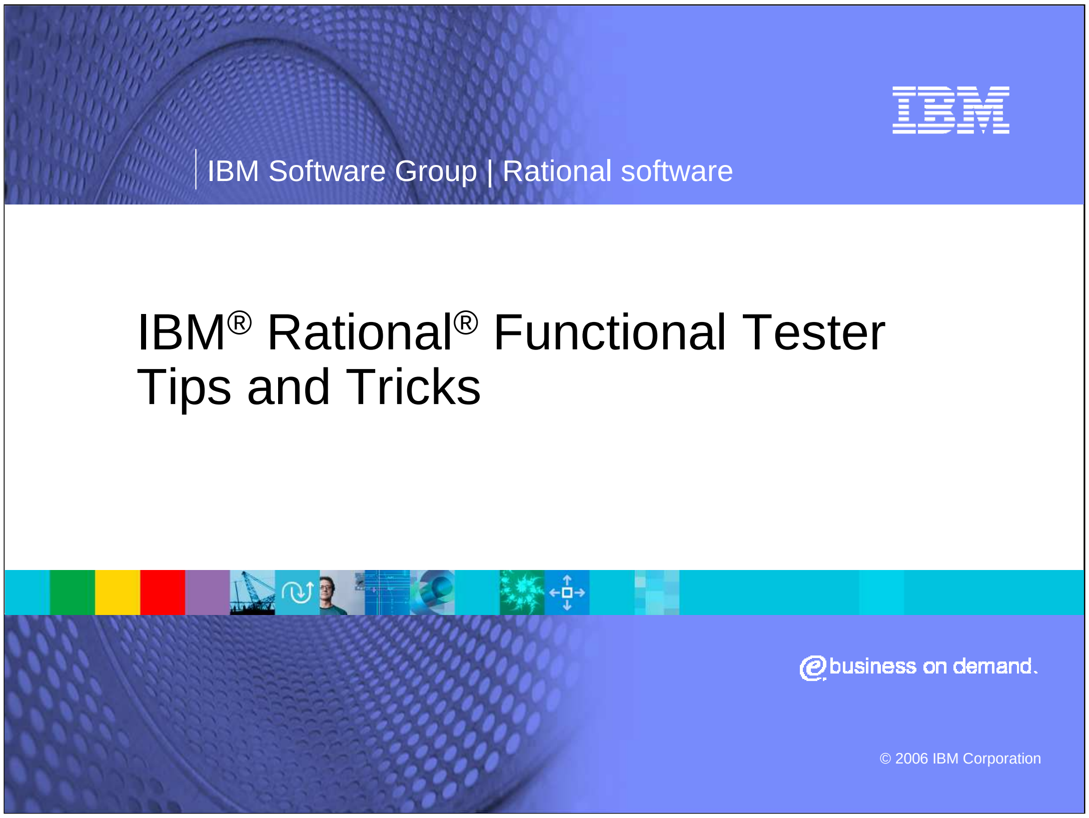
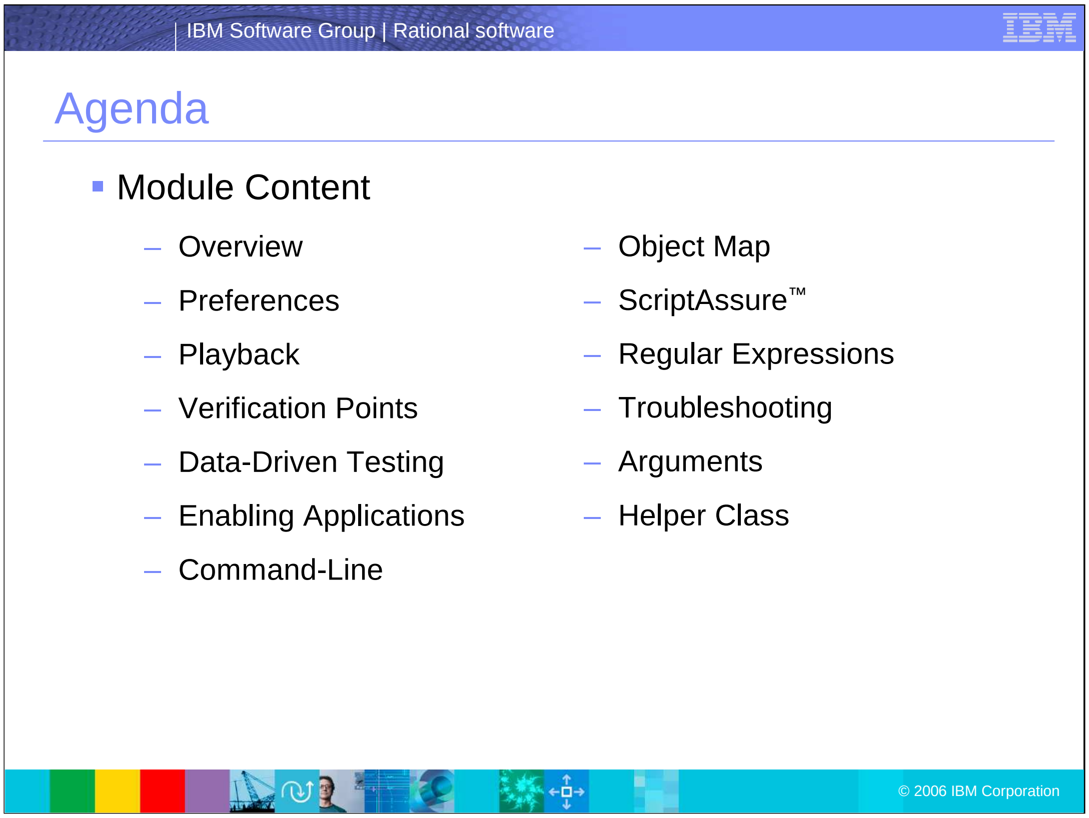


IBM Software Group | Rational software

IBM® Rational® Functional Tester Tips and Tricks

business on demand.

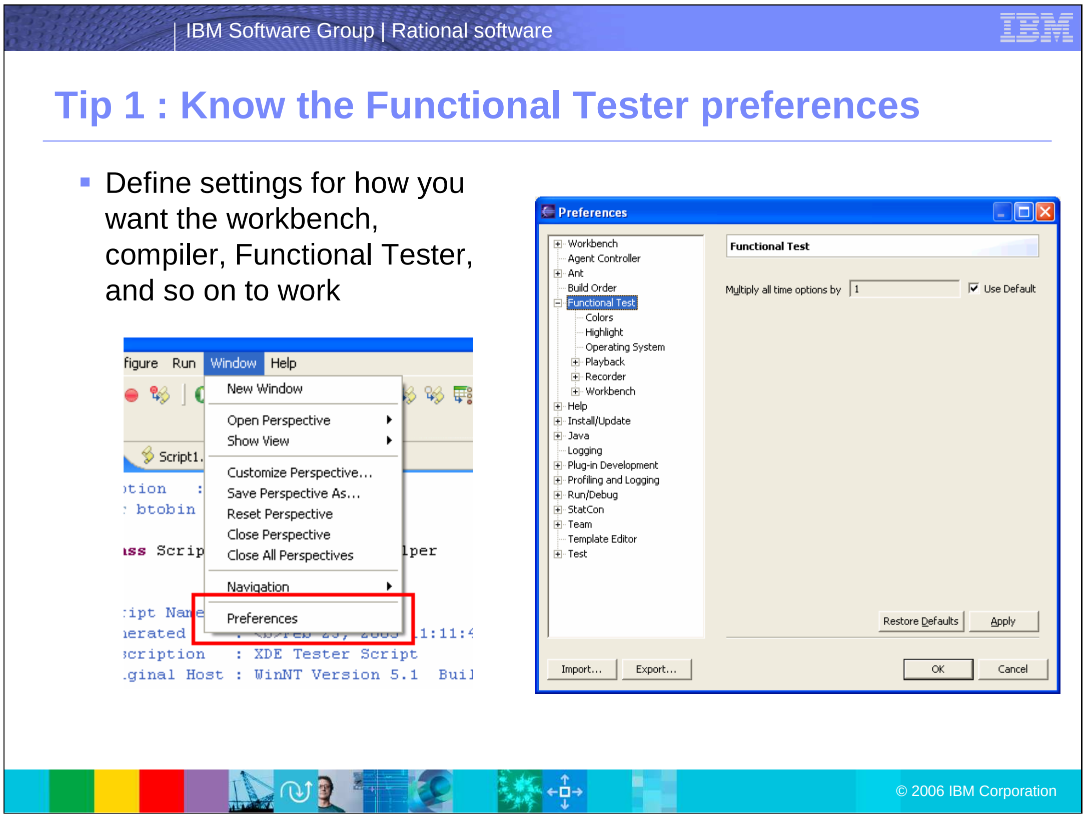
This presentation covers some tips and tricks for using IBM Rational Functional Tester.



Agenda

- Module Content
  - Overview
  - Preferences
  - Playback
  - Verification Points
  - Data-Driven Testing
  - Enabling Applications
  - Command-Line
  - Object Map
  - ScriptAssure™
  - Regular Expressions
  - Troubleshooting
  - Arguments
  - Helper Class

This module will cover preferences, playback, verification points, and the rest of the topics listed here. This training is intended to detail what can be done within Rational Functional Tester, rather than giving step-by-step instructions on how to perform these functions.



IBM Software Group | Rational software

Tip 1 : Know the Functional Tester preferences

- Define settings for how you want the workbench, compiler, Functional Tester, and so on to work

Multiply all time options by 1 ☑ Use Default

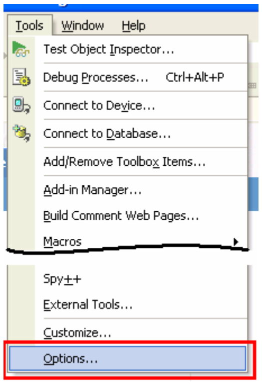
© 2006 IBM Corporation

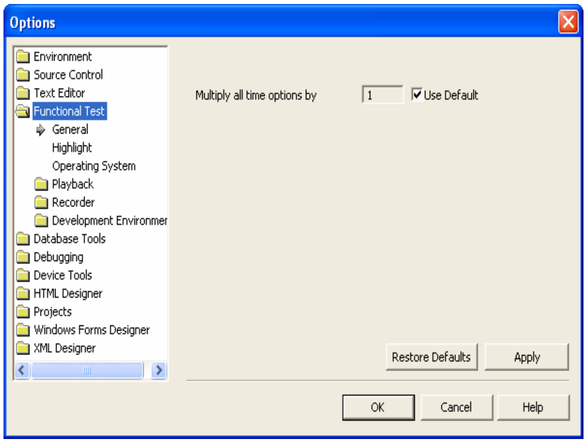
Tip 1: Know the Functional Tester Preferences.

Use the Preferences option in Functional Tester to define settings for how you want the workbench, compiler, and other environments to work. Note that Preferences are called “preferences” in Eclipse, while they are called “options” in VS.Net. Regardless of what they are called, this is the primary interface to Rational Functional Tester and IDE customization and control.

Tip 1 : Know the Functional Tester preferences

- Define settings for how you want the workbench, compiler, Functional Tester, and so on to work

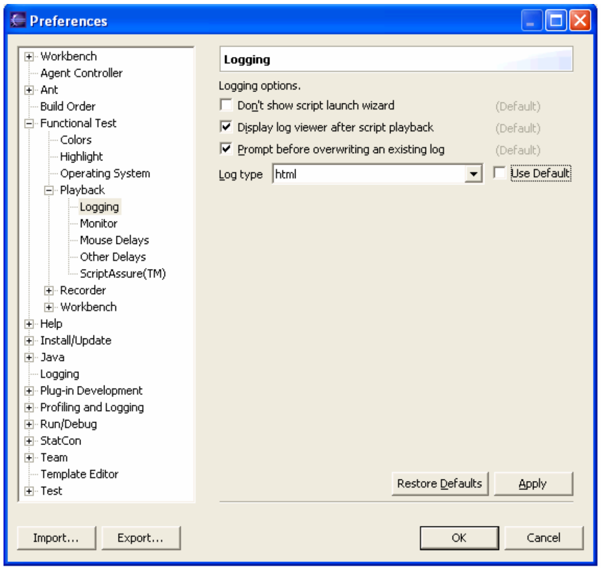




Multiply all time options by 1 ☑ Use Default

These are the VS.Net options windows. You would make the same changes here as you did to the preferences in the previous slide. This presentation will explore some of the more important options, but you can make changes that fit your needs.

Tip 1 : Know the Functional Tester preferences



Logging

Logging options.

☐ Don't show script launch wizard (Default)

☑ Display log viewer after script playback (Default)

☑ Prompt before overwriting an existing log (Default)

Log type: html ☐ Use Default

Show

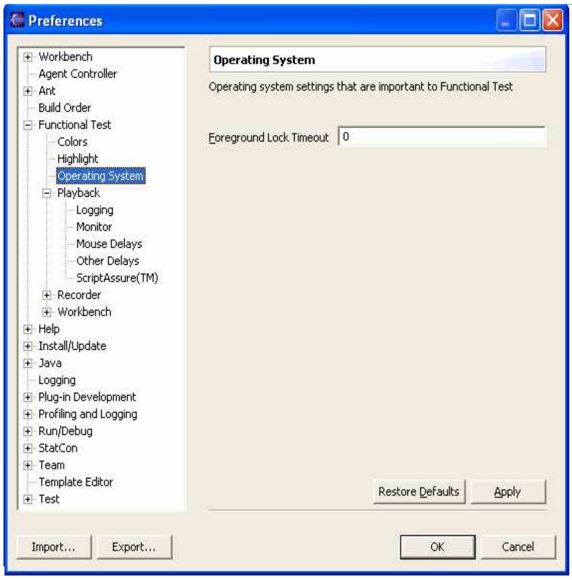
Display Log

Overwrite

Log Type

You can customize your settings for logging. For example, you can choose to display the log viewer after script playback, prompt before overwriting an existing log, and specify log types. It may be advantageous to select “none” under type when you do not really want a log to be generated, such as during script development.

