WebSphere Development Studio Client:

Java Development Tools

George Farr
IBM Toronto Laboratory

WDS for iSeries
New World
New Servers
New Tools

- This presentation looks at the Java Development Tools (JDT) included in WSSDa and the iSeries extensions added to this tooling in WDSc.
- 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.

Disclaimer

Acknowledgment:
- This presentation is a collaborative effort of the IBM Toronto AS/400 Application Development presentation team, including work done by:
  - *Don Yantzi, Phil Coulthard, George Farr, Claus Weiss, 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.

Agenda

► **Getting Started**
  ● **Creating a HelloWorld program**

► **The Java Editor**

► **Java Views**

► **Compile / Run / Debug**

► **iSeries Additions**

► **Reference**

IBM @server. For the next generation of e-business.

Eclipse

► **Quick Overview**

IBM @server. For the next generation of e-business.

Eclipse

WDSc: Java Tools

✓ Java Development Tools (JDT):

- **Choice of Java level**
  - ▶ **IBM JRE 1.3 built-in and offered as default**
- **Java projects**
  - ▶ **Optimized for Java development**
  - ▶ **Specialized views**
    - ✓ Package view and Hierarchy view;
    - ✓ Outline view: many tools launchable from here!
  - ▶ **Java Editor**
    - ✓ Content Assist: Ctrl+Space bar to get list of classes or methods
    - ✓ JavaDoc shown as hover help
    - ✓ F3 to open selected class
  - ▶ **Refactoring support**
    - ✓ Rename updates all references
  - ▶ **Incremental compile**
    - ✓ When Ctrl+S pressed in editor
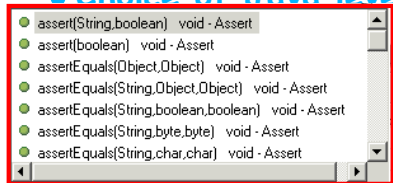   - ✓ Or on demand via Build Project action

IBM *@server.* For the next generation of e-business.

Eclipse

- ▶ By default JDT uses the IBM JRE 1.3 that is shipped with the Workbench however the user is able to switch this to any JDK that is installed on the local system.
- ▶ Java projects have an associated builder that knows how to compile Java classes. Properties for a Java project include the build path where the user specifies other projects, and jar files that are required for this project (i.e. sets up the classpath.)
- ▶ Refactoring support gives the user the ability to easily restructure their code without having to manually update other classes that are affected by the changes.
- ▶ Whenever a Java class is saved, the source is automatically compiled.

WDSc: Java Tools

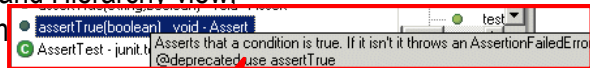✓ **Java Development Tools (JDT):**

● Choice of Java level

```
assert(String,boolean)   void - Assert
assert(boolean)   void - Assert
assertEquals(Object,Object)   void - Assert
assertEquals(String,Object,Object)   void - Assert
assertEquals(String,boolean,boolean)   void - Assert
assertEquals(String,byte,byte)   void - Assert
assertEquals(String,char,char)   void - Assert
```

-in and offered as default

a development

✓ Package view and Hierarchy view:
✓ Outline view: m

```
assertTrue(boolean)   void - Assert
AssertTest - junit.t   Asserts that a condition is true. If it isn't it throws an AssertionFailedErro
                       @deprecated use assertTrue
                                                      test
```

► **Java Editor**
  ✓ Content Assist: Ctrl+Space bar to get list of classes or methods
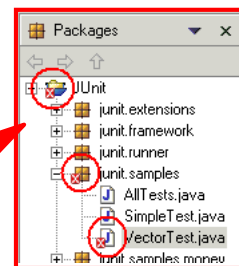 ✓ JavaDoc shown as hover help
  ✓ F3 to open selected class
► **Refactoring support**
  ✓ Rename updates all references
► **Incremental compile**
  ✓ When Ctrl+S pressed in editor
  ✓ Or on demand via Build Project action

```
Packages                 ▼  ✕
⇦ ⇨ ⇧
    JUnit
  junit.extensions
  junit.framework
  junit.runner
  junit.samples
    AllTests.java
    SimpleTest.java
    VectorTest.java
  junit.samples.money
```

IBM @server. For the next generation of e-business.

▸ This slide just shows examples of the content assist and JavaDoc hover help available in the Java editor.
▸ The packages view is also shown. The packages view flags classes that contain compile errors with a red x.

Java Development Tools

► **Getting Started**

IBM @server. For the next generation of e-business.

Eclipse

► Open the Java perspective

● Perspectives

- ► Define which views are shown and their layout in the Workbench
- ► Targeted towards a specific type of task
  - ✔ Java, Help, Web, WebFacing, Remote Systems Explorer, XML, ...

● Java perspective contains Java related views

- ► Packages
  - ✔ Shows only the Java projects in your workspace
  - ✔ Collapses package directories into a single folder in the tree view
- ► Outline (standard workbench view)
  - ✔ High level view overview of the structure of your Java class
- ► Hierarchy
  - ✔ Inheritance hierarchy of the Java class
- ► Tasks (standard workbench view)
  - ✔ Shows compile errors and warnings

IBM @server. For the next generation of e-business.

Eclipse

- ► 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 is outlined in more detail on the following slides.

Java Perspective

Shortcuts to related perspectives

All available perspectives
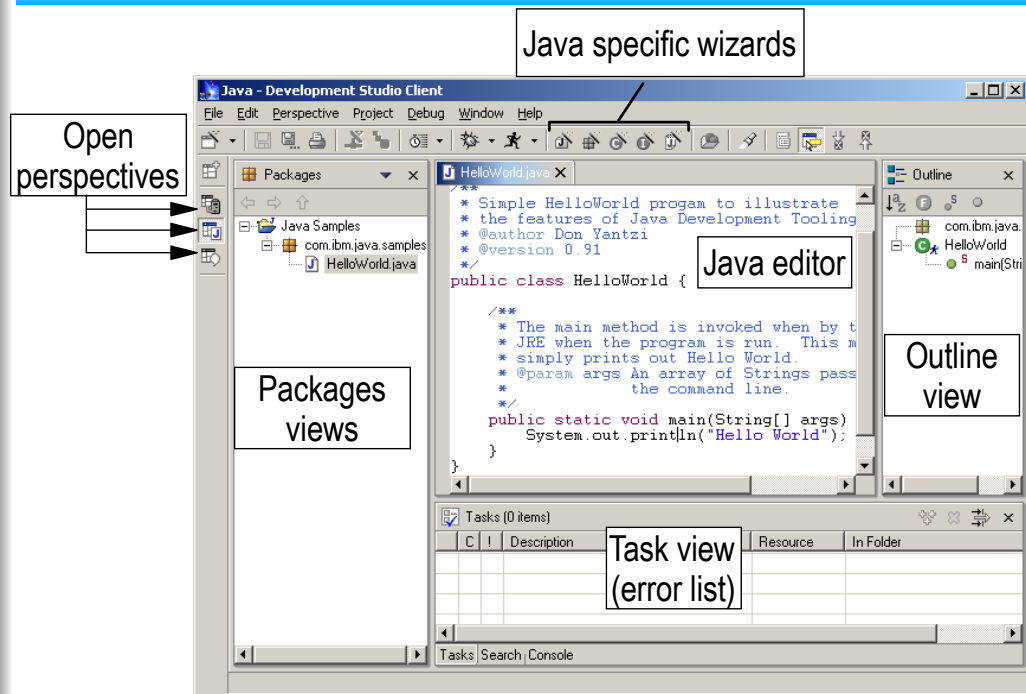
IBM @server. For the next generation of e-business.

► This slide shows how to open any perspective in the workbench.
► Select the Perspective -> Open -> Other menu option. This displays a list of all available perspectives. Select the desired perspective and click OK.
► Depending on the user preferences the perspective will either open in the same workbench window or a new workbench window.

Java Perspective

Java specific wizards

Open perspectives

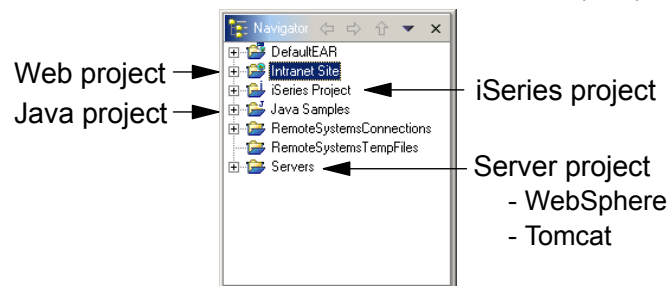Java editor

Outline view

Packages views

Task view (error list)

IBM @server. For the next generation of e-business.

Eclipse

▸ This slide shows 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.

Projects

▶ **Projects**

- **Highest level of organization for resources**
- **Contains files and folders**
- **Projects have**
  - ▶ A type
    - ✓ Java, iSeries, Server, Web, Simple (generic)
  - ▶ Properties
  - ▶ Associated builders
    - ✓ Builders know how to convert source artifacts (files) to executables

Web project → Navigator
Java project →

iSeries project

Server project
- WebSphere
- Tomcat

IBM *@*server. For the next generation of e-business.

▶ 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 Project Wizard

iSeries AD, IBM Toronto

- ▸ This slide shows 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.
- ▸ The user 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). The user can right click in the packages view and select New -> Java Project ... (as shown in the screen shot) or they could click the new Java Project icon in the toolbar or select File -> New -> Project...
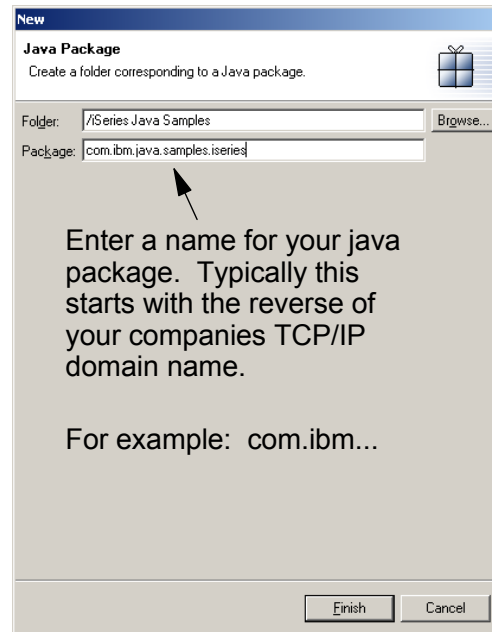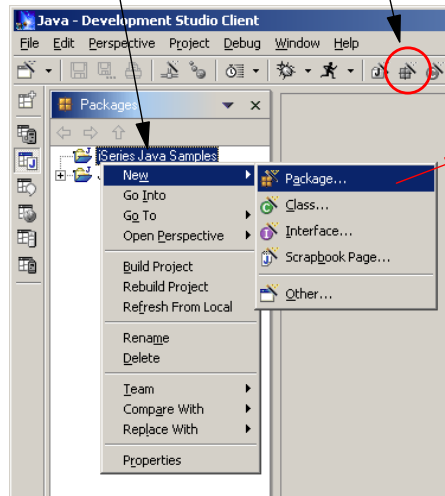
Creating a Java Package

iSeries AD, IBM Toronto

IBM @server. For the next generation of e-business.

