

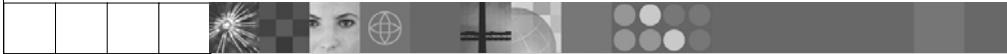


IBM Software Group

## Java Tools WebSphere Development Studio Client V5.0

Don Yantzi (yantzi@ca.ibm.com)  
IBM Toronto Lab

**WebSphere.** software



July 2003 | Java Tools

© 2003 IBM Corporation

This presentation looks at the Java Development Tools (JDT) included in WebSphere Site Studio Site Developer Advanced and the iSeries extensions added to these tools in WebSphere Development Studio Client for iSeries. To get a high level overview of the tools and Workbench we first create a simple HelloWorld Java application. Then we cover the Java editor, its accompanying views and the rest of the compile / run / debug development cycle. The presentation finishes off with a look at the iSeries extensions to the Java development tools.



## Agenda

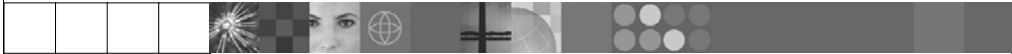
- Getting Started with Java Tools
- The Java Editor
- Java Views
- Visual Editor
- Compile / Run / Debug
- iSeries Additions
- Reference



IBM Software Group

# Getting Started with Java Tools

**WebSphere.** software



July 2003 | Java Tools

© 2003 IBM Corporation

## Java Perspective

- **Customizes Workbench for Java development**
- **Java perspective contains Java related views**
  - Packages view
  - Hierarchy view
  - Outline view (standard workbench view)
  - Tasks view (standard workbench view)
- **See also**
  - Java Browsing perspective
  - Java Type Hierarchy perspective
  - Build your own!
    - Customize, customize, customize ...

Perspectives are used in the Workbench to provide coherent selection and layout of views related to a specific type of development (Java, XML, WebFacing, ...) By default the Java perspective shows the packages, outline, hierarchy and tasks views. The Outline view is a standard Workbench view for showing the outline of the resource currently opened in the editor. It works for Java and other resource types like XML and SQL scripts. The Tasks view is another standard workbench view and shows all errors, warnings and tasks in the workspace not just Java errors. The packages and hierarchy views are Java specific. Each of the four views are outlined in more detail on the following slides.

### •Packages view

- Shows only the Java projects in your workspace
- Collapses package directories into a single folder in the tree view

### •Hierarchy view

- Inheritance hierarchy of the Java class

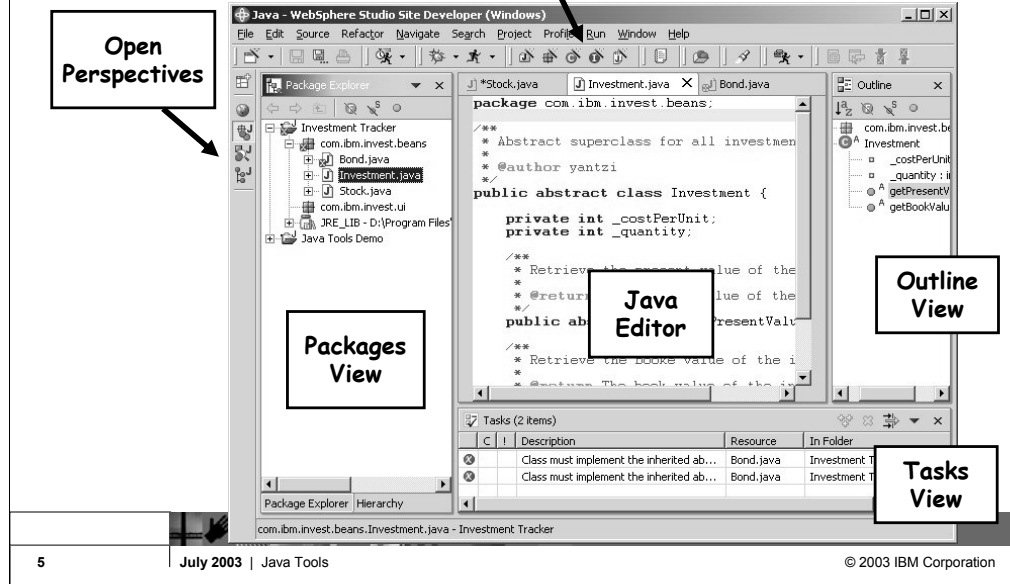
### •Outline view (standard workbench view)

- High level view overview of the structure of your Java class

### •Tasks view (standard workbench view)

- Shows compile errors and warnings

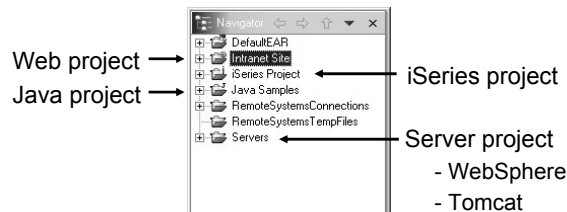
## Java Perspective



Here you see a screen shot of the Java perspective (default layout.) All open perspectives are shown in the left hand column of the workbench window. Simply click the perspectives icon to switch to the open perspective (this applies to all perspectives not just the Java perspective.) Each perspective typically contributes shortcuts to the workbench toolbar. The Java perspective adds some icons for creating a new Java project, package, class, interface and scrapbook page.

## Java Projects

- **Used for organizing Java code**
- **Properties**
- **Java builder**
  - Associated builders
    - Builders know how to convert source artifacts (files) to executables and validate resources in a project
    - When you save a .java file the builder automatically compiles it into a .class file



Projects provide the highest level of grouping resources (source files, graphics, executables, ...) in the workspace. Projects always have a type such as Java, WebFacing, Web. Those projects that are not linked to a specific type of development should use the "Simple" type. The project type specifies what properties are associated with the project. For example Java projects have a build path property where users can specify other projects and .jar files that this project requires. Projects also have an associated builder which knows how to build resources in the project (i.e. compile .java files into .class files.) By default projects map to a subdirectory of the workspace directory on the local file system with the same name as the project.

## New Java Class Wizard

Use the New Java Class wizard to add classes to your Java project

Folder and package names are prefilled from Workbench selection

Create this class as an inner class

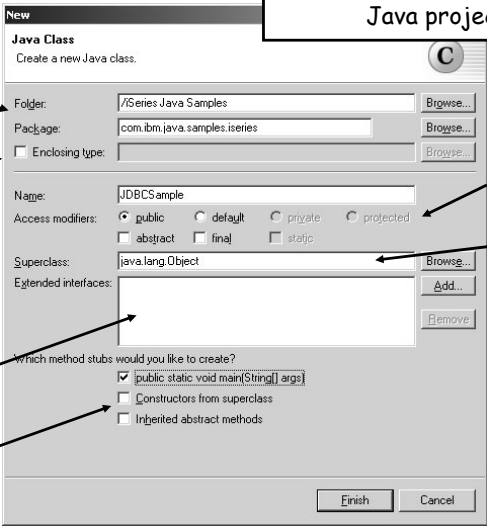
Enter a name for the class

Enter any interfaces you want your class to implement

The wizard can generate some common methods if you want them

Select any access modifiers for the class

Enter the name of a superclass or click Browse to select one from your workspace



7 | July 2003 | Java Tools | © 2003 IBM Corporation

Here you see the first page of the new Java Project wizard. This page prompts the user for the project name which is the only required piece of information for the project. You can also override the default workspace directory for this project, but this is not recommended. Subsequent pages of the wizard gather additional Java project properties which we will look at later in this presentation. Note: There are many different ways to launch the new Java Project wizard (or any of the new wizards). You can right click in the packages view and select New -> Java Project ... (as shown in the screen shot) or you can click the new Java Project icon in the toolbar or select File -> New -> Project...



IBM Software Group

# The Java Editor

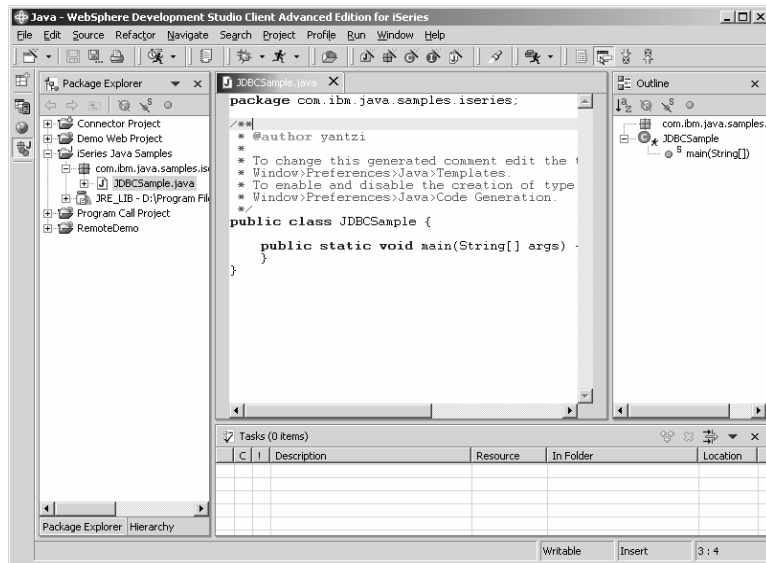
**WebSphere.** software



July 2003 | Java Tools

© 2003 IBM Corporation

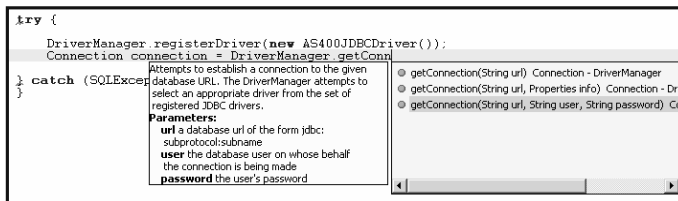




After completing the new Java Class wizard the .java file is created in the workspace and the file is opened in the Java editor.