Tip 1 : Know the Functional Tester preferences



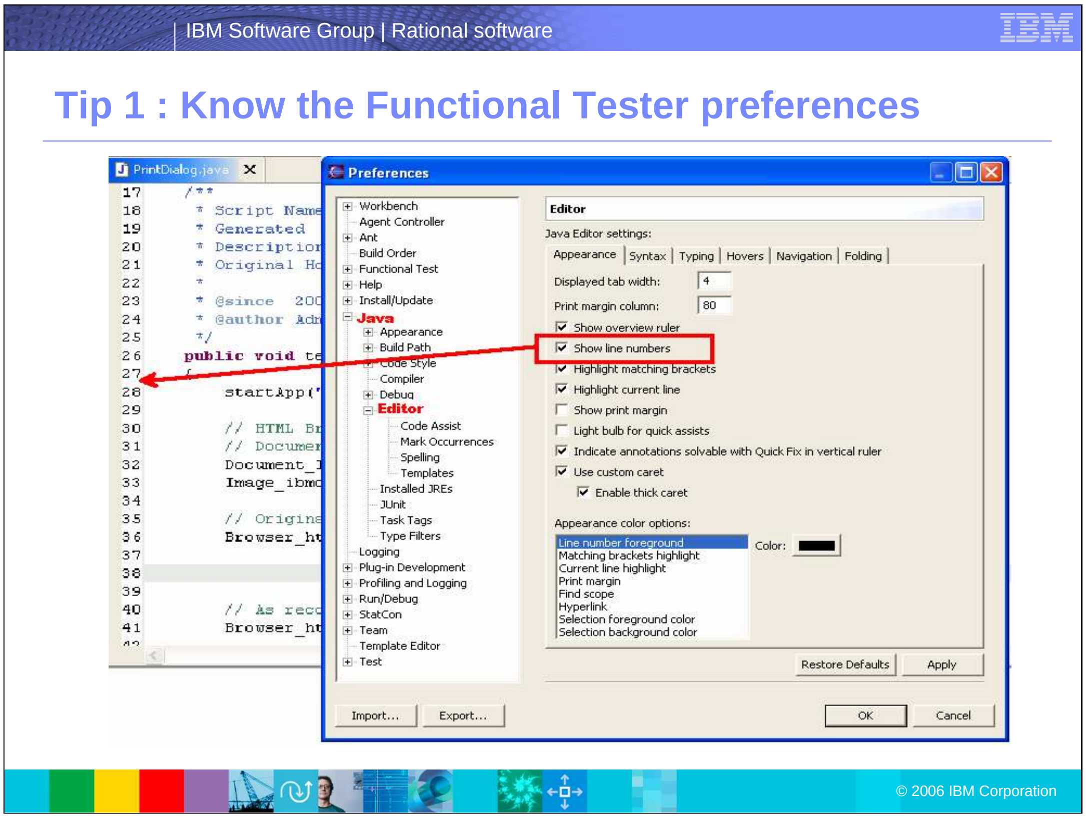
Operating System

Operating system settings that are important to Functional Test

Foreground Lock Timeout 0

Amount of time (in milliseconds) after user input, during which the operating system does not allow applications to force themselves into the foreground.

A foreground lock timeout specifies to operating systems the amount of time (in milliseconds) after user input to block applications from showing up in the foreground. Setting the lock timeout to zero as shown here allows the applications to work in real-time without imposed OS delays.



IBM Software Group | Rational software

Tip 1 : Know the Functional Tester preferences

PrintDialog.java

```
/**
 * Script Name
 * Generated
 * Description
 * Original Ho
 *
 * @since  200
 * @author Adr
 */
public void te
{
    startApp('

    // HTML Br
    // Documen
    Document_I
    Image_ibmc

    // Origina
    Browser_ht


    // As reco
    Browser_ht
```

Preferences

- Workbench
- Agent Controller
- Ant
- Build Order
- Functional Test
- Help
- Install/Update
- **Java**
  - Appearance
  - Build Path
  - Code Style
  - Compiler
  - Debug
  - **Editor**
    - Code Assist
    - Mark Occurrences
    - Spelling
    - Templates
  - Installed JREs
  - JUnit
  - Task Tags
  - Type Filters
- Logging
- Plug-in Development
- Profiling and Logging
- Run/Debug
- StatCon
- Team
- Template Editor
- Test

Editor

Java Editor settings:

Displayed tab width: 4

Print margin column: 80

☑ Show overview ruler
☑ Show line numbers
☑ Highlight matching brackets
☑ Highlight current line
□ Show print margin
□ Light bulb for quick assists
☑ Indicate annotations solvable with Quick Fix in vertical ruler
☑ Use custom caret
☑ Enable thick caret

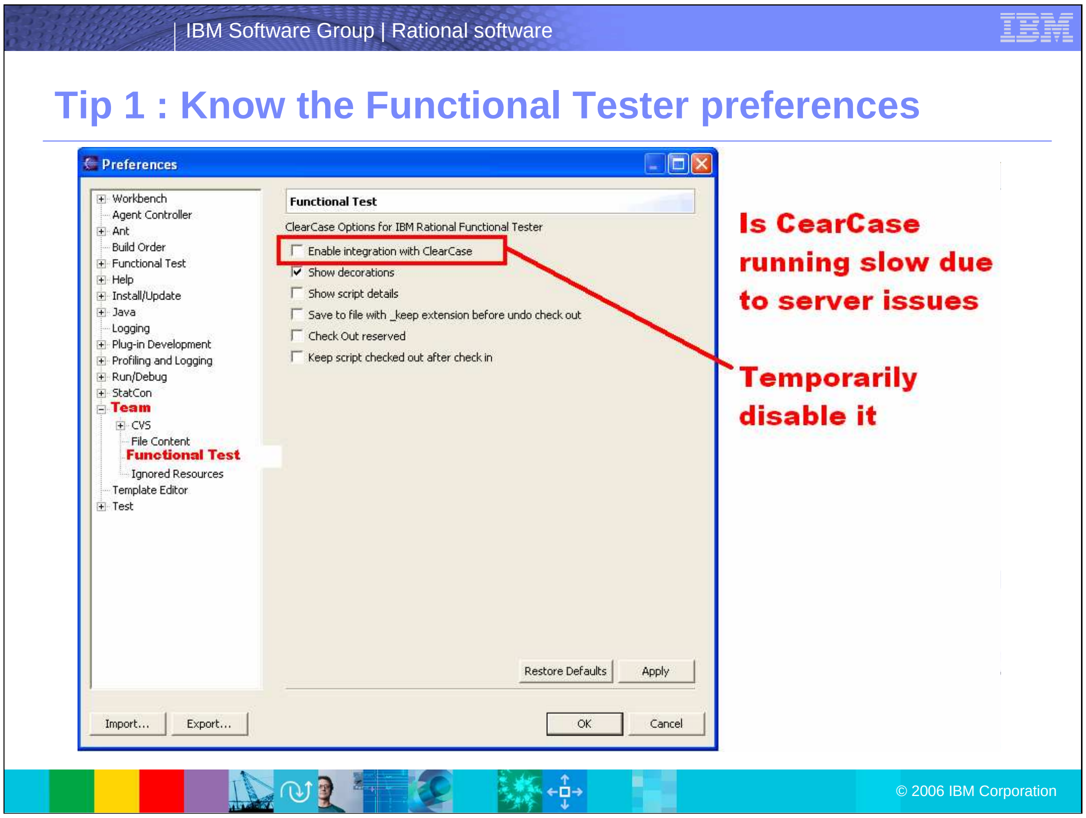
Appearance color options:

- Line number foreground
- Matching brackets highlight
- Current line highlight
- Print margin
- Find scope
- Hyperlink
- Selection foreground color
- Selection background color

Color: ■

© 2006 IBM Corporation

Settings in Rational Functional Tester preferences to customize the editor based on your needs are shown here. For example, non-Rational Functional Tester preferences such as adding line numbers to the editor affect Rational Functional Tester. Use these preferences to manage the editor behaviors.



IBM Software Group | Rational software

Tip 1 : Know the Functional Tester preferences

Preferences

Functional Test

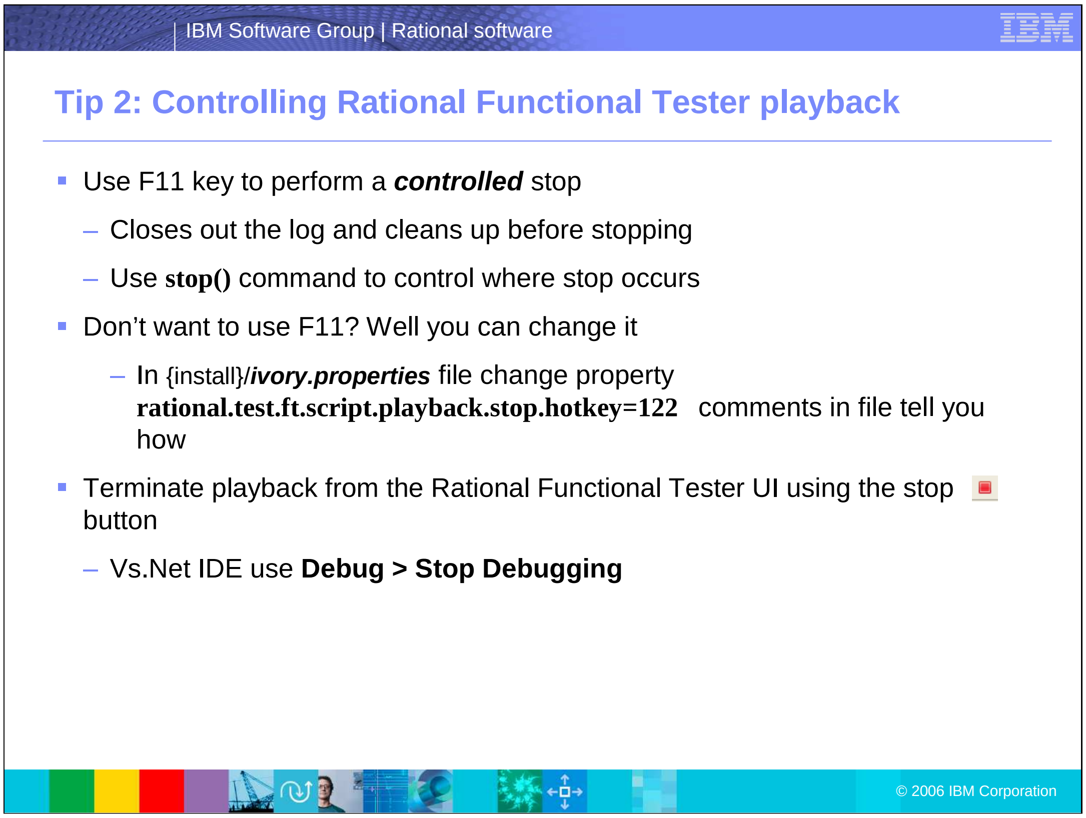
ClearCase Options for IBM Rational Functional Tester

- ☐ Enable integration with ClearCase
- ☑ Show decorations
- ☐ Show script details
- ☐ Save to file with _keep extension before undo check out
- ☐ Check Out reserved
- ☐ Keep script checked out after check in

Is CearCase running slow due to server issues

Temporarily disable it

When integrating with other Rational products, it is important to set preferences here. You can turn on or off integrations, such as with Rational ClearCase®. ClearCase support, in general, is a non-Rational Functional Tester function – but there is a Rational Functional Tester-specific set of options that need to be taken into account and are not specifically related to the Functional Test preferences.



