Introduction to IBM Rational Functional Tester 6.1

Mike Kelly (Mike@MichaelDKelly.com)

First published on IBM developerWorks: http://www-130.ibm.com/developerworks/

IBM® Rational® Functional Tester is an object-oriented automated testing tool that lets you test a variety of applications. You can quickly generate scripts by recording tests against an application, and you can test any object in the application, including the object's properties and data. Rational Functional Tester offers you a choice of scripting language and development environment -- Java™ in the Eclipse® framework or Microsoft® Visual Basic® .NET in the Microsoft Visual Studio® .NET Development Environment. That means that regardless of the language or platform your development staff has chosen, you should be able to integrate with them and leverage some of their expertise as you develop your automated tests.

In this article, we'll take a first look at Rational Functional Tester as it's implemented in both environments. For each version of the tool, we'll explore the user interface and then record and run a script. After a brief comparison of the scripts generated in each environment, I'll give you tips on where you can go for more help.

Editor's Note: This article is based on Version 6.0.0 of the IBM Software Development Platform, Version 7.1.3088 of the Microsoft Development Environment 2003, and Windows XP Professional SP2.

A first look at basic features

Basic to Rational Functional Tester are its object technology and its record-and-playback functionality for Java, .NET, and Web-based applications. The tool provides testers with automated capabilities for activities such as data-driven testing.

When you record a script, Rational Functional Tester automatically creates a test object map for the application-under-test. The object map contains recognition properties for each object. When you update the recognition information in the object map, any scripts that use this object map share the updated information, reducing the cost of maintenance and the overall complexity of script development. The object map also provides a quick way to add objects to a script. It lists the test objects available in the application, whether they're currently displayed or not. You can create a new test object map by either basing it on an existing map or adding objects as required.

During recording you can insert verification points into the script to confirm the state of an object across builds of the application-under-test. The verification point captures object information (based on the type of verification point, which can be an object properties verification point or one of five types of data verification points -- menu hierarchy, table, text, tree hierarchy, or list) and stores it in a baseline data file. The information in this file becomes the expected state of the object during subsequent builds. After a test executes, you can use the Verification Point Comparator to analyze differences or update the baseline (expected object state) if the behavior of the object changes.

Rational Functional Tester also offers these powerful capabilities:

- **Play back scripts against an updated application**. The ScriptAssure feature, which is Rational Functional Tester's object recognition technology, enables you to successfully play back scripts even when the application-under-test has been updated. You can set thresholds for recognition scores that a test object must pass to be considered as a candidate for recognition, and you can also write warnings to log files if Rational Functional Tester accepts a candidate with a score higher than the designated threshold.
- **Update recognition properties for an object**. In the test object map, you can update the recognition properties for a selected test object. Rational Functional Tester displays the

Update Recognition Properties page, which shows the updated test object properties, the original recognition properties, and all the recognition properties for the object. If necessary, you can modify the updated recognition properties.

- **Merge multiple test object maps**. Object maps can be either shared or private. A private map is attached to a script and accessed only by that specific script, whereas a shared map is shared by multiple scripts. The advantage of a shared map is that when objects need to be updated, only one update to one map will fix multiple scripts. You can merge multiple private or shared test object maps into a single shared test object map from the Rational Functional Tester Projects view and when creating a new test object map. Optionally, Rational Functional Tester can update scripts you select to point to the newly merged test object map.
- **Display associated scripts**. In the test object map, you can view a list of scripts associated with the map and use this list to select multiple scripts to add test objects to.
- **Use pattern-based object recognition**. You can replace a recognition property with a regular expression or a numeric range to allow for pattern-based recognition, instead of being limited to an exact match. This allows for more flexibility in object recognition. You can convert properties to regular expressions and numeric ranges from within the Verification Point Editor or the test object map. The Regular Expression Evaluator allows you to test an expression while you're editing it, saving you the time of having to run the script to see if the pattern worked.
- **Integrate with UCM**. Rational Functional Tester works in a ClearCase single stream Unified Change Management (UCM) enabled view. Artifacts created in Rational Functional Tester can be version controlled individually.

Both versions of Rational Functional Tester share these basic features, and both generate (roughly) the same script for a given application-under-test. This is important to note because it frees you to choose whichever environment or language best fits your needs at any given time. Don't think that if you start using the Visual Basic .NET version of Rational Functional Tester and switch to a Java project six months from now, you'll need to learn an entirely different tool. You won't! Both versions are basically the same, allowing you to feel confident that any investment (of time or money) in one tool can be carried over to the other.

To demonstrate this, I'll take you through recording and running an example script in each environment and then compare the resulting scripts at the end. Follow the steps for whichever environment suits your needs at this time and take a look at the steps for the other environment just to get an idea of how similar the steps are.

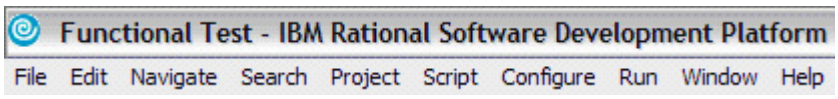
Working in the Java/Eclipse environment

Here we'll focus on using Rational Functional Tester with Java in the Eclipse framework. I'll show you the user interface and take you step by step through recording and running a script.

User interface

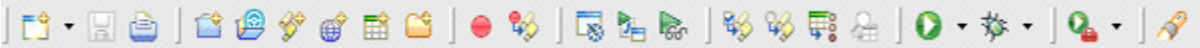
When you start Rational Functional Tester for Java, you see the Test Perspective window with eight main components: the main menu, the toolbar, the Projects view, the Java editor, the Script Explorer, the Console view, the Tasks view, and the status bar. Following is a brief description of each of these components.

Main menu



You can read about each of the main menu options in the Rational Functional Tester online help.

Toolbar



The toolbar contains these icons:

Open the New Wizard -- Displays the appropriate dialog box to create one of a number of items or to record a Functional Test script. Click to display the list of possible items to create.

Create New Functional Test Project -- Displays a dialog box that lets you generate a new project in Functional Test.

Connect to an Existing Functional Test Project -- Displays a dialog box that lets you connect to an existing project.

Create an Empty Functional Test Script -- Displays a dialog box that lets you create a script you can use to add Java code manually.

Create New Test Object Map -- Displays a dialog box that lets you add a new test object map to a project.

Create New Test Datapool -- Displays a dialog box that lets you create a new test datapool.

Create a New Test Folder -- Displays a dialog box that lets you create a new folder either for the project or for an existing folder.

Record a Functional Test Script -- Displays a dialog box that lets you enter information about the new script and start recording.

Insert Recording into Active Functional Test Script -- Starts recording at the cursor location in the current script, which enables you to start applications, insert verification points, and add script support functions.

Configure Applications for Testing -- Displays the Application Configuration tool, which enables you to add and edit configuration information -- such as name, path, and other information used to start and run the application -- for the Java and HTML applications you want to test.

Enable Environments for Testing -- Displays a dialog box that you use to enable Java environments and browsers and to configure your JREs and browsers.

Display the TestObject Inspector Tool -- Displays the TestObject Inspector tool, which enables you to display test object information such as parent hierarchy, inheritance hierarchy, test object properties, nonvalue properties, and method information.

Insert Verification Point -- Displays the Select an Object page of the Verification Point and Action Wizard, which enables you to select an object in your application that you want to test.

Insert Test Object into Active Functional Test Script -- Displays a dialog box that lets you select test objects to add to the test object map and a script.

Insert Data Driven Commands into Active Functional Test Script -- Displays the Data Drive Actions page of the Datapool Population Wizard, which enables you to select the objects in an application-under-test to data-drive an application.

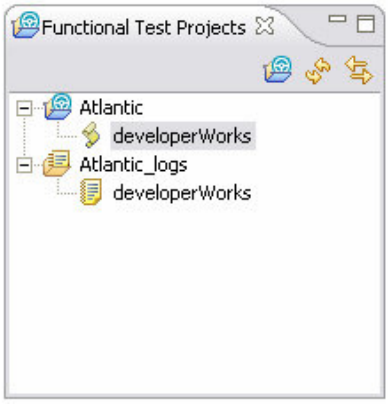
Replace Literals with Datapool Reference -- Replaces literal values with a datapool reference in a test script, which enables you to add realistic data to an existing test script.

Run Functional Test Script -- Runs your Functional Test script. Click to display the list of run commands.

Debug Functional Test Script -- Launches the current script and displays the Debug Perspective, which provides information as the script debugs. Click to begin debugging at method Main in the current script. Click to display the list of debug commands.

Run or Configure External Tools -- Enables you to configure an external tool that's not part of Workbench. Click to display the list of options.

Functional Test Projects view



The Functional Test Projects view, in the left pane of the Test Perspective window, lists test assets for each project, including these:

- Folders
- Scripts
- Shared test object maps
- Log folders
- Logs
- Java files

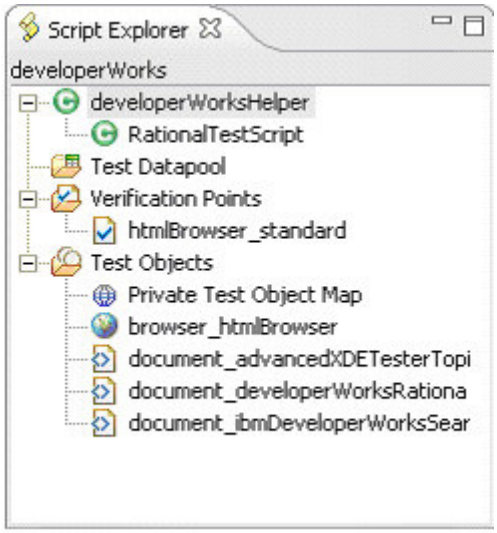
Java editor

developerWorks.java

```
import resources.developerWorksHelper;

/**
 * Description   : Functional Test Script
 * @author Administrator
 */
public class developerWorks extends developerWorksHelper
{
    /**
     * Script Name   : <b>developerWorks</b>
     * Generated     : <b>Nov 14, 2004 6:02:48 PM</b>
     * Description   : Functional Test Script
     * Original Host : WinNT Version 5.1  Build 2600 (S)
     *
     * @since  2004/11/14
     * @author Administrator
     */
    public void testMain(Object[] args)
    {
        startApp("http://www-136.ibm.com/developerworks/rational/");

        // HTML Browser
        browser_htmlBrowser(document_developerWorksRationa(),DEFAULT_FLAGS).click(atP
        browser_htmlBrowser(document_developerWorksRationa(),DEFAULT_FLAGS).inputChars
        browser_htmlBrowser(document_developerWorksRationa(),DEFAULT_FLAGS).click(atP
        browser_htmlBrowser(document_ibmDeveloperWorksSear(),DEFAULT_FLAGS).click(atP
        htmlBrowser_standardVP(document_advancedXDETesterTopi()).performTest();
        browser_htmlBrowser(document_advancedXDETesterTopi(),MAY_EXIT).close();
    }
}
```

You use the Java editor (the script window) to edit Java code. The name of the script or class you're currently editing appears on a tab in the Java editor frame. An asterisk on the left side of the tab indicates that there are unsaved changes. You can open several files in the Java editor and move between them by clicking on the appropriate tab. If there's a problem with the code while you're working in this window, a problem marker is displayed near the affected line. In addition, right-clicking in the Java editor displays various menu options for working with scripts. Other than that, it's like every other script editor.

