

IBM Software Group | Rational software

IBM Rational Functional Tester for Java

Recording an automated test script

Rational software

@business on demand.

© 2008 IBM Corporation
Converted to video May 22, 2015

The slide features a blue background with a circular pattern of dots. The IBM logo is in the top right corner. The title 'IBM Rational Functional Tester for Java' is in bold black text, followed by the subtitle 'Recording an automated test script' in italics. Below the title is the 'Rational software' logo. A horizontal bar with various icons (a person, a globe, a refresh icon, a network icon, and a play icon) is positioned below the logo. The text '@business on demand.' is in the bottom right, along with the copyright notice '© 2008 IBM Corporation' and 'Converted to video May 22, 2015'.

This module will provide an overview of recording an automated test script using IBM Rational® Functional Tester version 7.0. It will explain the steps a tester needs to take before recording the automated script.

Course topics

- General concepts
- Preparing the test environment
- Recording a test script
- Examining the recorded test script



The objective of this course is to cover the primary topics for generating an automated test script.

After a brief introduction to IBM Rational Functional Tester, you will go through the steps of creating a test project, which entails enabling the environment for testing, then configuring the application that will be tested. Next, you will learn about starting the recorder and recording a script, and, in the process, create a verification point and a data pool. After the script has been recorded, you will examine the script and review the object map created by functional tester, the verification point, the data pool, and the helper super class.

General concepts

- **Challenge: Test Windows, Web, Java™, Seibel and SAP applications**

- ▶ Test faster
- ▶ Test Smarter
- ▶ Test consistently
- ▶ Have test script maintenance
- ▶ Have test script scalability

- **Solution: IBM Rational Functional Tester for Java**

- ▶ Based on the Eclipse Platform
- ▶ Object Oriented Automation Testing Tool
- ▶ Supports Microsoft® Windows® XP, 2000, 2003 Server and Linux® Red Hat
- ▶ Supports Internet Explorer® and Mozilla



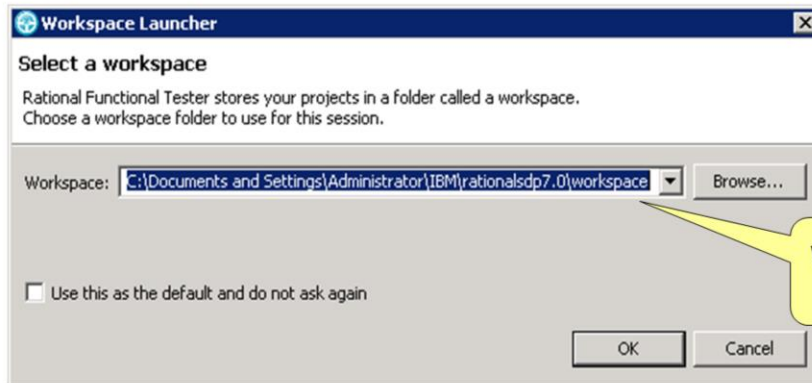
In today's IT world, software applications are increasingly more complex and are richer in features. The task of testing these applications can be daunting. A tester faces many challenges in testing these applications, and is continuously expected to test an application more quickly and precisely. The tests created should be consistent, scalable, and maintainable.

IBM Rational Functional Tester for Java is a test automation tool that addresses these challenges. It is based on the open source Eclipse Platform. Test scripts in Functional tester are created using Object Oriented methodologies. Functional Tester supports Microsoft Windows XP, Windows 2000, Windows 2003 Server, and Red Hat Linux. It also supports Internet Explorer and Mozilla Web browsers.

Preparing the test environment

▪ Creating a new functional test

- ▶ Programs > Software Development Platform > IBM Rational Functional Tester > Java
- ▶ Desktop Icon



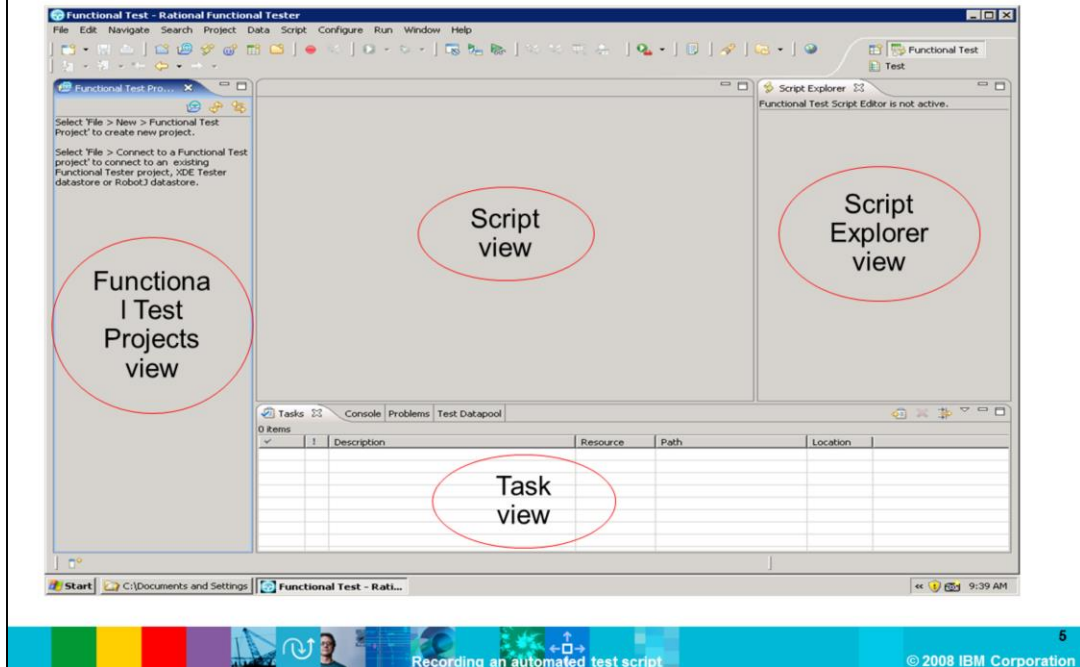
In Functional Tester, to start recording scripts against your applications, you must begin by creating a project, configuring your test environments, and configuring the application that is going to be tested.

If you accepted the defaults during installation, the program is located in the IBM Software Development Platform group.

Start Functional Tester from the start menu. Go to Programs, Software Development Platform, IBM Rational Functional Tester, Java. Otherwise, you can start the program from the alternate location you installed it to, or, you can start it from a desktop icon.