- ▸ This slide shows the new Java Package wizard. The wizard requires the target folder (Java Project or source folder) and package name.
- ▸ The folder can be prefilled by selecting the project or source folder in the workspace before launching the wizard.
- ▸ The package name will map to folders in the workspace with each segment (separated by '.') mapping to an individual folder.
- ▸ For example com.ibm.iseries -> com/ibm/iseries
- ▸ Again there are many different ways to launch the new Java Package wizard.
- ▸ Source folders are covered later in this presentation.

Creating a Java Class

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 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

New

Java Class
Create a new Java class.

Folder: /iSeries Java Samples   Browse...
Package: com.ibm.java.samples.iseries   Browse...
☐ Enclosing type: Browse...

Name: JDBCSample
Access modifiers: ⦿ public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object   Browse...
Extended interfaces: Add...
                                     Remove

Which method stubs would you like to create?
☑ public static void main(String[] args)
☐ Constructors from superclass
☐ Inherited abstract methods

Finish   Cancel

IBM @server. For the next generation of e-business.
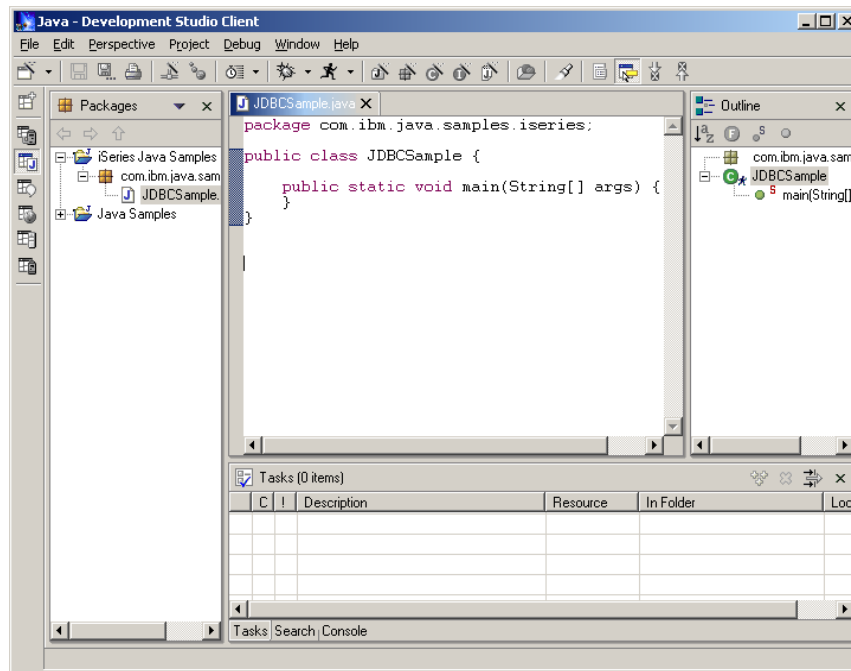
Eclipse

- ▸ This slide shows the new Java Class wizard. This wizard will generate the structure for the new Java class allowing the developer to focus on writing the logic!
- ▸ Selecting the target package in the workspace before launching the wizard will prefill the folder and package fields in the wizard.
- ▸ To generate an inner class, select the "Enclosing type" checkbox and enter the name of the class in which the inner class is to be created (or click the Browse button to select it from a list of classes.)
- ▸ The three checkboxes at the bottom will generate (if selected):
- ▸ The public static void main(String[] args) method for running this class as a standalone application
- ▸ Constructors with the same parameters as constructors in the super class. By default these generated constructors will just call the super class's equivalent constructor.
- ▸ Inherited abstract methods: These are methods that are defined in the interfaces which this class implements or methods marked as abstract in super classes.

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

► ## The Java Editor

IBM @server. For the next generation of e-business.

Eclipse

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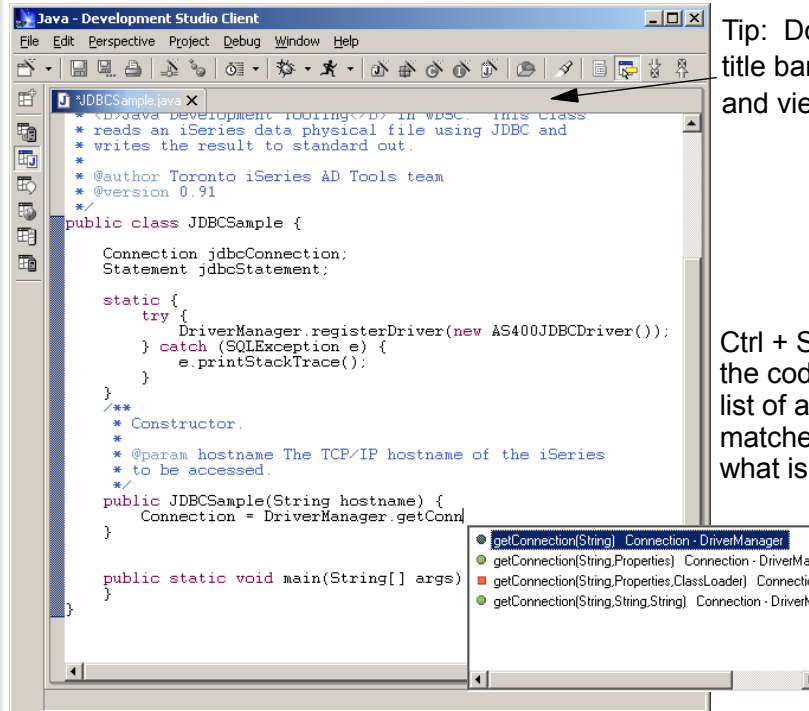► The next few slides will cover some of the additional features of the workbench Java editor.

Code Assist - Java Code

Tip: Double click on title bar to make editors and views full screen

Ctrl + Space brings up the code assist with a list of all possible matches based on what is already typed

- ▸ 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 subsetted 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 subsetted.
- ▸ 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.

Code Assist - JavaDoc

Code assist is also available in JavaDoc comments for:
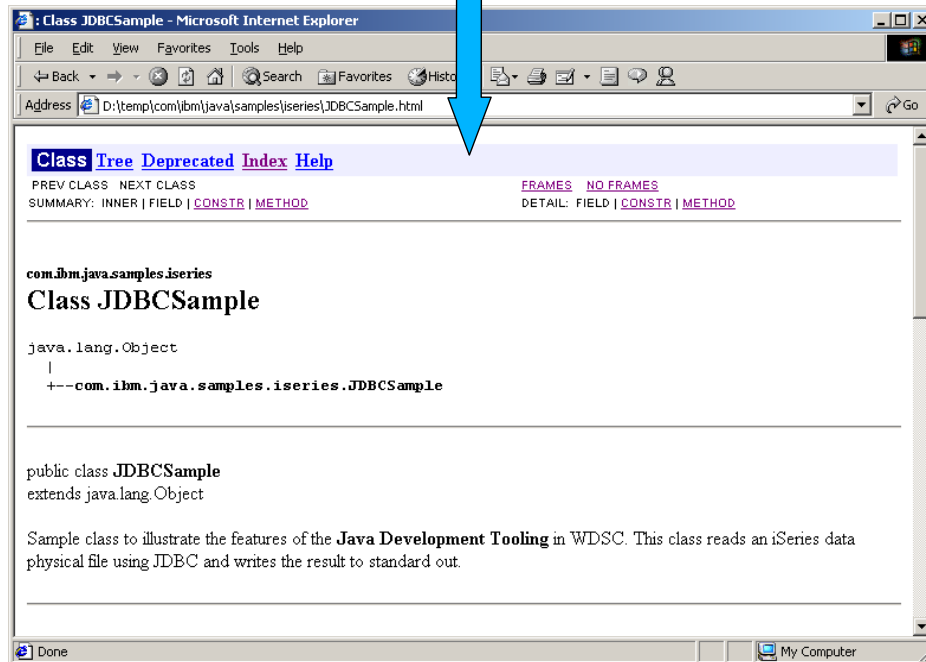- HTML
  - ✓ ...
  - ✓ <i> ... </i>
  - ✓ ...
- JavaDoc keywords
  - ✓ @param
  - ✓ @return
  - ✓ ...

IBM @server. For the next generation of e-business.

► 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 subsetted by what is already typed.
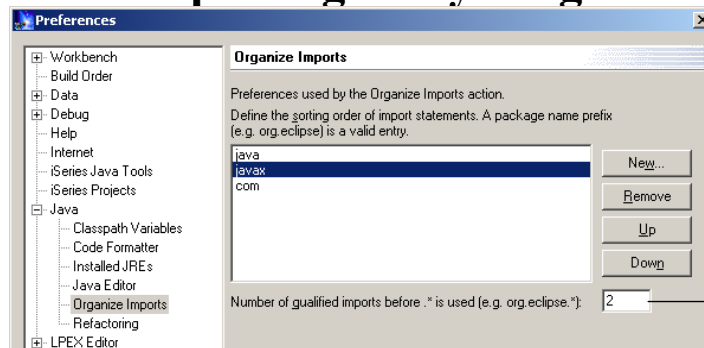
JavaDoc Reference

▶ This slide shows an example of generated Java Documentation (JavaDoc) from the previous slide.
▶ Generating JavaDoc is currently not integrated into the Workbench (next release) so the user needs to export the .java source files and run the javadoc command shipped as part of the Java SDK.