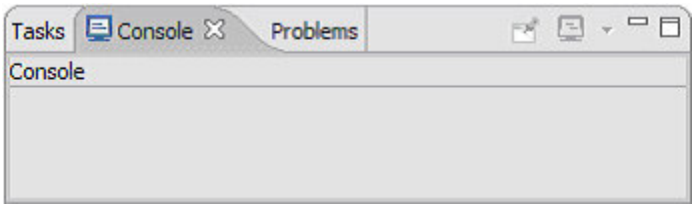
Script Explorer



The Script Explorer, in the right pane of the Test Perspective window, lists the script helper, helper superclass or helper base class, test datapool, verification points, and test objects for the current script. Note these things about the Script Explorer:

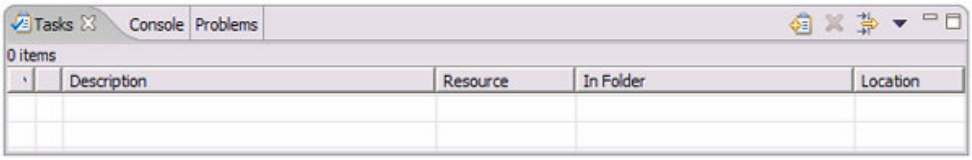
- The Verification Points folder contains all the verification points recorded for the script. Double-clicking a verification point displays the Verification Point Editor.
- The Test Objects folder contains a list of all the test objects available for the script. Each test object in the list is preceded by an icon that indicates its role. Double-clicking the Test Object Map icon displays the test object map.
- Right-clicking a verification point, test object map, or test asset listed in the Script Explorer displays various menu options.

Console view



The Console view displays output from the script or application -- for example, `System.out.print` statements or unhandled Java exceptions.

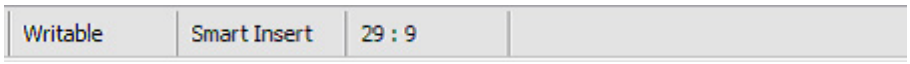
Tasks view



The Tasks view displays errors, warnings, or other information automatically generated by the compiler. By default, this view lists all tasks for all files in the project, but you may want to restrict it to

showing tasks associated with the current script. You can apply a filter by clicking the filter button in the Tasks view banner.

Status bar

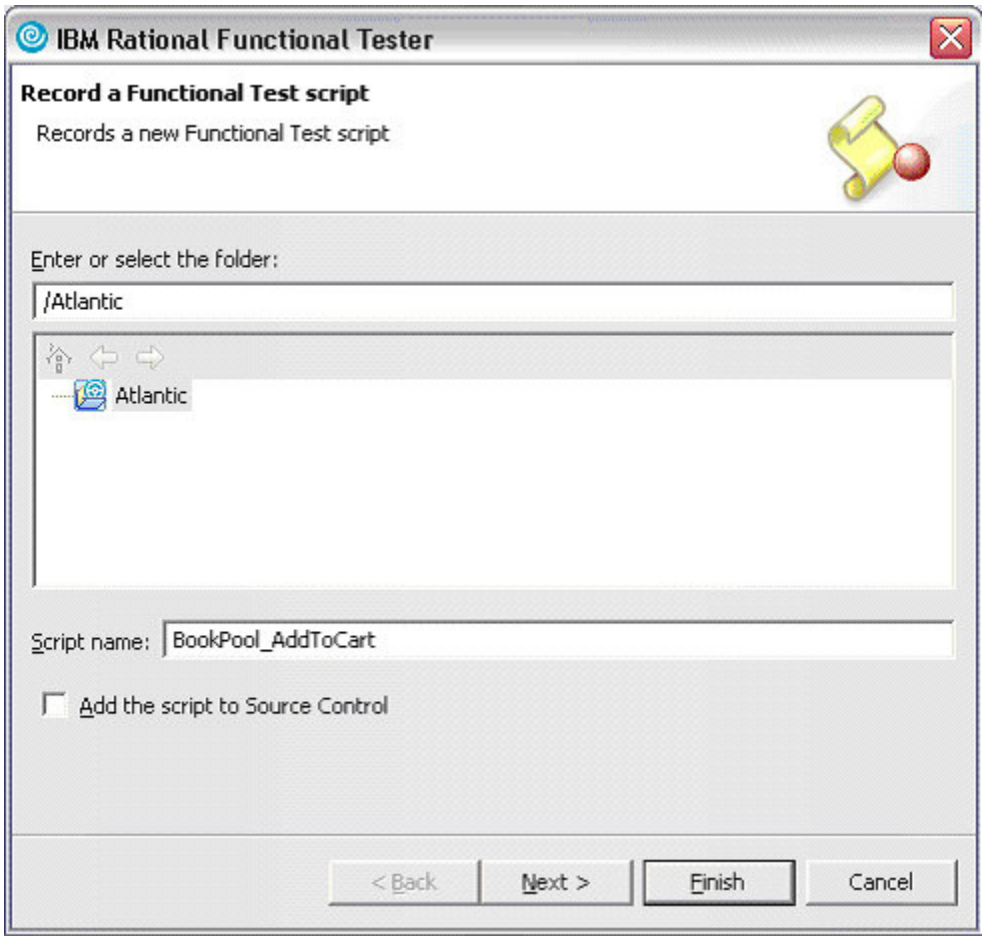


Rational Functional Tester uses the status bar at the bottom of the Test Perspective window to display messages.

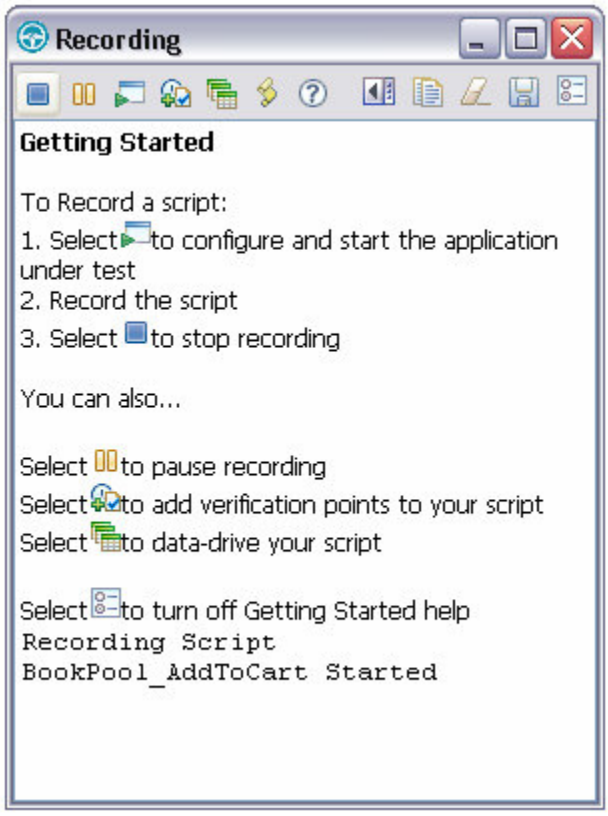
Recording a script

We'll record a script that tests adding an item to the shopping cart at BookPool.com.

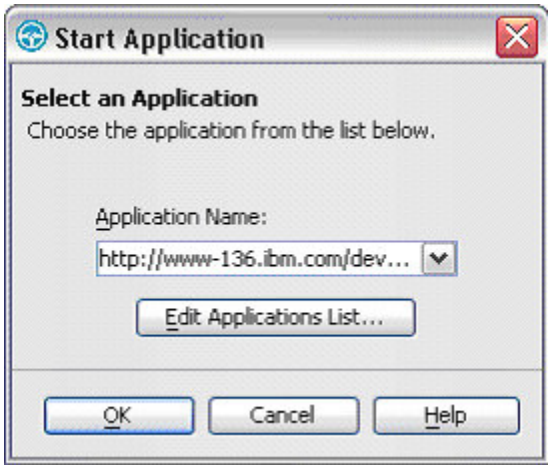
1. Start Rational Functional Tester and open an existing project.
2. Click the **Record a Functional Test Script** button and enter a name for the script.



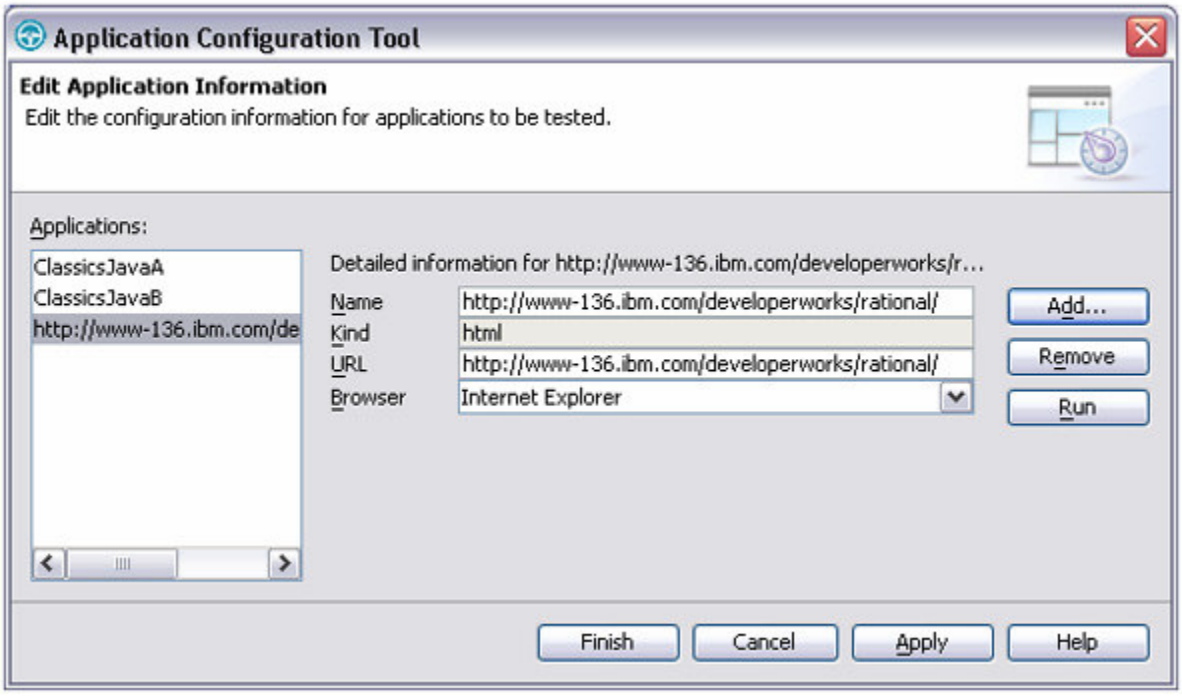
3. Click **Finish** to start recording. This will open the Recording window.



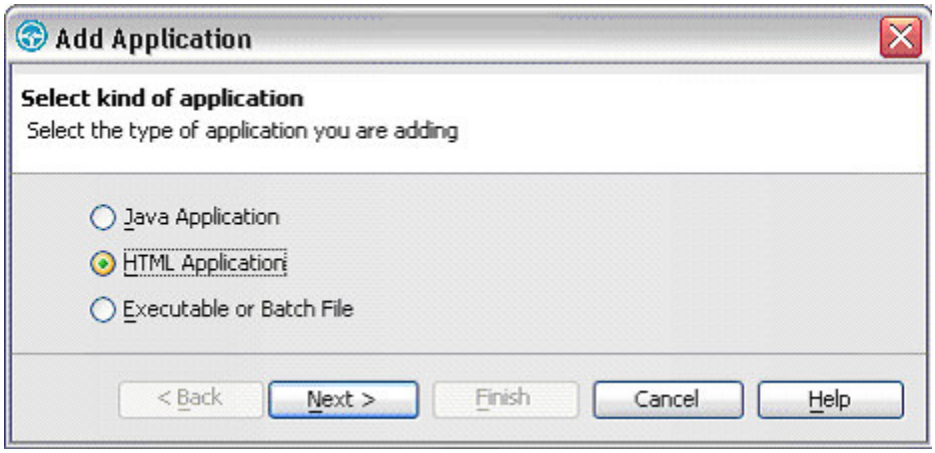
4. Click the **Start Application** button. This will open the Start Application window.