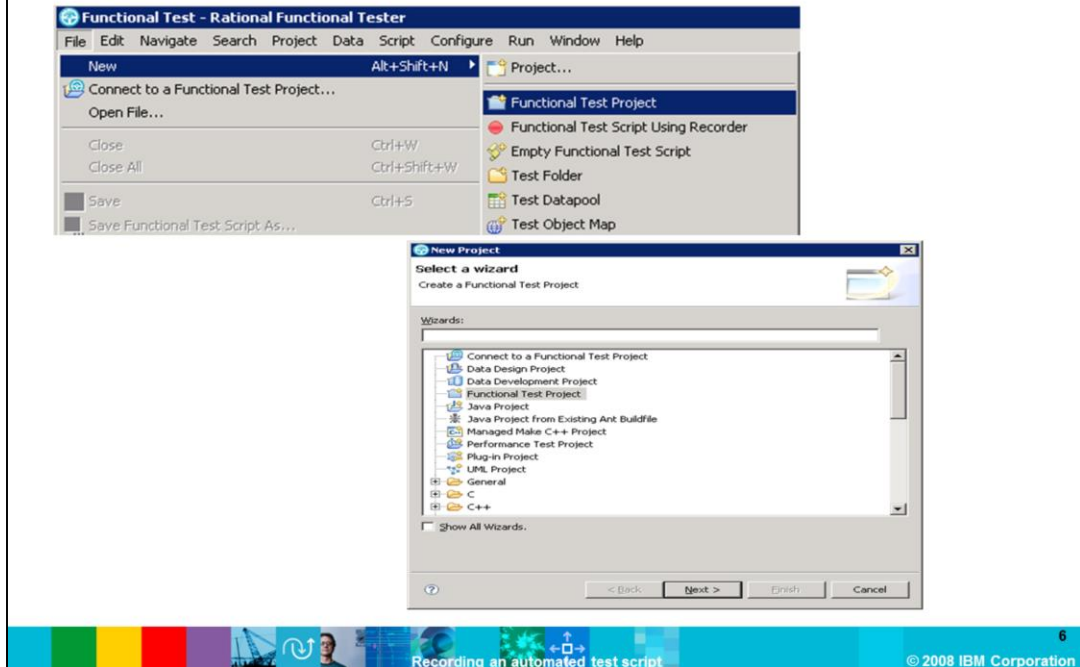
During startup, a location for a workspace needs to be specified in the Workspace Launcher window. By default, the location is set to the folder located at Documents and Settings > Administrator > IBM > rationalsdp7.0. Users have an option to change this to another location by clicking the Browse button. When finished, click OK.

Preparing the test environment



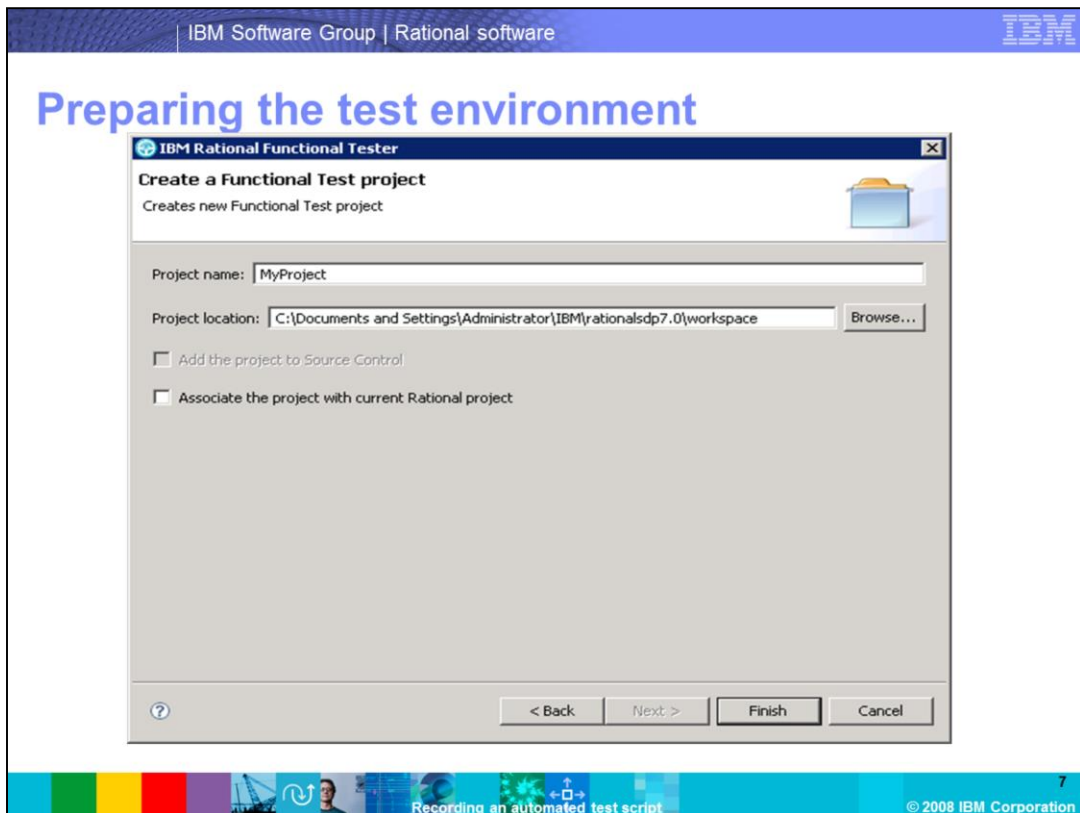
The Functional Tester perspective is a collection of views. The Functional Test Projects view, which is the left pane of the Functional Test perspective, lists the Projects, Scripts, and Logs. The Tasks view displays errors, warnings, or other information that is automatically generated by the compiler. A sub view within the Tasks view shows the Data pool. The Script view shows the script. When you double-click a script in the Functional Test Project View, it opens in the Script view. The Script Explorer view lists the script helper, helper superclass, test data pool, verification points, and test objects for the current script.

Preparing the test environment



A Functional Test project can be created by using the File menu.

From the File menu, select New > Functional Test Project. A New Project window will appear. Select Functional Test Project, and click Next.