## Content Assist – invoke by Ctrl + Space

- Suggesting class, method and field names
  - Select class / method / field from list of suggestions
  - JavaDoc comments shown for selection
- Overriding Methods
  - Use code assist to display list of methods in super classes
  - Select method from list to override in current class
- Adding import statements
  - Invoke code assist after entering class name to automatically import the class



```
try {
    DriverManager.registerDriver(new AS400JDBCDriver());
    Connection connection = DriverManager.getConnection( );
} catch (SQLException e) {
}
```

Attempts to establish a connection to the given database URL. The DriverManager attempts to select an appropriate driver from the set of registered JDBC drivers.

**Parameters:**  
**url** a database url of the form jdbc:subprotocol:subname  
**user** the database user on whose behalf the connection is being made  
**password** the user's password

- getConnection(String url) Connection - DriverManager
- getConnection(String url, Properties info) Connection - DriverManager
- getConnection(String url, String user, String password) Connection - DriverManager

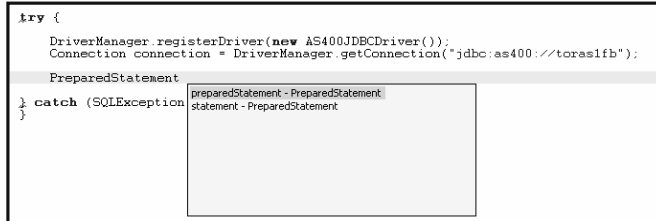
Content assist provides a list of class names, interfaces, fields and methods that are available in the current scope. To invoke content assist press Ctrl + space (or right click and select content assist from the popup menu). The list of options is subset by what you have already typed. For example: entering get and then pressing Ctrl + space will show only methods starting with get. If you keep typing then the list is further subset.

Tip: When using the workbench the user can double click on the title bar of any editor or view to maximize the editor / view to take up the entire workbench window.

## More Content Assist – Methods and Variable Names

- Displaying parameters for methods  
Invoke
- Suggesting variable names

```
try {  
    DriverManager.registerDriver(new AS400JDBCDriver());  
    Connection connection = DriverManager.getConnection("jdbc:as400://toras1fb");  
    PreparedStatement  
} catch (SQLException  
}
```

A screenshot of an IDE showing content assist for the variable 'PreparedStatement'. The code is: try { DriverManager.registerDriver(new AS400JDBCDriver()); Connection connection = DriverManager.getConnection("jdbc:as400://toras1fb"); PreparedStatement } catch (SQLException } The text 'PreparedStatement' is highlighted in the code. A tooltip box is open, showing 'PreparedStatement - PreparedStatement;' and 'statement - PreparedStatement'.

Content assist will display parameters for methods and suggest variable names.



# Content Assist - JavaDoc

Content assist for HTML inside JavaDoc comments

```

/**
 * Sample class to highlight the features of the
 * <b>Java Development Tools</ in WebSphere Development Studio Client V5.0.
 * This class reads an OS/400 physical file using JDBC and displays the results
 * to the console.
 *
 * @author yantzi
 */
public class JDBCExample {

    public static void main(

        try {
    
```

```

/**
 * Sample class to highlight the features of the
 * <b>Java Development Tools</ in WebSphere Development Studio Client V5.0.
 * This class reads an OS/400 physical file using JDBC and displays the results
 * to the console.
 *
 * @author yantzi
 */
public class JDBCExample {
    @return
    @see
    @serial
    @serialData
    @serialField
    @since
    @throws
    @version
}

```

Content assist for JavaDoc comments

Content assist is also available for JavaDoc comments inside the Java editor. Content assist for JavaDoc can provide suggestions for HTML tag names and JavaDoc keywords. Again the list will be subset by what is already typed.

## Templates

- Templates can be used to insert frequently used code patterns with content assist, for example:
- Lots of templates are predefined for Java Tools
  - Iterate over an array using for loop
    - for *<ctrl + space>*
  - Inserting a try / catch block
    - try *<ctrl + space>*
  - Public, protected or private method
    - public *<ctrl + space>*
- Define your own in the Java -> Templates preferences page
  - Insert a customized JavaDoc comment for your methods



# Creating Templates

**New in 5.0**

"Name" is used to match in content assist for listing templates

This code is inserted into the editor when the template is selected

Substitution variables can be filled in automatically or by the user

Name	Context	Description
<input checked="" type="checkbox"/> filecomment	java	File comment used by the c...
<input checked="" type="checkbox"/> for	java	Iterate over array w/ temp...
<input checked="" type="checkbox"/> for	java	Iterate over collection
<input checked="" type="checkbox"/> if	java	if statement
<input checked="" type="checkbox"/> ifelse	java	if else statement
<input checked="" type="checkbox"/> instanceof	java	dynamic type test and cast
<input checked="" type="checkbox"/> lazy	java	lazy creation
<input checked="" type="checkbox"/> main	java	main method
<input checked="" type="checkbox"/> new	java	create new object
<input checked="" type="checkbox"/> private_...	java	private method
<input checked="" type="checkbox"/> private_...	java	private static method
<input checked="" type="checkbox"/> protecte...	java	protected method
<input checked="" type="checkbox"/> public m...	java	public method

```

Preview:
for (int ${index} = 0; ${index} < ${array}.length;
    ${index}++)
    ${cursor}
    
```

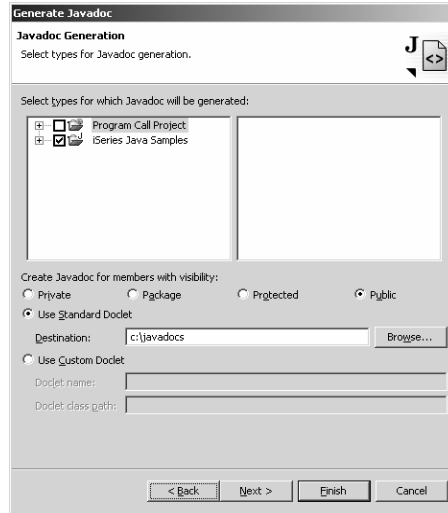
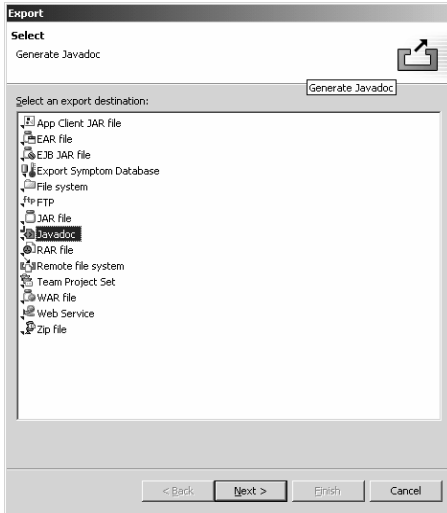
## JavaDocs – The Easiest Way to Document Code

- Java Documentation
  - Standard way for documenting Java code allows you to automatically generate HTML documentation for your classes!
  - You insert JavaDoc comments for your class, public constants and methods then run the class through the javadoc utility
  - Comments can include HTML markup and specialized JavaDoc keywords
  - Export the JavaDoc
- Workbench Steps
  - Use the JavaDoc export wizard to automatically generate JavaDocs for your Java project
  - Select your project in the Workbench and select Navigate -> Open External JavaDoc (Shift + F2)



**New in 5.0**

# Exporting JavaDocs







# Viewing the JavaDocs

The screenshot shows a web browser window titled "Help - WebSphere Development Studio Client Advanced Edition for iSeries". The search bar contains "GO" and "Advanced Search". On the left, under "All Classes", "JDBCExample" is listed. The main content area displays the following information for "Class JDBCExample":

```

com.ibm.java.samples.iseries
Class JDBCExample

java.lang.Object
|
+--com.ibm.java.samples.iseries.JDBCExample

public class JDBCExample
extends java.lang.Object

Sample class to highlight the features of the Java Development Tools

Author:
    yantzi

Constructor Summary
JDBCExample()

Method Summary
static void main(java.lang.String[] args)
    
```



IBM Software Group

## Java Visual Editor

WebSphere. software



July 2003 | Java Tools

© 2003 IBM Corporation

The Java editor has basic editing features like cut, copy, paste and color coding keywords and comments. The next few slides will cover some of the additional features of the workbench Java editor.

## Java Visual Editor

- Graphically layout the user interface for your Java applications
  - Drag and drop parts from the palette onto the Java window
  - Use the properties view to customize each part
  - Then go to the code and add the logic behind the user interface
    - No event support in the Visual Editor yet; Stay tuned
- Supports both AWT and Swing (the two most common Java widget toolkits)



# Java Visual Editor

**New in 5.0**

**Select parts from the palette and drop them on the window**

**Select a part on the window and change its properties**

**Click here to show the source**

The screenshot shows the Java Visual Editor interface. The main window is titled "To-Do List - Swing Controls" and contains a graphical representation of a Swing window with an "Add Item" button and a list. On the left, a "Swing Component Palette" lists various UI components like JButton, JCheckBox, JRadioButton, JToggleButton, JLabel, JTextField, JPasswordField, JTextArea, and JSlider. On the right, a "Properties" window displays a list of properties for the selected component, such as actionPerformed, alignmentX, alignmentY, autoscrolls, background, border, borderPainted, >bounds, componentOrientation, contentAreaFilled, cursor, disabledIcon, enabled, enabledSelected..., doubleBuffered, focusPainted, font, foreground, horizontalAlignment, horizontalTextPosition, icon, locale, location, and margin. At the bottom, a "Source" button is circled, with a callout pointing to it.