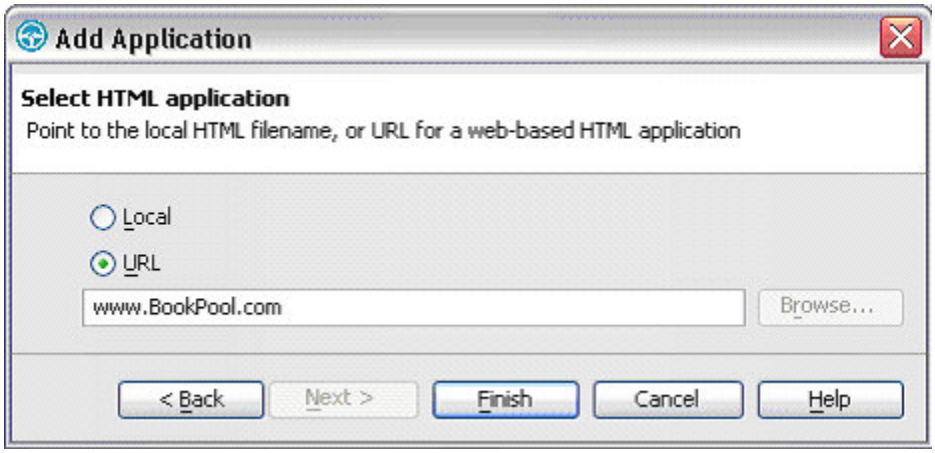
5. To add the BookPool.com URL to the list of application names, click the **Edit Applications List...** button.



6. In the Application Configuration Tool window, click the Add… button.



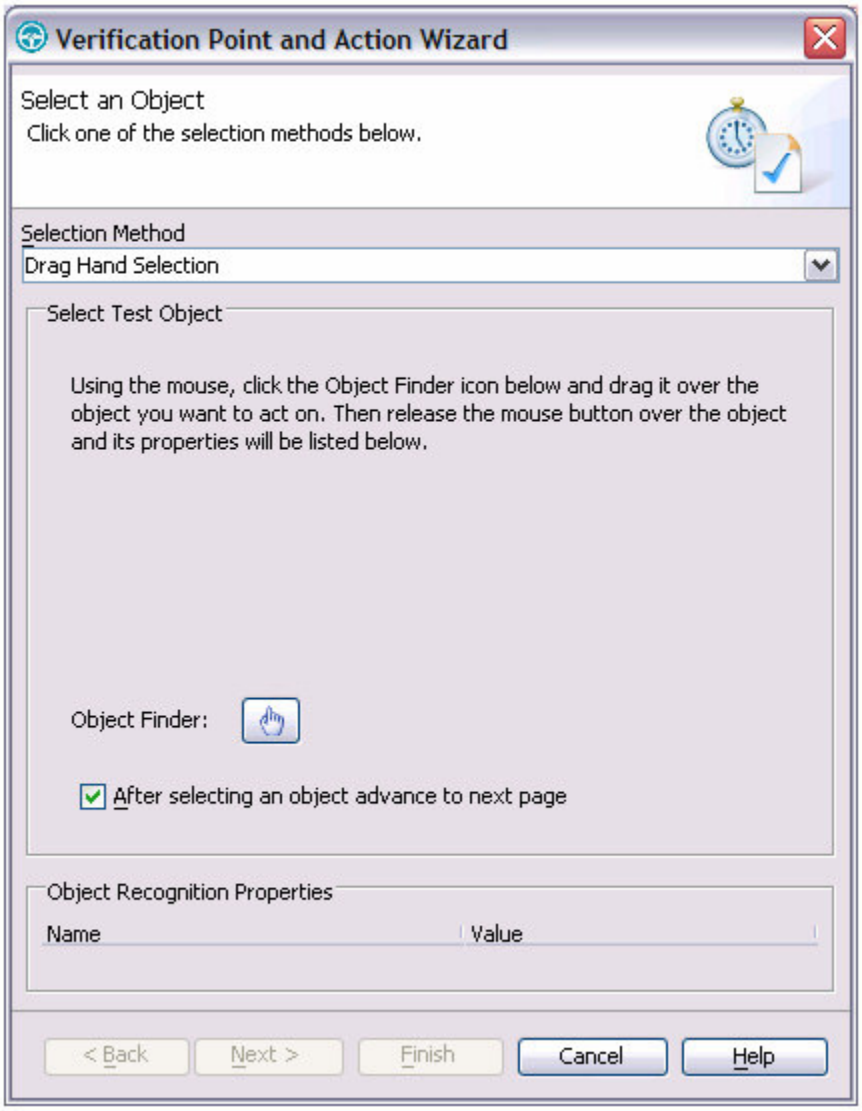
7. Select **HTML Application** and click **Next**.



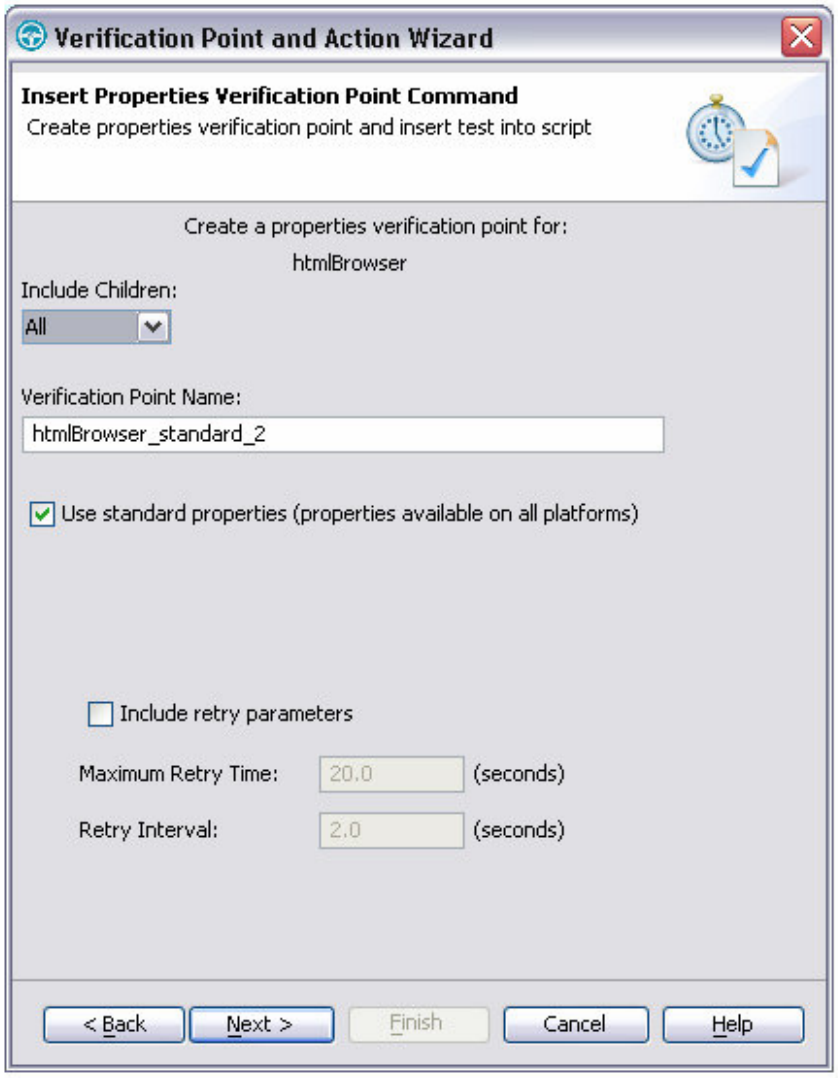
8. Enter the URL for BookPool.com and click **Finish**. You should now see the URL in the Applications list.
9. Click **Finish**, select the URL in the Application Name field of the Start Application window, and click **OK**. This should open your browser to BookPool.com.
10. Search for a book on Java.

11. At the results page, add the first book listed to the shopping cart (or basket).

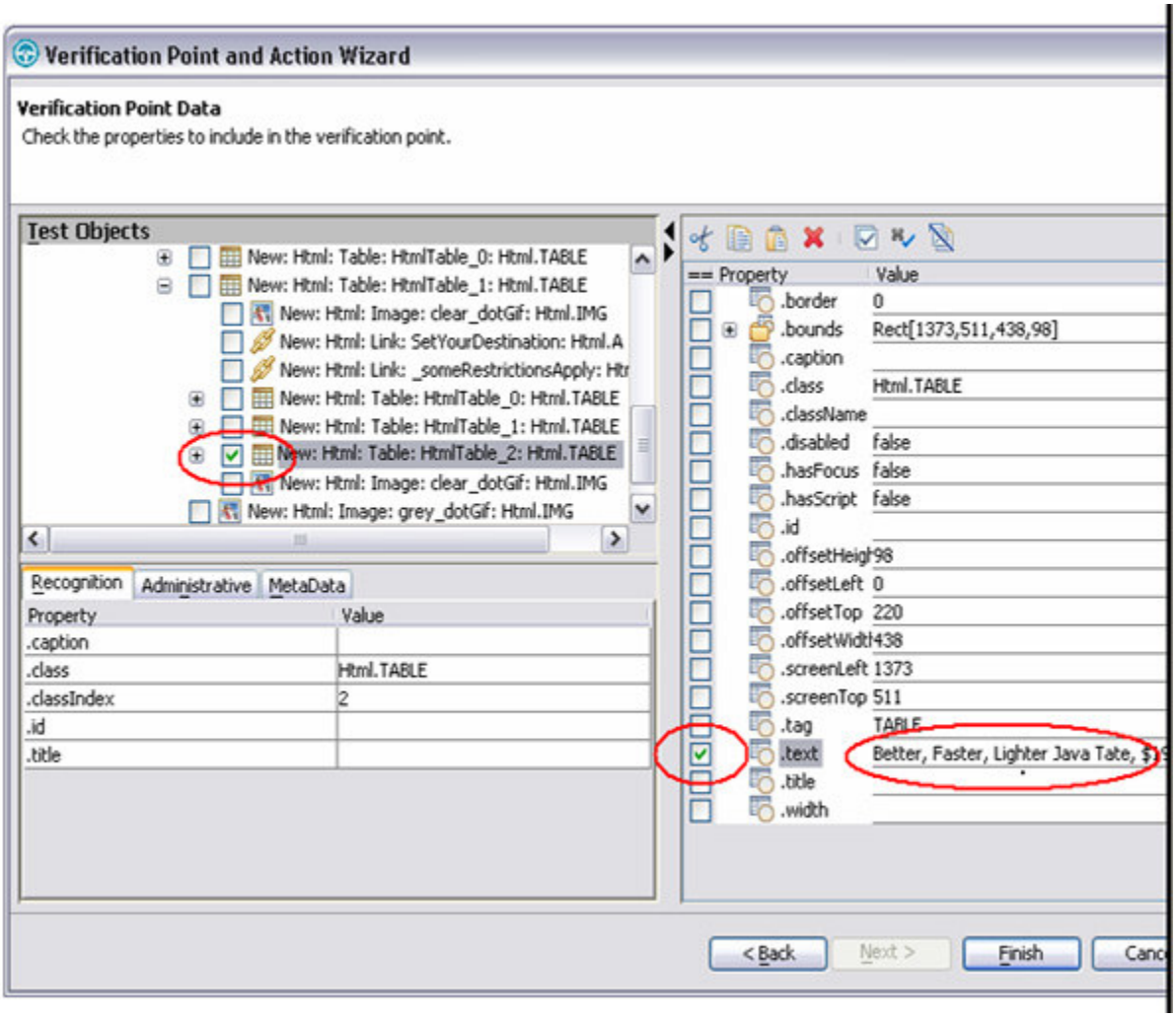
12. You should see the item in the shopping cart. To ensure that it's listed, enter a verification point by first clicking the  **Insert Verification Point** or **Action Command** button. This will open the Verification Point and Action Wizard.



13. Using the Object Finder, select the data listed in the table (you should see a red box highlight around the entire browser).
14. In the Verification Point and Action Wizard, select **Perform Data Verification Point** and click **Next**.
15. In the Insert Properties Verification Point Command window, make sure that Include Children is set to All and click **Next**.



16. On the next screen in the wizard, click **Finish**.
17. The next screen will prompt you to select the properties to include in the verification point. Navigate through the Test Objects tree to the table that contains the HTML for the book selected and check that box. Then check the box in the Property list that corresponds to the text for the book.