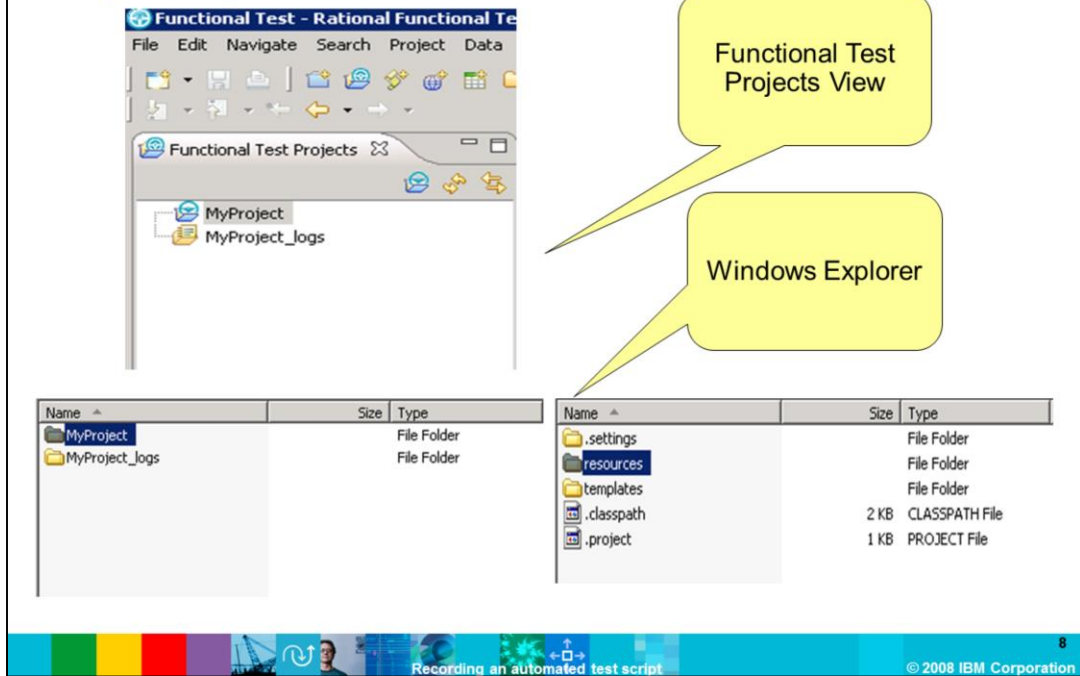
A window labeled “Create a Functional Test project” will appear.

In the Project name field, type the name of the new project. Project names cannot contain certain special characters like back or forward slashes, open or closed brackets, asterisks, colons, or question marks.

In the Project location field, type the data path for this project. By default, the location points to the Workspace folder. It is not necessary to create a project in the workspace folder; the workspace and the project can be in different locations. Click the Browse button to change the project location.

If you are using Rational ClearCase, and intend to check-in your scripts into a ClearCase VOB, select the “Add the project to Source Control” check box. If you are using Rational TestManager, and intend to associate the Functional Tester project to a TestManager project, select the “Associate the project with current Rational project” check box. These check boxes will not be available if the corresponding product is not installed on the machine. When complete, click Finish.

Preparing the test environment



Two new entries are created in the Functional Test Projects view. One is for the test artifacts, such as test scripts and shared object maps, and the other is for the Results logs.

If you go to your workspace folder in Windows Explorer, you will see the same two entries. If you open the project folder, you will see a resources folder that holds the object maps, verification points, data pools, a settings folder, and two configuration files, named .project and .classpath. There is also a templates folder that holds the script templates. The scripts themselves are stored directly in the project folder.

Preparing the test environment

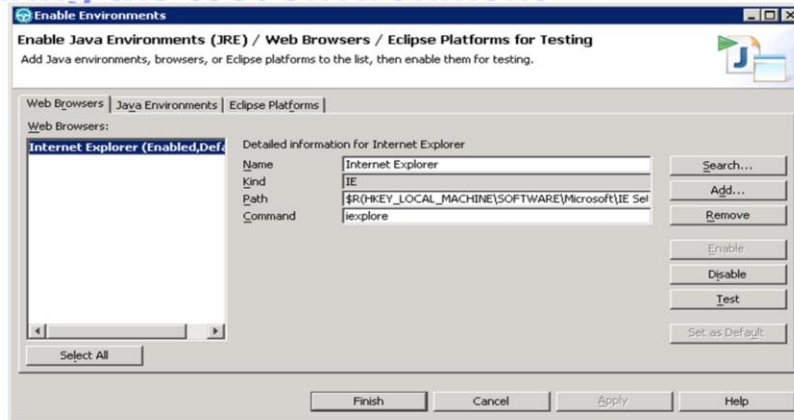
- **Enabling the test environment**

Configure>Enable Environment for Testing



After creating the project, you need to enable the test environment. This allows Functional Tester to see inside the runtime to identify objects within the application under test. From the Configure menu, select the “Enable Environments for Testing...” sub menu.

Preparing the test environment



Results Problem and Solution Details (Advanced)

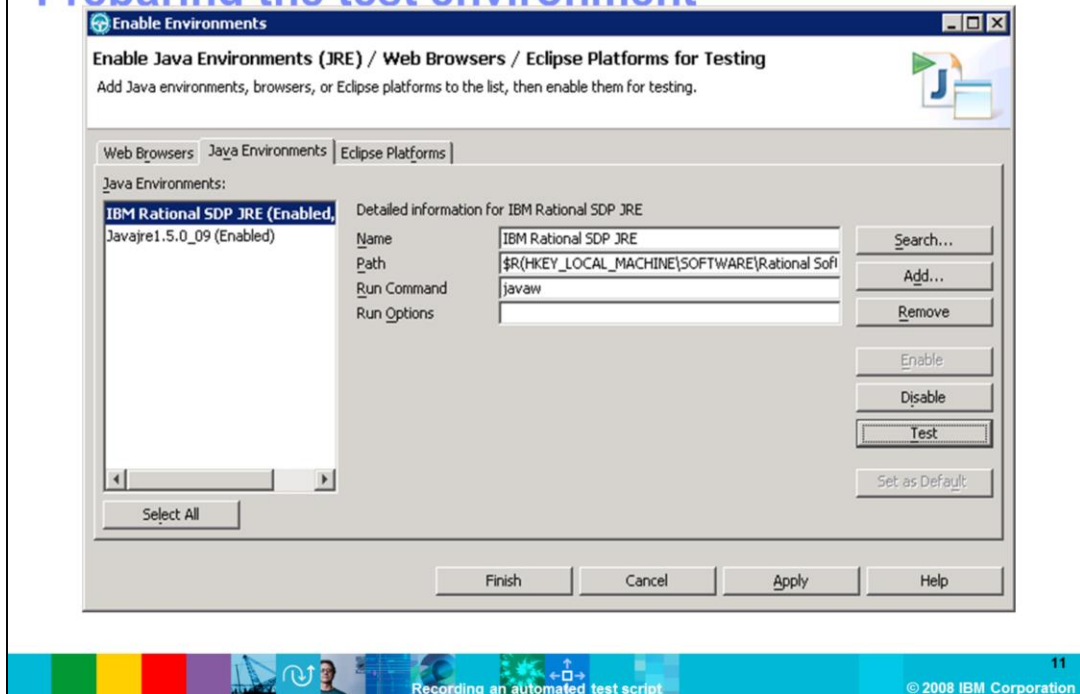
Browser Enablement Test Result: **Passed!**