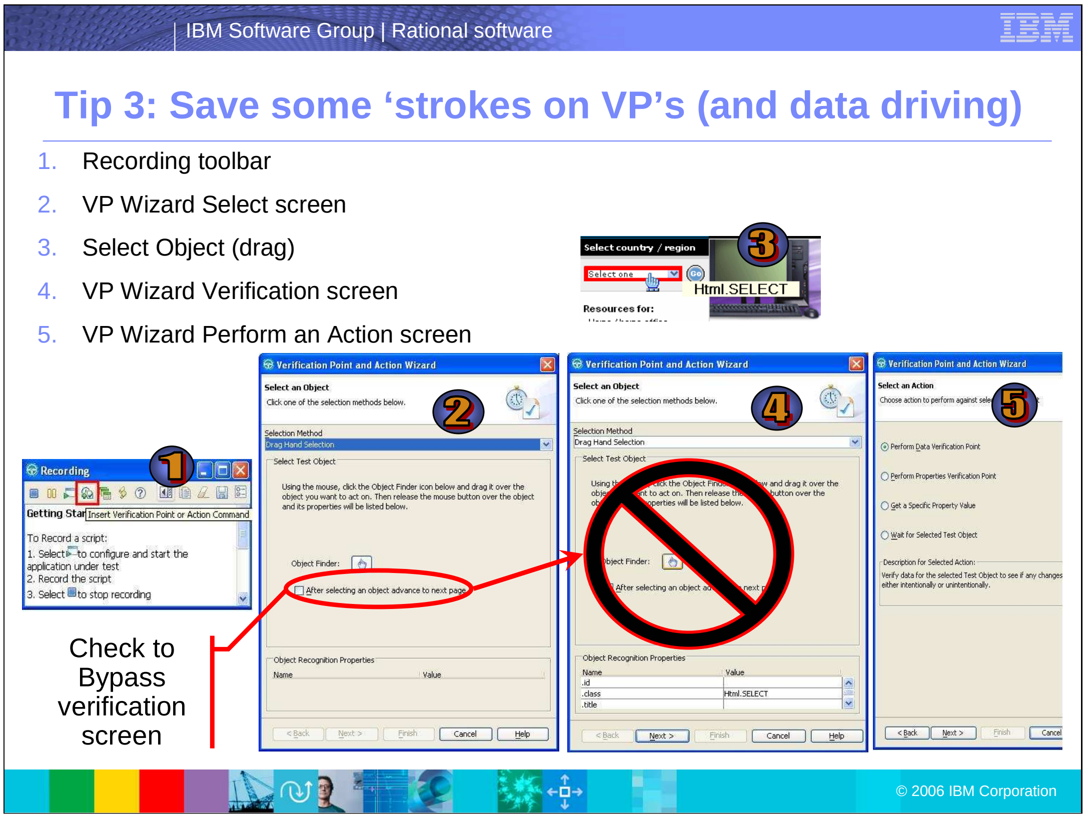
Tip 2: Controlling Rational Functional Tester playback

- Use F11 key to perform a ***controlled*** stop
  - Closes out the log and cleans up before stopping
  - Use **stop()** command to control where stop occurs
- Don’t want to use F11? Well you can change it
  - In {install}/***ivory.properties*** file change property **rational.test.ft.script.playback.stop.hotkey=122** comments in file tell you how
- Terminate playback from the Rational Functional Tester UI using the stop ■ button
  - Vs.Net IDE use **Debug > Stop Debugging**

Tip 2: Controlling Rational Functional Tester Playback

Rational Functional Tester has a little-known feature that allows you to stop script playback. In this example, use the F11 key to perform a controlled stop. The steps listed here give instructions on how to do this.

Why change the F11 key to some other key code? It is common for more advanced users to customize their function key settings to have alternate mappings for commonly used capabilities. If you already use F11 for something else that may be used during playback, there could be a conflict. Basically this is only interesting if you have remapped a debugger key, because temporarily taking over the F11 key would only really impact your ability to use mappings in the debugger.


IBM Software Group | Rational software
IBM

Tip 3: Save some ‘strokes on VP’s (and data driving)

1. Recording toolbar
2. VP Wizard Select screen
3. Select Object (drag)
4. VP Wizard Verification screen
5. VP Wizard Perform an Action screen

3
Select country / region
Select one
Go
Html.SELECT
Resources for:
1
Recording
Getting Star
Insert Verification Point or Action Command

To Record a script:
1. Select to configure and start the application under test
2. Record the script
3. Select to stop recording

2
Verification Point and Action Wizard
Select an Object

Click one of the selection methods below.

Selection Method: Drag Hand Selection

Select Test Object

Using the mouse, click the Object Finder icon below and drag it over the object you want to act on. Then release the mouse button over the object and its properties will be listed below.

Object Finder:

☐ After selecting an object advance to next page

Object Recognition Properties

Name	Value

Click one of the selection methods below.

Selection Method: Drag Hand Selection

Object Recognition Properties

Name	Value
.id	
.class	Html.SELECT
.title	

○ 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 to see if any changes either intentionally or unintentionally.

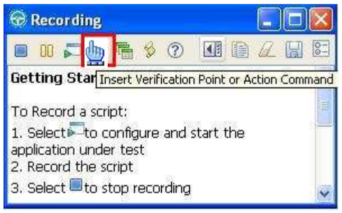
Check to Bypass verification screen

Tip 3: Save key strokes on VPs and Data Driving

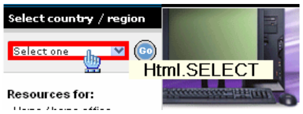
You can save some keystrokes when setting verification points in Rational Functional Tester. If you do not need to change anything on the Verification Point and Action Wizard panel (shown here in screen 4), uncheck the “After selecting object” option in screen 2. This provides short-cut-like functionality later.

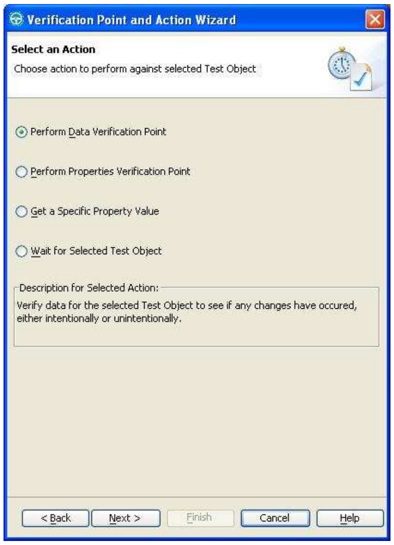
Tip 3: Save some ‘strokes on VP’s

- Drag from Recording toolbar to select object



To Record a script:
1. Select ▶ to configure and start the application under test
2. Record the script
3. Select ■ to stop recording





Select an Action

Choose action to perform against selected Test Object

● Perform Data Verification Point

○ Perform Properties Verification Point

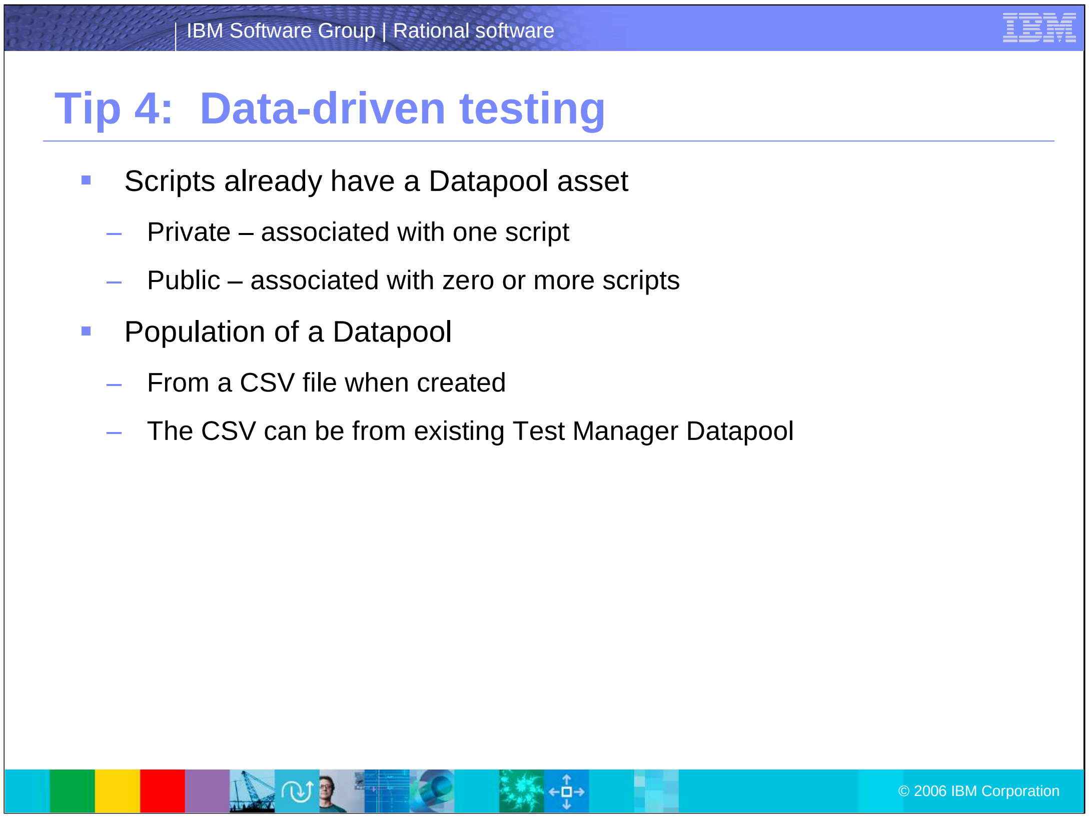
○ Get a Specific Property Value

○ Wait for Selected Test Object

Description for Selected Action:

Verify data for the selected Test Object to see if any changes have occured, either intentionally or unintentionally.

After setting up the quick keystrokes, you will then be able to use the drag (represented by the hand) pointer from the recording toolbar to select an object.



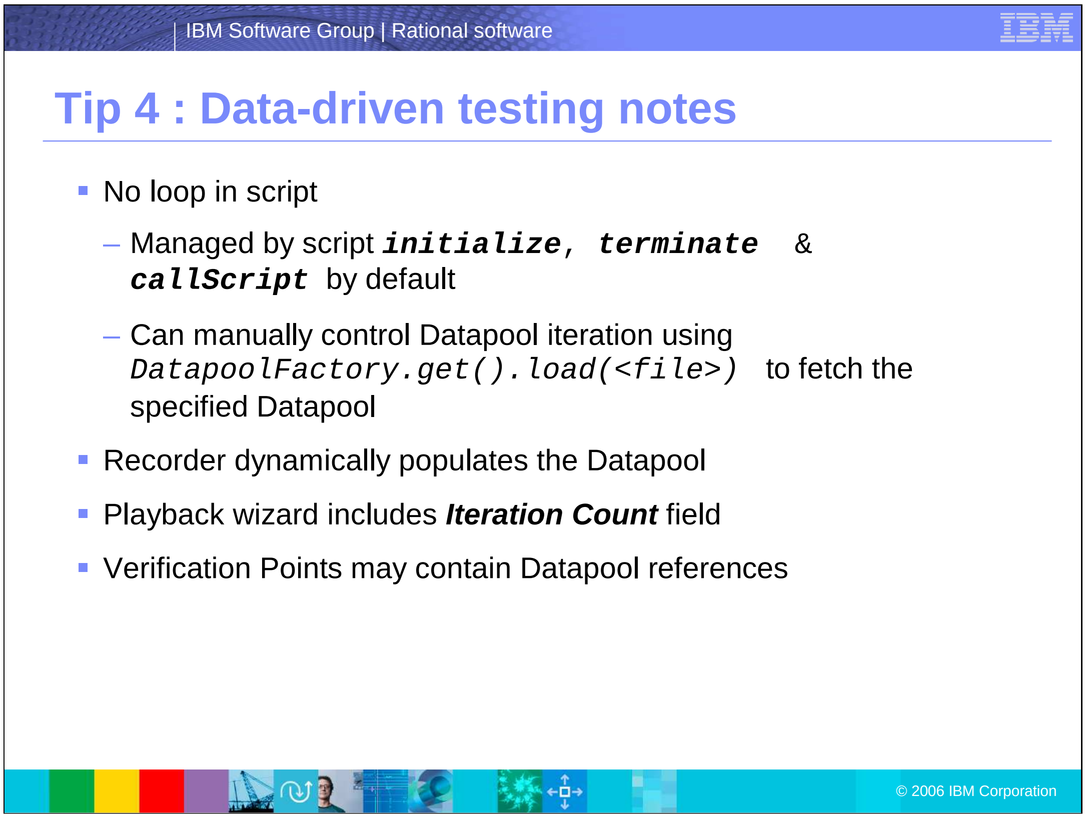
IBM Software Group | Rational software

Tip 4: Data-driven testing

- Scripts already have a Datapool asset
  - Private – associated with one script
  - Public – associated with zero or more scripts
- Population of a Datapool
  - From a CSV file when created
  - The CSV can be from existing Test Manager Datapool

Tip 4: Data-Driven Testing

Rational Functional Tester has a data-driving capability that is highly managed to make it easier for you to utilize. Scripts in Rational Functional Tester already have a Datapool asset: either private or public. You can manage the population of a Datapool from a CSV file when created, or, the CSV can come from an existing Test Manager Datapool.



IBM Software Group | Rational software

Tip 4 : Data-driven testing notes

- No loop in script
  - Managed by script ***`initialize`***, ***`terminate`*** & ***`callScript`*** by default
  - Can manually control Datapool iteration using *`DatapoolFactory.get().load(<file>)`* to fetch the specified Datapool
- Recorder dynamically populates the Datapool
- Playback wizard includes ***Iteration Count*** field
- Verification Points may contain Datapool references

© 2006 IBM Corporation

While the new datapool support is easy to use, it is not as easy to control the iteration through the datapool. However, you can still invoke a Test Manager-like datapool iteration if you choose.