18. Click **Finish**, close the browser, and click  to stop the recording.

Rational Functional Tester should now generate a script that looks similar to this:

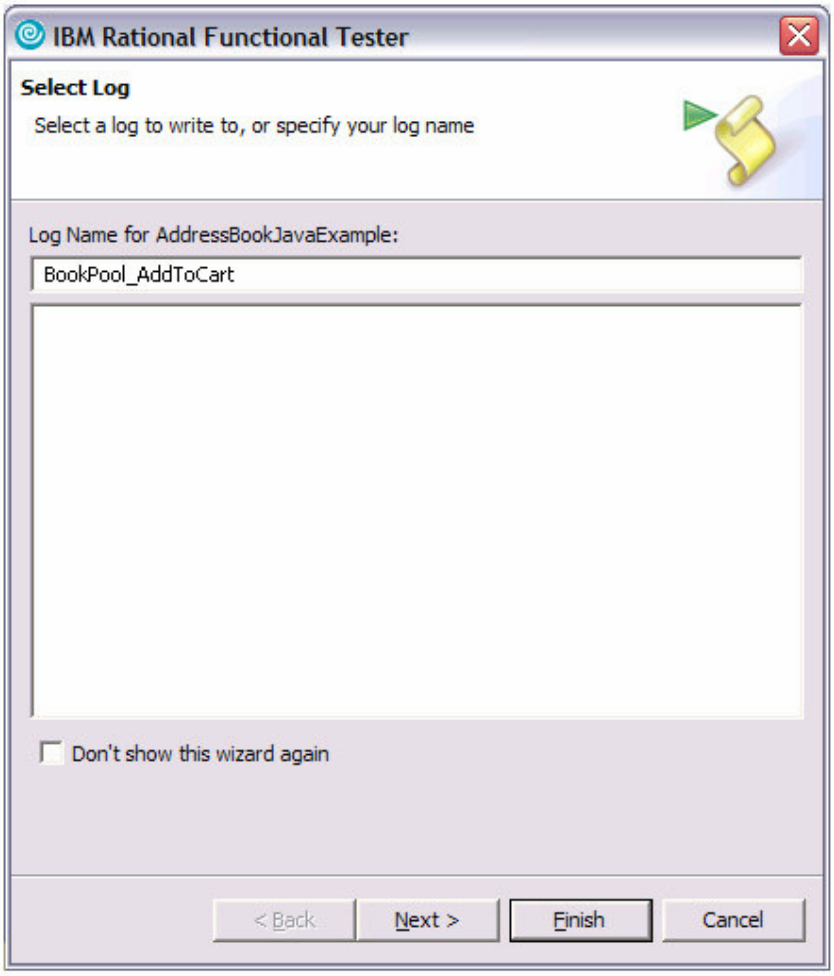
*BookPool_AddToCart.java

```
import resources.BookPool_AddToCartHelper;

/**
 * Description   : Functional Test Script
 * @author Administrator
 */
public class BookPool_AddToCart extends BookPool_AddToCartHelper
{
    /**
     * Script Name   : <b>BookPool_AddToCart</b>
     * Generated     : <b>Nov 14, 2004 6:15:57 PM</b>
     * Description   : Functional Test Script
     * Original Host : WinNT Version 5.1  Build 2600 (S)
     *
     * @since  2004/11/14
     * @author Administrator
     */
    public void testMain(Object[] args)
    {
        startApp("www.BookPool.com");

        // HTML Browser
        browser_htmlBrowser(document_bookpoolDiscountCompu(),DEFAULT_FLAGS).click(atP
        browser_htmlBrowser(document_bookpoolDiscountCompu(),DEFAULT_FLAGS).inputChar
        browser_htmlBrowser(document_bookpoolDiscountCompu(),DEFAULT_FLAGS).click(atP
        browser_htmlBrowser(document_bookpoolBooksFound(),DEFAULT_FLAGS).click(atPoir
        htmlBrowser_standardVP(document_bookpoolShoppingBaske()).performTest();
        browser_htmlBrowser(document_bookpoolShoppingBaske(),MAY_EXIT).close();
    }
}
```

Running the script

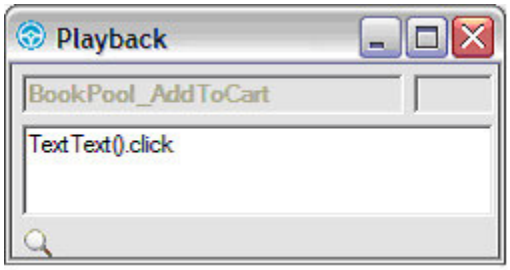
Now let's run the script we recorded.

1. With the script open, click the  Run Functional Test Script button in the toolbar. This will open the Select Log window.

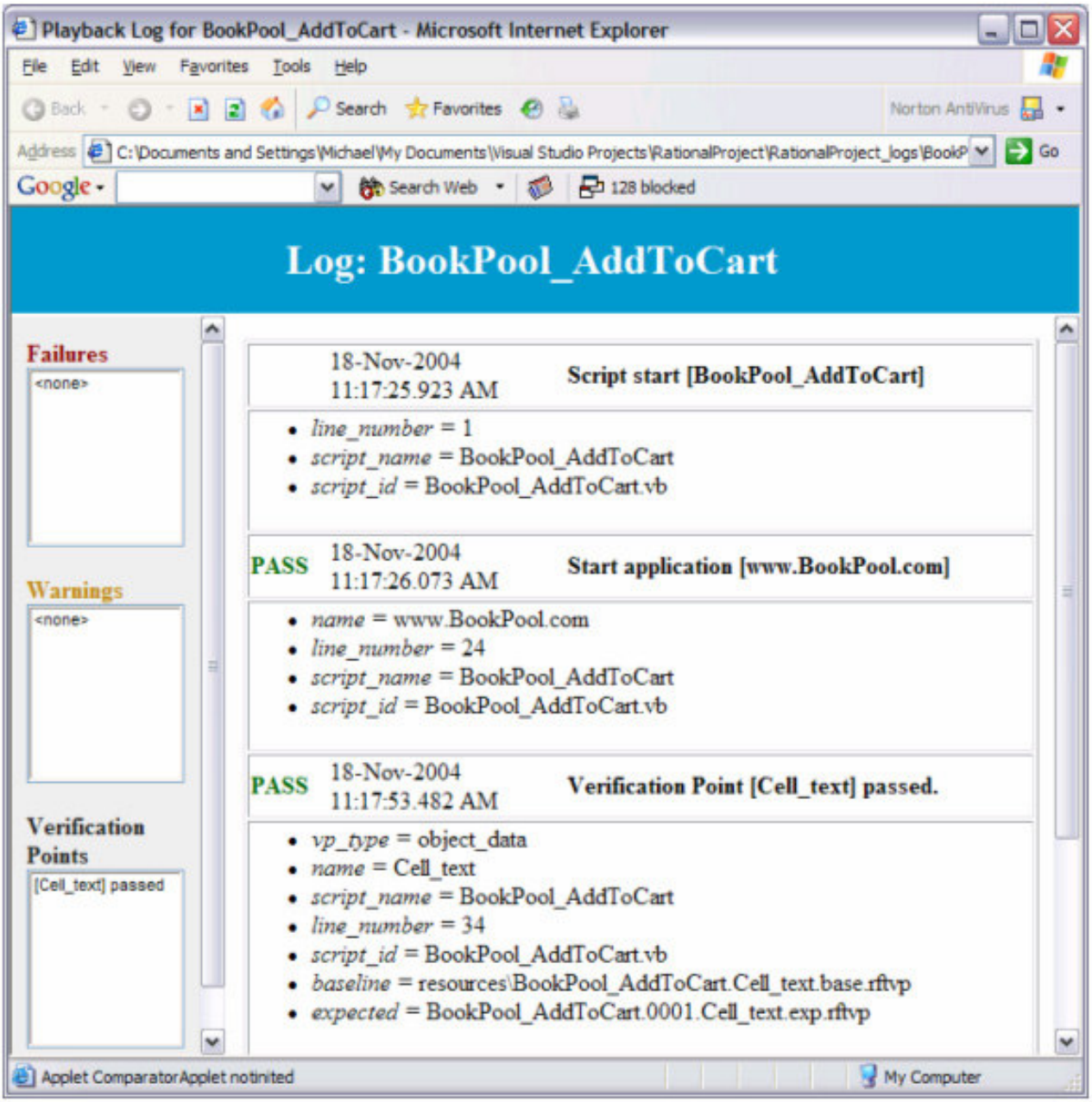


2. Click **Finish** to start script execution.

You should see the Playback window while the script is executing. This window can give you an idea of what's happening if the script appears to be "hanging" for some reason.



When your script finishes executing, a browser should open on your desktop displaying the results of the test run.



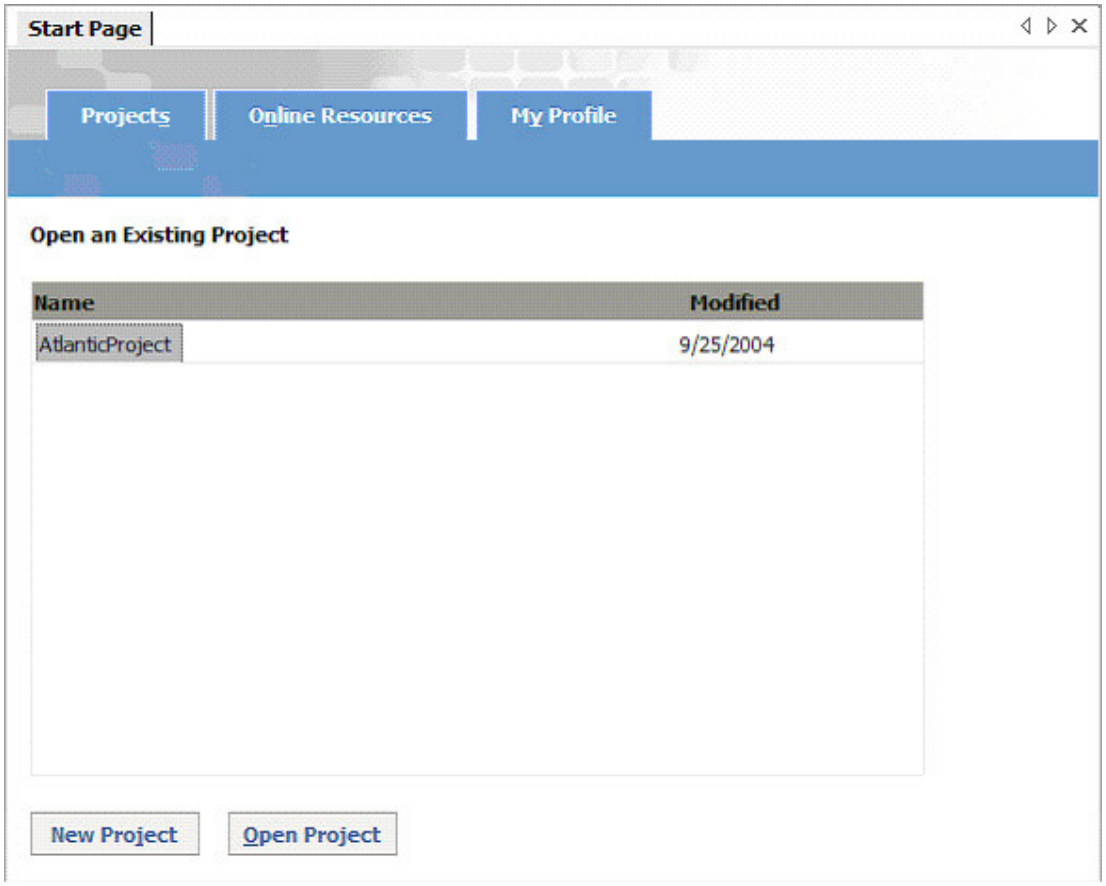
I really like this log interface. If this is new to you, start thinking about ways you can publish these results so the entire team can have access to them. Their format makes them very easy to parse (so you can combine results and display them however you want), and because they're in HTML you don't need any special software to view them. Very cool, if you ask me!

Working in the Visual Studio .NET environment

Here we'll focus on using Rational Functional Tester with Visual Basic .NET in the Visual Studio .NET Development Environment. I'll show you the user interface and take you step by step through recording and running a script.