Your browser is enabled properly.

The Enable Environments window will appear. This window will help you enable three different domains depending on the type of application you are testing. This window consists of three tabs which will be reviewed individually.

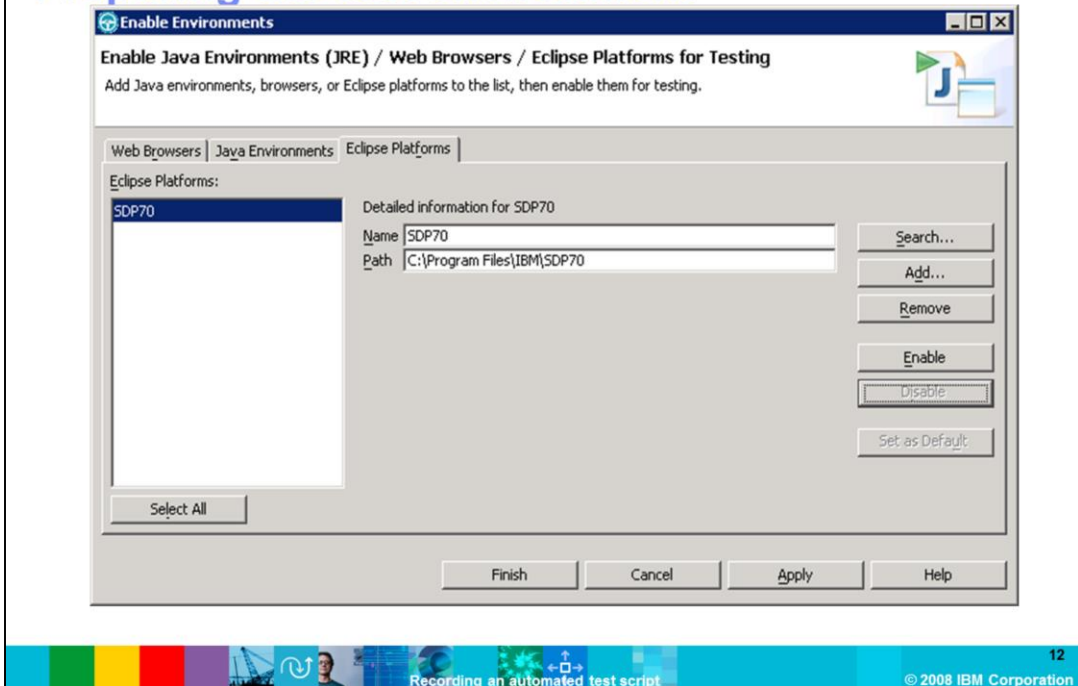
The first tab is the Web Browsers tab. If you are testing a Web application, you should make sure the Web browser you are going to use is enabled. If the browser is not already listed in the Web browsers section, click the Add button, browse to its location, and add it. This window has other buttons, as well. The Search button searches for all of the installed browsers on the machine. Remove will remove a Browser from the list. Enable will enable a browser, and Disable will disable a browser. The Test button lets you test the browser. Test the browser before recording scripts.

Preparing the test environment



The next tab is the Java Environments. On this tab, you can add and enable a JRE of your choice. The IBM Rational SDP JRE is the default JRE used to test Java applications, and is enabled for testing by Functional Tester. If the application uses another JRE, enable your JRE through this interface. This tab, like the Web Browser tab, has the search, add, remove, enable, disable and test buttons.

Preparing the test environment

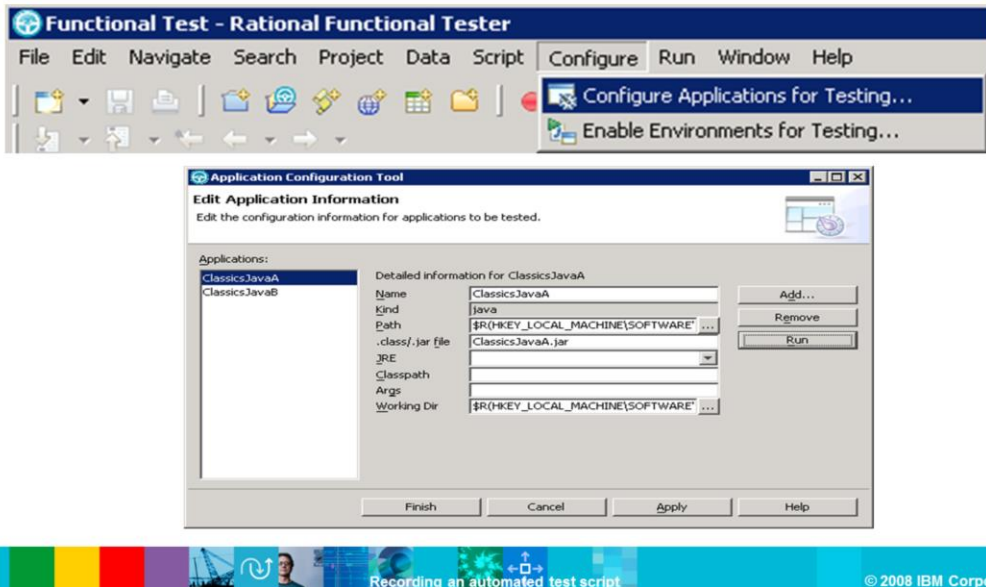


Functional Tester can test Eclipse based plug-ins and Rich Client Platform applications. If you have additional Eclipse platforms installed, you can enable them for testing on the Eclipse Platforms tab. The available button options on this tab are similar to the other two tabs.