Managing Your Imports

► **First setup preferences**
- **Window -> Preferences**
- **Specify order imports should be listed**
  - ► For example: java, then javax, then com
  - ► Add your own!!!
- **Specify how many qualified imports to use before importing everything**

Preferences		✕
⊞ Workbench	**Organize Imports**	
⋯ Build Order		
⊞ Data	Preferences used by the Organize Imports action.	
⊞ Debug	Define the sorting order of import statements. A package name prefix (e.g. org.eclipse) is a valid entry.	
⋯ Help		
⋯ Internet	java	New...
⋯ iSeries Java Tools	javax	
⋯ iSeries Projects	com	Remove
⊟ Java		Up
⋯ Classpath Variables		Down
⋯ Code Formatter		
⋯ Installed JREs		
⋯ Java Editor	Number of qualified imports before .* is used (e.g. org.eclipse.*):	2
⋯ Organize Imports		
⋯ Refactoring		
⊞ LPEX Editor		

See next slide ➡

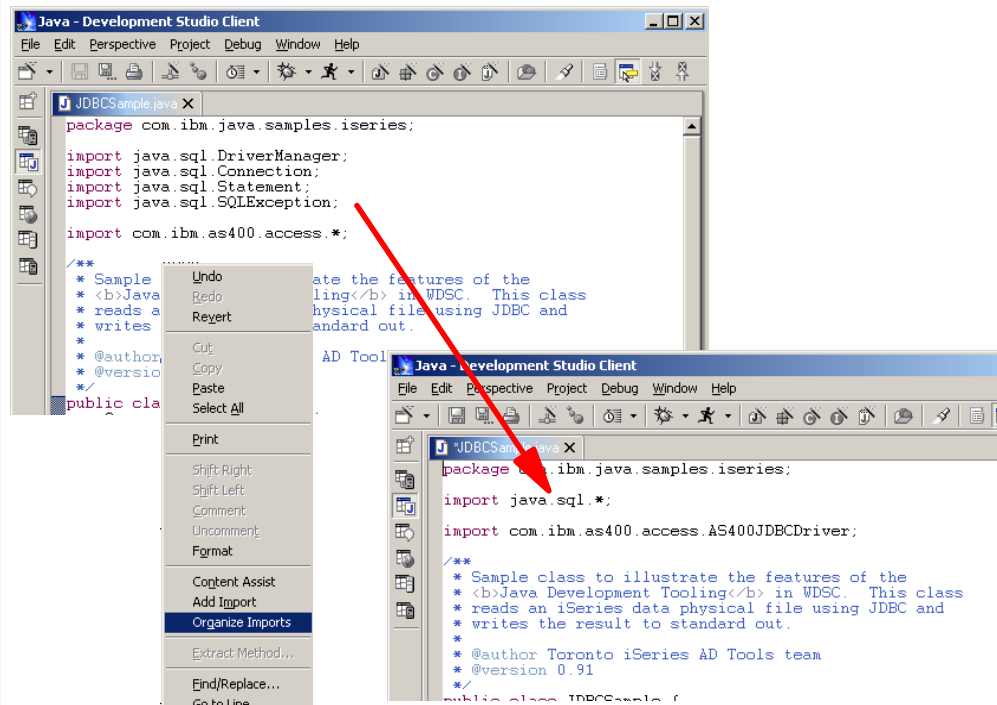**IBM @server.** For the next generation of e-business.

► Java tools can also be configured to manage the import statements in Java classes.
► The first thing the user should do is open the Preferences dialog and set the number of qualified imports that they want to have before the import is changed to a global import (i.e. java.sql.*;) By default this value is set to 99. A value of 2 or 3 is recommended.
► The user can also specify in which order the Java tooling should list imports.

Managing Your Imports

iSeries AD, IBM Toronto

▶ Once the user's organize imports preferences are set they can use the right click "Organize Imports" action from the Java editor. This will format the import statements in the class according to the preferences

Managing Your Imports

To import a class simply highlight the class name and select "Add Import" from the popup menu.

Tip: Double clicking on a class name or variable will highlight it.

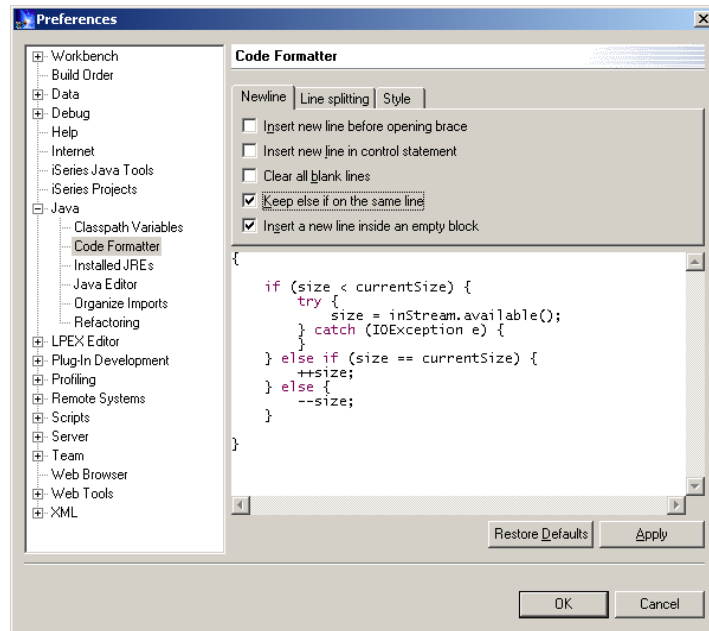Error in code because the class AS400 has not been imported yet.

IBM @server. For the next generation of e-business.
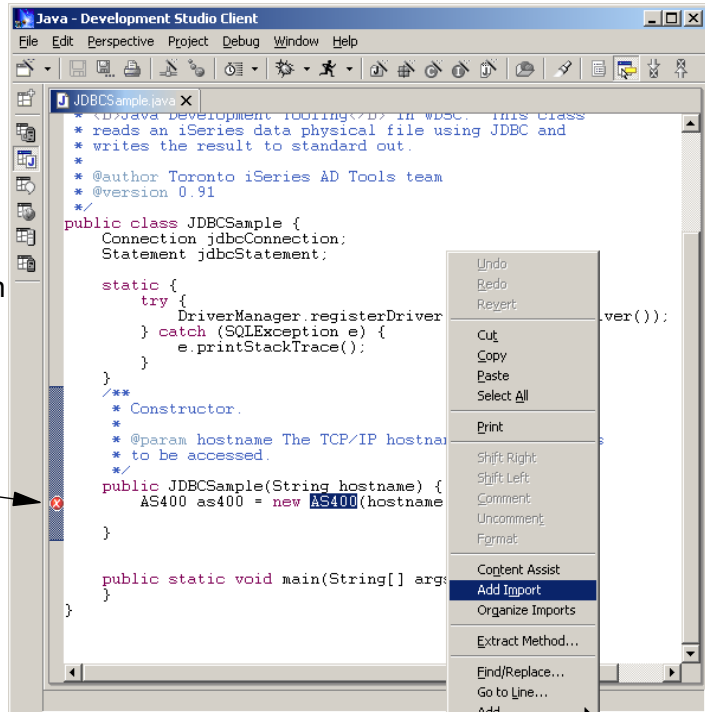
▸ Another handy feature of the Java editor which is to automatically add the import statement for a class.
▸ In this screen shot and error is shown on the one line because the class AS400 has not been imported yet. To automatically import this class:
▸ Highlight the class name in the editor
▸ Select "Add Import" from the right click popup menu
▸ This will add the import to the class according to the user's organize import preferences. For example: If two classes had already been specifically imported from the com.ibm.as400.access package and the user's preference was to globally import after 2 classes, then this action would cause those first two imports to be removed and a global import to be added because the AS400 class is also part of this package.

Formatting Your Code

- The Java Editor can automatically format your code however you would like
  - ► Maximum line length
  - ► { } preferences
    - ✓ New lines
    - ✓ Same line as code
  - ► ...

1. First set your preferences in the preferences dialog

   Then ...

Preferences

- Workbench
  - Build Order
- Data
- Debug
  - Help
  - Internet
  - iSeries Java Tools
  - iSeries Projects
- Java
  - Classpath Variables
  - Code Formatter
  - Installed JREs
  - Java Editor
  - Organize Imports
  - Refactoring
- LPEX Editor
- Plug-In Development
- Profiling
- Remote Systems
- Scripts
- Server
- Team
  - Web Browser
- Web Tools
- XML

Code Formatter

Newline | Line splitting | Style

- ☐ Insert new line before opening brace
- ☐ Insert new line in control statement
- ☐ Clear all blank lines
- ☑ Keep else if on the same line
- ☑ Insert a new line inside an empty block

```
{
    if (size < currentSize) {
        try {
            size = inStream.available();
        } catch (IOException e) {
        }
    } else if (size == currentSize) {
        ++size;
    } else {
        --size;
    }
}
```

Restore Defaults | Apply

OK | Cancel

IBM @server. For the next generation of e-business.
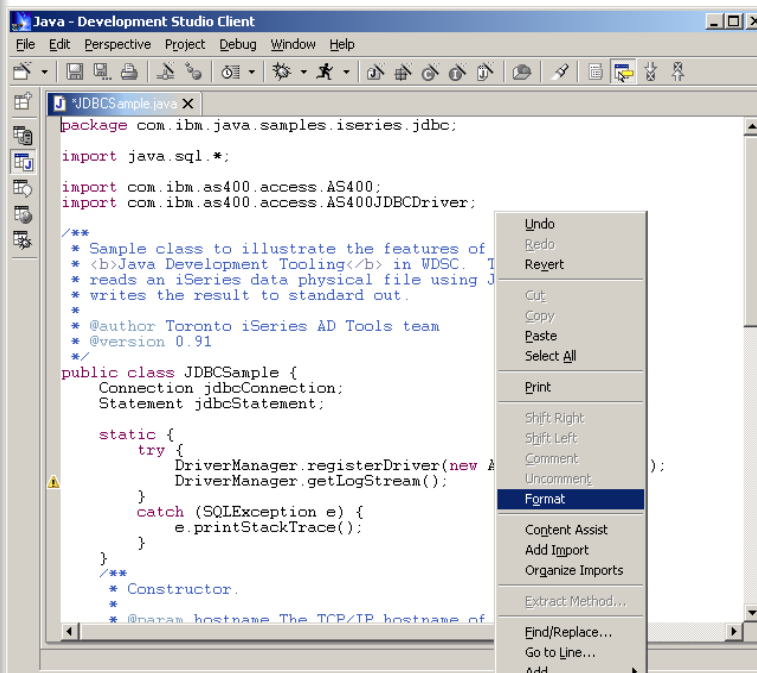
Eclipse

- ► There is another Java tooling preference that allows the user to specify how they would like their code formatted. The user can specify things like the maximum line length, whether { } brackets should start on a new line or not, tabbing, ...
- ► These preferences do not change what the user types in the editor until the format action is explicitly invoked as shown on the next slide.

