at all JSP requests

- By default, JSP enablers allow a Web application to consider all JSP requests (*.jsp) directed to the particular Web application.
- To restrict the attention of the Web application only to particular JSP files (instead of *.jsp), the administrator can remove the Web application Web path specifying *.jsp and replace it with Web paths specifying particular JSP files by name.

Place JSP files and configure Web applications to find them

- For the Web application to fulfill a JSP request, the requested JSP file must be in the document root of the Web application, or in a subdirectory of the document root, allowing the Web application to find it.
- If a JSP file depends on other files, such as servlets, JavaBeans, or the like, the files must reside in directories specified in the classpath setting of the Web application.

Alternatively, the resources can be specified in a more general classpath, such as that of the administrative server pertaining to the domain containing the Web application. See the Related information for details.

FP 3.5.2 JSP reloading for JSP 1.0 Web applications

Please note that the following parameters apply only to JSP 1.0 Web applications. They cannot be used for JSP 1.1 or JSP .91 Web applications.

You can configure how often each web application on the WebSphere Application Server will look for revised JSP files that need to be recompiled. By default, the Application Server will check for the presence of a class file every time a JSP is called, and compile the class file if one doesn't already exist. For performance purposes, you may wish to disable this check entirely or only after the JSP is called for the first time. Additionally, you may wish to configure the amount of time between checks.

Recall, To specify how you want this done for each Web application, specify these [Init Parameters](#) for the [JSP Enabler](#) servlet of the Web application:

1. **checkjspfiles**

This attribute has three possible values:

- "true" (or "always") - this will cause the Application Server to check for the presence of a class file each time the JSP is called or each X number of milliseconds specified in the *reloadinterval* below.
- "firsttime" - the Application Server will check for the presence of the class file only once:when it is called for the first time.
- "false" (or "never") - the Application Server will never check for the class file (you must ensure that a class file already exists for the JSP to work properly).

2. **reloadinterval**

The number of milliseconds between each check. This variable only works when you have selected "true" (or "always") as the value for *checkjspfiles* above.

6.6.10.0: JSP file properties

JSP files are not directly represented as resources in the Java console. (There is no JSP Files folder in the Type tree view). Therefore, they do not have property dialogs.

See the [help for JSP-related tasks](#) for JSP configuration information.

6.6.10.1: Administering JSP files with the Java administrative console

This article extends article 6.6.10 (the overview of administering JSP files) with information specific to the Java console.

The table answers the most basic questions. See the Related information for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	JSP files are represented indirectly in the tree views in two ways: <ul style="list-style-type: none">● Virtual hosts can contain Web resources representing JSP files (*.jsp and others)● Web applications that are enabled to handle JSP requests contain JSP enabler servlets, and their Web path configurations reflect the ability to serve *.jsp or specific JSP files
Any task wizards for manipulating this resource?	On the console menu bar: Console -> Task -> Add a JSP enabler (a WebSphere servlet to provide JSP .91 or 1.0 compiler support) Console -> Task -> Add a JSP file or Web resource (to a Web application)

6.6.10.1.1: Configuring new JSP files with the Java administrative console

1. Determine which Web application should serve the particular JSP file. Locate the Web application in the Topology tree view to verify that it is enabled to handle JSP requests:
 - If it contains the servlet with the class name JspServlet (the JSP 1.0 enabler) or PageCompileServlet (the JSP .91 enabler), it is already enabled to serve JSP files of the specified JSP level.

If it contains the JSP enabler servlet for one JSP specification level and the JSP file you want to add requires the other JSP specification level, then this Web application is not suitable to support the JSP file. Each Web application can support only one of the two supported JSP levels.
 - If it does not contain either JSP enabler servlet, add the servlet by completing the Add a JSP Enabler task available on the Console -> Tasks menu.
2. Place the JSP file in the document root specified in the Webapplication configuration settings, or in a subdirectory of the document root.
3. If the JSP file depends on other files that do not belong to the Web application, ensure the files are in the Web application classpath.

6.6.11: Administering HTTP session support (overview)

Each servlet engine automatically contains a single Session Manager. When configuring the Session Manager, the WebSphere administrator can:

- Enable sessions
- 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

 When moving from WebSphere ApplicationServer Version 3.0 or 3.01 to Versions 3.02 or 3.5, the administrator needs to migrate Session Manager settings pertaining to database persistence.

6.6.11.0: Session Manager properties

Allow Overflow

Specifies whether to allow the number of sessions in memory to exceed the value specified by Base In-memory Size property.

- True - Allow overflow
- False - Limit the number of sessions in memory to Base In-memory Size

Allow Overflow in use

Indicates the current value of the Allow Overflow property.

Base Memory Size

Specifies the number of sessions to maintain in memory. The meaning differs somewhat, 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 allow additional sessions to be stored in secondary tables.

For persistent sessions, this value specifies the size of the general cache. If the Cache property is enabled, the Base In-memory Size specifies how many session updates will be cached before the Session Manager reverts to reading session updates from the database automatically.

This value holds when you are using in-memory sessions, persistent sessions with caching, or persistent sessions with manual updates. (The manual update cache keeps the last n time stamps representing "last access" times, with n being the Base Memory Size value).

The default is 1000. How you customize this setting will depend on your hardware system, the usage characteristics of your site, and your willingness to increase the stack sizes of the Java processes for your Application Servers to accommodate a larger value.

Base Memory Size in use

Indicates the current value of the Base Memory Size property.

Cookie Comment

Specifies information about the cookie.

Cookie Domain

Specifies the value of the domain field of a session cookie. This value will restrict where the cookie is sent. For example, if you specify a particular domain, session cookies will be sent only to hosts in that domain.

Cookie Maximum Age

Specifies the maximum number of milliseconds the cookie will live on the client browser. Corresponds to the Time to Live (TTL) value described in the Cookie specification.

- Legal Values:
 - Positive integer indicates age in milliseconds
 - -1 indicates the cookie persists for the current browser session
- Default: -1

Cookie Name

Specifies a unique name for the cookie.

- Default: sessionid

Cookie Path

Specifies the value of the path field that will be sent for session cookies. Specify a value to restrict which paths on the server the cookie will be sent to. By restricting paths, you can keep the cookie from being sent to certain servlets, JHTML, and HTML files.

If you specify the root directory, the cookie will be sent whenever the given server is accessed.

- Legal Values:
 - Any string representing path on server
 - Blank indicates the root directory "/"

Cookie Secure

Specifies whether session cookies include the secure field. Specify Yes to restrict the exchange of cookies to only HTTPS sessions.

Current state

Indicates the state the session manager is currently in. The next time it is started, it will try to change to its desired state setting.

Data Source

Specifies the data source object from which the Session Manager will obtain database connections.

The data source represents a pool of database connections and a configuration for that pool (for example, the maximum number of connections to allow -- the pool size).

Data Source in use

Indicates the data source currently in use.

Desired state

Indicates the state the session manager should have the next time it is started.

Enable Cookies

Specifies whether session tracking uses cookies to carry session IDs.

- Yes - session tracking will recognize session IDs that arrive as cookies and tries to use cookies for sending session IDs
- No - session tracking will use URL rewriting instead of cookies

Enable Persistent Sessions

Specifies whether to save session data in a database, or lose the session data when the server shuts down.

- Yes - The session data is stored in a database for each session transaction
- No - The session data is kept in memory for a single instance of the servlet engine

For changes to take effect, this property requires stopping and restarting the application server with which the session manager is associated.

Enable Persistent Sessions in use

Indicates whether session data will be saved in a database, or will be lost the session data when the server shuts down.

Enable Protocol Switch Rewriting

Specifies whether the session ID is added to URLs when the URL requires a switch from HTTP to HTTPS or HTTPS to HTTP.

- Yes - The session ID is required in order to go between HTTP and HTTPS
- No - The session ID is not required in order to go between HTTP and HTTPS

Enable Sessions

Specifies whether session tracking is enabled, meaning the session-related methods for the request and response objects will be functional.

- True = The Session Manager is functional
- False = The Session Manager exists but is not functional

Enable Sessions in use

Indicates whether session tracking is currently enabled.

Enable URL Rewriting

Specifies whether the Session Manager uses rewritten URLs to carry the session IDs. If it is enabled, the Session Manager recognizes session IDs that arrive in the URL and, if necessary, rewrites the URL to send the session IDs.

HttpSessionRandIDGen

Specifies whether session IDs are randomly generated. Random ID generation can affect performance of the Session Manager. To disable random ID generation, set this property to false.

HttpSessionSecurity

Associates a user identity with a session. To disable association, set this property to false.

Invalidate Time

Specifies the number of seconds a session is allowed to go unused before it will no longer be considered valid.

Legal values:

- A positive integer represents the invalidate time in seconds
- -1 specifies that the session will not be invalidated

Default: 1800 seconds (30 minutes)

 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.

You should be aware of the relationship of this setting and the [Session Timeout setting of Web applications](#).

Number of connections

Specifies the maximum number of concurrent connections to establish with the database.

Number of connections in use

Indicates the maximum number of concurrent connections that will be permitted to be established with the database.

Password

Specifies the password for accessing the session database and tables.

Persistence Type

Specifies the mechanism to use for recording persistent session data.

Legal Values:

- Direct to database

Persistence Type in use

Indicates the mechanism that is currently in use for recording persistent session data.

Servlet Engine (or Servlet Engine Choices)

Indicates the servlet engine with which the Session Manager is associated.

Session Manager Name

Specifies a unique name for the Session Manager.

Start Time

Indicates the most recent time the Session Manager was started.

User ID

Specifies the user ID for accessing the session database and tables.

User ID in use

Indicates the user ID currently in use for accessing the data source for the user session.

Using Cache

Specifies whether to maintain an in-memory list of the most recently used sessions. In some cases, the list (cache) can help avoid database accesses.

- True - Maintain the list
- False - Do not maintain the list

This value applies only when persistent sessions are enabled and the Persistence Type is "direct to database."

Using Cache in use

Indicates whether an in-memory list of the most recently used sessions will be maintained.

Using Manual Update

Specifies whether to automatically send session updates to the database. By default, Application Server updates the database with the last access time and any changes effected by the servlet, such as updating or removing application data.

With manual update, Application Server caches the last update times and updates the database asynchronously prior to checks for session invalidation. For other changes, servlet writers perform the updates themselves, using the sync method provided as part of the WebSphere extension to `HttpSession`, `com.ibm.websphere.servlet.session.IBMSession`.

- False - Application Server automatically updates the database at the end of a `service()` method
- True - Servlets must call a sync method to update the database manually

This value applies only when persistent sessions are enabled and the Persistence Type is "direct to database."

Using Manual Update in use

Indicates whether manual update is currently in use.

Using Multirow Sessions

Specifies whether to place each instance of application data in a separate row in the database, allowing larger amounts of data to be stored per session. This can yield better performance in certain usage scenarios.

- True - Place each instance of application data in a separate row of the database
- False - Allow instances of application data to be placed in the same row

Using Multirow Sessions in use

Indicates whether multirow sessions are currently in use.

Using Native Access

Specifies whether to perform session persistence database updates using optimized JNI-based SQL access, written in the C programming language.

- True - Use JNI-based SQL written in C to access databases
- False - Use the JDBC default to access databases

This option is valid only if you are using IBM DB2 as your database.

Using Native Access in use

Indicates whether native access is currently in use.

6.6.11.1: Administering session support with the Java administrative console

Use the Java administrative console to administer Session Managers, which provides support for HTTP sessions.

This article extends article 6.6.11 (the overview of administering session support) with information specific to the Java console.

The table answers the most basic questions. See the Related information for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains a Session Managers folder object.</p> <p>The Topology tree can contain zero or more existing Session Managers. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	No

6.6.11.1.1: Configuring new Session managers

Use menus on resources in the Topology and Type tree to configure new Session Managers (see Related information for instructions).

6.6.12: Administering user profile support (overview)

Each servlet engine automatically contains a single User Profile Manager. When configuring the User Profile Manager, the administrator can specify:

- The servlet engine hosting the User Profile Manager
- An administrative name for the User Profile Manager
- A user ID and password for accessing the user profile database and table
- The "data wrapper" class implementing the 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 a fair amount of knowledge about the classes implementing user profile support, particularly if the implementation involves enterprise beans (Advanced Edition only).

If he or she is not familiar with the implementation, the administrator should obtain the information from the application developer.

6.6.12.0: User Profile Manager properties

Current state

Indicates the state the user profile is currently in. The next time it is started, it will try to change to its desired state setting.

Desired state

Specifies the state to which the user profile will try to change the next time it is started.

Data Source

Specifies the data source object from which the User Profile Manager will obtain database connections.

The data source represents a pool of database connections and a configuration for that pool (for example, the maximum number of connections to allow --the pool size).

Data Source in use

Indicates the data source currently in use.

Data Wrapper Class

Specifies the class implementing the User Profile. By default, this is the User Profile base implementation provided by Application Server Version 3.x:

- Default: `com.ibm.servlet.personalization.userprofile.UserProfile`

Data Wrapper Class in use

Indicates the Data Wrapper Class currently in use.

Desired state

Indicates the state the user profile should have the next time it is started.

Enable User Profile

Specifies whether to enable user profiles. After creating a user profile, change the value of this property from No to Yes to enable the user profile.

Enable User Profile in use

Indicates whether user profiles are currently in use.

Password

Specifies the password for accessing the userprofile database and tables. This is used only in userprofile base enterprise beans implementation. If userprofile enterprise beans are extended, specify the database parameters for extended bean at container level.

Servlet Engine

Indicates the servlet engine with which the user profile is associated.

Servlet Engine Choices

Specifies the servlet engine with which the user profile can be associated.

Start Time

Indicates the time at which the object representing the user profile in the administrative server was most recently started.

User ID

Specifies the user ID for accessing the userprofile database and tables. This is used only in userprofile base enterprise beans implementation. If userprofile enterprise beans are extended, specify the database parameters for extended bean at container level.

User ID in use

Indicates the user ID currently in use for accessing the data source for the user profile.

User Profile Name

Specifies a name for the user profile. The name must be unique within the administrative domain containing the user profile.

6.6.12.1: Administering user profile support with the Java administrative console

This article extends article 6.6.12 (the overview of administering user profile support) with information specific to the Java console.

The table answers the most basic questions. See the [Related information](#) for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains a User Profile Managers folder object.</p> <p>The Topology tree can contain zero or more existing User Profile Managers. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	No

6.6.12.1.1: Configuring new User Profile Managers with the Java administrative console

Use menus on resources in the Topology and Type tree to configure new User Profile Managers (see Related information for instructions).

6.6.14: Administering database connections (overview)

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 Drivers and data sources are used by Java components requiring database access, such as servlets.

Procedure for introducing JDBC drivers

To introduce JDBC drivers into the WebSphere Application Server environment:

1. Create a WebSphere JDBC driver configuration that specifies where to find the driver code on the physical machine.
2. Create a WebSphere data source configuration.
3. Perform an administrative "installation" of the JDBC driver into WebSphere.

Use the administrative client of your choice to perform the above steps (see [Related information](#)). The [Create Data Source task wizard](#) in the [Java administrative console](#) is recommended because it leads you through all of the above steps.

6.6.14.0: Properties of JDBC drivers: WebSphere Application Server

Description

A description of the driver, for your administrative records.

Driver Implementation Class, Implementation Class

Specifies the name of the Java class that implements the driver.

Jar file, Classpath

Specifies 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.

JTA Enabled

Specifies whether the driver can handle Java-based two-phase commit transactions. JTA is a transaction API for Java applications.

If **not** performing distributed transactions, set this value to False.

Note that for Application Server Version 3.0x, the URL Prefix setting had to contain "jta," in addition to selecting JTA Enabled. For Version 3.5, selecting JTA Enabled is sufficient. The URL Prefix value is independent of whether JTA is enabled.

Name

Specifies 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

Specifies the node (machine) on which to install the driver.

URL prefix

Specifies 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"> ● DB2: $\\$DB2_HOME/java12/db2java.zip$ or $\\$DB2_HOME/java/db2java.zip$ ● Oracle: $\\$ORACLE_HOME/jdbc/lib/classes12.zip$ ● Sybase: $sybase_install_root/jConnect-5_2/classes/jconn2.jar$ ● InstantDB: product_installation_root/lib/idb.jar ^{FP} 3.5.3 Merant SequelLink: <ul style="list-style-type: none"> ○ Complimentary copies with WebSphere Application Server: <ul style="list-style-type: none"> ■ product_installation_root/lib/sljc.jar ■ product_installation_root/lib/sljcx.jar
HP-UX	<ul style="list-style-type: none"> ● DB2: $\\$DB2_HOME/java12/db2java.zip$ or $\\$DB2_HOME/java/db2java.zip$ ● Oracle: $\\$ORACLE_HOME/jdbc/lib/classes12.zip$ ● InstantDB: product_installation_root/lib/idb.jar ● ^{FP} 3.5.3 Merant SequelLink: <ul style="list-style-type: none"> ○ Complimentary copies with WebSphere Application Server: <ul style="list-style-type: none"> ■ product_installation_root/lib/sljc.jar ■ product_installation_root/lib/sljcx.jar
Linux (Intel)	<ul style="list-style-type: none"> ● DB2: $\\$DB2_HOME/java12/db2java.zip$ or $\\$DB2_HOME/java/db2java.zip$ ● Oracle: $\\$ORACLE_HOME/jdbc/lib/classes12.zip$ ● InstantDB: product_installation_root/lib/idb.jar
Linux on S/390	<ul style="list-style-type: none"> ● Oracle: $\\$ORACLE_HOME/jdbc/lib/classes12.zip$ ● InstantDB: product_installation_root/lib/idb.jar
NetWare	<ul style="list-style-type: none"> ● DB2: $\\$DB2_HOME/lib/db2java.zip$ ● Oracle: $\\$ORACLE_HOME/jdbc/lib/classes12.zip$ ● InstantDB: product_installation_root/lib/idb.jar
Solaris	<ul style="list-style-type: none"> ● DB2: $\\$DB2_HOME/java12/db2java.zip$ or $\\$DB2_HOME/java/db2java.zip$ ● Oracle: $\\$ORACLE_HOME/jdbc/lib/classes12.zip$ ● Sybase: $sybase_install_root/jConnect-5_2/classes/jconn2.jar$ ● InstantDB: product_installation_root/lib/idb.jar ● ^{FP} 3.5.3 Merant SequelLink: <ul style="list-style-type: none"> ○ Complimentary copies with WebSphere Application Server: <ul style="list-style-type: none"> ■ product_installation_root/lib/sljc.jar ■ product_installation_root/lib/sljcx.jar

Windows

- DB2: C:\SQLLIB\java\db2java.zip
- Oracle: C:\Oracle\Ora81\jdbc\lib\classes12.zip
- InstantDB: *product_installation_root*/lib/idb.jar

- ^{FP} 3.5.3

Merant SequelLink:

- Complimentary copies with WebSphere Application Server:
 - *product_installation_root*\lib\sljc.jar
 - *product_installation_root*\lib\sljcx.jar

Note that the same data source implementation class is used for both non JTA and JTA enabled datasources. Also, this class is found in the sljcx.jar, not the sljc.jar.

6.6.14.0.1: Properties of data sources

Connection timeout

Specifies 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.

Database Name

Specifies the name of the database used to store entity bean data.

For example, enter WAS. This would make your data source point to jdbc:db2:WAS.

Driver

Specifies the name of the JDBC driver that this data source is using.

Idle timeout

Specifies 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.

JNDI Name, JNDI Binding Path

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 a client applications deployment descriptor and actual resources bound into JNDI by the platform.

Maximum connection pool size

Specifies 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.

Minimum connection pool size

Specifies the minimum number of connections in the pool.

This value must be a positive integer.

Name

Specifies 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 JNDI lookup for such a data source would be jdbc/WASDataSource.

Orphan timeout

Specifies the maximum time in seconds that an inactive (but allocated) connection can remain in the pool before being removed.

This value must be a positive integer.

 The administrative server and console can exhibit odd behavior in response to database URLs that are not valid. When you use a database URL, ensure you have typed it correctly.

6.6.14.1: Administering database connections with the Java console

This article extends article 6.6.14 (the overview of administering database connections) with information specific to the Java console. As discussed in 6.6.14, database connections available to applications are represented by administrative "JDBC driver" and "data source" configurations.

The table answers the most basic questions. See the related information for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains JDBC Drivers and Data Sources folder objects.</p> <p>The Topology tree can contain zero or more existing JDBC drivers and data sources. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	<p>On the console menu bar:</p> <p>Console -> Task -> Create Data Source</p>

6.6.14.1.1: Configuring new database connections with the Java administrative console

The product offers a couple of approaches to configuring JDBC drivers and data sources, the resources necessary to allow applications to connect to databases:

- By clicking Console -> Tasks -> Create Data Source from the console menu bar.
- Using menus on resources in the Topology and Type trees (see Related information)

The first two methods lead to the Create Data Source task wizard, for which detailed help is provided here.

1. Follow the wizard instructions. On the first page, specify whether to use a JDBC driver you have already configured, or to configure a new one as part of the wizard.
2. Click Next to proceed. If you specified to use an existing JDBC driver, skip to the next step in these instructions. Otherwise, complete the next two panels to configure a new JDBC driver:
 - Specify JDBC driver properties, including a name by which to administer the driver.

[View data access properties help](#)

- Select a node on which the driver will reside. Then Browse for the JAR file containing the driver code.

Click Next to proceed. Note, the wizard will not let you proceed until the "Jar file" field contains a value.

3. Specify a name by which to administer the data source. Specify the name of the database to which the data source should provide connections.

[View data access properties help](#)

4. Click Finish.

6.6.14.3: Administering database connections with the Web console

Use the Web console to edit the configurations of JDBC drivers and data sources, which are used by your installed applications to access data from databases. You can create, modify, and install JDBC drivers. You can create and modify data sources.

Click **Tasks** -> **Create Objects** -> **Create JDBC Driver** to create a new JDBC driver.

Click **Tasks** -> **Create Objects** -> **Create Data Source** to create a new data source.

In order to create a data source, a driver must already exist with which you can associate the data source. Existing JDBC drivers and data sources in the administrative domain are displayed in the **Resources** section of the navigation tree.

 Creations and changes made with this console are not applied to the administrative domain until you Commit them. Refer to section 6.6.0.3.5 for details.

6.6.14.4: Property files pertaining to database connections

The database connection properties are in file:

- *admin.config*

This file is located in directory:

<WebSphere/Appserver>/bin

The following entries in the *admin.config* file apply to database connections:

com.ibm.ejs.sm.adminServer.dbUrl	URL for JDBC access
com.ibm.ejs.sm.util.adminServer.dbSchema	database schema name
com.ibm.ejs.sm.adminServer.dbDriver	Classname of JDBC driver
com.ibm.ejs.sm.adminServer.dbPassword	Password for database access
com.ibm.ejs.sm.adminServer.dbUser	User ID for database access

See the Related information for a description of database connections and their properties.

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 via Merant SequeLink 5.1 on Windows NT](#)
- [For Sybase on AIX](#)
- [Tips for selecting JDBC drivers](#)

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/sqlib/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/sqlib/java12/db2java.zip` then JDBC 2.0 is in use.

If it contains

```
$INSTHOME/sqlib/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/sqlib/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.

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 the IBM WebSphere Application Server administrative service.
3. 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.)
4. Make sure that you already [enabled JDBC 2.0](#).
5. Start the DB2 services.
6. 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**.

7. 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).

8. 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:
 - o **Class name** = COM.ibm.db2.jdbc.app.DB2Driver
 - o **URL prefix** = jdbc:jta:db2
 - o **JTA enabled** = True

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. 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:
 - o **Class name** = COM.ibm.db2.jdbc.app.DB2Driver
 - o **URL prefix** = jdbc:jta:db2
 - o **JTA enabled** = True

Using JTA drivers for Oracle via Merant SequeLink 5.1 on Windows NT

To enable JTA drivers for use with Oracle via SequeLink on the Windows NT operating system, follow these steps:

1. Setting up Oracle 8.1.6:
 - o Use the following command at the Oracle command prompt to create the user:

```
create user dbuser1 identified by dbpwd1 default tablespace users temporary tablespace temp profile default account unlock;
```
 - o Ensure XA connectivity for the user by issuing the following command at the Oracle command prompt:

```
grant select on "SYS"."DBA_PENDING_TRANSACTIONS" to dbuser1;
```
2. Setting up direct database authentication through SequeLink Manager:
 - o In **SequeLink Services, SLOracle51, Configuration, Service Settings**, and **User Security**, set **ServiceAuthMethods** to **Anonymous**.
 - o In **SequeLink Services, SLOracle51, Configuration, Data Source Settings, Default**, and **User Security**, set **DataSourceLogonMethod** to **DBMSLogon(UserID,Password)**.
 - o Because the XAOpen string, containing the user ID and password, is stored in the trace log, to ensure security, enter the following setting: In **SequeLink Services, SLOracle51, Configuration, Service Settings, Logging** set **ServiceDebugLogLevel** to either **Fatal** or **Errors**.
3. Setting up WebSphere Application Server:
 - o When you create your new JDBC driver, the URL prefix defaults to the following:

```
jdbc:sequelink//hostname:1996
```

Change *hostname* to the name of your machine, and enter the port number you are using, represented here by the SequeLink default of *:1996*.

- When you create the data source, enter the Oracle environment variable `ORACLE_SID` in the Database Name field.

Enter the Oracle environment variable `ORACLE_SID`.

- Select your server, and on the **General** tab, click **Environment**. Add the `CLASSPATH` variable, and set its value to the path in which the SequeLink Client is installed, for example:

```
#$WAS_HOME/driver/lib/sljcx.jar
```

where `$WAS_HOME` is the location where WebSphere Application Server is installed.

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.

Tips for selecting JDBC drivers

Here are some tips for selecting JDBC drivers:

- For DB2, consider type 2 application drivers, which are typically faster than type 2 network drivers.
- DB2 6.1 on HP-UX does not support JDBC 2.0.
- For Sybase, consider type 4 thin drivers.
- On Sybase, JDBC 2.0 support is provided by the jConnect 5.2 component
- For Oracle, consider type 4 thin drivers.

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.

- Do not drop the administrative database while the administrative server is running.
- DB2 performance on a local machine can be improved by setting up a local database as a remote instance. On UNIX systems, this is required. The configuration uses TCP/IP instead of shared memory.

Oracle and Sybase also support client/server connections. Consult their product documentation for specifics.

- To use SQLJ static queries with DB2/MVS, be sure to add sqlj.jar and runtime.jar to the application server classpath only.
- 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;
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.

6.6.14.7: Notes about InstantDB

InstantDB limitations

InstantDB is provided only for running the Hit Count enterprise bean example, and the WebSphere Samples Gallery.

Two instances of InstantDB databases are created when the product is installed:

- sampleDB (created in *product_installation_root*/bin)
- IDBDatabase *product_installation_root*/installedApps/Samples.ear)

 The use of InstantDB in a development, test, or production environment (other than to run the WebSphere Samples) is not supported.

Do not use InstantDB in a production environment. Do not use InstantDB in a development or test environment that requires functionality beyond the limitations described below.

InstantDB does not support:

- Persistence of application data
- CMP function for enterprise beans
- Access by administrative clients on remote machines (unless using administrative server agent)
- Multinode function
- Cloning of application servers
- Redirection of application requests from Web servers on one machine to application servers on another
- Distributed transactions
- Workload manager persistence

InstantDB tracing

InstantDB places its tracing information in the file:

product_installation_root/bin/trace.log

where *product_installation_root* refers to the IBM WebSphere Application Server installation root.

To turn logging off:

1. Open the following file in a text editor: *product_installation_root*/bin/was.prp
2. Locate the line:
`traceFile=./trace.log`
3. Comment the line out:
`! traceFile=./trace.log`

6.6.14.10: Establishing database failover support with HACMP

This document describes a high availability database failover scenario for IBM WebSphere Application Server Advanced Edition using two AIX systems and a shared external disk that stores the databases used by WebSphere applications. The scenario uses an IBM product called High Availability Cluster Multi-Processing (HACMP).

- [Configuration overview](#)
- [Hardware and software](#)
- [Building a two node hot standby HACMP cluster](#)
- [WebSphere and Web server configuration options](#)
- [Performing a controlled failover to verify the configuration](#)
- [Current limitations and considerations](#)

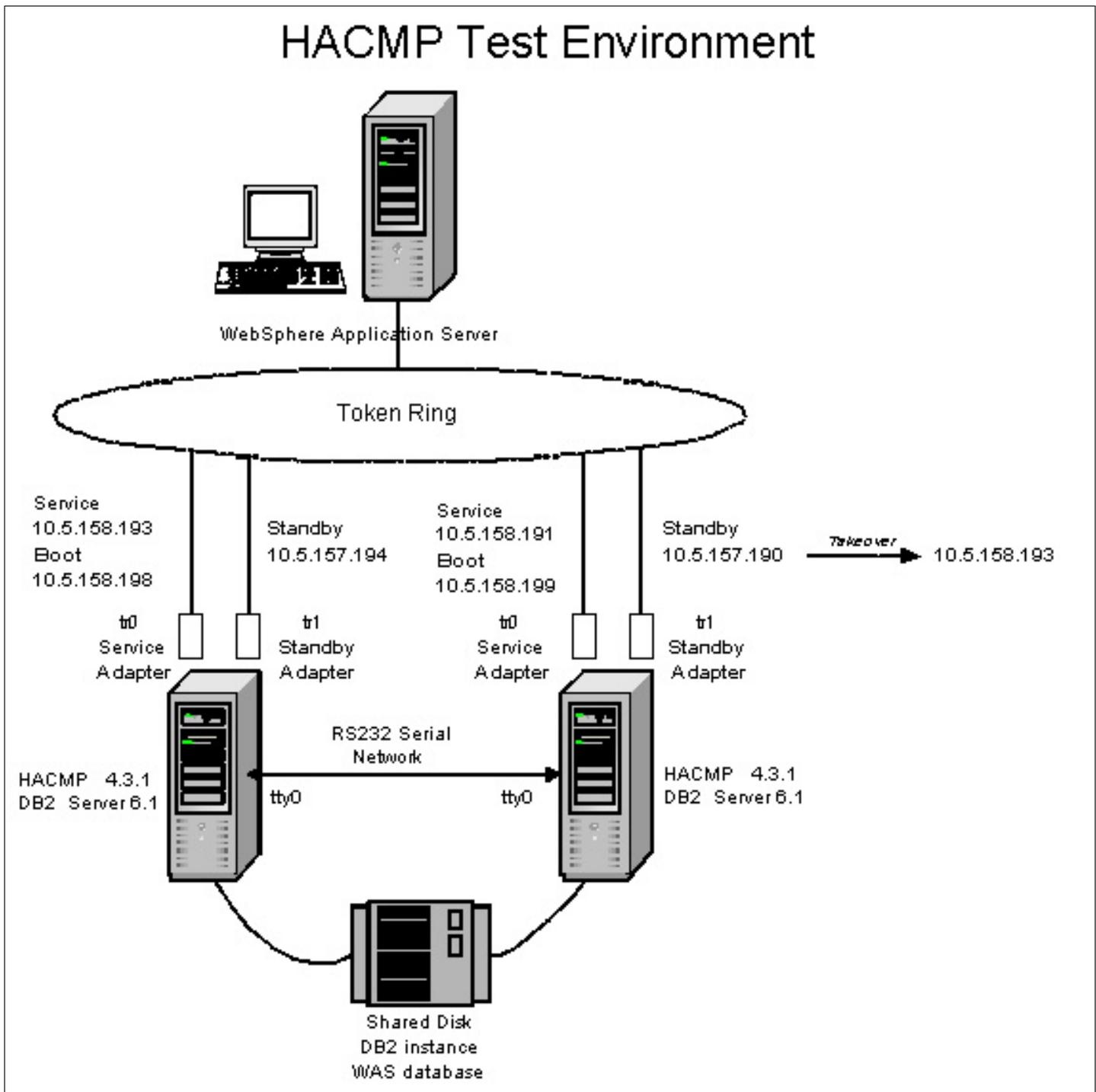
For some steps in the setup procedure, you are referred to related documents for more detailed instructions. If you need these documents, see the **Related information** at the bottom of this article for links.

Configuration overview

A cluster of two AIX systems can be used to build a high availability database environment. One AIX system is used as the primary DB2 server. The second system is used as the backup DB2 server system providing standby failover support for when the first system has a failure.

The cluster in the figure below demonstrates a two node **Hot Standby** configuration. It shows a common cascading scenario. Resources move to the second "hot standby" node if the primary node fails. The two RS/6000 machines run DB2 server software and the HACMP software.

HACMP Test Environment



The HACMP software is configured to control the DB2 software that accesses the databases located on the shared disk. By using a shared disk, either system can access the same databases, including the WebSphere administrative database (named "WAS" by default) and any other databases supporting applications managed by WebSphere Application Server.

This eliminates the first primary AIX system and its DB2 server software from being a single point of failure.

IBM WebSphere Application Server is installed on one or more other systems that will access the DB2 databases remotely over the network. Initially, it will connect through the first DB2 server system. When that first DB2 primary system has a failure, the backup system takes over as if it is the primary system and assumes the TCP/IP address of the first system.

When a failure occurs, HACMP will:

1. Detect the failure, such as a system, network, or application failure
2. Stop the DB2 server on the first system
3. Release the shared resources from the first system (disk volume groups)
4. Assume the service IP address on the standby adapter of the second system
5. Assign the shared resources to the second system
6. Start DB2 server on the second system

Hardware and software

In addition to IBM WebSphere Application Server, this scenario requires:

Software	Hardware
<ul style="list-style-type: none"> ● Supported AIX version ● HACMP Version 4.3.1 ● Supported DB2 version 	<ul style="list-style-type: none"> ● 2 RS/6000 workstations ● 4 Token Ring 16/4 adapters ● 1 shared external disk configuration using Serial Storage Architecture (SSA) ● 1 IBM serial null modem cable

See the [product prerequisites](#) for information about supported software.

Building a two node hot standby HACMP cluster

The following procedure demonstrates how to set up a two node hot standby HACMP cluster. For more detail, consult the *HACMP for AIX - Installation Guide*.

1. Install network adapters in the cluster nodes

Install two adapters on each node:

- service/boot adapter
- standby adapter.

2. Configure the network settings

Configure the four adapters on the cluster nodes. First, configure the two standby adapters with the standby TCP/IP addresses.

Second, configure the two service adapters with the boot addresses. The service addresses for the same two service adapters will be configured later by HACMP.

Notice that the service adapter and standby adapters have to be on the *same* physical network but on *different* subnets. Some details are shown on the HACMP Test Environment figure in the configuration overview section.

3. Interconnect the workstations for HACMP

Use the null modem cable to connect the two nodes through their serial ports. This serial connection will act as a private network between the two HACMP cluster nodes and will carry the "keep alive" packets between them without using the public TCP/IP network.

Test the RS232 network by issuing the command:

```
stty < /dev/ttyx
```

on each system. The stty attributes should be displayed on both systems.

As an alternative to using the RS232 connection, if you use a SCSI device or SSA shared disk system, you can set the Target Mode SCSI/SSA connections to provide an alternative serial network. For details, see the *HACMP for AIX - Installation Guide*.

4. Install shared disk devices

The application data for applications being managed by IBM WebSphere Application Server needs to be on a shared device that both nodes can access. It can be a shared disk or a network file system. The device itself should be mirrored or have data protection to avoid data corruption.

The configuration described in the configuration overview depicts an IBM SSA Disk Subsystem for this purpose.

5. Define shared volume groups and file systems

Creating the volume groups, logical volumes, and file systems shared by the nodes in an HACMP cluster requires steps to be performed on all nodes in the cluster.

In general, define the components on one node and then import the volume group on the other nodes in the cluster. This ensures that the ODM definitions of the shared components are the same on all nodes in the cluster.

Whether to define a Non-concurrent access or Concurrent access volume group depends on how you set up the cluster. In the hot standby configuration, a shared Non-Concurrent access volume group with a Journaled file system was used so that only one node can access the volume group and the file system at a time. HACMP will switch the resource from one node to the other node.

To learn more about defining shared volume groups and file systems, see the *HACMP for AIX - Installation Guide* and your AIX documentation.

6. Install HACMP software

Use "smit install_latest" to install:

- cluster.adt
- cluster.base
- cluster.clvm
- cluster.cspoc

and related files on both nodes. HAView, a monitor tool, is not needed in the configuration.

7. Install DB2 server

Install DB2 server on both nodes. The DB2 installation path can either be in a directory shared by both nodes or on a non-shared file system. When using a non-shared file system, the installation level must be identical.

8. Create DB2 instance

Create a DB2 instance for the database. The DB2 instance path, as with the installation path can either be on a shared file system or on a manually mirrored file system. For the configuration discussed in the overview, the DB2 instance was created on the shared SSA disk system.

9. Confirm that the WAS database exists for the IBM WebSphere administrative server to use

Make sure the WAS database exists. If not, create one. In either case, ensure that the application heap size (APPLHEAPSZ) of the database is set to 256.

To manually create the WAS database and set the application heap size, execute these commands:

```
db2 CREATE DATABASE wasdb2 UPDATE DB CFG FOR was USING APPLHEAPSZ 256
```

If you later need to repeat the installation procedure, be sure to drop the WAS database before you install

again. Use IBM DB2 Control Center or the following command to drop the database:

```
db2 DROP DATABASE WAS
```

10. Define the cluster topology

Use "smit hacmp" to define clusters, cluster nodes, network adapters and network modules. In the configuration above, "cascade" was used so node 1 always has higher priority than node 2.

For details, see the *HACMP for AIX - Installation Guide*.

11. Configure cluster resources

In HACMP terms, an "application server" is a cluster resource made highly available by the HACMP software. In the configuration shown in the overview section, the DB2 instance on the shared disk is the "application server."

An application server has a start script and a stop script. The start script starts the application server. The stop script stops the application server so that the application resource can be released, allowing the second node to take it over and restart the application.

Sample start script (start.sh):	db2start
Sample stop script (stop.sh):	db2 force application all; db2stop
Sample start usage:	su - db2inst1 start.sh
Sample stop usage:	su - db2inst1 stop.sh

Create the start and stop scripts for both cluster nodes. Configure the application server with the path to them.

12. Start the HACMP cluster

Start HACMP on the first node. Start HACMP on the second node after the start is complete on the first node. Use the /tmp/cm.log file to monitor the cluster events.

WebSphere and Web server configuration options

Now you can install and configure WebSphere and a Web server on other systems. You can set up the front end systems in a variety of ways. It is most simple to use just one system to run both a Web server and WebSphere Version 3.02. This is appropriate for a test environment.

1. Install the DB2 client software
2. Configure the DB2 client to connect to the remote WAS database
3. Install the Web server and WebSphere Application Server.

When the Application Server installation prompts you for information about the administrative database, specify the previously configured remote database. (Alternatively, you can change the database setting by modifying the "com.ibm.ejs.sm.adminServer.dbUrl" directive in the admin.config file).

Performing a controlled failover to verify the configuration

1. Start HACMP on the first node. Monitor the /tmp/cm.log file for completion messages.
2. Start HACMP on the second node.
3. On the WebSphere system, start the WebSphere administrative server. Monitor the /logs/tracefile for the message:
WebSphere Administration server open for e-business
4. Start the WebSphere administrative console and any application servers.

5. Initiate a controlled failover on the first node:

1. Issue the command:

```
smit hacmp
```

2. From the menus, select Cluster Services -> Stop Cluster Services -> Stop now with "Shutdown mode - graceful with takeover." You can monitor the "/tmp/cm.log" on both systems to watch the progress.

6. When the failover is complete, refresh or start a new administrative console. Check the topology to see that the servers are functional.

Current limitations and considerations

During times when there is no active database connection, such as when the remote database server is stopped, and soon after the database connection is reestablished, using a Java administrative console (WebSphere Administrative Console) will produce some warning and error messages. The most common messages are:

```
getAttributeFailureFailed to roll back ... Connection is closed
```

In such circumstances, it is recommended that you close the console and start a new one.

6.6.14.11: Recovering from data source configuration problems using the XAResources file

When a WebSphere application server starts, the product stores information about any JTA-enabled data sources used by the server. This information is needed to recover from server failures. The information is kept in the file:

`product_installation_root/properties/application_server_nameXAResources`

Every time the server comes up, the server runtime examines this file and attempts any needed recovery on the resources described in this file.

In the event that a JTA-enabled DataSource is configured incorrectly, you might see server startup failures or warning messages due to the incorrectly-configured resources.

For example, you might see the warning message:

```
WTRN0025W: Can not create XAResource object
```

There is a work-around for this problem, *if* you are sure that you do not have other JTA-enabled data sources on this server for which recovery needs to be driven. You can simply remove the `application_server_nameXAResources` file from the properties directory, correct the configuration error, and attempt to start the server again. The XAResources file will be re-created when the server starts.

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:

- 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

Aliases

Specifies one or more DNS (Domain Name Service) aliases for the virtual host.

Aliases must be unique within the administrative domain. If two virtual hosts have the same alias, when a request arrives for that alias, the results will be unpredictable. The request could be routed to either host.

Each alias has the format *name:port*. If the port is not specified, port 80 (the Web server port) is assumed.

Resources such as servlets, Web pages, and JavaServer Pages (JSP) files can be associated with a virtual host. When a resource associated with a virtual host is served, an alias of the virtual host comprises part of the resource's served path.

For example, `http://myalias:9876/servlet/snoop` might be the served path of the *snoop* servlet residing on a virtual host with the alias *myalias* on port 9876.

Aliases must be unique within the administrative domain -- two virtual hosts cannot have the same alias.

Default Virtual Host

Indicates whether this virtual host is the default virtual host. The value can be true or false.

MIME Table Parameters

Specifies the MIME Type/Subtype pairs to recognize.

- MIME Type - The MIME Type, such as text, image, or application
- Extension - The MIME Subtype, such as html, gif, or exe

When Application Server recognizes a MIME type, the servlet associated with that MIME type is invoked as a filter. (Use the [Filters property of the Web application](#) to correlate MIME types with particular servlets).

FP 3.5.2 Each Web application has its own MIME settings that take precedence over settings at the virtual host level (local scope).

Virtual Host Name, Virtual Host

Specifies the name by which to administer the virtual host. The name must be unique within the administrative domain containing the virtual host.

6.6.16.1: Administering virtual hosts with the Java administrative console

This article extends article 6.6.16 (the overview of administering virtual hosts) with information specific to the Java console.

The table answers the most basic questions. See the related information for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains a Virtual Hosts folder object.</p> <p>The Topology tree can contain one or more existing virtual hosts (there is always a default virtual host). Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	<p>On the console menu bar:</p> <p>Console -> Task -> Create a virtual host</p>

6.6.16.1.1: Configuring new virtual hosts with the Java administrative console

1. Click the Tasks tab.
2. Click Create a virtual host.
3. On the first page of the wizard, name the virtual host. Click Next.
4. On the second page, specify [virtual host properties](#).
5. Click Finish.

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 **Tasks** -> **Create Objects** -> **Create Virtual Host** to create a new virtual host.

Existing virtual hosts in the administrative domain are displayed in the **Resources** section of the navigation tree.

 Creations and changes made with this console are not applied to the administrative domain until you Commit them. Refer to section 6.6.0.3.5 for details.

6.6.16.4: Property files pertaining to virtual hosts

The virtual hosts properties are in file:

- *vhosts.properties*

A sample, inactive *vhosts.properties* file is located in directory:

<WebSphere/Appserver>/properties

When Application Server is started, the runtime version of the *vhosts.properties* file is created and stored in directory:

<WebSphere/Appserver>/temp

Each virtual host created through the GUI, should have the following mappings in the active *vhosts.properties* file:

- TCP/IP address
- short host name
- fully qualified host name
- local host alias
- loopback address

If no virtual hosts are created, the *vhosts.properties* file will contain information on the default virtual host called, *default_host*.

See the Related information for a description of virtual hosts and their properties.

6.6.17: Administering Web resources (overview)

Web resources include configurations for identifying the relative locations of:

- JavaServer Pages (JSP) files
- Web pages
- Alternate Web paths for servlets

Web resources provide Web paths for identifying the locations of servlets, Web pages, and JSP files. A Web resource is a configuration consisting of:

- A name by which users should request the resource
- A Web application to which the resource belongs
- A virtual host by which the resource is hosted

Put together, the virtual host, Web application, and Web resource name comprise the information a user should type into a browser to request the resource.

For example, based on the Web resource configuration:

- Name: /myServlets/myAlternateServletPath
- Virtual host: myVirtualHost

and the association of the Web resource with myWebApp, a user should get a particular servlet when he or she types this into a browser:

`http://myVirtualHost/myWebApp/myServlets/myAlternateServletPath`

6.6.17.0: Web resource properties

Web Resource Name

Specifies a unique name for the Web resource. The name will be part of the Webpath by which the resource is accessed from a browser or other client.

Virtual Host

Indicates the virtual host with which the Web resource is associated.

6.6.17.1: Administering Web resources with the Java administrative console

This article extends article 6.6.17 (the overview of administering Web resources) with information specific to the Java console.

The table answers the most basic questions. See the Related information for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	The Type tree contains a Web Resources folder object. The Topology tree can contain zero or more existing Web resources. Their names vary; they are configured automatically as a result of other administrative actions. Use the View menu on the console menu bar to toggle between tree views.
Any task wizards for manipulating this resource?	On the console menu bar: Console -> Task -> Add a JSP file or Web resource

 The administrator cannot update the properties of a Web resource. Problems could easily result if both manual and automatic updates were allowed.

6.6.17.1.1: Configuring new Web resources with the Java administrative console

The product offers several ways to configure new Web resources:

- By clicking Console -> Tasks -> "Add a JSP file or Web resource" from the console menu bar.
- By clicking Add a JSP file or Web resource from the drop-down list on the Wizards toolbar button.
- Using menus on resources in the Topology and Type trees (see Related information)

The first two methods lead to the Add a JSP file or Web resource task wizard, for which detailed help is provided here.

1. Follow the wizard instructions. Answer Yes to the question about whether you want to copy a Web resource to a predefined Web application.
2. Click Next to proceed. Specify the Web application to which to add the Web resource.
3. Click Next to proceed. Browse for the JSP or HTML file in your filesystem. This correlates the Web resource to the actual file it represents.
4. Click Next to proceed. Now specify the Web resource settings:
 - Specify a name by which users can request the resource.
 - Specify the virtual host for the Web resource.

These two values and the Web application name will be put together into a Web path for the resource:

`http://virtualHostName/myWebApp/WebResourceName`

The Web Resource name can have multiple parts, such as /examples/snoop

[View Web resource properties help](#)

5. Click Finish.

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.
3. Securing specific HTTP methods (optional).

Securing applications with IBM WebSphere Application Server 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.

Security task wizards in Java console

Of the current administrative clients, WebSphere Administrative Console provides the most comprehensive support for securing applications, in the form of security task wizards for:

- Enabling product security
- Defining a security realm and set of valid users
- Specifying how to authenticate users seeking access to applications
- Organizing methods (functions, operations) into groups for protection
- Granting users permissions to access applications

6.6.18.1: Securing applications

The table summarizes the security wizards provided for accomplishing the tasks necessary to secure an application.

Goal	Wizard	Conceptual overview	Instructions
1. Enable security; set application security default and global values; specify how to authenticate users	Configure Global Settings	6.6.18.1a	6.6.18.1.1a
2. Secure a particular application, making users authenticate their identities before using it	Configure Application Security	6.6.18.1b	6.6.18.1.1b
3. Configure custom method groups as an optional step towards defining who will be allowed to access applications	Configure Method Groups	6.6.18.1c	6.6.18.1.1c
4. Assign the methods in a resource, such as a servlet, to a custom or default method group for protection	Configure Resource Security	6.6.18.1d	6.6.18.1.1d
5. Specify which users and groups can access which methods in which applications	Configure Permissions	6.6.18.1e	6.6.18.1.1e

Test the application

At this point, your resources will be secure. A user who runs a client program that accesses secured resources will be prompted to log in. The user must log in with an account that has been granted access to the resources, otherwise the user will be denied access. The visible effect of this denial is that the client program will trigger an authorization failure, for example, a `java.rmi.ServerException` that contains a `com.ibm.ejs.EJSSecurityException`.

6.6.18.1.1: Securing applications

The procedure for securing an application with the Java console (WebSphereAdministrative Console) is as follows:

1. [Specify global and default security settings for all applications](#)
2. [Configure security for the particular application](#)
3. [Configure custom method groups to protect methods](#)
4. [Assign methods to custom or default method groups](#)
5. [Assign permissions allowing users access to methodgroups and applications](#)

6.6.18.1.1a: Specifying global settings

1. Start the Configure Global Settings task by one of two methods:
 - By clicking Console -> Tasks -> Configure Global Settings from the console menu bar.
 - By clicking Configure Global Settings from the drop-down list on the Wizards toolbar button.
2. Complete the task, referring to the information below for assistance.
3. Stop the administrative server and start it again for the changes to take effect.

The next time the administrator opens the WebSphere Administrative Console, the administrator will be prompted to log in, using an ID and password specified during Global Settings configuration.

General

Use the General tab to specify whether to enable security. If the check box is *not* selected, any other security settings you specify will be disregarded.

This page also contains an option for setting a security cache timeout. The security system caches authentication lookup information it receives from the user registry or directory service. Use this field to specify how long to cache the information (in seconds). Caching can improve lookup performance.

Application Defaults

Specify a default security realm and challenge type for applications. The administrator can later override these values in the security settings for an individual application.

The challenge type is the way in which users will be challenged for their credentials, for example, using a digital certificate or user ID and password combination.

To refuse to service requests that are not transmitted over SSL (Secure Sockets Layer), click the option for using SSL to connect the client and Web server.

Authentication Mechanism

Use the Authentication Mechanism tabbed page to specify how to authenticate the information presented by users trying to access an application or resources.

The administrator can have users or groups authenticated against either the local operating system user registry (such as Windows NT User Manager program) or an LDAP-enabled directory service product.

User Registry

Use the User Registry page to specify details about the authentication mechanism you chose.

The contents of this page vary according to the authentication mechanism. If you chose the directory service option, consult the properties help for filling in the filters and other values.

6.6.18.1.1b: Configuring application security

1. Start the Configure Application Security task by one of two methods:
 - By clicking Console -> Tasks -> Configure Application Security from the console menu bar.
 - By clicking Configure Application Security from the drop-down list on the Wizards toolbar button.
2. Specify the enterprise application to which to apply security.
3. Click Next to proceed. Modify the application security defaults if necessary.
4. Click Finish.

6.6.18.1.1c: Configuring custom method groups

1. Start the Configure Method Groups task by one of two methods:
 - By clicking Console -> Tasks -> Configure Method Groups from the console menu bar.
 - By clicking Configure Method Groups from the drop-down list on the Wizards toolbar button.
2. Specify to add a new method group.
3. Click Next to proceed. Type a name for a new method group.
4. Click Finish.

When finished configuring method groups, exit the task by clicking any other resource or task in the administrative console.

6.6.18.1.1c.1: Viewing custom method groups

To confirm the existence of a custom method group:

1. Start the Configure Method Groups task.
2. Specify to remove an existing method group.
3. On the resulting page, verify that the new custom method group is displayed in the set of existing method groups.
4. Cancel the task without actually removing a method group.

Of course, even if the administrator does not perform the above task, the administrator will detect any problems with custom method group creation when he or she proceeds to assign methods to the method groups.

6.6.18.1.1d: Configuring resource security

1. Start the Configure Resource Security task by one of two methods:
 - By clicking Console -> Tasks -> Configure Resource Security from the console menu bar.
 - By clicking Configure Resource Security from the drop-down list on the Wizards toolbar button.
2. Specify the resource to which to apply security.

Note that servlets, JSP files, and Web pages are not represented directly, but can be selected according to their "Web resource" configurations. Web resources specify Web paths to these resources. If configuring resource security for a servlet, JSP file, or Web page, select the appropriate Web resource.

3. Click Next. A prompt asks whether to use the default method groups.

If you specify Yes, the security system will take a first pass at grouping the methods of the resource into the default method groups. It will use the method names to decide which groups to put them in.

For example, it will protect HTTP PUT methods with the WriteMethods group.

If you decline to use the default method groups, you will need to specify a method group for each method in the resource.

i The distribution of the methods into the default method groups is not finalized until you finish this task wizard. You can always try the default method groups, then reverse the operation if you do not like the results. Just do not click the Finish button until you are sure about the method groups!

4. If the Finish button is available, click it. If configuring resource security for an enterprise bean, click Next to proceed to a final panel for specifying delegation settings.
5. Click Finish.

i If your Web server is already running when you configure resource security for an HTML file, JSP file, or other Web resource, you need to stop the WebSphere administrative server and start it again for the change to take effect.

6.6.18.1.1d.1: Default method groups

The product predefines these method groups:

Method group	Typical Web resource methods
ReadMethods	GET and POST
WriteMethods	PUT
RemoveMethods	DELETE
CreateMethods	None
FinderMethods	None
ExecuteMethods	Methods that do not fit in other groups

In addition, the administrator can create custom method groups.

6.6.18.1.1e: Configuring permissions

1. Start the Configure Permissions task by one of two methods:
 - By clicking Console -> Tasks -> Configure Permissions from the console menu bar.
 - By clicking Configure Permissions from the drop-down list on the Wizards toolbar button.
2. Click a permission, such as *AnyApplicationName-ReadMethods*.

The administrator can view permissions by application or by method group:

- Viewing by application shows only the permissions associated with a particular application, such as:
 - Application_A-CreateMethods
 - Application_A-WriteMethods
 - Application_A-CustomMethodGroup
 - Viewing by method group shows only the permissions associated with a particular method group, such as:
 - Application_A-CreateMethods
 - Application_B-CreateMethods
 - Application_C-CreateMethods
3. Click the Add button to produce a search dialog.
 4. Use the search dialog to give permission to everyone or selected users or groups. You can search for a user or group in your local operating system user registry or directory service product.
 5. When finished with the search dialog, click the OK button.
 6. Back in the main console window, verify that the user or group is listed under the permission you granted to the user or group.

When finished configuring method groups, exit the task by clicking any other resource or task in the administrative console.

Securing WebSphere administrative accounts

Ability to administer WebSphere Application Server after it has been secured is governed by a Web application. You can set up an initial account and additional administrative accounts to access the secured product. See [the information about administrative accounts](#) for details and instructions.

Setting permissions to authenticate against local and domain registries (Windows)

WebSphere Application Server security supports authentication both against the domain registry and the local registry of a supported, Windows-based machine. The administrator can force authentication against the local registry by setting permissions appropriately.

If a machine is part of a Windows domain, when a user authenticates to WebSphere Application Server security, the user is first authenticated against the domain registry. If that fails, the user is authenticated against the local operating system registry.

If the user exists in both the local and domain registries, and authorization has been granted to the local user, it becomes necessary to qualify the user name when logging on to WebSphere security.

For an example of the implications of setting permissions, suppose a machine named "LOCAL" belongs to a domain named "DOMAIN." The users "user1" and "user2" exist in both the LOCAL and DOMAIN registries:

- LOCAL\user1
- LOCAL\user2
- DOMAIN\user1
- DOMAIN\user2

Suppose the WebSphere administrator configures permissions such that the following users can access a WebSphere resource:

- LOCAL\user1
- DOMAIN\user2

When user1 logs on to access a resource, he or she must specify LOCAL\user1 (not simply user1) as the user name for successful authentication. When user2 logs on, he or she can specify simply user2.

6.6.18.1.4: Properties related to security

The WebSphere Administrative Console provides security wizards for accomplishing various goals. Property (field) help provides detailed information about options and data fields in the wizards.

Goal	Wizard	Property help
Enable security; set application security default and global values; specify how to authenticate users	Configure Global Settings	6.6.18.1.4a: Global Security settings
Secure a particular application, making users authenticate their identities before using it	Configure Application Security	6.6.18.1.4b: Application security settings
Configure custom method groups as an optional step towards defining who will be allowed to access applications	Configure Method Groups	6.6.18.1.4c: Method group settings
Assign the methods in a resource, such as a servlet, to a custom or default method group for protection	Configure Resource Security	6.6.18.1.4d: Resource Security settings
Specify which users and groups can access which methods in which applications	Configure Permissions	6.6.18.1.4e: Permission settings
Search for users defined in the operating system registry or LDAP server	Available from multiple security tasks	6.6.18.1.4f: Security search dialog

6.6.18.1.4a: Properties for configuring global settings

Settings on the General, Authentication Mechanism, and User Registry tabbed pages specify global settings for all applications to share. These values cannot be customized for individual applications.

The Application Defaults page specifies default settings that the administrator can accept or override for individual applications.

Goal	Wizard page/Property help
Enable security; specify how long to cache authentication lookup results	6.6.18.1.4a.1: General page
Define a security realm; specify a default challenge type; specify a Web page for user logon	6.6.18.1.4a.2: Application Defaults page
Select either the operating system registry or an LDAP directory service to authenticate users	6.6.18.1.4a.3: Authentication Mechanism page
Specify the identity under which the product security server will run; provide details about LDAP directory if it is the chosen authentication mechanism	6.6.18.1.4a.4: User Registry page

6.6.18.1.4a.1: General settings of the Configure Global Settings task

Enable security

Specifies whether to enable or turn off WebSphere Application Server security.

If the administrator deselects the check box, all other security settings will be disregarded and applications and resources will be unprotected.

Security Cache Timeout

Specifies how many seconds servers should cache security information received from the user registry or directory service, improving the performance with respect to authorization lookups.

To make changes to this property take effect, stop and restart the application server or servers under which the applications using the security settings will run.

6.6.18.1.4a.2: Application Default settings of the Configure Global Settings task

Realm Name

Specify the security realm to which the application should belong. See [article 0.18.7](#) to learn more.

Challenge Type - None

Specifies that clients will not be challenged for authentication information.

 If the administrator has protected a resource within an application, selecting None will deny users access to that resource.

Challenge Type - Basic

Specifies that clients will be prompted for a user ID and password, usually acquired through a basic HTTP 401 challenge.

Challenge Type - Certificate

Specifies that clients must provide a digital certificate for authentication.

If the administrator additionally selects Default to Basic, clients without certificates will be permitted to use the basic authentication scheme.

Challenge Type - Custom

Specifies that clients will log in using servlet-generated Web pages you specify in the Login URL and Relogin URL fields.

Currently, the administrator needs to enter the same URL in each of the two fields. The URL is intended to reference a Web page containing an HTML-based login form, but the administrator can enter the URL of any Web page, whether or not it offers a login form.

For example, the field could contain the URL

```
http://host.name.com/login/deny.html
```

for a Web page created to deny access to users without allowing the users to attempt login.

Login URL

Specifies the fully-qualified path to the Web page to be presented for users to log on to. The administrator should complete this field if he or she specified the Custom challenge type. Currently, this field must match the Relogin URL.

The product does not validate this field or the Relogin URL.

If Single Sign-On (SSO) is enabled, the URL must be contained within the domain specified in the Single Sign-On configuration.

Relogin URL

Specifies the fully-qualified path to the Web page to be presented when the connection is released and a user must log on again. Complete this field if you specified the Custom challenge type. Currently, this field must match the Login URL.

Use SSL to connect client and Web server

Specifies that an SSL connection is required between the client and Web server. Requests that do not arrive over SSL will be refused.

This check box applies to the Basic, Certificate, and Custom challenge types.

6.6.18.1.4a.3: Authentication Mechanism settings of the Configure Global Settings task

Local Operating System

Specifies that information will be authenticated with the underlying operating system's user registry. Usually, such registries apply basic authentication, checking a user ID and password.

This selection influences the fields displayed on subsequent tabbed pages. If the administrator enables authentication by the Local Operating System, some properties described in this file will not be displayed because they do not apply to that situation.

If using a Windows-based operating system belonging to a domain, [see the note about configuring permissions](#).

Lightweight Third Party Authentication (LTPA)

Specifies that basic or certificate authentication will be used to authenticate the user with an LDAP directory service. If you select LTPA, provide additional information:

- **Token Expiration:** Specifies how many minutes can pass before a client using an LTPA token must authenticate again. LTPA uses tokens to store the authenticated status of a client.

Legal Values:

- A **positive** integer indicates the token life, in minutes
- **Generate Keys:** Specifies whether the LTPA mechanism should generate a new set of encrypted keys right now. When prompted for a password, supply a string that is used by the underlying key generation mechanism.

When the administrator selects LTPA as the authentication mechanism, encryption keys are generated automatically. The administrator need not click this button unless he or she would like those initial keys to be replaced by new keys.

- **Import from File:** Specifies whether to import a file containing the encryption keys. This allows IBM WebSphere Application Server to share keys from other IBM products that support this functionality.

i If the administrator specified a password when he or she created the key file, the administrator will be prompted for that password when he or she tries to import the key file.

- **Export to File:** Specifies whether to export a file containing the encryption keys. This allows IBM WebSphere Application Server to share the keys with other IBM products that support this functionality.
- **Enable Single Sign On:** Enabling Single Sign On (SSO) tells LTPA to store extra information in the tokens so that other applications can accept clients as already authenticated by WebSphere Application Server. When clients try to access the other applications, they will not be interrupted and asked to log in.
 - **Domain:** Restrict SSO to servers in the domain you specify in this field.
 - **Limit to SSL connections only** Specifies to use a connection with SSL for Single Sign On, to prevent the SSO token from flowing over non-secure connections.

WebSphere Application Server Version 3.5 introduces support for Single Sign On with Domino Server. WebSphere Application Server can import and export keys and provide a Single Sign On between the WebSphere Application Server and Domino environments.

6.6.18.1.4a.4: User Registry settings of the Configure Global Settings task

The content of the User Registry tabbed page changes depending on the selections on the Authentication Mechanism tabbed page:

- If the administrator selected Local Operating System on the Authentication Mechanism tabbed page, only the Security Server ID and Security Server Password properties will be displayed on the User Registry page.
- If the administrator selected LTPA on the Authentication Mechanism tabbed page, several additional properties described in this file will be available on the User Registry tabbed page.

Security Server ID

Specifies the user ID the Application Server Version 3 security server component will run under.

The ID corresponds either to an operating system ID or an LDAP directory ID, depending on the selection on the Authentication Mechanism tabbed page.

Security Server Password

Specifies the password the Application Server Version 3 security server will run under.

Directory Type

Specifies the directory service product to use to locate information against which to authenticate users and groups.

[View supported directory services](#)

All of the supported directory service choices have predefined filters and ID maps the administrator can view by clicking the Advanced button. If the administrator changes the filters or ID maps, the Directory Type will automatically change to Custom.

"Custom" can refer to any of the supported directory types, with customized filters and ID maps.

Advanced

Specifies optional properties the administrator can use to define search filters and ID maps for the selected directory service. The administrator can also specify how certificates will be used to locate entries in the LDAP directory service.

- **Initial JNDI Context Factory:** Specifies the JNDI Context Factory to use. If the field is blank, Application Server uses the Context Factory provided by IBM.
- **Directory Type:** Specifies the brand of the directory service. Only directory services compatible with Application Server Version 3 are listed.

If the administrator changes the filter and ID map values on the Advanced dialog box, the Directory Type will change to Custom, even if the filters and ID maps the administrator is defining apply to a supported directory service.

- **User Filter:** Specifies the property by which to look up users in the directory service. For example, to look up users based on their user IDs, specify `(ampersand(uid=%v)(objectclass=inetOrgPerson)` where *ampersand* is the ampersand symbol.

For more information about this syntax, see the LDAP directory service documentation.

- **Group Filter:** Specifies the property by which to look up groups in the directory service.
- **User ID Map:** Specifies the piece of information that should represent users when users are

displayed. For example, to display entries of the type object class = inetOrgPerson by their IDs, specify `inetOrgPerson:uid`.

This field takes multiple `objectclass:property` pairs delimited by a semicolon (";").

- **Group ID Map:** Specifies the piece of information that should represent groups when groups are displayed. For example, to display groups by their names, specify `*:cn`.

The `*` is a wildcard character that searches on any object class in this case. This field takes multiple `objectclass:property` pairs delimited by a semicolon (";").

- **Group Member ID Map:** Specifies which property of an objectclass stores the list of members belonging to the group represented by the objectclass.

This field takes multiple `objectclass:property` pairs delimited by a semicolon (";"). For more information about this syntax, see the LDAP directory service documentation.

- **Certificate Mapping:** Specifies the certificate field(s) against which to check certificate validity.
 - **Exact Distinguished Name:** Checks certificate validity against the exact distinguished name held by the LDAP directory service. It locates the subject DN of the certificate in the directory.
 - **Unique Key:** Checks certificate validity using a hash function on two predetermined attributes.
 - **Certificate Filter:** Enables the Filter field for specifying an property of your choice.

Create an LDAP search filter with the contents of the certificate that will attempt to match a single Directory entry. An example of a search filter is:

```
( ampersand( cn=${ Subject : cn } ) ( version=${ Version } ) )
```

where *ampersand* is the ampersand symbol.

The list of possible variable substitutions referring to portions of the certificate is given here (in the format "variable = meaning"):

- **PublicKey** = Public Key of the certificate
- **Issuer:attribute** = The issuer distinguished name of the certificate. An attribute value must be specified that allows the administrator to select a specific attribute of the Distinguished Name. To retrieve the entire Distinguished Name, use the "DN" attribute.
- **NotAfter** = The date at which the certificate is no longer valid
- **NotBefore** = The date before which the certificate is not valid
- **SerialNumber** = The serial number of the certificate
- **SigAlgName** = The signature algorithm name
- **SigAlgOID** = The OID of the signature algorithm
- **SigAlgParams** = The DER encoded signature algorithm parameters
- **Subject:attribute** = The subject distinguished name of the certificate. An attribute value must be specified that allows the administrator to select a specific attribute of the Distinguished Name. To retrieve the entire Distinguished Name, use the attribute DN.
- **Version** = The version number
- **Certificate Filter:** If you specified the Filter Certificate Mapping, this property specifies the certificate property against which to check certificate validity.

Specifies the host name of the machine on which the directory service resides.

Port

Specifies a port number for the directory service. Port 389 is the LDAP default.

Base Distinguished Name

Specifies the base distinguished name of the directory service, indicating the starting point for LDAP searches of the directory service. (See RFC 1779 for a discussion of this technique).

Some examples include:

- uid=anyusername
- ou=people
- o=ibm

This field is required unless the product will be using a Domino directory service, in which case the administrator can leave the field blank to bind anonymously.

The host name, port, and base DN you specify in the Host, Port, and Base Distinguished Name fields are combined to form an LDAP URL, such as

```
ldap://myserver:1234/o=ibm
```

where myserver:1234 is the host name and optional port number for the directory service, and o=ibm is the base distinguished name.

Bind Distinguished Name

Specifies the distinguished name for Application Server to use to bind to the directory service. If left blank, the Application Server binds anonymously.

See the previous Base Distinguished Name field description for examples of distinguished names.

Bind Password

Specifies the password for the Application Server to use to bind to the directory service.

Use SSL to connect to directory

Specifies whether to use an SSL connection between the security server and your LDAP directory service.

If the administrator selects this option, the SSL connection will use the same SSL keyring as the one defined for SSL connections between application servers.

6.6.18.1.4a.4.1: Supported directory services

For a list of supported directory services, see the prerequisites Website discussed in article 1.3. An additional Custom option is available for tailoring any of the default filters to fit a directory service not listed above.

6.6.18.1.4b: Properties for configuring application security

Application Identity - User ID

Specifies the user ID under which the application will run. The ID is used for delegation of the application's resources.

Application Identity - Password

Specifies a password for the Application Identity.

6.6.18.1.4: Properties for configuring method groups

New Method Group

Specifies a new method group. Type the group name and click Add.

To see the new method group, expand the Method Groups folder.

Remove Method Group

Removes the selected method group. Note, the default (predefined) methodgroups cannot be removed.

6.6.18.1.4d: Properties for configuring resource security

Resource

Indicates the resource with which the administrator is working. To change the selected resource, return to a previous wizard page, on which you can select a different resource.

6.6.18.1.4e: Properties for configuring permissions

WebSphere Permissions

Specifies the cross product of all available Applications and MethodGroups. Each pair (permission) is represented by a `Application Name-MethodGroup Name` notation. Expand a permission to see the users granted that permission.

6.6.18.1.4f: Properties for the security search dialog

The search dialog lets the administrator locate users and groups in the underlying operating system user registry or in the directory service, whichever is the current authentication mechanism.

Use this reference to look up a particular property as you configure this object type. Please note, not all properties are available from every administrative interface. To learn how to access these properties from a given administrative interface, refer to the task help for configuring this object type.

Everyone, All Authenticated Users, or Selection

Specifies for who to search for, with respect to the user registry or directory service.

Search For

Specify whether to search for users, groups, or roles. The administrator can only search for roles if the directory service supports them.

Search Filter

Specify a pattern against which to search for principals. A wildcard character ("*") can be used.

Search Results

Lists matches to the specified Search Filter. The Search Results area remains empty until the administrator clicks the Search button.

6.6.18.1a: About setting global security settings

Use the Configure Global Settings task wizard to specify global and default security settings for all applications:

- Global settings apply to existing and future applications and cannot be customized.
- Default settings apply only to future applications and can be customized.

The default settings are used as a template or starting point for configuring individual applications. The administrator should still explicitly configure security settings for each application.

Goal	Wizard page description	Global or default?
Enable security; specify how long to cache authentication lookup results	6.6.18.1a.1: General	Global
Specify default security realm and challenge type for applications	6.6.18.1a.2: Application Defaults	Default
Specify how to authenticate users	6.6.18.1a.3: Authentication Mechanism	Global
Provide detail about the selected authentication mechanism	6.6.18.1a.4: User Registry	Global

IBM WebSphere Application Server provides security at several levels. The security characteristics of an individual application can come from many of these levels. At the most general level are the global security characteristics set up to act as application defaults. This file briefly describes these global values.

In WebSphere, the global defaults for security apply to all applications. Some of the values can be changed on an application-by-application basis, and others remain constant across all applications.

An example of a value that can be set on a per-application basis is the type of authentication procedure. You must establish a default procedure, but this value is used for applications that do not explicitly indicate how they will authenticate users.

An example of a value that cannot be changed on a per-application basis is whether to ignore security or not. In Application Server, security is either enabled or disabled. If it is enabled, all applications are secured according to their configurations. If security is disabled, all applications run unsecurely, regardless of their configurations.

6.6.18.1a.1: About enabling security

Configure Global Settings task:

- ▶ 1. Enable security
2. Set application security defaults
3. Specify how to authenticate users
4. Provide details about the authentication mechanism

IBM WebSphere Application Server security can be enabled or not enabled. If security is not enabled, all other security settings are ignored.

6.6.18.1a.2: About setting application security defaults

Configure Global Settings task:

1. Enable security
- ▶ 2. Set application security defaults
3. Specify how to authenticate users
4. Provide details about the authentication mechanism

Use the Application Defaults tab of the Configure Global Settings wizard to specify the default security realm and challenge type for applications. These values provide a starting configuration for applications, but some can be refined on an application-by-application basis. For example, you can set a default challenge type for authentication but allow some applications to use different challenge types.

Selecting a default security realm

All applications need to belong to a security realm. In Version 3, all applications must belong to the same security realm.

The administrator can specify a default realm. When a particular application is configured, the administrator can override the default realm by specifying a different realm for the application.

Single Sign-On (SSO) support is applied to realms. If a user who has logged on to one realm tries to access an application in another realm, the user will be prompted to log into the second realm.

Selecting a default challenge type

The challenge type specifies how users will be challenged for authentication credentials when they try to access a resource or application.

Challenge types range from no challenge, a user ID and password, or a digital certificate to a custom challenge using Web pages.

6.6.18.1.a.3: About specifying how to authenticate users

Configure Global Settings task:

1. Enable security
2. Set application security defaults
- ▶ 3. Specify how to authenticate users
4. Provide details about the authentication mechanism

Use the Authentication Mechanism tab of the Configure Global Settings wizard to specify how to authenticate or verify the user data received as a result of a challenge (such as a logon screen).

The WebSphere security server must have some way to check the user ID and password, digital certificate, or other user identification for credibility. It relies on the authentication mechanism specified by the administrator.

Before performing this subtask

Before completing the Authentication Mechanism subtask, the administrator needs to use other Configure Global Settings subtasks to specify how to challenge users for identification when they try to access applications.

Selecting how to authenticate user data

Users can be authenticated by one of two authentication mechanisms, either the operating system user registry, or a supported directory service.

Whichever authentication mechanism the administrator selects, the administrator can use the General tabbed page to specify a Security Cache Timeout value. The timeout specifies how long the security system should keep authentication data received from the directory service or user registry.

The timeout value specifies the number of seconds after which authentication information will be considered unreliable. The next time the information is needed, it will be sought again using the authentication mechanism.

6.6.18.1a.4: About providing authentication mechanism details

Configure Global Settings task:

1. Enable security
2. Set application security defaults
3. Specify how to authenticate users
- ▶ 4. Provide details about the authentication mechanism

Use the User Registry tab of the Configure Global Settings wizard to specify details about the chosen authentication mechanism, such as an operating system user registry or LDAP directory service.

Before performing this subtask...

Before completing the User Registry information, the administrator needs to use other Configure Global Settings subtasks to specify (1) how to collect identity information from users trying to access applications and other resources and (2) how to authenticate the information received.

Selecting either the OS user registry or LDAP directory service

The user registry or directory service keeps records of users with permission to access resources in the system administration domain. The security system looks to the user registry or directory service to provide information for determining whether to authenticate a user or group successfully.

The operating system user registry simply compares users to valid users in the underlying operating system. When the administrator selects the Local Operating System challenge type, the User Registry tabbed page dynamically changes to allow the administrator to set a security ID and password under which the application will run. The information is used for delegation of the application resource.

When the administrator selects Lightweight Third-Party Authentication (LTPA) as the authentication mechanism, the User Registry tabbed page changes. This change enables the administrator to specify information about the Lightweight Directory Access Protocol (LDAP)-compliant directory service product to be used.

6.6.18.1b with the Java console: About configuring application security

IBM WebSphere Application Server provides security at several levels. The security characteristics of an individual application can come from many of these levels. At the most general level are the global security characteristics set up to act as application defaults. However, the administrator can and should set application-specific values that either comply with or override the global defaults.

The Configure Application Security task lets the administrator override the challenge type and realm defaults specified as the [global settings](#). Using the task wizard, the administrator specifies the realm, challenge type and security identity (user ID and password) for a particular application.

For other properties, such as the authentication mechanism and user registry details, the application will share the global settings. These settings cannot be customized for a particular application.

Enabling security in the administrative console is also a global setting, applied as "all or none" to applications in the administrative domain.

6.6.18.1c: About assigning method groups with the Java console

The Work with Method Groups task wizard provides an easy way to create custom method groups.

Assuming you have configured one or more applications already, defining custom methods groups can be considered the optional first step of a three-part task:

1. Create custom method groups, if desired. WebSphere provides a set of default method groups, which you can use instead of creating your own.
2. Assign the methods of each resource to method groups. WebSphere provides a default assignment of methods to the predefined method groups, if you choose to use it. If you do, you should check the assignment and modify it appropriately.
3. Assign permissions (access to method groups of applications) to users.

Example

Suppose you have defined an application named `MonthlySalesChart`. The application contains a servlet. You can configure application-specific security as follows:

Create custom methods	You can protect all methods that allow writing to a database with a method group called <code>AllowedToWrite</code> and put read-only methods in a method group called <code>AllowedToRead</code> .
Assign methods to method groups	You can put the <code>HTTP_PUT</code> method of the servlet into the <code>AllowedToWrite</code> method group, and the <code>HTTP_POST</code> method into the <code>AllowedToRead</code> method group.
Assign permissions	Give users the following permissions as appropriate: <ul style="list-style-type: none">● <code>MonthlySalesChart-AllowedToRead</code>● <code>MonthlySalesChart-AllowedToWrite</code>

6.6.18.1d: About assigning methods to method groups with the Java console

After defining optional custom method groups, the administrator can specify which methods in application resources belong to each method group.

Because IBM WebSphere Application Server provides default method groups, defining custom groups is optional.

If the administrator specifies to use the default groups when working with method groups, WebSphere Application Server will take a first pass at categorizing the bean methods into the various groups, based on the method names.

Afterwards, the administrator can create additional groups and move methods there, or move methods among the default groups if needed. Use the Work with Method Groups task to establish new method groups.

To sort resource methods into method groups for protection, perform the Configure Resource Security task once for each resource in the application.

 If the Web server is already running when the administrator configures resource security for an HTML file, JSP file, or other Web resource, the administrator must stop the WebSphere administrative server and start it again for the protection to take effect.

6.6.18.1e: About assigning permissions

Use the Assign Permissions task to assign permissions to users and groups, giving them access to method groups in enterprise applications.

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.

6.6.18.5: Managing security IDs for the application server and administrative accounts

Choosing the process identity

During installation, you must identify an existing user ID and password under which the WebSphere administrative server and application servers will run. It is the operating system identity associated with the process. The operating system uses the identity to determine access to resources such as files and sockets. It is not an ID that is typically used by a human user.

If you are using the operating system registry as the authentication mechanism for checking the identity, then the identity must meet the following requirements:

- On UNIX platforms, you must use the **root** account.
 - On Windows NT, the account must be a member of the Administrators group and must have the rights to "Log on as a service" and to "Act as part of the operating system."
- i** Do not use an account whose name matches the name of your machine or Windows Domain. The WebSphere administrative server will not work in such a case.
- i** WebSphere requires the NT Browser Service to be active because WebSphere uses this service to contact the NT Primary Domain Controller (PDC). Also, be aware that, although WebSphere uses the NT PDC, it does not make use of the NT Backup Domain Controller (BDC). If the PDC is not available, WebSphere does not default to the BDC.

If you are using an LDAP directory service for authentication, then the process identity does not need any special privileges. See the [information about running as non-root on UNIX-based systems](#).

Establishing the administrative identity

When you enable WebSphere security by using the [Configure Global Settings](#) security administration task, you configure an initial administrative identity for WebSphere. This identity needs to be a valid user for the authentication mechanism you have chosen (an operating system user registry or LDAP directory service), but it does not need "root" or other special privileges.

After configuring the administrative identity, when you restart the administrative server and try to administer the product, you must login with the administrative identity when you are prompted for a user ID and password.

You can also configure the product security to allow administrative access by other IDs, in addition to the initial ID you established.

Setting up additional administrative accounts

During the installation of WebSphere Application Server, you must identify an existing account that will act as the first administrative account for WebSphere. After enabling security, this account will be the only one authorized to administer WebSphere. You can, however, use the account to authorize other administrative users.

To authorize other valid accounts defined in the operating system user registry or in your directory service product, use the AssignPermissions task on the Tasks tab of the WebSphere administrative console (in the Security task group). With this task, you can grant users access to the protected functions, which are listed in the format `AdminApplication-function_name` in the task.

Access to the administrative functions of the IBM WebSphere Application Server product is controlled by the

*admin*application, to which the functions belong.

Steps

1. Click the Tasks tab to display the Tasks tree.
2. Click Security --> Assign Permissions.
3. Click an AdminApplication-*function_name*function.
4. Click the Add button to produce a search dialog.
5. Use the search dialog to give permission to everyone or selected users or groups. You can search for a user or group in your local operating system user registry or directory service product.
6. Click the OK button when you are finished with the search dialog.
7. Back in the main console window, verify that the user or group is listed under the permission you granted to the user or group.
8. Exit this task by choosing another task on the Tasks tab.

Giving NT users administrative privileges

During the installation of WebSphere Application Server, you must identify an existing account that will act as the first administrative account for WebSphere. On Windows NT, the account must be a member of the Administrators group and must have the rights to "Log on as a service" and to "Act as part of the operating system."

To give an account these rights, follow this procedure:

1. Start the user manager for Windows NT or Domains and click Start --> Programs --> Administrative Tools (Common) --> User Manager.
2. Select Policies --> User Rights from the menu bar on the dialog box.
3. Check the Show Advanced User Rights check box in the dialog box.
4. From the list labeled Right:, select Log on as a service.
5. If the administrative account is not listed in the Grant To: list:
 - o Click Add.
 - o Click the Show Users button in the resulting dialog box.
 - o Select the individual User or Group.
 - o Click Add to include the account in the Add Names list.
 - o Click OK to exit the dialog box.
6. Click OK in the User Rights Policy dialog box.
7. Return to the second step and repeat the procedure, specifying the "Act as part of the operating system" right instead of the "Log on as a service" right.
8. Close the User Manager window.

If you then open the Services menu and modify the Log On As account for the service, the account you specify here will automatically be granted the "Log on as a service" right.

i Do not use an account whose name matches the name of your machine or Windows Domain as the administrative account. The WebSphere administrative server will not work in such a case.

Changing passwords for administrative accounts

Good security requires the periodic changing of passwords, and this includes those for your WebSphere administrative accounts. These passwords have to be changed in two places, in a particular order. If this is done incorrectly, it can create a situation in which the WebSphere administrative server cannot restart. This file describes the best way to change an administrative password.

Steps

1. Make sure the WebSphere administrative server is running. This is crucial. Do *not* change an administrative password unless the server is running.
2. Change the password in the user registry by using the utility for your operating system or LDAP service.
3. Login to the WebSphere administrative console using the new password. Attempts to use the old password will fail.
4. Click Security --> Specify Global Settings --> User Registry in the administrative console.
5. Change the password for the administrative user to the new password.
6. Stop and restart the administrative server.

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 wide range of needs. What is reasonable in one environment can be considered a breach of security in another.

In WebSphere, Web resources are not protected by default. Additionally, the WebSphere example applications install some demonstration servlets that perform administrative tasks. Because Web resources are not secured by default, and because these servlets perform work usually reserved for administrators, they represent a possible security problem, particularly for production applications.

The following describes some user practices and their potential risks. When applicable, it uses components of the example application to illustrate the point.

Serving static HTML files

Purpose: This servlet serves static HTML pages. For example, the "file" servlet in the default configuration under Web application called "examples" serves static pages from the application's document directory (e.g., `<WAS_HOME>/hosts/default_host/examples/web`). If you access `http://<host>/webapp/examples/index.html`, this servlet is invoked to serve the page.

Security consideration: If you place a file containing confidential information within these directories, someone who knows the filename but does not have access to it through the operating system can invoke this servlet to obtain the file.

Solution: Protect all the files by protecting the URI associated with the "file" servlet. Such a URI typically ends with a "/" (e.g., `/webapp/examples/`). Alternatively, if you want to protect only certain files (e.g., `/webapp/examples/test.html`) served by this servlet, then protect the files individually. To do this, you must:

1. Create a URI for each file.
2. Associate the URI with the file servlet by adding the URI to the web-path list of the servlet.
3. Secure access to the URI.

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.

The Web application "admin"

Purpose: To administer WebSphere configuration.

Security consideration: Currently, security is *not* supported for this feature, even if you have enabled security. If you install this application, anyone can change the configuration, even if WebSphere security is enabled.

Solution: The "admin" 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:**

The "unique key certificate" filter is offered as a security option in the WebSphere administrative console, but is not supported for Application Server Version 3.x.

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

Security is not automatically updated when changes are made to a bean. You must redeploy the resource security in order for the method groups to pick up the changes to the bean.

Adding a method to a bean

If you add a method to a bean, you must go back into resource security and associate the new method with a method group.

Modifying a method on a bean

If you modify a method on a bean, you must resecure the bean as follows:

1. Delete the method group for the bean.
2. Click **Finish**.
3. Re-associate the method group with the modified method.

Unprotecting resources

Resources protected under WebSphere can be unprotected, if necessary. Depending on the resources and how they are configured into applications, the techniques for removing security differ. This file describes how to remove security in the following situations:

- All resources associated with an enterprise application
- A particular bean associated with an enterprise application
- All URIs associated with a web application
- A particular URI associated with a web application

Unprotecting all resources associated with an enterprise application

If you want to remove protection from all the resources associated with an enterprise application, the most efficient approach is to unprotect the application itself. For example, if you have granted the permissions associated with the application ("*application-methodgroup*" pairs) to a specific user, group or to all authenticated users, the resources are considered protected. To unprotect these resources, you can grant those permissions to "Everyone". By granting the permissions to everyone, a user need not be authenticated to access the resources under that application.

Unprotecting an enterprise bean associated with an enterprise application

If you want to remove protection from a specific bean (or set of beans) associated with an application while maintaining the security on the other resources in the application, remove the bean (or beans) from the application and create a new application that is explicitly unprotected.

When you remove beans from the application, the security configuration associated with the application no

longer applies to them. However, enterprisebeans are protected unless security policies to the contrary are specified. To completely unsecure them, you need to create a new application consisting of the beans to be unsecured. After performing security configuration steps, grant the permissions associated with the new application to "Everyone." This is equivalent to unprotecting all the resources associated with the new application.

To remove resources from a secured enterprise application, use the "EditEnterprise Application" task. On the last panel, you can remove resources associated with the application. Use it to remove the desired beans.

Unprotecting all URIs associated with a web application

If you want to remove protection from a web application (including all associated URIs) while maintaining the security on the other resources in the enterprise application, remove the web application (or applications) from the enterprise application.

To remove resources from a secured enterprise application, use the "EditEnterprise Application" task. On the last panel, you can remove resources associated with the application. Use it to remove the desired web applications.

Unprotecting specific URIs

If you want to remove protection from specific URIs in a web application, remove the method-group configuration for the URIs. Use the "ConfigureSecurity Method Groups" task and select the URI you want to unprotect. After the URI is selected, proceed to the next screen, where you view the classification of methods into method groups. For example, the HTTP_GET method may belong to the ReadMethods method group. Select the method groups associated with the methods you want to unprotect and remove them. This eliminates the association between a method group and a URI, leaving the URI unprotected. Because web resources are unprotected by default, no authentication is required to access them.

Protecting individual JSP files

This file describes the steps necessary for selectively protecting JSP files, that is, how to protect individual JSP files based on their Web paths (URIs) when you do not want to apply the same protection to *all* the JSP files in the system.

Note, the instructions for adding a JSP Web path to a web application advise you to use the "Add a JSP or a web resource" task wizard in the administrative console. This action adds the JSP Web path, not the actual JSP file, to the Web application. But when you follow the configuration steps to protect a JSP Web path, the Web path is treated separately from the Web application; instead, it is treated as a Web server resource. Therefore, security does not work as intended.

The following procedure will be needed until product defect number 88065 is addressed. Check the "fixed defects" list accompanying IBM WebSphere Application Server fix packs to ascertain whether a given fix pack has addressed the defect.

To protect individual JSP files using WebSphere security, follow these steps:

1. If you used the "Add a JSP or web resource task" to introduce a new JSP Web path and associate with Web applications, remove all of the Web paths.
2. Start the [WebSphere administrative console](#).
3. Select the Topology view.
4. Expand the Topology tree to show the node, application server, and servlet engine containing the Web application to which you want to add the JSP.

5. Select the JSP processor servlet in that Web application.
6. In the list of Web paths, locate:
`/default_host/<webapp-path>/*.jsp`
where `default_host` is the default virtual host or one that you have created, and `<webapp-path>` is the path to the Web application.
7. Click "Add" to add to the Web path list.
8. Enter the JSP Web path (URI) that you want to protect, such as:
`/default_host/<webapp-path>/toBeProtected.jsp`.
If you have multiple files to protect, enter the URI for each one.
9. Apply your changes.
10. Follow the [resource security configuration steps](#) to protect these newly added JSP files.
11. Restart the application server hosting the Web application and JSP files.

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`
 - 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

During product installation, you must decide whether or not to check a box labeled "Use demo keyring file." Keyrings are used for certain types of authentication.

If you plan to produce your own keyrings, you do not need to check this check box. If you are not sure, check the box. This way, if you later need the functionality for testing, you'll have it. Your decision for this check box will *not* affect the overall success of the security installation.

i 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.

6.6.18.11: SecureWay Directory Version 2.1

- [Overview](#)
- [Software requirements](#)

Overview

Version 2.1 of the SecureWay Directory provides many new enhancements over its predecessor, eNetwork LDAP Directory Version 1.1.1, which was originally only available on AIX 4.3.1. The major enhancements include:

- DB2 Version 5.0 as the directory data storage facility
- Alias support
- Improved search and ACL support
- Support for popular Web servers
- Significant scalability
- Improvements to replication and performance

The IBM SecureWay Directory V2.1 includes an LDAP Version 2 server (RFC 1777, 1778, 1779). It is enhanced to support aliases and IETF LDAP Version 3 extensions for SSL, referrals, replication and access control.

SSL provides encryption of data and authentication using X.509v3 public-key certificates. The server may be configured to run with or without SSL support. The server supports LDAP referrals, allowing directories to be distributed across multiple LDAP servers. Replication is supported which makes additional read-only copies of the directory available, improving performance and reliability of access to the directory information. A powerful, easy-to-manage access control model is supported. Configuration and administration of the LDAP Directory is accomplished through an improved web-based interface.

This product is available on AIX, Windows NT/Intel and Solaris platforms. It currently supports ten languages including English, French, German, Italian, Spanish and Brazilian Portuguese on AIX and NT. Catalan is also supported on AIX. It does not support DBCS languages on Solaris.

The SecureWay Directory, Version 2.1 supports up to fifteen million entries with peak sub-second response time for searches.

Performance of the Directory is improved with statement caching and optimization. Multi-threading improvements allow Directory clients to perform true multi-threaded connections and make concurrent operations with the DB2 server. The DB2 program provided with the Directory may only be used by the SecureWay Directory function.

ACL support provides role-based authorization, assigns multiple user ownership of an entry, and removes the requirement to set ACLs on every node to give users access to their own information.

The Web servers Apache, Lotus Domino Go, Netscape FastTrack, and Netscape Enterprise Web servers are supported for LDAP administration.

Directory client access is supported using LDAP or HTTP protocols. Client applications can be developed using the enhanced elements provided for supporting LDAP Version 3 protocols and APIs. Also included is the Java Naming and Directory Interface (JNDI) client API that provides Java applications with access to LDAP-enabled directories. Both clients support access to SecureWay Directories using LDAP Version 2 or Version 3. Directory client applications can be built for Windows NT, Windows 95, Solaris, and HP-UX using the IBM LDAP Client Pack, which can be ordered separately (PRPQ 5799-GAN). Also shipped with the SecureWay Directory is Directory Sample 1, a client application that creates a directory for testing LDAP.

Directory Sample 1 is provided without support.

Also shipped with the SecureWay Directory is Directory Sample 1, a client application that creates a directory for testing LDAP. Directory Sample1 is provided without support.

Standards:

- RFC 1777 Lightweight Directory Access Protocol
- RFC 1778 String Representation of Standard Attribute Syntaxes
- RFC 1779 String Representation of Distinguished Names
- RFC 1823 LDAP Application Program Interface
- RFC 1960 A String Representation of LDAP Search Filters

Interoperability/Compatibility: The SecureWay Directory replication interoperates with the OS/390 LDAP Server.

Software requirements

The product supports three operating systems:

- Microsoft Windows NT (3) Workstation/Server Version 4.0 with service pack 3 or later (NTFS file system is required for security support)
- Sun Solaris Version 2.5.1 (SunOS 5.5.1) or Version 2.6 (SunOS 5.6) (4)
- IBM AIX Version 4.2.1 with APAR IX72127; or Versions 4.3.0 and 4.3.1 with APAR IX72439, IX74821, IX75022 and PTF U457544; or Version 4.3.2 (2)

The required IBM Universal Databases are:

- For NT, one of the following:
 - UDB V5.0 for Windows NT with fixpak US9044f
 - UDB V5.2 for Windows NT
- For Solaris, one of the following:
 - UDB V5.0 for Solaris with fixpak U457228f
 - UDB V5.2 for Solaris
- For AIX, one of the following:
 - UDB V5.0 for AIX with fixpak U457227f
 - UDB V5.2 for AIX

For Directory server, the requirements are:

- One of the following installed and configured Web servers:
 - Apache 1.2.5 or later
 - Lotus Domino Go Webserver 4.6.2 or later
 - Netscape FastTrack Server, Version 2.0.1 or later
 - Netscape Enterprise Server, Version 3.5.1 or later
 - Microsoft IIS 2.0
- Java Runtime 1.1.6
- A minimum of 64 MB RAM
- DB2 for AIX, Version 5.0.0.39, Workgroup Edition

Note: Lotus Domino Go Webserver 4.6.2.5 and Netscape FastTrackVersion 3.0.1 are available in the AIX Version 4.2 and 4.3 Bonus Packs. DB2 for AIX, Version 5.0.0.39 is included with the IBM SecureWay Directory. DB2 fixes to upgrade from Version 5.0.0.0 can be found on the Web at URL:

<http://www.software.ibm.com/data/db2/library>

The Directory client requires:

- AIX Version 4.2.1, Version 4.3, Version 4.3.1, or Version 4.3.2
- A frame-enabled browser that supports HTML Version 3.0, or later, and a browser that supports Java Runtime 1.1.6.

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.

Trace Output 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. 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, they actually refer to the standard error and standard out streams (such as an error file, or the screen, or so on).

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 Specification, Trace String

Specifies the application server components to trace.

On an administrative server property is called `traceString`.

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 classes or 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/disable action 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.

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.

6.6.19.1: Administering the product messages, logs, and traces

The Application Server product provides various levels of information, from high-level messages in the console to more detailed server execution logs and comprehensive tracing.

General approach to tracing a problem

You should not need to trace IBM WebSphere Application Server unless directed to do so by IBM support personnel. In case you would like to perform your own tracing, here is a procedure:

1. Determine the activity during which the problem usually occurs.
2. Configure an application server for a dump, but do not trigger the dump yet.
3. Perform the activity during which the problem usually occurs.
4. After the problem occurs, proceed with the dump.
5. Check the dump file, which will contain only the tracing performed between the time you specified the classes to trace and the moment when you dumped the trace data to the file.
6. Optionally, based on the dump contents, speculate which server classes or components are involved in the problem. Repeat the procedure, this time narrowing the scope of the trace to just those components, isolating the problem area.

Dump the server trace to a file

Sometimes the administrator might find it useful to dump the contents of the ring buffer that holds trace data for an administrative or application server.

1. Display the Topology tree.
2. To access the Trace Administration dialog box, right-click a running node or application server in the Topology tree and click Trace.
3. In the Trace Administration dialog box, specify the classes to trace. Verify the dump file name.
4. Now or later, click the Dump button.

To isolate the activity you are trying to trace, you might want to delay clicking the Dump button until after the server performs the activity. See the general trace approach section above for more details about this strategy.

Trace an administrative server from startup

To trace an administrative server from the time that it starts, modify its properties related to tracing:

1. Open the `admin.config` file in the `bin` directory of the WebSphere Application Server installation root.
2. Set the `traceString` property to specify the server classes to trace. For valid syntax, see the trace properties help.
3. Set the `traceOutput` property to specify the file to which to send the trace data. For valid syntax, see the [server trace properties help](#).
4. Save the file.
5. Start the server (or stop it and start it again) for the new trace parameters to take effect.

Trace an application server from startup

To trace an application server from the time that it starts, modify its properties related to tracing:

1. Display the Topology tree.
2. In the tree, locate and click the application server to display its properties on the right side of the console.
3. Set the trace-related properties. Consult the [server trace properties help](#) for syntax.
4. Save the property changes.
5. Start the server (or stop it and start it again) for the new trace parameters to take effect.

6.6.19.1.1: Administering messages with the Java administrative console

Please select a sub-topic.

6.6.19.1.1.1: Filtering messages with the Java administrative console

Use the Console -> Trace menu on the console menu bar to specify which classes of the Java console (WebSphere AdministrativeConsole) to include in the console messages output.

6.6.19.1.1.2: Collecting serious events with the Java administrative console

Click Console -> Trace -> Serious Events to display the Serious Events Viewer. This viewer displays records of the last several serious events to occur since the administrative server was started.

Use the Preferences tabbed page to specify Serious Event Viewer properties, such as how many event records to keep at one time and how often to update the messages on the console with data from the administrative server.

 The Serious Event Viewer "Log Limit" property specifies how many serious event records to keep. They are stored in the administrative repository (database). If your database is becoming too full, set the Log Limit to the minimum value that is reasonable for your environment.

6.6.19.1.2: Viewing logs and messages

Please select a sub-topic.

6.6.19.1.2.1: Viewing messages with the Java administrative console

The Console Messages area is located in the bottom part of the WebSphere AdministrativeConsole. Resize or scroll through the area as needed to review message events that provide a good high-level indication of administrative domain health.

6.6.19.1.2.2: Viewing serious events with the Java administrative console

Click Console -> Trace -> Serious Events to display the Serious Events Viewer. This viewer displays records of the last several serious events to occur since the administrative server was started.

Use the History tabbed page to browse through serious event records.

6.6.19.1.2.3: Viewing logs

[Article 8.3 in the problem determination section](#) provides instructions for viewing and interpreting logs.

6.6.19.1.2.4: Viewing traces

The administrator specifies the location of trace output for various servers. The table summarizes where to find trace output:

Type of output	Property that specifies location
Application or administrative server dump	Trace Administration dialog box
Administrative server trace	traceOutput property in admin.config file in bin directory of product installation root
Application server trace	Trace output file property of the application server

See the article about collecting traces provide detailed instructions for accessing the above-mentioned trace properties and dialogs.

6.6.19.3: Administering server traces with the Web console

The Web console does not support configuration of the trace service. However, you can define standard in, out, and error logs for new and existing application servers. See [Administering application servers with the Web console](#).

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

General tabbed page

Application is recoverable

Indicates whether the application is recoverable. Recoverable processes (sometimes called recoverable applications) have the ability to undo operations on data when a transaction aborts, and they can recover modifications to data in the case of a system failure.

Local ID

Indicates a Transaction identifier (TID), a unique ID assigned to each transaction to identify all actions associated with that transaction. Each TID is unique to a given server and is known only by that server.

Global ID

Indicates an identifier that ties a transaction to all of its participating applications across different servers.

Top Level

Indicates a transaction that does not run within the scope of another transaction. A top-level transaction is the root of a transaction family, even if it is the only transaction in the family.

Local State

Indicates the state of a subtransaction visible only to its parent transaction.

Global State

Indicates the state of a transaction from the perspective of the transaction coordinator. This state is based on the local state of all the participants of the transaction.

Beginner Application

Indicates the application that originated the transaction.

Family tabbed page

A transaction family consists of all nested transactions that share a common ancestor. All members of a transaction family commit together and drop their locks simultaneously.

Parent

Indicates a transaction that starts another transaction. Subtransactions are called child transactions.

Top Ancestor

When transactions are nested, this value indicates the transaction that is the parent of the entire tree (family) of transactions.

Descendants

Indicates all subtransactions (child transactions) in a family of transactions.

Family Begin Time

Indicates the time that the transaction family was started.

Properties tabbed page

Application properties

Indicates the application properties associated with the transaction. These properties are for use by IBM support only.

System properties

Indicates the system properties associated with the transaction. These properties are for use by IBM support only.

Participants tabbed page

A participant is an application that takes part in a transaction. An application is considered a participant in a transaction if it either initiates the transaction or receives a request on behalf of the transaction.

Direct participant list

Indicates the applications that are direct participants in the transaction.

Indirect participant list

Indicates the applications that are indirect participants in the transaction.

Family participant list

Indicates the transaction family members participating in the transaction.

6.6.28: Administering IBM Distributed Debugger and OLT

IBM Distributed Debugger for Workstations is a client/server application that enables you to detect and diagnose errors in your programs. This client/server design makes it possible to debug programs running on systems accessible through a network connection. You can also debug programs running on your local workstation (assuming they are capable of being debugged).

The graphical IBM Distributed Debugger interface allows you to debug programs. The Object Level Trace portion of the Distributed Debugger allows you to monitor the flow of a distributed application and debug the client and server code from a single workstation.

Note, Distributed Debugger and OLT may not support all operating systems supported by this version of IBM WebSphere Application Server. See article 1.2.7.2.1 for the latest information, as well as known limitations, problems, and workarounds.

6.6.28.1: Installing Debugger and OLT code and documentation

This article describes how to obtain the Debugger and OLT products that are most compatible with IBM WebSphere Application Server, and what to do if you have previous versions of Debugger and OLT. It also has important instructions for obtaining updated and translated help files for the interfaces of the two products.

- [Obtaining the Debugger and OLT servers and clients](#)
- [Viewing the latest Debugger and OLT help files and documentation](#)
- [Prerequisites of Debugger and OLT](#)
- [Known limitations, problems, and workarounds](#)

Obtaining the Debugger and OLT servers and clients

The IBM WebSphere Application Server product CD provides the OLT server-side runtime. The Debugger and OLT graphical user interfaces must be obtained and installed separately.

Installing the Debugger and OLT runtimes from the WebSphere CD

It is recommended that you install the Debugger and OLT runtimes using the IBM WebSphere Application Server Version 3.5 installation program available on the Application Server product CD. This ensures compatibility and recognition of one another among the Debugger, OLT, and WebSphere Application Server.

You need to perform the **Full** Debugger and OLT installation, as described in the installation README file and installation example.

Handling existing Debugger and OLT installations

If using existing Debugger or OLT versions, and you have an IBM WebSphere Application Server CD, it is recommended that you uninstall or upgrade the existing versions and migrate to the versions available from the Application Server Version 3.5 installation program.

If you are using the Application Server download, from which the new Debugger and OLT versions are not available, then you will need to keep your previous versions.

Obtaining the Debugger and OLT client interfaces

The client interfaces are available as part of the OLT product package available from the IBM WebSphere Application Server Web site. Be sure to review the accompanying README file that discusses basic installation and usability issues.

Viewing the latest Debugger and OLT help files and documentation

The Debugger and OLT installation images contain help files that can be accessed from their product interfaces. By default, the installed products contain help files that are less updated than the Debugger and OLT help available from the InfoCenter, and that are available in English only.

If Debugger and OLT are installed on a machine containing the IBM WebSphere Application Server InfoCenter in its default location, the Debugger and OLT products will ignore the outdated help files and instead use the help files contained in the InfoCenter, in the language in which you installed the InfoCenter.

To ensure you are viewing the updated, translated help files:

- Install Debugger and OLT from the Application Server product CD.
- Install the full InfoCenter on the same machine, in its default location (you might not have a choice):

`product_installation_root/web/InfoCenter/...`

Complete the above steps in either order, but do not access Debugger or OLT help from the Debugger or OLT interfaces until both steps are complete.

Obtaining the Debugger and OLT documentation

In addition to the Debugger and OLT help files, the full InfoCenter contains the entire Debugger and OLT documentation set. It is the version to which you should refer if you installed Debugger and OLT from the WebSphere Application Server installation program.

To access the documentation:

1. [Make sure you have obtained the full InfoCenter](#), which has an "olt" directory containing the Debugger and OLT documentation and help.
2. In your local WebSphere Application Server installation, open:

`product_installation_root/web/InfoCenter/olt/index.html`

You can also browse the online InfoCenter on the product Library page. See the [README file](#) for details.

Prerequisites of IBM Distributed Debugger and OLT

IBM Distributed Debugger and OLT are separate product runtimes bundled with IBM WebSphere Application Server for your convenience. Because the Distributed Debugger and OLT support for operating systems and JDK levels can differ from those supported by IBM WebSphere Application Server, the

Debugger and OLT support is summarized in the table below.

Inclusion of a prerequisite in the table does not imply that IBM WebSphere Application Server also supports the prerequisite. Always refer to the [IBM WebSphere Application Server prerequisites](#) to determine which prerequisites are supported by the base Application Server product and its Fix Packs.

IBM WebSphere Application Server Release	Operating systems supported by Debugger and OLT
Version 3.5 base	Remote and distributed debugging support and tracing capabilities, as follows: <ul style="list-style-type: none"> Windows NT Java debugging is supported on JDK 1.2.2 AIX and Solaris Java debugging is not supported on JDK 1.2.2
Version 3.5 plus Fix Pack 1	Remote and distributed Javadebugging support and tracing capabilities, as follows: <ul style="list-style-type: none"> Windows NT Java debugging is supported on JDK 1.2.2 AIX and Solaris Java debugging is not supported on JDK 1.2.2 Tracing capabilities are supported on JDK 1.2.2 on Windows NT, AIX, and Solaris
Version 3.5 plus Fix Pack 2	<ul style="list-style-type: none"> No changes
Version 3.5 plus Fix Pack 3	Remote and distributed Javadebugging support and tracing capabilities, as follows: <ul style="list-style-type: none"> Windows NT and AIX Java debugging is supported on JDK 1.2.2 Solaris Java debugging is supported on JDK 1.1.8 but not JDK 1.2.2 Tracing capabilities are supported on JDK 1.2.2 on Windows NT, AIX, and Solaris

Known limitations, problems, and workarounds

Check the following settings:

- On Windows NT, ensure that you configure the service IBM WebSphereAdmin Server to interact with the desktop.
- Ensure that there are no conflicting JRE processes running on the application server platform. There might be a JRE process invocation of the debugger.

Problem	Operating systems	Workaround (if available)
When debugging a JSP (1.0 OR 0.91) Distributed Debugger does not step over empty lines. For example, when stepping into a JSP like: Line 1: Hello Line 2: Line 3: World! the debugger stops at line 2.	AIX, Windows NT	None
When running the OLT Viewer, if you try to change the Execution Mode in the Client Controller for a connected client and then click Apply , the viewer might freeze.	Windows 2000 for WebSphere Application Server Versions 3.5, 3.5.1, and 3.5.2 (fixed in 3.5.3)	Ensure that you have set the Execution Mode properly in the Client Controller for the Default settings entry before you try tracing. Do not attempt to change the Execution Mode after you start a trace.
The OLT executable fails	AIX	If the OLT executable fails on AIX, edit the libpath environment variable: <ol style="list-style-type: none"> Source the OLT profiles in the /usr/idebug/bin directory. Ensure that the following information is in the libpath: LIBPATH=\$LIBPATH: 'usr/idebug/lib' export LIBPATH
The following error appears in the server's stderr file: IVB3643E: The Debugger/OLT encountered an IO error while communicating with the client controller.java.net.SocketException	All supported	
The debugger does not start when a debug-enabled application is requested	All supported	The problem might be due to inconsistent OLT properties. To fix the problem: <ol style="list-style-type: none"> Stop the application server. Delete the OLT DbgProf directory. On AIX, it is in the home directory. On NT, it is in the userprofiles directory. Open a command shell on the application server and run the olt executable. This action resets the DbgProf directory. Exit the olt program after it starts. Restart the application server. Debug the application.

6.6.28.1.1: Installing Debugger and OLT clients (README)

These instructions were added for IBM WebSphere Application Server Version 3.5.3. If using an earlier version, consult the information about installing Debugger and OLT code and documentation, and the [Prerequisites Webpage](#) to find out which operating systems are supported by your current level of IBM WebSphere Application Server and its accompanying Debugger and OLT code.

IBM Distributed Debugger and Object Level