Preparing the test environment

- **Configuring the application under test**

Configure > Enable Environments for Testing

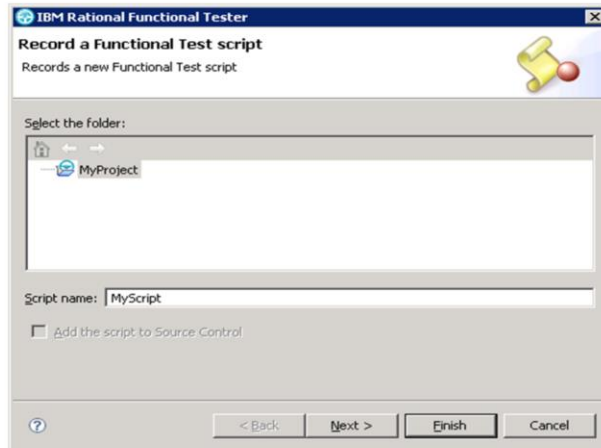
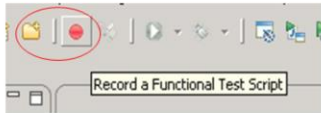


After enabling the test environment, you should configure the application that you are going to test. This creates a shortcut in the Functional Tester environment that makes it easier to launch the application and makes the tests more portable to other test machines.

From the Configure menu, select the “Configure Applications for Testing” sub menu. The Application Configuration Tool window will appear. A list of applications that have been configured for testing will be displayed on this window. You can use the Add button on this window to add your application. Functional Tester comes with a sample Java Swing application called ClassicsJava. You will be using this application to record a script. Click Apply, then click Finish.

Recording a test script

- Start the recorder



To start recording a script, click the red button on the toolbar. Clicking on the red button will open the "Record a Functional Test script" window.

On this window, you can select the project in which you want to record the script if you have multiple projects. You can specify a name for the script in the script name field. Click Next.

Recording a test script

The screenshot shows the 'Select Script Assets' dialog box in IBM Rational Functional Tester. The dialog has a title bar with the IBM logo and the text 'IBM Rational Functional Tester'. Below the title bar, the main heading is 'Select Script Assets' with a yellow pushpin icon. The instruction text reads: 'Select a Test Object Map, Helper Superclass and Test Datapool to use with new script'. There are three text input fields: 'Test Object Map' with the value 'Private Test Object Map', 'Helper Superclass' with the value 'RationalTestScript', and 'Test Datapool' with the value 'Private Test Datapool'. Each field has a 'Browse...' button to its right. Below these fields is a 'Datapool Record Selection Order' dropdown menu set to 'Sequential'. At the bottom of the dialog, there is a section titled 'Set as test asset default for new scripts in this project:' with two checkboxes: 'Test Object Map' and 'Helper Superclass', both of which are currently unchecked. At the very bottom of the dialog are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. A help icon (?) is located to the left of the 'Back' button.

The Select Script Assets window lets you choose a Test Object Map, Helper Superclass and a test data pool. These fields are populated with default values. You can click on the Browse buttons and select a different value for each of the fields. For example, if you are going to use a shared object map, then you select it by clicking on the browse button next to the Test Object Map field.

The Datapool Record Selection Order selection box lets you define the order in which you want Functional Tester to select the rows in a .

The Test Object Map and Helper Superclass check boxes let you set as the default the Object Map and Superclass you selected for this script for all future script recordings in this project.

When complete, click Finish.

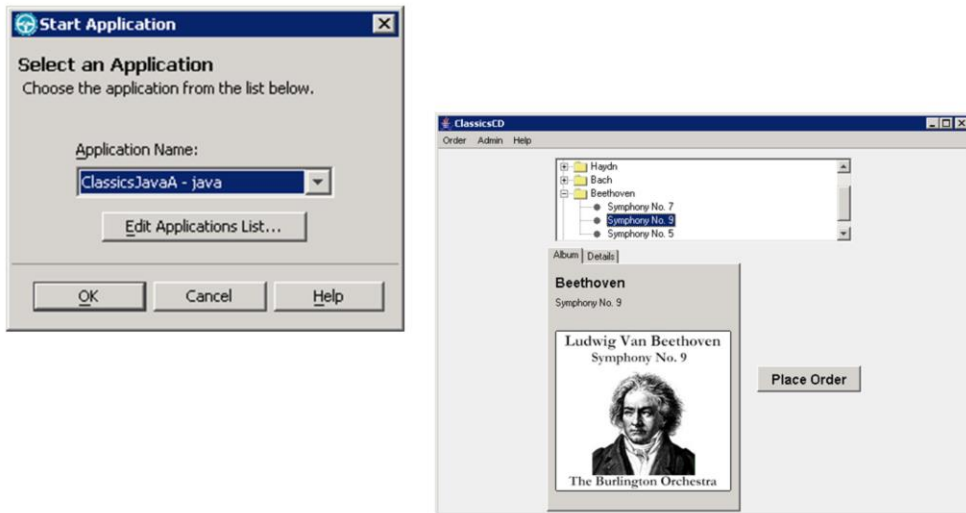
Recording a test script



The Functional Tester window will minimize and the Recording window will appear. The Recording window has a variety of icons. The main section of this window gives a description of each icon. While recording a script, the code that is generated is visible in the main section of this window. To begin recording, click the Start Application icon, which is the third option from the left.