User interface

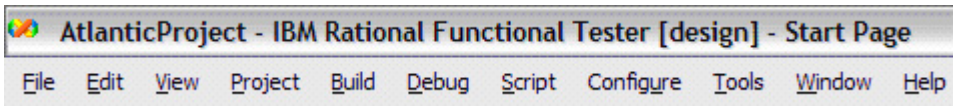
Each time you launch the Visual Studio .NET integrated development environment (IDE), you first see the Start Page.



The Start Page above is open to the Projects tab, enabling you to create a new project or open an existing one. Depending on the tab you click, the Start Page can provide the latest developer news, help you create new solutions, or promote interaction with the online developer community.

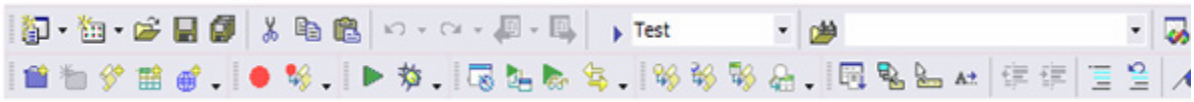
Once you open a project and select a script, the Visual Studio .NET window contains these components: the main menu, the toolbar, the Script Explorer, the Solution Explorer, the Code and Text Editor, the Server Explorer, the Class View, and the Properties window. Following is a brief description of each of these components.

Main menu



You can read about each of the main menu options in the Rational Functional Tester online help.

Toolbar



The toolbar contains these icons:

New Project -- Displays a dialog box that enables you to create a new project or solution. Click to select New Project or New Blank Solution.

Add New Item -- Displays a dialog box that lets you add an item to the currently selected project. Click to display the list of items to create.

Run Functional Test Script -- Compiles and plays back the test script currently displayed in the Code Editor.

Solution Explorer -- Displays the Solution Explorer, which provides you with an organized view of your projects and their files as well as access to the commands that pertain to them.

Properties Window -- Displays the Properties window, which you can use to view and change the design-time properties and events of selected objects that are located in editors and designers. You can also use the Properties window to edit and view file, project, and solution properties.

Object Browser -- Displays the Object Browser, which enables you to browse, filter, and search for components that you might want to include in your projects.

Class View -- Displays the Class View, which provides a programmatic view of symbols in your code -- for example, namespaces, classes, methods, and functions. Click ***icon37*** to list other IDE components to display.

New Functional Test Script Using Recorder -- Displays a dialog box that lets you enter information about the new script and start recording.

New Functional Test Empty Script -- Displays a dialog box that lets you create a script you can use to add code manually.

New Test Object Map -- Displays a dialog box that lets you add a new test object map to a project.

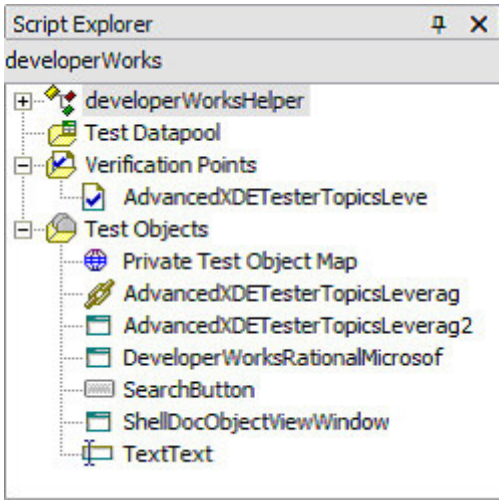
New Test Datapool -- Displays a dialog box that lets you create a datapool.

Insert Recording -- Starts recording at the cursor location in the current script, which enables you to start applications, insert verification points, and add script support functions.

Configure Applications for Testing -- Displays the Application Configuration Tool, which enables you to add and edit configuration information -- such as name, path, and other information used to start and run the application -- for the Java and HTML applications you want to test.

Enable Environments for Testing -- Displays a dialog box that you use to enable and configure Java environments, browsers, and Eclipse platforms.

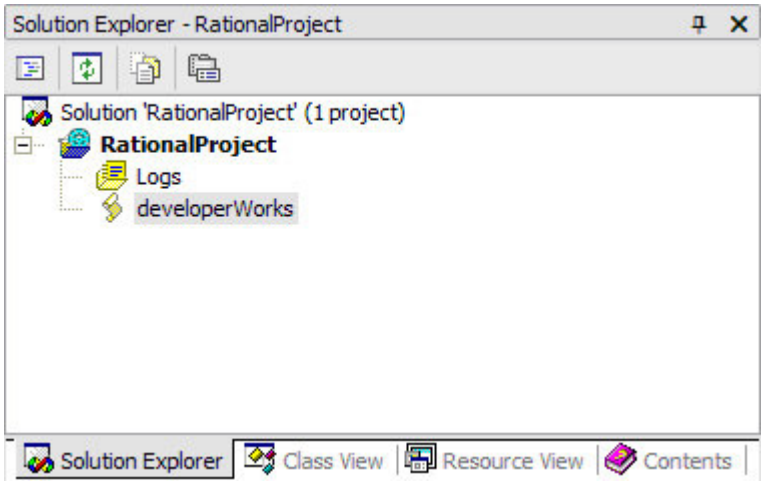
Script Explorer



The Script Explorer, in the left pane of the Visual Studio .NET window, lists the script helper, helper superclass or helper base class, test datapool, verification points, and test objects for the current script. Note these things about the Script Explorer:

- The Verification Points folder contains all the verification points recorded for the script. Double-clicking a verification point displays the Verification Point Editor.
- The Test Objects folder contains a list of all the test objects available for the script. Each test object in the list is preceded by an icon that indicates its role. Double-clicking the Test Object Map icon displays the test object map.
- Right-clicking a verification point, test object map, or test asset listed in the Script Explorer displays various menu options.

Solution Explorer



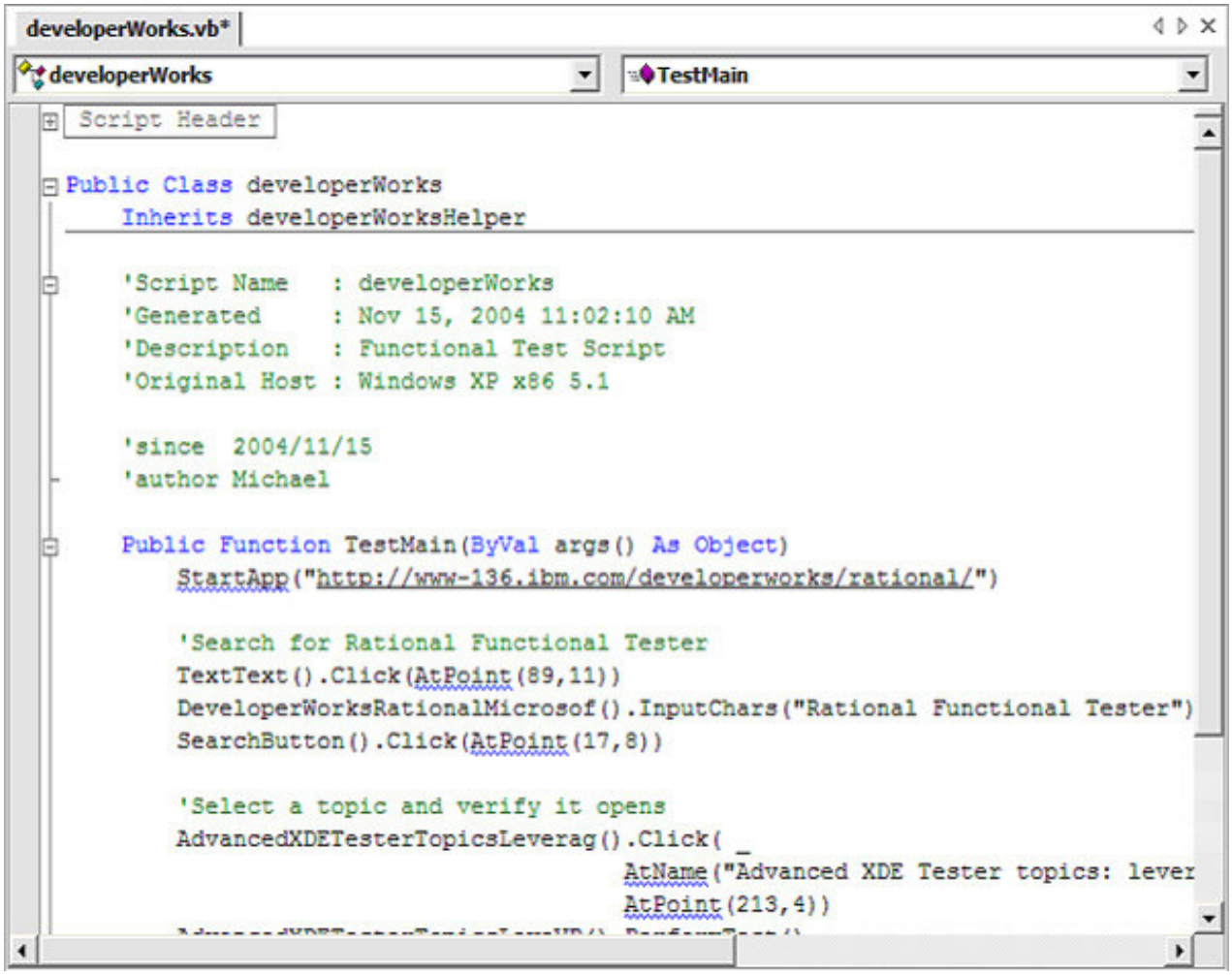
The Solution Explorer, in the right pane of the Visual Studio .NET window, provides an organized view of your projects and their files, including these:

- Projects
- Object maps
- Scripts
- Folders
- Log folders
- Logs

The Solution Explorer toolbar has the following buttons:

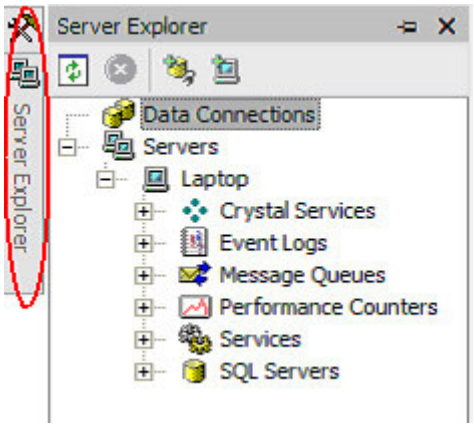
- The View Code button opens the selected script for editing in the Code Editor.
- The Refresh button updates the Solution Explorer display with information from the database.
- The Show All Files button displays all the project items, including those that have been excluded and those that are normally hidden.
- The Properties button displays the properties for the selected item in the appropriate window.

Code and Text Editor



This is a word-processing utility where you enter, display, and edit code or text. It can be referred to as either the Text Editor or the Code Editor, depending on its content.