IBM Software Group

## Nominees For Best Supporting View

WebSphere. software



@business software

July 2003 | Java Tools

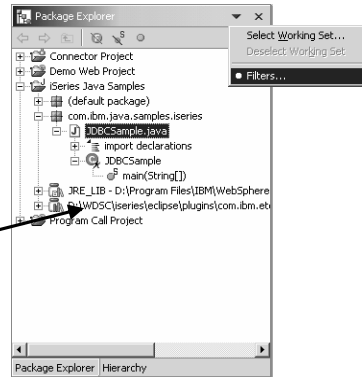
© 2003 IBM Corporation

Within the workbench views can be used to greatly increase productivity when used in conjunction with the editor. The Java tools use the following views which are outlined on the coming slides:

- Packages View
- Hierarchy View
- Outline View (this is a standard workbench view used by many perspectives)
- Tasks View (this is a standard workbench view used by all perspectives)

## Packages View

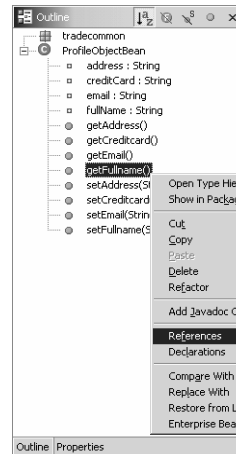
- Central location for managing your Java resources
  - Shows only Java related projects
  - Web projects are also Java projects
- Use **Working Sets** to show only a subset of all Java projects
- Displays external jar files included in the project's build path
  - Hint: Can be turned off with filters
- Use filters to control what is shown in the packages view



Packages are shown as their corresponding folders (com / ibm / etools / iseries / ...) which takes up a lot of the room in the tree view. The Packages view addresses both of these issues by showing only Java projects (and those projects which have a Java nature like Web projects) and showing Java packages as a single entry in the tree view. The Packages view also lets you see any referenced libraries directly in the tree view. Reference libraries are setup in the properties for the Java project and are covered later in this presentation.

## Outline View

- Works in concert with the Java Editor
- Shows imports, fields, methods
  - Can subset what is shown
  - Can sort by name
- Can show only selected member in editor
- Generate getter and setter methods for fields
- Generate Javadoc comments



**Refactoring allows you to easily rename or move a method and the tool automatically finds and updates all references**

**Shortcut for menu options for searching**

The Outline view shows the structure of the Java class currently being edited. Its main function is to provide you with a high level view of the structure of the class and provide easy navigation in the editor. Selecting a method / inner class in the outline view automatically positions the editor to the corresponding source code. The Outline view also provides many additional actions. These are available when you right click on the different items (classes, methods, fields) in the Outline view. For example: Right click on a field in the Outline view to see the option to generate standard getter and setter methods for the field. To remove a method right click on the method in the Outline view and select "Delete". The method and its corresponding code are deleted from the editor. The Outline view also provides some additional functionality such as:

- Toggling showing fields on / off
- Toggling showing static methods on / off
- Sorting the list of fields / methods
- Showing only the selected field or method in the editor instead of the entire class file

## Hierarchy View

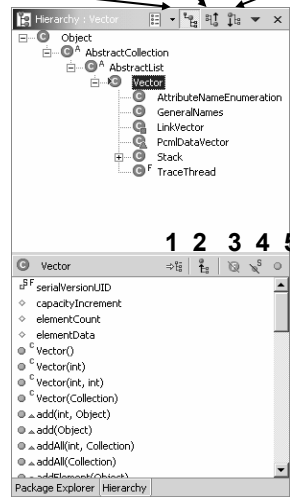
- **Provides 3 different ways to view a class in its type hierarchy**
  - Show only supertype hierarchy
    - This includes superclasses and interfaces!
  - Show only subtype hierarchy
  - Show class in its full hierarchy tree
    - Includes both superclasses and subclasses
- **Ability to perform actions**
  - Create copies of methods from superclasses in current class
    - Right click on method and select "Create in *classname*"
  - Refactoring
    - Rename methods and fields
    - Rename parameters for methods
    - Generate getter and setter methods for fields

The Hierarchy view uses a tree view to show a class relative to its superclasses and subclasses. This is very useful to search for a method in the superclasses of the class currently being edited. Using the hierarchy view you can override a method in a superclass by selecting the method in the superclass, right clicking and selecting "Create in ..." from the popup menu. This creates a method stub for the selected method in the current class which you can then go and provide the new implementation for.



Type Hierarchy    Supertype Hierarchy    Subtype Hierarchy

To open a class in the hierarchy view highlight the class name in any source file or view and press F4.



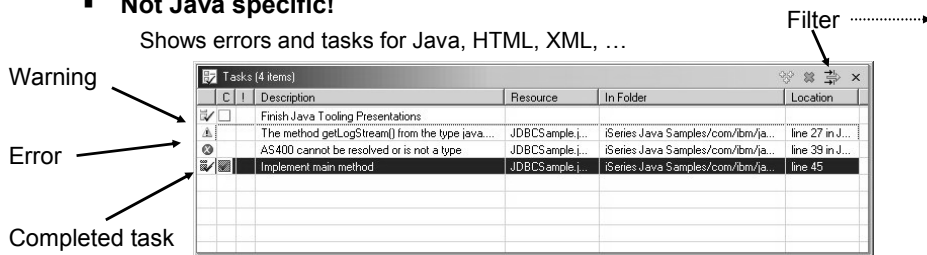
1. Shows selected members in hierarchy at all locations where they are declared.
2. Show / hide inherited members
3. Show / hide fields
4. Show / hide static members
5. Show / hide nonpublic members

Here you see the Hierarchy view with descriptions for the various features of the view. The first option (show selected members in hierarchy) lets you select a member from the list at the bottom of the view and the tree view at the top will be updated to show which classes have defined this method. This allows you to quickly see where the method has been overridden.

## Tasks View

- **Shows:**
  - ▶ All compile errors, warnings and information associated with resources in the workspace
  - ▶ Tasks defined by the user
    - Tasks can be set on a specific line in a resource or globally
    - "Don't forget to add error checking here!"

- **Not Java specific!**
  - Shows errors and tasks for Java, HTML, XML, ...



Warning

Error

Completed task

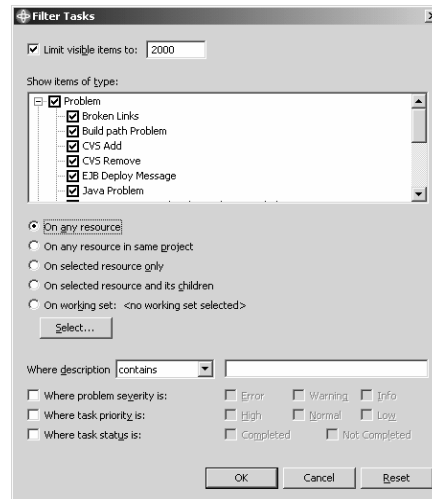
C	I	Description	Resource	In Folder	Location
<input type="checkbox"/>	<input type="checkbox"/>	Finish Java Tooling Presentations			
<input type="checkbox"/>	<input type="checkbox"/>	The method getLogStream() from the type java...	JDBCsample.j...	iSeries Java Samples/com/ibm/ja...	line 27 in J...
<input type="checkbox"/>	<input type="checkbox"/>	AS400 cannot be resolved or is not a type	JDBCsample.j...	iSeries Java Samples/com/ibm/ja...	line 39 in J...
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Implement main method	JDBCsample.j...	iSeries Java Samples/com/ibm/ja...	line 45

Filter

The Tasks view is a standard workbench view showing all errors, warnings and user defined tasks in the workspace. Because it shows all workspace tasks it provides filtering capabilities to scope what is shown in the actual view by the type of problem / task or based on what is selected in the Navigator view or Packages view.

## Tasks View - Filtering

- Type of problem
  - Java problem, XML Schema problem, broken links in HTML or JSP
- Severity of the problem or priority of the task
- Problem / task description
- Based on what is selected in the navigator / packages view
- Status of the task



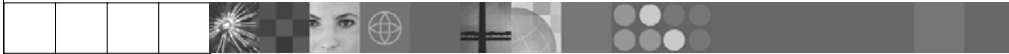
You can filter the Tasks view by type of problem, problem severity, problem or task description, the selection in the Navigator view or Packages view or the task status.



IBM Software Group

# Compiling Your Java Code

**WebSphere.** software

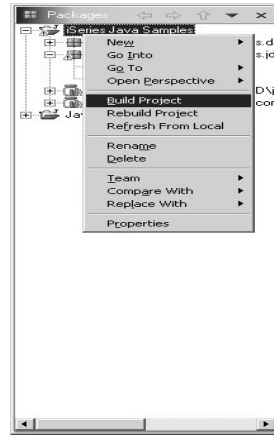


July 2003 | Java Tools

© 2003 IBM Corporation

## Compiling your Java class

- Every time you save your file it is automatically compiled
- Manual compile
  - Right-click project, package or class, select
    - Build Project
    - Rebuild Project



When you save a Java class (.java file) it is automatically compiled and all errors / warnings are shown in the tasks view. However you can also force a recompile using either the build or rebuild project actions. The Build Project action is an incremental build that builds only resources that have changed since the last build. The Rebuild Project is a full build that discards the previous build stat and rebuilds everything.