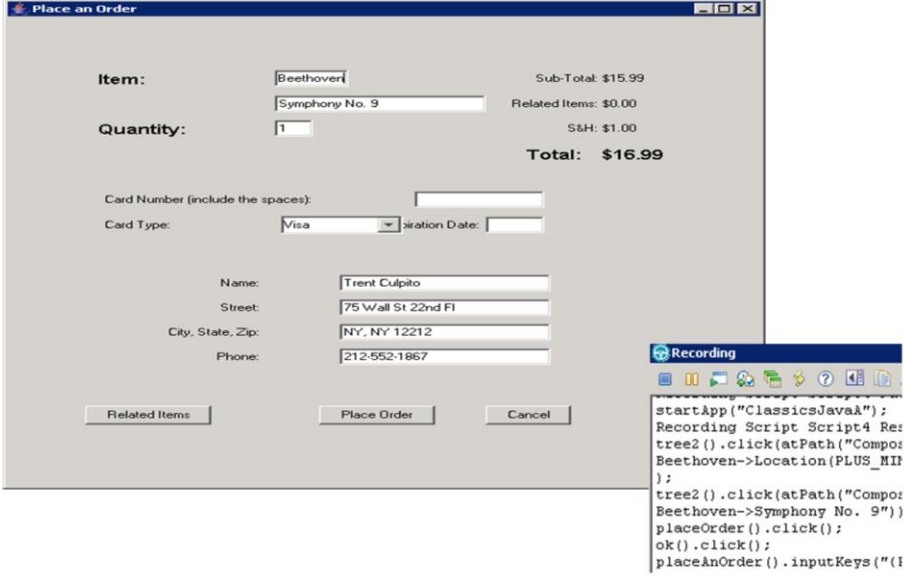
Recording a test script

- Record an automation script



The Start Application window will appear. The “Application Name:” drop-down box lists all of the applications added from the Configure menu in Functional Tester. Alternately, you can click the “Edit Applications List...” button, and add the application you are going to test. You are going to use the ClassicsJavaA sample application that is installed along with Functional Tester. You will go through the process of ordering a CD.

Recording a test script



The screenshot shows a web application window titled "Place an Order". The form contains the following fields and values:

- Item: Beethoven
- Quantity: 1
- Card Type: Visa
- Name: Trent Culpito
- Street: 75 Wall St 22nd Fl
- City, State, Zip: NY, NY 12212
- Phone: 212-552-1867

Summary information on the right side of the form:

- Sub-Total: \$15.99
- Related Items: \$0.00
- S&H: \$1.00
- Total: \$16.99**

Buttons at the bottom include "Related Items", "Place Order", and "Cancel".

An overlay window titled "Recording" is visible in the bottom right corner, displaying the following test script code:

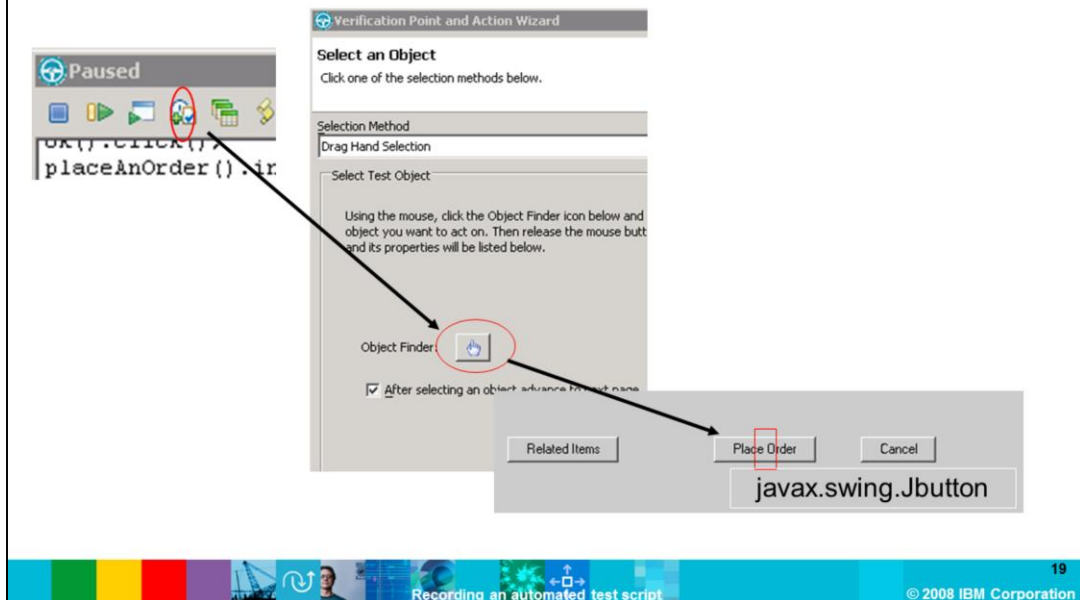
```
startApp("ClassicsJavaA");
Recording Script Script4 Re
tree2().click(atPath("Compo
Beethoven->Location(PLUS_MI
));
tree2().click(atPath("Compo
Beethoven->Symphony No. 9"))
placeOrder().click();
ok().click();
placeAnOrder().inputKeys("I
```

Start entering values in the boxes on the "Place an Order" screen.

By default, all keyboard entries are captured in the test script. The advantage in doing so is that the same test procedure can be used over and over with different sets of varied data, enabling you to reuse common tests and greatly reducing the time and effort involved in creating repetitive tests.

Recording a test script

- **Creating a verification point**



When you are testing an application, check to see if the application is working as expected. You want to make sure that the error messages are displayed when they are supposed to, that the data displays correctly, and that the properties of a particular object are correct.

You can include those checks in your script by using verification points. A verification point is a point in a script that you create to confirm the state of the application being tested across builds or runs.

On the Recording window, select the fourth icon from the left. A “Verification Point and Action Wizard” window will appear. This window has a button with a hand symbol on it.

Put your mouse pointer on this button, and, with the left mouse button pressed down, drag the hand button to the location where you want to create the verification point. In this case, you will put the Place Order button on the Place an Order window, and then release the left mouse button.

Recording a test script

Verification Point and Action Wizard

Select an Action
Choose action to perform against selected Test Object

Perform Data Verification Point

Perform Properties Verification Point

Get a Specific Property Value

Wait for Selected Test Object

Description for Selected Action:
Verify data for the selected Test Object intentionally or unintentionally.

Insert Verification Point Data Command
Create data Verification Point and insert test into script

Create a data verification point for:
PlaceOrder
Data Value:
Visible Text

Verification Point Name:
PlaceOrder_text

Verification Point Data
Check the properties to include in the verification point.