Formatting Your Code

Java - Development Studio Client

File Edit Perspective Project Debug Window Help

*JDBCSample.java

```
package com.ibm.java.samples.iseries.jdbc;

import java.sql.*;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400JDBCDriver;

/**
 * Sample class to illustrate the features of
 * <b>Java Development Tooling</b> in WDSC.  T
 * reads an iSeries data physical file using J
 * writes the result to standard out.
 *
 * @author Toronto iSeries AD Tools team
 * @version 0.91
 */
public class JDBCSample {
    Connection jdbcConnection;
    Statement jdbcStatement;

    static {
        try {
            DriverManager.registerDriver(new A          );
            DriverManager.getLogStream();
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
    }
    /**
     * Constructor.
     *
     * @param hostname The TCP/IP hostname of
```

Undo
Redo
Revert
Cut
Copy
Paste
Select All
Print
Shift Right
Shift Left
Comment
Uncomment
Format
Content Assist
Add Import
Organize Imports
Extract Method...
Find/Replace...
Go to Line...
Add ▶

2. Right click in the editor and select "Format"

VOILA!

IBM @server. For the next generation of e-business.

Eclipse

▸ The "Format" popup menu action formats the current file as outlined in the user's Java tooling code formatter preferences.

Other Misc. Functions

- Editor also lets you manipulate blocks of code at a time
  - ▶ Shift right
  - ▶ Shift left
  - ▶ Comment
  - ▶ Uncomment
- Select the block of code, right click and select the required function from the popup menu

IBM @server. For the next generation of e-business.
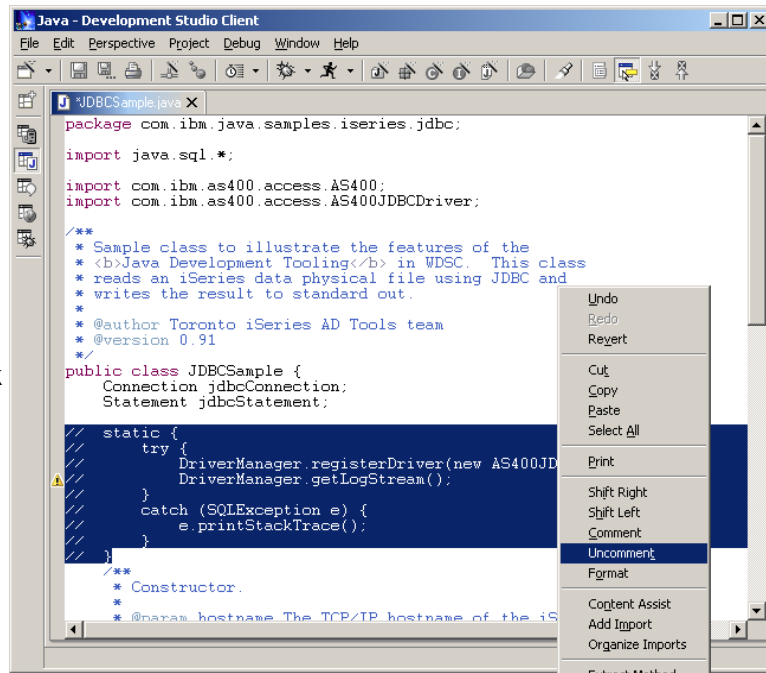
- ▶ This slide just shows some other misc. Java editor functions for handling blocks of code
- ▶ Although fairly trivial, these functions can be quite useful. For example:
- ▶ To add a try ... catch around a block of code, select those lines and "shift right" so they are properly indented
- ▶ To temporarily remove a block of code, use the "Comment" action to temporarily comment out those lines

► **Nominees For Best Supporting View**

IBM @server. For the next generation of e-business.

► Within the workbench views can be used to greatly increase productivity when used in conjunction with the editor. The Java tooling uses the following views which are outlined on the following 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

▶ **Navigator View**

- **The Eclipse tree view for managing _all_ resources in the workspace**
- **Shows Java classes in their package directories**
  - ▶ This takes a lot of space in the tree view
  - ▶ For example: com/ibm/java/samples

▶ **Packages View**

- **Provides central place to manage your _Java_ resources**
- **Shows only Java related projects**
  - ▶ This includes Web projects!
- **Packages shown as single entry in tree view**

IBM @server. For the next generation of e-business.

Eclipse

▸ The Navigator view is the standard workbench view for managing (view, renaming, copying, deleting, ...) resources in the workspace. There are a few things that make this view difficult to use when working with Java resources:

▸ The view shows all projects in the workspace whether they are Java or not

▸ 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.

Packages View

Projects

Packages

Source files
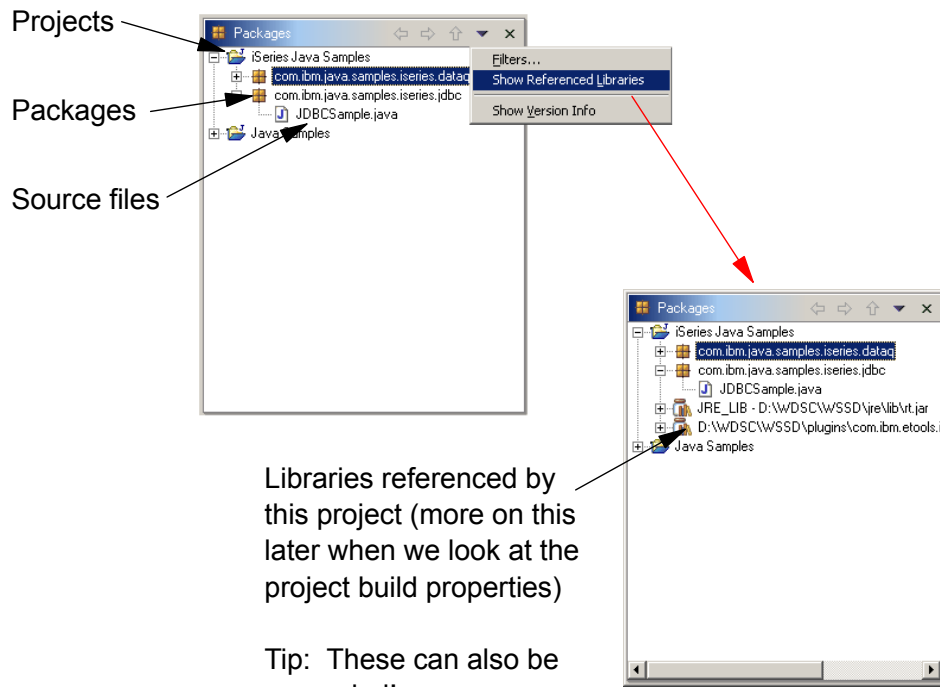
Libraries referenced by this project (more on this later when we look at the project build properties)

Tip: These can also be expanded!

Eclipse

IBM @server. For the next generation of e-business.

▸ The packages view will also let the user see any referenced libraries directly in the tree view.
▸ Reference libraries are setup in the properties for the Java project and will be covered later in this presentation.

Outline View

► Java Outline View

- **Works in concert w/ Java Editor**
- **Shows imports, fields, methods**
  - ► Can subset what is shown
  - ► Can sort by name
- **Can show only selected member in editor**
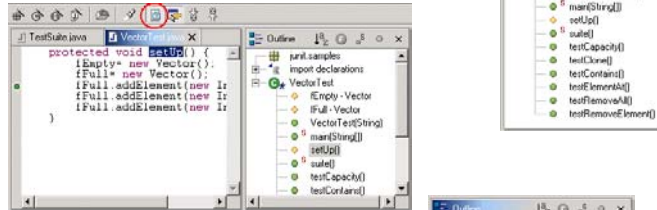
- **Can auto-add imports**
- **Can auto-gen getter and setter methods**
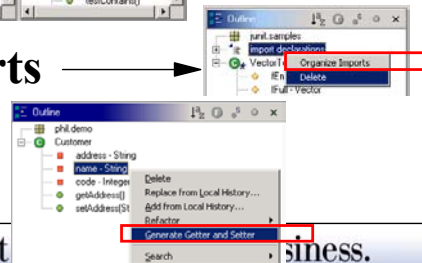
IBM @server. For the next ... siness.

- ► The outline view shows the structure of the Java class currently being edited.
- ► Its main function is to provide the user 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 will automatically position the editor to the corresponding source code.
- ► The outline view also provides many additional actions. These are available by right clicking on the different items (classes, methods, fields) in the outline view. For example:
- ► Right clicking on a field in the outline view will give you the option to generate standard getter and setter methods for the field.
- ► To remove a method simply right click on the method in the outline view and select "Delete". The method and its corresponding code will be 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

WDSc: Java Tools

► **Java Outline View**

- **Works in concert w/ Java Editor**
- **Shows imports, fields, methods**
  - ► Can subset what is shown
  - ► Can sort by name
- **Can restrict to selected member only in editor**

- **Can auto-add imports**
- **Can auto-gen getter and setter methods**

IBM *e*server. For the next generation of e-business.
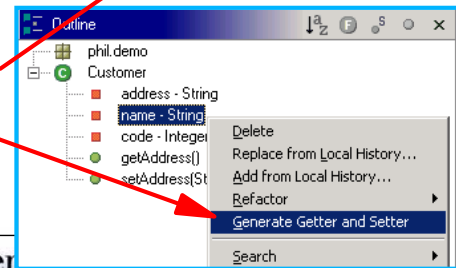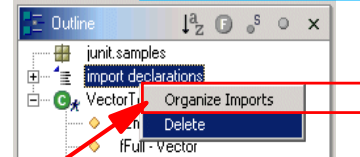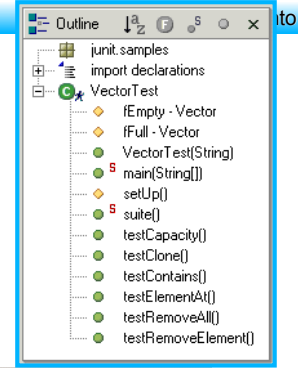
Hierarchy View

► **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

IBM @server. For the next generation of e-business.

Eclipse

► 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 the user can easily 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.

Hierarchy View

Supertype hierarchy

Type hierarchy

Subtype hierarchy

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

Currently showing all methods in AbstractList class

Types

- Object
  - AbstractCollection
    - AbstractList
      - Vector
        - AttributeNameEnum
        - GeneralNames
        - LinkVector
        - PcmlDataVector
        - Stack
        - TraceThread

AbstractList   1 2 3 4 5

- modCount
- AbstractList()
- add(int, Object)
- add(Object)
- addAll(int, Collection)
- clear()
- equals(Object)
- get(int)
- hashCode()
- indexOf(Object)
- iterator()
- lastIndexOf(Object)
- listIterator()
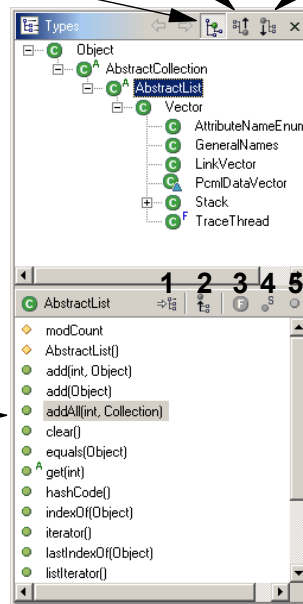
1. Shows selected members in hierarchy at all locations where they are declared.