# Running Your Java Code





## Running your Java Class

- Standalone Java application
  - Class implements method
  - Run locally
  - Run remotely
  
- Servlet
  - Class extends `javax.servlet.http.HttpServlet`
  - Servlet can be run on different test environments and servers

How you run a Java class will depend on what type of Java application it is. If it is a standalone Java application (i.e. it contains a public static void main(String[] args) method) then it can be run locally using either the JDK shipped with the workbench (this is the default) or any JDK installed on the local workstation. You can specify which JDK to use in the Java project properties. You can also run the standalone Java application remotely using iSeries extensions.

If the Java class is a servlet then it must be run in either the local WebSphere Test Environment (default) or any other

application server environment supported by the workbench. Currently this includes:

- Local WebSphere Test Environment
- Local Tomcat Test Environment
- Local Tomcat Application Server
- Remote WebSphere Application Server

## Standalone Java Application

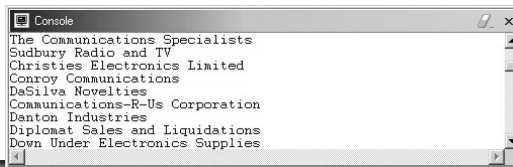
1. Select java file in packages view

2. Select Run -> Java Application



Debug perspective opens and output shown in Console view

Tip: Turn off debug preference to "Show debug perspective when program launched in run mode".



Here you see how to run the standalone JDBC Sample Java application used through out this presentation. To run the application simply select the class in the packages view and from the Run icon in the toolbar select Run -> Java Application. If the application writes any output to either System.out or System.err then the workbench console view will appear with the output. If the Java class were a servlet then to run it in the local WTE you would select the class from the packages view and select "Run on Server" from the popup menu.

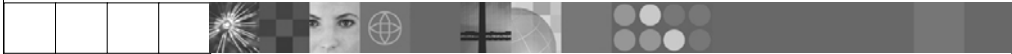




IBM Software Group

# Debugging Your Java Code

**WebSphere.** software



July 2003 | Java Tools

© 2003 IBM Corporation

## Debug



Shortcut: Run and debug menus remember the last few applications that were run.

You can select this instead of selecting the class and then choosing Debug -> Java Application.

Tip: Set breakpoints in your source code before launching the debugger.

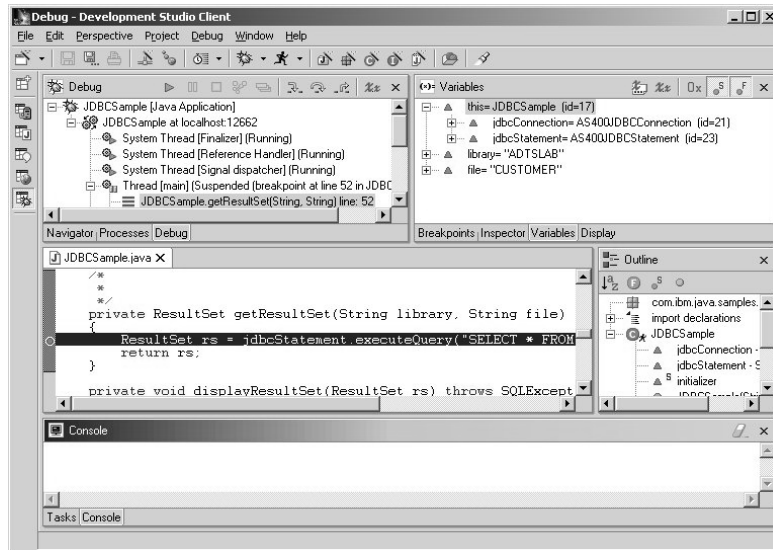
You can set breakpoints in the Java editor by double clicking in the left hand margin of the editor.



Opens debug perspective ...

To debug the Java application select the class in the packages view and select Debug -> Java Application from the Debug toolbar icon drop down menu. It is a good idea to set breakpoints in your source code before launching the application in debug mode. This will cause the debugger to stop at the first breakpoint it hits in the application.

## Debug Perspective

Details 

Here you see the Debug perspective.

## Debug Perspective – Tools and Views

Shortcut buttons for:

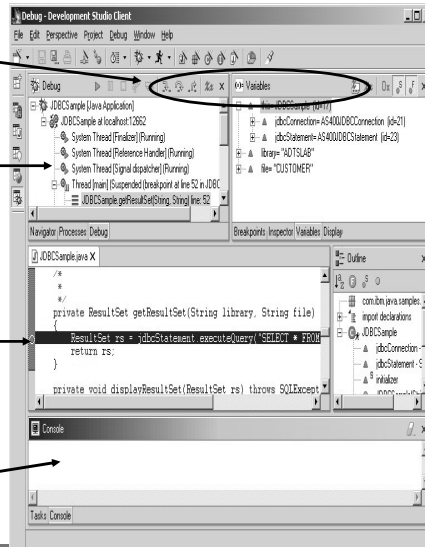
- ▶ Resume
- ▶ Suspend
- ▶ Stop
- ▶ Step into / over / return

Debug view shows:

- ▶ All running applications
  - ▶ Threads
    - ▶ Execution stack for suspended threads

Breakpoint

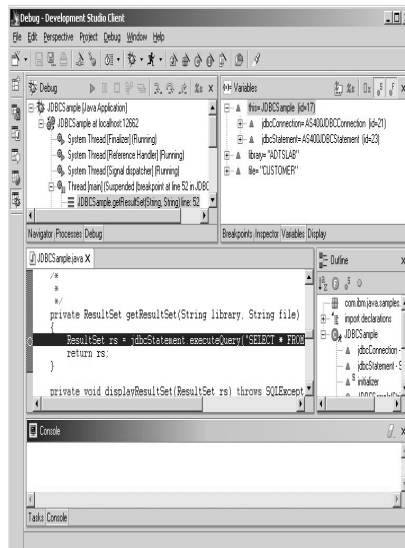
Console View



The top view shown in the screen shot (Debug view) shows all applications that are currently being debugged in a tree view. Below each application all threads are shown, and for each suspended thread the corresponding stack trace. The Debug view contains icons for resuming a suspended thread, suspending a thread, stopping an application and for stepping into / over a line of code or running to the end of a method. These options are also available from the

Debug menu and have associated shortcut keys. Below the Debug view is the source view where source files are opened if a breakpoint is hit.

## Debug Perspective - Views



- Breakpoints View

- Inspector View

- Variables View

- Display View

To the right of the Debug view is another pane showing the 4 views listed on this slide. The two main views are the Breakpoints view (used for managing breakpoints) and the variables view (used for viewing and changing the contents of variables in the application).

### Breakpoints View

- Manage breakpoints in the workspace

- Add Java exception breakpoints

- Stop when a NullPointerException is thrown

- This can be very handle in locating problems!

### Inspector view

- Detailed view for inspecting variables

### Variables View

- Shows all variables that are visible in the current stack frame

### Display View

- Displays result of executing an expression in context of current stack frame

- Highlight expression in editor and select "Display" from the popup menu

## Two main ways to search

- Global workspace
  - Search the entire workspace
- Context sensitive searching
  - Scope to workspace or class hierarchy
  - Packages / Classes / Interfaces / Variables / Members
- Interfaces
- Results are shown in the Search view

There are two main ways to search for "things" in the workspace. The first is just a simple raw text search that will search for a string pattern in any type of file.

The second is a "Java-aware" search that will let you search Java files for strings that are context specific. Such as Declarations of all methods that start with getA\*

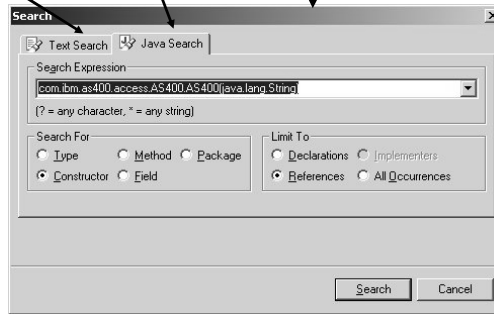
References to all methods that start with getA\*

# Java Aware Search



Simple text searching  
 Java "aware" searching

Find all types, methods, packages, constructors or fields that match the search expression.



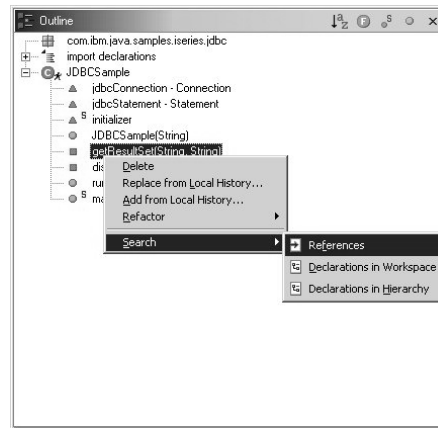
Search only for declarations or references to the specified type, method, package, ....

Here you see the "Java-aware" search dialog. You use the radio buttons to define the context of the search.

## Context Searching

Tip: Anywhere you see a package / method / variable name you can right click on the name and select "Search -> ..." from the popup menu.

This is equivalent to selecting the search dialog entering the name and setting the "Search For" and "Limit To" fields.



Instead of having to type the class / method name and select all the radio buttons in the previous slide, the user can use the popup menu on fields / methods in the Outline view to also search for references and declarations. This way you will not be prompted.