Some items to be aware of include:

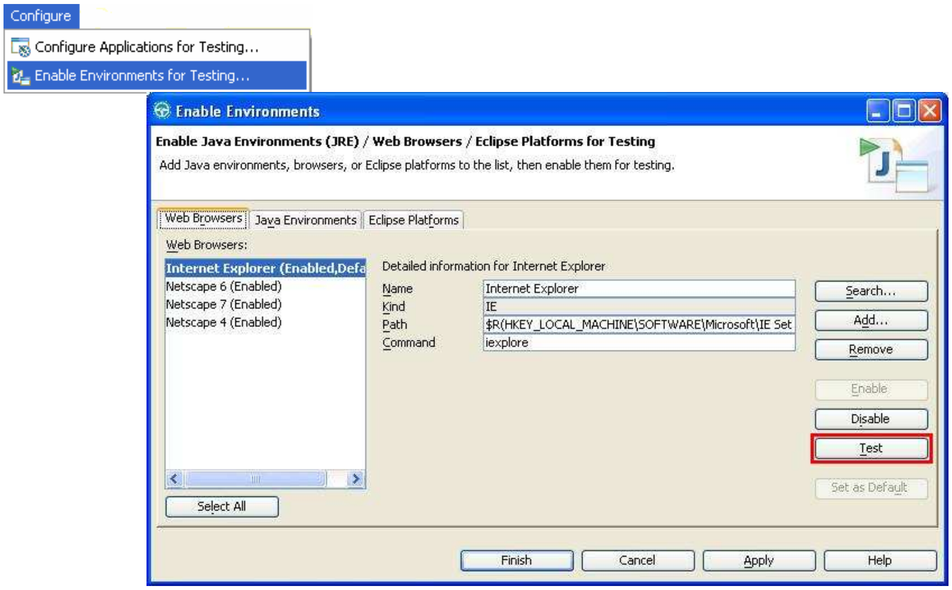
Iteration control is available on the playback wizard for the top level script or on the callScript command for nested scripts.

The recorder dynamically populates the Datapool

The playback wizard includes ***Iteration Count*** field

Verification points may contain Datapool.

Tip 5 : *Test* your enablement



Enable Java Environments (JRE) / Web Browsers / Eclipse Platforms for Testing

Add Java environments, browsers, or Eclipse platforms to the list, then enable them for testing.

Web Browsers:

Internet Explorer (Enabled,Defa
Netscape 6 (Enabled)
Netscape 7 (Enabled)
Netscape 4 (Enabled)

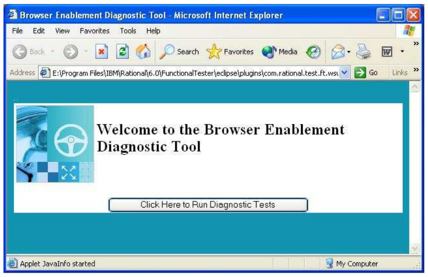
Detailed information for Internet Explorer

Name	Internet Explorer
Kind	IE
Path	$R(HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\IE Set
Command	iexplore

Tip 5 : ***Test*** Your Enablement

This section shows you how to enable Web browsers to allow Rational Functional Tester to infest in HTML domain applications. From the Configure menu, select “Enable Environment for Testing.” From the dialog box, click the Search button to locate browser candidates. Select the browser you want to test and click the Test button.

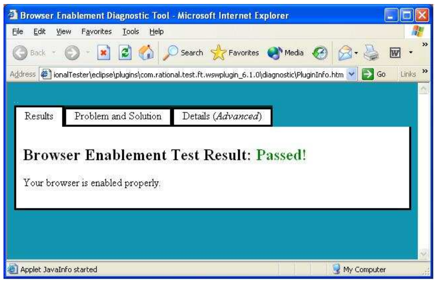
Tip 5 : *Test* your enablement


Browser Enablement Diagnostic Tool - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Go
Links
Click Here to Run Diagnostic Tests
My Computer

Address E:\Program Files\IBM\Rational\6.0\FunctionalTester\eclipse\plugins\com.rational.test.ft.wsv

Welcome to the Browser Enablement Diagnostic Tool

Applet JavaInfo started

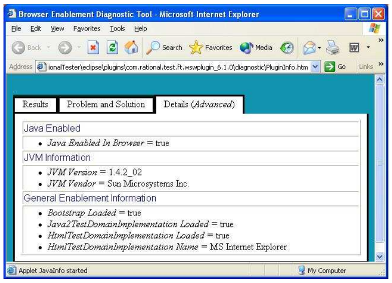


Address ionalTester\eclipse\plugins\com.rational.test.ft.wswplugin_6.1.0\diagnostic\PluginInfo.htm

Browser Enablement Test Result: Passed!

Your browser is enabled properly.

Applet JavaInfo started



Address ionalTester\eclipse\plugins\com.rational.test.ft.wswplugin_6.1.0\diagnostic\PluginInfo.htm

Java Enabled

- *Java Enabled In Browser* = true

JVM Information

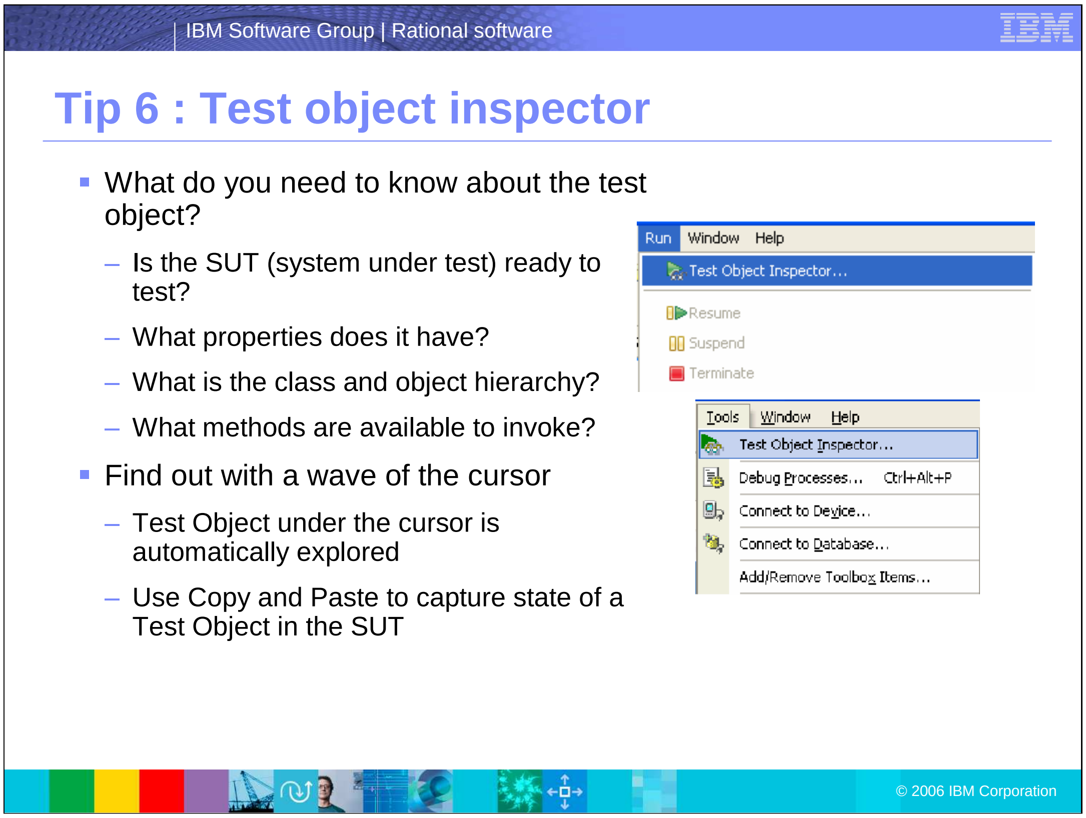
- *JVM Version* = 1.4.2_02
- *JVM Vendor* = Sun Microsystems Inc.

General Enablement Information

- *Bootstrap Loaded* = true
- *Java2TestDomainImplementation Loaded* = true
- *HtmlTestDomainImplementation Loaded* = true
- *HtmlTestDomainImplementation Name* = MS Internet Explorer

Applet JavaInfo started

Use the Browser Enablement Diagnostic tool to test the browser you selected. Once complete, you will see a Passed! Or Failed! Message.

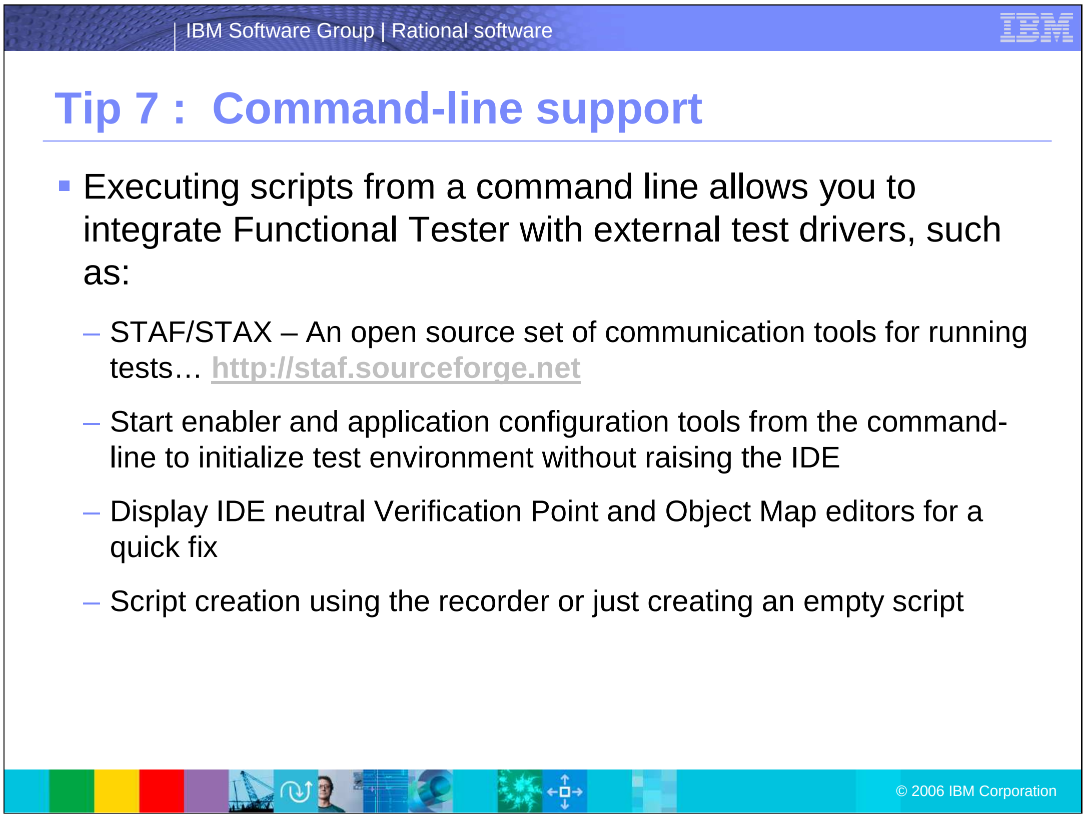


Tip 6 : Test object inspector

- What do you need to know about the test object?
  - Is the SUT (system under test) ready to test?
  - What properties does it have?
  - What is the class and object hierarchy?
  - What methods are available to invoke?
- Find out with a wave of the cursor
  - Test Object under the cursor is automatically explored
  - Use Copy and Paste to capture state of a Test Object in the SUT

Tip 6 : Test Object Inspector

Test Object Inspector opens up a view when you hover over an application under test, the component being moused over will be displayed in the Object Inspector view. Through this inspector, you can find information such as, properties, classes and object hierarchy, methods, and so on. Advanced users can take advantage of the ability to view method signatures.



IBM Software Group | Rational software

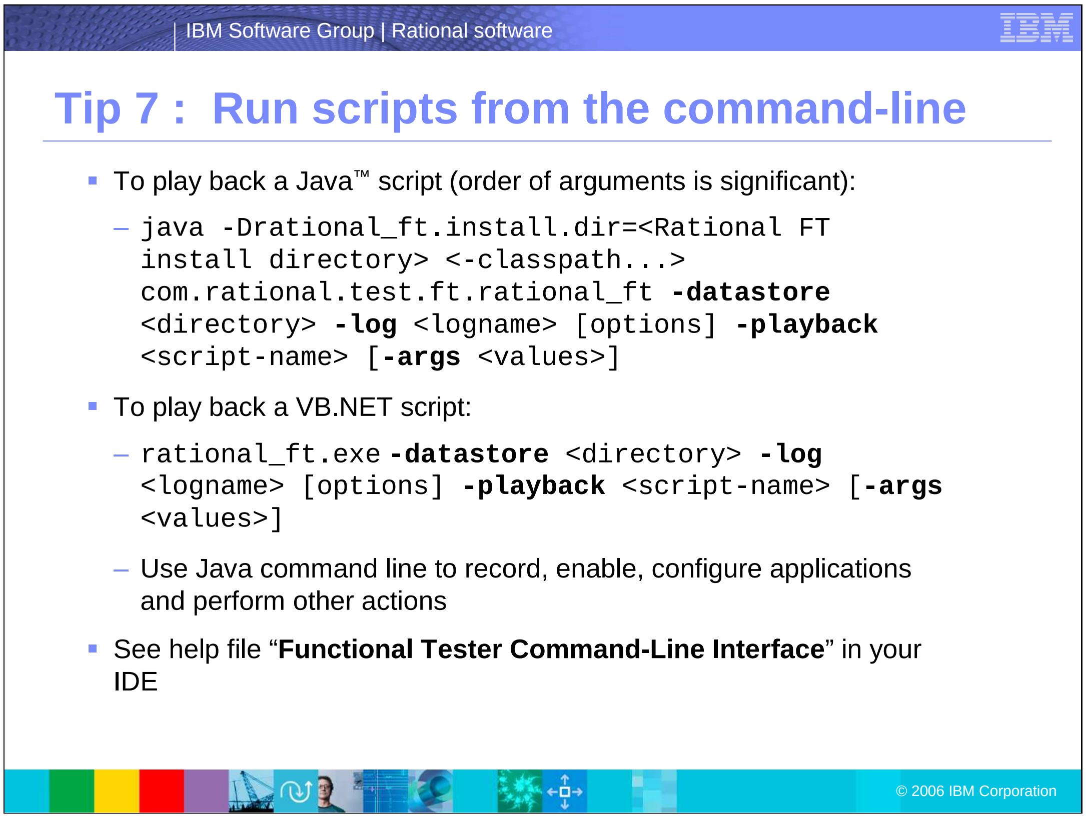
Tip 7 : Command-line support

- Executing scripts from a command line allows you to integrate Functional Tester with external test drivers, such as:
  - STAF/STAX – An open source set of communication tools for running tests… http://staf.sourceforge.net
  - Start enabler and application configuration tools from the command-line to initialize test environment without raising the IDE
  - Display IDE neutral Verification Point and Object Map editors for a quick fix
  - Script creation using the recorder or just creating an empty script

© 2006 IBM Corporation

Tip 7: Run scripts from the command line

Rational Functional Tester allows you to run scripts from a command line. This function gives you the ability to integrate Rational Functional Tester with external test drivers, such as STAF and STAX.