```
TraceINSTALLATION=====1 General remarks about
installation2 Installation on Windows (R)3 Installation on Linux4 Installation on AIX5 Installation
on Solaris1 General remarks about installation-----The following
remarks regarding installation and prerequisites are applicable on all platforms for the installation
of the IBM Distributed Debugger and Object Level Trace tools. Additional prerequisites and
platform-specific installation issues are mentioned in the subsections for each platform.General
prerequisites:Netscape Navigator, Version 4.5 or later is recommended for displaying debugger online
help. Using an earlier version may give you a browser exception if you reopen a help window multiple
times.For interpreted debugging, the following additional prerequisites apply:* You should install
the appropriate level of the JDK, as specified by the product with which the Distributed Debugger and
Object Level Trace shipped.* If you install this version of the debugger over WebSphere Application
Server V3.02 on Windows NT, then you must go into the IBMDebug\bindirectory and rename ivbtr30i.302
to ivbtr30i.dll.2 Installation on Windows (R)-----Prerequisites:For
standalone debugging, the location of Java executable specified in the PATH environment variable must
be from the JDK "bin" directory.For example, if, on Windows NT, the PATH environment variable
points to the java.exe in the \WINNT\SYSTEM32 directory, the debugger cannot debug the application. To
avoid this problem, insert the full path to java.exe in the JDK bin directory to the start of the
PATH environment variable.If you are using remote stored procedure debugging on DB2, you must add the
\IBMDebug\lib\dertrjrt.jar file to your system CLASSPATH.Installation:Installing the Distributed
Debugger on Windows is automatic and does not require you to manually issue any commands or move any
files. Simply download oldtbg.zip, unzip it, and double click on the setup icon to start the
installation process.Any beta version of the IBM Distributed Debugger must be uninstalled before
installing the current version that accompanies this product.The debugger will not function if
installed in a directory whose name contains DBCS characters.3 Installation on
Linux----- (1) The JDK 1.2.2 is required by the Distributed Debugger.The location
of Java executable specified in the PATH environment variable must be from the JDK "bin" directory.
For example, if the PATH environment variable points to java in the /usr/local/bin directory, the
debugger cannot debug the application. To avoid this problem, insert the full path to java in the JDK
bin directory to the start of the PATH environment variable.(2) Some JDK implementations may
reference fonts that may not be installed or accessible. When this situation occurs, a message
similar to the following may appear when the Debugger or any other graphical Java application is
run:Font specified in font.properties not found [--zapf dingbats-medium-r-
normal--*-%d-*-*p-*-*adobe-fo]These messages do not indicate a fatal condition. Usually these missing
font warnings are soft errors that do not impede running the application. To correct the "font not
found" problem, visit the JDK supplier's website. Solutions to this issue may include
installing missing fonts and/or updating the JDK/JRE font.properties file.Installation:1. Download
oldtbg.tar2. su to root if you are not logged in as root already.3. Go to your download directory.4.
From a command prompt, issue the following commands:tar -xvf oldtbg.tar.rpm -Unh
DERJPICL-9-0.i386.rpm4 Installation on AIX-----Prerequisites:The IBM JDK 1.2.2 PTF9
is a prerequisite for interpreted debugging on this platform. If you are using remote stored
procedure debugging on DB2, you must add the /usr/lpp/idebug/lib/dertrjrt.jar file to your
system CLASSPATH.Installation:Search for the install image of the distributed debugger. Copy
the install file(s) to the machine on which the debugger is to be installed. Use installp or 'smitty
install_latest'.To use the IBM Distributed Debugger on AIX, you must add the following lines to your
.profile:# Start the OLT login profile if [ -f /usr/idebug/bin/OLT.profile ]; then..
/usr/idebug/bin/OLT.profile fi5 Installation on Solaris-----Prerequisite:If you
are using remote stored procedure debugging on DB2, you must add the /opt/IBMdebug/lib/dertrjrt.jar
file to your system CLASSPATH.Installation:Download the solaris oldtbg.zip image. Unzip the image.
Issue the command "./dbgsetup"
```

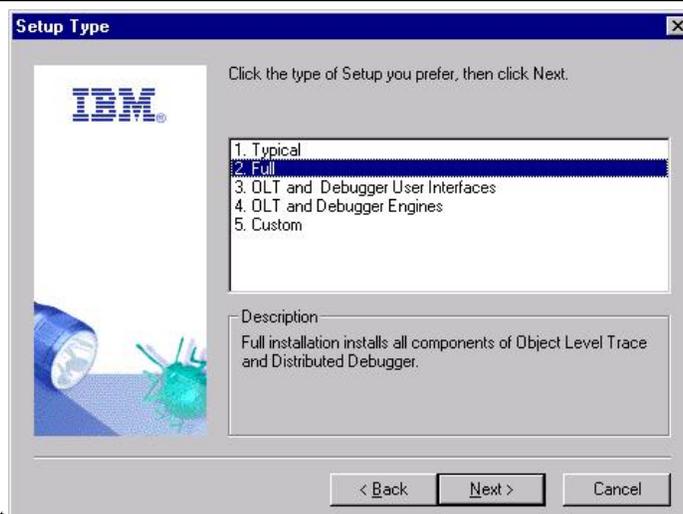
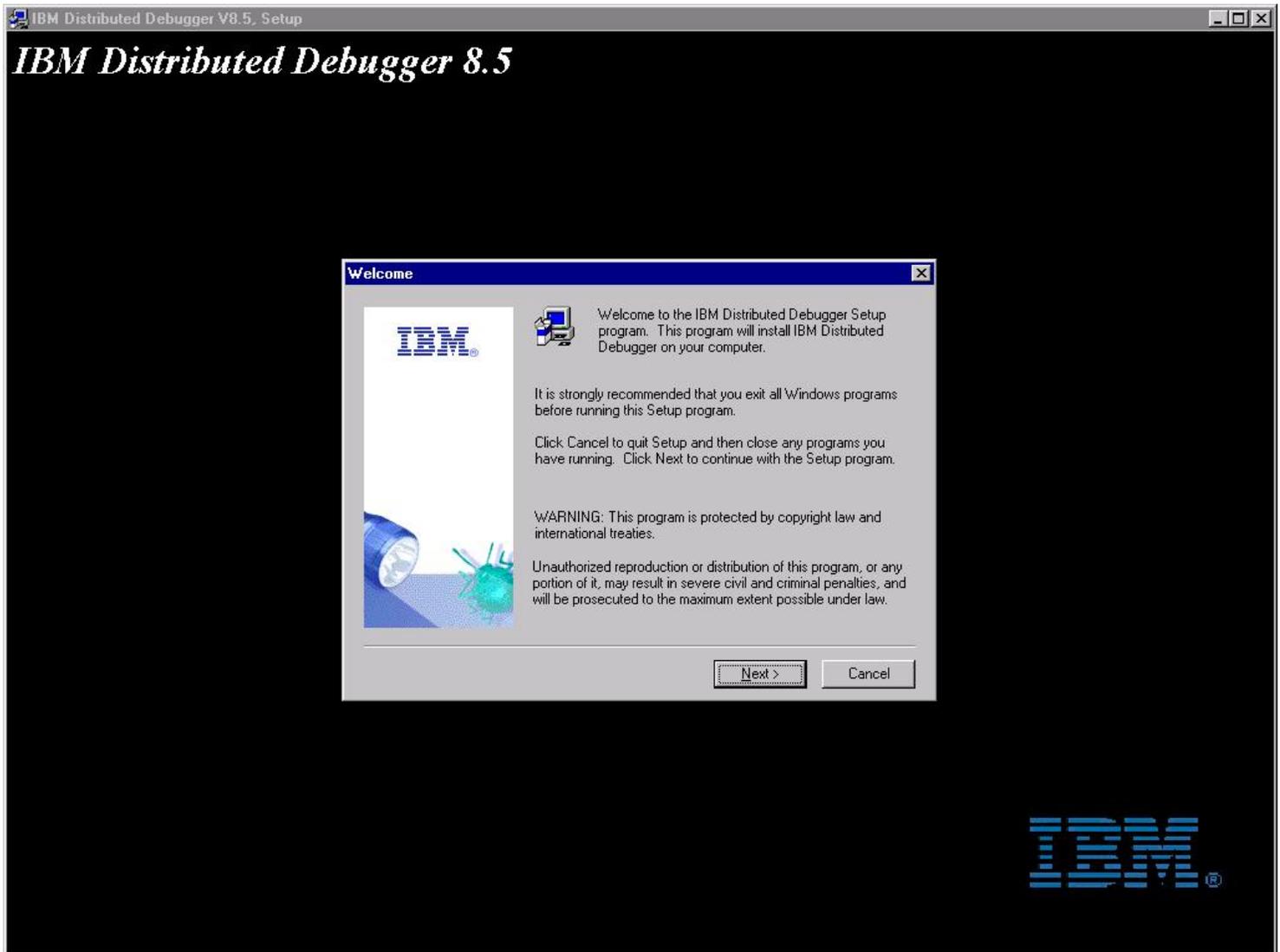
6.6.28.1.2: Example: Installing OLT and Debugger clients on Windows NT

This article provides detailed instructions for installing the IBM Distributed Debugger and Object Level Trace (OLT) clients on Windows NT.

1. Run the OLT/Debugger installation program on the machine where you have IBM WebSphere Application Server Advanced or Standard Editions installed. This will be the machine where you will view the traces and perform the debugging.

Note, you should follow the recommendation in the installation dialog about shutting down all other Windows programs before attempting the installation.

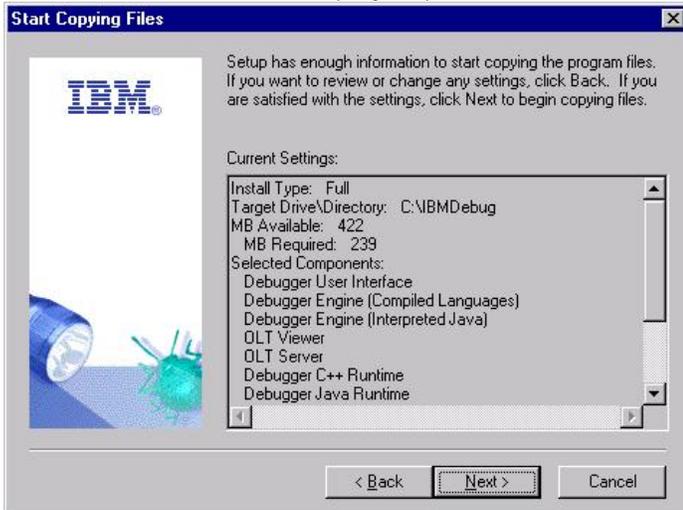
On the initial installation screen, click Next.



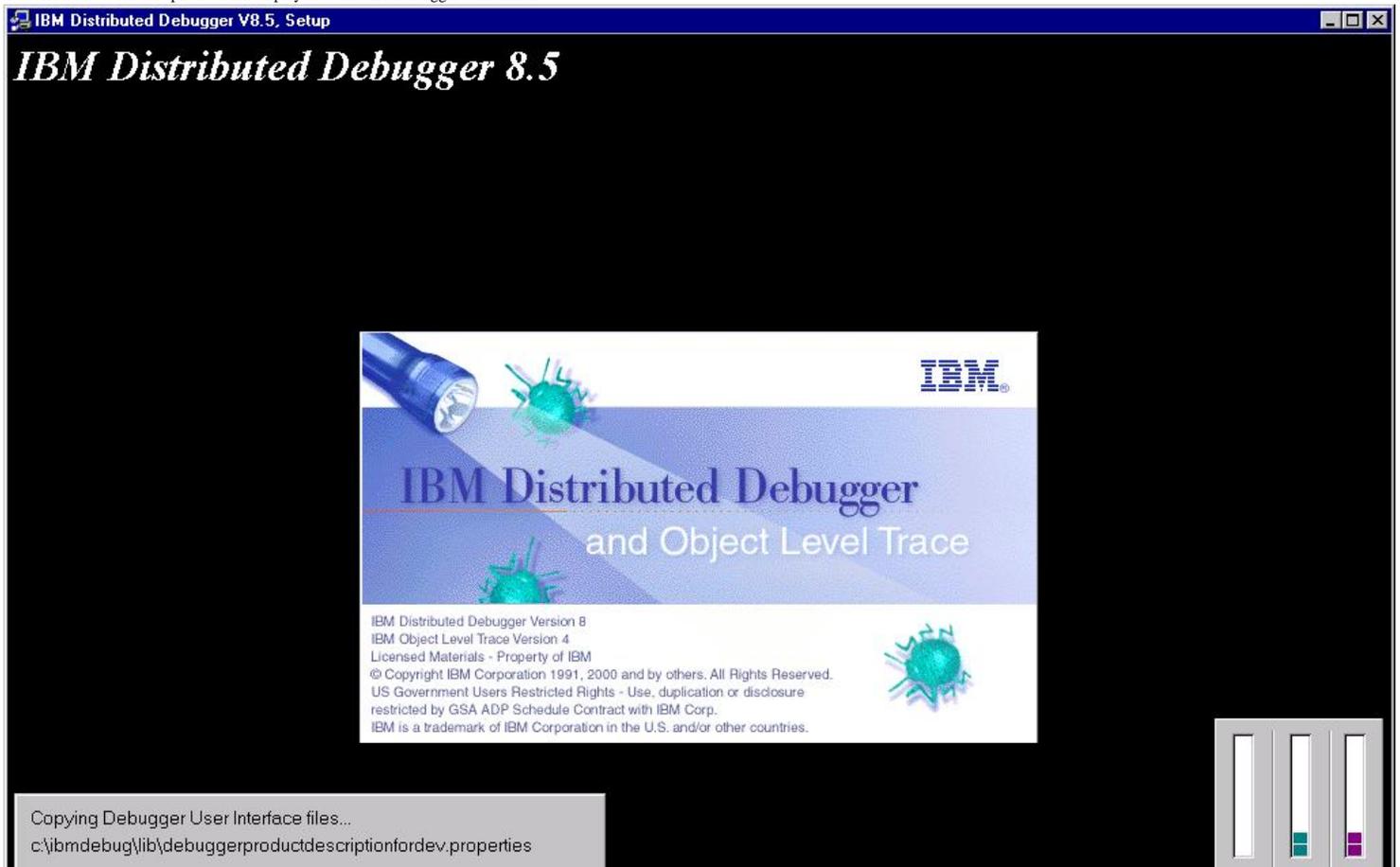
2. Choose the Full option, then click Next.
3. You can change the default location where OLT/Debugger is installed by clicking Browse. When ready to proceed, click Next.



4. The next screen is a confirmation screen. If everything is okay, click Next. Otherwise, click the Back button to return to previous screens on which you can make changes.

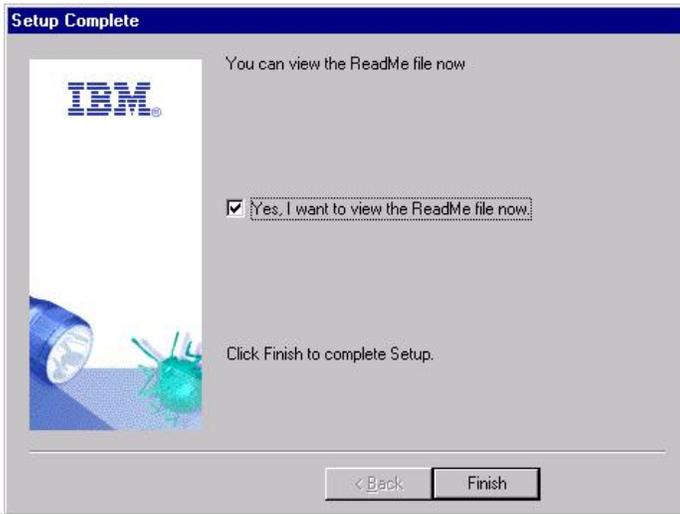


5. You should now see the splash screen displayed as the OLT/Debugger is installed:

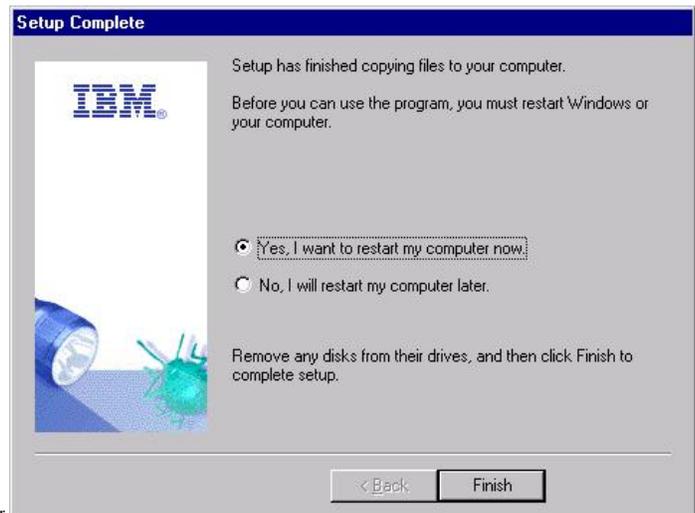




6. After successful installation, you will be asked if you want to view the README file. It is recommended you do so, to learn late-breaking product information.



To proceed, click Finish.



7. You will be asked whether to reboot your computer. You must reboot before you use OLT/Debugger.

8. To ensure your new installation is accessing the most updated Debugger/OLT documentation and help, make sure the InfoCenter is installed on the same machine.

6.6.28.2: Using Debugger and OLT

This article provides instructions for using IBM Distributed Debugger and Object Level Trace (OLT) in a WebSphere Application Server environment.

- [Before you trace or debug](#)
- [Procedural overview](#)
- [Detailed steps](#)
- [Examples of debugging local applications](#)
- [Examples of tracing local applications](#)
- [Examples of debugging remote applications](#)
- [Examples of tracing remote applications](#)
- [Troubleshooting tips](#)

Before you trace or debug

When compiling code, be sure to include debugging information by using the `-g` command line argument. For example:

```
javac -g myClass.java
```

Procedural overview

Here is an overview of the procedure for using tracing and debugging, with steps that link to detailed instructions below:

1. [Configure the application server Debug properties](#)
2. [Start the application server](#)
3. [Optionally, minimize the WebSphere Administrative Console](#)
4. [Start the Web server](#)
5. [Start the OLT or Debugger/OLT client and adjust settings](#)

For a JSP .91 file, copy the file to Web server HTML directory

6. [Access the application or Java component that you are tracing or debugging](#)
7. [Trace or debug the code in the Debugger/OLT](#)

To reiterate, the steps **must** be performed in order. Specifically:

1. Configure an application server's debugging properties and start it
2. Start the OLT executable on the machine where you want to view OLT, debug output
3. Request a resource or application deployed in the application server

Detailed steps

Each step in the procedure is now discussed in detail.

Step	Description
1: Configure the application server Debug properties	<p>To trace or debug code running on an application server, you must enable tracing or debugging on the server. The OLT and debug enabled buttons are on the Debug tab.</p> <ol style="list-style-type: none">1. Click the Topology tab to display the Topology tree.2. Click the application server whose OLT tracing and debugging settings you want to specify. The server properties are displayed on the right side of the console.3. Configure the application server:<ul style="list-style-type: none">○ To enable tracing, select the Object Level Tracing enabled check box.○ To enable debugging, select the Object Level Tracing enabled and Debug enabled check boxes.4. Save the property changes. <p>Now you will be able to trace or debug all servlets and JavaServer Pages (JSP) files deployed in the application server, according to the application server Debug properties you specified.</p>
2: Start the application server	<ol style="list-style-type: none">1. Display the Topology tree view.2. Expand the tree to locate the application server on which to perform tracing or debugging.3. Right-click the server and click Start.
3: Optionally, minimize the WebSphere Administrative Console	<p>When finished configuring and starting the application servers in the WebSphere Administrative Console, you can minimize the console because the rest of the tracing or debugging procedure is conducted outside of the console.</p>
4: Start the Web server	<p>Tracing and debugging in the Debugger and OLT client interfaces begin when the Java component being traced is accessed by a user at a Web browser. Your Web server must be started in order for this to work correctly.</p>

5: Start the Debugger/OLT client and adjust settings	<p>To start the client interface:</p> <ol style="list-style-type: none"> 1. Open a system command window. 2. Issue the command: olt <p>Ensure that the default execution mode is set to trace and debug in the ClientController. Set the checkbox to use the default settings.</p>
6: Access the application or Java component that you are tracing or debugging	<p>Make a request for Java component or application that you are tracing or debugging, as though you are atypical user. Information is sent to the Debugger/OLT.</p>
7: Trace or debug the code in the Debugger/OLT client interfaces	<p>When the Debugger/OLT clients are first displayed, you are prompted to provide the location of the source code for the Java component or application you are tracing or debugging. After you supply the information, you can start tracing or debugging the code in the Debugger/OLT clients.</p>

Examples of debugging local applications

- [Debug a servlet](#)
- [Debug a JSP .91 file](#)
- [Debug a JSP 1.0 file](#)

Debugging the Snoop servlet -- local Debugger/OLT client

1. Configure the application server containing Snoop servlet for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the Debugenabled and Object Level Trace enabled check boxes.
 3. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server (or other application server you are using) is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the debugging interface.
4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a system command window.

2. Issue the command:

```
olt
```

This starts the OLT daemon, which listens on a port and displays the debugger interface when a request arrives.

Ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.

6. In a Web browser, request the Snoop servlet:

```
http://localhost/servlet/snoop
```

7. Use the Debugger/OLT:

1. Watch for the Debugger/OLT. The debugger interface is displayed in response to your requesting a servlet running on an application server (default server) for which debugging is enabled.

When the interface is displayed, make sure that Options -> Step by Step debugging mode is set.

The debugger will find source files that are in the classpath of the JVM running the code being debugged. If it does not find a source file, it will prompt you for the location of the file, `HttpServlet.java` in this case.

`HttpServlet` is the parent class of the Snoop servlet, and is the class in which the `service()` method is implemented. Starting with the requested servlet, the debugger will look for the first `service()` method implementation it can find. Because Snoop does not implement `service()`, the debugger looked in Snoop's parent class, `HttpServlet`.

2. When prompted for the source path, browse `<server root>/hosts/default_host/default_app/servlets`.
3. Watch the Debugger/OLT, which should stop in the `service()` method. To step into the Snoop servlet, perform a series of stepovers until you get to the `doGet()` method call. Do a step into on this method to reach Snoop.
4. Watch for the debugger to break in the service method of the Snoop servlet. You can step debug, or simply click Run to complete the request.

Debugging very_simple.jsp (JSP .91) -- local OLT client

1. Configure the application server containing the JSP file for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the Debug enabled and Object Level Trace enabled check boxes.
 3. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on

your desktop for the debugging interface.

4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command:
`olt`

This starts the OLT daemon, which listens on a port and displays the debugger interface when a request arrives.

Ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.

6. Copy the file `<install_root>/hosts/default_host/default_app/web/very_simple.jsp` to the document root of the Web server.
7. In a Web browser, request the very_simple JavaServer Pages (JSP) file:
`http://localhost/very_simple.jsp`
8. Use the Debugger/OLT:
 1. Watch for the Debugger/OLT.

When the interface is displayed, make sure that Options -> Step by Step debugging mode is set.

You will likely be prompted for the location of the source file `pagecompile/_very_simple_xjsp.java`.

2. Browse for and select the file, which is located in the directory `<install_root>/hosts/default_host/default_app/web/very_simple_jsp.java`
3. Watch for the debugger to break in the service method of the `very_simple.jsp`. You can step debug, or simply click Run to complete the request.

Debugging very_simple.jsp (JSP 1.0) -- local OLT client

1. Configure the application server containing the JSP file for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the Debug enabled and Object Level Trace enabled check boxes.
 3. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the debugging interface.

4. Make sure your Web server is started.

5. Start the Debugger/OLT:

1. Open a system command window.

2. Issue the command:

```
olt
```

This starts the OLT daemon, which listens on a port and displays the debugger interface when a request arrives.

Ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.

6. Copy the file `<install_root>/hosts/default_host/default_app/web/very_simple.jsp` to the document root of the Web server.

7. In a Web browser, request the very_simple JavaServer Pages (JSP) file:

```
http://localhost/very_simple.jsp
```

8. Use the Debugger/OLT:

1. Watch for the Debugger/OLT.

When the interface is displayed, make sure that Options -> Step by Step debugging mode is set.

You will likely be prompted for the location of the source file `pagecompile/_very_simple_xjsp.java`.

2. Browse for and select the file, which is located in the directory `<install_root>/hosts/default_host/default_app/pagecompile/_very_simple_xjsp.java`

3. Watch for the debugger to break in the service method of the `very_simple.jsp`. You can step debug, or simply click Run to complete the request.

Debugging the Hello enterprise bean -- local OLT client

1. Configure the application server containing the enterprise bean for debugging:

1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.

2. For the default server's debugging properties, select the Debug enabled and Object Level Trace enabled check boxes.

3. Save the properties.

2. If you have not already done so, install and deploy the Hello enterprise bean:

1. In the Topology tree, find and right-click the Default Container in the Default Server.

2. In the resulting menu, click Create -> Enterprise Bean.

3. In the resulting window, click Browse.

4. In the Browse dialog, browse for `<install_root>/deployableEJBs/Hello.jar`. Double-click it to display its deployment descriptor. Click the deployment descriptor and exit the Browse dialog.

5. Back in the dialog window for creating an enterprise bean, click Create.
 6. When the deployment process is complete, verify that the Helloenterprise bean now is displayed in the Default Container on theTopology tree.
3. Start the application server:
 1. In the Topology tree, make sure the Default Serveris selected.
 2. Start it. Either click the Start toolbar button or usethe Start option on the server's right-click menu.
 4. Make sure your Web server is started.
 5. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command:


```
olt
```
 3. This starts the OLT daemon, which listens on a port and displays thedebugger interface when a request arrives.

Ensure that the default execution mode is set to trace and debugin the Client Controller page. Set the check box to use the defaultsettings.
 6. Run the Hello sample. On Windows NT, click Start -> Programs ->IBM WebSphere -> Application Server version 3.x.0 -> Samples,then select the Hello enterprise bean.
 7. Use the Debugger/OLT:
 1. Watch for the Debugger/OLT.

When the interface is displayed, make sure that Options -> Step by Step debugging mode is set.

You will likely be prompted for the location of the source file
_WebSphereSamples/_EJBs/_HelloEJB/_HelloEJB.jsp.
 2. Browse for and select the file, which is located in the directory
<install_root>/WebSphereSamples/EJBs/HelloEJB/HelloEJB.jsp
 3. Watch for the debugger to break in the service method of theHelloEJB.jsp. You can step debug, or simply click Run tocomplete the request.

Examples of tracing local applications

- Trace a servlet
- Trace a JSP .91 file
- Trace a JSP 1.0 file

Tracing the Snoop servlet -- local Debugger/OLT client

1. Configure the application server containing Snoop servlet for tracing:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties willbe displayed on the right side of the console.

2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
3. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command to start the Debugger/OLT:

```
olt
```

This starts the Debugger/OLT, which will display the trace information generated by the objects that service your request.
6. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.
7. In a Web browser, request the Snoop servlet:

```
http://localhost/servlet/snoop
```
8. Watch the Debugger/OLT for a line representing Snoop servlet. It will be labeled HttpServlet because that is the class that implements the service method. (HttpServlet is the parent class of Snoop servlet). On the line, you should see circles that represent the beginning and end of the service method.

Tracing very_simple.jsp (JSP .91) -- local Debugger/OLT client

1. Configure the application server containing the JSP file for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
 3. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
4. Make sure your Web server is started.

5. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command to start the debugger daemon:

```
olt
```
6. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.
7. Copy the file `<install_root>/hosts/default_host/default_app/web/very_simple.jsp` to the document root of your Web server.
8. In a Web browser, request the very_simple JSP file:

```
http://localhost/very_simple.jsp
```
9. Watch the Debugger/OLT for a line representing very_simple.jsp. On the line, you should see circles that represent the beginning and end of the service method.

Tracing very_simple.jsp (JSP 1.0) -- local Debugger/OLT client

1. Configure the application server containing the JSP file for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
 3. Click Apply.
2. Start the application server:

In the Topology tree, make sure the Default Server is selected.

 1. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command to start the debugger daemon:

```
olt
```
6. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.
7. Copy the file `<install_root>/hosts/default_host/default_app/web/very_simple.jsp` to the document root of your Web server.
8. In a Web browser, request the very_simple JSP file:

```
http://localhost/admin/very_simple.jsp
```
9. Watch the Debugger/OLT for a line representing very_simple.jsp. On the line, you should see circles that

represent the beginning and end of the service method.

Tracing the Hello enterprise bean sample --local Debugger/OLT client

1. Configure the application server containing the enterprise bean for tracing:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
 3. Click Apply.
2. If you have not already, install and deploy the Hello enterprise bean:
 1. In the Topology tree, find and right-click the Default Container in the Default Server.
 2. In the resulting menu, click Create -> Enterprise Bean.
 3. In the resulting window, click Browse.
 4. In the Browse dialog, browse for <install_root>/deployableEJBs/Hello.jar. Double-click it to display its deployment descriptor. Click the deployment descriptor and exit the Browse dialog.
 5. Back in the dialog window for creating an enterprise bean, click Create.
 6. When the deployment process is complete, verify that the Hello enterprise bean now is displayed in the Default Container on the Topology tree.
3. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
4. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
5. Make sure your Web server is started.
6. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command to start the debugger daemon:
`olt`
7. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.
8. Run the Hello sample. On Windows NT, click Start -> Programs -> IBM WebSphere -> Application Server version 3.x.0 -> Samples, then select the Hello enterprise bean.
9. Watch the Debugger/OLT for lines representing the HelloEJB JSP file and the Hello enterprise bean itself.

Examples of debugging remote applications

- Debug a servlet
- Debug a JSP .91 file
- Debug a JSP 1.0 file

Debugging snoop servlet from a remote machine

1. Configure the application server containing the servlet for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed in the right side of the console.
 2. For the default server's debugging properties, select Debug enabled and OLT enabled check boxes.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.
2. Start the application server.
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.

3. Make sure your Web server is started.

4. Start the Debugger/OLT:

1. Open a system command window on the remote machine.
2. Issue the command to start the debugger daemon:

```
olt
```

This starts the OLT daemon, which listens on a port and displays the debug user interface when a request arrives.

5. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.

6. In a Web browser, request the Snoop servlet:

```
http://your.server.name/servlet/snoop
```

7. Use the Debugger/OLT:

1. Watch for the Debugger/OLT. When the interface is displayed, make sure that Options -> Step by Step debugging mode is set.

The debugger interface is displayed in response to your requesting a servlet running on an application server (default server) for which debugging is enabled.

The debugger will find source files that are in the classpath of the JVM running the code being debugged. It does not find a source file, it will prompt you for the location of the file, `HttpServlet.java` in this case.

`HttpServlet` is the parent class of the Snoop servlet, and is the class in which the `service()` method is implemented. Starting with the requested servlet, the debugger will look for the first `service()` method implementation it can find. Because Snoop does not implement `service()`, the

debugger looked in Snoop's parent class, HttpServlet.

2. Browse for and select HttpServlet, which is located in the directory `<install_root>/classes/com/javax/http`
3. Watch the Debugger/OLT, which should stop in the service() method. To step into the Snoop servlet, perform a series of stepovers until you get to the doGet() method call. Do a step into on this method to reach Snoop.
4. Watch for the debugger to break in the service method of the SnoopServlet. You can step debug, or simply click Run to complete the request.

Debugging very_simple.jsp (JSP .91) from a remote machine

1. Configure the application server containing the JSP page for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start Task toolbar button or use the Start option on the server's right-click menu.
3. Make sure your Web server is started.
4. Start the Debugger/OLT:
 1. Open a command window on the remote machine.
 2. Run the following command to start the OLT daemon:

```
olt
```

This starts the OLT daemon, which listens on a port and displays the debugger interface when a request arrives.
5. Adjust the Debugger/OLT settings. Ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.
6. Copy the file `install_root\hosts\default_host\default_app\web\very_simple.jsp` to the document root of your Web server.
7. In a Web browser, request the very_simple JSP file:

```
http://your.server.name/very_simple.jsp.
```
8. Use the Debugger/OLT:
 1. When the Debugger/OLT interface is displayed, make sure that Options -> Step by Step debugging mode is set.
 2. You will likely be prompted for the location of the source file `pagecompile_very__simple.java`.

3. Browse for and select the file, which is located in the directory
`<install_root>\hosts\default_host\default_app\web\very__simple.jsp`.

You must install the source on the machine where you are running the debugger user interface.

4. Watch for the debugger to break in the service method of the `very__simple.jsp`. You can step debug, or simply click Run to complete the request.

Debugging `very__simple.jsp` (JSP 1.0) from a remote machine

1. Configure the application server containing the JSP page:
 1. In the Topology tree, locate the click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the Debug enabled and OLT enabled check boxes.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start Task toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the debugging interface.
4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a command window on the remote machine.
 2. Run the following command to start the OLT daemon:

```
olt
```

This starts the OLT daemon, which listens on a port and displays the debugger user interface when a request arrives.
6. Ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.
7. Copy the file `<install_root>\hosts\default_host\default_app\web\very__simple.jsp` to the document root of your Web server.
8. In a Web browser, request the JSP page:

```
http://your.server.name/very__simple.jsp
```
9. Use the Debugger/OLT:
 1. Watch for the Debugger/OLT.

When the interface is displayed, make sure that Options -> Step by Step debugging mode is set.

You will likely be prompted for the location of the source page `compile_very__simple_jsp_1`. You must install the source on the machine where you are running the debugger user interface.

2. Browse for and select the file, which is located in the directory
`<install_root>\temp\default_host\default_app\pagecompile_very__simple_xjsp.java`.
3. Watch for the debugger to break in the service method of the `very_simple.jsp`. You can step debug, or simply click Run to complete the request.

Debugging the Hello enterprise bean sample from a remote machine

1. Configure the application server containing the enterprise bean file:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the server's debugging properties, select the Debug enabled and Object Level Trace enabled check boxes.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.

If you have not already done so, install and deploy the Hello enterprise bean:

1. Select the Default Server.
2. Select Default Container and right click.
3. Select Create, Enterprise bean.
4. In the window that appears, click browse. Go to `AS_root\deployableEJBs\Hello.jar`.
5. Double click on the Hello.jar and select the deployment descriptor.
6. Select Create to start the deployment process. An icon representing the Hello enterprise bean should appear in the Default Container.

2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the green Start Task toolbar button or use the Start option on the server's right-click menu.

3. Make sure your Web server is started.

4. Start the Debugger/OLT:

1. Open a command window on the remote machine.
2. Run the following command to start the OLT daemon:

```
olt
```

This starts the OLT daemon, which listens on a port and displays the debugger user interface when a request arrives.

Ensure that the default execution mode is set to trace and debug in the Client Controller page. Set the check box to use the default settings.

5. Run the enterprise bean sample. Use your Web browser to open:

`http://your.server.name/WebSphereSamples/index.html`

Click Enterprise beans, click Hello, and then run Hello.

6. Use the debugger interface:

1. Watch for the debugger interface.

When the interface is displayed, make sure that Options -> Step by Step debugging mode is set.

You will likely be prompted for the location of the source file. The debugger will automatically find and display any source that is in the classpath of the JVM being debugged. The debugger will prompt for any source that is not in the JVM classpath. The source for HelloEJB.jsp is probably not in the classpath, so the debugger will prompt you for the location of the source `_WebSphereSamples\EJBs\HelloEJB\HelloEJB.jsp`.

2. Browse for and select the file, which is located in the directory `<install_root>\WebSphereSamples\EJBs\HelloEJB\HelloEJB.jsp`.
3. The debugger should break in the service method of HelloEJB.jsp. You can step debug, or simply click Run to complete the request.

Examples of tracing remote applications

- Trace a servlet
- Trace a JSP .91 file
- Trace a JSP 1.0 file

Trace Snoop servlet from a remote machine

1. Configure the application server containing the servlet:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the Debugenabled and Object Level Trace enabled check boxes.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the green Start Task toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but do not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a command window on the remote machine.
 2. Issue the command to start the OLT daemon:

```
olt
```

This starts the Debugger/OLT, which will display the trace information generated by the objects that service your request.

6. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace only in the Client Controller page. Set the check box to use the default settings.
7. In a Web browser, request the servlet:

```
http://your.server.name/servlet/snoop
```
8. Watch the Debugger/OLT for a line representing Snoop servlet. It will be labeled HttpServlet because that is the class that implements the service method. On the line, you should see circles on the line representing the beginning and end of the service method.

Trace very_simple.jsp (JSP .91) from a remote machine

1. Configure the application server containing the JSP file for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command to start the debugger daemon:

```
olt
```
6. Ensure that the default execution mode is set to trace only in the Client Controller page. Set the check box to use the default settings.
7. Copy the file `<install_root>\hosts\default_host\default_app\web\very_simple.jsp` to the document root of your Web server.
8. In a Web browser, request the JSP file:

```
http://your.server.name/very_simple.jsp
```
9. Watch the Debugger/OLT for a line representing very_simple.jsp. On the line, you should see circles representing the beginning and end of the service method.

Trace very_simple.jsp (JSP 1.0) from a remote machine

1. Configure the application server containing the JSP file for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.
2. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
3. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
4. Make sure your Web server is started.
5. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command to start the debugger daemon:

```
olt
```
6. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace only in the Client Controller page. Set the check box to use the default settings.
7. Copy the file `<install_root>\hosts\default_host\default_app\web\very_simple.jsp` to the document root of your Web server.
8. In a Web browser, request the JSP file:

```
http://your.server.name/very_simple.jsp
```
9. Watch the Debugger/OLT for a line representing `very_simple.jsp`. On the line, you should see circles representing the beginning and end of the service method.

Trace Hello enterprise bean WebSphereSample from a remote machine

1. Configure the application server containing the enterprise bean for debugging:
 1. In the Topology tree, locate and click the DefaultServer, which is the default application server. Its properties will be displayed on the right side of the console.
 2. For the default server's debugging properties, select the ObjectLevel Trace enabled check box.
 3. Set the OLT server host to the hostname of the remote machine on which OLT will be running.
 4. Click Apply.
2. If you have not already, install and deploy the Hello enterprise bean:
 1. In the Topology tree, find and right-click the Default Container in the Default Server.
 2. In the resulting menu, click Create -> Enterprise Bean.

3. In the resulting window, click Browse.
 4. In the Browse dialog, browse for <install_root>/deployableEJBs/Hello.jar. Double-click it to display its deployment descriptor. Click the deployment descriptor and exit the Browse dialog.
 5. Back in the dialog window for creating an enterprise bean, click Create.
 6. When the deployment process is complete, verify that the Hello enterprise bean now is displayed in the Default Container on the Topology tree.
3. Start the application server:
 1. In the Topology tree, make sure the Default Server is selected.
 2. Start it. Either click the Start toolbar button or use the Start option on the server's right-click menu.
 4. Optionally, minimize (but not close) the WebSphere Administrative Console to allow more space on your desktop for the tracing interface.
 5. Make sure your Web server is started.
 6. Start the Debugger/OLT:
 1. Open a system command window.
 2. Issue the command to start the debugger daemon:
`olt`
 7. Adjust the Debugger/OLT settings. When the interface is displayed, ensure that the default execution mode is set to trace only in the Client Controller page. Set the check box to use the default settings.
 8. Run the Hello sample. On Windows NT, click Start -> Programs -> IBM WebSphere -> Application Server version 3.x.0 -> Samples, then select the Hello enterprise bean.
 9. Watch the Debugger/OLT for lines representing the HelloEJB JSP file and the Hello enterprise bean itself.

Troubleshooting tips

See the prerequisites and limitations for known limitations, problems, and workarounds.

6.6.28.3: Prerequisites and limitations of IBM Distributed Debugger and OLT

This article provides information about Distributed Debugger and OLT prerequisites, and how they relate to IBM WebSphere Application Server prerequisites. It also lists current limitations, problems, and workarounds.

Supported prerequisites

IBM Distributed Debugger and OLT are separate products bundled with IBM WebSphere Application Server for your convenience. Because the Distributed Debugger and OLT support for operating systems and JDK levels can differ from those supported by IBM WebSphere Application Server, the Debugger and OLT support is summarized in the table below.

Inclusion of a prerequisite in the table does not imply that IBM WebSphere Application Server also supports the prerequisite. Always refer to the [IBM WebSphere Application Server prerequisites](#) to determine which prerequisites are supported by the base Application Server product and its Fix Packs.

IBM WebSphere Application Server Release	Operating systems supported by Debugger and OLT
Version 3.5 base	Remote and distributed debugging support and tracing capabilities on JDK 1.2.2 on Windows NT, AIX, and Solaris: <ul style="list-style-type: none"> ● Windows NT Java debugging is supported on JDK 1.2.2 ● AIX and Solaris Java debugging is supported on JDK 1.1.8
Version 3.5 plus Fix Pack 1	Remote and distributed Java debugging support and tracing capabilities on Windows NT, AIX, and Solaris, as follows: <ul style="list-style-type: none"> ● Windows NT Java debugging is supported on JDK 1.2.2 ● AIX and Solaris Java debugging is supported on JDK 1.1.8 ● Tracing capabilities are supported on JDK 1.2.2 on Windows NT, AIX, and Solaris
Version 3.5 plus Fix Pack 2	<ul style="list-style-type: none"> ● AWAITING UPDATE

Known limitations, problems, and workarounds

Problem	Operating systems	Workaround (if available)
When debugging a JSP (1.0 OR 0.91) Distributed Debugger does not step over empty lines. For example, when stepping into a JSP like: Line 1: Hello Line 2: Line 3: World! the debugger stops at line 2.	Windows NT	
The JVM arguments are not being passed to the Java program if the user invokes idebug using the -qjvmargs option.	AIX	Start the OLT Viewer and place it in the Online mode. Then debug the Java program using idebug with the -qjvmargs option as usual.
Add irmtdbgj.exe to the nanny.path in the admin.config file for Object Level Trace.		Before running OLT/Debugger support in WebSphere Application Server, you must add the location of the executable irmtdbgj.exe to the nanny.path variable in the WebSphere Application Server admin.config file. For example, on Windows NT, add the path <drive>\IBMDebug\bin\irmtdbgj.exe to the variable com.ibm.ejs.sm.util.process.Nanny.path in the WebSphere\AppServer\bin\admin.config file.

When running the OLT Viewer, if you try to change the Execution Mode in the Client Controller for a connected client and then click Apply , the viewer might freeze.	Windows 2000	Ensure that you have set the Execution Mode properly in the Client Controller for the Default settings entry before you try tracing. Do not attempt to change the Execution Mode after you start a trace.
---	--------------	--

- [IBM Distributed Debugger and OLT documentation](#)

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 and administrative 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

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)

IBM WebSphere Application Server works in DNS environments.

The product provides some support for the use of Java RMI. It can run applications that rely on Java Naming and Directory Interface (JNDI).

The remainder of this article contains miscellaneous notes about using name and location support. See the related links for additional topics.

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 security manager, and start the application server again.

6.6.36: Administering Java virtual machines (JVMs)

A JVM underlies each application server. You can use the application server command line settings (available in the administrative console) to set commandline arguments for the JVM process.

WebSphere Application Server provides a tool -- the JDK conversion assistant -- for switching the JVM vendor implementation that the server is using.

6.6.36.0: 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

6.6.36.5: Using the JDK conversion assistant to switch Java 1.2.2 vendor implementations

Suppose you want to switch from one vendor's implementation of the supported JDK level to that of another vendor -- for example, from the IBM JDK to the Sun JDK, or the other way around. You might be switching to a JDK that is optimized for a particular operating environment. The JDK conversion assistant guides you through the modifications required to allow IBM WebSphere Application Server to successfully find and recognize the new JDK.

The JDK to which you switch must be supported according to the [software prerequisites Web site](#), or you will be limiting your IBM support options.

Obtaining and running the JDK conversion assistant

1. Create a directory named SwitchJDK. Make sure it is a subdirectory of the "WebSphere" directory in the product *installation_root*.
2. Obtain the SwitchJDK file (a ZIP or JAR) from the among the tools offered on the [Support page of the product Web site](#).
3. Extract the JDK conversion assistant file contents into the SwitchJDK directory.
4. Run the assistant:
 - **UNIX** switchjdk.sh
 - **NT** switchjdk.bat
5. Follow the instructions in the assistant interface.

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.
```

Administering WebSphere plug-ins for Web servers

The remainder of this article discusses the main plug-in tasks and when to perform them:

- [Automatically triggering a regeneration of the plug-in configuration](#)
- [Manually triggering a regeneration of the plug-in configuration](#)
- [Manually editing the plug-in configuration](#)

Automatically triggering a regeneration of the plug-in configuration

Most of the time, you will perform administrative tasks without even thinking about dynamic plug-in configuration. The configuration is regenerated automatically every time you stop an application server process and start it again, as explained below.

The following general steps cause plug-ins to be regenerated automatically:

1. Make one or more configuration changes that are considered "start up" changes, in contrast to changes to that take effect immediately.
2. In the Topology tree on the Topology tab, locate the application servers containing the objects to which you made the configuration changes.
3. Stop the application servers and start them again to cause the dynamic regeneration of the Web server plug-ins.

Now the objects for which you made the configuration changes are operating with the new values you specified. The Web servers are aware of the changes, too.

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 OSE plug-in, the length of the delay is determined by the `ose.refresh.interval` setting in the IBM WebSphere Application Server `bootstrap.properties` file. The plug-ins poll the disk at this interval to see whether their configurations have changed.

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, stop the affected application servers and start them again to trigger an automatic regeneration of the plug-in configuration files.

When you trigger the regeneration of the plug-in configuration:

1. The servlet engine containing the Web application registers the configuration changes to those objects.
2. The servlet engine 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.

For the OSE plug-in configuration files, you should **always** use the WebSphere administrative console to modify the files, rather than modifying them directly. If you directly edit the files, the results cannot be guaranteed. It is likely your changes will be overwritten because the product periodically performs automatic

updates of these files.

Sometimes you might find it useful to *view* the OSE plug-in configuration files. For example, the files can reveal which requests are being handled by the servlet redirector (using RMI-IIOP), and which are being handled on the machine local to the Web server (using an OSE transport).

6.6.45.0: Properties of WebSphere plug-ins for Web servers

- [Finding the plug-in configuration files](#)
- [OSE plug-in properties](#)

Finding the plug-in configuration files

The working or active versions of the OSE plug-in files reside in the directory:

`product_installation_root/temp`

Reference versions of the OSE plug-in files reside in the directory:

`product_installation_root/properties`

OSE plug-ins

The OSE plug-in configuration consists of three files that are read by the plug-in code. The files define the selection criteria by which the plug-in code will accept and handle requests.

The following .properties files track the configuration of a WebSphere plug-in for a Web server:

- **vhosts** - This file is used to map the virtual hosts mentioned in a request to an actual host. If the virtual host in the request is a valid virtual host for an actual host in the administrative domain, the Application Server product adds the rest of the URL.
- **rules** - This file is used for looking up the URL constructed from the information in the vhost file.
- **queues** - This file is used to find the connection parameters for the Web server queue.

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 OSE plug-in:

```
LoadModule app_server_module C:/WebSphere/AppServer/bin/mod_app_server.dll
```

Apache (srm.conf)

For the OSE plug-in:

```
Alias /IBMWebAS/samples/ "C:/WebSphere/AppServer/samples/"Alias /IBMWebAS/  
"C:/WebSphere/AppServer/web/"NcfAppServerConfig  
BootFileC:\WEBSPH~1\APPSE~1\properties\bootstrap.propertiesNcfAppServerConfig LogFile  
C:\WebSphere\AppServer\logs\apache.log
```

Domino (httpd.cnf)

For the OSE plug-in:

```
Service IBMWebSphere C:\WebSphere\AppServer\bin\go46.dll:service_exitServerInit  
C:\WebSphere\AppServer\bin\go46.dll:init_exitC:\WebSphere\AppServer\properties\bootstrap.propertiesServerTerm  
C:\WebSphere\AppServer\bin\go46.dll:term_exitAuthorization *  
C:\WebSphere\AppServer\bin\go46.dll:authorization_exitNameTrans *  
C:\WebSphere\AppServer\bin\go46.dll:nametrans_exitPass /IBMWebAS/samples/*  
C:\WebSphere\AppServer\samples\*Pass /IBMWebAS/* C:\WebSphere\AppServer\web\*
```

IBM HTTP Server (httpd.cnf)

For the OSE plug-in:

```
LoadModule ibm_app_server_module C:/WebSphere/AppServer/bin/mod_ibm_app_server.dllAlias  
/IBMWebAS/samples/ "C:/WebSphere/AppServer/samples/"Alias /IBMWebAS/  
"C:/WebSphere/AppServer/web/"NcfAppServerConfig BootFile  
C:\WEBSPH~1\APPSE~1\properties\bootstrap.propertiesNcfAppServerConfig LogFile  
C:\WebSphere\AppServer\logs\apache.log
```

Netscape or iPlanet (obj.conf)

For the OSE plug-in:

```
Init fn="load-modules"  
funcs="init_exit,service_exit,auth_exit,term_exit"shlib="C:/WebSphere/AppServer/bin/ns36.dll"Init  
fn="init_exit"  
bootstrap.properties="C:/WebSphere/AppServer/properties/bootstrap.properties"NameTrans  
from="/IBMWebAS/samples" fn="pfx2dir" dir="C:/WebSphere/AppServer/samples"NameTrans from="/IBMWebAS"  
fn="pfx2dir" dir="C:/WebSphere/AppServer/web"Service fn="service_exit"
```

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. If things do not seem right in the administrative domain, you might find it useful to manually force the plug-in configurations to regenerate themselves. For example, you might notice that an object, such as a servlet, is operating under old settings, rather than using the newer settings you just specified. You have tried stopping the application server containing the object and starting it again, but the new settings do not seem to take effect.

 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](#).

Use the Regen Plug-in Configuration button that is part of any [servlet engine object in the administrative console](#).

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 Web server.

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 where xx varies, depending upon the version of your server.

6.6.46: Administering WebSphere administrative servers

The administrative server manages configurations and states of application servers, applications, and other resources in the WebSphere administrative domain. It keeps its administrative data in a database that you define during product installation.

- [Configuring the administrative server and database](#)
- [Creating the database tables and default configuration](#)
- [Establishing centralized administration](#) (shared administrative database)
- [What to do if you must drop the administrative database tables](#)
- [Miscellaneous topics](#)

Configuring the administrative server and database

There are a couple of ways to configure the administrative server, and provide information about its administrative database.

Administrative configuration file

The administrative server obtains its own configuration from the [properties in the administrative configuration file](#). Some values are set during product installation, while others are primarily for modification later.

Adding Java command line arguments for the administrative server

The `admin.config` file contains many administrative server properties you can set. In addition, you might want to pass the administrative server some generic Java command line arguments.

In the `admin.config` file, add the properties and their values as arguments on the `com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs` property. Add them using standard Java command line syntax:

```
-D property_name=property_value
```

For example, this setting passes [performance-related Java parameters](#) to the Java process constituting the administrative server:

```
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-ms32m -mx128m
```

The `admin.config` file contains both default properties that apply to most configurations and non-default properties.

Creating the database tables and default configuration

In most cases (see below for special cases), the administrative database tables are created, and populated with a default configuration, when you install WebSphere Application Server. During product installation, you must provide information regarding the administrative database you would like the installation's administrative server to use.

Settings in the administrative configuration file record the database information, and determine whether the database tables are created and (or) populated with the default configuration.

Special cases requiring additional steps after product installation

In some cases, the database tables are not created automatically during product installation, or the tables are not populated with the default configuration. If you need to, trigger the actions manually by setting properties in the administrative configuration file and starting the administrative server again.

 Users of WebSphere Application Server *Advanced Edition* Version 4.0 on Solaris and AIX operating systems with an Oracle database as the administrative database should note the following. In those cases, the default server is not created when you start the administrative server for the first time after installing the product. To create the default server and other default resources, set:

```
install.initial.config=true
```

in the administrative configuration file and start the administrative again.

 If you use an Oracle database user ID other than EJSADMIN, the default value assigned during the WebSphere Application Server installation, you will need to edit the administrative configuration file.

Ensure that the user ID and schema lines reflect your actual database user ID and password:

```
com.ibm.ejs.sm.adminServer.dbuser=your_user_name com.ibm.ejs.sm.adminServer.dbSchema=your_schema_name
```

Then start the administrative server.

Establishing centralized administration

Multiple administrative servers can be configured to use the same administrative database, allowing configuration data to be kept in a centralized place for administration of WebSphere Application Server across multiple machines (known as administrative nodes). In a successful multiple node configuration, the [administrative console](#) tree view will show each of the administrative nodes and its contents.

Configuring the 2nd through Nth machine in a cluster

Do not create the database tables or populate them with the default configuration.

When you are setting up a cluster of administrative nodes, only the WebSphere applicationserver product installation needs to install the default configuration:

1. On the first machine, ensure that the database tables have been created, and populated with the configurations for the default resources. In the administrative configuration file, this means:
`install.initial.config=true`
2. On all subsequent machines sharing the same administrative database, ensure the database tables are *not* created and the default resources are *not* installed a second time.
`install.initial.config=false`
`com.ibm.ejs.adminServer.dbInitialized=false`

If you have already installed the second (or later) machine, with `install.initial.config` set to true, then the problem should work itself out after all machines in the cluster have been shut down and started again. Until then, you might notice these problems:

- Absence of the **WebSphereAdministrative Domain** resource in the tree view of the administrative console, or other peculiarities in the administrative topology
- Administrative server on the second or later machine will throw exceptions when you first start it

Configuring a remote DB2 database as the administrative database

 If the administrative server is going to store its administrative data in a remote DB2 database, you need to catalog the remote database. See your database documentation for instructions.

Setting the `dbserverName` and `dbportNumber` for the data source (flags in the administrative configuration file) is not working in the case of a remote DB2 database. For example, if you have the application server and the DB2 client installed on one machine, and are trying to connect the client to the DB2 server on another machine.

This problem *could* be overcome in a subsequent WebSphere Application Server fixpak. Be sure to consult the release notes for each fixpak to see whether the problem is fixed.

What to do if you must drop the administrative database tables

If you must drop the tables in the administrative database for some reason, then reset the pertinent settings in the administrative configuration file to create the tables again.

You will always need to set:

```
com.ibm.ejs.adminServer.dbInitialized=true
```

If the machine is the first machine in a cluster of machines sharing an administrative database, then populate it with the default configuration:

```
install.initial.config=true
```

When you start the administrative server again, the settings will be applied.

Miscellaneous topics

The remainder of this article discusses miscellaneous topics pertaining to the administrative server and database.

Backup copies of admin.config

Note, commented out lines in `admin.config` will be removed completely when the administrative server starts.

It is recommended you periodically save a backup copy of `admin.config`. For example, you might delete lines that you want to restore later.

Running the administrative server in the background

To run the server as a background process, add this parameter to `admin.config`:

```
com.ibm.ejs.sm.adminServer.processPriority=>number
```

where *number* is 28 for AIX and 24 for Solaris. It could also be the default priority of the non-root user ID under which the administrative server is running. (See also [the information on running as non-root](#)).

You will also need to [edit the properties of each application server](#) associated with this administrative server. Modify the process priority of each

server to have the same process priority (*number*) you assigned to the administrative server.

Configuring the nanny process

On UNIX-based systems, the nanny process tries to start an administrative server when the administrative server fails. The nanny settings are available in the `admin.config` file for the administrative server that the nanny is tending.

6.6.46.0: Administrative server configuration file properties

The admin.config file is located in:

`product_installation_root/bin/admin.config`

 Directives in admin.config are similar to their counterparts on the Java command line for the administrativeserver. The command line argument name is appended to a standard package name for the administrative server. For example, the command line argument:

`-lsdPort`

becomes

`com.ibm.ejs.sm.adminServer.lsdPort`

in admin.config.

Some admin.config file properties are platform specific. Review your admin.config file to see which properties appear in your installation. Click any property in the table for a description of that property. Platform specific properties are identified by an icon.

Administrative server and database settings

`com.ibm.ejs.sm.adminServer.agentMode`

Whether this administrative server should run as a full administrative server (false), or in administrative server agent mode (true).

- o true
- o false

`com.ibm.ejs.sm.adminServer.bootFile`

The full path to a file containing configuration data for starting the administrativeserver, and product in general.

`com.ibm.ejs.sm.adminServer.bootstrapHost`

Fully qualified host name of the machine containing the administrative server

`com.ibm.ejs.sm.adminServer.bootstrapPort`

Port number for the administrative server. See also the information about [administering ports](#).

`com.ibm.ejs.sm.adminserver.classpath`

Values for the administrative server classpath, such as:

`d\:/WebSphere/AppServer/properties;`

`d\:/WebSphere/AppServer/lib/bootstrap.jar`

`com.ibm.ejs.sm.adminServer.connectionPoolSize`

The number of concurrent database connections to allow from the administrative server to the administrative database. The default value is 5.

`com.ibm.ejs.sm.adminServer.dbdataSourceClassName`

The data source corresponding to the administrative database.

`COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource`

`com.ibm.ejs.sm.adminServer.dbInitialized`

Whether to create the administrative database tables the next time you start the administrative server (typically, the first time you start the administrative server after product installation).

- o true
- o false

By default, this flag is initially set to false. The first time you run the administrative server, the administrative server creates the database table or tables that it needs to store administrative data. It sets this flag to true.

See the information about [administering administrativeservers](#) for additional discussion of this setting.

In Version 4.0, this setting is called `com.ibm.ejs.sm.adminServer.createTables`.

`com.ibm.ejs.sm.adminServer.dbpassword`

The password for the administrative database

`com.ibm.ejs.sm.adminServer.dbSchema`

The database schema for the administrative database

Data source properties - `com.ibm.ejs.sm.adminServer.dbUrl`

What was called `com.ibm.ejs.sm.adminServer.dbUrl` in Version 3.5 is replaced by specific [data source properties](#) for each database brand, including a `dbUrl` property for certain brands.

`com.ibm.ejs.sm.adminServer.dbuser`

The administrative database user ID, such as `db2user`

`com.ibm.ejs.sm.adminServer.diagThreadPort`

The port on which DrAdmin listens

`com.ibm.ejs.sm.adminServer.disableAutoServerStart`

Whether to disable the feature that tries to start the application servers on an administrativenode when the administrative node is started.

- o true - start the administrative server, but not any of the application servers on it
- o false - when the administrative server is started, try to start the application servers whose administrative settings indicate they should be started

`com.ibm.ejs.sm.adminServer.disableEPM`

Whether to disable the performance monitoring classes, to avoid collecting performance data when it is not needed.

- o true - disable EPM classes
- o false

`com.ibm.ejs.sm.adminServer.disableLocationService`

Whether to disable the location service daemon.

- o true - disable LSD
- o false

Applies only to Version 3.5, Standard Edition.

`com.ibm.ejs.sm.adminServer.exclusiveDBAccess`

Whether to *true* indicates exclusive access to the database.

- o true - exclusive access enforced
- o false

`com.ibm.ejs.sm.adminServer.jarFile`

Paths to the JAR files required by the administrative server, such as:

`product_installation_root/lib/repository.jar;product_installation_root/lib/tasks.jar`

`com.ibm.ejs.sm.adminServer.logFile`

The log files for recording administrative server events, such as:

`product_installation_root/tranlog/your_server_name_tranlog1,product_installation_root/tranlog/your_server_name_tranlog2`

`com.ibm.ejs.sm.adminServer.lsdHost`

The host name of the administrative server

com.ibm.ejs.sm.adminServer.lsdPort

Port number for the administrative server Location Service Daemon. See also the information about [administering ports](#).

com.ibm.ejs.sm.adminServer.managedServerClassPath

Used by the application server

com.ibm.ejs.sm.adminServer.nameServiceJar

A value such as:

[product_installation_root](#)/lib/ns.jar

UNIX com.ibm.ejs.sm.adminServer.nannyPort

The port on which the *nanny* process listens

UNIX com.ibm.ejs.sm.adminServer.processPriority

Possible values are:

- o 24 for Solaris
- o 28 for AIX

UNIX com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs

JVM arguments for the nanny process

UNIX com.ibm.ejs.sm.util.process.Nanny.path

Classpath for the nanny process, such as:

[product_installation_root](#)\bin;[product_installation_root](#)\SQLLIB\bin;[product_installation_root](#)\SQLLIB\function;[product_installation_root](#)\jdk\bin

com.ibm.CORBA.ConfigURL

The path to the file specifying ORB configuration values, such as:

file\:/[product_installation_root](#)/properties/sas.server.props

com.ibm.CORBA.ServerSocketQueueDepth

The ServerSocket maximum queue depth. Specify an integer greater than 50. Values less than 50 or invalid integer values will cause a warning to be logged to the message file and the default value of 50 to be used.

This property allows the ServerSocket maximum queue depth to be increased beyond its default value of 50. The default value is according to the JDK122 javadoc for java.net.ServerSocket.

The ServerSocket queue is where incoming socket connection requests from clients are held until the server can accept the request. When the queue becomes full, other open socket requests from clients are rejected with a java.net.ConnectException. This exception will be returned to the client application. If clients are receiving java.net.ConnectExceptions, it is an indication that you should consider increasing the queue depth.

Note, some operating systems impose other limitations on the server socket queue depth. Become familiar with the guidelines imposed by your operating system. Setting this property to a value beyond the platform-imposed limitation will likely not have any effect.

install.initial.config

If set to *true*, attempts to create the default resources (such as the default applicationserver) the next time you start the product administrative server.

- o false
- o true

server.root

See the [installation root reference](#) for the default values.

com.ibm.ejs.sm.adminServer.seriousEventLogSize

The number of entries to record in the serious events log. Allowing the log to grow too large can affect performance negatively. The default value is 1000 entries.

UNIX com.ibm.ejs.sm.util.process.Nanny.errTraceFile

The trace file for the nanny process, such as:

[product_installation_root](#)\logs\adminserver_stderr.log

com.ibm.ejs.sm.adminServer.qualifiedHomeName

This value can be:

- o true
- o false

install.initial.config.file

The full path to the property file defining the default configuration, such as:

[product_installation_root](#)/properties/initial_setup.config

com.ibm.ejs.sm.adminServer.traceString

The trace specification for the administrative server. See the [trace property reference](#) for a description.

com.ibm.ejs.sm.adminServer.traceOutput

The trace output file for the administrative server. See the [trace property reference](#) for a description.

UNIX com.ibm.ejs.sm.util.process.Nanny.traceFile

The trace file for the nanny process, such as:

[product_installation_root](#)/logs/nanny.trace

UNIX com.ibm.ejs.sm.util.process.Nanny.maxTries

How many times the nanny process should attempt to start the administrative server before giving up. The default is 3.

com.ibm.ws.jdk.path

The path to the root directory of the JDK installed with the product, such as:

[product_installation_root](#)/jdk

6.6.47: Administering generic servers

By making generic servers or processes known to the WebSphere administrative domain, the administrator can stop and start these processes from the administrative console when stopping or starting the applications that rely on them.

The WebSphere administrator configures one or more generic servers with settings for:

- Specifying the machine on which the server resides
- Specifying Java command line arguments and environment variables for starting the server process (if applicable)
- 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 and standard out streams for the server runtime
- Specifying a working directory

6.6.47.0: Properties of generic servers

Command line arguments

Specifies the command-line arguments to pass to the Java virtual machine (JVM) when the generic server is started.

Current State

Indicates the current state of the generic server. The next time the server is started, it will try to change to its desired state setting.

Desired state

Indicates the state the generic server should have the next time it is started.

Environment

Specifies the environment variables and variable values to be used by the server.

To set variables, click the Environment field to display a dialog box. In this box, enter variable names and values, clicking the Add button after each entry.

Executable

Specifies the path name of the server executable. Relative path names are associated with the server working directory.

Executable in use

Indicates the current executable.

Generic Server Name

Specifies a name by which to manage the generic server.

Group ID

Specifies the name of the operating system group under which to run the server.

Note that the operating system group must exist on the machine where the server will run before that server is started. This group must be assigned the necessary operating system privileges for performing operations, such as creating output files on the local file system. [Additional information specific to operating system](#)

Group ID in use

Indicates the current group ID in use by the server.

Maximum startup attempts

Specifies the number of times to try starting the server before discontinuing attempts.

Name

Specifies a name for the generic server. The name must be unique within the administrative domain to which the generic server belongs.

Node

Specifies the name of node on which the server runs.

Parent

Specifies the node (physical machine) on which the generic server code is located.

Ping initial timeout

Specifies 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

Specifies the frequency of communication attempts between the server and the administrative server, to ensure that the server is running.

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 and reduces system overhead.

This value must be less than or equal to the Ping timeout value and less than or equal to the Ping initial timeout value.

Ping timeout

Specifies the maximum time in seconds that can elapse after the last successful ping, before the administrative server assumes the server has failed.

Adjust this value based on your requirements for restarting failed servers. Decreasing the value shortens the time a server can be down before any restart attempts.

This value must be a positive integer.

Process ID (PID)

Indicates the process ID of the generic server.

Process priority

Specifies the operating system process priority under which to run the server. The lower the number, the greater the importance of the process. [Additional information specific to operating system](#)

This value must be a positive integer.

Process priority in use

Indicates the current process priority of the server.

Server Name

Specifies a name by which to manage the generic server.

Standard error (stderr)

Specifies the standard error stream for the operating system.

If the value of this property is set to the null string (""), the stream is set to the null device.

If this property is set to a relative path name, the path is associated with the server working directory. Any class of trace output can be redirected to this file. By default, the output of the fatal, error and audit trace classes is sent to this file.

[Additional information specific to operating system](#)

- Default: The file stderr.txt in the server working directory

Standard error (stderr) in use

Indicates the current standard error stream in use by the server.

[Additional information specific to operating system](#)

Standard input (stdin)

Specifies the standard input stream for the operating system.

If this property is set to the null string (""), the stream is set to the null device.

If this property is set to a relative path name, the path is relative to the server working directory.

[Additional information specific to operating system](#)

Standard input (stdin) in use

Indicates the current standard input stream in use by the server.

[Additional information specific to operating system](#)

Standard output (stdout)

Specifies the standard output stream for the operating system.

If this property is set to the null string (""), the stream is set to the null device.

If this property is set to a relative path name, the path is associated with the server working directory.

[Additional information specific to operating system](#)

- Default: The file stdout.txt in the server working directory.

Standard output (stdout) in use

Indicates the current standard output stream in use by the server.

[Additional information specific to operating system](#)

Start time

Indicates the most recent start time of the server.

State

Indicates the current state of the generic server. The next time this server is started, it will try to change to its desired state setting.

UID

Specifies the user ID of the operating system under which to run the generic server.

Note that the operating system user must exist on the machine where the server will run before that 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.

[Additional information specific to operating system](#)

- Legal Values: String of 8 or fewer characters
- Default: The ID used by the administrative server

Umask

Specifies an octal value that sets the operating system file creation mask for the server.

The file creation mask specifies permissions that cannot initially be granted for new files. When a new file is created, the system refuses to grant the permissions specified in the file creation mask.

For example, a mask of 022 prevents write permission to group members who own the file and to all other users who do not own the file.

A mask of 022 grants the owner all permissions; it leaves the owner's permissions the way the system specifies them.

If the system creates a file with privilege values of 777 (read, write and execute permissions for owner, group and other), a file creation mask of 022 causes the creation of the file with 755 privilege values (all permissions for the owner, but only read and execute permissions for group and other).

- Legal Values: An integer in the octal range 0 through 0777

Umask in use

Indicates the mask in use by the server.

User ID

Specifies the User ID (UID) under which the generic server should run.

User ID in use

Indicates the user ID (UID) in use.

Working directory

Specifies 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 starting a server, it is recommended that you do not change the server working directory.

6.6.47.1: Administering generic servers with the Java administrative console

Use the Java administrative console to administer generic servers.

This article extends article 6.6.47 (the overview of administering generic servers) with information specific to the Java console.

The table answers the most basic questions. See the [Related information](#) for links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource represented in the console tree views?	<p>The Type tree contains a Generic Servers folder object.</p> <p>The Topology tree can contain zero or more existing generic servers. Their names vary; they are supplied by the administrator.</p> <p>Use the View menu on the console menu bar to toggle between tree views.</p>
Any task wizards for manipulating this resource?	No

6.6.47.1.1: Adding generic servers with the Java administrative console

Use menus on resources in the Topology and Type trees to configure new generic servers (see [Related information](#) for instructions).

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. For example, changing the bootstrap port from the product default requires a new parameter when you start the Java administrative console.

Bootstrap port

- One for each administrative server
- Default value is 900
- To change it, modify the administrative server configuration file

Location Service Daemon (LSD) port

- One for each administrative server
- Default value is 9000
- To change it, modify the administrative server configuration file

ORB listener port for RMI/IIOP

- One for each application server and one for each administrative 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.
 - For administrative servers, add the argument to the `com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs` parameter in the administrative 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](#)

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-related settings in the WebSphere Application Server administrative console

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. That is, the administrative server on Machine A can have bootstrap port 900, while administrative servers on other machines have this same number.
- For administrative servers, pick port numbers above 1024 if not using the default port and **running as non-root**.
- Changing the bootstrap port from its default affects the administrative clients. When starting the Java administrative console, you must specify the new port number:

On UNIX-based operating systems:

```
adminclient.sh hostname port
```

On Windows-based operating systems:

```
adminclient hostname port
```

- 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, as well as workarounds for locale-specific defects.

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 zh_TW.big5	zh_TW.big5	zh_TW.big5 *

Limitations and workarounds

Problem	Op System	Workaround
All Languages		
Installing WebSphere Application Server in more than one causes the different language versions to overwrite each other. Also, with native install (using SMIT), when more than one language is installed, an error displays.	AIX	The installation should allow you to install more than one language and switch between languages. Currently, because all of the languages are named the same in the same installation path, they overwrite each other. Install only one language of WebSphere Application Server on AIX. If you need to install a different language, uninstall the previous language before installing the one desired.

<p>The Cancel button in <code>uninstallshield</code> does not work.</p> <p>When you click Cancel in <code>uninstallshield</code>, the executable loops and the <code>uninstallshield</code> automatically reloads.</p>	<p>AIX</p>	<p>To cancel the uninstall, press Ctrl+C at the command prompt in the shell from which the uninstall program was initiated.</p>
<p>Brazilian Portuguese</p>		
<p>The time displayed with events on the administrative console is one hour earlier than the actual system time.</p>	<p>Windows NT/2000, AIX, Solaris</p>	
<p>Japanese</p>		
<p>Japanese characters are distorted on the following installation panels:</p> <ul style="list-style-type: none"> ● Choose Application Server Components panel, all of the characters in Component Description. ● On some panels after the Database Options panel, kana-kan status characters at the bottom left corner. ● Selected Install Options panel, all of the selected option strings. 	<p>AIX</p>	<ol style="list-style-type: none"> 1. Install <code>X11.fnt.ucs.ttf</code> - AIXWindows unicode TrueType fonts. 2. Logout and login again. <p style="padding-left: 40px;">This must be done to reflect installation of the proper fonts.</p> <ol style="list-style-type: none"> 3. Run <code>xlsfonts grep 0208</code> to confirm monotype fonts are installed. 4. Install WebSphere Application Server Version 3.5. You will see DBCS characters in TxTArea.
<p>When you install WebSphere Application Server on the Japanese Solaris operating system, the dialog on which you select the database home directory (<code>DB_HOME</code>) does not display the directory entry. <code>startupServer.sh</code> and <code>admin.config</code> are not updated.</p>	<p>Solaris</p>	<ol style="list-style-type: none"> 1. Set the following value for the locale on the shell: <code>LANG=C</code> 2. Start <code>install.sh</code>.

<p>Java administrative console pages do not display properly for Japanese AIX customers using the eucJP locale.</p>	<p>AIX</p>	<p>You will need to change some text files:</p> <p>If you see pages in the AdminConsole containing lots of ??????????????, install Japanese WebSphere on AIX.</p> <p>If ja_JP locale is used, the following .txt files in the /web/help directory need to be converted with the command (on one line):</p> <pre>iconv -f IBM-932 -t IBM-eucJP Originalfile.txt > Originalfile_ja.txt</pre> <p>Then, rename the files:</p> <pre>mv Originalfile_ja.txt Originalfile.txt</pre> <p>The list of files:</p> <pre>AddJSPEnabFrontPage.txt AddServletFrontPage.txt AddWebResFrontPage.txt AppSecFrontPage.txt CfgAppSvrFrontPage.txt CfgEntAppFrontPage.txt CfgServEngFrontPage.txt CfgVirtHostFrontPage.txt CfgWebAppFrontPage.txt EditEntAppFrontPage.txt GlobSetFrontPage.txt MethGroupFrontPage.txt PermFrontPage.txt ResAnalyzFrontPage.txt ResSecFrontPage.txt</pre>
---	------------	--

Spanish

<p>When using the Types tab, then selecting Model and create, and then choosing Application server, the properties window is initially too small. The right side of the window is cut off by about five characters.</p>	<p>AIX</p>	<p>Manually resize the window.</p>
---	------------	------------------------------------

Chinese Simplified

<p>On AIX V4.3.3:</p> <ul style="list-style-type: none"> ● Fix Pack U466148 must be applied before installing WebSphere Application Server. ● WebSphere Application Server only supports one Simplified Chinese Charset, BGK. 	<p>AIX</p>	<p>Fix Pack U466148 is available on the 4.3.5 installation media. It is not necessary to upgrade the operating system to 4.3.3 to install this Fix Pack.</p>
---	------------	--

Chinese Traditional

--	--	--

Problems with character sets other than BIG5 on AIX V4.3.3.	AIX	WebSphere Application Server only supports the BIG5 Traditional Chinese Character Set for AIX V4.3.3.
While using the product, corrupt panels and code page conversion errors appear.	Solaris	<p>To install and use the WebSphere Application Server with a minimal amount of corrupted panels, follow the procedure below:</p> <ol style="list-style-type: none"> 1. From a command line, type: export LC_ALL=tchineseexport LANG=tchinese 2. Install WebSphere Application Server. 3. From a command line, type: export LC_ALL=zh_TW.BIG5export LANG=zh_TW.BIG5 4. Create the WebSphere Application Server database as instructed in the documentation. 5. Repeat step 1. 6. Start the administrative server. 7. Repeat step 3. 8. Start the administrative console. <p>To uninstall WebSphere Application Server with a minimal amount of corrupted panels:</p> <ol style="list-style-type: none"> 1. From a command line, type: export LC_ALL=tchineseexport LANG=tchinese 2. Uninstall WebSphere Application Server as instructed in the documentation.
All Double-Byte Languages		
In the WebSphere Administrative Console, Asian characters show up as empty boxes.	HP-UX	<p>The Java swing libraries require TrueType fonts. HP-UX does not providethese fonts for Asian languages. The SDK release notes state that HP-UX is working to provide these fonts in the future. In the meantime, Japanese, Korean, Chinese Traditional, and Chinese Simplified customers must purchase their own TrueType fonts from a third party vendor. They can also force the WebSphere Administrative Console to run in English by running the following command from the command prompt before starting the WebSphere Administrative Console:</p> <pre>export LANG=en_US.iso88591</pre> <p>Also, for more information on this limitation, refer to the (HP-UX SDK Release Notes).</p>
Double-byte server name is corrupt in console messages. If you name a server with Asian-Pacific characters and start it, the name in the status message (starting, stopping) on the Administrative Console is garbage (question marks and square blocks).	All	Use only English characters and numbers for your server name.

<p>An application server will not start if named with more than three Chinese characters.</p> <p>With more than three Chinese characters, the name in the status message (starting, stopping and others) is made up of meaningless characters (for examples, question marks and square blocks).</p>	<p>All</p>	<p>Any combination of less than three Chinese characters or any number of English characters works fine. Use English or fewer than three double-byte characters.</p>
---	------------	--

6.6.50: Administering coexisting product versions and editions

Coexistence of Standard and Advanced Editions of Application Server

Running Standard Edition and Advanced Editions on the same machine is not supported. Simultaneous use of a Web server by both editions (trying to install the Standard Edition and Advanced Edition WebSphere plug-in on the same Web server instance) is not supported.

Coexistence of Versions 2.0x and 3.x of Application Server

IBM WebSphere Application Server Versions 2.0x and 3.x can exist on the same machine, but it is not recommended. If 2.0x must reside on a machine with Version 3.x (3.0x or 3.5), make sure each is associated with a separate Web server. Also, make sure they do not access the same resources, such as ports and files. Finally, be alert to problems stemming from using two different versions of the JDK and other prerequisites on the same machine.

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>ipconfigWindows NT IP ConfigurationEthernet adapter adapter_ID_1:IP Address. . . . .
: aaa.bbb.ccc.ddd (Primary IP)Subnet Mask . . . . . : 255.255.255.0Default Gateway . . . . .
: 208.180.235.1Ethernet adapter adapter_ID_2:IP Address. . . . . : eee.fff.ggg.hhhSubnet
Mask . . . . . : 255.255.254.0Default Gateway . . . . . : 129.17.32.1d:\work>nslookup
aaa.bbb.ccc.dddServer: my.server.netAddress: mmm.nnn.ooo.ppp Name: cdm-235-122-pflu.cox-internet.com
(Resolves to hostname)Address: aaa.bbb.ccc.ddd:\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.netAddress:
mmm.nnn.ooo.pppName: cdm-235-122-pflu.cox-internet.comAddress: 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.netAddress: mmm.nnn.ooo.pppName: cdm-235-122-pflu.cox-internet.comAddress: aaa.bbb.ccc.ddd
(IP when fully qualified)
```

6.6a: Starting and stopping the server

1. [Editing administrative server startup scripts \(UNIX\)](#)
2. Starting administrative servers:
 - [On UNIX systems](#)
 - [On Windows systems](#)
3. [Obtaining feedback that the administrative server started successfully](#)
4. [Starting the Java-based administrative console](#)
5. [Troubleshooting the administrative server startup](#)
6. [Starting and stopping application servers](#)
7. Stopping administrative servers:
 - [On UNIX systems](#)
 - [On Windows systems](#)

See also: [Running administrative and application servers as non-root on UNIX systems](#)

Scripts for configuring how administrative servers start

On UNIX systems, you can configure the starting parameters for an administrative server by editing its startup file:

```
product\_installation\_root/bin/startupServer.sh
```

See below for recommended startup instructions. It is not always advisable to run the startup file directly.

Starting administrative servers on UNIX-based systems

To start the administrative server:

1. Ensure you are running as the root user.
2. From a command prompt, change to the directory containing the startup script (as described above) and enter:

```
./startupServer.sh
```

Running as non-root on UNIX-based systems

Normally, the administrative server must be run under the system root ID. To avoid this, see [Running the administrative server as non-root](#).

Stopping servers on UNIX-based systems

To stop an WebSphere administrative server on UNIX-based systems, use an administrative client, such as the Java administrative console. Using the kill command is **not** recommended.

Starting administrative servers on Windows-based systems

The administrative server is started and stopped from the Control Panel Services menu.

1. From the Start menu, select Settings > Control Panel.
2. Open the Services icon.
3. From the Services panel, select IBM WS AdminServer.
4. Use the Start and Stop buttons. Stopping the administrative server also stops the application server.
5. To enable or disable automatic startup of the administrative server on system reboot, click Startup and select the Automatic or Manual.
6. If starting the administrative server, start your Web server. The WebSphere administrative server and the Web server run in separate processes. You can start them in any order.
7. Close Services.

Stopping servers on Windows-based systems

To stop the server, stop the corresponding Windows service.

Obtaining feedback

After an administrative server starts, messages are written to the administrative servertrace file at:

[product_installation_root](#)/logs/tracefile

Look for the following messages to confirm that the server started successfully:

```
Managed Server E Version: Runtime 3.0 dev Config: ES-jmon-3.0Admin Server E Admin Server open for e-business.
```

Troubleshooting

See the Problem Determination section for methodology and problem descriptionsfor debugging the failure of the server to start. Here are some highlights. Ifthe server will not start ...

- ... and you encounter this error (in the Services panel on Windows NT):

```
The IBM WebSphere AdminServer service returned service specific error 10
```

Add a newline character at the end of the [administrative server configuration file](#).

- ... when WebSphere security is enabled, and it is on a Windows machine configured asa part of a Windows domain and not connected via the network to the domain server,then to solve the problem, configure the Windows NT/2000 machine so that it is *not*part of a Windows NT/2000 domain.

6.6.a.1: Running the product servers and consoles as non-root

Any application server or administrative server can be run using a non-root ID. The tradeoff is that you must use an LDAP directory for the authentication mechanism for WebSphere security. You can no longer use the local operating system.

By default, WebSphere servers use a root ID. Use the instructions below to change the ID. The Java administrative console can be accessed from a non-root ID, provided security permissions are configured appropriately.

- [Running application servers as non-root](#)
- [Running administrative servers as non-root](#)
- [Running Java administrative consoles as non-root](#)
- [Running as non-root on Solaris: ndd](#)
- [Problems and symptoms based on running as non-root incorrectly](#)
- [Cases in which you must run the server as root](#)

Running application servers as non-root

1. [Start the WebSphere administrative server](#) under the root ID.
2. [Start the Java administrative console](#).
3. In the tree view, locate and click the application server to display its properties.
4. In the Advanced properties, modify the User ID and Group ID to be the user and group for the application server to "run as."
5. In the General properties, modify the standard output and standard error log path to refer to directories to which the "run as" identity has access.
6. Remove any temporary files that were created by previous executions of the application server when it was "run as" the previous user ID. The files are of the form:

```
/tmp/.asXXXXXX
```

where XXXXXX is a communications queue name used by WebSphere Application Server, such as /tmp/.asibmappserve1

7. Start the application server, using the new ID.

Running administrative servers as non-root

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/logs/*](#)
 - [product_installation_root/properties/*](#)
 - [product_installation_root/tranlog/*](#)
 - [product_installation_root/temp/*](#)
 - [product_installation_root/bin/admin.config](#)

Make sure that the user has write and execute privileges.

2. Remove any temporary files that were created by previous executions of the application server when it was "run as" the previous user ID. The files are of the form:

```
/tmp/.asXXXXXX
```

where XXXXXX is a communications queue name used by WebSphere Application Server, such as /tmp/.asibmappserve1

3. Change the bootstrap port of the administrative server to a value greater than or equal to 1024:

- a. Open the [administrative configuration file](#) in a text editor.

- b. Add the following:

```
com.ibm.ejs.sm.adminServer.bootstrapPort=2222
```

where 2222 is just an example of a new port that you might use.

Changing the bootstrap port affects the administrative clients that connect to the server. See the [port administrative overview](#) for details.

i While you are in `admin.config`, you might want to configure the administrative server to run in the background as non-root.

4. Start the administrative server, using the new ID.

Running Java administrative consoles as non-root

1. Change the ownership of the following directories and files to the user and group that you would like the console to "run as":
`product_installation_root/binproduct_installation_root/properties/sas.client.props`
2. Make sure the user has permission to access the secured administrative account.

What you should know about running as non-root on Solaris: ndd

On Solaris, the "nnd" commands in the administrative server `startupServer.sh` script need to be commented out unless you are running as root.

If an "nnd" command is being executed by a non-root user, the following error message will be issued to stdout or stderr:

```
operation failed, Not owner
```

The 'nnd' command is for dynamically adjusting certain IP stack parameters. It attempts to operate on operating system level kernel device settings, which can only be performed by root. Thus the error message.

The workaround is to either run the administrative server as root or edit the `startupServer.sh` script, commenting out the nnd command.

It is still strongly recommended that the changes to the TCP parameters that the "nnd" command makes be made by root on all machines running the application server and Web server (in case they are not the same box).

Problems and symptoms based on running as non-root incorrectly

The following problems indicate the need to review the above instructions to ensure that your configuration is correct for running as non-root.

- When running a servlet, the following message is displayed in the Web browser:

```
Server internal error
```

Then the following message is displayed in `your_server-stderr.log`:

```
open_unix_domain_server_socket_listener - bind/listen: The socket name is already in use.  
com.ibm.servlet.engine.osel.listener.outofproc.ServerQueueException: EError: create_server 2
```

- The following error message is displayed when to start the administrative server as non-root on Solaris:

```
$ ./startupServer.sh operation failed,Not owner
```

Cases in which you must run the server as root

The administrative and application servers must be running as root if:

- You are using the thin servlet redirector configuration

On UNIX-based systems, a secured Java console must be operated using a root ID when the authentication mechanism for WebSphere security is the operating system. If using an LDAP directory service, the console can be run as non-root.

6.7: Tutorial -- Application configuration and deployment

Use the illustrated, step-by-step tutorial to practice the end-to-end procedure for defining, configuring, and securing an application.

[Begin the tutorial](#)

Administrative console tutorial: Standard Edition

Last updated: 7/25/2000

MANY GRAPHICS: PLEASE BE PATIENT AS IT LOADS.

Welcome to the IBM WebSphere Application Server Version 3.5 Standard Edition administrative console tutorial.

The tutorial demonstrates how, when, and why to use the central administrative tasks. It also discusses tasks you need to complete outside of the administrative console to complement and support the administrative tasks.

Using some of the Application Server sample files, you can configure and secure an application comprised of HTML files, servlet files, and JavaServer Pages (JSP) files.

In the course of completing the sections of this tutorial, you will experience the end-to-end process of managing applications in IBM WebSphere Application Server Version 3.5:

- [1. Plan and prepare](#)

Learn about the scenario on which the tutorial is based and the sample files the tutorial will reference. Think about some common planning considerations.

- [2. Configure a Web application and supporting resources --](#)

Configure a Web application comprised of servlets, Web pages, and JSP files. Also configure an application server, servlet engine, and database access support.

- [3. Configure an enterprise application --](#)

Wrap your Web application in an enterprise application in order to apply security.

- [4. Apply security](#)

Enable and configure security to protect the application and its resources.

- [5. Verify the application -- try it!](#)

Verify that the configuration and security you applied to your application is working as you intended.

Step 1: Plan and prepare

Learn about:

- [The tutorial scenario](#)
- [Sample files used in the tutorial](#)

The tutorial scenario

You are part of a Web site team. To save time maintaining sites, a team member has written a handy application that lets you specify "expiration dates" for Web pages. When page expires, it is automatically replaced by another page.

These files comprise the "Expiring Page" application:

- [ExpiringHTMLInputPage.html](#)
- [ExpiringHTMLBeforePage.jsp](#)
- [ExpiringHTMLAfterPage.jsp](#)
- [ExpiringHTMLServlet.class](#)

For more information about ExpiringHTMLPage, open the index.html file in the WebSphereSamples directory of your product installation.

Sample files used in the tutorial

ExpiringHTMLInputPage.html

This HTML file provides an input form where you set the expiration date.

The sample file is located in:

`install_root\hosts\default_host\WSSamples_app\web\WebSphereSamples\ExpHTMLServlet`

where `install_root` is the root of your WebSphere Application Server installation. Note, the slashes will vary ("\" or "/") depending on your operating system.

ExpiringHTMLAfterDatePage.jsp

This is the page you want served after the expiration date.

This sample file is located in:

`install_root\hosts\default_host\WSSamples_app\web\WebSphereSamples\ExpHTMLServlet`

ExpiringHTMLBeforeDatePage.jsp

This is the page that you want served up until the expiration date.

This sample file is located in:

`install_root\hosts\default_host\WSSamples_app\web\WebSphereSamples\ExpHTMLServlet`

ExpiringHTMLServlet.class

This is a Java servlet that compares the expiration date to the current date and decides which page to serve.

This sample file is located in:

`install_root\hosts\default_host\WSSamples_app\servlets\WebSphereSamples\ExpHTMLServlet`

If these files were not part of a predefined tutorial, some planning might be involved prior to beginning administration:

- Which files should comprise an application?

For example, consider the increasingly popular Internet sites that allow parents to create virtual bank accounts for their children. The children can then manage the accounts to learn about money and fiscal responsibility.

Such a site might have a VirtualAccount application that performs the following functions:

- Allow users to create accounts
- Allow users to maintain accounts (check balance, transfer balances, and such)

In WebSphere Application Server Version 3.5, the files in an application can be managed as a unit. In the above case, it might be wise to break the one large application into two smaller applications for administrative purposes. This way, even if one of the functional areas is temporarily out of service, the other part of the application can remain online. Children can continue to access their existing bank accounts, even on a day when the functionality allowing parents to create new bank accounts is unavailable.

- Who is the audience for this application? What implications does this have for security?
- Does this application require database access?

Servlets requiring database access demand some additional steps.

The servlet in the Expiring Page sample you will be configuring does not require database access, but the tutorial reviews the necessary steps anyway because they are important to know if you plan to use database connection pooling for your own applications.

Configure a Web application and supporting resources

This case will demonstrate how to quickly configure an application using mostly the default settings. It will show where to move the application files to, allowing the application to work correctly.

If you selected the "default configuration" option during your Custom installation of WebSphere Application Server, you will already have a default application server, servlet engine, and other resources required to support an application.

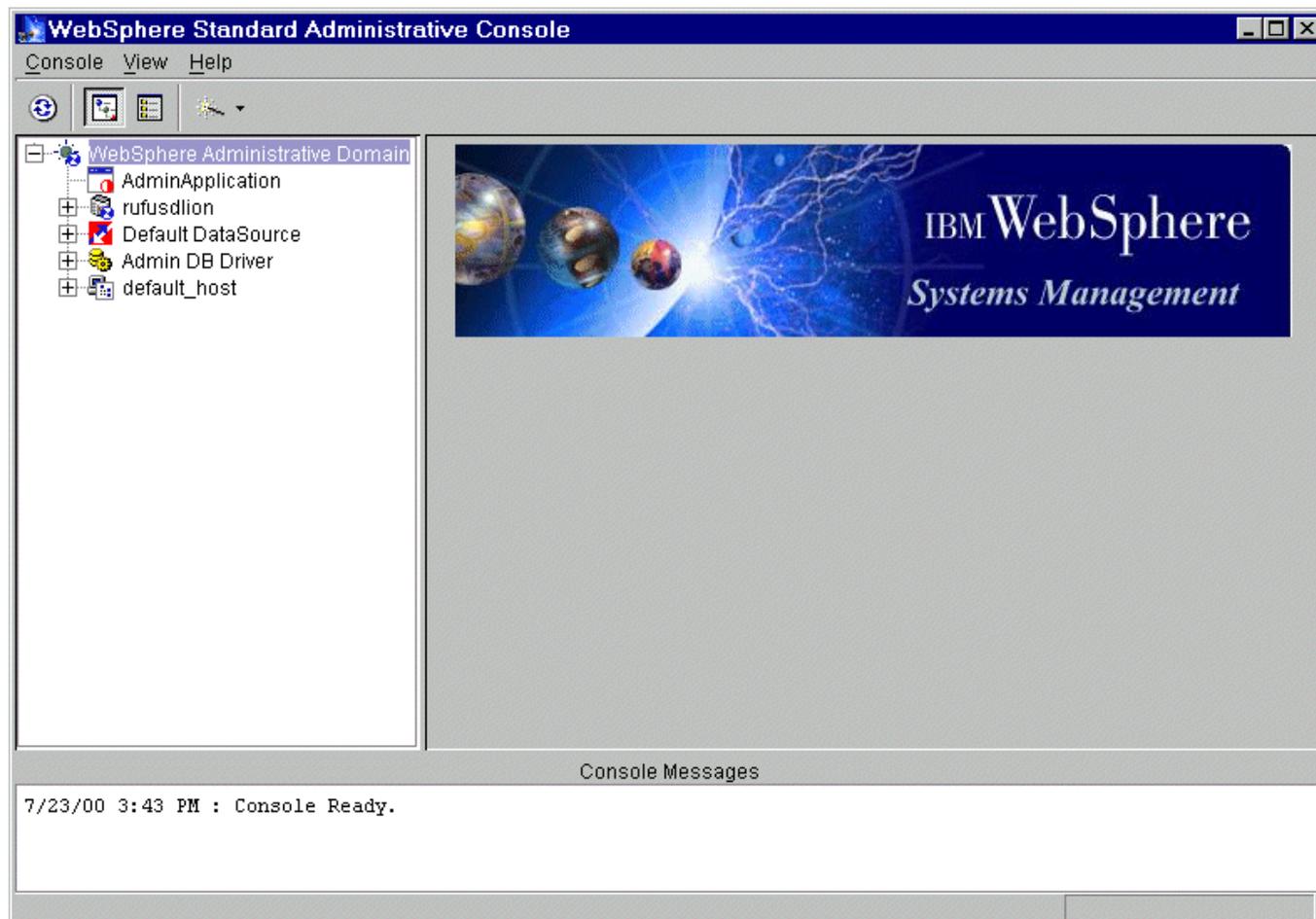
This tutorial does not assume you have the default resources installed. The first step of this section demonstrates configuring your own application server and other supporting resources.

In this section, you are going to:

1. Configure a Web application and supporting resources
2. Place the application files in WebSphere directories

Note: Slight discrepancies in the screen captures in this tutorial and the actual WebSphere Administrative Console are possible.

Open the WebSphere Administrative Console:



This is the point from which you can perform most of the administration, using task wizards that guide you through the process of configuring, securing, and analyzing the performance of resources and applications.

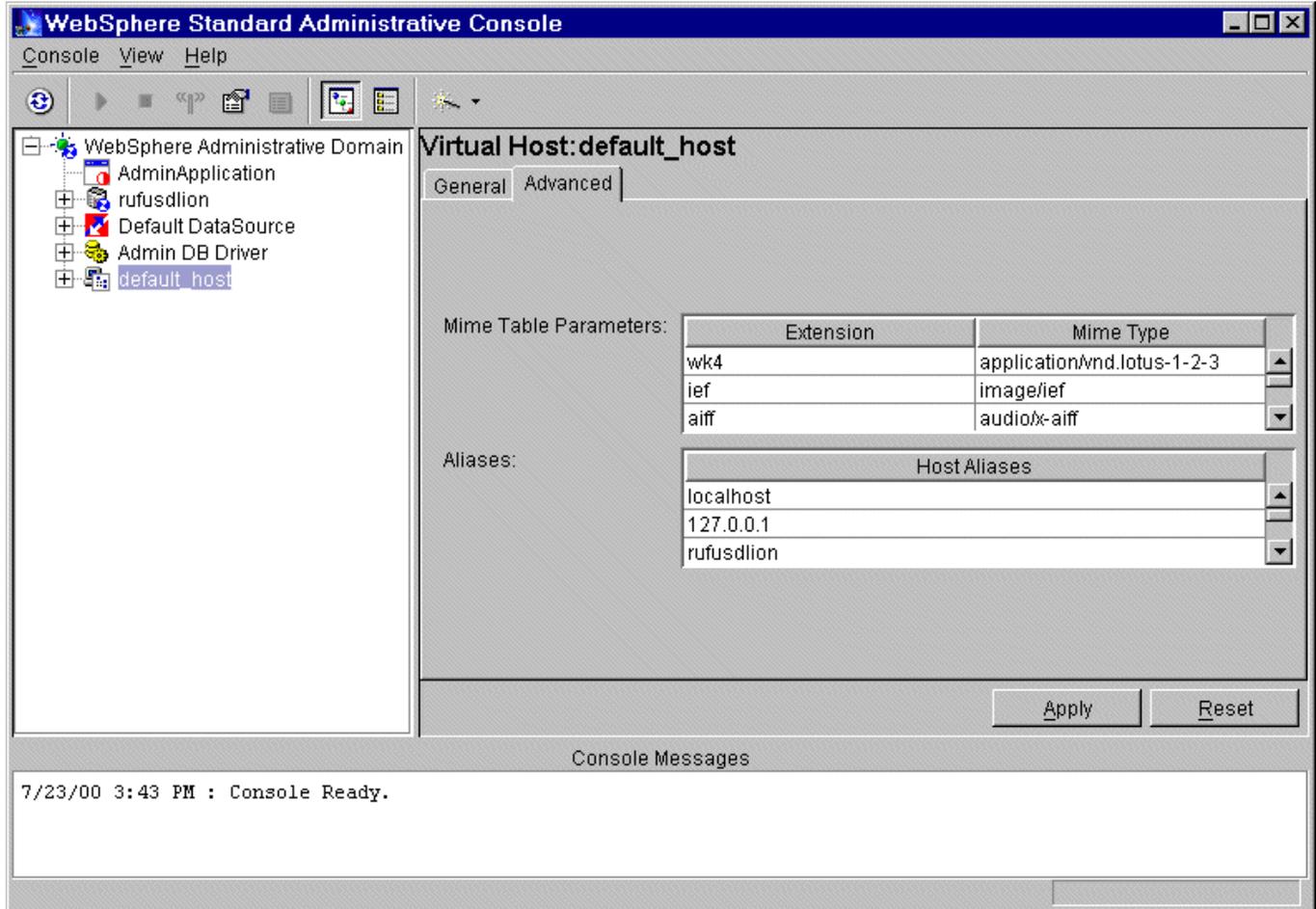
Notice the "Console ready" message; this indicates that the console is ready for use.

Configure a virtual host

A virtual host is a configuration that allows a single physical machine to resemble multiple host machines. You can isolate applications so that they seem to be on entirely different boxes.

You can configure your own virtual host, but for the purposes of the tutorial, use the default virtual host. You do not need to change any of its settings. Simply verify that it exists.

To view the default host, click the Topology button to display the Topology view. Expand the tree until you locate the "default_host." It should be near the bottom:



When you click it, its properties are displayed on the right side of the console. Most notable are its aliases (click the Advanced tab). Because the default host represents your local machine, its aliases include "localhost," the loopback address (127.0.0.1), and the short and fully-qualified host names of the machine.

It might not be necessary for you to configure a new virtual host for your production environment. A production environment does not necessarily require multiple virtual hosts. If yours requires only one virtual host, then the default_host is probably suitable.

- Browse [virtual hosting help files](#).

Next, configure resources for database support.

Configure database support

The "Expiring Page" application's servlet does not require database access, but it is likely that you will want to set up database access for one or more of your own servlets. If not, you can skip ahead to [configuring the Web application, servlet engine, and application server](#).

The tutorial assumes your servlets are accessing an IBM Universal Database (DB2) Version 6.1.

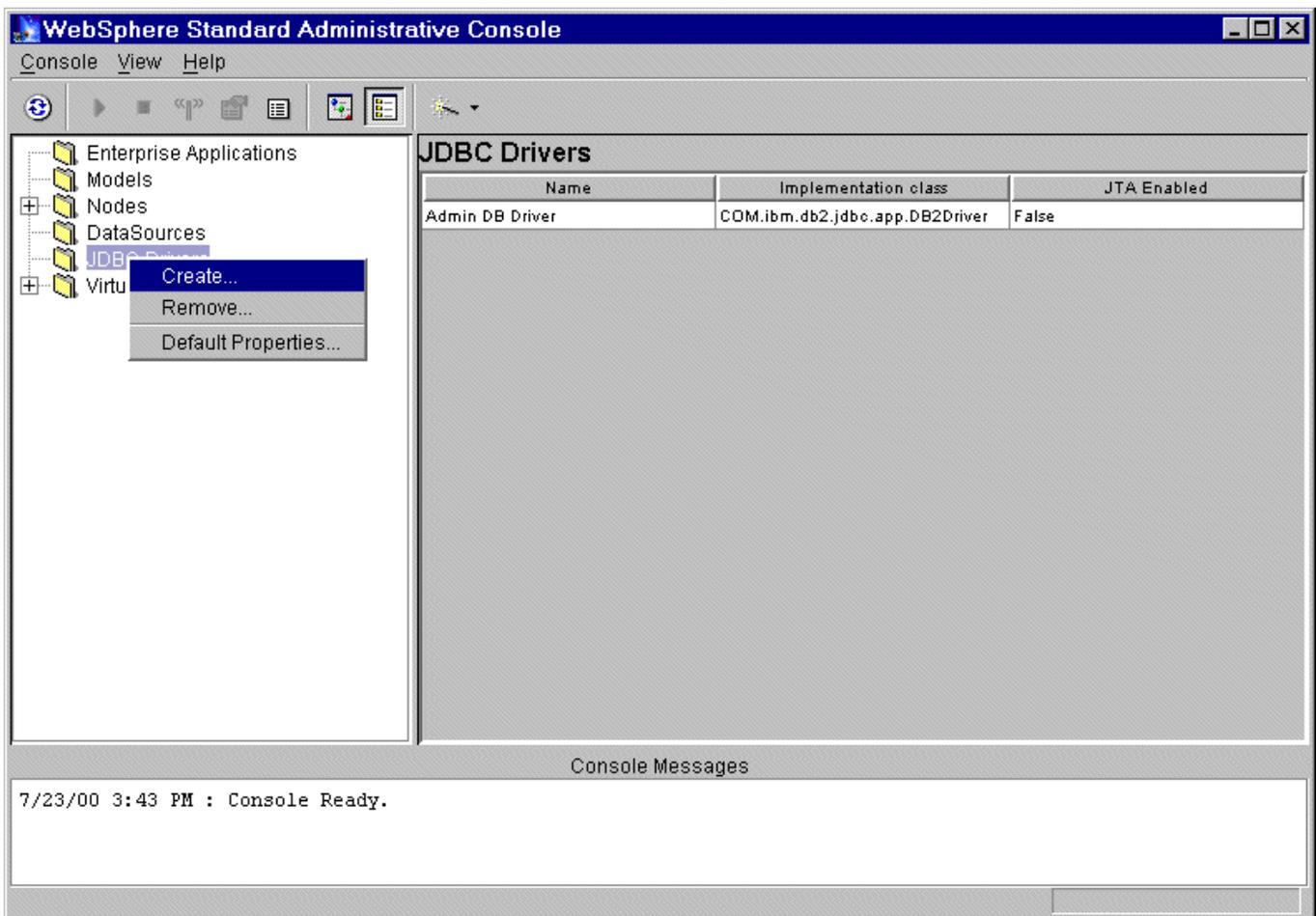
I Even if you do not have a database right now, you can try these steps for practice. You will receive errors that you can disregard.

During this part of the tutorial, you will:

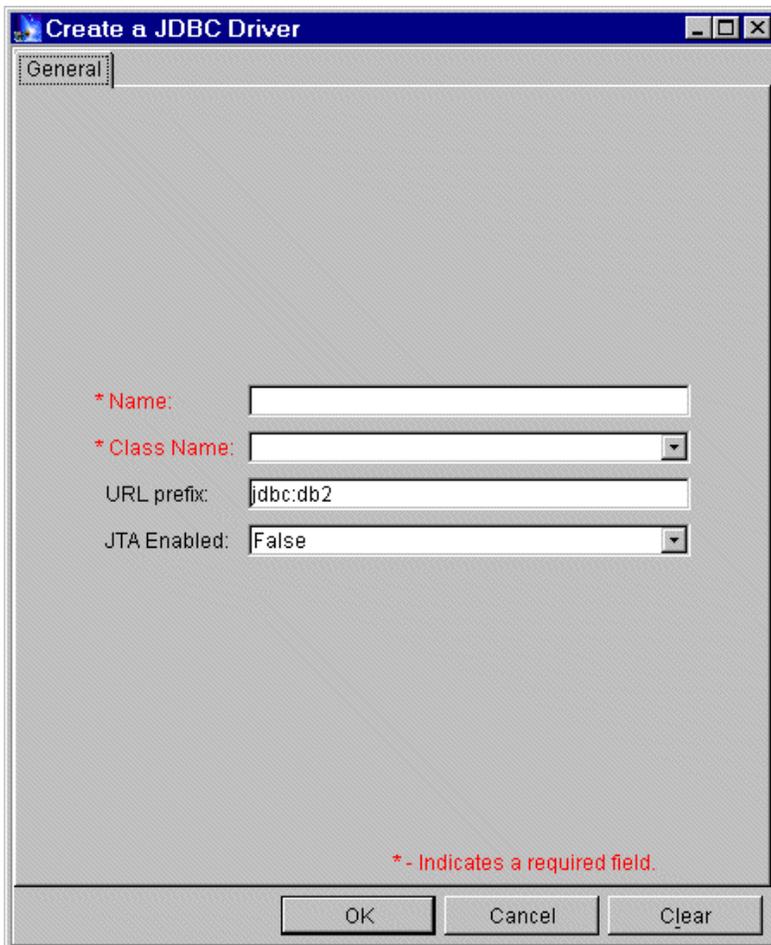
- Configure a JDBC driver
- Configure a data source
- Specify the location of the driver code

Let's begin.

Click the Types button to display the Types view. Locate the JDBCDrivers folder. Right-click the folder and select its Create menu option:



You are going to specify the type of JDBC driver you will be using. A JDBC driver is the code that lets Java components access databases. The driver properties dialog will be displayed:



Specify properties for the JDBC driver:

Name

Specify ExpiringPageDB2Driver.

This is the name by which to administer the driver. Here, you have given it a name that indicates it is a DB2 driver for the Expiring Page application to use.

Implementation class

Specify the DB2 choice, com.ibm.db2.jdbc.app.DB2Driver

This is the implementation class of the driver code. If you were using another database such as Oracle instead, you would select the class for that database type.

JTA enabled

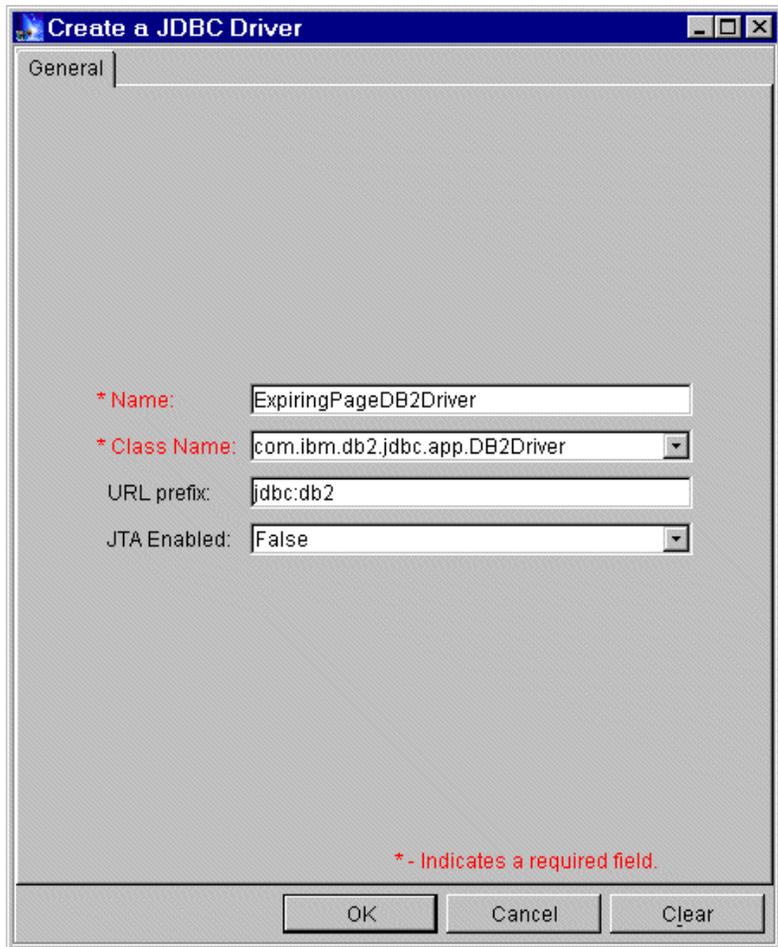
Specify No.

This specifies whether the database brand and version supports the Java transaction services.

The remainder of the properties are optional. Do not specify anything for them.

- [Examine field help for JDBC driver properties](#)

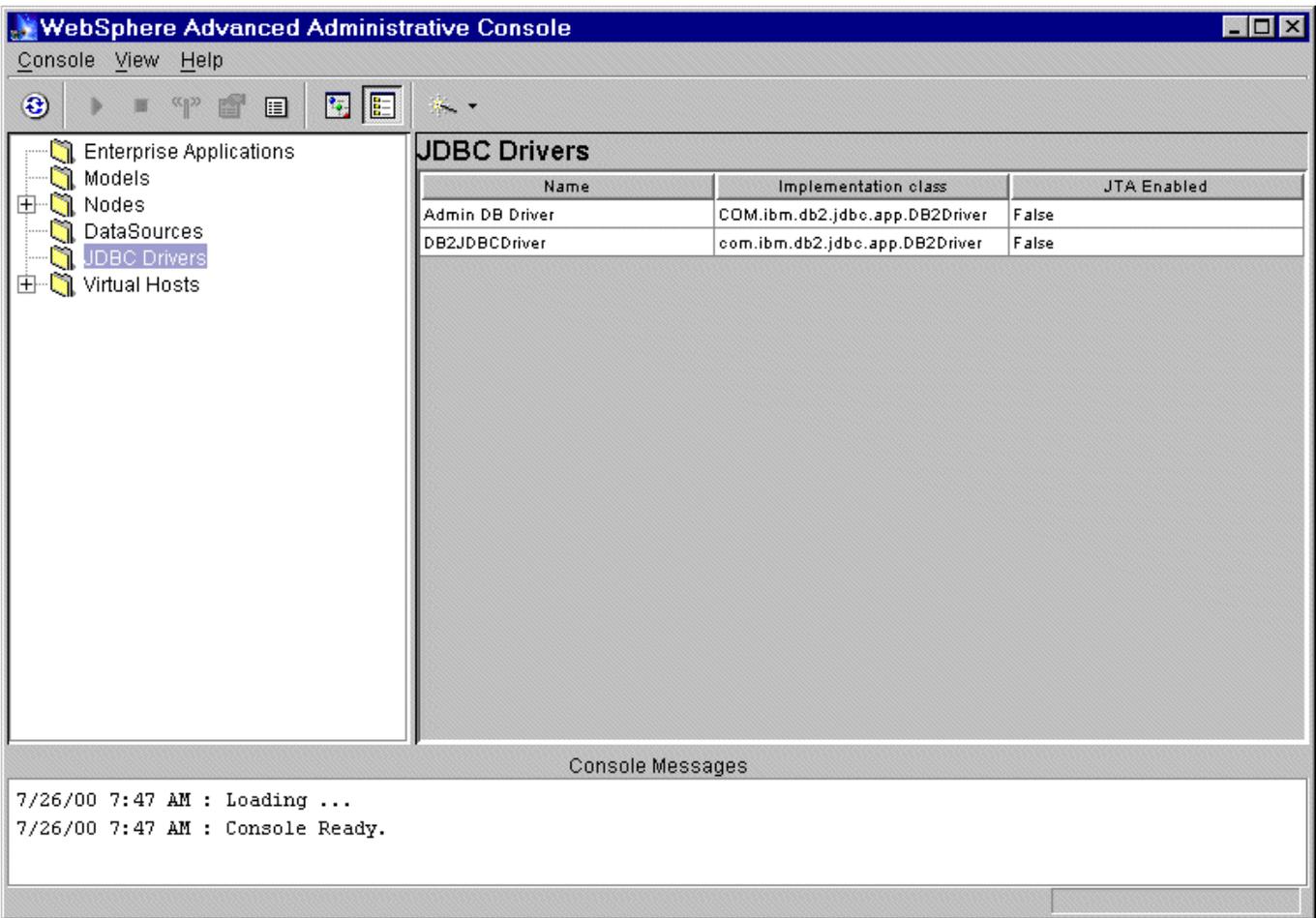
When you finish specifying the properties, click OK:



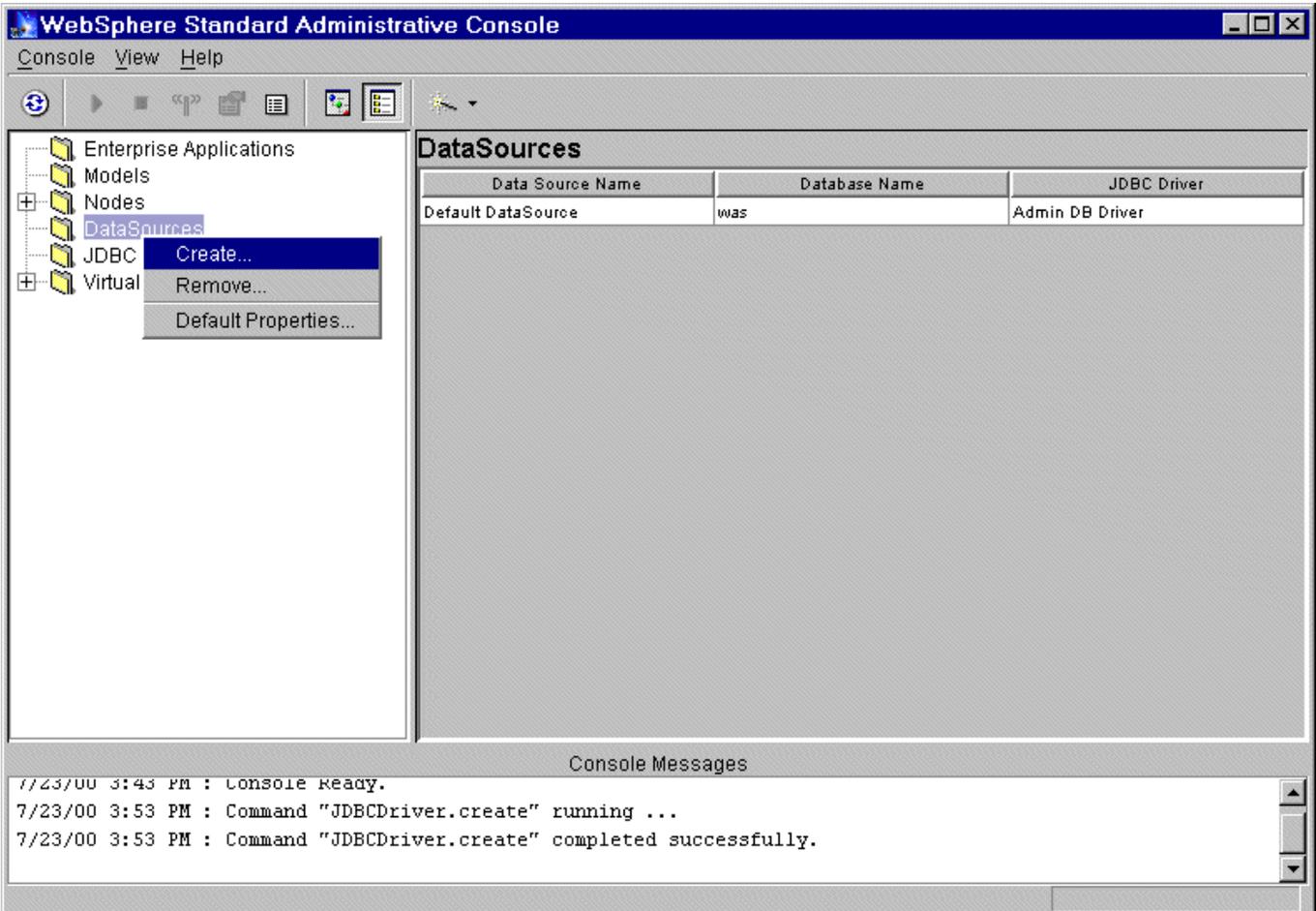
A confirmation dialog verifies that you configured the driver successfully:



The main console window displays the driver in the JDBCDrivers list:



Next, configure a data source that specifies connection pool settings for your database brand and version. In the Types tree, right-click the DataSources folder and click Create:



The data source properties dialog will be displayed:

General | Advanced

* Data Source Name:

* Database name:

* Driver: ExpiringPageDB2Driver

* - Indicates a required field.

OK Cancel Clear

Specify properties for the data source:

Name

Specify ExpiringPageDB2Dsrc.

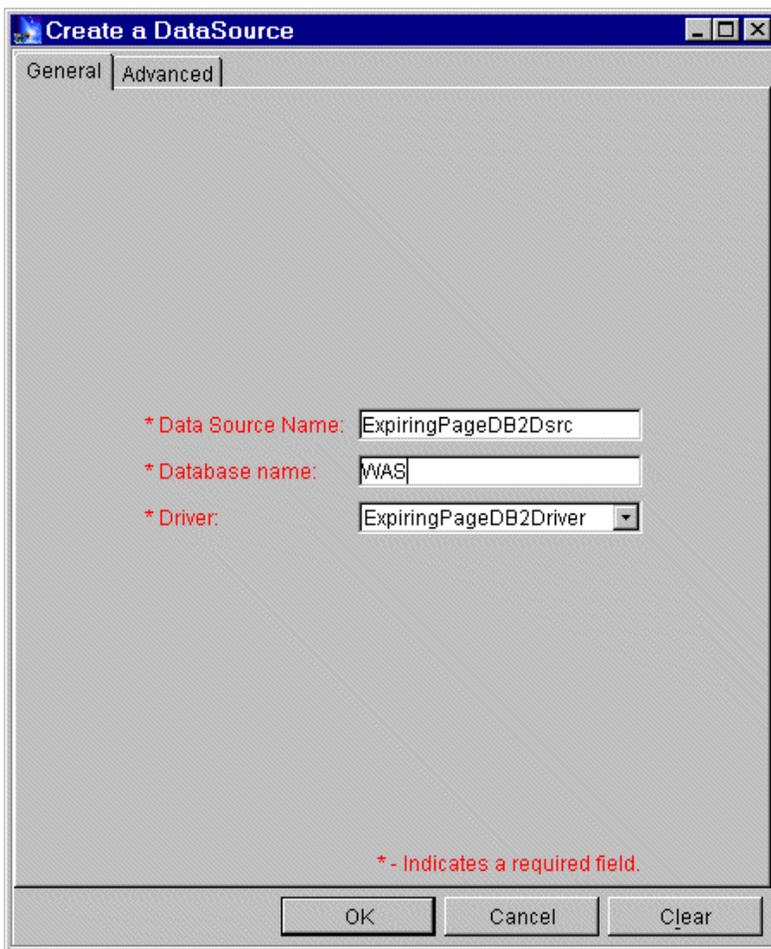
This is the name by which you will administer the data source. You have specified a name indicating that it is a DB2 database for the Expiring Page application's use.

Database Name

Specify WAS. This will be the name of the database you have configured for holding the servlet data (if you chose the default installation option, then this database was created for you automatically).

Driver

Select the ExpiringPageDB2Driver you configured in the previous step.

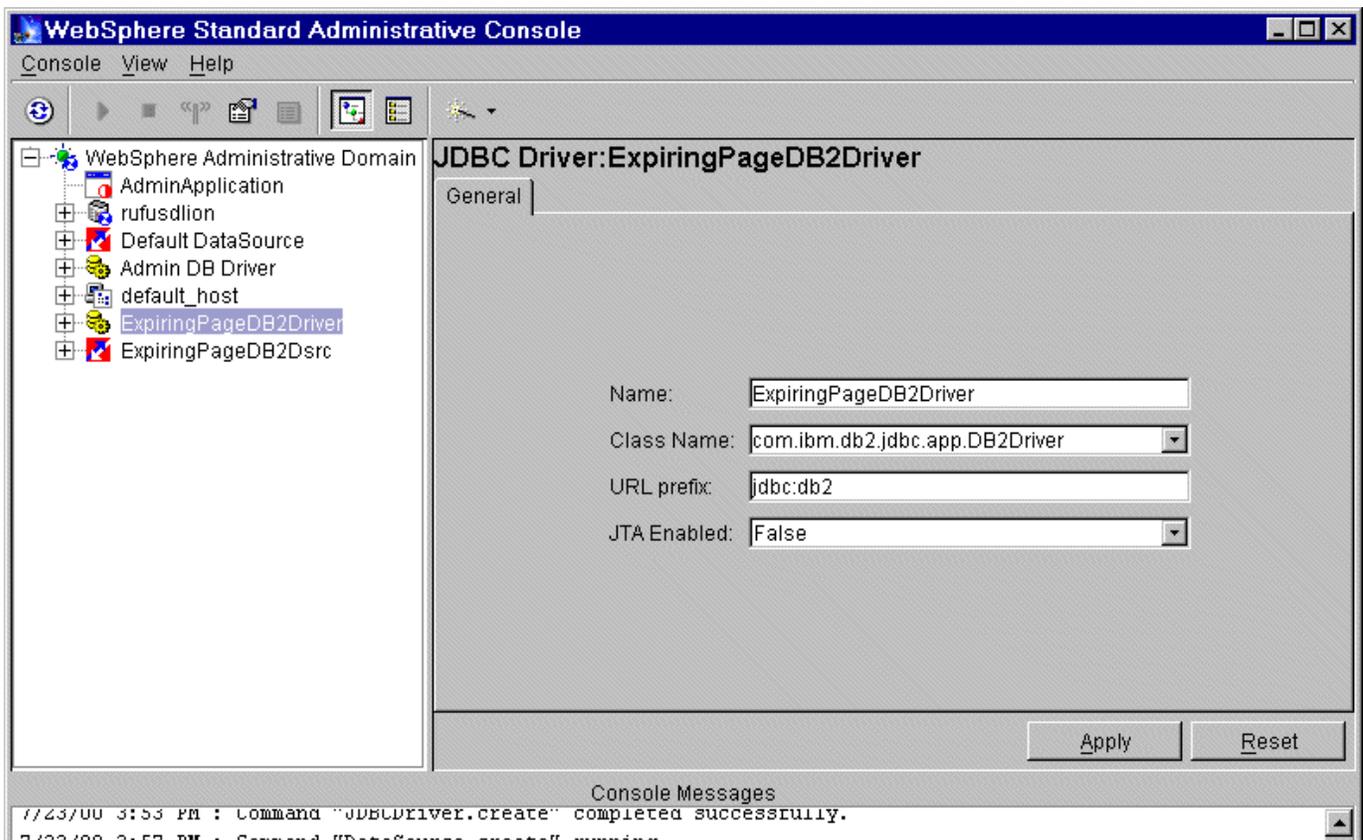


The remainder of the properties are optional. Click the Advanced tab to see the default values for the connection pool settings. In this case, leave the default values as they are.

- [Examine field help for data source properties](#)

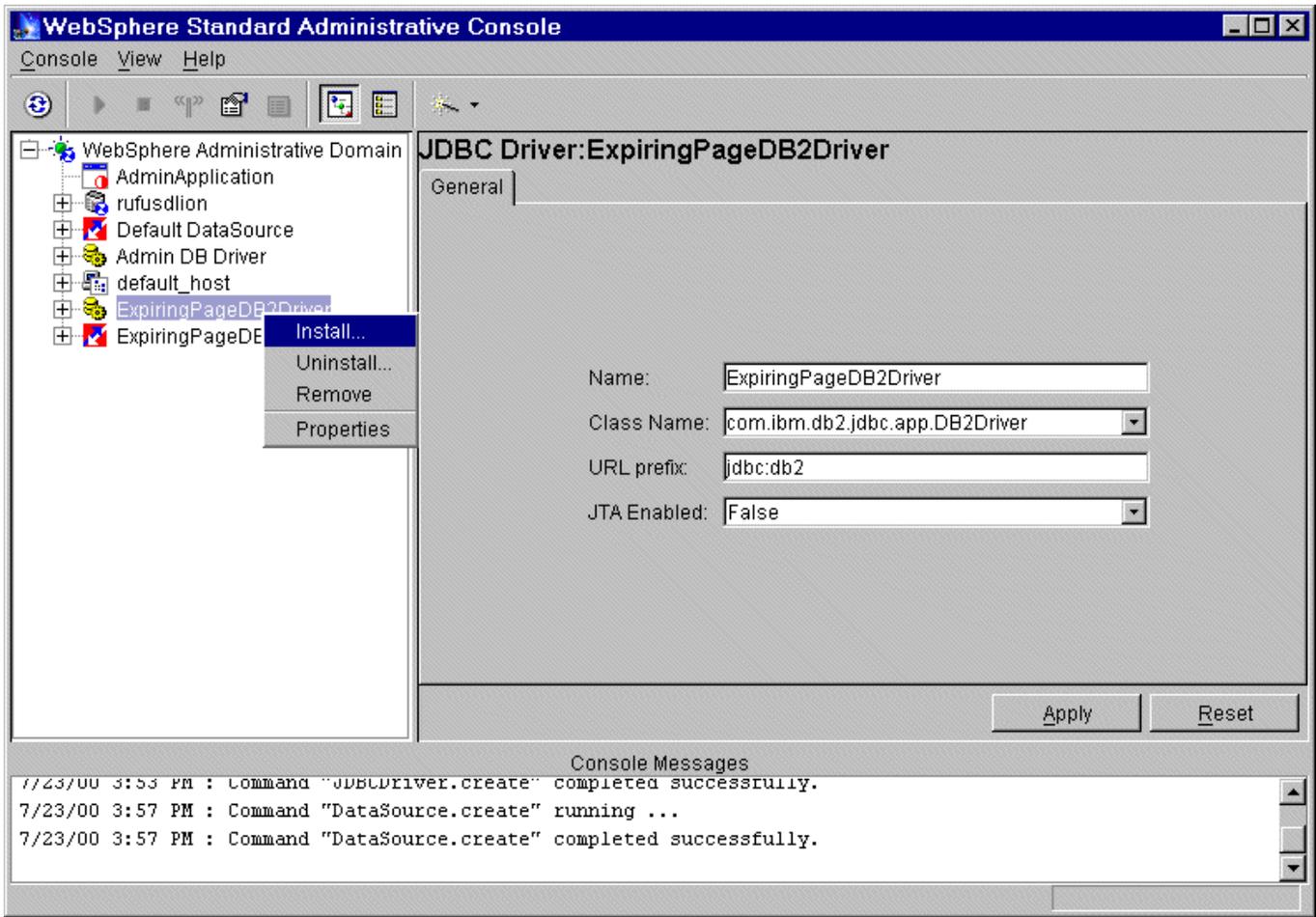
When you finish specifying the properties, click OK. A confirmation dialog tells you the data source configuration was successful.

Now that you have a data source, you can return to the JDBC driver and "install" it, specifying the location of the Java code for the driver. Switch to the Topology view, and the WebSphere Admin Domain. You will see, among other things, the JDBC driver and data source you configured:

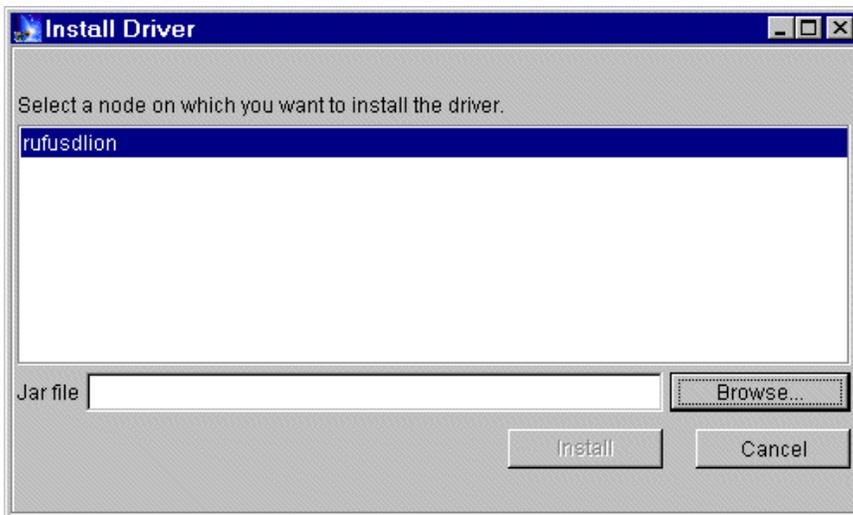