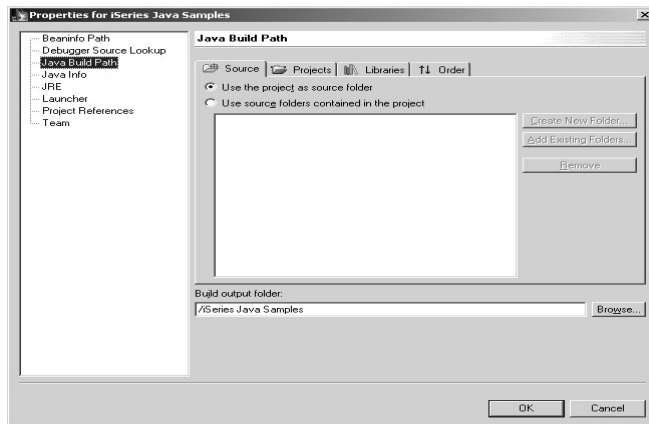


## Java Build Path

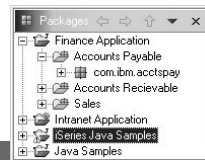
- Each Java project has a build path the defines
  - How resources are built
    - Where output (.class files) are placed after a build
  - Where external resources are located
    - Source from other projects in the workspace
    - JAR files from other projects
    - External JAR files
- Similar to setting the CLASSPATH when running applications outside the workbench

Each Java project has a build path property which specifies which projects and internal / external (relative to the workspace) jar files this project requires in order to compile and run. This is very similar to setting the CLASSPATH environment variable for a JDK. To set the build path, right click on the Java project and select "Properties" from the popup menu.

## Java Build Path - Source

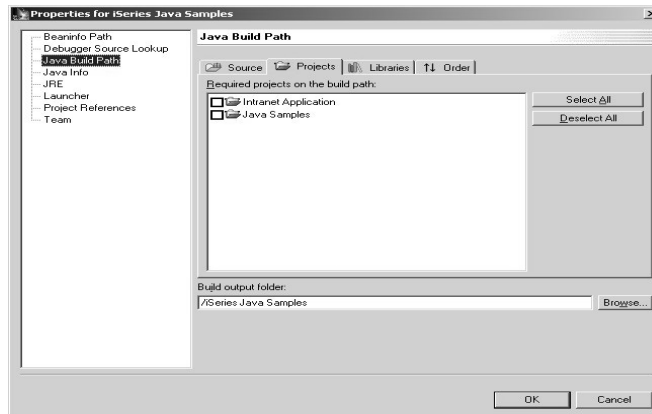


- ▶ Specify whether project uses source folders or not
- ▶ Source folders are recommended for large projects
  - ✓ Provides an additional level of grouping between the project and Java packages



In the Java project properties dialog select "Java Build Path". This displays a four page notebook. The first page lets you specify whether or not this project uses source folders to organize Java packages. For large projects, source folders give you an extra level of organization for your Java code.

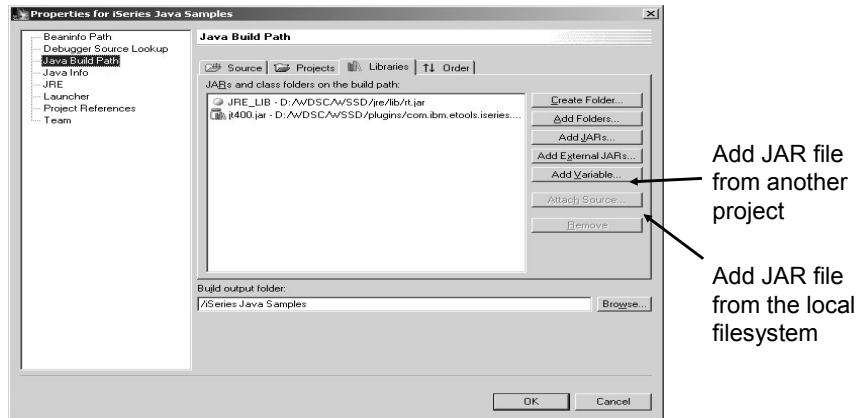
## Java Build Path - Projects



- ▶ Select all projects in the workspace that are required by this project
  - ✓ i.e. This project uses classes in the other project

The "Projects" page lets you specify which other Java projects in the workspace should be included in the build path for the current Java project.

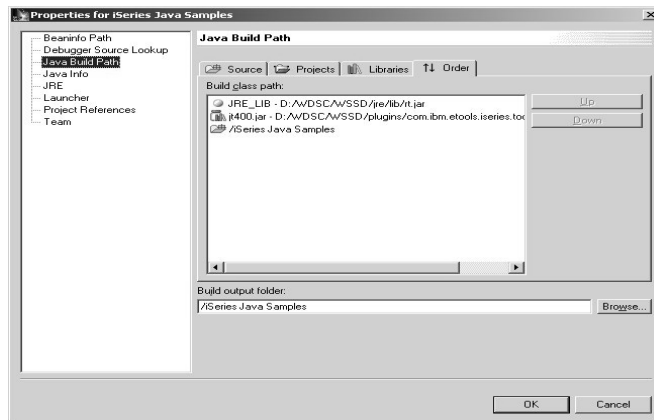
## Java Build Path - Libraries



- ▶ These are NOT OS/400 libraries!
- ▶ Specify all JAR files required by your Java project
  - ✓ i.e. IBM Toolbox for Java
    - d:\WDSC\WSSD\plugins\com.ibm.etools.iseries.toolbox\runtime\jt400.jar

The "Libraries" page lets you specify which JAR files should be included in the build path for the current project. Use the Add JARs button if the JAR file is already located in the workspace. Use the Add External JARs button if the JAR file is located somewhere else on your local hard disk. The Add variable button is similar to Add External JARs except that it lets you defined a variable for the source path where the JAR file is located. If you have multiple JAR files in the same directory you can use a variable to point to the directory. That way if the directory name changes you only have to update the variable not every single entry.

## Java Build Path - Order



- ▶ Specify the order of projects and libraries included in the build path
- ▶ This is important if the different versions of the same class exists in two different entries

The "Order" page lets you specify the order that other projects and JAR files should be included in the build path. Select the entry in the list and use the move up / down buttons to change the order.



IBM Software Group

## iSeries Extensions to Java Tools

WebSphere. software



July 2003 | Java Tools

© 2003 IBM Corporation

So far this presentation has looked at the generic Java tools provided by WebSphere Studio Site Developer Advanced, nothing specific to the iSeries platform. Now we will look at iSeries specific extensions to the Java tools.

## Java Tools for iSeries

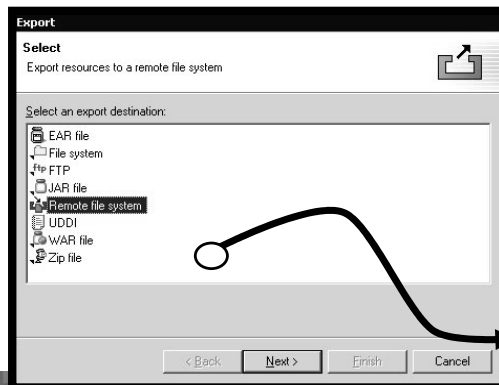
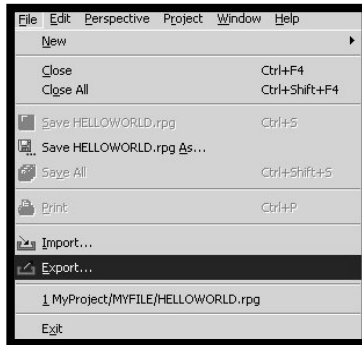
- Import and export from/to remote system
- Remote run and debug launch configurations
- Remote Compile and Run
- Remote Debug
- Program Call Wizard
- Toolbox for Java built-in
- Supplied Java-beans

Here is an overview of those extensions. The following slides will look at each one in more details.

- Export and import to/from IFS, through common remote system frame-work support
- Special Java "compile" actoin/view for remotely compiling Java classes
- Special Java run action/view for remotely running Java applications
- Special Java debug action/view for remotely debugging Java applications
- Program Call wizard to "wrapper" any \*PGM/\*SRVPGM as a Java Bean
- Toolbox for Java built-in: integrated help and runnable samples
- Previously supplied Java beans: datafile updates/access (DFU beans), object lists beans (PDM – library, object, member, field, record lists), and Swing GUI beans (dspf-like function)

## Import and Export

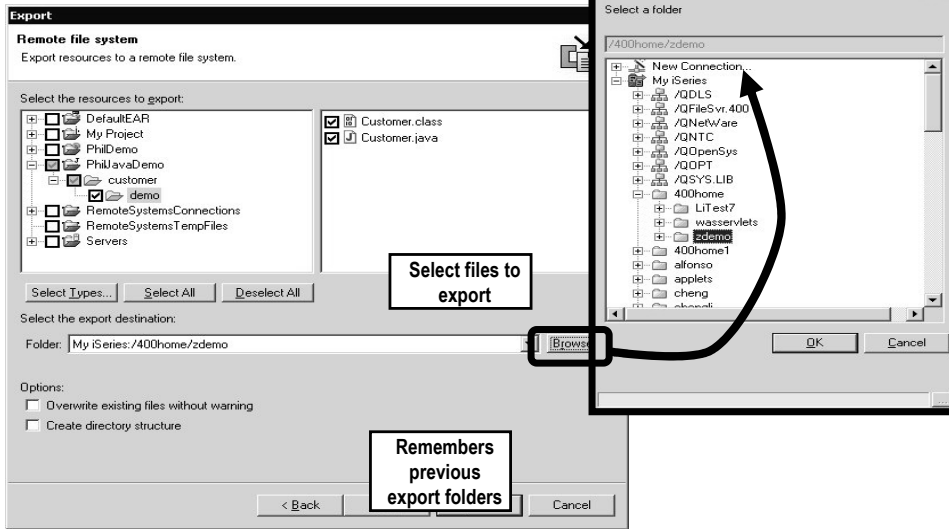
- ▶ Builds on base support for import/exporting files/jars/zips
- ▶ Add support to import/export files from remote system
  - ✓ Import any files from any remote system (iSeries/Unix/Linux/Windows)
    - ✓ Eg: IFS on iSeries; Linux LPAR on iSeries
  - ✓ Export files from any project to any remote system



The workbench allows you to import and export your code (Java and non-Java code) from / to a variety of locations. To this list we added the ability to import from and export to a remote system (iSeries or non-iSeries). Like all the other import / export wizards, this ability is not restricted to Java only. On the iSeries this wizard exports to and imports from the integrated filesystem (IFS).