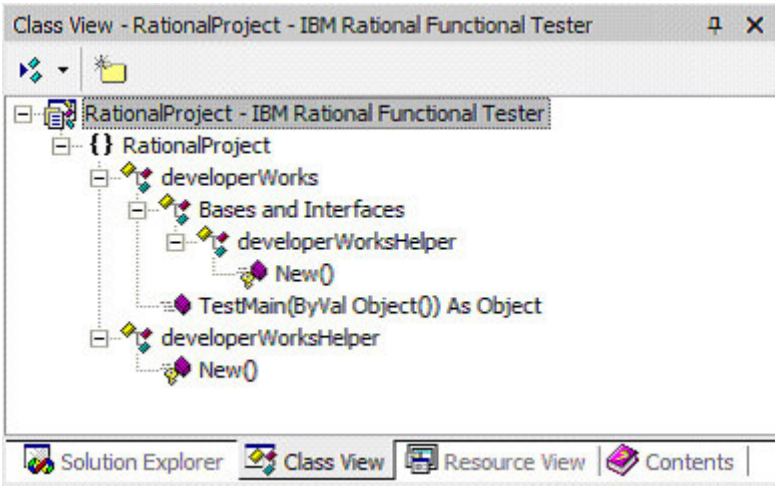
Server Explorer



By default, the Server Explorer isn't visible when you start Rational Functional Tester. Clicking the Server Explorer button on the upper right side of the screen expands the frame. You can use the

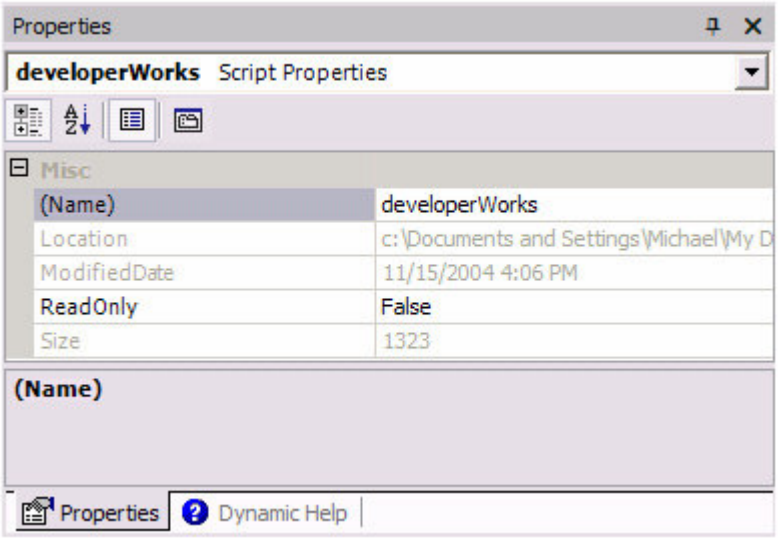
Server Explorer to view and manipulate data links, database connections, and system resources on any server to which you have network access.

Class View



The Class View is a programmatic view of symbols in your code -- for example, namespaces, classes, methods, and functions. Symbols are organized by projects. Within each project, the contents are shown in a hierarchical tree view, indicating the containment relationships between the symbols. Other structural information, such as base classes and interfaces and overrideable methods, may also appear. This logical view of your solution and its projects enables you to better understand the structure of your code and the interrelationships among its various symbols.

Properties window

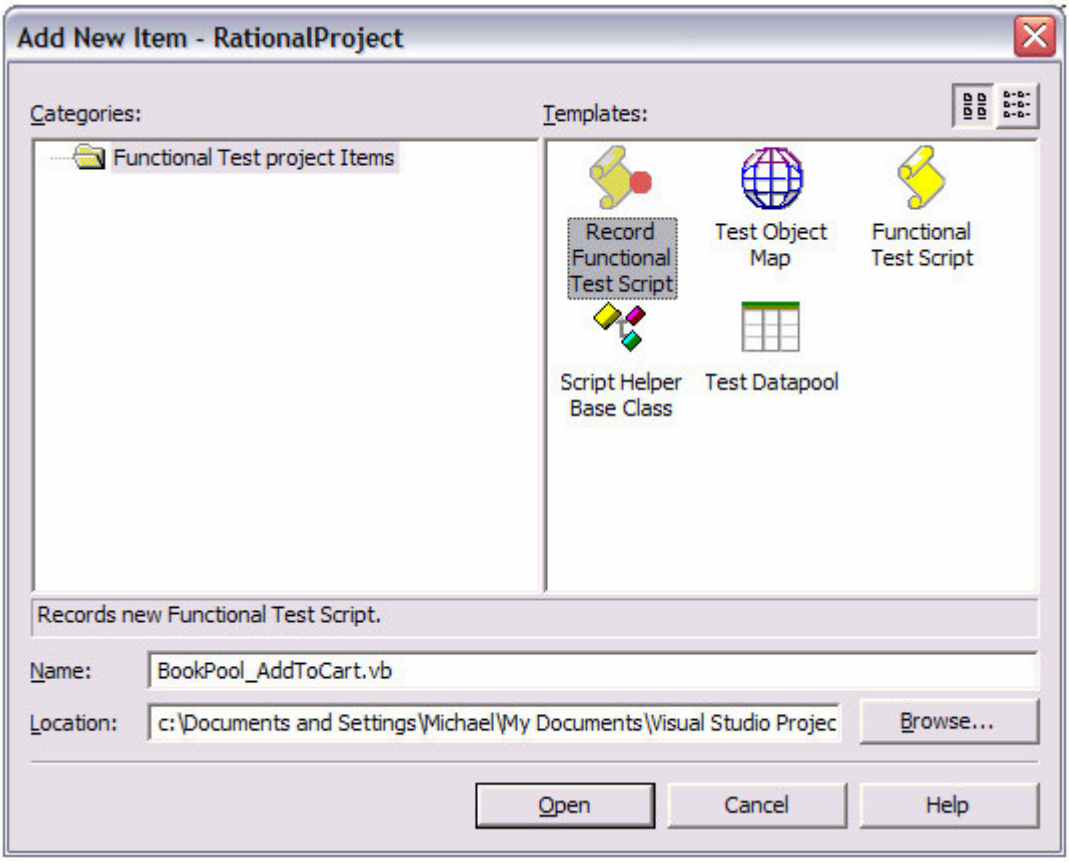


You can use the Properties window to view and change the design-time properties and events of selected objects that are located in editors and designers. You can also use the Properties window to edit file, project, and solution properties.

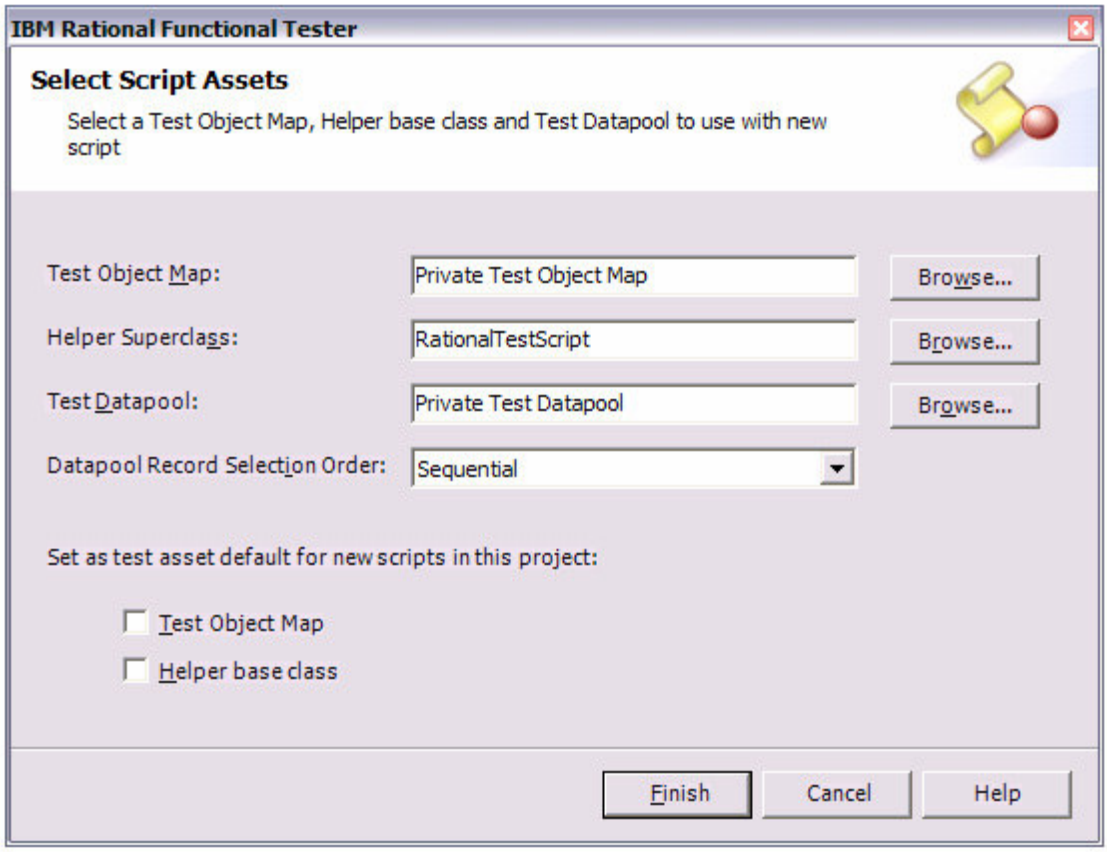
Recording a script

Now we'll record a script in Visual Basic .NET that tests adding an item to the shopping cart at BookPool.com.

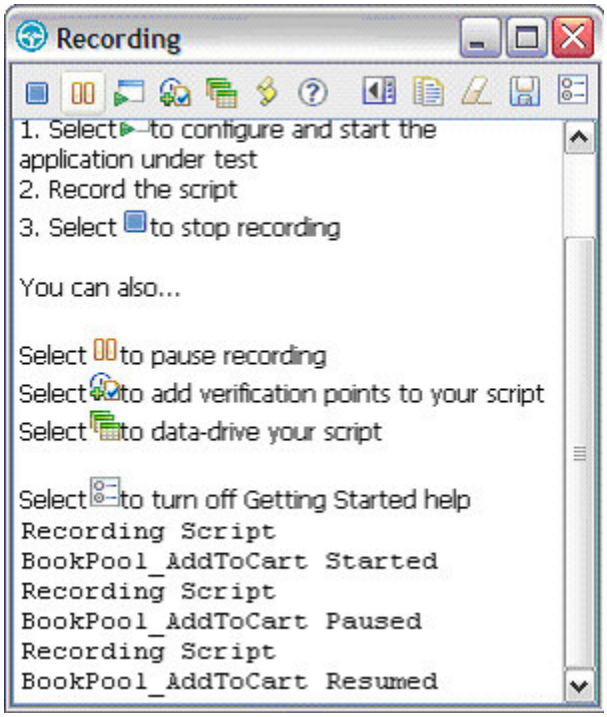
1. Open the Rational Functional Tester Visual Basic .NET scripting interface and either open an existing project or create a new project.
2. Click the Record a Functional Test Script button. This will open the Add New Item window.



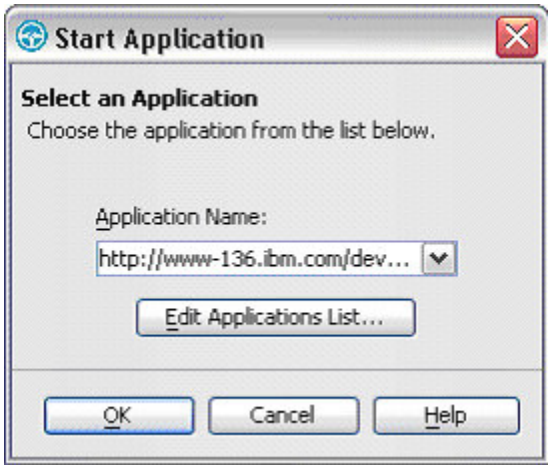
3. Select the **Record Functional Test Script** template.
4. Enter a script name and click **Open**. This will open the Select Script Assets window.