Property	Value
data	Place Order
description	Visible Text
ignoreCase	false
name	PlaceOrder_text
type	text
whiteSpaceMode	Use All Characters

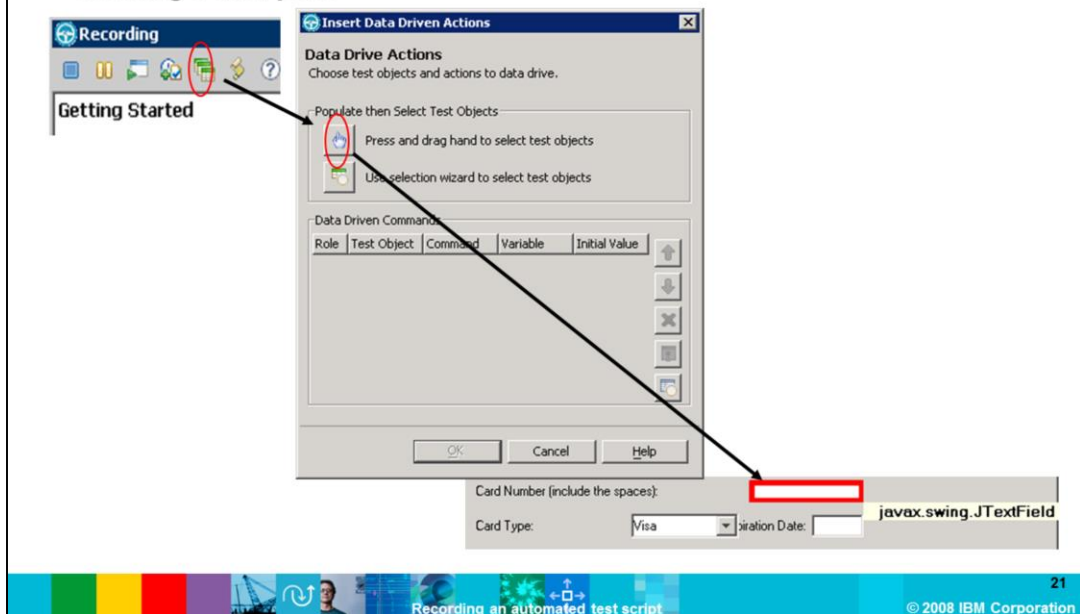
Place Order

< Back Next > Finish Cancel Help

The Verification Point and Action Wizard will give the options to create four kinds of verification points. In this module, you will create a data verification point by selecting the first option. This verification point helps in making sure that the data is accurate. When you click the Next button, the windows that appear show the data that was captured. In this case, it is the label on the button you previously selected.

Recording a test script

▪ Creating a data pool



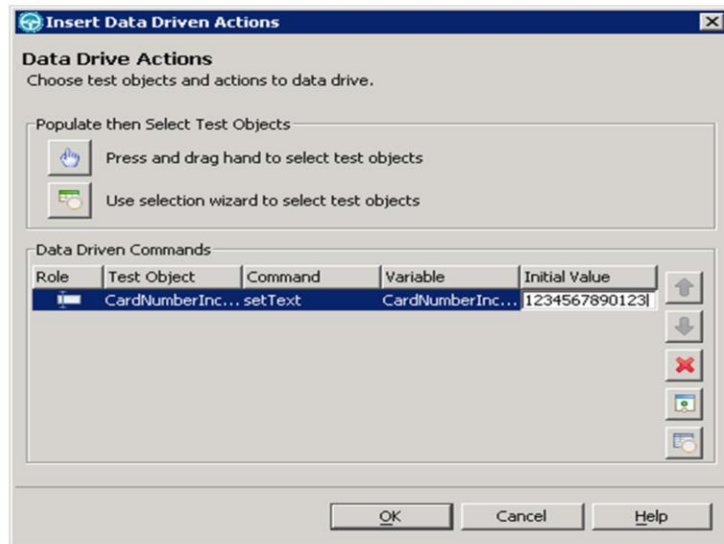
When you record a script using literal values, that data is hard-coded into the script. A script with hard-coded data executes only one test case or one set of valid test inputs, and it is difficult to maintain and use. The way to get around this is to create data driven test scripts.

In data driven testing, you separate the data from the test script. Instead of hard-coding data into the script, the script is coded to accept variable data from an external source. Because data is separated from the test script, you can modify test data without affecting the test script and share the test data among many test scripts.

In Functional Tester, the data sheet that holds the data values is called a data pool.

On the Recording window, select the fifth icon from the left. An “Insert Data Driven Actions” window will appear. This window has a button with a hand symbol on it. Put your mouse pointer on this button, and, with the left mouse button pressed down, drag the mouse pointer and place the hand on the object where you want to create the data pool. In this case, you will put the card number field on the Place an Order window, and then release the left mouse button.

Recording a test script



22

Recording an automated test script

© 2008 IBM Corporation

In the Data Driven Commands section of the Insert Data Driven Actions window, a new row of data will be created. Each row has five columns.

The Role column indicates object type.

The Test Object column indicates the name of the object.

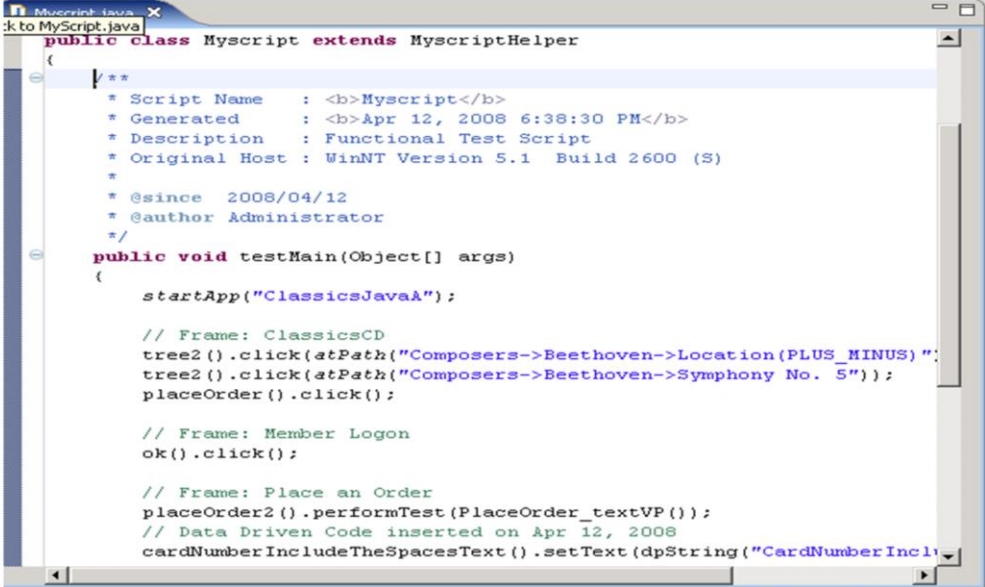
The Command column shows the command that is going to be used to set the value from the data pool into the field.

The Variable column indicates the variable name created for the object.

The Initial Value column indicates the initial value for the field.