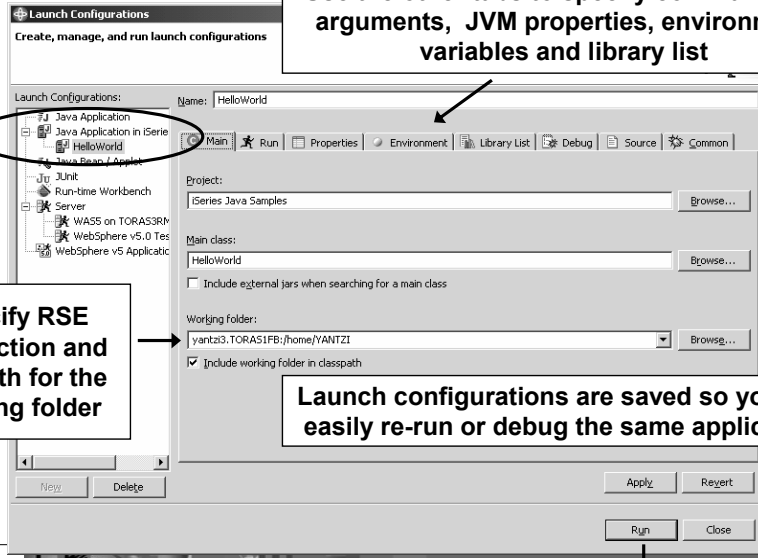


## Export resources



Here you see the next page of the export to remote file system wizard. Here you specify which files to export from their workspace and to which remote system and directory on that remote system. The import wizard has a similar page.

# Remote Run Launch Configuration



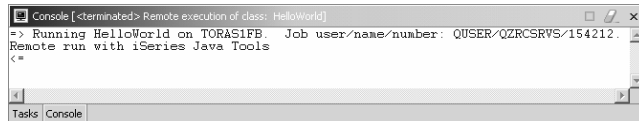
Use the other tabs to specify command line arguments, JVM properties, environment variables and library list

Specify RSE connection and IFS path for the working folder

Launch configurations are saved so you can easily re-run or debug the same application

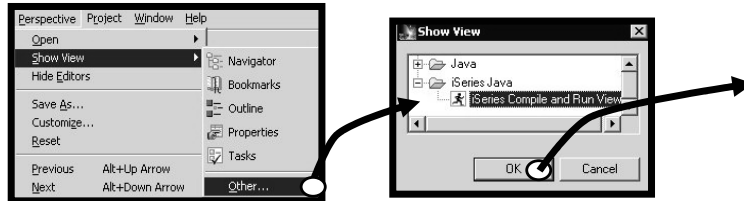
## Remote Run on iSeries

- Output from the Java application running (or being debugged) remotely appears in the Workbench's Console view
- The same launch configuration can be used to debug the Java application remotely
  - Same Java debugger is used for debugging Java applications running locally or remotely



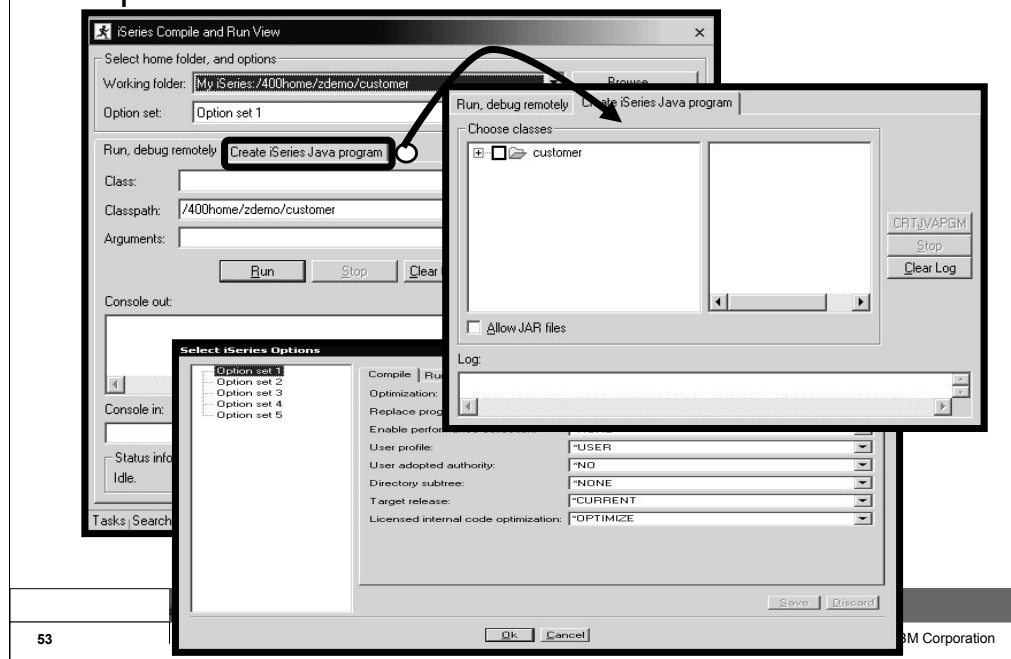
## Remote Compile and Run

- ▶ After exporting Java files to iSeries, you:
  - ✓ Transform them there using CRTJVAPGM
  - ✓ Run them there, see results via Console



Here you see how to open the Remote Compile and Run View. Open the "iSeries Console and Run View" to natively compile (CRTJVAPGM), run and debug your Java applications remotely on the iSeries.

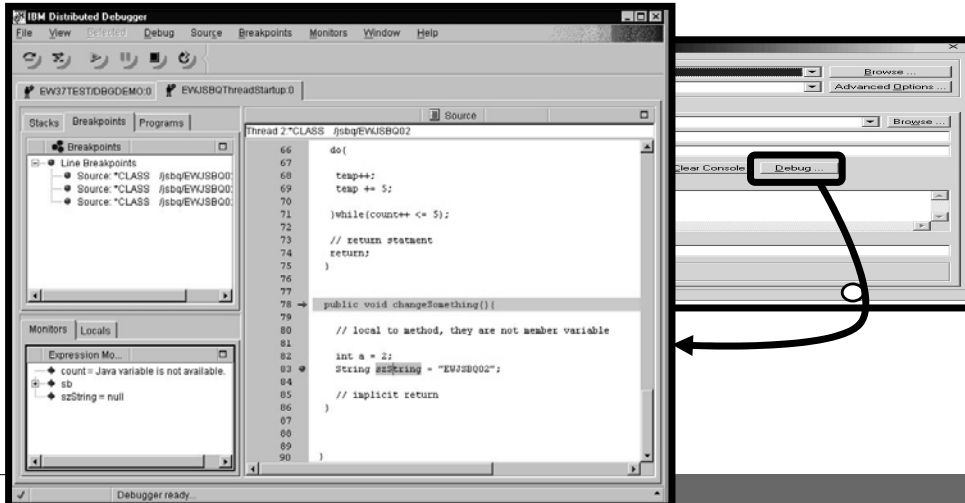
## Compile and Run View



At the top of the iSeries Compile and Run View you specify the remote system and directory (typically this is the same system and directory you specified on the export wizard.) Before running or compiling the Java class you can setup your options. For the compile action these options are the parameters that are passed to the CRTJVAPGM command. For running these options you setup your library list and environment variables. On the Run, Debug remotely tab of the view you can specify the Java class you want to run / debug along with the classpath and any command line arguments that need to be passed to the application. Console out and in allow you to view output from and send input to the remotely running Java application. Switching over to the Create iSeries Java Program tab allows you to run the CRTJVAPGM command on the iSeries to natively compile the selected Java class to machine instructions. These machine instructions are stored in a hidden object associated with the .class (or .jar) file and used by the JVM on the iSeries to increase runtime performance.

## Remote Debug

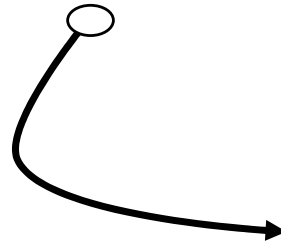
► Launches debugger for iSeries Java



Instead of running the Java application remotely you can instead debug the Java application as it runs on the iSeries. Clicking the Debug button launches the IBM Distributed Debugger.