✓ Lets you see where members are declared and overridden.

2. Show / hide inherited members

3. Show / hide fields

4. Show / hide static members

5. Show / hide nonpublic members

IBM @server. For the next generation of e-business.

- ► Here is a screen show of the hierarchy view with descriptions for the various features of the view.
- ► The first option (show selected members in hierarchy) will let 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 the user to quickly see where the method has been overridden

Tasks View

► Tasks view

- **Shows:**
  1. All compile errors, warnings and information associated with resources in the workspace
  2. 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, RPG, ...
- **Provides ability to filter by**
  - ► 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

IBM @server. For the next generation of e-business.

- ► The tasks view is a standard workbench view showing all errors, warnings and user defined tasks in the workspace.
- ► Because it shows contains 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 / packages view.

Tasks View

Project and folder(s) where the resource is located

Resource where the problem / task occurs

Global task
➡ no resources is specified

Filter

Warning

Error

Completed task

	C	!	Description	Resource	In Folder	Location
			Finish Java Tooling Presentations			
			The method getLogStream() from the type java....	JDBCSample.j...	iSeries Java Samples/com/ibm/ja...	line 27 in J...
			AS400 cannot be resolved or is not a type	JDBCSample.j...	iSeries Java Samples/com/ibm/ja...	line 39 in J...
			Implement main method	JDBCSample.j...	iSeries Java Samples/com/ibm/ja...	line 45

Tasks (4 items)

Tip: Double click on the task / problem to open the resource in the editor at the correct line!

IBM @server. For the next generation of e-business.

▶ This slide shows a screen shot of the tasks view with descriptions for all the features / columns.

► **Compiling**

IBM *e*server. For the next generation of e-business.

Eclipse

Compiling Your Java Class

► **Done!**

- **Everytime you save your file it is automatically compiled!**
- **Manually compiling**
  - ► Right click on the project, package or class and select either:
    - ✓ Build Project
      - ✓ Incremental build
      - ✓ Builds only resources that have changed since the last build
    - ✓ Rebuild Project
      - ✓ Full build
      - ✓ Discards previous build state and rebuilds everything

IBM @server. For the next generation of e-business.
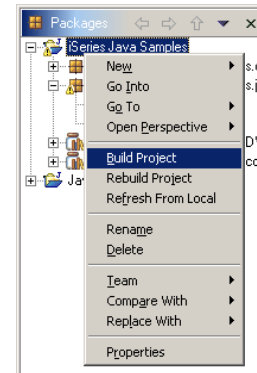
► When a user saves a Java class (.java file) it is automatically compiled and all errors / warnings are shown in the tasks view. However we can also force a recompile using either the build or rebuild project actions.

► **Running**

IBM @server. For the next generation of e-business.

Eclipse

Running Your Java Class

► How you run your class depends on what it is

- **Standalone Java application**
  - ► Class implements method:
    - ✓ public static void main(String[] args)
  - ► Run locally using
    - ✓ JDK shipped with the workbench
    - ✓ Any locally installed JDK
  - ► Run remotely using iSeries extensions
- **Servlet**
  - ► Class extends javax.servlet.http.HttpServlet
  - ► Servlet can be run in
    - ✓ Local WebSphere Test Environment
    - ✓ Local Tomcat Test Environment
    - ✓ Local Tomcat Application Server
    - ✓ Remote WebSphere Application Server

IBM @server. For the next generation of e-business.

Eclipse

- ► How the user runs 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 PC. The user can specify which JDK to use in the Java project properties.
- ► 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 the 4 scenarios listed on this slide.

- ▶ This slide shows how to run the standalone JDBCSample 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.

Java Development Tools

► **Java Debug**
Not that you will need it!

IBM @server. For the next generation of e-business.

Eclipse
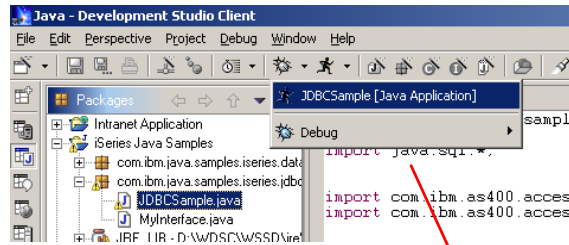
Debug

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 ...

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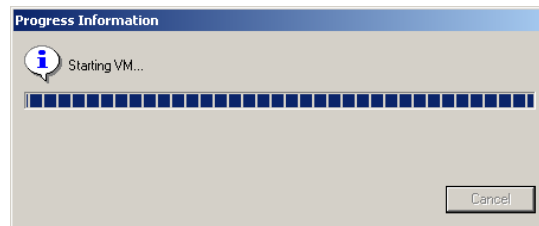
▶ To debug the Java application select the class in the packages view and select Debug -> Java Application from the Debug toolbar icon drop down menu.
▶ It is a good idea to set breakpoints in your source code before launching the application in debug mode. This will cause the debugger to stop at the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

Debug Perspective

iSeries AD, IBM Toronto
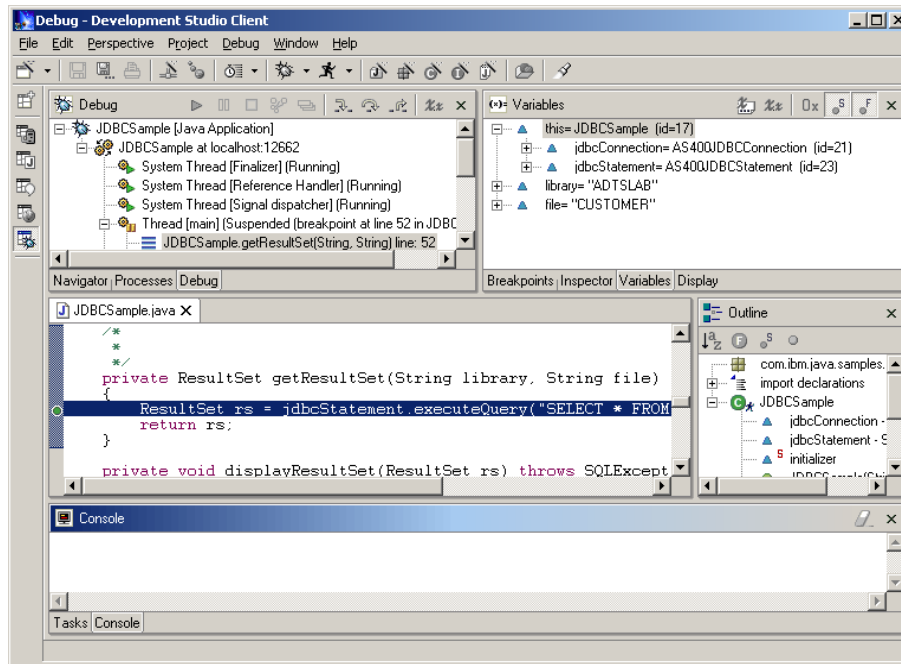
Details ——▶

IBM *e*server. For the next generation of e-business.

▶ This slide just shows a screen shot of the overall debug perspective. The next couple of slides provide more details.

Debug Perspective

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

Debug - Development Studio Client

File   Edit   Perspective   Project   Debug   Window   Help

Debug

JDBCSample [Java Application]
  JDBCSample at localhost:12662
    System Thread [Finalizer] (Running)
    System Thread [Reference Handler] (Running)
    System Thread [Signal dispatcher] (Running)
    Thread [main] (Suspended (breakpoint at line 52 in JDBC
      JDBCSample.getResultSet(String, String) line: 52

Navigator   Processes   Debug

JDBCSample.java ✕

```
    /*
     *
     */
    private ResultSet getResultSet(String libra
    {
        ResultSet rs = jdbcStatement.executeQue
        return rs;
    }

    private void displayResultSet(ResultSet rs)
```

Console

Tasks   Console

IBM @server. For the next generation of e-business.

- ► 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
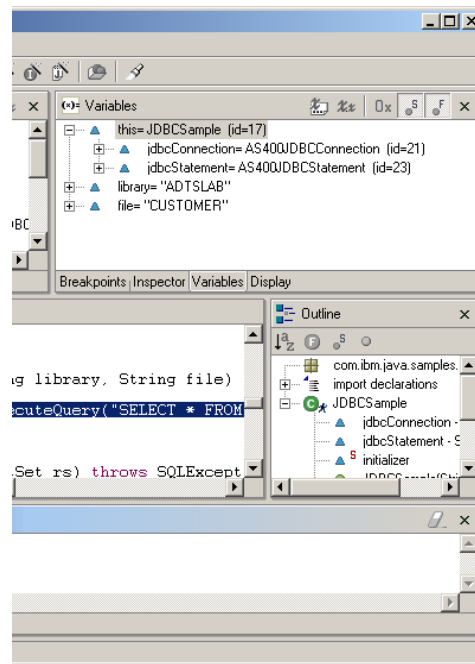
- ● Breakpoints View
  - ► Manage breakpoints in the workspace
  - ► Add Java exception breakpoints
    - ✓ Stop when a NullPointerException is thrown
    - ✓ This can be very handle in locating problems!
- ● Inspector view
  - ► Detailed view for inspecting variables
- ● Variables View
  - ► Shows all 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

IBM @server. For the next generation of e-business.

► 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).

► **Searching ....**
Now where did that go?

IBM *e*server. For the next generation of e-business.

Eclipse

Searching

► **Two main ways to search**

- **Global workspace searching**
  - ► Search the entire workspace
- **Context sensitive searching**
  - ► Scope to workspace or class hierarchy
  - ► Packages / Classes / Interfaces / Variables / Members
    - ✓ Find all places where referenced
    - ✓ Find all places where they are declared
      - ✓ In workspace, or
      - ✓ In class hierarchy
  - ► Interfaces
    - ✓ Find all implementors
- **Results are shown in the "Search" view**

IBM @server. For the next generation of e-business.