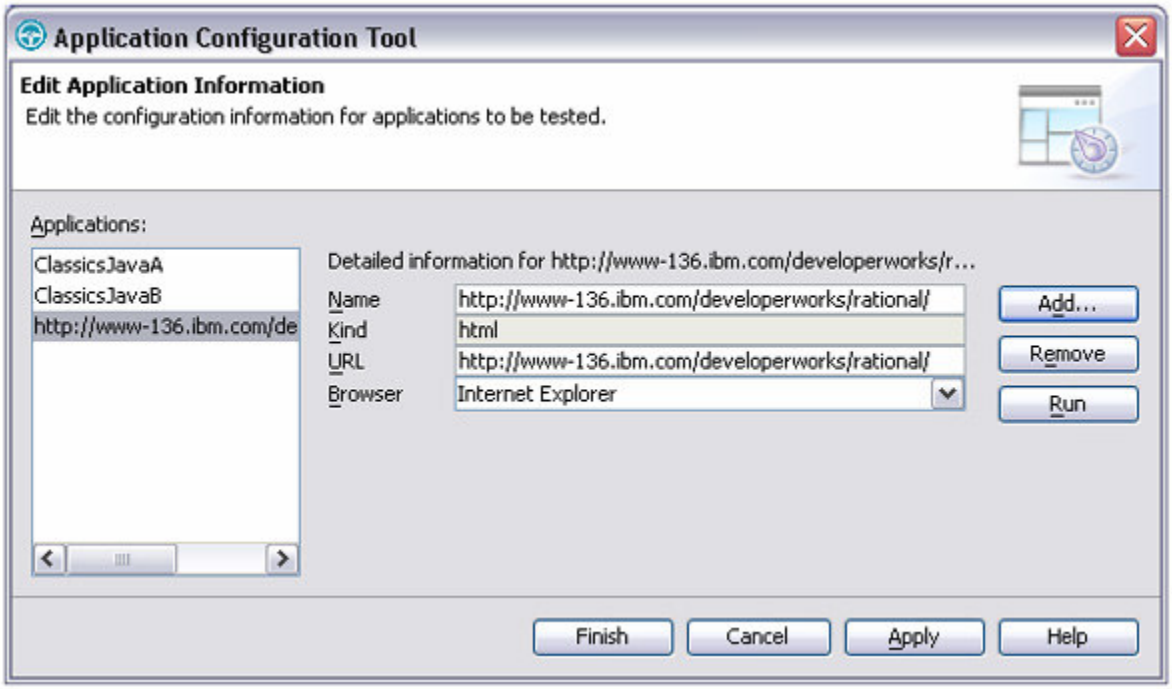
5. For this example, we'll keep all of the default information, so you can simply click **Finish** to start recording. This will open the Recording window.



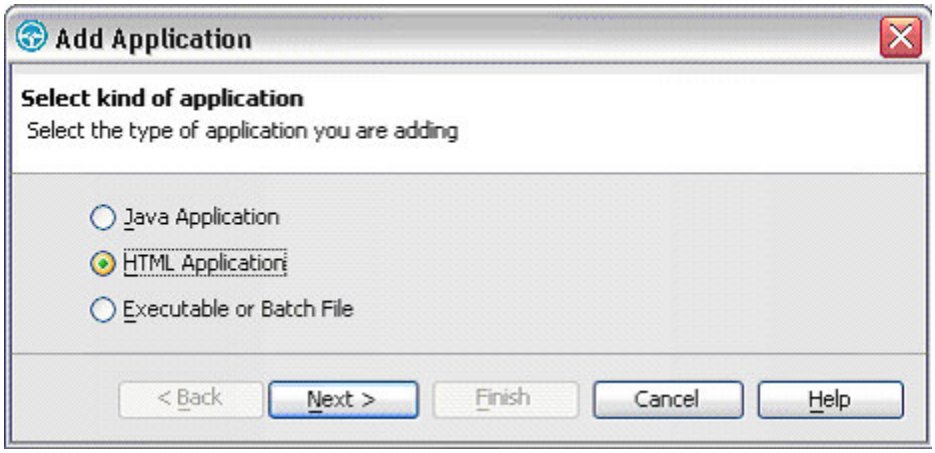
6. Click the Start Application button. This will open the Start Application window.



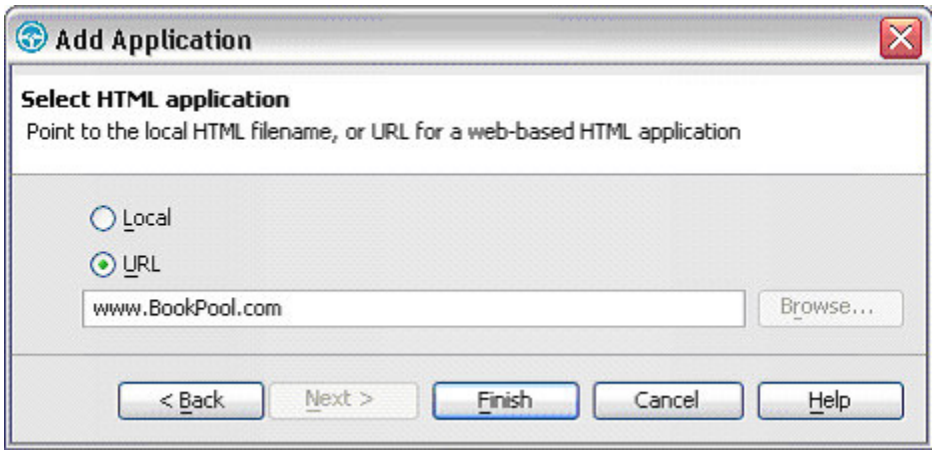
7. To add the BookPool.com URL to the list of application names, click the **Edit Applications List...** button.



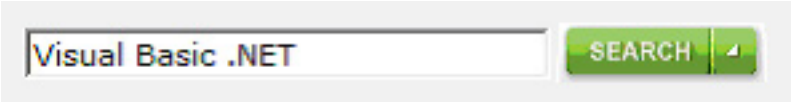
8. In the Application Configuration Tool window, click the **Add...** button.



9. Select **HTML Application** and click **Next**.

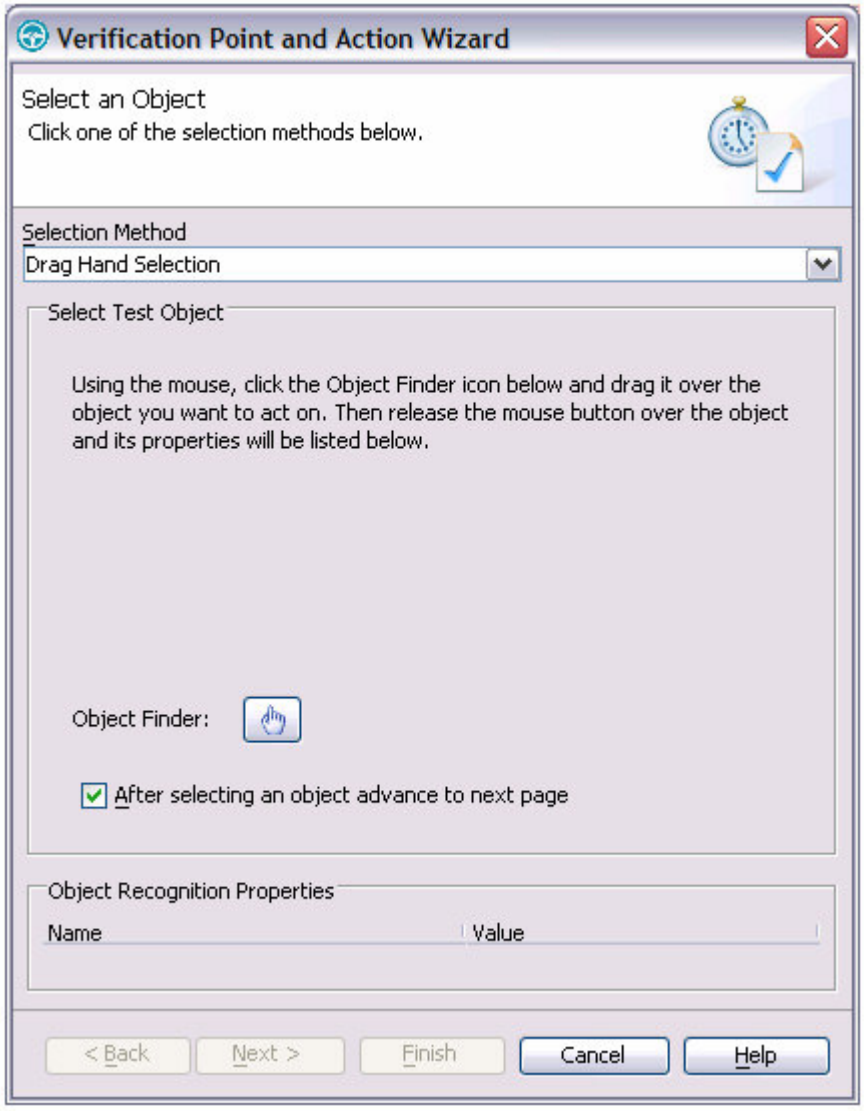


10. Enter the URL for BookPool.com and click **Finish**. You should now see the URL in the Applications list.
11. Click **Finish**, select the URL in the Application Name field of the Start Application window, and click **OK**. This should open your browser to BookPool.com.
12. Search for a book on Visual Basic .NET.