IBM Software Group | Rational software

Tip 7 : Run scripts from the command-line

- To play back a Java™ script (order of arguments is significant):

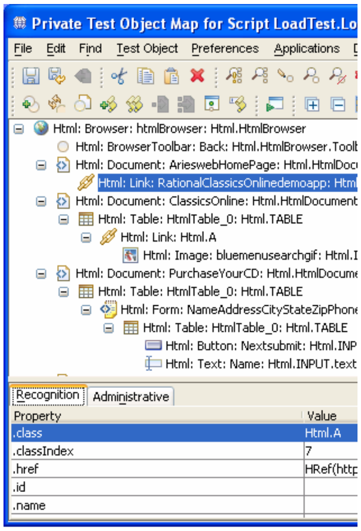
  -
    ```
    java -Drational_ft.install.dir=<Rational FT
    install directory> <-classpath...>
    com.rational.test.ft.rational_ft -datastore
    <directory> -log <logname> [options] -playback
    <script-name> [-args <values>]
    ```
- To play back a VB.NET script:

  -
    ```
    rational_ft.exe -datastore <directory> -log
    <logname> [options] -playback <script-name> [-args
    <values>]
    ```
  - Use Java command line to record, enable, configure applications and perform other actions
- See help file “**Functional Tester Command-Line Interface**” in your IDE

You can actually do a lot more than just run scripts, although that is probably the primary use of the Command line interface. You can use the sample commands shown here to play back Java™ or VB.NET scripts. You can find more information and examples in the Functional Tester Command Line interface help in your IDE.

What is an Object Map?

- Static Hierarchical Representation of the SUT.
  - Static
  - Hierarchical
  - Representation
- Two types of relationships in Map
  - Parent/Child
  - Owner/Owned



Private Test Object Map for Script LoadTest.Lo

```
Html: Browser: htmlBrowser: Html.HtmlBrowser
    Html: BrowserToolbar: Back: Html.HtmlBrowser.Toolb
  Html: Document: ArieswebHomePage: Html.HtmlDocu
      Html: Link: RationalClassicsOnlinedemoapp: Html
  Html: Document: ClassicsOnline: Html.HtmlDocument
    Html: Table: HtmlTable_0: Html.TABLE
      Html: Link: Html.A
        Html: Image: bluemenusearchgif: Html.I
  Html: Document: PurchaseYourCD: Html.HtmlDocume
    Html: Table: HtmlTable_0: Html.TABLE
      Html: Form: NameAddressCityStateZipPhone
        Html: Table: HtmlTable_0: Html.TABLE
          Html: Button: Nextsubmit: Html.INP
          Html: Text: Name: Html.INPUT.text
```

Property	Value
.class	Html.A
.classIndex	7
.href	HRef(http
.id	
.name	

Object maps are static, hierarchical representations of the system under test.

•Static means that the object map includes all relevant TestObjects in the SUT and is not timing-sensitive. There is no TestObject interaction data at this time.

-Hierarchical means that the object map follows a strict hierarchy. There are no cyclic dependencies or any indirect associations.

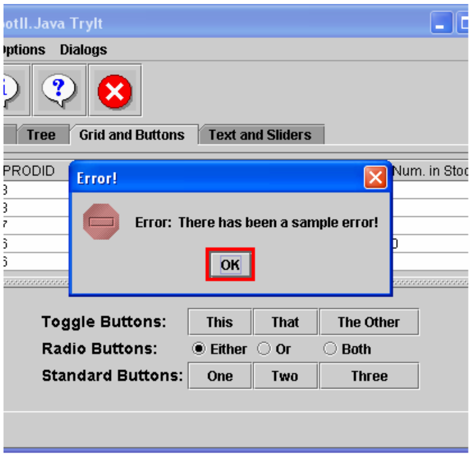
-Representation means that the object map maintains recognition properties that “describe” each TestObject.

There are two types of relationships in Map: Parent/Child and Owner/Owned.

> Parent/Child is a containership relationship. For GUI Test Objects, this means the parent includes the coordinate space of the child.
>
> Owner/Owned is a non-contained relationship, for objects such as a dialog to the parent window.

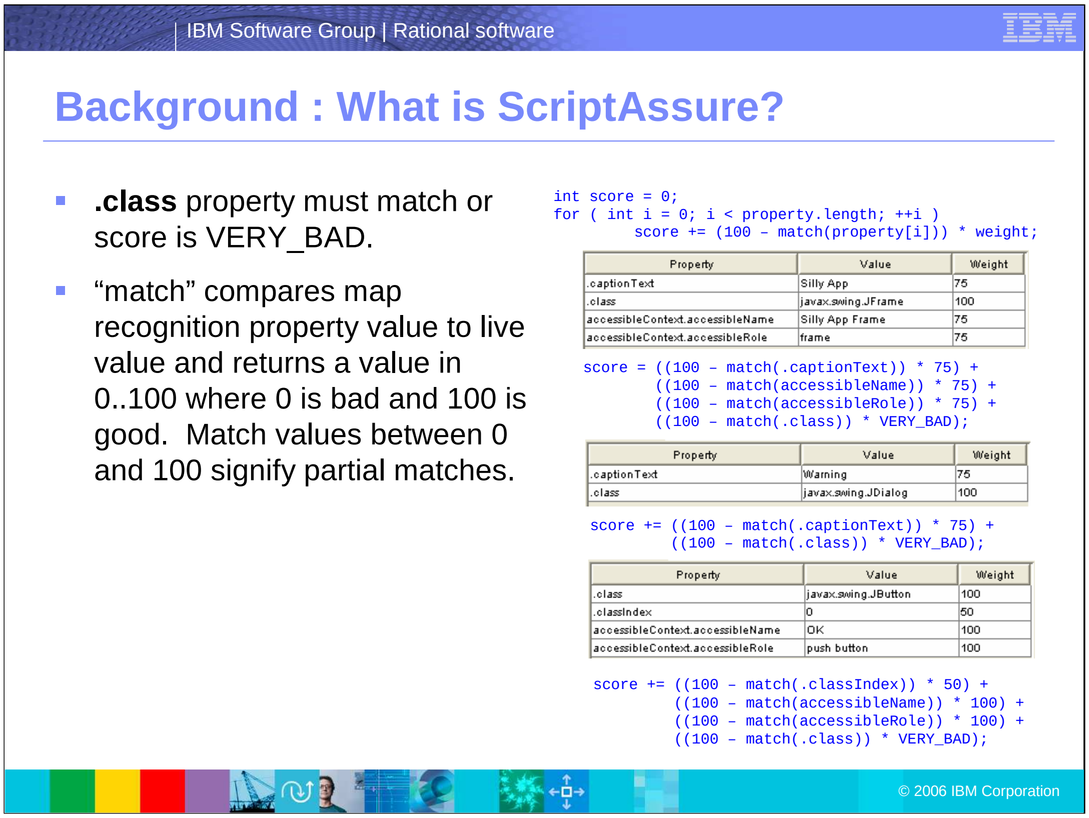
What is ScriptAssure?

- ScriptAssure is just the fuzzy matching logic associated with the **Find** algorithm
- Scores are cumulative



Error: There has been a sample error!

ScriptAssure is just the fuzzy matching logic associated with the find algorithm in Functional tester. Like in Golf, the scores are cumulative for the entire game, and the lowest score wins. Here, a zero is a perfect score.



IBM Software Group | Rational software

Background : What is ScriptAssure?

- **.class** property must match or score is VERY_BAD.
- “match” compares map recognition property value to live value and returns a value in 0..100 where 0 is bad and 100 is good. Match values between 0 and 100 signify partial matches.

```
int score = 0;
for ( int i = 0; i < property.length; ++i )
        score += (100 – match(property[i])) * weight;
```

Property	Value	Weight
.captionText	Silly App	75
.class	javax.swing.JFrame	100
accessibleContext.accessibleName	Silly App Frame	75
accessibleContext.accessibleRole	frame	75

```
score = ((100 – match(.captionText)) * 75) +
        ((100 – match(accessibleName)) * 75) +
        ((100 – match(accessibleRole)) * 75) +
        ((100 – match(.class)) * VERY_BAD);
```

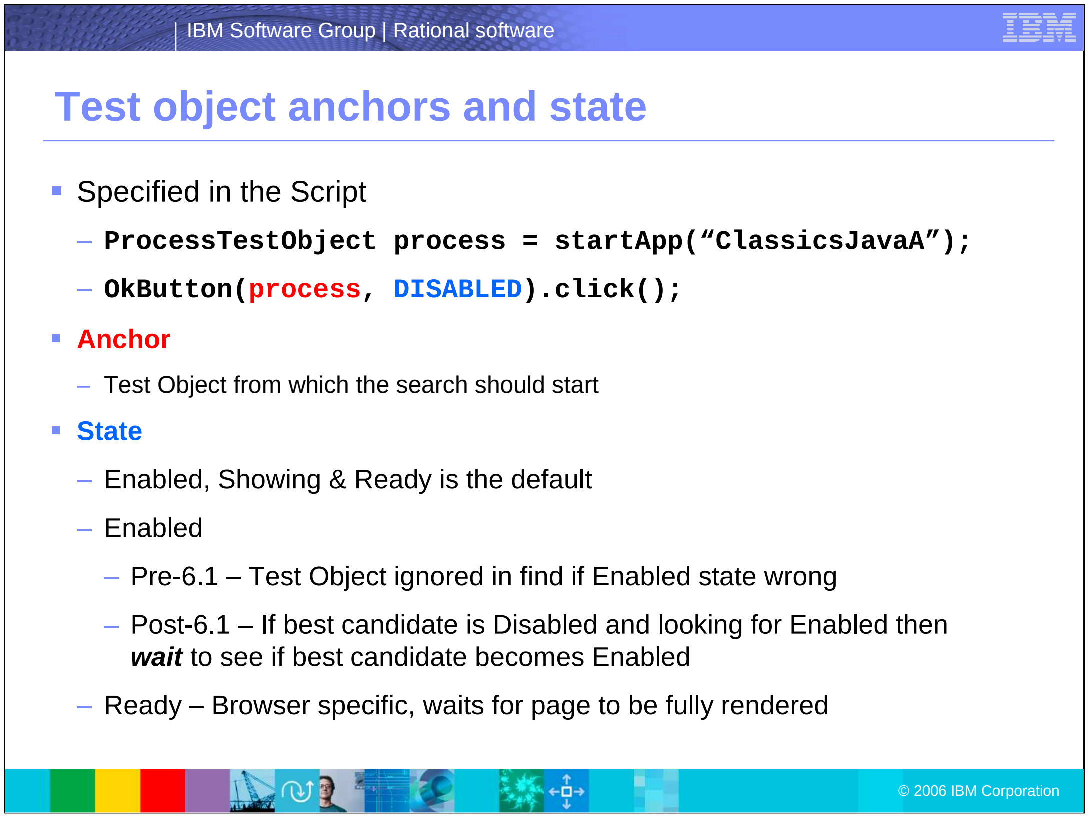
Property	Value	Weight
.captionText	Warning	75
.class	javax.swing.JDialog	100

```
score += ((100 – match(.captionText)) * 75) +
         ((100 – match(.class)) * VERY_BAD);
```

Property	Value	Weight
.class	javax.swing.JButton	100
.classIndex	0	50
accessibleContext.accessibleName	OK	100
accessibleContext.accessibleRole	push button	100

```
score += ((100 – match(.classIndex)) * 50) +
         ((100 – match(accessibleName)) * 100) +
         ((100 – match(accessibleRole)) * 100) +
         ((100 – match(.class)) * VERY_BAD);
```

In this figure, you can see the sample code that the thresholds set for the match properties. The **.class** property must match or score is VERY_BAD. (VERY_BAD is simply a number that will definitely push the score over any threshold settings). “match” compares map recognition property value to live value and returns a value in 0 to 100 where 0 is bad and 100 is good. Match values between 0 and 100 signify partial matches.