```
7/23/00 3:57 PM : Command "DataSource.create" completed successfully.
```

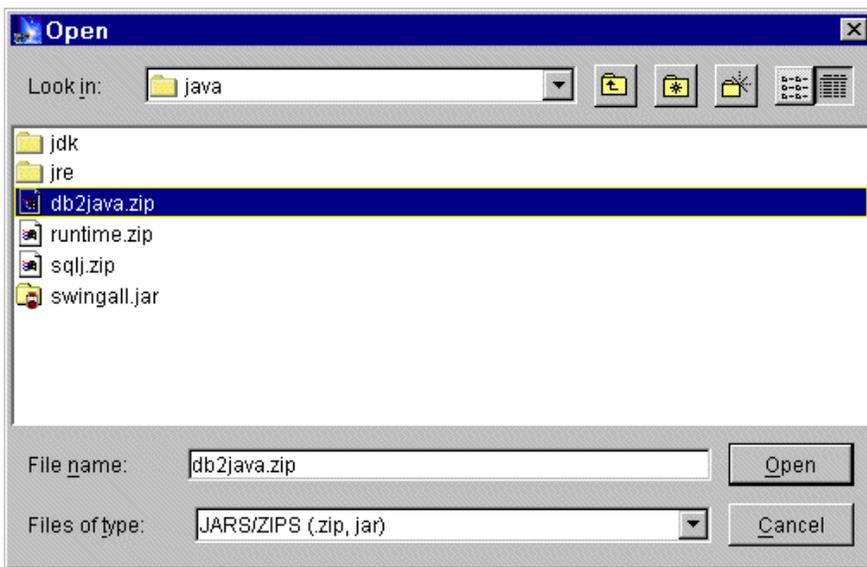
Click the ExpiringPageDB2Driver. Its properties are displayed on the right side of the console. Right-click it and select Install:



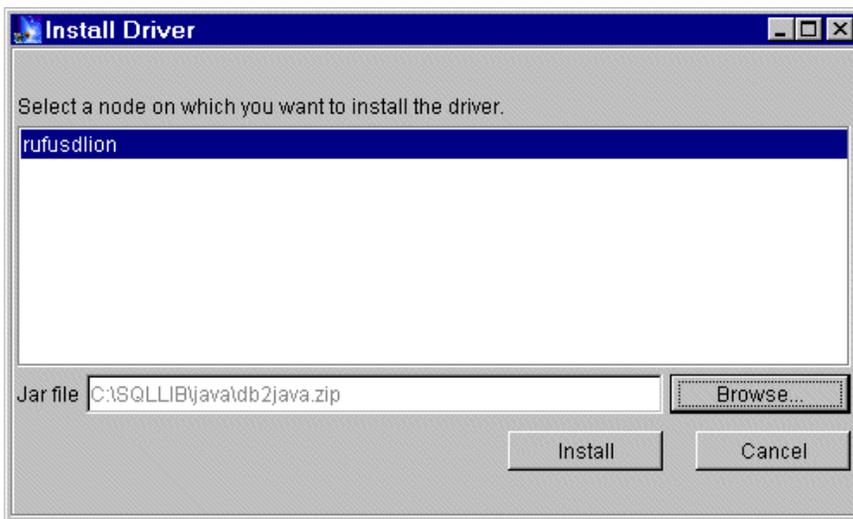
Select your local machine (shown here as rufusdlion) as the node on which to install the driver:



When you do that, the Browse button will become active. Click it to browse for the DB2 JDBC driver code. It should be in the sqllib\java directory of your DB2 installation root:



When you locate the db2java.zip file, select it and click the Open button to return to the driver installation dialog. The zip file will be displayed in the Jar file field:



Click the Install button. Note, this operation does not change the location of the db2java.zip file. It simply specifies the location of the file to the WebSphere administrative server.

Look for the confirmation dialog.

That's it. You now have the resources necessary to support database access by your applications.

Configure a Web application

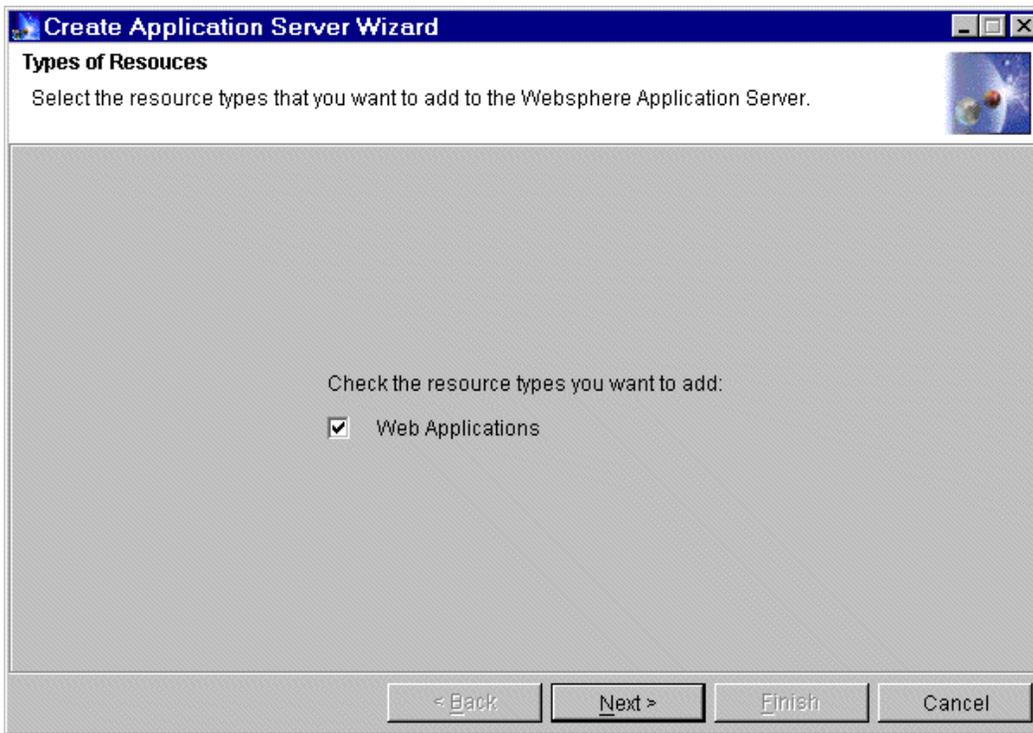
In this step, you will configure a *Web* application containing your servlet, JSP, and HTML files. You will also configure an application server and servlet engine to support your Web application.

In the next step, you will wrap the Web application with an *enterprise application* for the purpose of applying security.

Let's begin.

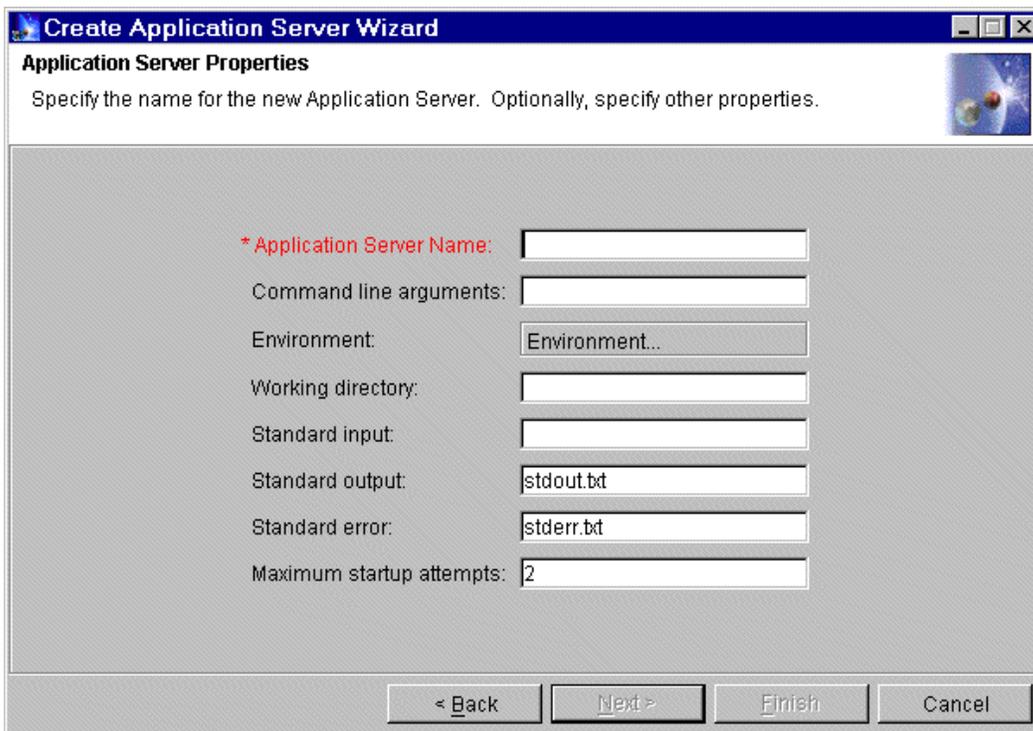
Press the Wizards button, and select Configure an application server (the first task on the pulldown menu).

The first page of the Wizard is displayed:



On this first panel, you choose whether to configure just the application server, or also the Web application support, including a Web application and a servlet engine. In this case, select Web Applications. Click Next.

The next screen allows you to configure an application server to handle applications containing servlets:



Specify properties for the application server:

Application Server Name

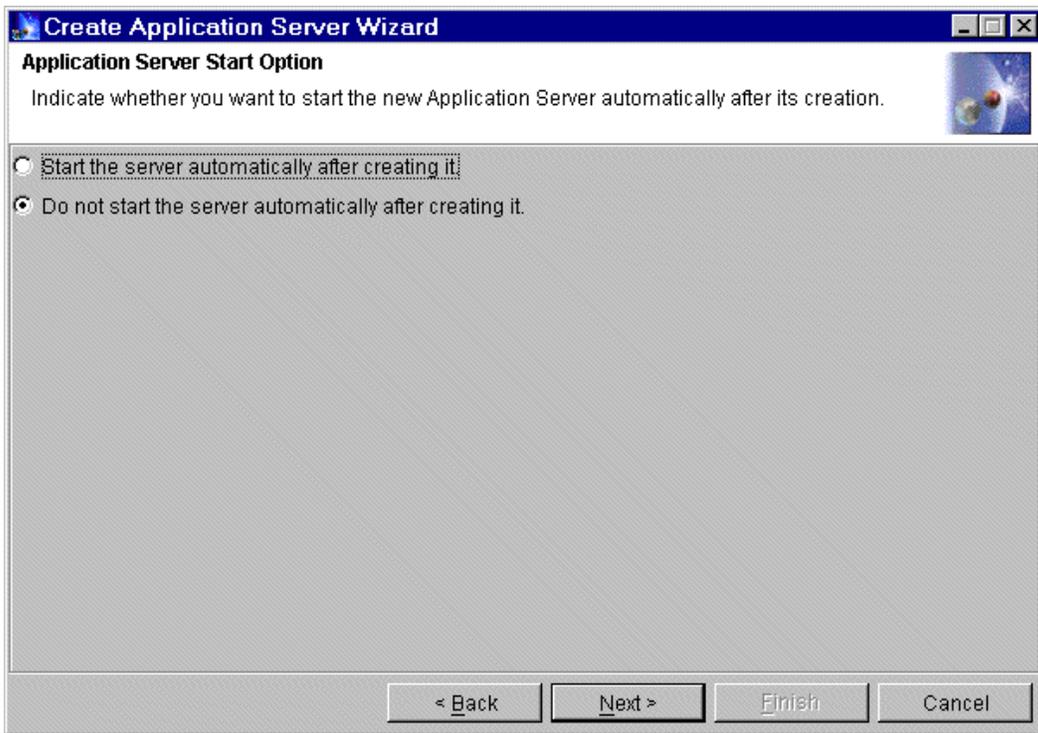
Specify MyAppServer.

This is the name by which to administer the application server.

The remainder of the properties are optional. Do not specify anything for them.

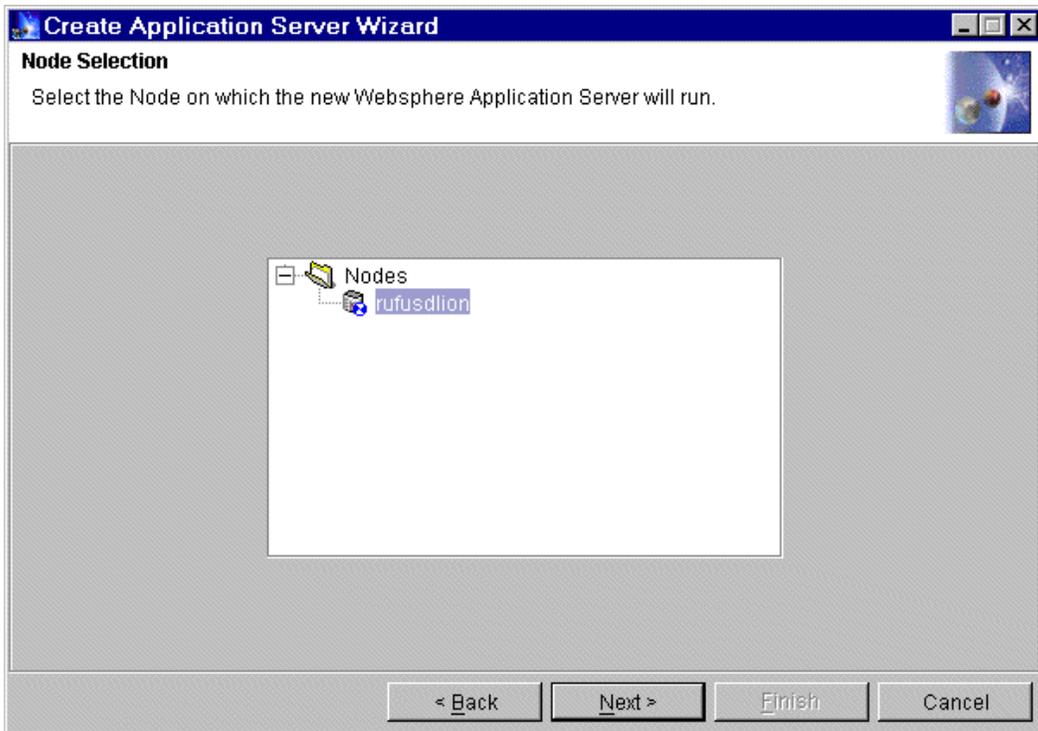
- [Examine field help for application server properties](#)

Click Next to continue. The next page lets you decide whether to start the server after configuring it:

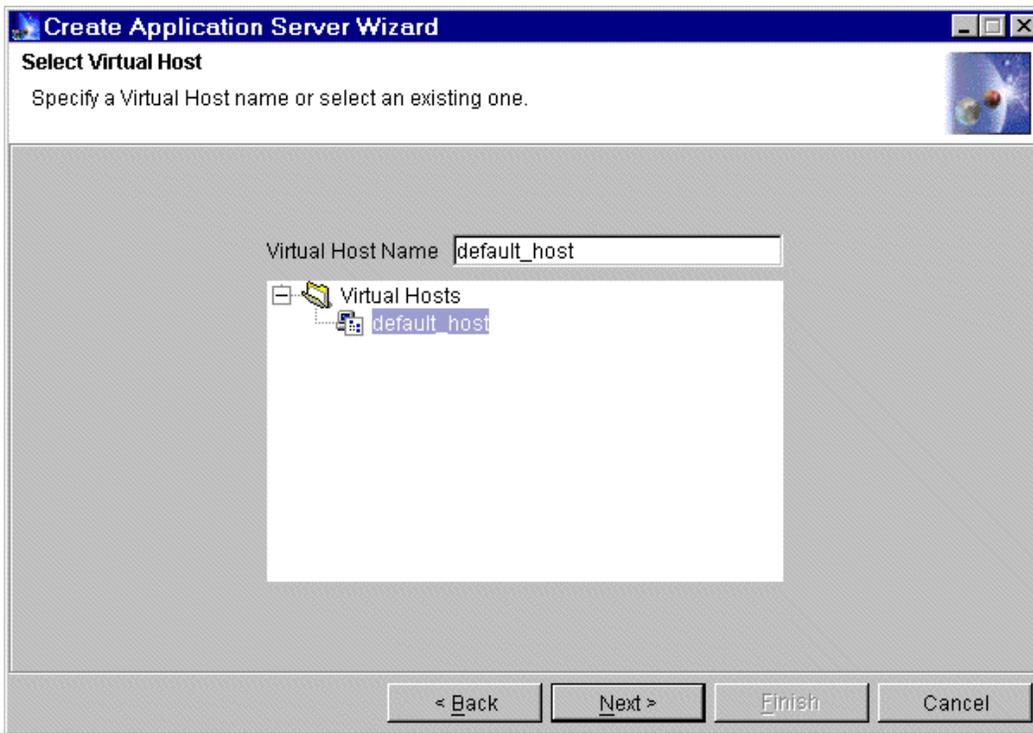


Specify that you do *not* want to start the server automatically. Later, you can practice starting it from the Topology tree.

Click Next to proceed. Now specify the physical machine on which the application server will reside. In this case, select your local node. The hostname in this example is rufusdlion. On your installation, it will be the hostname of your machine.

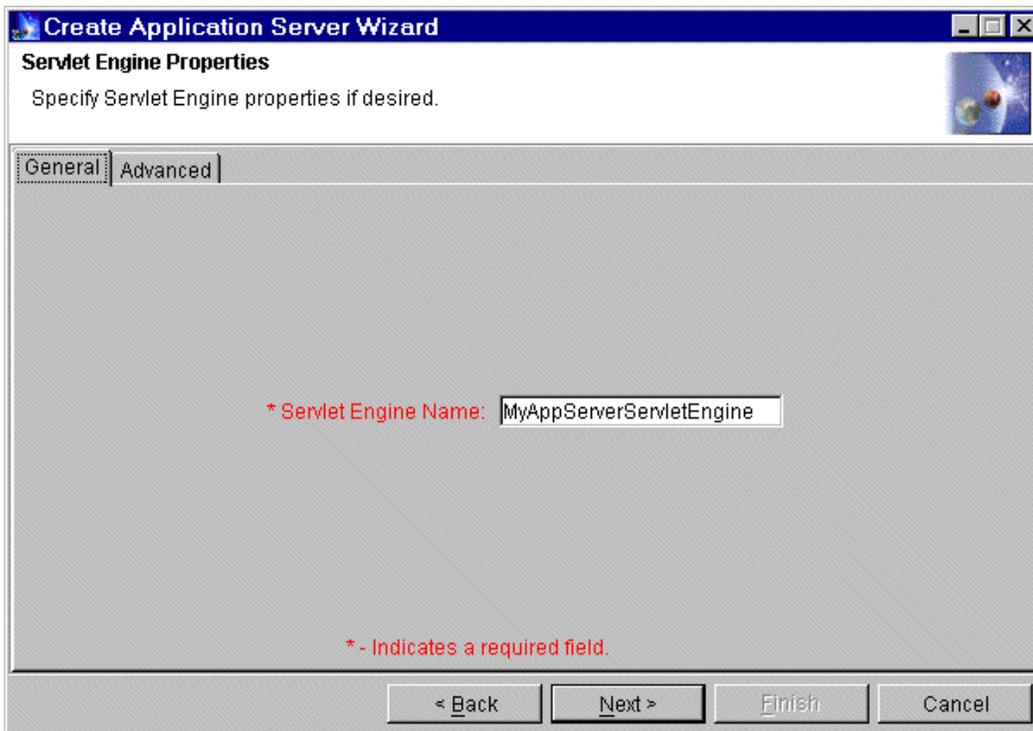


Click Next to proceed to a panel for selecting a virtual host. Recall, you are going to use the default virtual host. Expand the Virtual Hosts folder to display the default_host:



Notice there is also an option here for configuring a new virtual host, by specifying its name in the Virtual Host Name field. If you did that, the next panel or two of the wizard would let you configure the virtual host properties.

Select default_host and click Next to proceed. You will not see any virtual host configuration panels because default_host is already configured. Instead, you will see servlet engine properties:



Specify properties for the servlet engine:

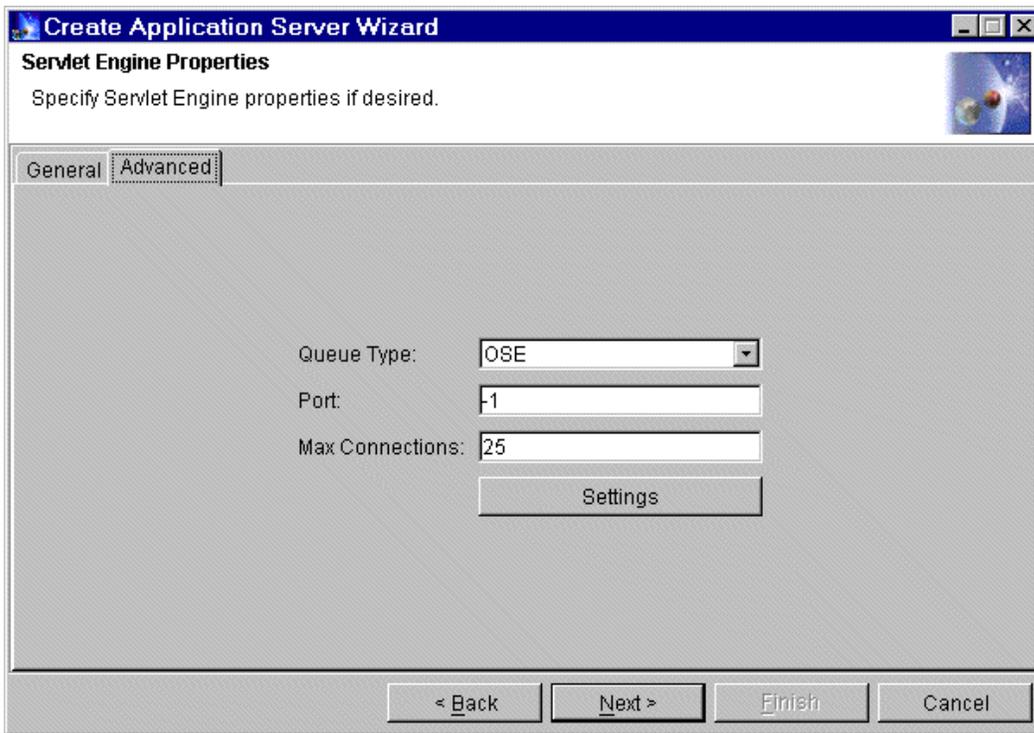
Servlet Engine Name

Specify MyAppServerServletEngine. This should be the default value anyway.

This is the name by which to administer the servlet engine.

The remainder of the properties are optional. Do not specify anything for them.

To see the settings for communication between the servlet engine and the Web server (for routing requests for servlets from the Web server to the servlet engine), browse the Advanced tab. Notice the Settings button:



- [Examine field help for servlet engine properties](#)

Click Next to proceed. Now you are ready to configure a Web application comprised of the servlet, JSP, and HTML files for the Expiring Page application.

Specify the following properties:

Web App Name

Keep the default value, MyAppServerWebApp.

This is the name by which to administer the Web application, and part of the path by which users can access the Web application from a browser. That will be discussed shortly. For your own Web applications, you might consider more descriptive names.

Virtual Host

Specify default_host.

Web Application Web Path

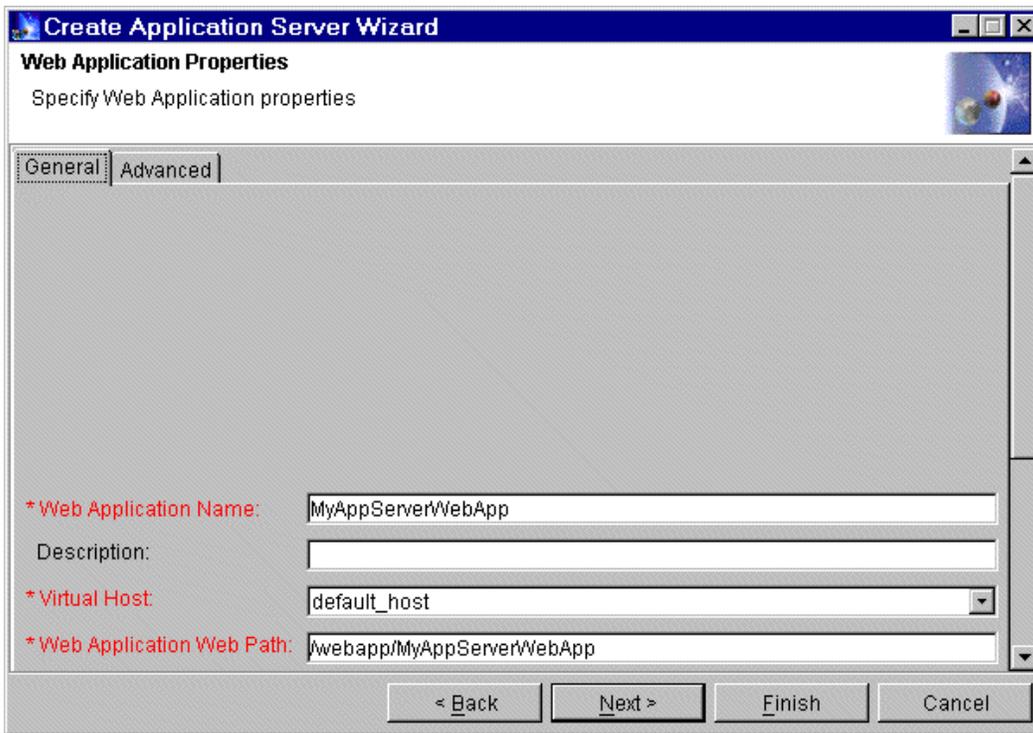
Keep the default value.

Together, the virtual host and the Web path indicate a path for accessing the Web application from a browser:

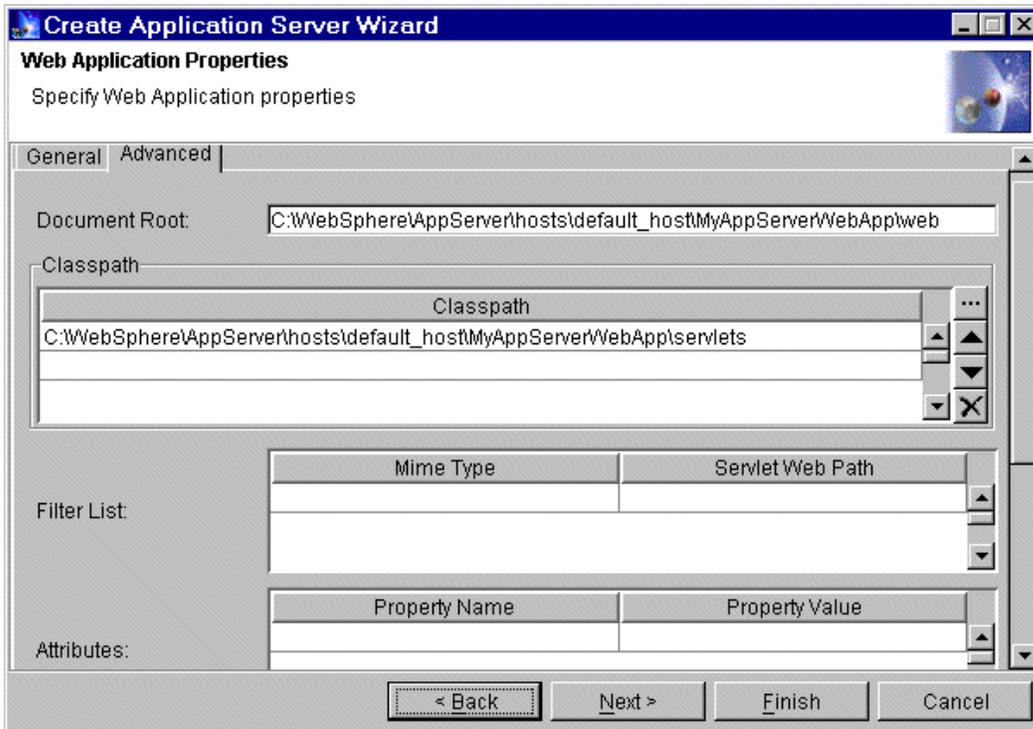
`http://Valid_default_host_alias/webapp/MyAppServerWebApp`

where *Valid_default_host_alias* is any of the aliases you saw earlier in the default_host properties -- the short or fully qualified name of the local machine, the loopback address, "localhost," and so on.

The Web application properties should now look like this:



Click the Advanced tab to view some additional properties.



I If you are performing Step 2b, advanced application configuration, [link back to that section now](#) -- this is the point at which the advanced application configuration procedure diverges from the basic application configuration procedure. If you are performing Step 2a, the basic application configuration, ignore this note and continue.

These Web application properties are most notable:

App Doc Root

Accept the default value, *install_root*\hosts\default_host\MyAppServerWebApp\web

This specifies the location of the HTML and JSP files for the application. After completing this task wizard ("Configure an application server"), you are going to move the files there.

App Classpath

Accept the default value, *install_root*\hosts\default_host\MyAppServerWebApp\servlets

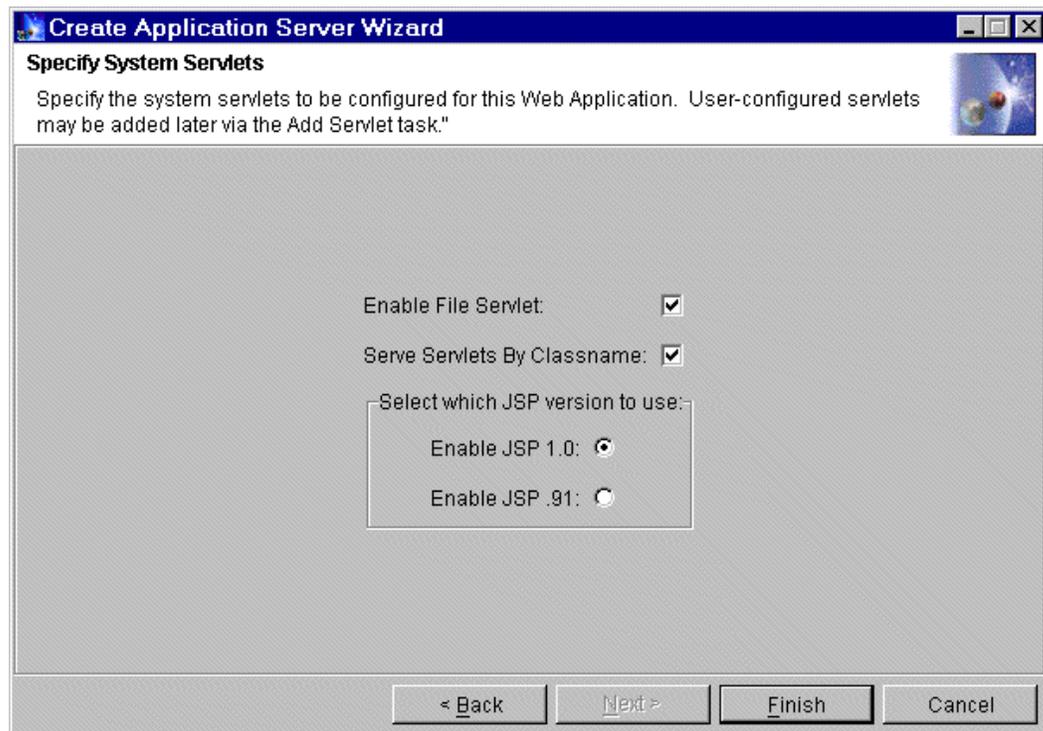
This specifies the location of the servlets for the application. You will later move the servlet class file there.

As with the above settings, it is optional that you change the rest of the settings from their default values. In this case, accept the default values.

- [Examine field help for Web application properties](#)

Click Next to proceed. On this panel, you can add system servlets to the Web application. System, or internal, servlets are servlets shipped with WebSphere Application Server to

support various functions in your Web applications:



Specify which system servlets to use. For the basic application configuration, this is especially important.

Enable File Servlet

Select this check box. It is required because your Web application contains HTML and JSP files.

This enables Web pages to be served from the App Doc Root you specified on an earlier Web application configuration panel. It adds a file servlet to your Web application for this purpose.

In addition to selecting this check box, make sure your Web server configuration file does not contain any pass rules that will override the App Doc Root.

Serve Servlets By Classname

Select this check box. It is required *if* your Web application contains servlets and you do not plan to configure the servlets explicitly in the administrative domain.

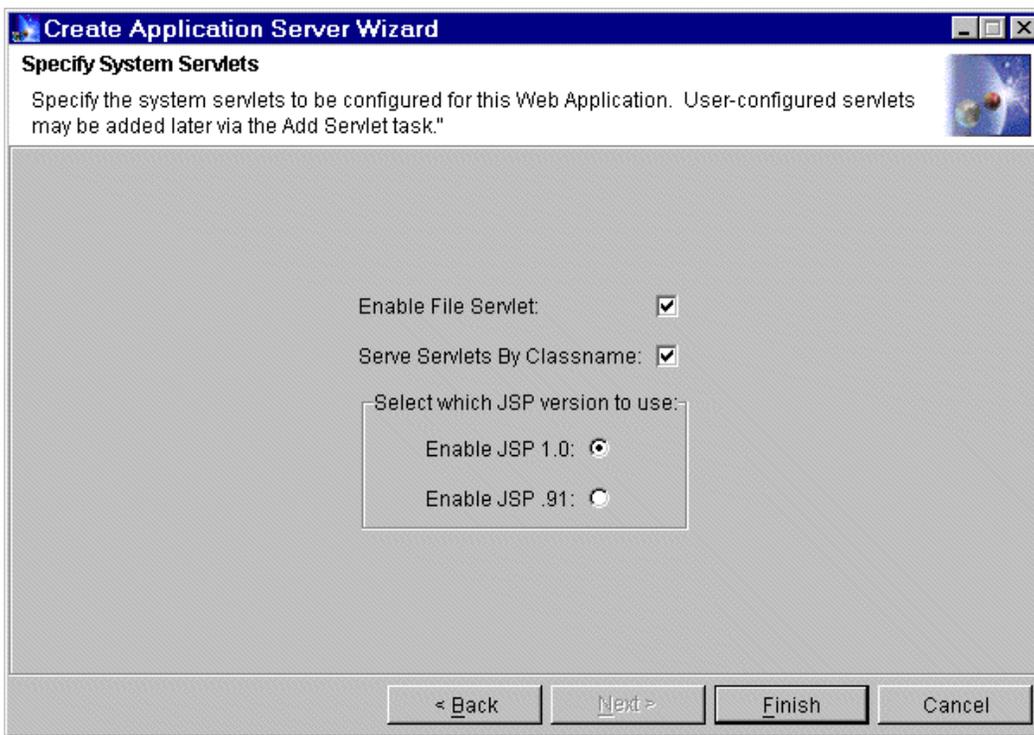
This enables servlets to be served from the servlets directory named in the Web application classpath. Any servlet class you drop into the directory can be loaded by the WebSphere administrative server, without requiring that you identify the servlet by name.

Enable JSP 1.0, Enable JSP .91

Select JSP 1.0. It is required because your Web application contains JSP files.

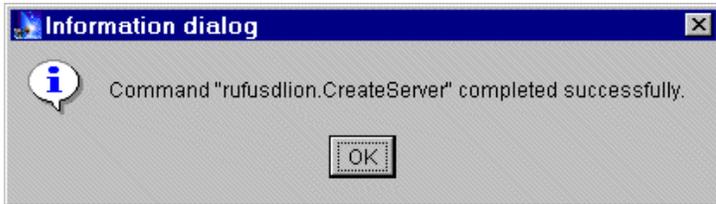
This specifies to include the JSP enabler servlet supporting the 1.0 level of the JavaServer Pages (JSP) specification. The JSP file for the Expiring Page application is written against the 1.0 specification.

 All three of these settings are required in order for your Web application containing HTML, servlets, and JSP files to work correctly. Make sure they are all selected:



Here's a tip. Suppose you forget to select one of the check boxes. You can use the "Add a servlet" task on the Wizards button to add the file servlet, enable servlets to be served by class name, and add a JSP enabler.

Now you are ready to finish configuring the application server and its contents. Click Finish and await the confirmation dialog:



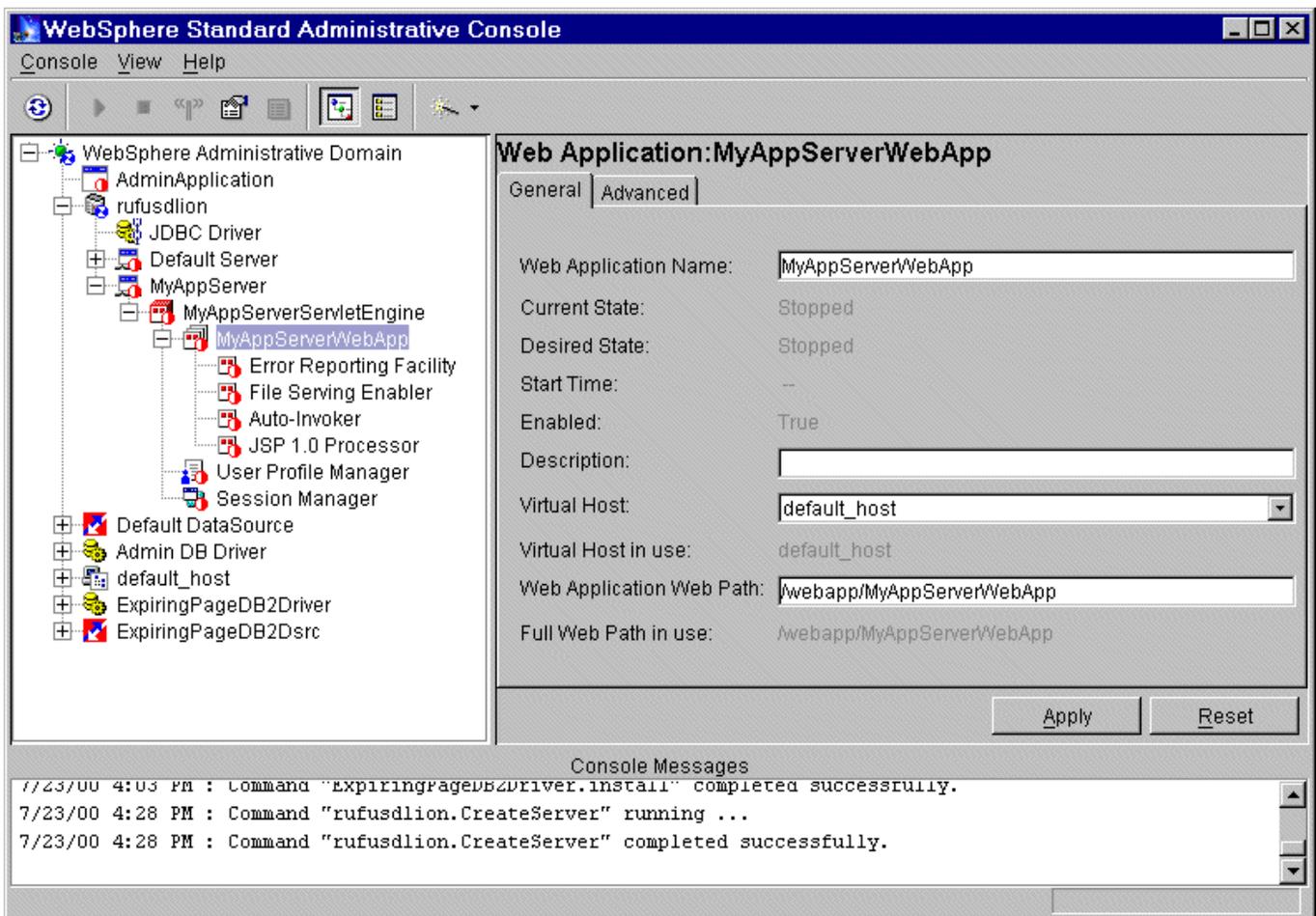
You have configured many resources. Use the Topology view to view them. Notice that when you click a resource, such as MyAppServer, its properties are displayed on the right side of the console. You can edit most of them, including adding values for properties you formerly left blank.

You can confirm that the MyAppServerWebApp contains the system servlets you intended. Under the MyAppServerWebApp in the Topology tree, you can see the:

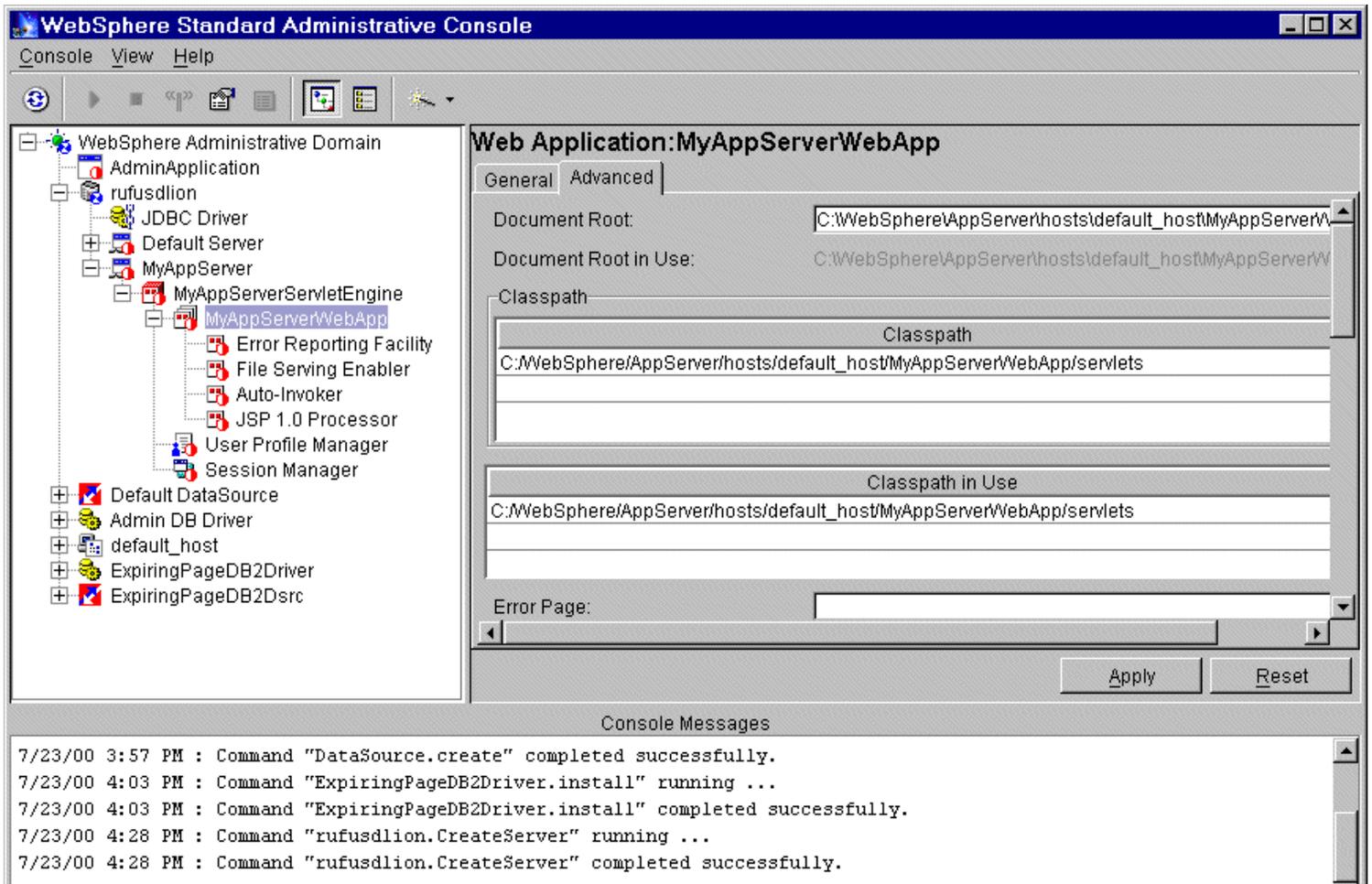
- "File Serving Enabler" for serving HTML and JSP files from the App Doc Root
- "Auto-Invoker" for serving servlets by classname
- "JSP 1.0 Processor" supporting ExpiringHTMLBeforeDatePage.jsp and ExpiringHTMLAfterDatePage.jsp

Notice you do *not* see the ExpiringHTMLServlet servlet listed under the Web application. This is because you have used the basic application configuration procedure in which you do not specify servlets by name in the administrative domain. You would have to configure a servlet resource in the administrative domain and associate it with the Web application in order to see it displayed here.

You can click MyAppServerWebApp to view its properties:



If you forget its document root or classpath values, you can visit the Advanced page:



These directories are where you will be placing the files in the next step.

One more note -- now that you have configured some resources, you can use the Types view to "take inventory" of how many of each type you have. Press the Types view button, and click the ServletEngines folder to view your servlet engines:

The screenshot shows the WebSphere Standard Administrative Console interface. The title bar reads "WebSphere Standard Administrative Console". The menu bar includes "Console", "View", and "Help". The left-hand navigation pane shows a tree structure with "Servlet Engines" selected. The main content area displays a table titled "Servlet Engines" with the following data:

Servlet Engine Name	Current State	Start Time
Default Servlet Engine	Stopped	--
MyAppServerServletEngine	Stopped	--

Below the table is a "Console Messages" section with a scrollable list of log entries:

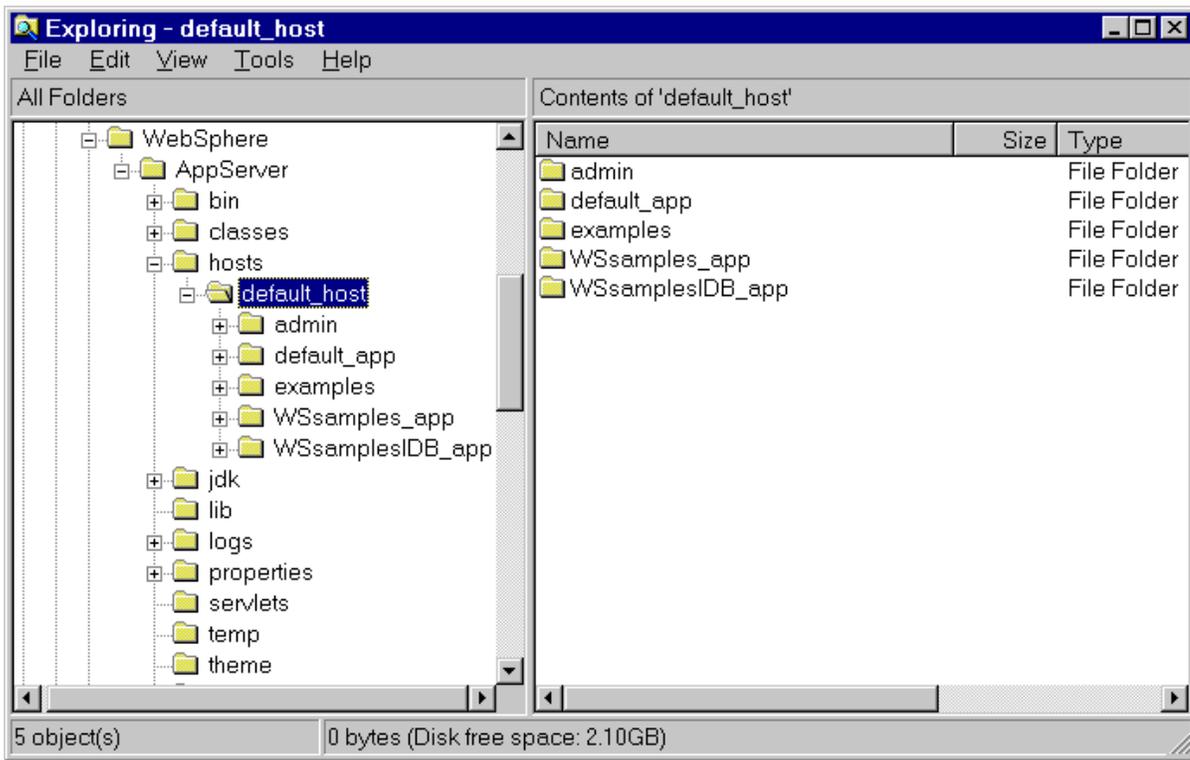
```
7/23/00 3:57 PM : Command "DataSource.create" completed successfully.  
7/23/00 4:03 PM : Command "ExpiringPageDB2Driver.install" running ...  
7/23/00 4:03 PM : Command "ExpiringPageDB2Driver.install" completed successfully.  
7/23/00 4:28 PM : Command "rufusdlion.CreateServer" running ...  
7/23/00 4:28 PM : Command "rufusdlion.CreateServer" completed successfully.
```

So, now you have a Web application and the infrastructure to support it and your Expiring Page enterprise application, once you configure one. You have also configured database access, though in this case you will not need it.

Place the application files in WebSphere directories

Now you must place the Web application files where WebSphere Application Server expects them to be, based on the Web application configuration you specified.

Currently, the directory structure looks like this:



1. Create any necessary directories to place the files in.

In this case, recall you accepted the default values for the Web application document root and classpath, which are:

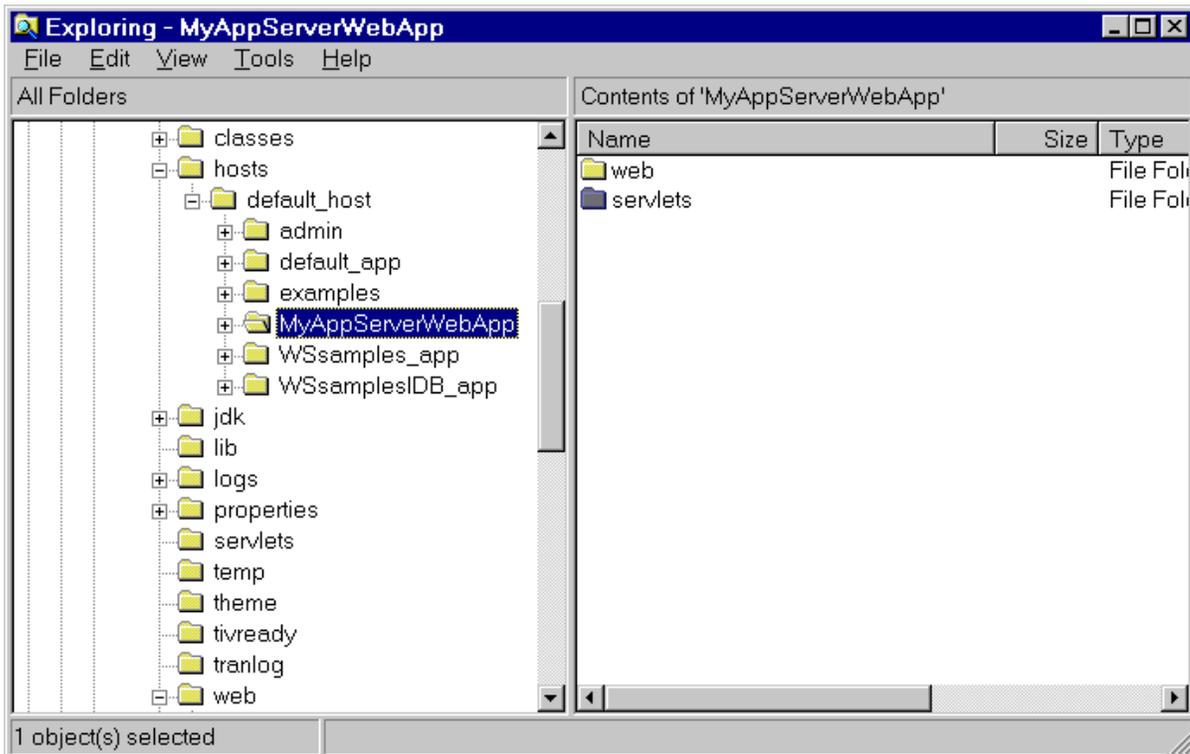
- o App Doc Root: install_root\hosts\default_host\MyAppServerWebApp\web
- o App Classpath: install_root\hosts\default_host\MyAppServerWebApp\servlets

In the WebSphere Application Server installation root, create the directory structure to support the Web application configuration.

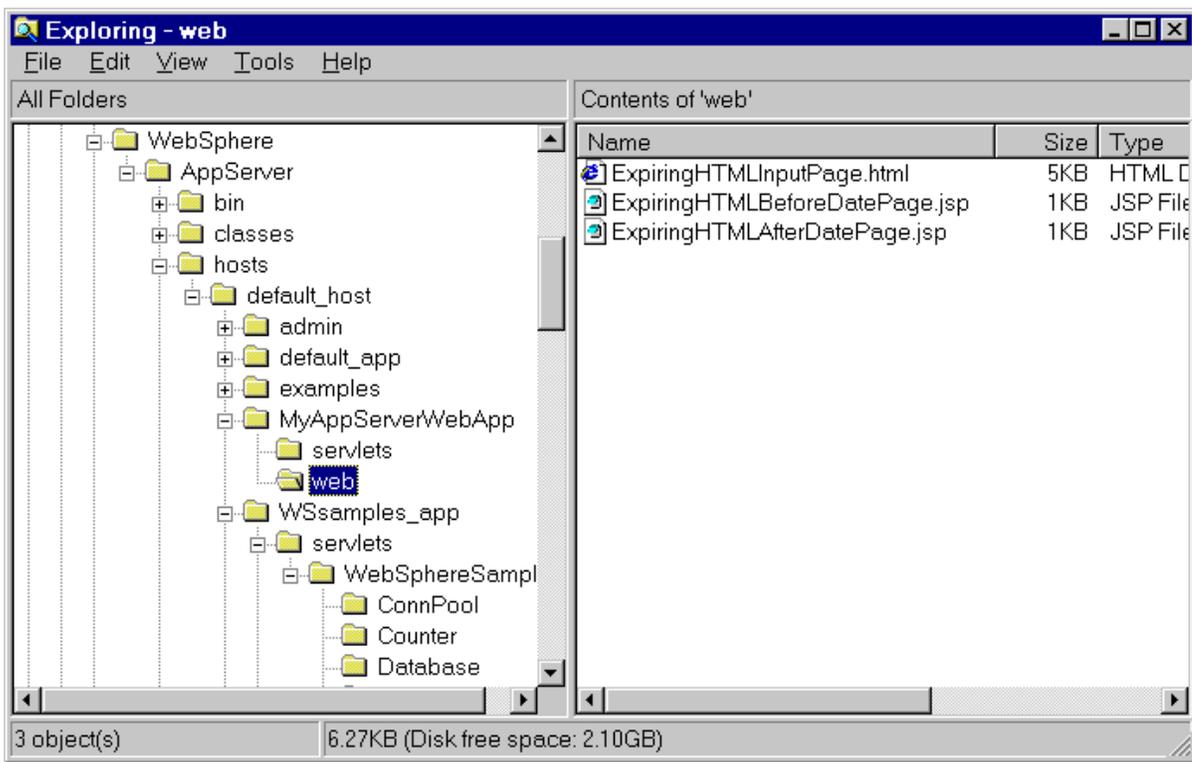
2. Place the HTML and JSP files in directory you specified as the document root of the Web application.
3. Place the servlet file in the servlet directory you specified in the classpath of the Web application.

[Click here](#) to review the names and locations of the HTML, JSP, and servlet files in the Expiring Page sample application.

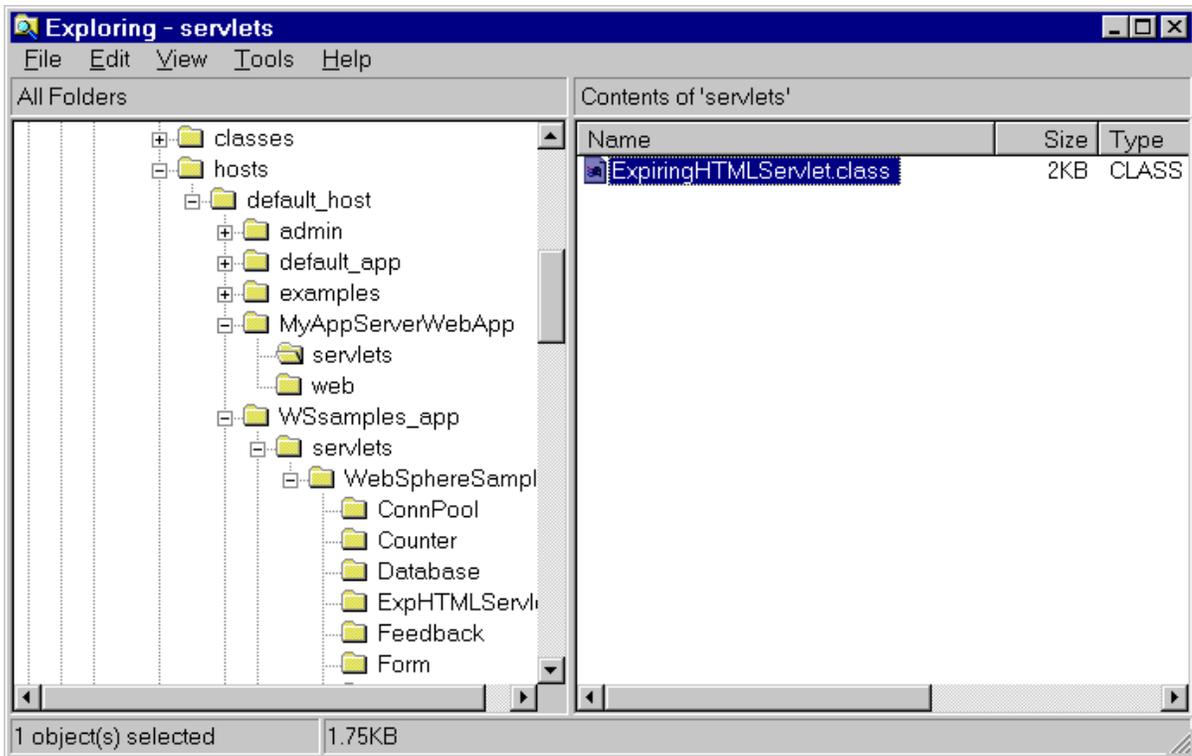
The directory structure on Windows NT now looks like this:



The HTML and JSP files are in the web directory under the Web application:



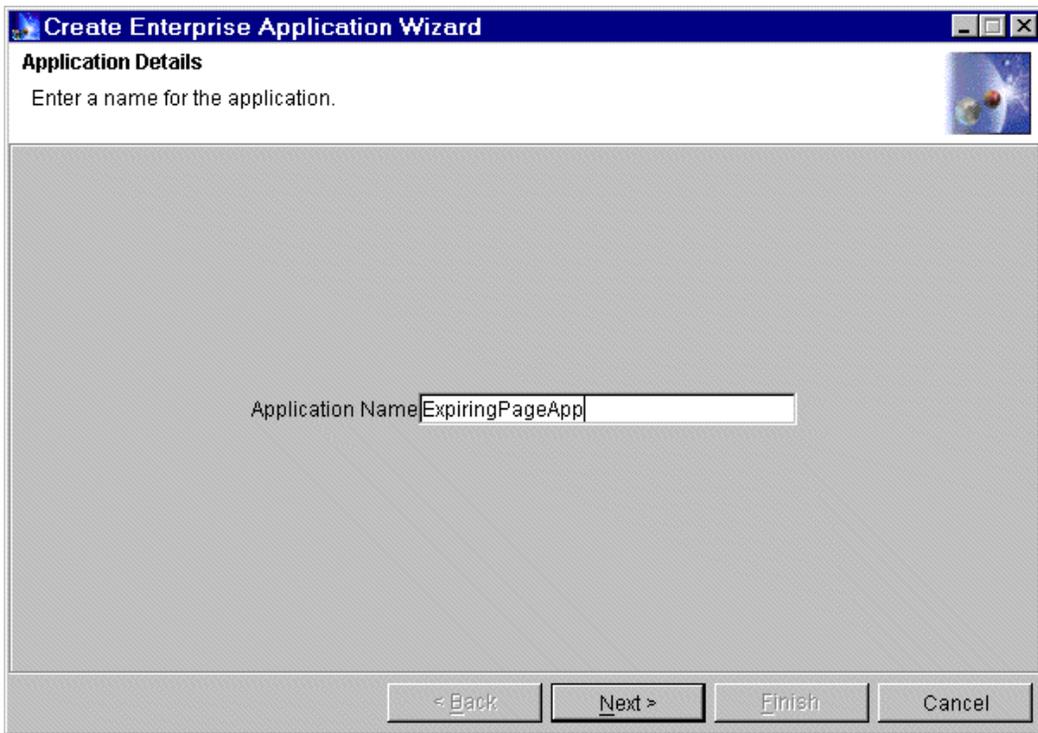
And the servlet is in the servlets directory:



Step 3: Configure an enterprise application

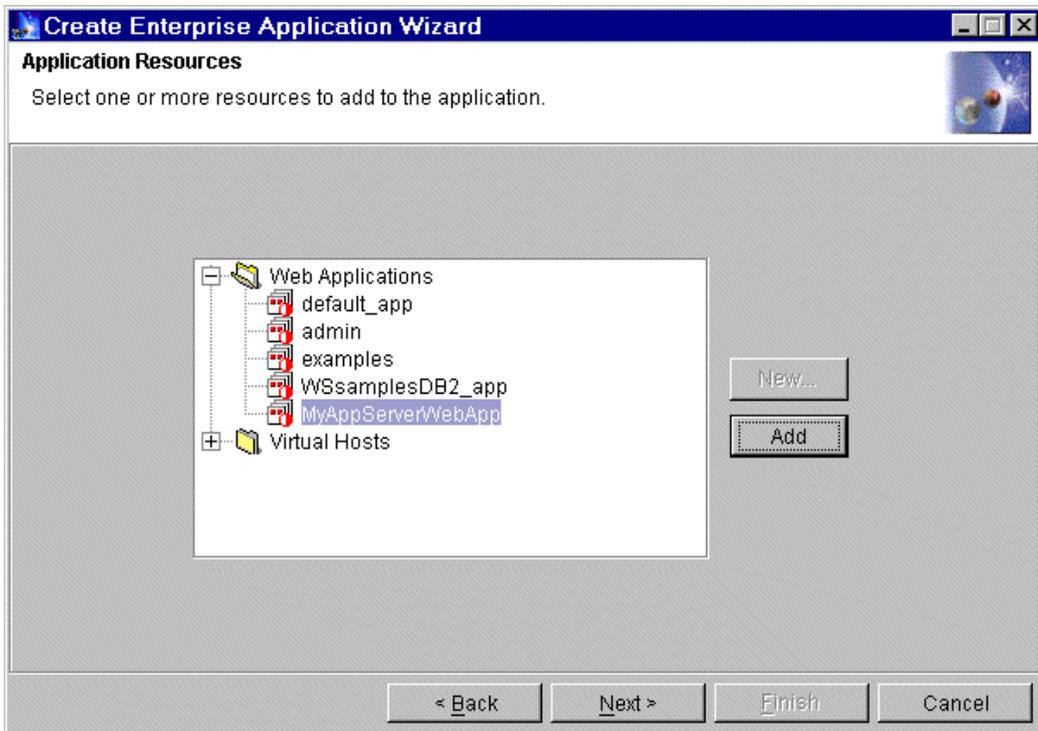
In this step, you will in essence "wrap" your Web application with an enterprise application for the purpose of applying security. This is necessary because the product applies security to enterprise applications, not to Web applications.

Press the Wizards button and select Create Enterprise Application:



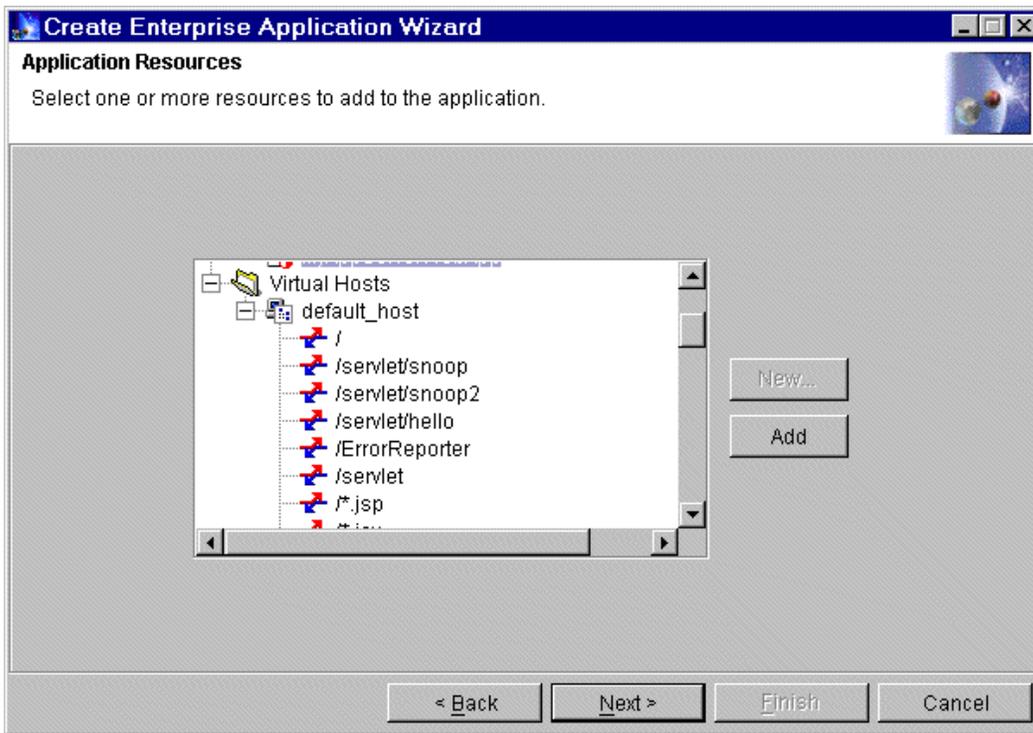
Name the enterprise application ExpiringPageApp. Click Next to proceed.

For the contents of the application, select the Web application you configured earlier, MyAppServerWebApp. Click Add:

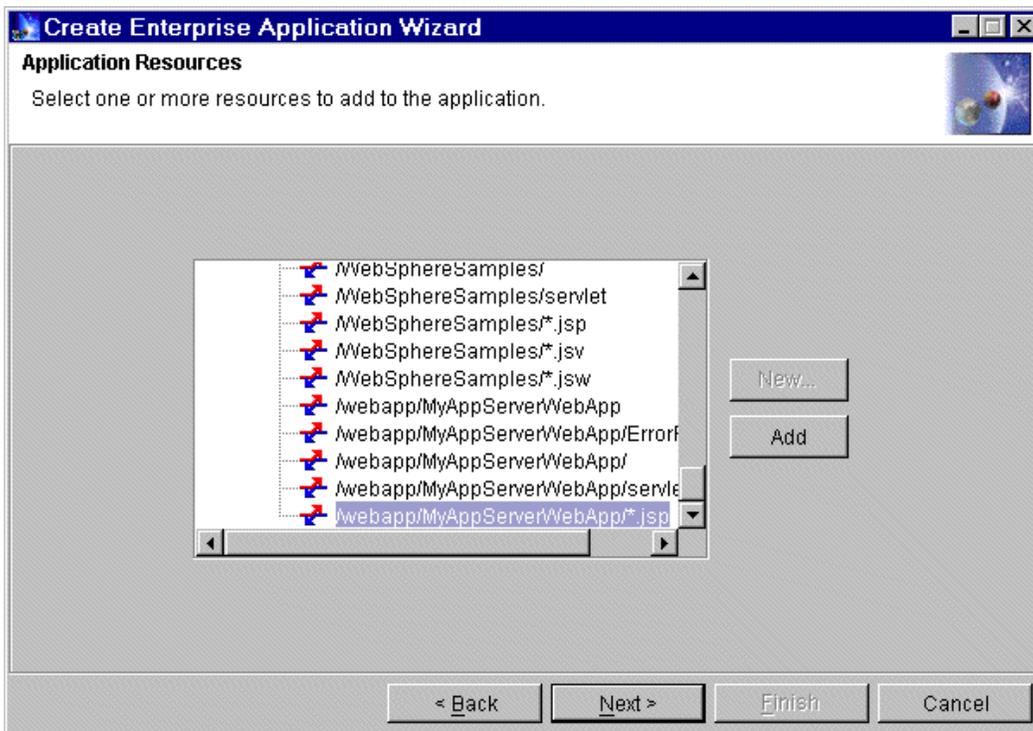


The Console Messages area displays a confirmation message that the operation was successful.

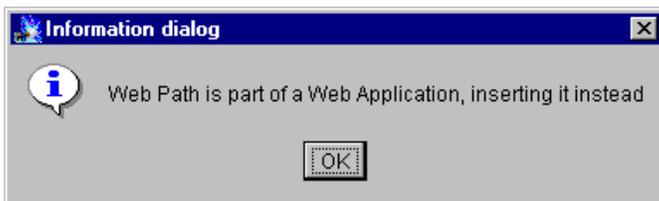
Click Next and expand default_host to see the many Web resources associated with it:



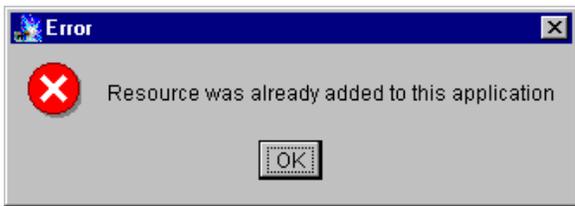
Scroll down until you see the Web resources for MyAppServerWebApp. Select the *.jsp Web resource:



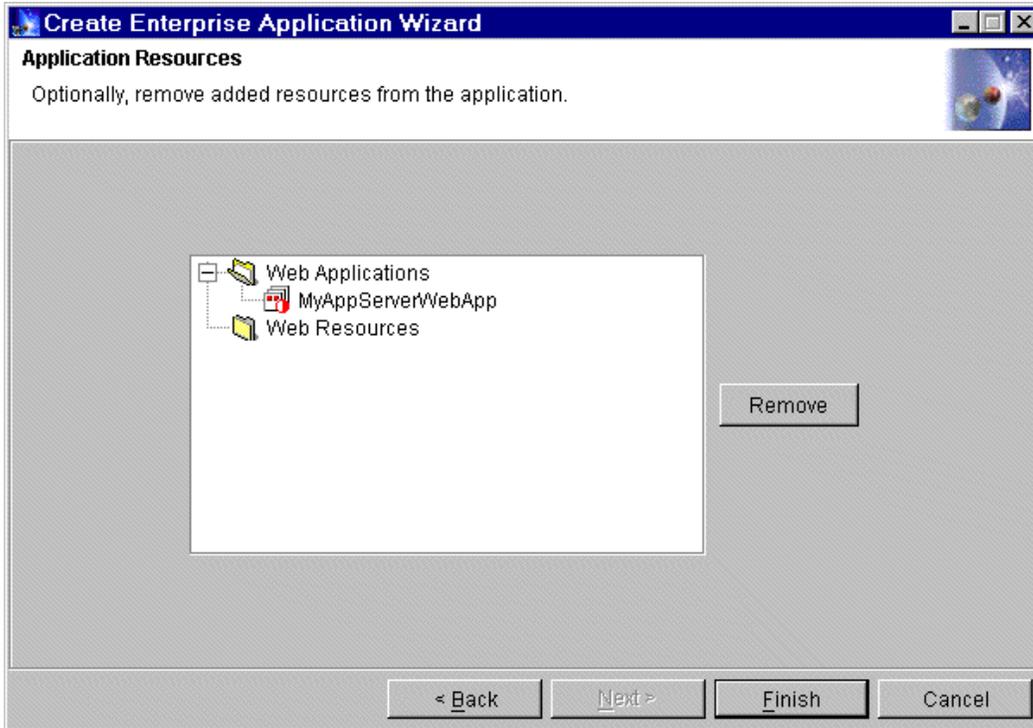
Click Add. It is okay if you receive this message:



If you try to add the same Web resource twice or more, you will receive this message:

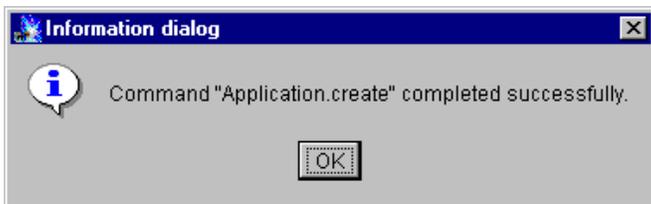


When finished, click Next to proceed. The next panel is for removing resources you did not really want to add. It also provides the opportunity to confirm that everything was added. Expand the tree to ensure all of the resources are there.

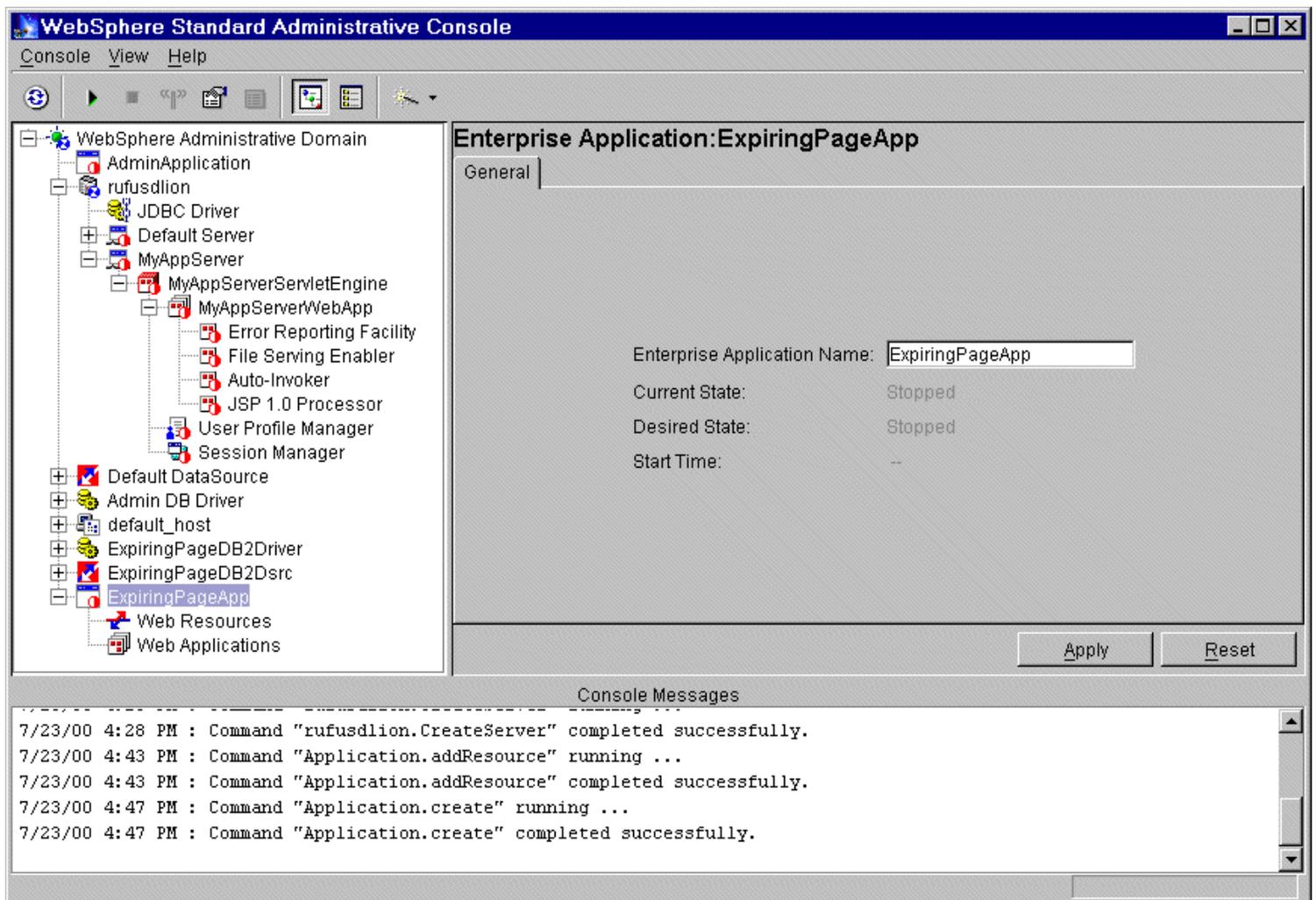


When satisfied with the content of the application, click Finish. If some files are missing, instead click the Back button to return to the panel for adding resources.

A confirmation dialog informs you that the application was successfully configured:



To view the properties of your new enterprise application, click the Topology button. Locate and click ExpiringPageApp:



Step 4: Secure the administrative domain and applications

Now that you have an enterprise application, you can secure it:

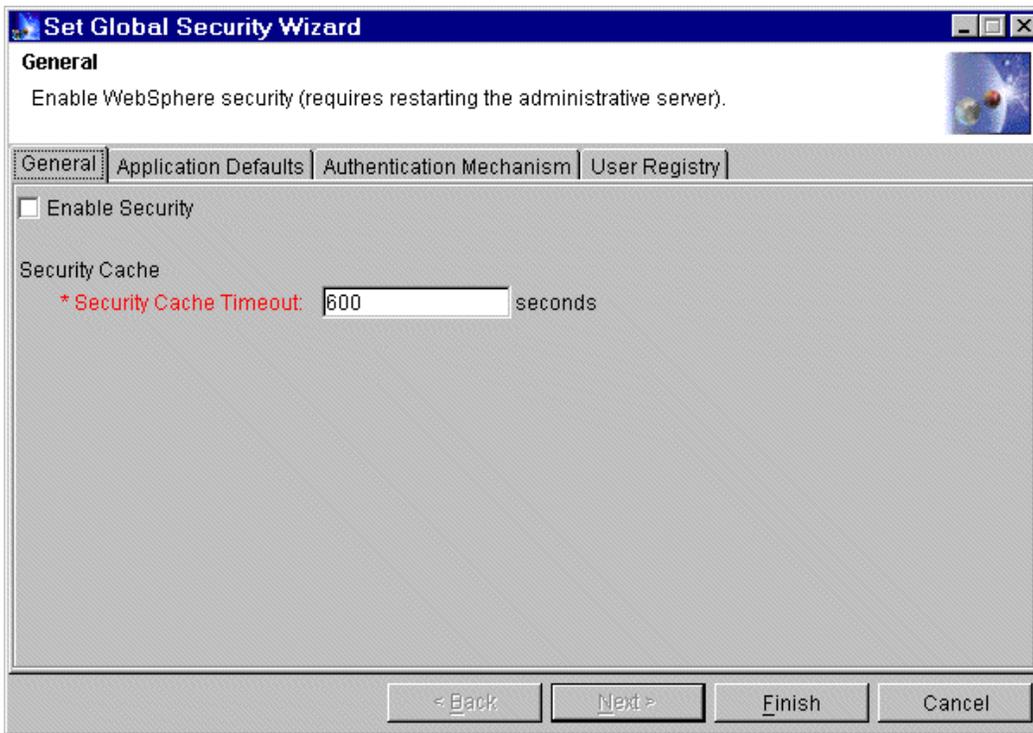
1. [Set up global security and method groups](#)
2. [Secure the application](#)
3. [Create custom method groups](#)
4. [Secure the Account Bean, JSP files, Web pages, and servlets](#)
5. [Grant security permissions](#)

Set up global security and method groups

Follow these instructions to establish default security settings for applications. New applications you create will use the defaults. You can also specify authentication and user registry information that will apply to all applications.

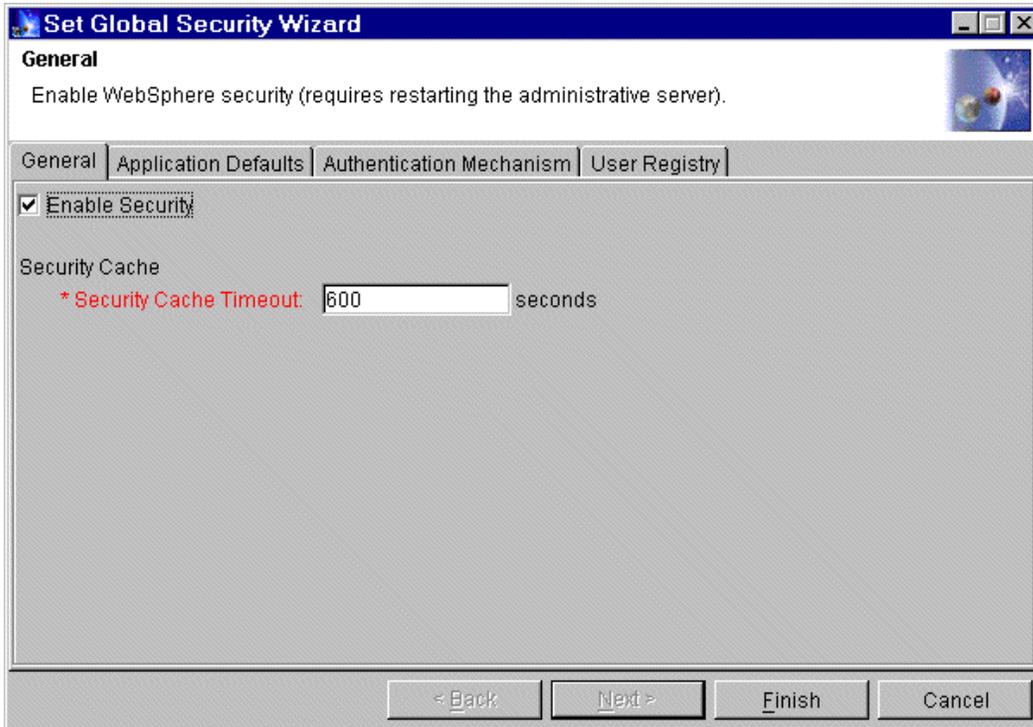
- [Read about this task: key concepts, settings, when and why to perform it](#)

Press the Wizards button and select Configure Global Security Settings. The first page of the task looks like this:



This page is where you enable WebSphere Application Server security. If the Enable Security check box is *not* selected, all security settings in this task and others will be disregarded.

Select the Enable Security check box:



Next, click the Application Defaults tab:

As its name indicates, this tab contains *default* settings for application security. In other words, when you configure an application, you can override or accept these default values for the individual application.

The rest of the settings in this task are *global*, meaning they are shared by all applications. They cannot be customized for particular applications.

The Application Defaults tab contains a realm setting.

When a user tries to access the application, he or she will be prompted to log in to the realm. A realm is the domain over which Single Sign On (SSO) is applied. For more information, see "[Securing applications.](#)"

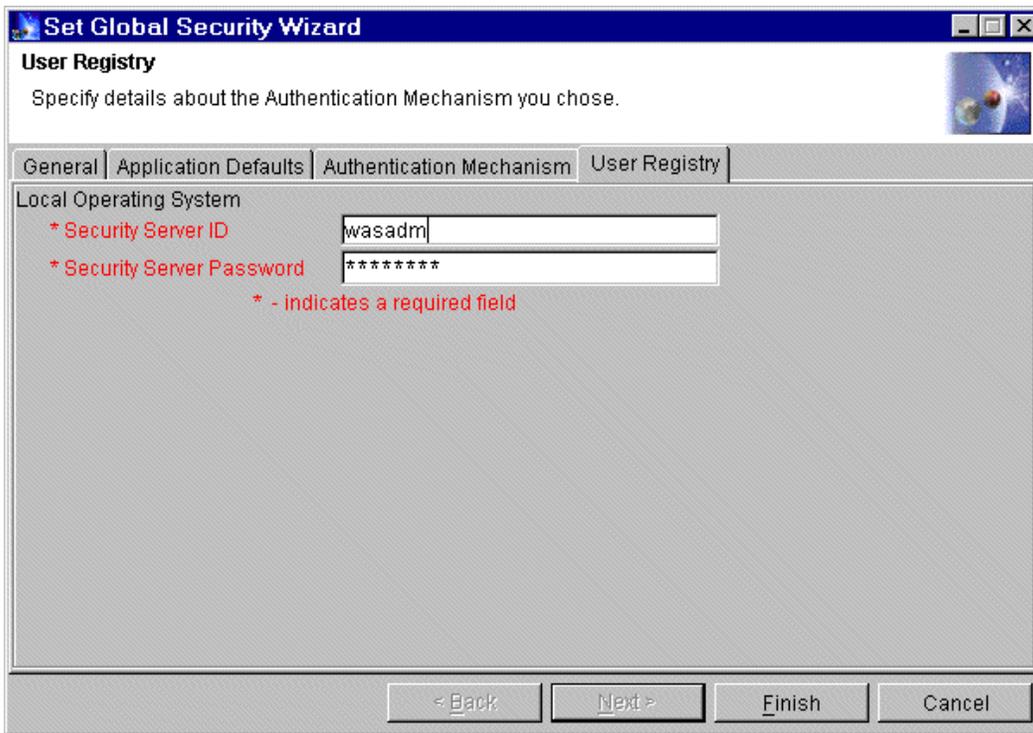
On the Application Defaults tabbed page, you can specify how users will be challenged for security credentials. For example, they can be prompted for a user ID and password, or a digital certificate instead.

For the Account application, specify Basic authentication.

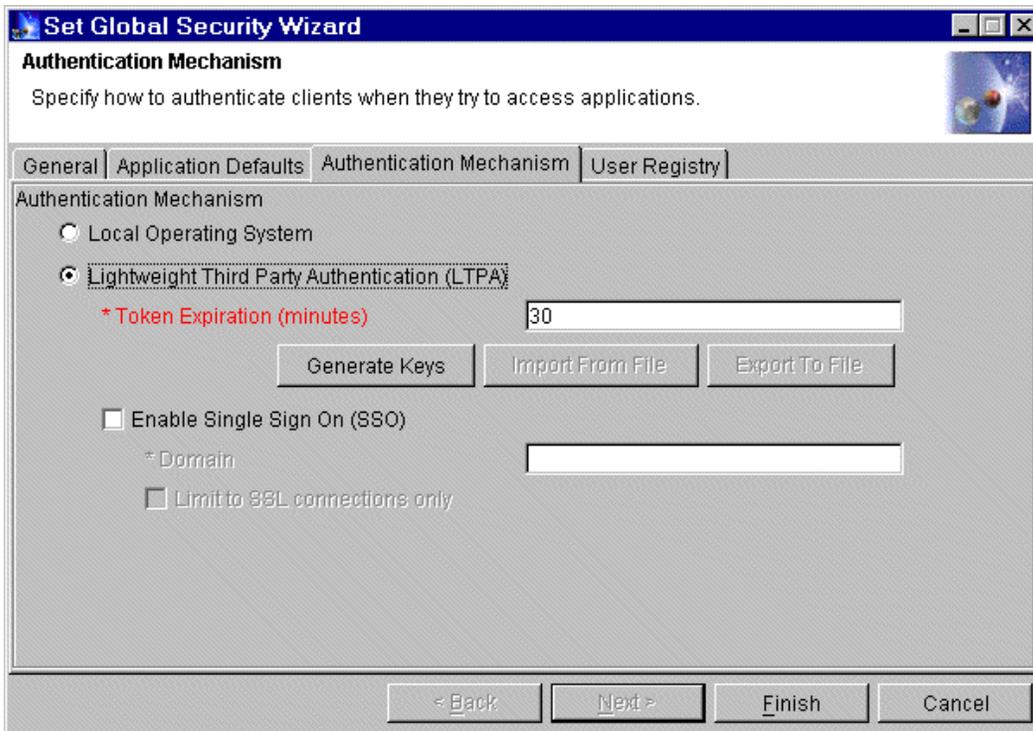
Click the Authentication Mechanism tab to specify how to authenticate the information presented by users trying to access an application or resources:

For the Account application, specify that users will be authenticated against the users defined in the user registry of the local operating system.

Now click the User Registry tab. Specify the security ID and password under which the WebSphere Application Server security server will run:



The User Registry tab content is dynamic, depending on whether you selected Local Operating System or an LDAP directory service supporting Lightweight Third Party Authentication as the Authentication Mechanism. To observe this, return to the Authentication Mechanism tab and click the LTPA option:



Now click the User Registry tab. You will see a different set of properties than before. With these LDAP properties you can configure authentication with an LDAP-enabled directory service product instead of with the local operating system:

Set Global Security Wizard

User Registry

Specify details about the Authentication Mechanism you chose.

General | Application Defaults | Authentication Mechanism | **User Registry**

LDAP

* Security Server ID

* Security Server Password

* Directory Type: Netscape

* Host

Port

Base Distinguished Name

Bind Distinguished Name

Bind Password

Use SSL to connect to directory

* - Indicates a required field

< Back | Next > | Finish | Cancel

Return to the Authentication Mechanism tab and re-select Local Operating System. Click Finish to complete the global security settings task.

Start the administrative server again to pick up the changes

The settings on the General, Authentication Mechanism, and User Registry tabbed pages, such as "Enable security," do not take effect until you stop the administrative server and start it again. At this time, switch to the Topology view. Right-click the node for the local machine and click Stop for restart.

This will cause (after the confirmation dialog) the console and administrative server to shut down. Start the server and console again to resume the rest of the steps in the tutorial. You can continue the tutorial without doing this, but you will not be able to verify that your application has been secured properly. Also, you might encounter errors during the other security tasks.

- [Browse security settings for this task wizard](#)

Secure the applications

Now configure an individual application, the Account application. Press the Wizards button and select Configure Application Security.

- [Read about this task: key concepts, settings, when and why to perform it](#)

On the first panel, select to apply application security to the ExpiringPageApp you configured earlier:

Configure Application Security Wizard

Enterprise Applications

Select an enterprise application from the folder.

Enterprise Applications

- AdminApplication
- ExpiringPageApp

< Back | Next > | Finish | Cancel

Click Next to proceed. The next panel lets you override the default settings you specified in the "Specify Global Settings" task. Notice you do not have the option to change the truly global security settings you specified in the earlier task. They are shared by this application and all others:

The screenshot shows a Windows-style dialog box titled "Configure Application Security Wizard". The main heading is "Realm and Challenge Type" with a sub-heading "Add the application to a security realm. Customize the challenge type." Below this, there are several input fields and radio buttons. The "Realm" section has a red asterisk and the label "* Name" next to a text box containing "Default". The "Challenge Type" section has four radio buttons: "None", "Basic (User ID and Password)" (which is selected), "Certificate", and "Custom". Under "Certificate", there is a checkbox labeled "Default To Basic". Under "Custom", there are two text boxes labeled "* Login URL" and "* ReLogin URL". At the bottom of the form area, there is a checkbox labeled "Use SSL to connect client and Web server" and a red asterisk with the text "* - indicates a required field". At the bottom of the dialog, there are four buttons: "< Back", "Next >", "Finish", and "Cancel".

Click Finish to complete the task.

- [Browse security settings](#)

Create custom method groups

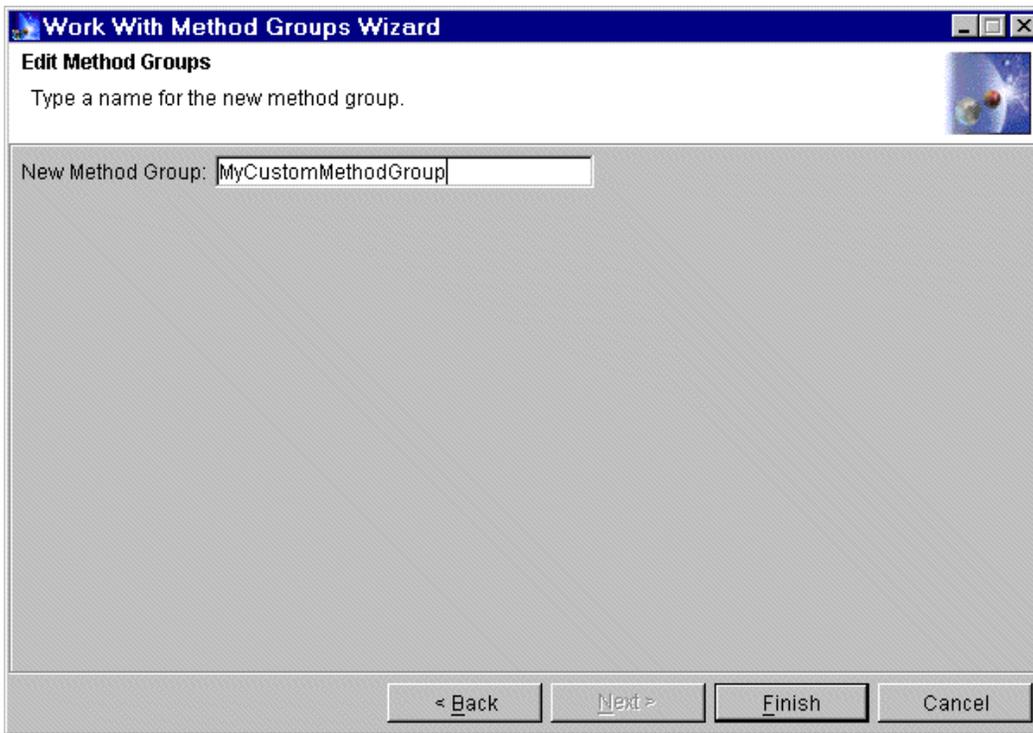
Use this task to create custom method groups for protecting the methods of individual resources. You only need custom method groups if you are not satisfied with the default method groups.

- [Read about this task: key concepts, settings, when and why to perform it](#)

Go ahead and create a custom method group. Press the Wizards button and select Configure Security Method Groups. Select Add a new method group:

The screenshot shows a Windows-style dialog box titled "Work With Method Groups Wizard". The main heading is "Add or Remove" with a sub-heading "Want to add a new method group, or delete an existing group?". Below this, there are two radio buttons: "Add a new method group" (which is selected) and "Remove existing method groups". At the bottom of the dialog, there are four buttons: "< Back", "Next >", "Finish", and "Cancel".

Click Next, and type MyCustomMethodGroup in the New Method Group field and click Finish:



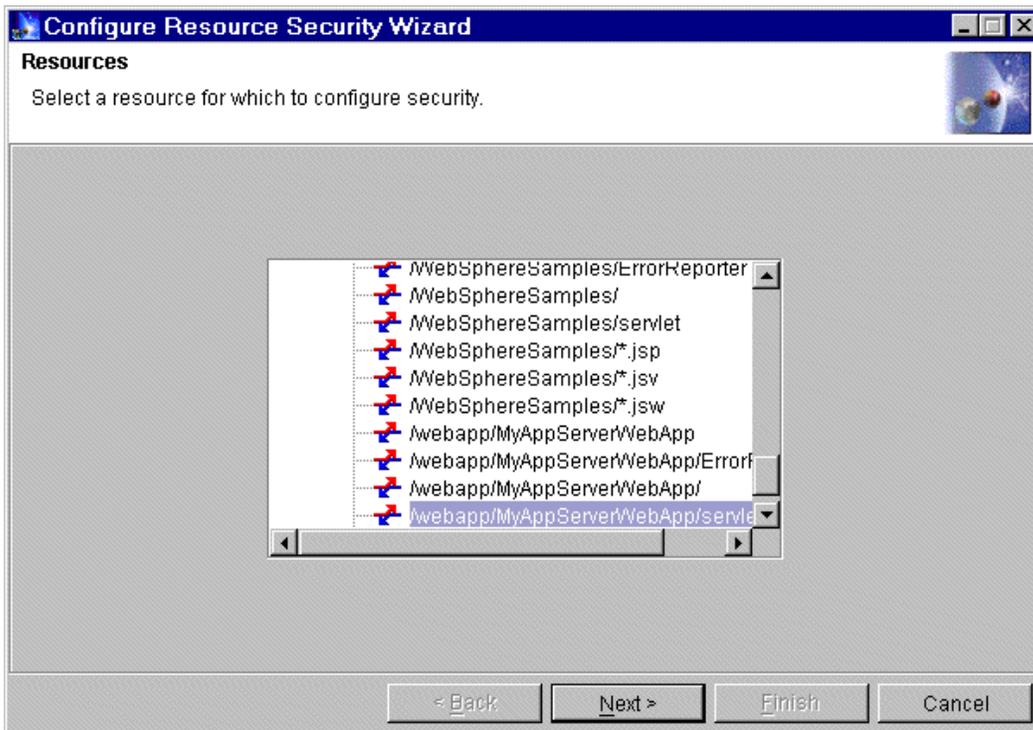
- [Browse security settings](#)

Secure the JSP files, Web pages, and servlets

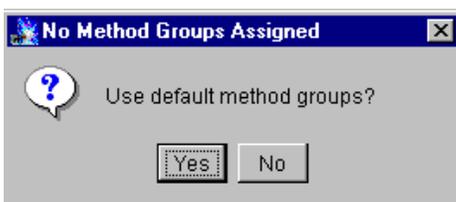
In addition to applying security at the application level, configure security for the HTTP methods of each resource.

- [Read about this task: key concepts, settings, when and why to perform it](#)

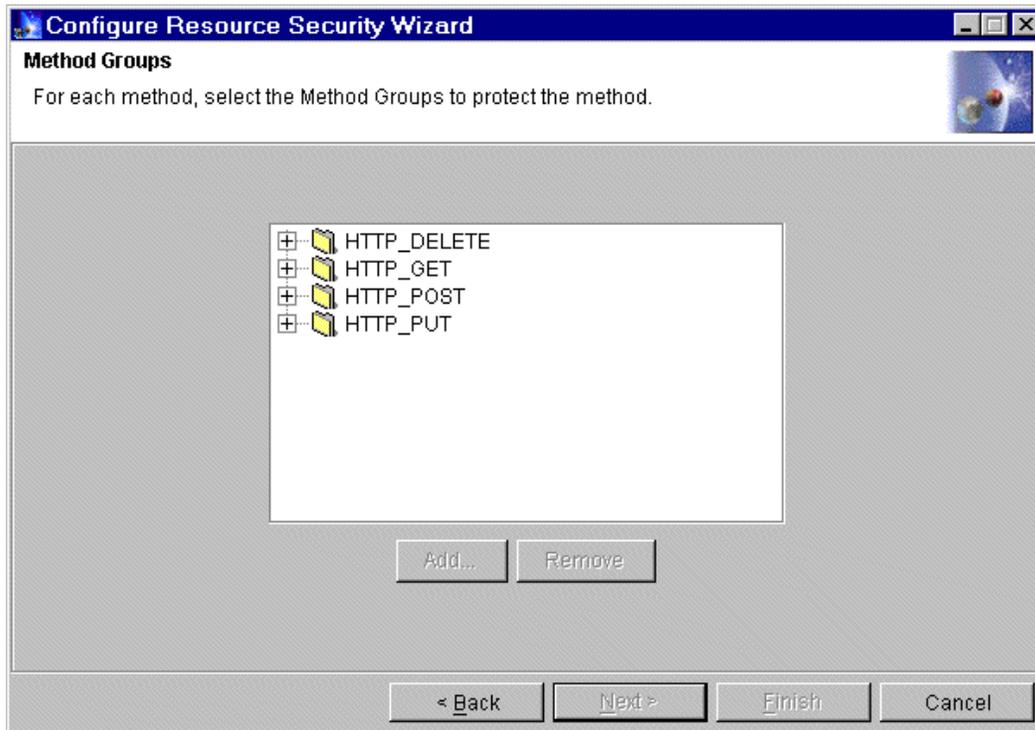
Press the Wizards button and select Configure Resource Security. On the first panel, specify to apply resource security to the servlet directory of the Web application:



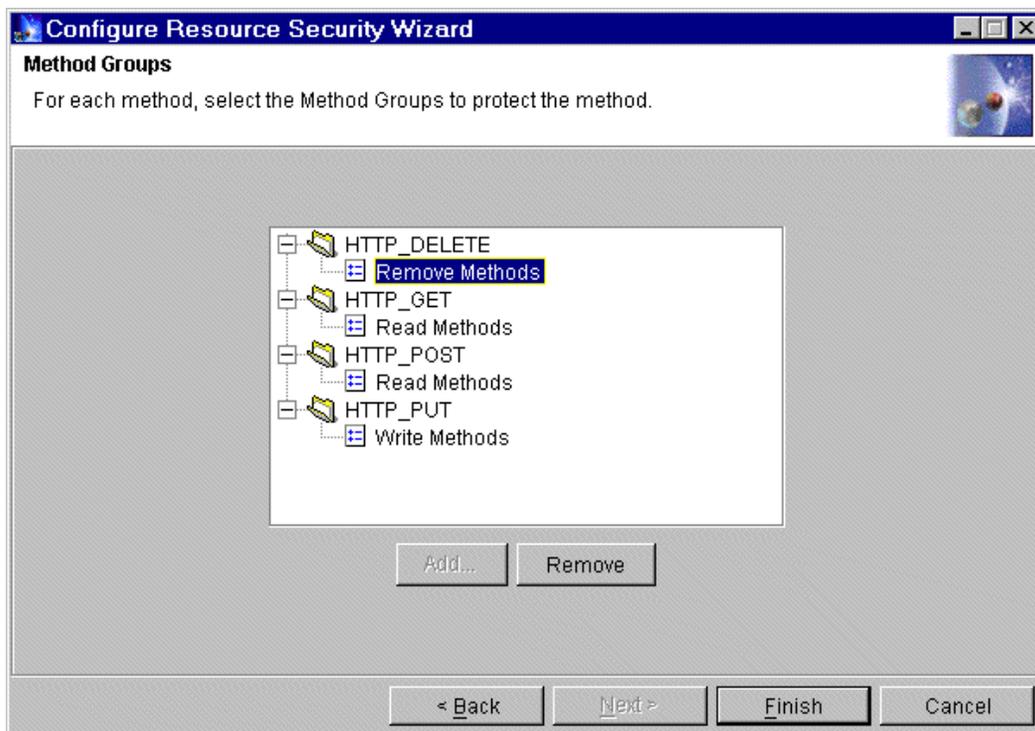
You might have to scroll down to locate it. Click Next to proceed. A prompt asks whether to use the default method groups. Specify Yes:



The next panel shows how WebSphere Application Server has grouped the servlet HTTP methods into method groups for protection. The folders represent types of methods encountered in HTTP:

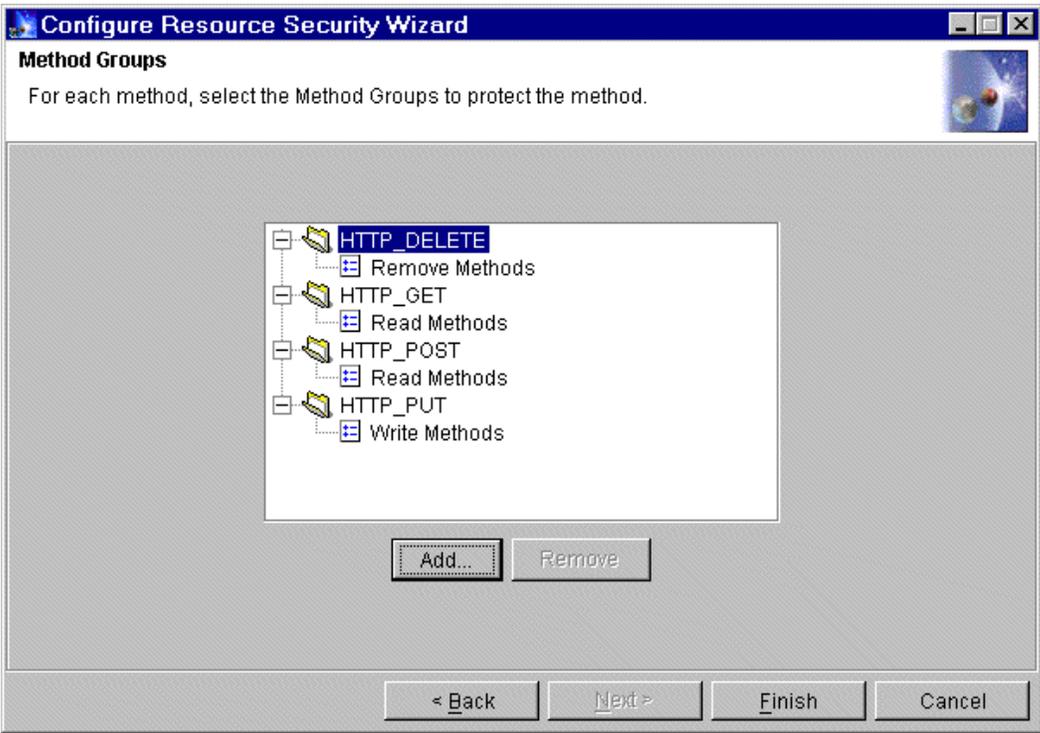


Expand the folders to see which method groups are associated with the methods. The method groups will protect the methods:

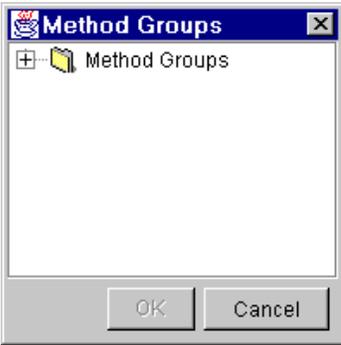


For example, the HTTP_DELETE method has been put in the Remove Methods group. Using the final security task, you will specify who is allowed to access the Remove Methods group.

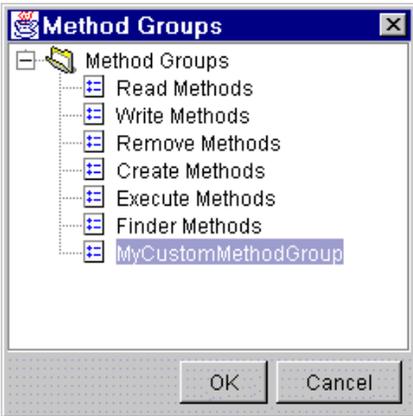
Suppose you also want your *custom* method group to protect the HTTP_DELETE method. Select the method and click the Add button:



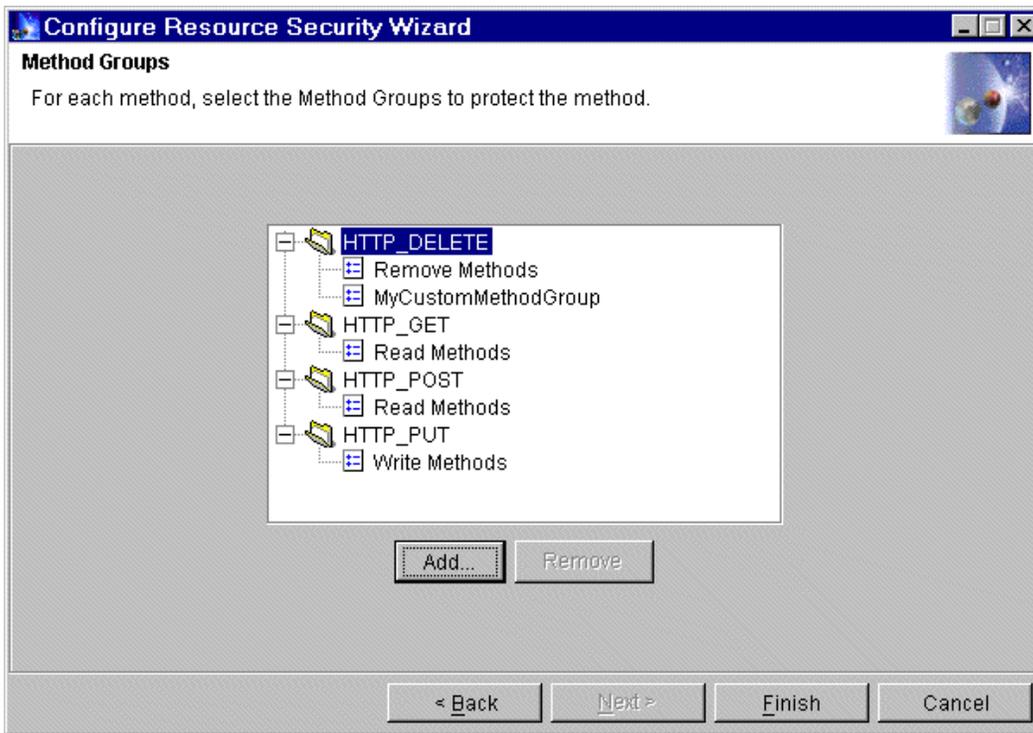
A dialog box is displayed for selecting a second method group to protect HTTP_DELETE methods:



Specify MyCustomMethodGroup and click OK:

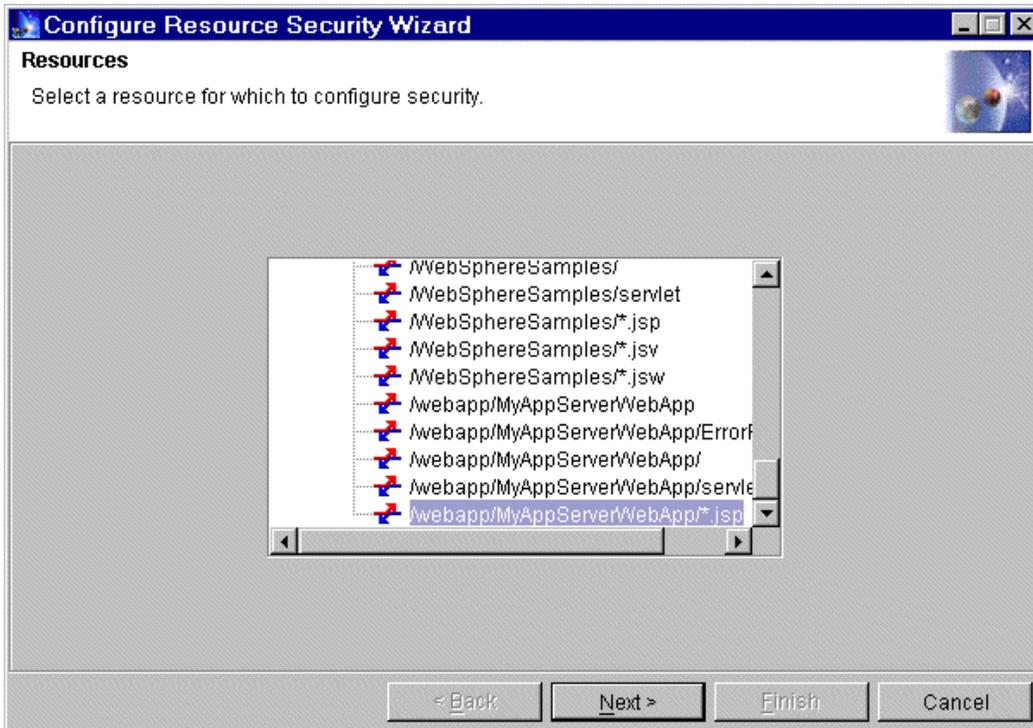


Now MyCustomMethodGroup is displayed below the HTTP_DELETE method, indicating it is protecting that method of the servlets you drop in the Web application's servlets directory:



Click Finish. Start the Configure Resource Security task over again, this time specifying to configure security for all JSP files in the MyAppServerWebApp Web application.

Hint: When you reach the page for specifying the Web resource to protect, select:



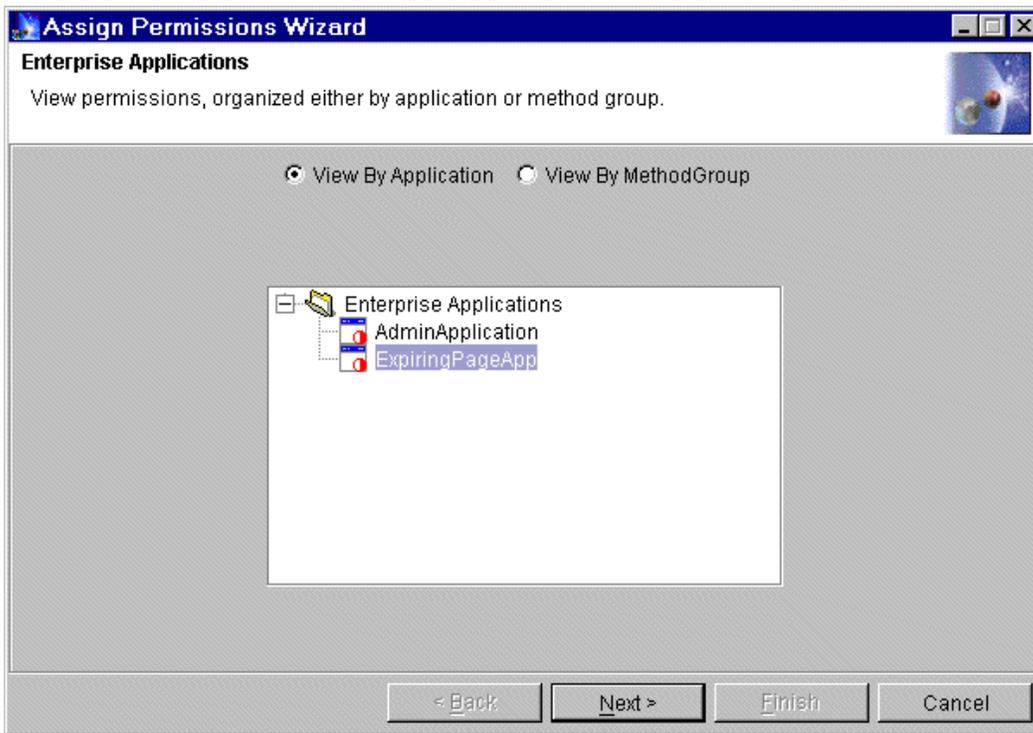
- [Browse security settings](#)

Grant security permissions

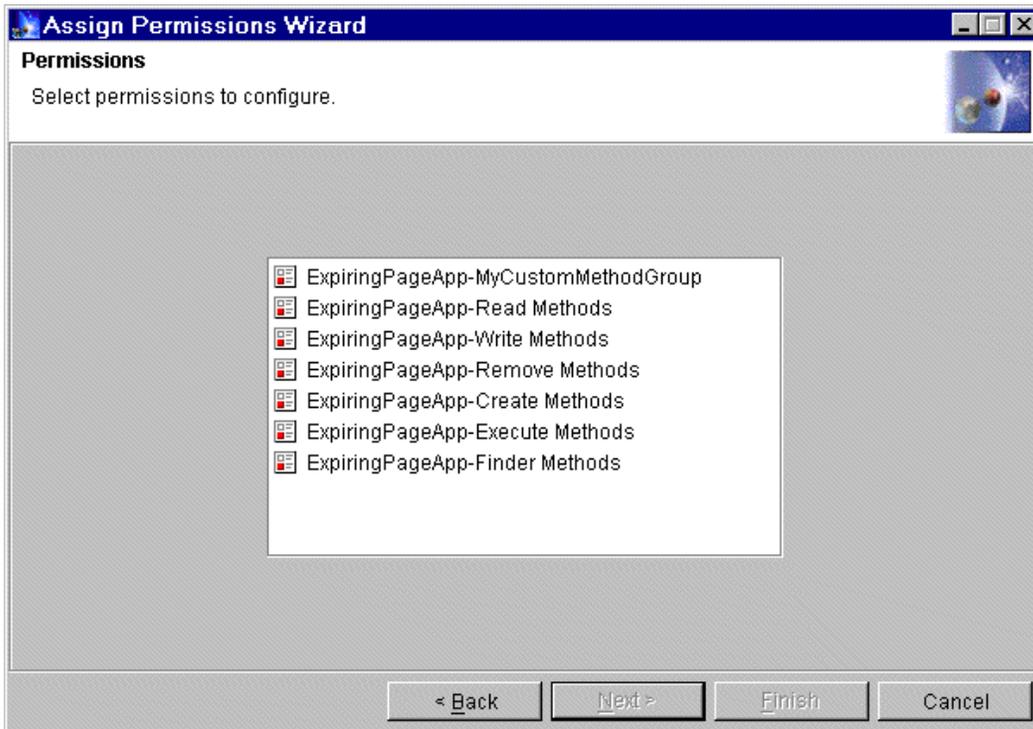
Now give users and groups permission to access the resources within your applications.

- [Read about this task: key concepts, settings, when and why to perform it](#)

Press the Wizards button and select Configure Security Permissions. You may view permissions by application or by method group. Select View By Application, expand the Enterprise Applications folder and select ExpiringPageApp:

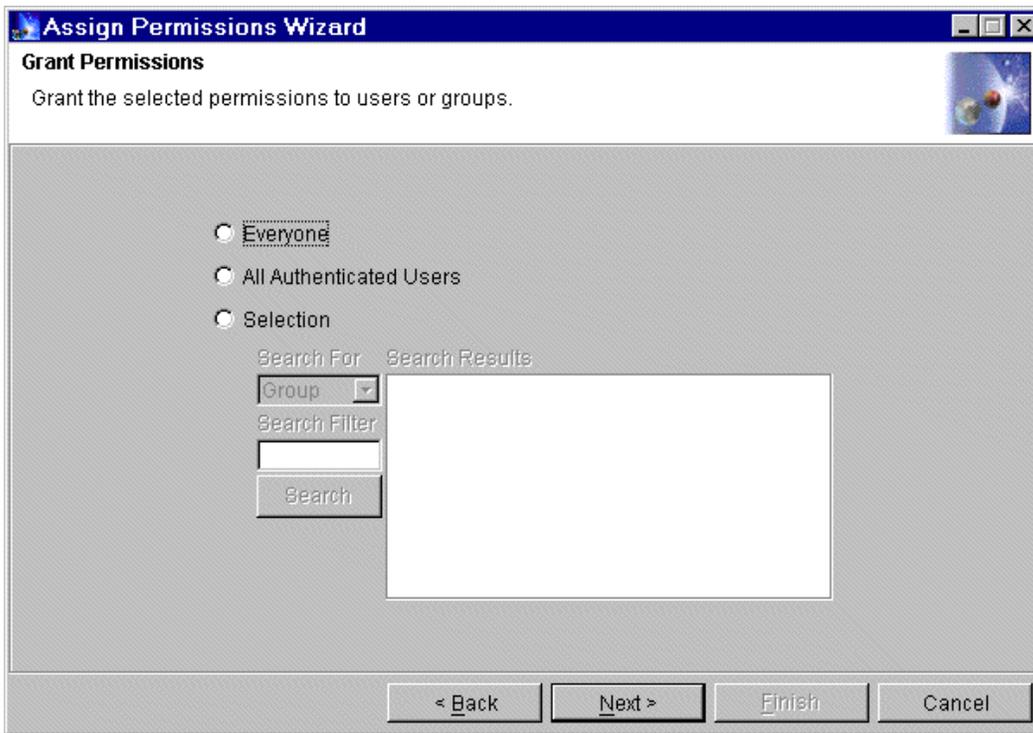


Click Next to display this screen, which shows the *permissions* you can possibly grant to users and groups:

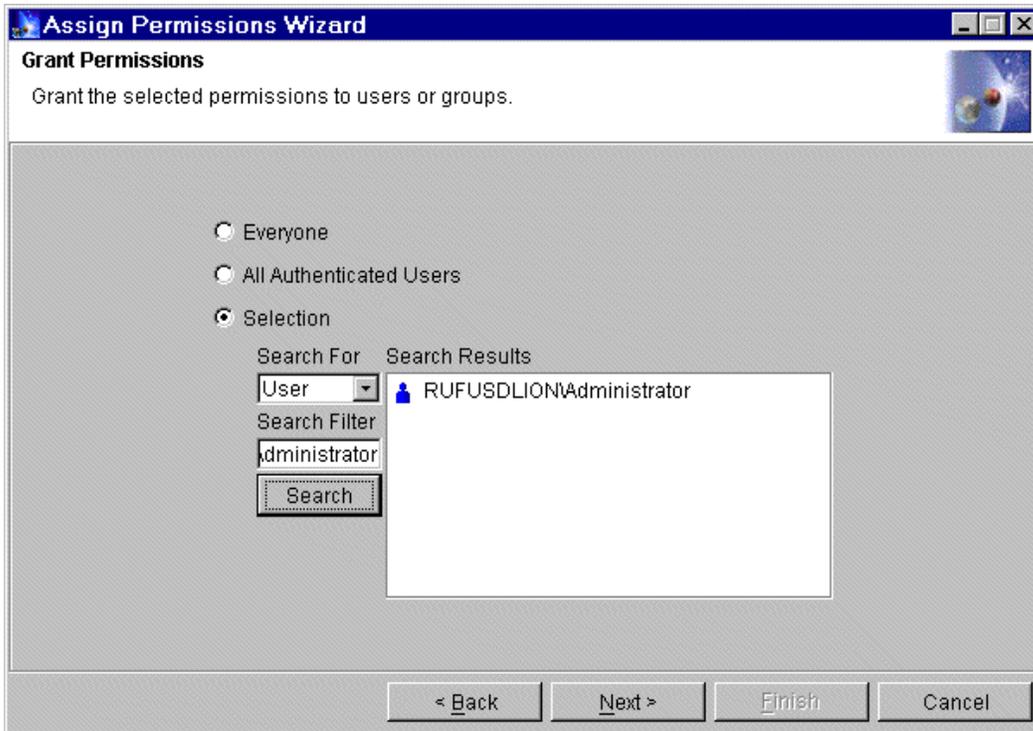


Before granting the ExpiringPageApp permissions, try the following exercise.

Click the AdminApplication-WriteMethods permission. Click the Next button to display a search dialog for specifying the users or groups to whom you want to grant permission to access this method group:

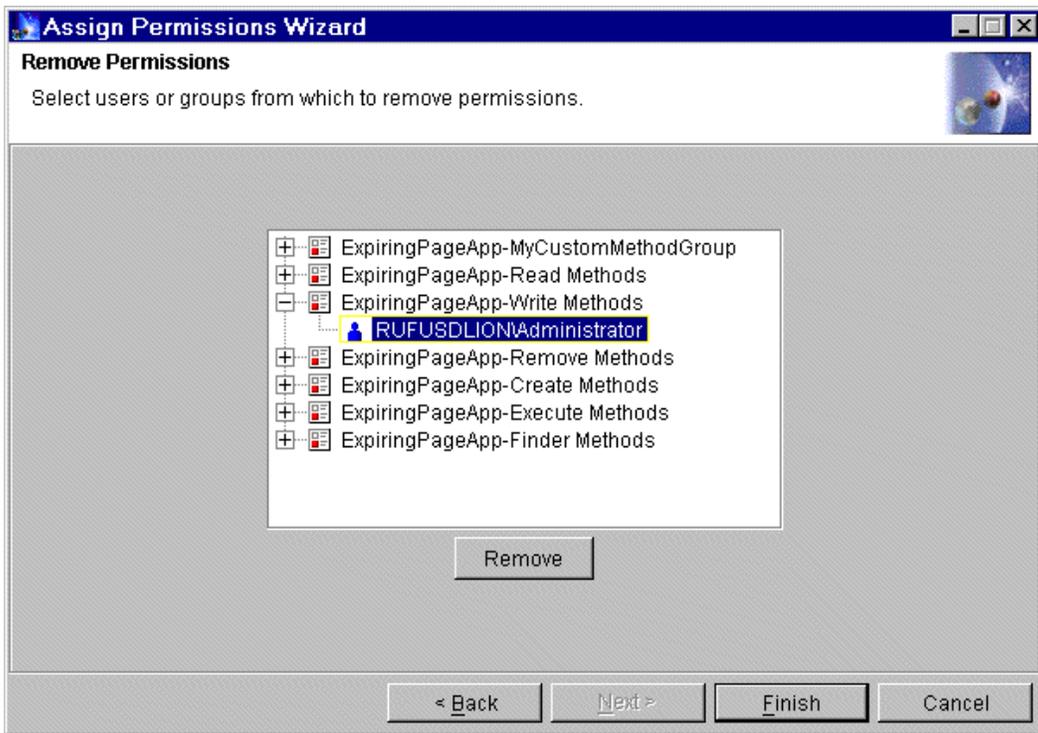


You can select to grant Everyone access to those methods. Instead, limit the access to the local operating system administrator. Click Selection to search for that identity. Specify to search for a user, and type "Administrator" in the Search Filter field and click Search. The Administrator is displayed in the Search Results box. Click the Administrator, then click Next:

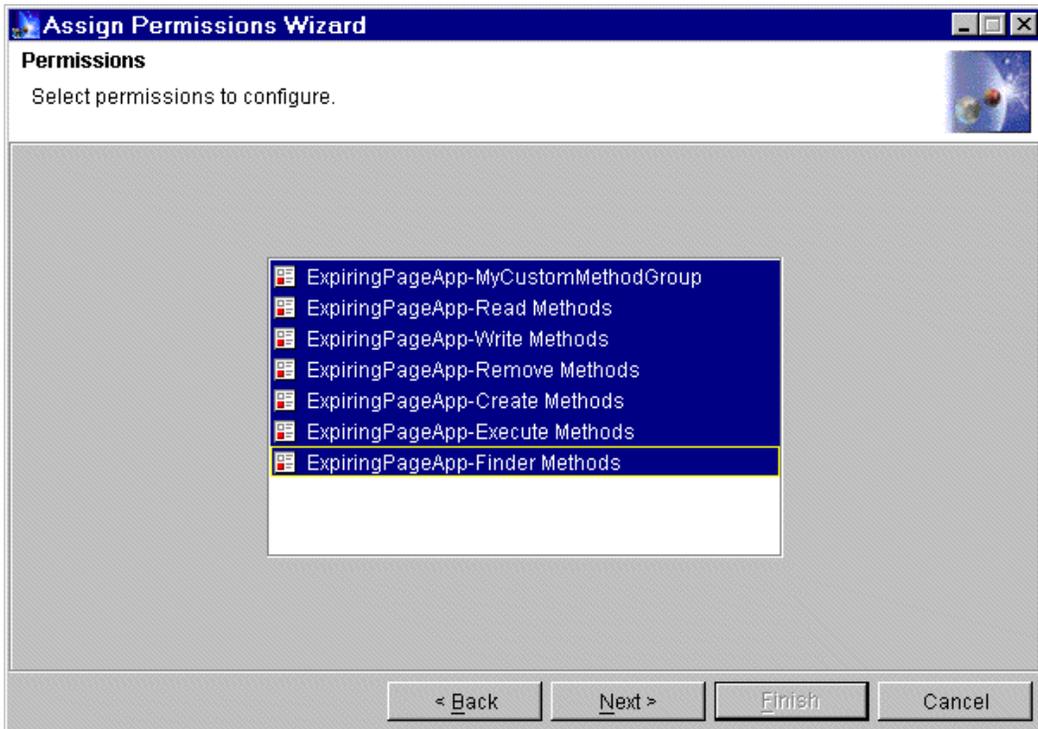


I Searching for the Administrator is likely to work only on Windows NT machines. If using UNIX, try searching for the "Staff" group, or another user or group ID that you are sure exists on the machine.

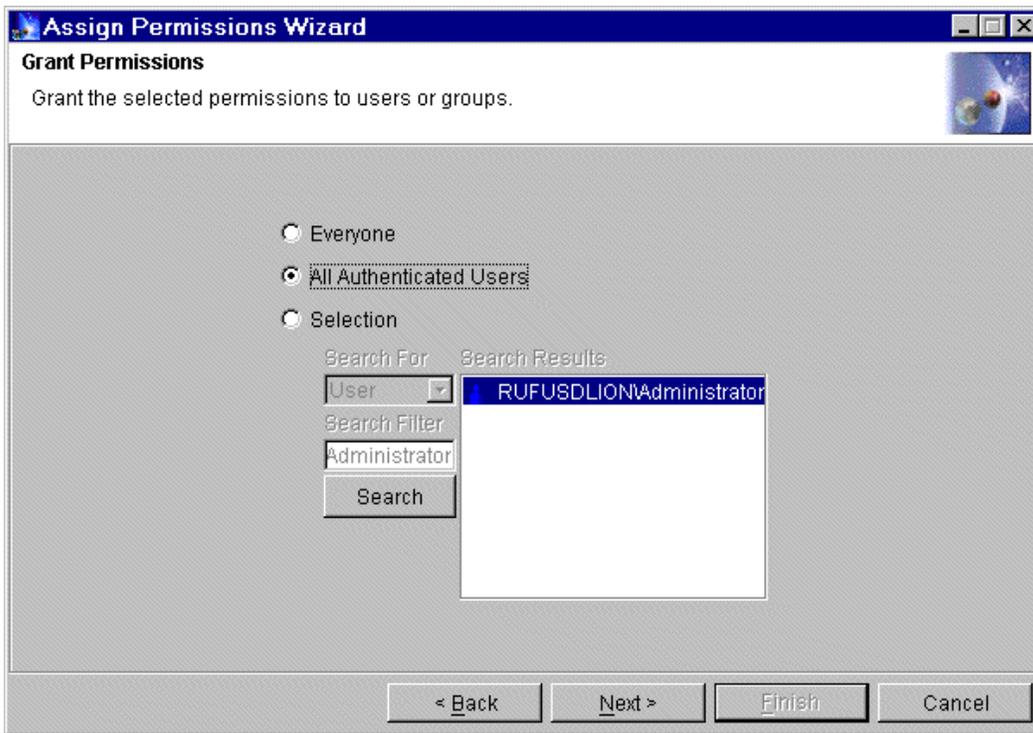
Now you see that the administrator has been granted permission to access read-only methods in the Account application:



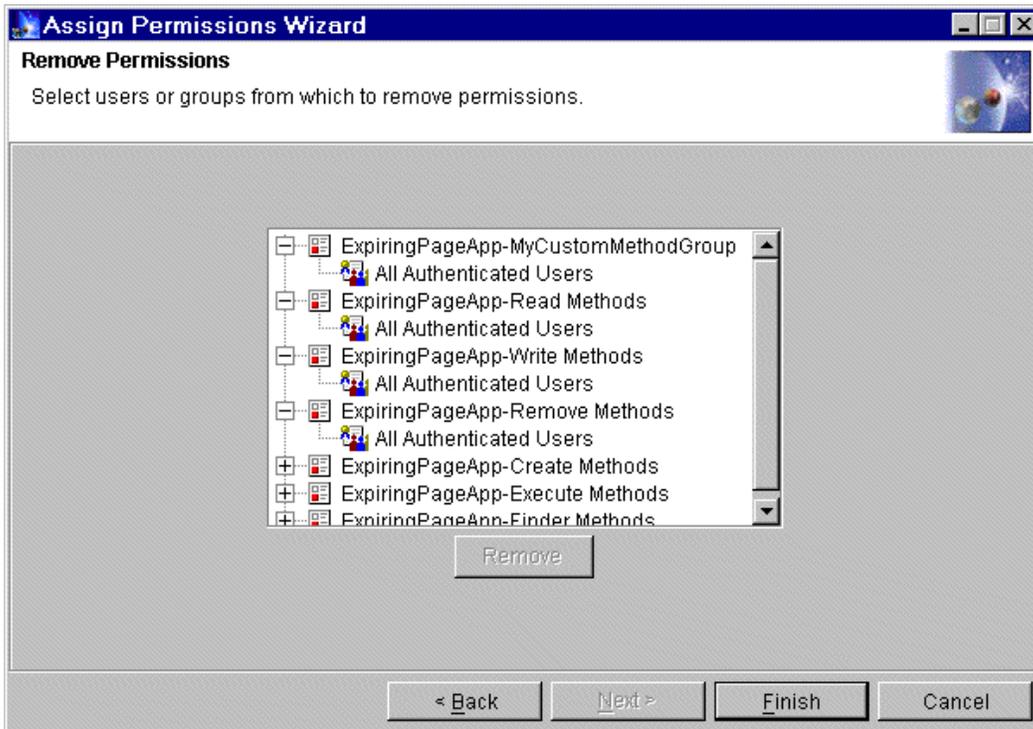
Now specify permissions for the ExpiringPageApp. Press Back until you see the Permissions panel, then highlight all of the ExpiringPageApp permissions and click Next:



Click Add:



Now change the search dialog box to "All Authenticated Users." Click OK. Verify that each of the ExpiringPageApp permissions has been granted to All Authenticated Users.



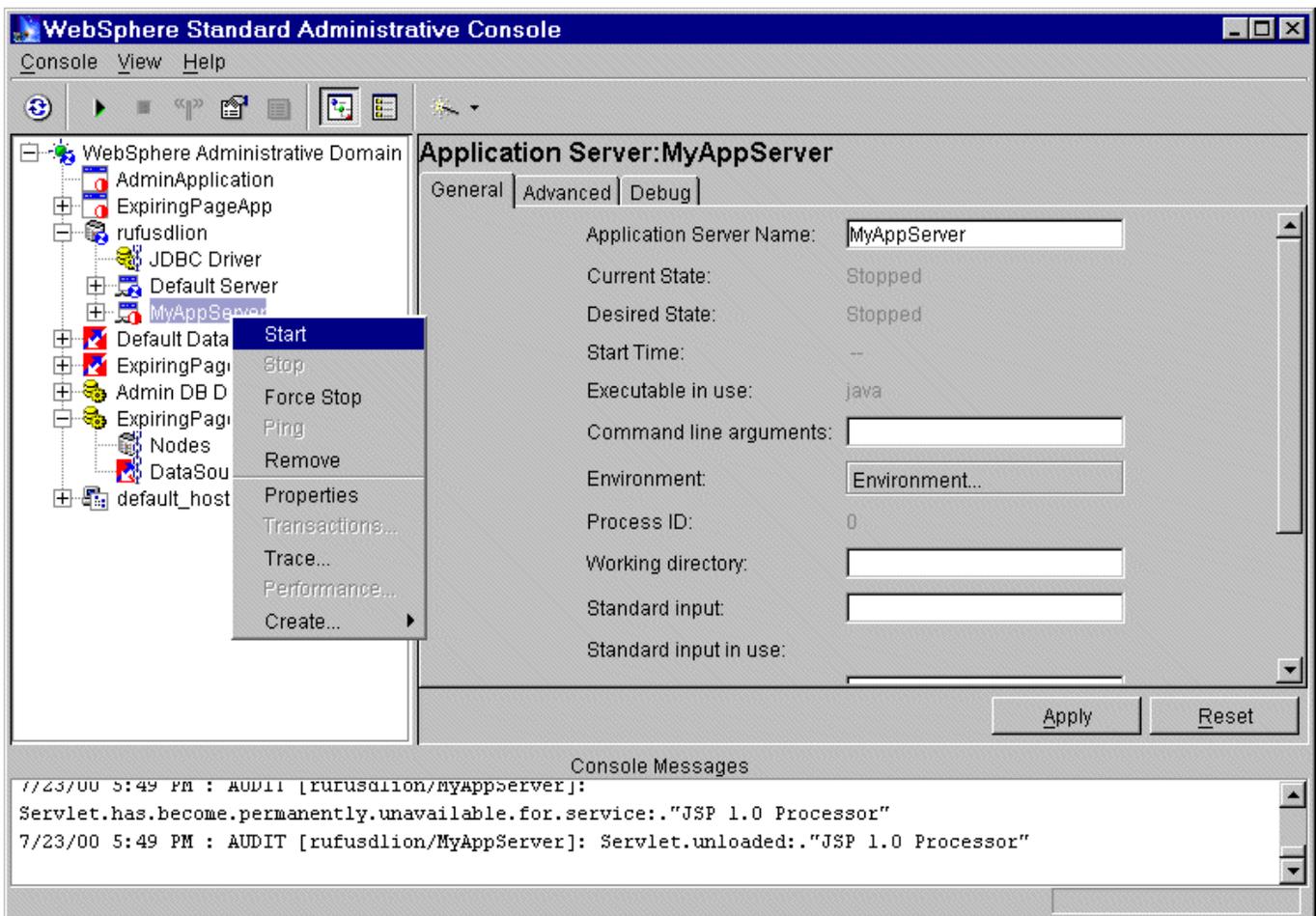
- [Browse security settings](#)

Step 5: Verify the application configuration and security

This section describes how to verify that the ExpiringPageApp can be accessed successfully, given its configuration and the security you applied.

Remember that this step is unlikely to be successful if you did not stop the administrative server and start it again after specifying the global security settings.

Now proceed with verification.

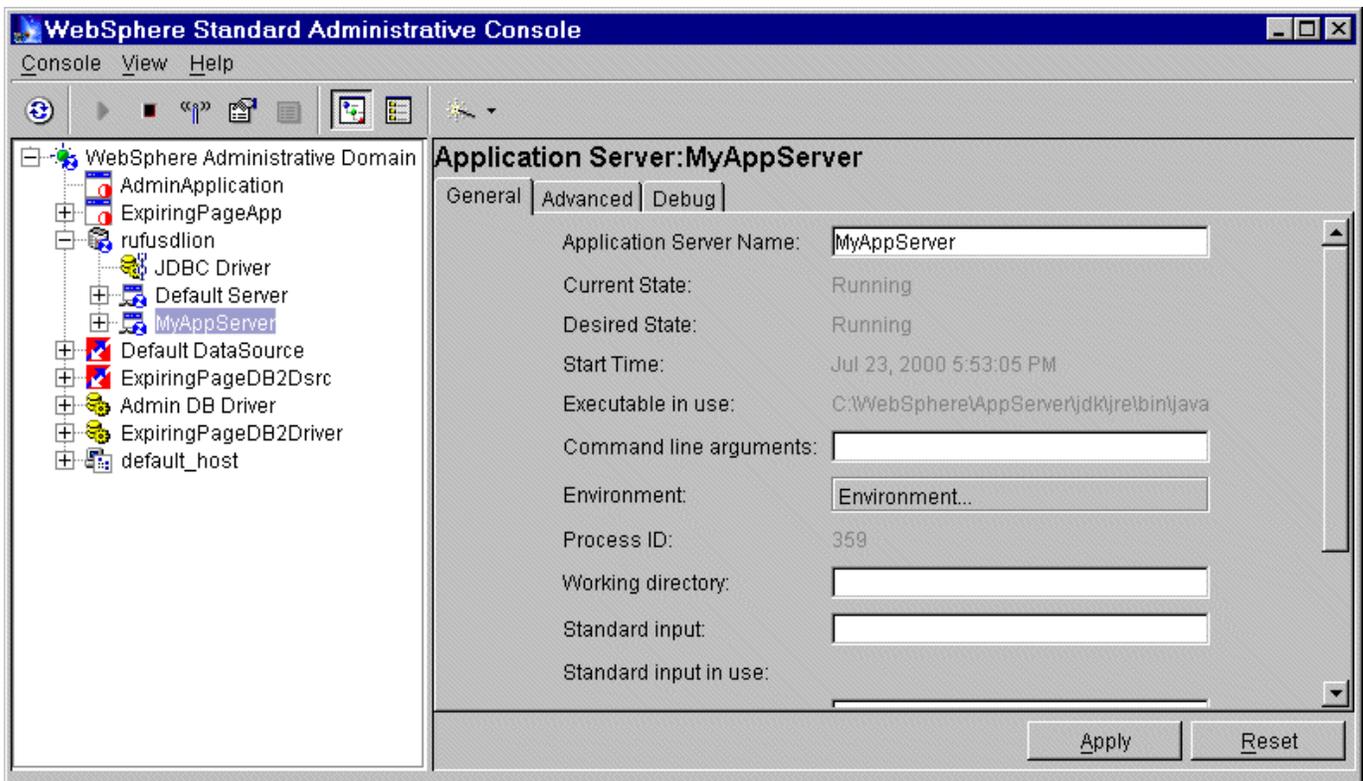


First, make sure the application server is aware of the new Web application.

1. Click the Topology button to display the Topology view.
2. Expand the tree until you locate MyAppServer.
3. Click MyAppServer so that it is selected. If the server is currently stopped, click the start button on the console toolbar to start the server. It might take a couple of minutes to start. Messages in the Console Messages area of the WebSphere Administrative Console help you monitor its progress.

If the server was already started, first stop it (using the stop button on the console toolbar), then start it again.

Now the server and its contents are started:



Console Messages