- ► As outlined on this slide, 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

iSeries AD, IBM Toronto
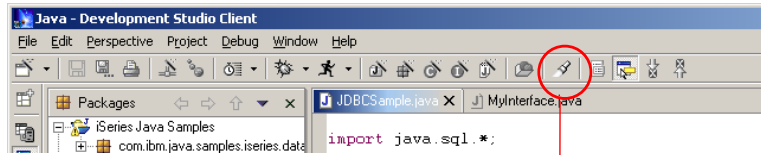
Java "aware" searching

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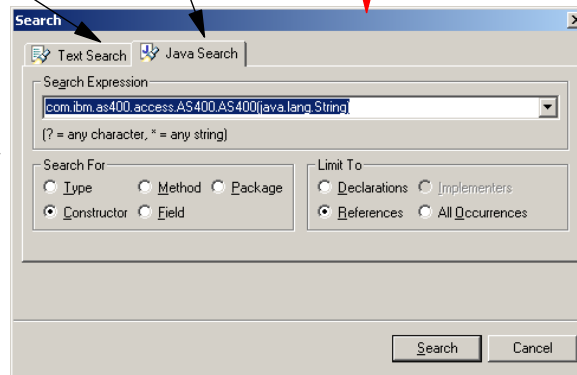
Find all types, methods, packages, constructors or fields that match the search expression.

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

IBM @server. For the next generation of e-business.

▶ This slide shows the "Java-aware" search dialog. 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.

IBM @server. For the next generation of e-business.
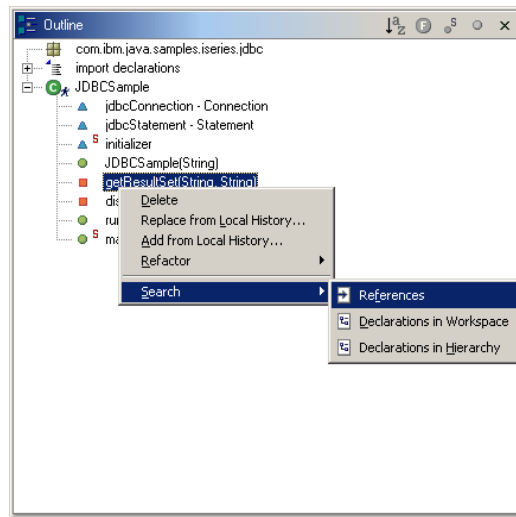
▸ Instead of having to type the class / method name and select all the radio buttons in the previous slide, the user can use the popup menu on fields / methods in the outline view to also search for references and declarations. This way the user will not be prompted.

► Java Build Path

- **Each Java project has a build path that 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**

IBM @server. For the next generation of e-business.

► 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

Properties for iSeries Java Samples

- Beaninfo Path
- Debugger Source Lookup
- Java Build Path
- Java Info
- JRE
- Launcher
- Project References
- Team

Java Build Path

Source | Projects | Libraries | Order

○ Use the project as source folder
○ Use source folders contained in the project

Create New Folder...
Add Existing Folders...
Remove

Build output folder:
/iSeries Java Samples   Browse...

■ 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

Packages
- Finance Application
  - Accounts Payable
    - com.ibm.acctspay
  - Accounts Recievable
  - Sales
- Intranet Application
- iSeries Java Samples
- Java Samples

IBM @server. For the next generation of e-business.
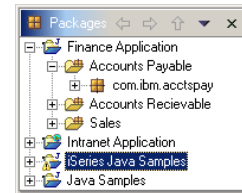
► In the Java project properties dialog select "Java Build Path". This displays a four page notebook. We will look at each page on the upcoming slides.
► The first page lets you specify whether or not this project uses source folders to organize Java packages. For large projects, source folders give you an extra level of organization for your Java code.

Java Build Path - Projects
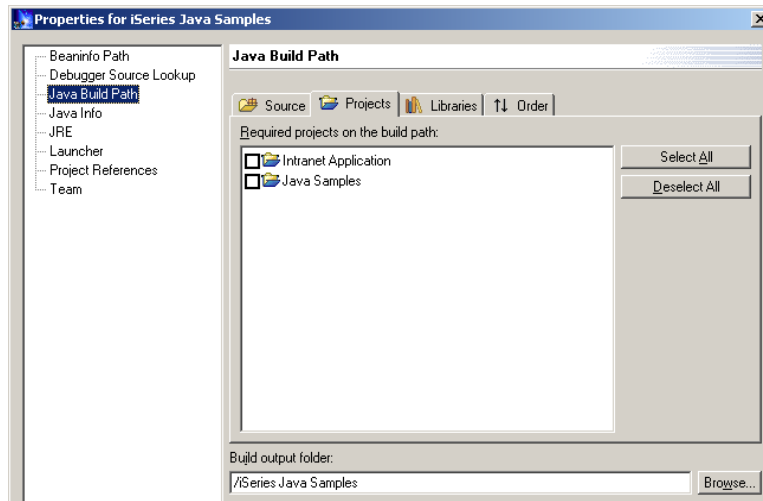
- **Projects**
  - ► Select all projects in the workspace that are required by this project
    - ✓ i.e. This project uses classes in the other project

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

Java Build Path - Libraries

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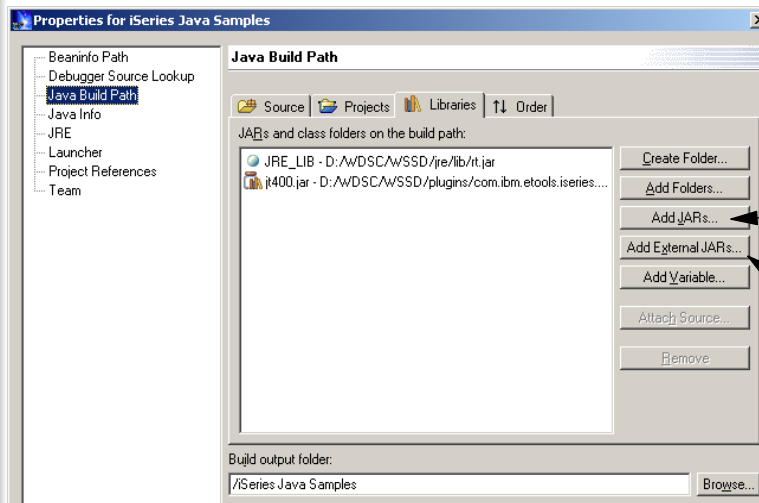
- **Libraries**
  - ► These are NOT OS/400 libraries!
  - ► Specify all JAR files required by your Java project
    - ✓ i.e. IBM Toolbox for Java
      d:\WDSC\WSSD\plugins\com.ibm.etools.iseries.toolbox\runtime\jt400.jar

IBM @server. For the next generation of e-business.

- ► 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

Properties for iSeries Java Samples

- Beaninfo Path
- Debugger Source Lookup
- Java Build Path
- Java Info
- JRE
- Launcher
- Project References
- Team

Java Build Path

Source | Projects | Libraries | Order

Build class path:

- JRE_LIB - D:/WDSC/WSSD/jre/lib/rt.jar
- jt400.jar - D:/WDSC/WSSD/plugins/com.ibm.etools.iseries.too
- /iSeries Java Samples

Up
Down

Build output folder:

/iSeries Java Samples   Browse...

- **Order**
  - ► Specify the order the projects and libraries are included in the build path
  - ► This is important if the different versions of the same class exists in to different entries

IBM @server. For the next generation of e-business.

► 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.
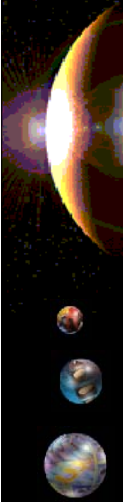
► **iSeries Additions**

IBM *@*server. For the next generation of e-business.

Eclipse

► So far this presentation has looked at the generic Java tooling provided by WSSDa, nothing specific to the iSeries platform. Now we will look at iSeries specific extensions to the Java tooling. This slide shows an overview of those extensions. The following slides will look at each one in more details.

Java Tools for iSeries

► iSeries extensions for Java

● Overview / Nutshell:

- ► Import and export from/to remote system
- ► Remote Compile and Run
- ► Remote Debug
- ► Program Call wizard
- ► Toolbox for Java built-in
  - ✓ As an Eclipse Java project (i.e.: easy to add to classpath of your project!)
  - ✓ Integrated help
  - ✓ Runnable samples
- ► Supplied Java-beans
  - ✓ Swing GUI beans (dspf-like functionality)
  - ✓ DFU beans (database access)
  - ✓ Object list beans (library, object, member, field, record lists)

IBM @server. For the next generation of e-business.

- ► Export and import to/from IFS, via 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
- ► Previously supplied Java beans: datafile updates (DFU), objects lists (PDM), and Swing formatting

► iSeries extensions for Java

● 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

Eclipse

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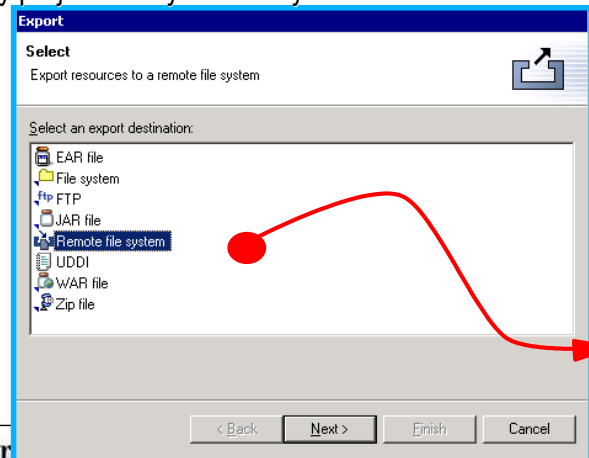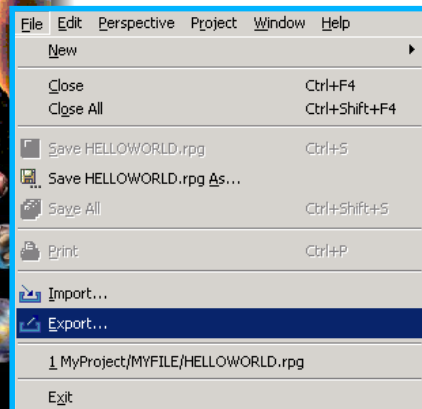
- ► The workbench allows the user to import and export your code (Java and non-Java code) from / to a variety of locations. To this list we added the ability to import from and export to a remote system (iSeries or non-iSeries).
- ► Like all the other import / export wizards, this ability is not restricted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