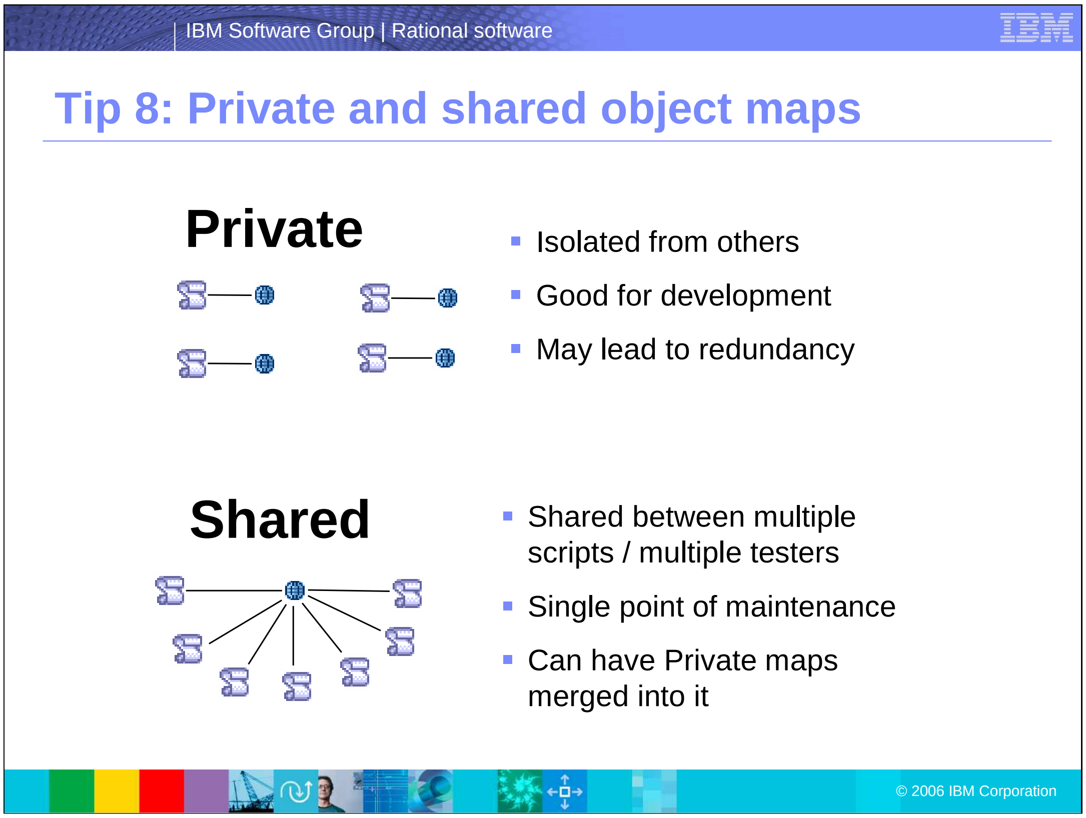


IBM Software Group | Rational software

Test object anchors and state

- Specified in the Script
  - **`ProcessTestObject process = startApp(“ClassicsJavaA”);`**
  - **`OkButton(process, DISABLED).click();`**
- **Anchor**
  - Test Object from which the search should start
- **State**
  - Enabled, Showing & Ready is the default
  - Enabled
    - Pre-6.1 – Test Object ignored in find if Enabled state wrong
    - Post-6.1 – If best candidate is Disabled and looking for Enabled then ***wait*** to see if best candidate becomes Enabled
  - Ready – Browser specific, waits for page to be fully rendered

Test Object anchors and state are specified in the script as shown above. The Anchor (shown in red) is the test object from which the search should start. You could use ProcessTestObject as the anchor as a starting point with anchors. As for the state, which is shown in blue, the default state is enabled, showing, and ready. When enabled, depending on which version of Rational Functional Tester you are using, you can see the options here.



IBM Software Group | Rational software

Tip 8: Private and shared object maps

Private

- Isolated from others
- Good for development
- May lead to redundancy

Shared

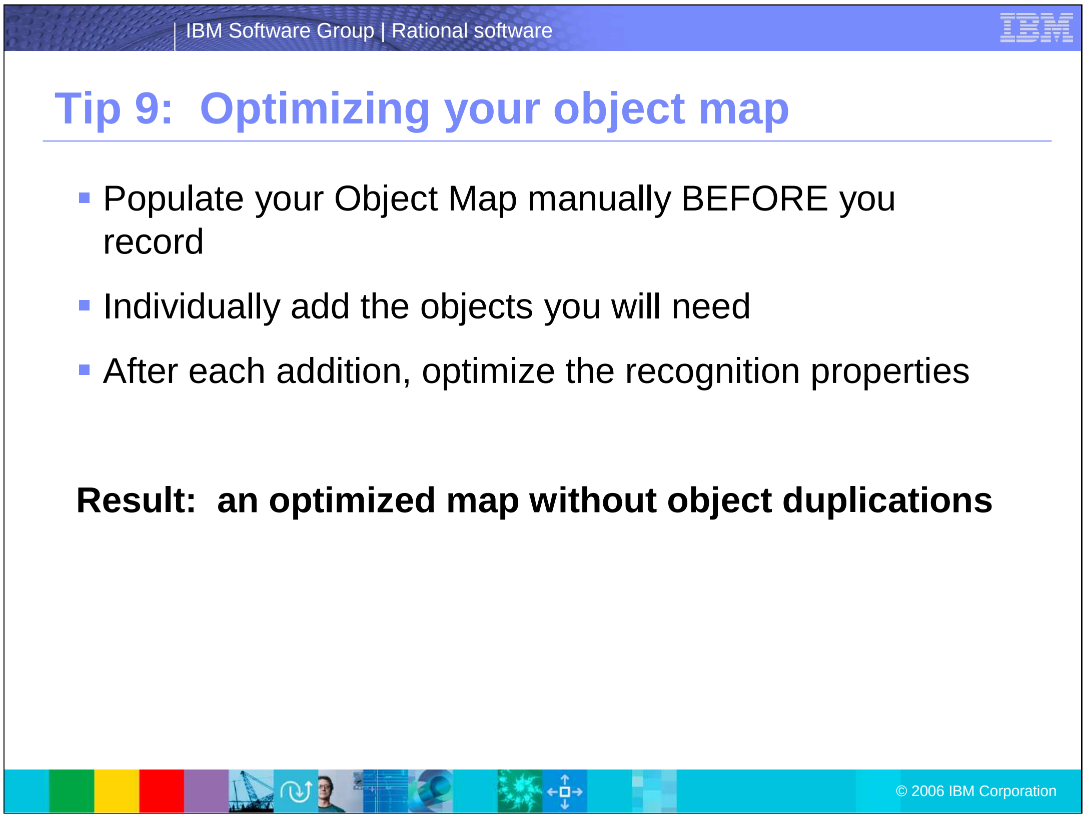
- Shared between multiple scripts / multiple testers
- Single point of maintenance
- Can have Private maps merged into it

© 2006 IBM Corporation

A private object map is used by one and only one script. Private maps provide good isolation from other scripters in that changes one person makes to the map of one script do not impact others. If multiple scripts interact with the same set of objects, those objects will appear in the object maps of multiple scripts, which leads to redundancy and higher maintenance. When an object's properties change, each instance of that object in every private map must be updated individually.

Shared maps enable multiple scripts to use a single map. A shared map provides a single point of maintenance where object property changes only need to be done in one place. That is a double-edged sword, however. Because a map can be shared among multiple scripters, it is possible that multiple scripters will want to edit the map at the same time. Use of ClearCase for SCM – or at least a well defined change management process is a requirement for using shared maps.

Your team will probably use some combination of Private and Shared maps. Different teams will use different strategies and you can always merge private maps into shared maps. One strategy is to use more private maps early in a project and as things begin to stabilize, merge them into shared maps.



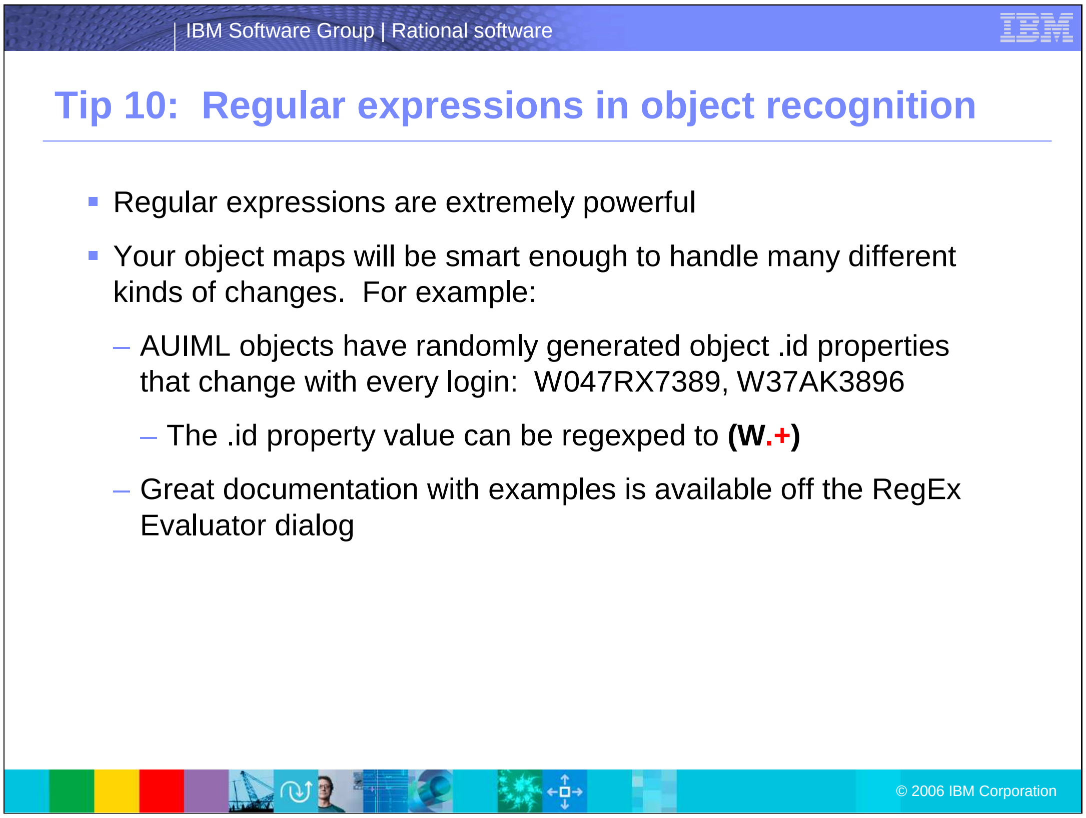
Tip 9: Optimizing your object map

- Populate your Object Map manually BEFORE you record
- Individually add the objects you will need
- After each addition, optimize the recognition properties

Result: an optimized map without object duplications

Tip 9: Optimizing your object map.

Instead of populating your object map automatically when you record a script, consider populating it manually BEFORE you record. Using the “Insert Test Object into Object Map” tool, individually add the objects you will need to interact with to the object map. After each addition, view the object map and optimize the recognition properties. Your new additions to the map will produce an optimized map without object duplications.



IBM Software Group | Rational software

Tip 10: Regular expressions in object recognition

- Regular expressions are extremely powerful
- Your object maps will be smart enough to handle many different kinds of changes. For example:
  - AUIML objects have randomly generated object .id properties that change with every login: W047RX7389, W37AK3896
    - The .id property value can be regexped to **(W.+)**
  - Great documentation with examples is available off the RegEx Evaluator dialog

© 2006 IBM Corporation

Tip 10: Regular Expressions in Object Recognition

The use of regular expressions in the property value fields of your object map allows you to handle changes in the AUT without changing your object map. For example:

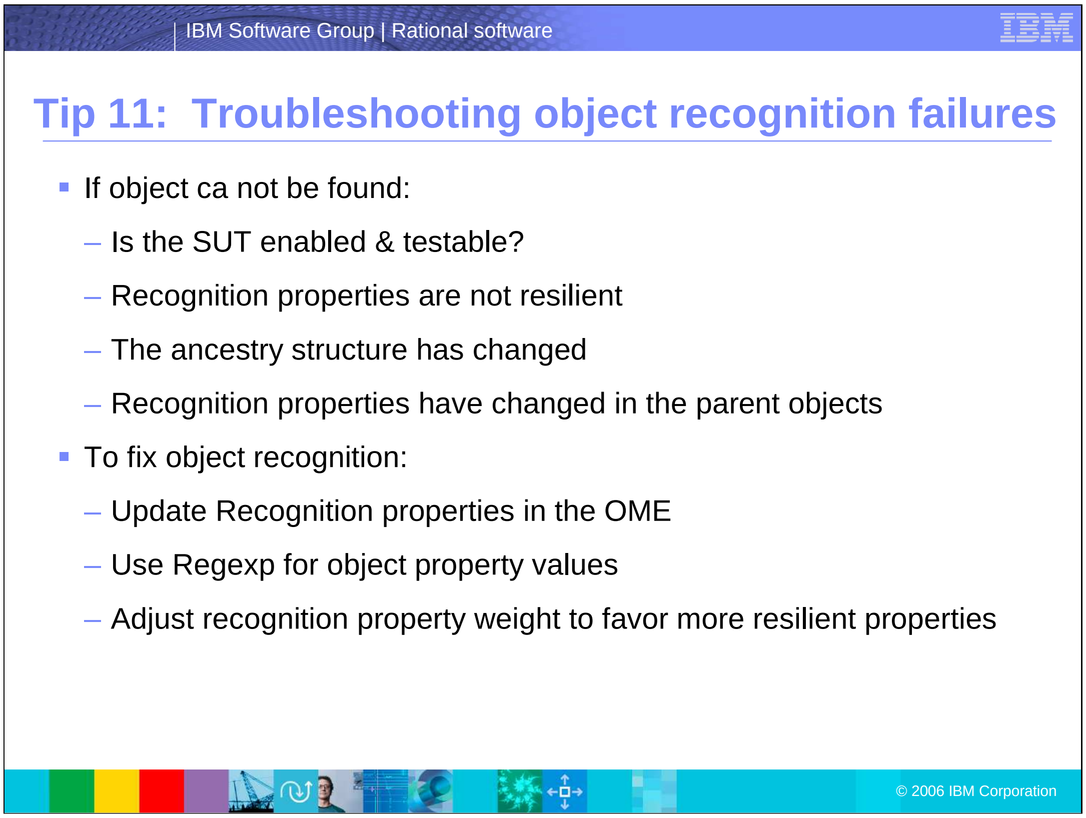
AUIML objects have randomly generated object .id properties that change with every login as shown above.