```
7/23/00 5:53 PM : AUDIT [rufusdlion/myAppServer]: Loading.servlet:."JSP 1.0 Processor"  
7/23/00 5:53 PM : AUDIT [rufusdlion/MyAppServer]: [Servlet.LOG]:."com.sun.jsp.runtime.JspServlet: init"  
7/23/00 5:53 PM : AUDIT [rufusdlion/MyAppServer]: Servlet.available.for.service:."JSP 1.0 Processor"
```

Next, use a browser to test the Web application. Based on the Web application configuration, here is how to assemble the URL for accessing the Web application:

- Because the Web application uses the default host, the first part of the URL can be any valid alias for the default host. Remember, the default_host "Aliases" property lists the valid alias for the host. It can be viewed by clicking the default_host in the Topology tree and clicking the Advanced tab.
- Because the Web path of the Web application is MyAppServerWebApp/ExpHTMLServlet, that is the second part of the URL.

Therefore, valid URLs include:

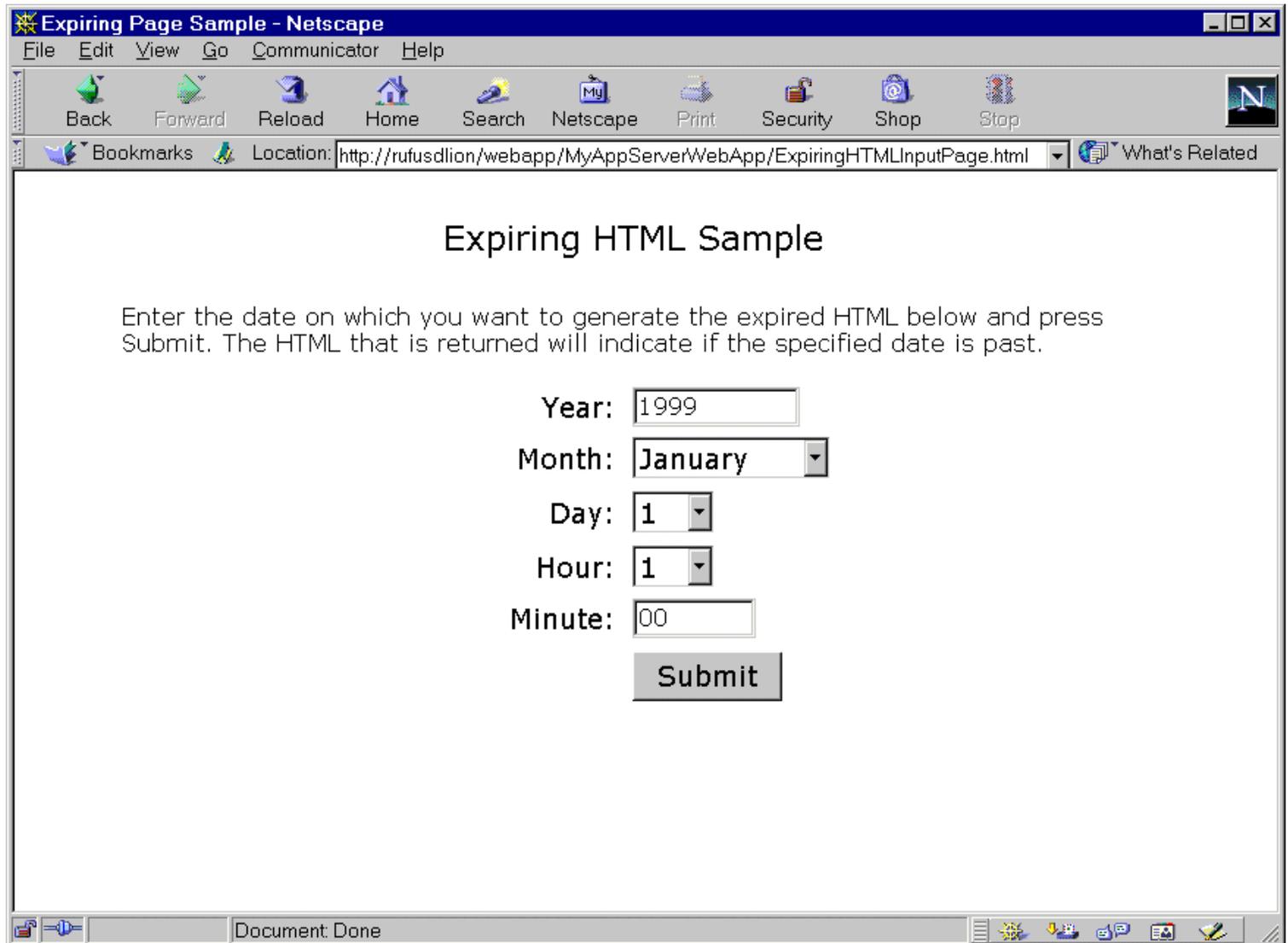
`http://hostname/webapp/ExpHTMLServlet`

and

`http://IP address/webapp/ExpHTMLServlet`

where *hostname* and *IP address* are valid default_host aliases as discussed above.

Try one or more of these URLs in a browser to ensure they work. When the ExpiringHTMLInputPage is displayed, try using it to ensure you can access the servlet and JSP functionality. If so, you have successfully configured and applied security to your Web application.



Troubleshooting

If you receive an error:

- Ensure the Web server is running.
- Ensure the application server and all of its contents are running. In particular, ensure the Web application's system servlets are running in order to support the HTML or JSP file you are trying to access.
- Check the URL. For example:
 - Are you using correct values for the virtual host, Full Web path (for Web application), and individual resource name?
You can click resources in the Topology tree to view their properties and check these values.
 - If you are accessing an explicitly configured servlet, does the Web path you are using match the Web paths defined in the servlet's configuration.

If the Web path is missing, you can add it, then restart the servlet's application server for the change to take effect.

If the Web path is configured in the servlet's Web path list, but is not in the servlet's "Web paths in use" list, stop the server and start it again to put the Web path in use.

This concludes the administrative console tutorial. You have configured an application, the resources it contains, and all the necessary supporting resources, such as an application server. You have also tested the application configuration and security.

Be sure to explore other areas of WebSphere Application Server Version 3 functionality, including personalization (sessions and user profiles), resource analysis, and Object Level Trace (OLT) and debugging, to name just a few.

Click **Help -> Contents** on the console menu bar to access an HTML frame set providing navigation among the help files. Click **Help -> Documentation** to view the Documentation Center, which contains the product documentation, programming information, and links to external resources on the Web.

6a: Administrative overview of Version 3.5

- [Administrative model](#)
- [Relationships among resources \(the topology tree\)](#)

Looking for a summary of the tasks and resources you can manage with IBM WebSphere Application Server Version 3.5? See [the quickreference](#), including links to instructions and descriptions of resource settings.

Administrative model

IBM WebSphere Application Server provides administrators with a single system view of applications that are typically deployed across multiple machines. An administrator working on a physical machine can remotely administer resources located on another physical machine.

These elements comprise the WebSphere administrative model:

Element	Role
administrative server	Resides on a node (machine) for administering resources on the node
administrative database	Stores administrative data for an administrative server
administrative domain or topology	An administrative server and database
resources	Files and servers in the administrative domain. The administrative database holds representations of them to mirror the actual resources residing on the nodes in the administrative domain
administrative clients	Enable administrators to access the administrative server on nodes in the administrative domain. The graphical clients (Java and Web browser clients) provide a view of the domain topology

Relationships among resources (the topology tree)

The Topology tree demonstrates the relationships among resources. The relationship is the "containment hierarchy." The Topology tree is visible in the Java console (WebSphere Administrative Console).

The containment hierarchy imposes a structure on the topology of the administrative domain. Becoming familiar with the hierarchy will make administrative tasks seem easier to accomplish.

Anyone who has worked with directory and file representation systems, such as Microsoft Windows Explorer, has already been introduced to the concept of containment. Containment defines a relationship in which one resource is a part or "child" of another resource.

If you have ever tried creating a directory in a file, you know that your file management program will not allow it. The directory-file containment hierarchy imposes the following rule:

Directories can contain files, but files cannot contain directories.

Similar relationships and rules exist among resources in the WebSphere Application Server administrative domain. For example, an application server resource can contain just one servlet engine.

Some relationships depend on each other. Adding a servlet to the administrative domain requires a Web application. A servlet engine must exist to contain the Web application. Finally, an application server must exist to support the Web application containing the servlet.

The containment hierarchy enables:

- Efficient operations

A "parent" resource and its "children" can be treated as a unit. For example, starting a servlet on an application server automatically starts the Web application, servlet engine and application server supporting the servlet.

- Relative naming

Containment represents a hierarchical naming structure, much like a file directory structure. Just as a file name must be unique within a directory, a resource name must be unique within its container.

For example, the servlet named `my_servlet` in `Web_application_A` is considered different from the servlet named `my_servlet` in `Web_application_B`, even if both use the same servlet class.

A resource position in the containment hierarchy is used to generate the full, unique name of the resource in the administrative database.