## Program Call Wizard

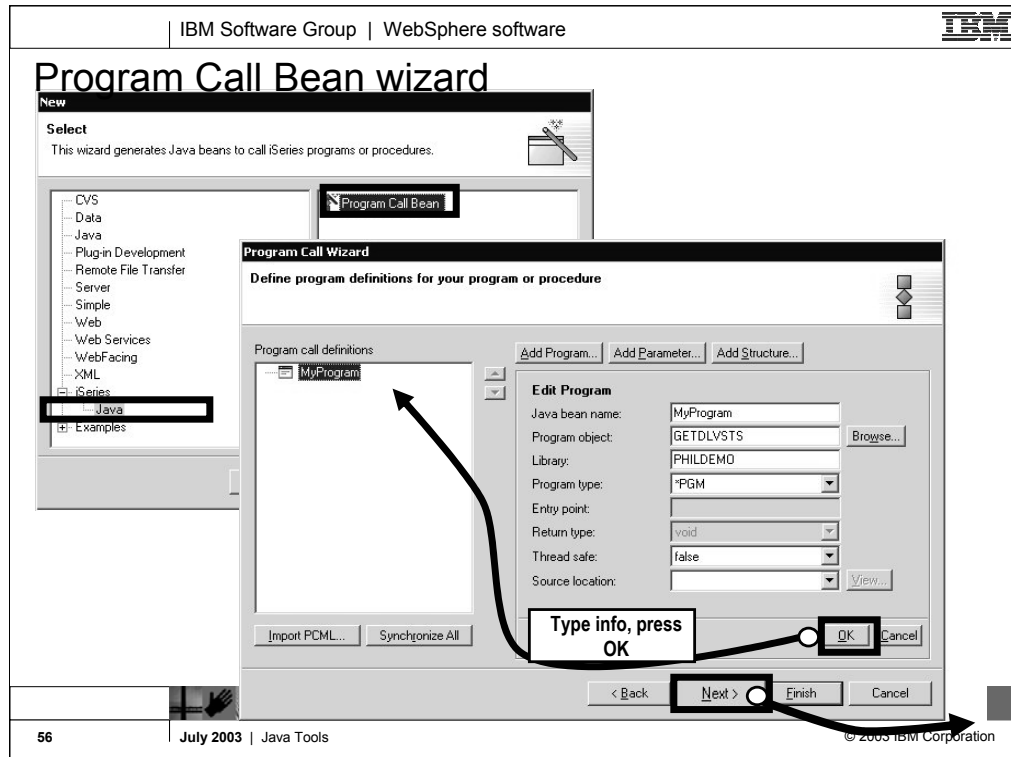
- Given signature of \*PGM or \*SRVPGM proc:
  - Generates Java Bean to call it
- Steps:
  - Select Java/Web/WebFacing project
  - Select package
  - Use File > New > Other
    - iSeries Java Program Call Bean



Here you step through the very popular Program Call wizard, which has been significantly enhanced from its old VisualAge for Java days. Using this wizard is easy.

First, within any view that lists projects, select a project to contain the generated Java code. You are allowed to select from Java, Web or WebFacing projects.

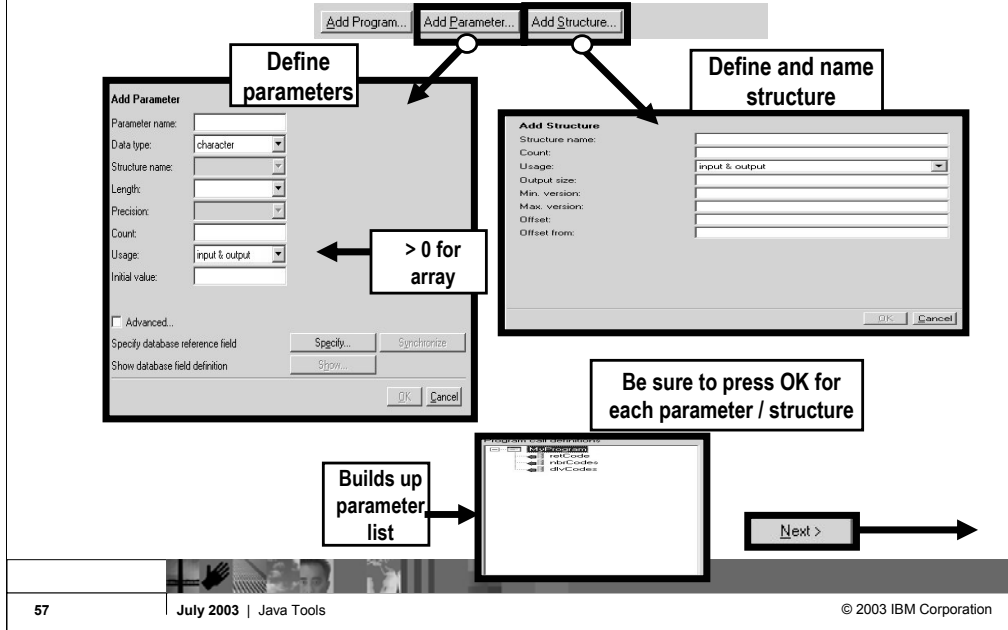
Only these types of projects support Java. Second, select the Java package within that project, into which the Java code will be generated. You must pre-create such a package if you don't have one. Third, launch the program call wizard, by using the File->New->Other menu item. In the resulting wizard, select the iSeries Java and then Program Call Bean. Follow the directions on the coming slides.



On the File->New->Other wizard selection page, select the Program Call Bean wizard and press Next. In the Program Call wizard, select the program or ILE procedure to be called from Java. This must be a non-interactive program or procedure.



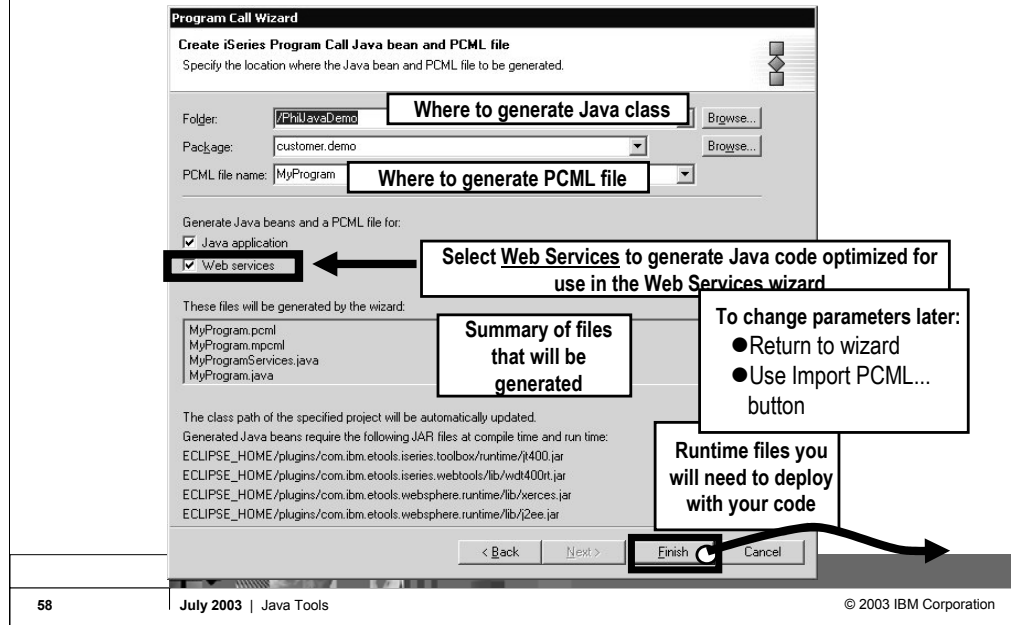
## Enter parameter information



57 | July 2003 | Java Tools | © 2003 IBM Corporation

Use the Add Parameter and Add Structure buttons to describe all the parameters to the program or procedure, including data type, length and decimals. Most importantly, specify if the parameters are input, output or both with respect to what the program does with the parameters (read, write or both). When all parameters are described, click Next to continue to the next page of the wizard.

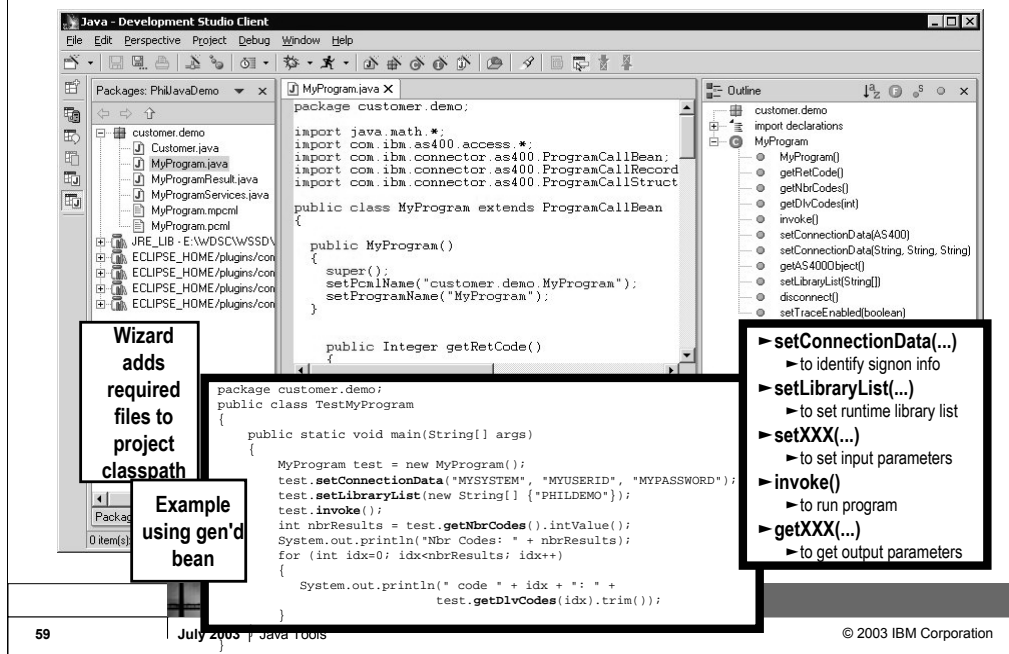
## Final page



Here you see that you can confirm where you want the Java code to be generated, including the Program Call Markup Language (PCML) file that is also generated.