The .id property value can be regexped to **(W.+)**

Secure websites often have randomly generated session ids that are contained in the URLs you access inside a site

The URL values can be regexped to

http://hostname/index.htmlSID=[0-9]+



IBM Software Group | Rational software

Tip 11: Troubleshooting object recognition failures

- If object ca not be found:
  - Is the SUT enabled & testable?
  - Recognition properties are not resilient
  - The ancestry structure has changed
  - Recognition properties have changed in the parent objects
- To fix object recognition:
  - Update Recognition properties in the OME
  - Use Regexp for object property values
  - Adjust recognition property weight to favor more resilient properties

Tip 11: Troubleshooting Object Recognition Failures

When an object cannot be found in the AUT, either:

Property-value pairs have changed in the object itself, the ancestry structure has changed, or property-value pairs have changed in the parent object or objects.

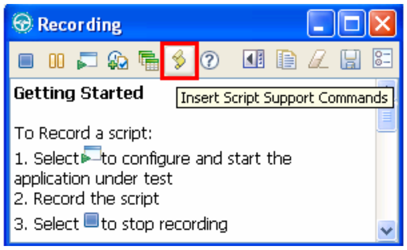
To fix an ancestry structure change you should re-add the object to the map and delete the old entry.

To fix object property-value change:

If it is a one-time change, re-add the object to the map and delete the old entry

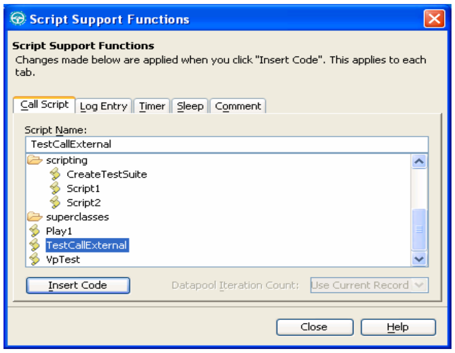
If there are ongoing changes, use Regexp to change object property values

Tip 12: Picking arguments - *Passing data with callScript*



Getting Started

To Record a script:
1. Select to configure and start the application under test
2. Record the script
3. Select to stop recording



Script Support Functions

Changes made below are applied when you click "Insert Code". This applies to each tab.

Script Name: TestCallExternal

Datapool Iteration Count: Use Current Record

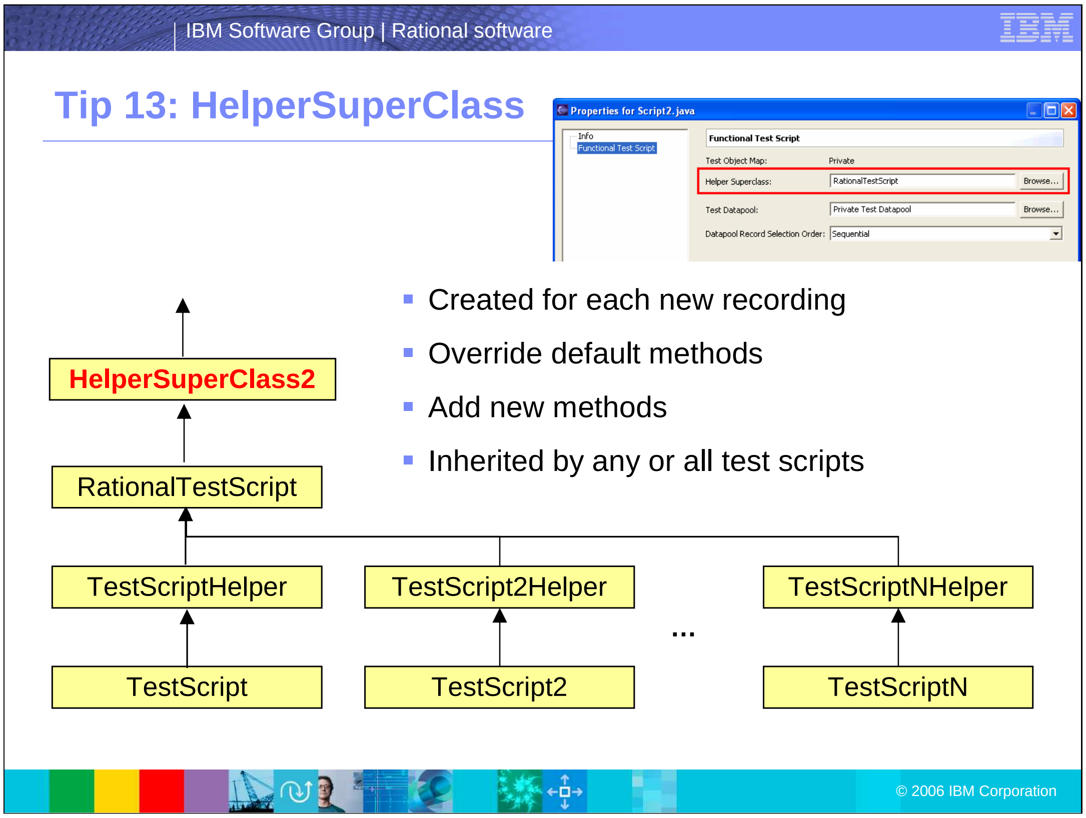
- By default, automated callScript does not include arguments
- RationalTestScript API overloads callScript with arguments

```
callScript("TableHyperlinks");
```

```
protected java.lang.Object callScript(
            java.lang.String scriptFullName,
            java.lang.Object[] args)
```

Tip 12: Picking Arguments

To insert script support commands, use the button as illustrated to view the script support panels shown here. By doing this, you can choose the script to be inserted into the main script. By default, automated callScript does not include arguments. RationalTestScript API overloads callScript with arguments you pass in as a parameter when invoking the callScript.



IBM Software Group | Rational software

Tip 13: HelperSuperClass

Properties for Script2.java

Functional Test Script

Test Object Map: Private

Helper Superclass: RationalTestScript

Test Datapool: Private Test Datapool

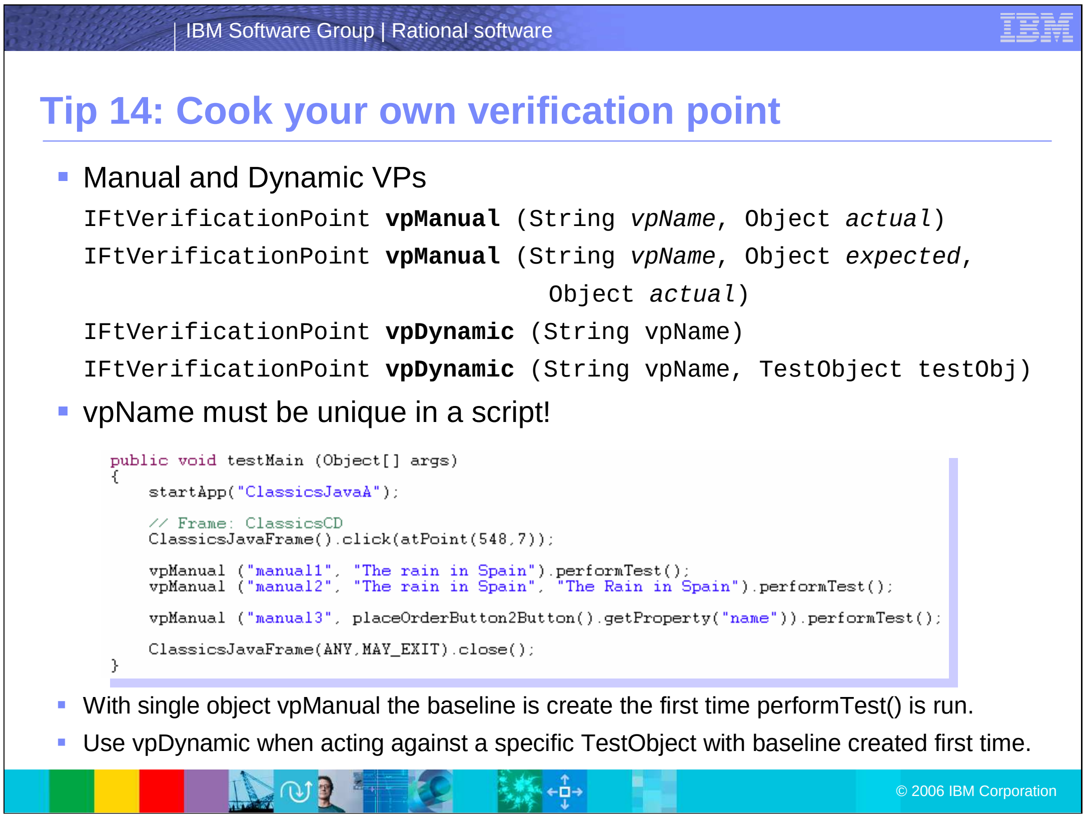
Datapool Record Selection Order: Sequential

- Created for each new recording
- Override default methods
- Add new methods
- Inherited by any or all test scripts

© 2006 IBM Corporation

Tip 13: HelperSuperClass

Rational Functional Tester creates a HelperSuperClass for each new recording. By default, all Functional Tester scripts extend the RationalTestScript class, and thereby inherit a number of methods, such as callScript. HelperSuperClass overrides default methods and adds new methods from RationalTestScript.



IBM Software Group | Rational software

Tip 14: Cook your own verification point

- Manual and Dynamic VPs

```
IFtVerificationPoint vpManual (String vpName, Object actual)
IFtVerificationPoint vpManual (String vpName, Object expected,
                              Object actual)
IFtVerificationPoint vpDynamic (String vpName)
IFtVerificationPoint vpDynamic (String vpName, TestObject testObj)
```

- vpName must be unique in a script!

```
public void testMain (Object[] args)
{
    startApp("ClassicsJavaA");

    // Frame: ClassicsCD
    ClassicsJavaFrame().click(atPoint(548,7));

    vpManual ("manual1", "The rain in Spain").performTest();
    vpManual ("manual2", "The rain in Spain", "The Rain in Spain").performTest();

    vpManual ("manual3", placeOrderButton2Button().getProperty("name")).performTest();

    ClassicsJavaFrame(ANY,MAY_EXIT).close();
}
```

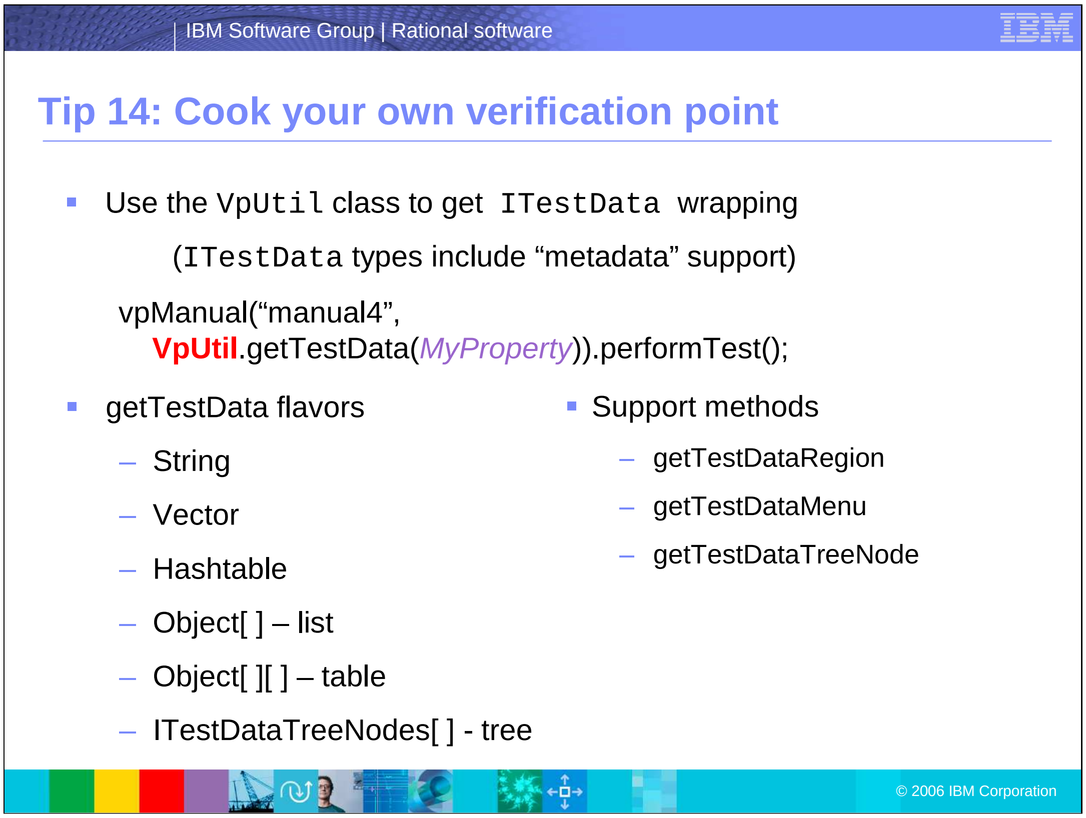
- With single object vpManual the baseline is create the first time performTest() is run.
- Use vpDynamic when acting against a specific TestObject with baseline created first time.