Click the OK button, and then click the Stop icon in the Recording window.

Examining the recorded test script



```
public class Myscript extends MyscriptHelper
{
    /**
     * Script Name      : <b>Myscript</b>
     * Generated       : <b>Apr 12, 2008 6:38:30 PM</b>
     * Description     : Functional Test Script
     * Original Host   : WinNT Version 5.1 Build 2600 (S)
     *
     * @since 2008/04/12
     * @author Administrator
     */
    public void testMain(Object[] args)
    {
        startApp("ClassicsJavaA");

        // Frame: ClassicsCD
        tree2().click(atPath("Composers->Beethoven->Location(PLUS_MINUS)");
        tree2().click(atPath("Composers->Beethoven->Symphony No. 5"));
        placeOrder().click();

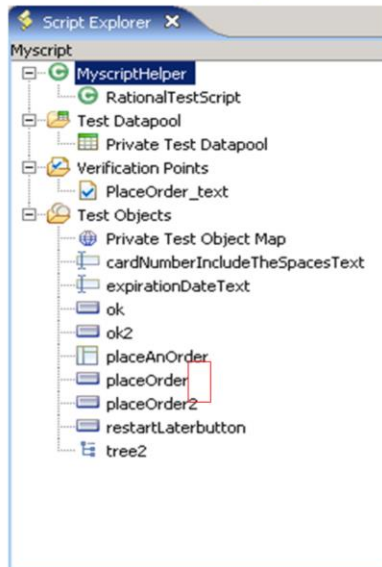
        // Frame: Member Logon
        ok().click();

        // Frame: Place an Order
        placeOrder2().performTest(PlaceOrder_textVP());
        // Data Driven Code inserted on Apr 12, 2008
        cardNumberIncludeTheSpacesText().setText(dpString("CardNumber Incl
```

Now that you have stopped recording, go ahead and examine the script. When the recording is stopped, the Functional Tester main window comes up. In the script view, you will see the recorded script.

Notice that the script is Java Code. Functional Tester uses its default script templates to format and provide basic information when you create a script. You can customize the information and format by customizing the script templates. You can find the templates in the templates folder in your project folder.

Test object map



24

Recording an automated test script

© 2008 IBM Corporation

When you record a script, Functional Tester creates a test object map for the application under test. Each script is associated with a test object map file. The object map is like a repository of all the test objects in the application under test. It is a static view that describes test objects known to Functional Tester. When a script is played back, Functional Tester uses the object map to identify the objects it needs to access. It reads the objects in the script and tries to make a 1 to 1 match with the objects in the object map.

The Test Object Map is visible in the Script Explorer view in Functional Tester. A test object map can be shared among multiple scripts.

IBM Software Group | Rational software

Data pool

Script Explorer view

Task view

CardNumberIncludeTheSpacesText::java.lang.String
0 1234567890123

Script view

```
// Data Driven Code inserted on Apr 12, 2008
cardNumberIncludeTheSpacesText().setText(dpString("CardNumberIncludeTheSpacesText"));
```

Recording an automated test script

© 2008 IBM Corporation 25

When you create a data pool during the recording of a script, the data pool shows up in the Script Explorer view. When you double click it, a data pool sub view will open in the Task View area. Here you can add more rows of data. The Script View will show the line of script that was generated for the data pool.

Helper superclass

The screenshot displays the IBM Rational software interface. At the top, the title bar reads "IBM Software Group | Rational software" and the IBM logo is in the top right corner. The main window is titled "Script Explorer" and shows a tree view under "MyScript". The tree includes "MyScriptHelper" (highlighted with a red oval and a yellow callout box labeled "Script Explorer view"), "RationalTestScript", "Test Datapool", "Private Test Datapool", "Verification Points", and "PlaceOrder_text". Below the tree, a script editor shows the following code:

```

import resources.Script4Helper;

/**
 * Description : Functional Test Script
 * @author Administrator
 */
public class Script4 extends Script4Helper
{
    /**
     * Script Name : <b>Script4</b>
     * Generated : <b>Apr 22, 2008 9:44:2
     * Description : Functional Test Script
     * Original Host : WinNT Version 5.1 Bu
    */
}

```

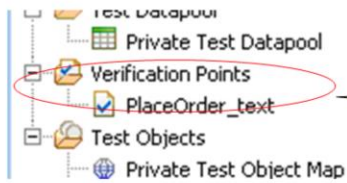
A context menu is open over the "Script4Helper" class name in the code, with the "Open Declaration F3" option selected. The menu also includes "Undo Ctrl+Z", "Revert File", "Save", "Open Type Hierarchy F4", "Open Call Hierarchy Ctrl+Alt+H", "Quick Outline Ctrl+O", "Quick Type Hierarchy Ctrl+T", and "Show In Alt+Shift+W".

At the bottom of the screenshot, there is a blue bar with a colorful rainbow pattern on the left, a central icon of a person, and the text "Recording an automated test script". On the right side of this bar, the number "26" and the copyright notice "© 2008 IBM Corporation" are visible.

Functional Tester scripts use a Helper Superclass. By default, all Functional Tester scripts extend the RationalTestScript class, and thereby inherit several methods. You can create your own helper Superclass if you want to add additional methods or override the methods within RationalTestScript.

The Helper Superclass is visible in the Script Explorer view. In the Script view, you can highlight ScriptHelper, right-click, and select Open Declaration. The Helper Superclass will open in the Script view.

Verification points



Script Explorer view

```
// Frame: Place an Order  
placeOrder2().performTest(PlaceOrder_textVP());
```

Script view



The verification point is also visible in the Script Explorer view. You can double-click it, open the Verification Point window, and make any modifications to the verification point you created. The Script view will show the line of script that was generated for the data pool.

Summary

- This module showed an overview of recording a script in IBM Rational Functional Tester for Java
- A Tester can
 - ▶ Set up the environment for testing
 - ▶ Configure the application that is going to be tested
 - ▶ Record a script
 - ▶ Modify some of the aspects of a test



This module showed an overview of recording a script in IBM Rational Functional Tester for Java.

A Tester can set up the environment for testing, configure the application that is going to be tested, and record a script. A tester can modify some of the aspects of a test, such as adding rows to a data pool, modifying the verification point that was created, or creating a customized Helper Superclass.

Trademarks, copyrights, and disclaimers

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

IBM Rational

A current list of other IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation in the United States, Other Countries, or both.

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. 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.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