Java Tools for iSeries
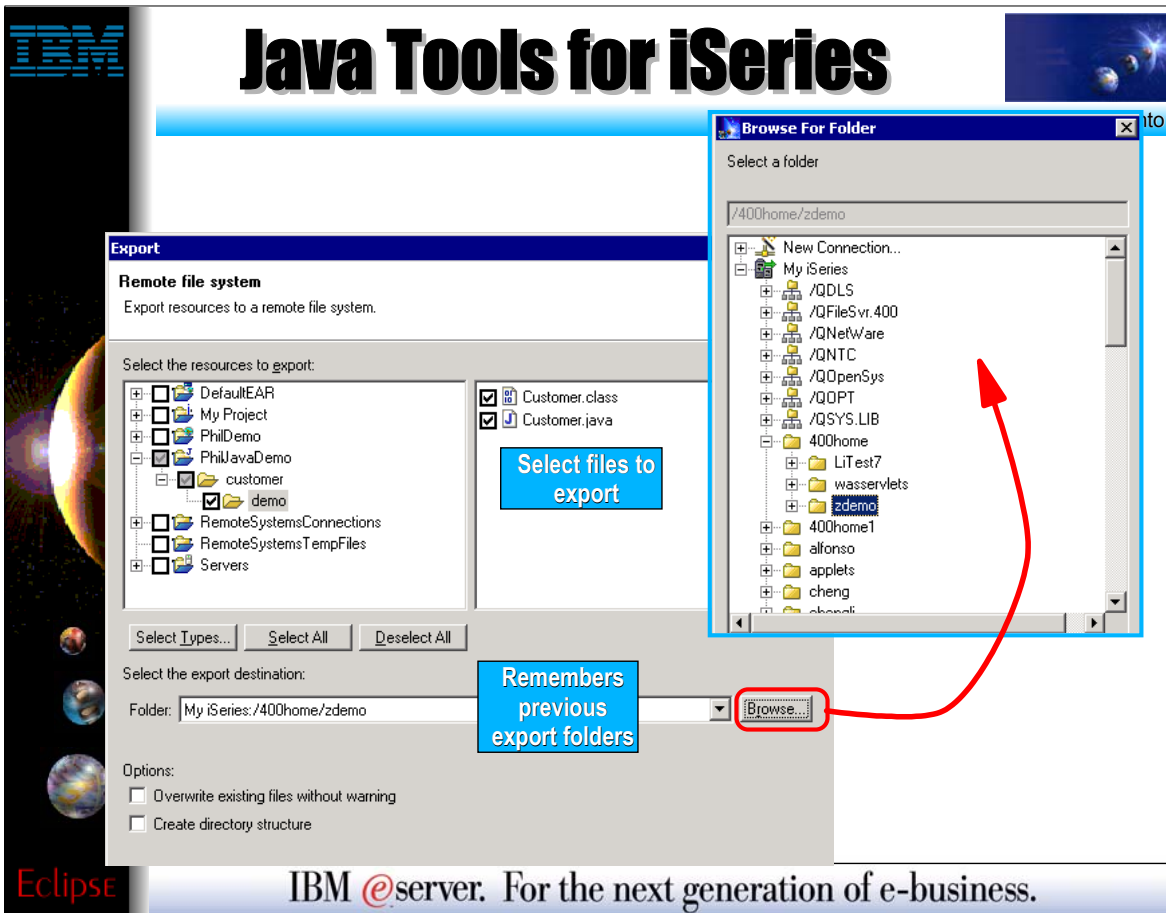
▸ This slide shows the next page of the export to remote file system wizard. Here the user specifies which files to export from their workspace and to which remote system and directory on that remote system.
▸ The import wizard has a similar page.

Java Tools for iSeries

iSeries AD, IBM Toronto

► **Remote Compile and Run**

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

IBM *e*server. For the next generation of e-business.
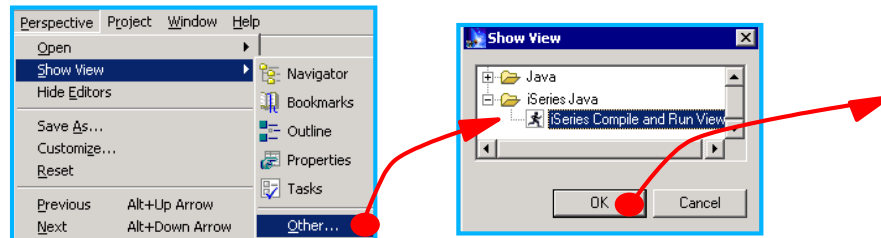
► This slide shows how to open the Remote Compile and Run View.
► Open the "iSeries Console and Run View" to natively compile (CRTJVAPGM), run and debug your Java applications remotely on the iSeries.
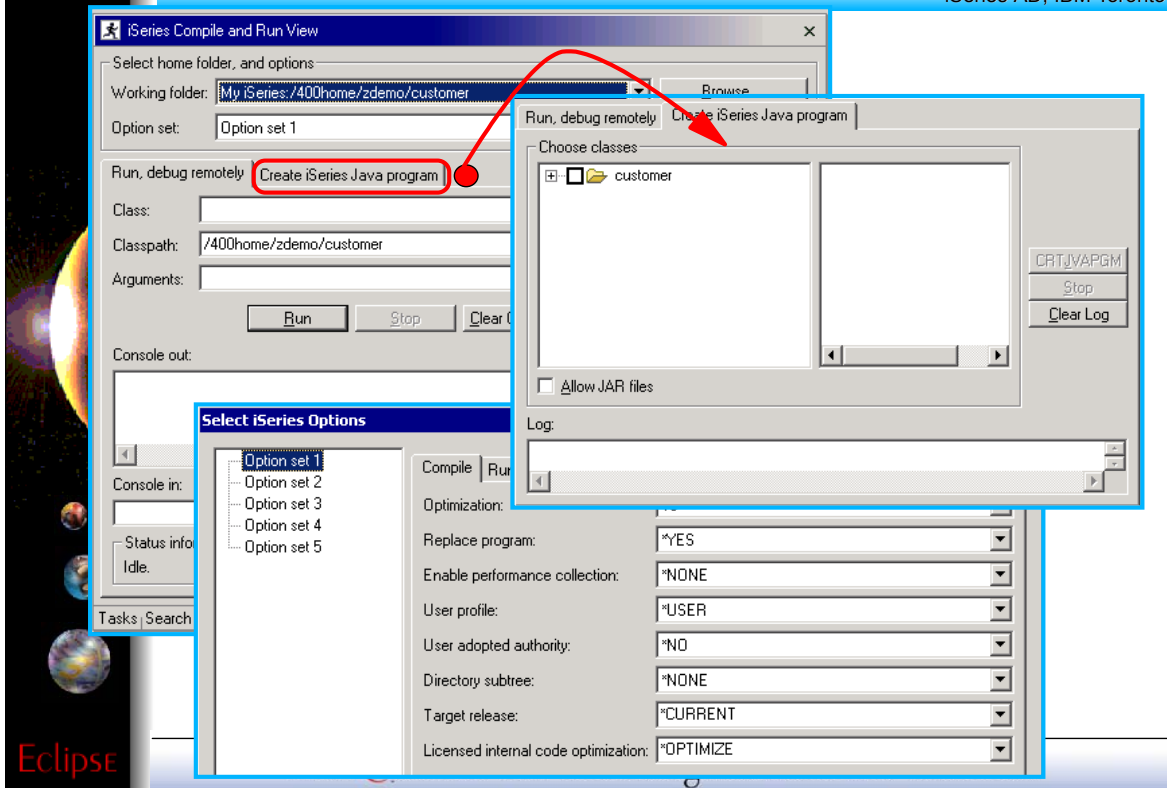
- At the top of the iSeries Compile and Run View the user needs to specify the remote system and directory (typically this is the same system and directory you specified on the export wizard.)
- Before running or compiling the Java class the user can setup their options. For the compile action these options are the parameters that are passed to the CRTJVAPGM command. For running these options we setup our library list and environment variables.
- On the Run, Debug remotely tab of the view the user can specify the Java class they 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 the user to view output from and send input to the remotely running Java application.
- Switching over the the Create iSeries Java Program tab allows the user 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.

Java Tools for iSeries

► **Remote Debug**
  ►Launches debugger for iSeries Java
    ✓Same debugger as RPG, CBL, CL, C, C++
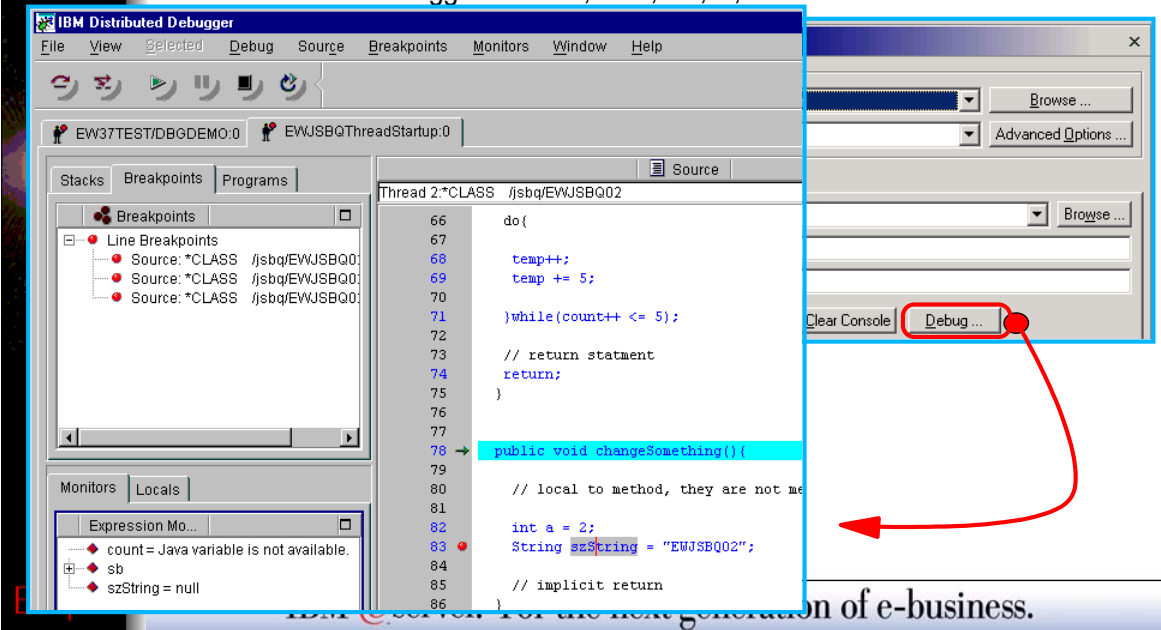
► Instead of running the Java application remotely the user can instead debug the Java application as it runs on the iSeries. Clicking the Debug button launches the IBM Distributed Debugger.

Java Tools for iSeries

✓ Program Call Wizard

- **Given signature of *PGM or *SRVPGM proc:**
  - ► Generate Java Bean to call it
- **Steps:**
  - ► 1. Select Java/Web/WebFacing project
  - ► 2. Select package
  - ► 3. Use File->New->Other...
    - ✓ ...iSeries Java Program Call Bean

IBM @server. For the next generation of e-business.