For the generated Java bean to be used as a Web Service, select the Web Services checkbox. This generates a Java bean that is easily consumable by the Web Services wizard. Notice how the wizard will automatically update our project's classpath to include the necessary AS/400 Toolbox for Java JAR files so that our generated code will compile. Finally, press Finish to generate the Java bean!!

## Generated Java Bean



The screenshot shows the IBM Java Development Studio interface. The main editor displays the source code for `MyProgram.java` in the `customer.demo` package. The code includes imports for `java.math.*`, `com.ibm.as400.access.*`, and `com.ibm.connector.as400.*`. The `MyProgram` class extends `ProgramCallBean` and implements methods like `getRetCode()` and `getDivCodes(int)`.

Three callout boxes provide additional information:

- Wizard adds required files to project classpath:** Points to the Package Explorer on the left.
- Example using gen'd bean:** Points to the `TestMyProgram` class in the editor, which shows how to instantiate `MyProgram` and call its methods.
- Method List:** A box on the right lists methods available on the bean:
  - `setConnectionData(...)` - to identify signon info
  - `setLibraryList(...)` - to set runtime library list
  - `setXXX(...)` - to set input parameters
  - `invoke()` - to run program
  - `getXXX(...)` - to get output parameters

Page number: 59 | Date: July 2003 | Page: Java Tools | Copyright: © 2003 IBM Corporation

Here we see the result of the Program Call wizard. The selected Java package in the upper right now contains all the generated Java classes, one of which is shown in the editor in the middle view. To use the generated Java bean, we write Java code to instantiate it and call the important methods in it, which are listed in the information box in the lower left. An example of Java code to use the bean is shown in the lower middle information box. The pattern for using the generated bean is this:

1. Instantiate the bean using the New operator
2. Call the `setLibraryList` method with an array of library names to set up the library list
3. Call the `setXXX` methods to set the input parameter data. There will be one such method for each input parameter.
4. Call the `invoke` method to call the program or procedure
5. Test the boolean result of `invoke` for true or false to determine if the `invoke` was successful
6. Call the `getXXX` methods to read the output parameter data that the program updated. There will be one such method for each output parameter.

## GUI Beans for iSeries

- Java Beans supplied for DDS-like field error checking and formatting
  - JFormattedTextField (smart entry field)
  - JFormattedLabel (smart lable constant)
  - JFormattedComboBox (smart dropdown)
  - JFormattedTable (smart-multi-column list)

Included with the iSeries extensions to the Java tools are some GUI and non-GUI Java beans that can be used in applications developed by WebSphere Development Studio Client customers. Here you see the GUI beans included with Development Studio Client. These beans extend their Swing counterparts but add customizations that iSeries developers are used to with display files such as:

Restricting a value to numeric, string, ...

Specifying the values length and precision

Specifying formatting using edit codes / words

### JFormattedTextField

- Error checking based on data type, length, decimals
- DDS-like validity checking: range or comparison
- Editcode or editword formatting and masking
- Auto-advance

### JFormattedLabel

- Editcode or editword formatting

### JFormattedComboBox

- Combo of JFormattedTextField and JFormattedLabels

### JFormattedTable

- A subfile or multi-file column list box
- Each cell is a JFormattedTextField or JFormatted Label or JFormattedComboBox

# GUI Beans Examples

•File->New->Other->Examples->iSeries

► JFormattedTable

ItemNo	Model	Size	Price
00001	2000-AB-1	small	100
00002	2000-AB-2	small	200.50
00003	2000-AB-3	medium	369
00004	2000-AB-4	big	700
00005	2000-AB-5	small	405
00006	2000-AB-6	small	110
00007	2000-AB-7	small	230.49
00008	2000-AB-8	medium	268
00009	2000-AB-9	big	732

► JFormattedText

Symbol:

Price:

Quantity:

Account No.:

Password:

Contact Phone:

## DFU Beans for iSeries (non-GUI)

- Java Beans supplied for querying data from DB2/400, using direct record access (versus SQL):
  - RecordIOManager
  - ListManager
  - FormManager

Here you see some non-GUI beans that can be used in a Java application to assist the developer in reading records from an iSeries physical file. The ListManager retrieves lists of records for display in a GUI list, whereas the FormManager can be used to retrieve a single record for display in a GUI form.

### RecordIOManager

- Query, add, update, or delete records in a database
- Vastly simplifies task of writing database access code using Toolbox for Java classes

### ListManager

- Maps list of records from RecordIOManager to a Swing GUI list
- Makes it easy to populate GUI list with DB2/400 data

### FormManager

- Maps details of a single row from RecordIOManager to a single GUI dialog page
- Makes it easy to populate GUI form with DB2/400 data

# DFU Beans Examples

- ▶ File->New->Other->Examples->iSeries
- ▶ ListManager

FormManager with a multiple record format logical file

Order: DEF456 Next Order

Order Date: 1999-03-24 INVREC  
SALESREC

Customer Order: aaaaaaaa

Customer: bbbbbbbbbbb

Ship Via: ccc

Ship To: 0555555

Price: 55555.33

Goods: 6666

PARTNO	MODEL	PARTPRI	PARTMSR	PARTDIS	PARTSHIP
00562	AR-10	26.10	29.00	Y	1991-05-10
00562	AR-11	32.04	36.00	Y	1992-12-02
00562	AR-12	7.92	9.00	N	1994-05-07
00562	AR-13	11.31	13.00	N	1991-09-25
00074	AR-1	35.64	36.00	N	1990-07-07
00074	AR-11	31.15	35.00	N	1990-01-08
00074	AR-12	29.92	34.00	N	1994-03-19
00074	AR-14	39.56	46.00	N	1993-12-22
01807	AR-10	24.30	27.00	N	1994-06-16
01807	AR-11	20.47	23.00	N	1993-10-13

readAllRecords

▶ FormManager

## List Beans for iSeries (non-GUI)

- For querying lists of libraries, objects, members, records or fields
  - AS400ListLibraries
  - AS400ListObjects
  - AS400ListMembers
  - AS400ListRecords
  - AS400ListFields
- Four levels of detail possible per object

Here you see some additional non-GUI beans for retrieving lists of libraries, objects, members, records and fields from an iSeries.

### AS400ListLibraries

- List libraries given simple or generic or special name

### AS400ListObjects

- List objects given simple or generic library and object name
- Can also subset by type and attribute (one or multi, simple or generic)

### AS400ListMembers

- List members given simple or generic library, file, member name
- Can also subset by member type (one or multi, simple or generic)

### AS400ListRecords

- List records given simple or generic library, file, record name

### AS400ListFields

- List fields given simple or generic library, file, record, field name

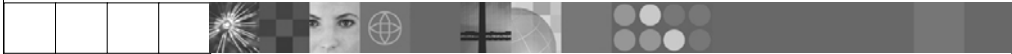




IBM Software Group

## Summary

**WebSphere.** software



July 2003 | Java Tools

© 2003 IBM Corporation

## Summary

- Java Development Tools included with Development Studio Client
  - Part of WebSphere Studio Site Developer Advanced
- Java perspective provides optimized view for Java development
  - Java Editor + Java Views = Easier Development
  - Integrated Compile/Run/Debug
  - iSeries Additions for using Java with iSeries
- Lots more
  - Team development
  - Integrated, online help system
  - Extendable platform – you can add your own tools
  - Integrated Java, Web, XML, iSeries development



## Resources

- Eclipse. Org  
Web site for the open source Workbench
- WebSphere Development Studio Client  
[ibm.com/software/adwtools/series](http://ibm.com/software/adwtools/series)
- WebSphere Developer Domain  
[ibm.com/websphere/developer](http://ibm.com/websphere/developer)
- IBM Developer Domain  
[ibm.com/developer](http://ibm.com/developer)
- Java  
[java.sun.com](http://java.sun.com)



## Trademarks & Disclaimers

© IBM Corporation 1994-2003. All rights reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AS/400	IBM (logo)
AS/400e	iSeries
e (logo) business	OS/400
IBM	

Lotus, Freelance Graphics, and Word Pro are registered trademarks of Lotus Development Corporation and/or IBM Corporation. Domino is a trademark of Lotus Development Corporation and/or IBM Corporation.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.



## Disclaimer

- **Acknowledgement:**

This presentation is a collaborative effort of the IBM Toronto iSeries Application Development presentation team, including work done by:

Phil Coulthard, George Farr, Claus Weiss, Don Yantzi, David Slater, Alison Butteril, Linda Cole

- **Disclaimer:**

The information contained in this document has not been submitted to any formal IBM test and is distributed on an as is basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customers' ability to evaluate and integrate them into the customers' operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

- **Reproduction:**

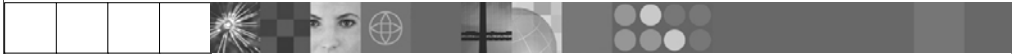
The base presentation is the property of IBM Corporation. Permission must be obtained PRIOR to making copies of this material for any reason.



IBM Software Group

# Java Tools WebSphere Development Studio Client V5.0

**WebSphere.** software



July 2003 | Java Tools

© 2003 IBM Corporation