© 2006 IBM Corporation

Tip 14: Cook Your Own Verification Point

When using verification points in Rational Functional Tester, you have two options; manual and dynamic.

A Manual verification point allows you to specify the expected data and the actual data using Rational Functional Tester API to verify. A dynamic verification point allows you to use Rational Functional Tester GUI settings to verify certain components.



IBM Software Group | Rational software

Tip 14: Cook your own verification point

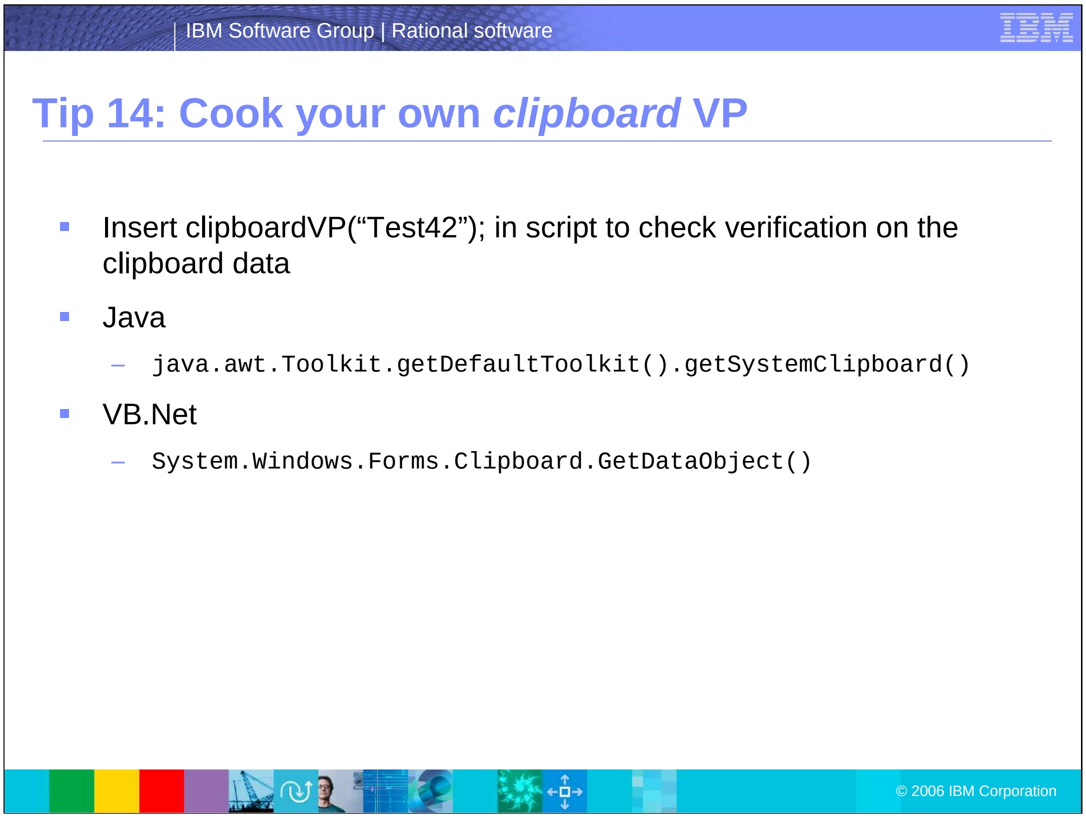
- Use the `VpUtil` class to get `ITestData` wrapping

  (`ITestData` types include “metadata” support)

```
vpManual(“manual4”,
   VpUtil.getTestData(MyProperty)).performTest();
```

- getTestData flavors
  - String
  - Vector
  - Hashtable
  - Object[ ] – list
  - Object[ ][ ] – table
  - ITestDataTreeNodes[ ] - tree
- Support methods
  - getTestDataRegion
  - getTestDataMenu
  - getTestDataTreeNode

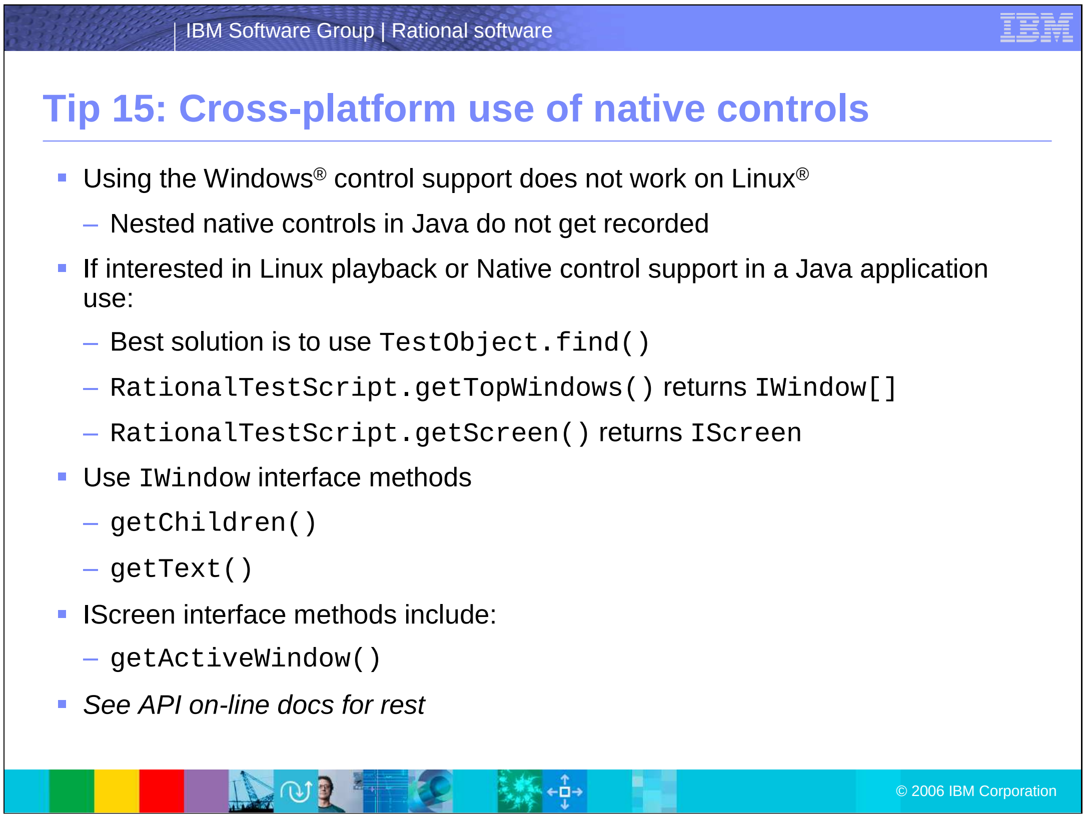
Using ITestData based types means you get metadata properties, such as the ones shown here. Even if you are not interested in ITestData today, you might want it in the future, so using VpUtil is a good pattern to set.



Tip 14: Cook your own *clipboard* VP

- Insert clipboardVP(“Test42”); in script to check verification on the clipboard data
- Java
  - `java.awt.Toolkit.getDefaultToolkit().getSystemClipboard()`
- VB.Net
  - `System.Windows.Forms.Clipboard.GetDataObject()`

You can check verification on the clipboard data by invoking the clipboardVP method. You can get the clipboard data by invoking the methods listed above.



Tip 15: Cross-platform use of native controls

- Using the Windows® control support does not work on Linux®
  - Nested native controls in Java do not get recorded
- If interested in Linux playback or Native control support in a Java application use:
  - Best solution is to use `TestObject.find()`
  - `RationalTestScript.getTopWindows()` returns `IWindow[]`
  - `RationalTestScript.getScreen()` returns `IScreen`
- Use `IWindow` interface methods
  - `getChildren()`
  - `getText()`
- IScreen interface methods include:
  - `getActiveWindow()`
- *See API on-line docs for rest*

Tip 15: Cross-Platform Use of Native Controls

You can use the IWindow and IScreen interface methods to access native controls. Native control support does not work on Linux® platforms.

IBM Software Group | Rational software

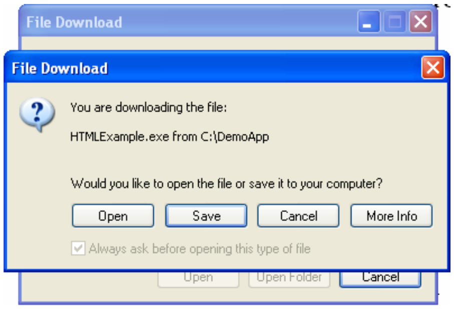
Tip 15 : Example of file download dialog

```
    IWindow[] children = getScreen().getActiveWindow().getChildren();

    for (int i = 0; i < children.length; ++i) {
        if (children[i].getText().equals("&Open")) {
            children[i].click();
            break;
        }
    }
```

Get children of active window

Find object with text “&Open” and click it



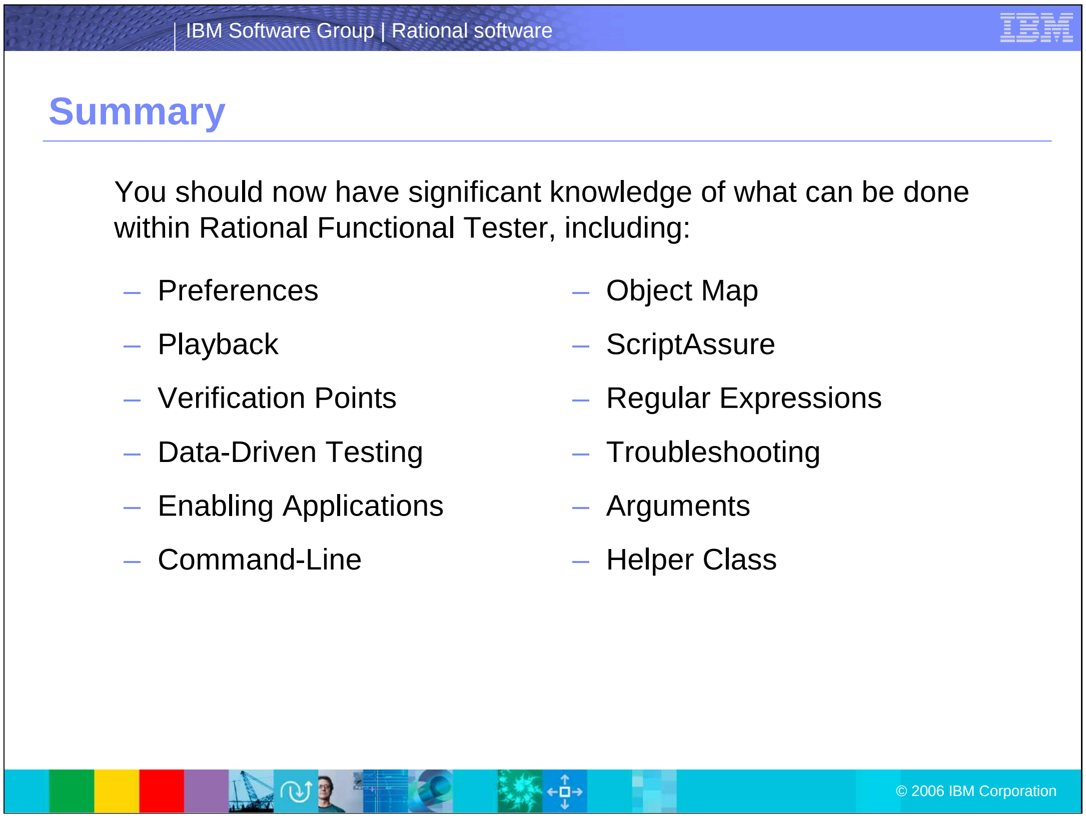
File Download

You are downloading the file:

HTMLExample.exe from C:\DemoApp

Would you like to open the file or save it to your computer?

Here is an example of the File Download dialog box. You can see callouts for the Find object and the getChildren methods.



Summary

You should now have significant knowledge of what can be done within Rational Functional Tester, including:

- Preferences
- Playback
- Verification Points
- Data-Driven Testing
- Enabling Applications
- Command-Line
- Object Map
- ScriptAssure
- Regular Expressions
- Troubleshooting
- Arguments
- Helper Class

In summary, this presentation provided you with a good understanding of what can be done within Rational Functional Tester, including setting preferences, playbacks, verification points, and the other items on this list.   For more information, reference the Rational Functional Tester API from the help menu.

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
e(logo)business
AIX
CICS
DB2
DB2 Universal Database
IBM(logo)
iSeries
Rational
ClearCase
OS/400
ScriptAssure
OS/390
xSeries
zSeries

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

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

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

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

Linux is a registered trademark of Linus Torvalds.

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

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 and/or changes in the product(s) and/or program(s) 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 (e.g., 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.

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.