Eclipse
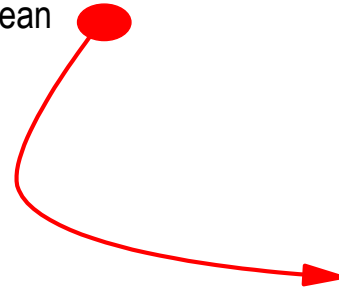
- ▸ Here we step through the very popular Program Call wizard, which has been significantly enhanced from its old VisualAge for Java days.
- ▸ Using this wizard is easy.
- ▸ First, within any view that lists projects, select a project to contain the generated Java code. You are allowed to select from 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 directions on the following slides.

Java Tools for iSeries

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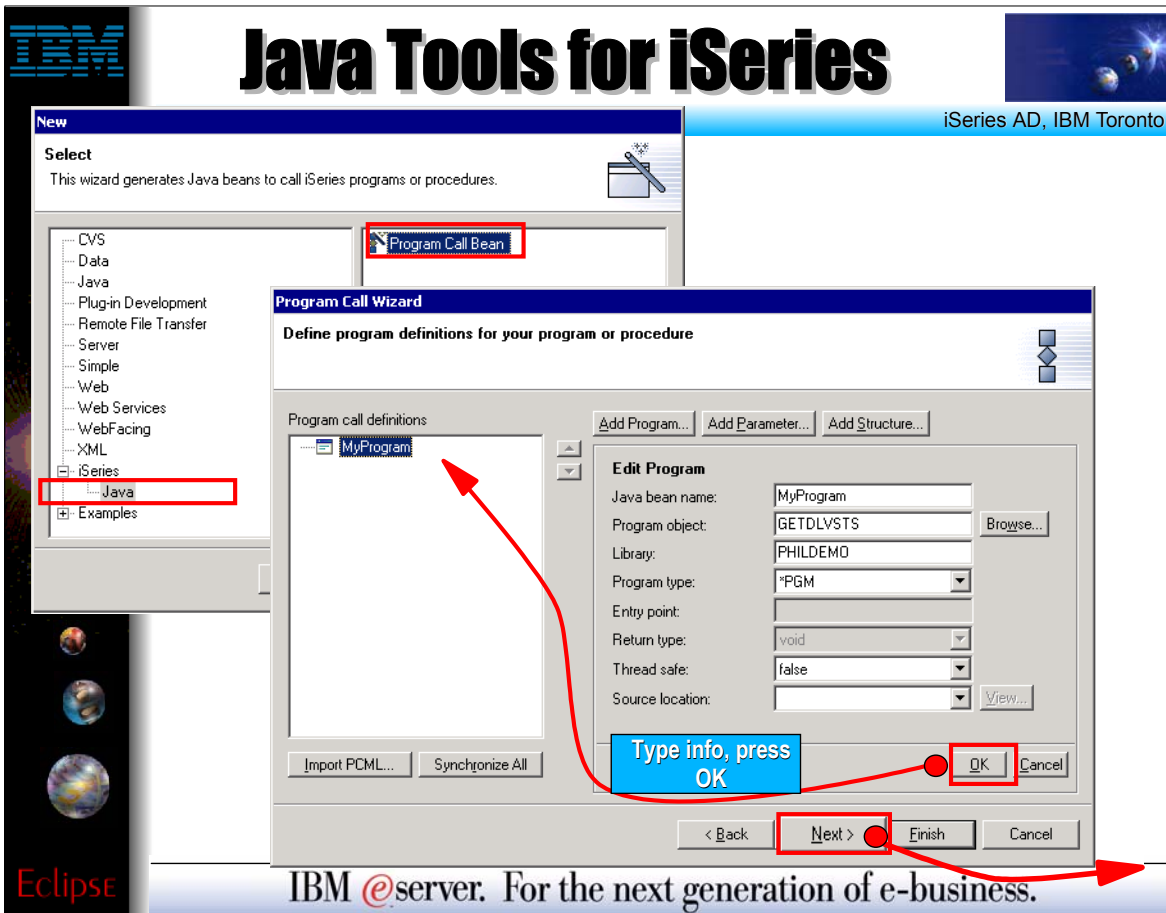
- ► On the File->New->Other wizard selection page, select the Program Call Bean wizard and press Next
- ► In the Program Call wizard, select the program or ILE procedure to be called from Java. This must be a non-interactive program or procedure

Enter Parameter Info

Define parameters

Define and name structure

Add Parameter

Parameter name:
Data type: character
Structure name:
Length:
Precision:
Count:
Usage: input & output
Initial value:

☐ Advanced...

Specify database reference field   Specify...
Show database field definition   Show...

> 0 for array

Add Structure

Structure name:
Count:
Usage: input & output
Output size:
Min. version:
Max. version:
Offset:
Offset from:

Be sure to press OK for each parameter / structure

Program call definitions
☐ MyProgram
  retCode
  nbrCodes
  dlvCodes

Builds up parameter list

Next >

IBM @server. For the next generation of e-business.

Eclipse

► 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, press Next to continue to the next page of the wizard

- This slide shows the page where the user can confirm where they 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

Java - Development Studio Client

File Edit Perspective Project Debug Window Help

Packages: PhilJavaDemo

- customer.demo
  - Customer.java
  - MyProgram.java
  - MyProgramResult.java
  - MyProgramServices.java
  - MyProgram.mpcml
  - MyProgram.pcml
- JRE_LIB - E:\WDSC\WSSD\
- ECLIPSE_HOME/plugins/con
- ECLIPSE_HOME/plugins/con
- ECLIPSE_HOME/plugins/con
- ECLIPSE_HOME/plugins/con

MyProgram.java

```java
package customer.demo;

import java.math.*;
import com.ibm.as400.access.*;
import com.ibm.as400.connector.as400.ProgramCallBean;
import com.ibm.as400.connector.as400.ProgramCallRecord
import com.ibm.as400.connector.as400.ProgramCallStruct

public class MyProgram extends ProgramCallBean
{

    public MyProgram()
    {
        super();
        setPcmlName("customer.demo.MyProgram");
        setProgramName("MyProgram");
    }


    public Integer getRetCode()
    {
```

Outline

- customer.demo
- import declarations
- MyProgram
  - MyProgram()
  - getRetCode()
  - getNbrCodes()
  - getDlvCodes(int)
  - invoke()
  - setConnectionData(AS400)
  - setConnectionData(String, String, String)
  - getAS400Object()
  - setLibraryList(String[])
  - disconnect()
  - setTraceEnabled(boolean)

Wizard adds required files to project classpath

Example using gen'd bean

```java
package customer.demo;
public class TestMyProgram
{
    public static void main(String[] args)
    {
        MyProgram test = new MyProgram();
        test.setConnectionData("MYSYSTEM", "MYUSERID", "MYPASSWORD");
        test.setLibraryList(new String[] {"PHILDEMO"});
        test.invoke();
        int nbrResults = test.getNbrCodes().intValue();
        System.out.println("Nbr Codes: " + nbrResults);
        for (int idx=0; idx<nbrResults; idx++)
        {
            System.out.println(" code " + idx + ": " +
                             test.getDlvCodes(idx).trim());
        }
    }
}
```

- ►**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

Eclipse

IBM @server. For the next generation of e-business.

► 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 rea 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)
    - ✓ Error checking based on data type, length, decimals
    - ✓ DDS-like validity checking: range or comparison
    - ✓ Editcode or editword formatting and masking(!)
    - ✓ Auto-advance
  - ► JFormattedLabel (smart label constant)
    - ✓ Editcode or editword formatting
  - ► JFormattedComboBox (smart dropdown)
    - ✓ Combo of JFormattedTextField and JFormattedLabels
  - ► JFormattedTable (smart multi-column list
    - ✓ A "subfile" or multi-column list box
    - ✓ Each cell is a JFormattedTextField or JFormattedLabel or JFormattedComboBox

IBM @server. For the next generation of e-business.

Eclipse

- ► Included with the iSeries extensions to the Java tooling are some GUI and non-GUI Java beans that can be used in applications developed by WDSc customers.
- ► This slide shows the GUI beans included with WDSc. 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

► 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

EQUITY ORDER ENTRY

Symbol:	IBM
Price:	$200.00
Quantity:	1,000
Account No.:	1234-56790
Password:	********
Contact Phone:	(555)111-2222

IBM @server. For the next generation of e-business.

► DFU Beans for iSeries (non GUI)

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

IBM @server. For the next generation of e-business.

Eclipse

► This slide lists some non-GUI beans that can be used in a Java application to assist the developer in reading records from and iSeries physical file. The ListManager retrieves lists of records for display in a GUI list, where as the FormManager can be used to retrieve a single record for display in a GUI form.

► DFU Beans Examples

- ● File->New->Other->Examples->iSeries
  - ► ListManager

ListManagerJFormattedTableSample

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 with a multiple record format logical file

Order: **DEF456**

Order Date: 1998-03-24

Customer Order: aaaaaaa

Customer: bbbbbbbbbbb

Ship Via: ccc

Ship To: ddddddd

Price: 55555.33

Goods: 6666

► FormManager

Eclipse

IBM @server. For the next generation of e-business.

► **List Beans for iSeries (non-GUI)**

- **For querying lists of libraries, objects, members, records or fields**
  - ► 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 attr (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
- **Four levels of detail possible per object**

IBM *e*server. For the next generation of e-business.

Eclipse

► This slide shows some additional non-GUI beans for retrieving lists of libraries, objects, members, records and fields from an iSeries.

► **Summary**

IBM @server. For the next generation of e-business.

Eclipse

Java Tools Not There

► **Not all VAJava function is available <u>yet</u>:**

  ✖ Visual Composition Editor
  ✖ DataBase Beans
    ✓ Actually, there are replacements! Just no visual metaphor
    ✓ See com.ibm.db.beans: DBSelect, DBProcedureCall, DBModify
    ✓ There is also a Database access wizard in Web Tools
  ✖ SQLJ support
  ✖ Stored Procedure Builder
  ✖ Live editing while debugging

► **For VAJava migration...**

  ► www.ibm.com/software/wsdd
    ✓ "VisualAge for Java coexistence with WebSphere Studio Application Developer"
    ✓ http://www7b.boulder.ibm.com/wsdd/library/techarticles/0110_searle/searle.html

IBM @**server.** For the next generation of e-business.

Eclipse

Summary

- ▶ **Java Development Tooling included in WDSC**
  - ● **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
- ▶ **Enjoy!**

IBM @server. For the next generation of e-business.

Resources

► **Eclipse.org**
- **Website for the open source Workbench**

► **WDSC**
- **ibm.com/software/ad/iseries/**

► **WebSphere Developer Domain**
- **ibm.com/websphere/developer**

► **IBM Developer Domain**
- **ibm.com/developer**

► **Java**
- **java.sun.com**

IBM *e*server. For the next generation of e-business.

Trademarks & Disclaimers

iSeries AD, IBM Toronto

Eclipse

IBM @server. For the next generation of e-business.