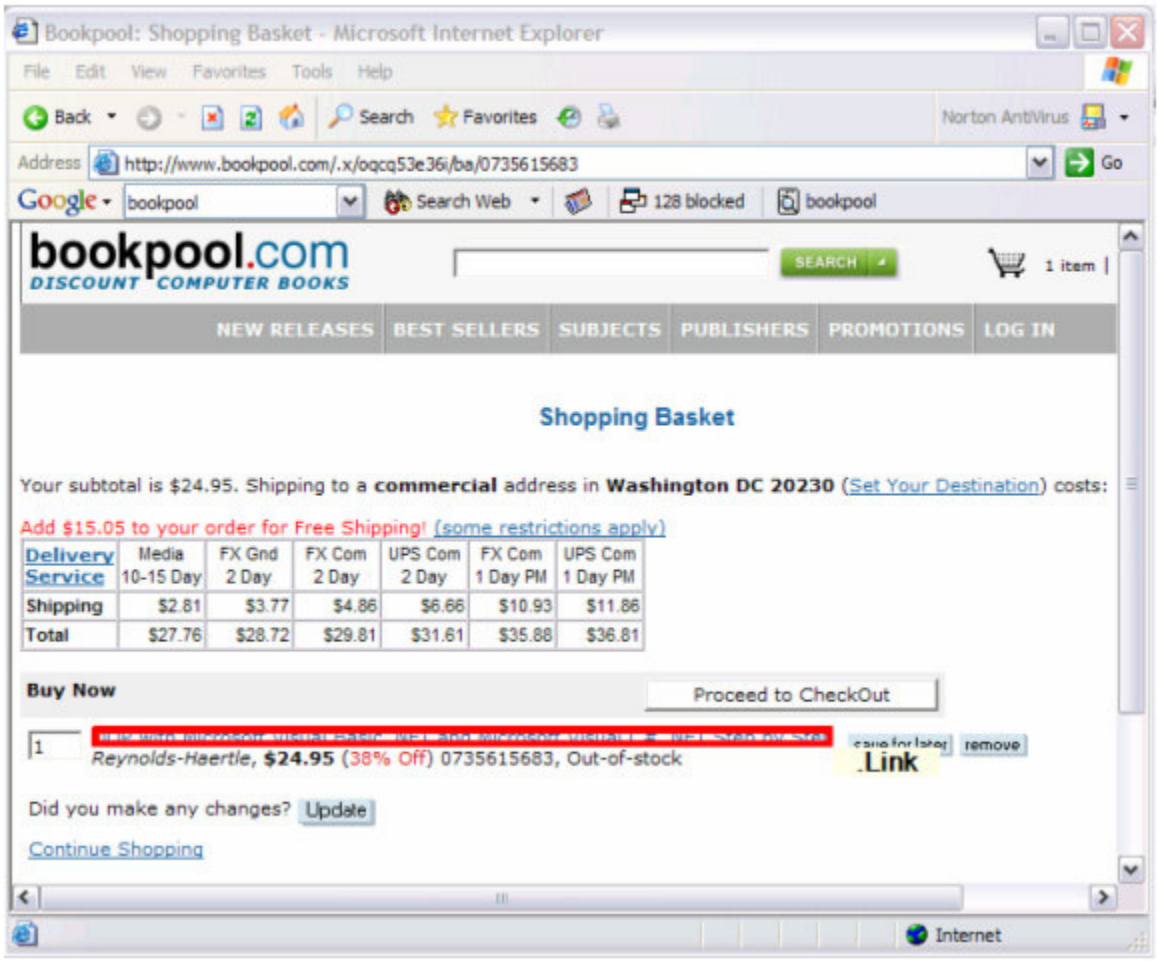


13. At the results page, add the first book listed to the shopping cart (or basket).

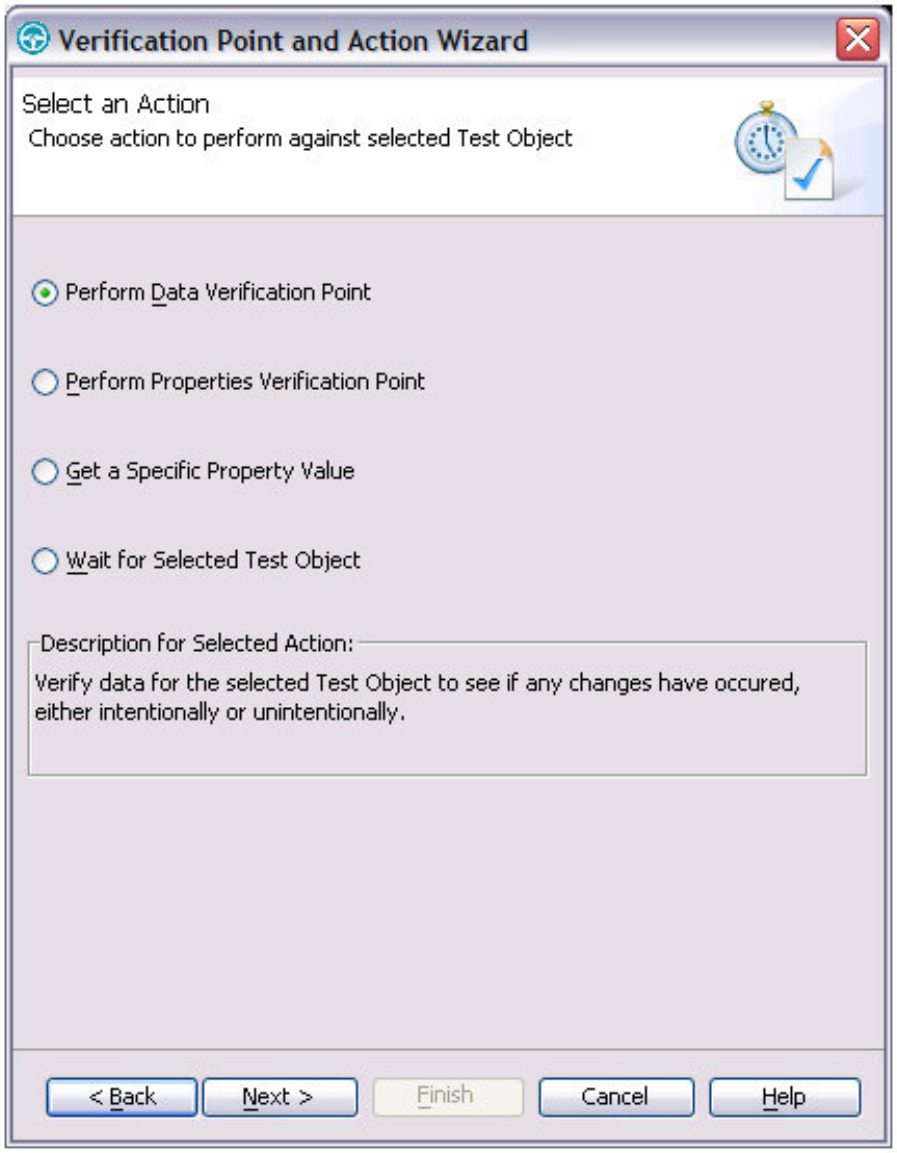
14. You should see the item in the shopping cart. To ensure that it's listed, enter a verification point by first clicking the  **Insert Verification Point** or **Action Command** button. This will open the Verification Point and Action Wizard.



15. Using the Object Finder, select the link in the table (you should see a red box highlight around the link).

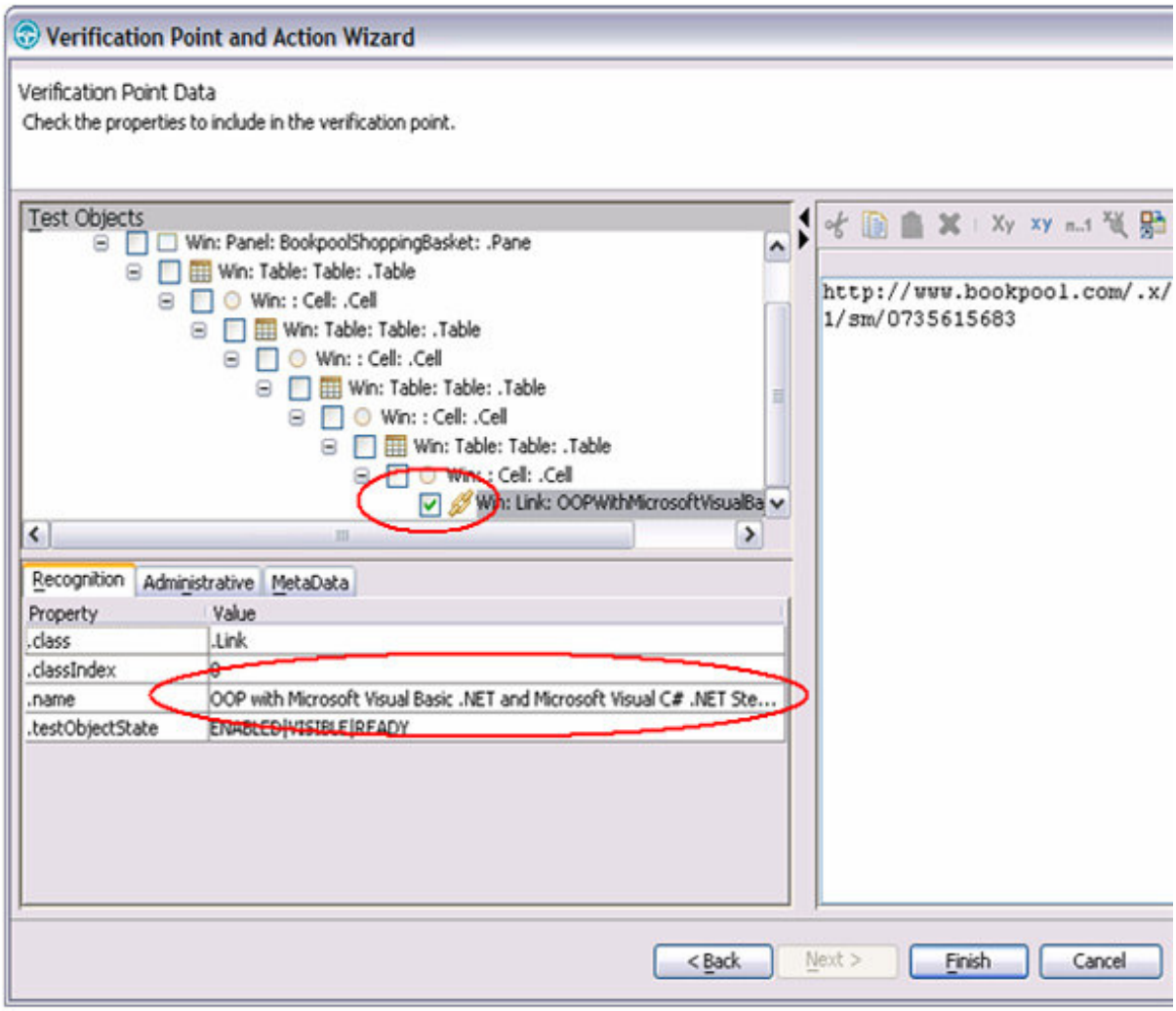


16. In the Verification Point and Action Wizard, select **Perform Data Verification Point** and click **Next**.



17. In the Insert Properties Verification Point Command window, make sure that Include Children is set to All and click **Next**.

18. The next screen will prompt you to select the properties to include in the verification point. Navigate through the Test Objects tree to the table that contains the HTML for the book selected and check that box.



20. Click **Finish**, close the browser, and click to stop the recording.

Rational Functional Tester should now generate a script that looks similar to the following. All I've done is add some formatting and comments.

BookPool_AddToCart.vb*

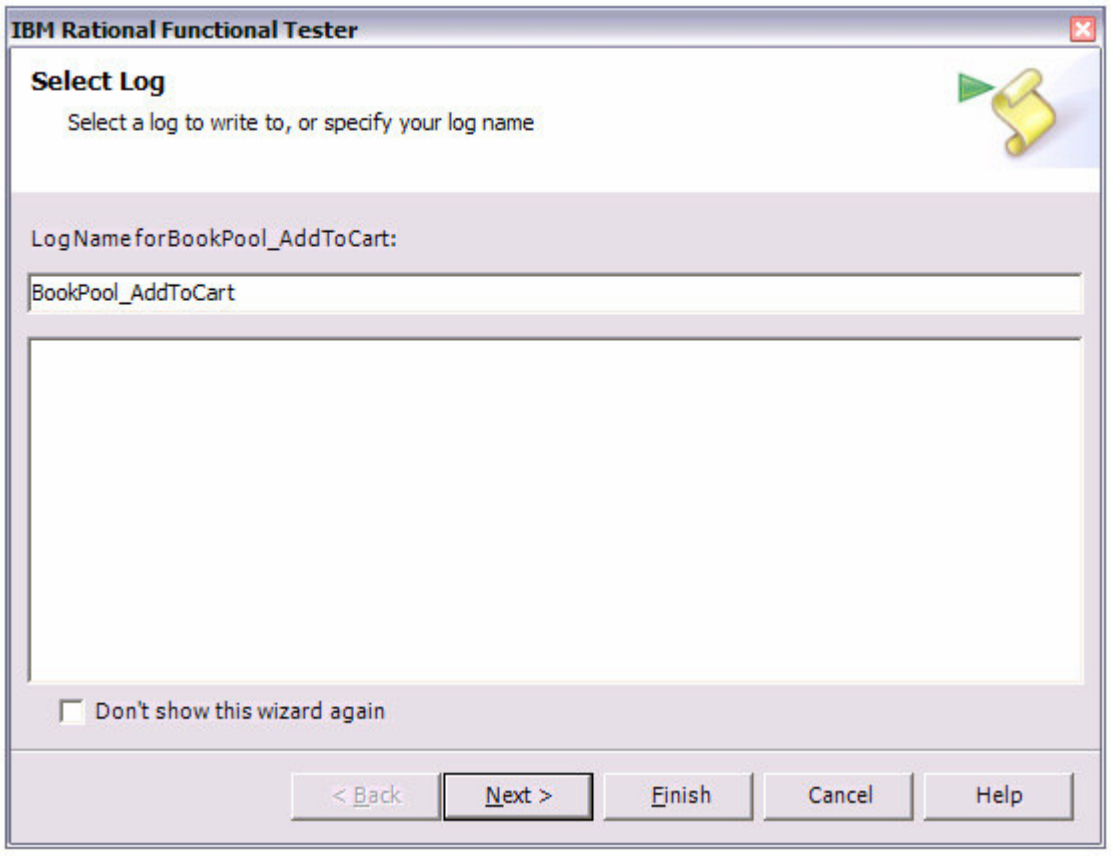
BookPool_AddToCart | TestMain

```
Public Class BookPool_AddToCart
    Inherits BookPool_AddToCartHelper

    'Script Name   : BookPool_AddToCart
    'Generated     : Nov 15, 2004 11:22:19 AM
    'Description   : Functional Test Script
    'Original Host : Windows XP x86 5.1

    'since  2004/11/15
    'author Michael

    Public Function TestMain(ByVal args() As Object)
        StartApp("www.BookPool.com")

        'Search for Visual Basic .NET
        TextText().Click(AtPoint(81,7))
        BookpoolDiscountComputerBooksW().InputChars("Visual Basic .NET")
        PushbuttonButton().Click(AtPoint(25,11))

        'Select the first book and verifiy it's added
        PushbuttonButton2().Click(AtPoint(19,8))
        OOPWithMicrosoftVisualBasicVP().PerformTest()

        'Close the browser
        BookpoolShoppingBasketMicrosof(ANY,MAY_EXIT).Click(CLOSE_BUTTON)
```

Running the script

Now let's run the script we recorded.

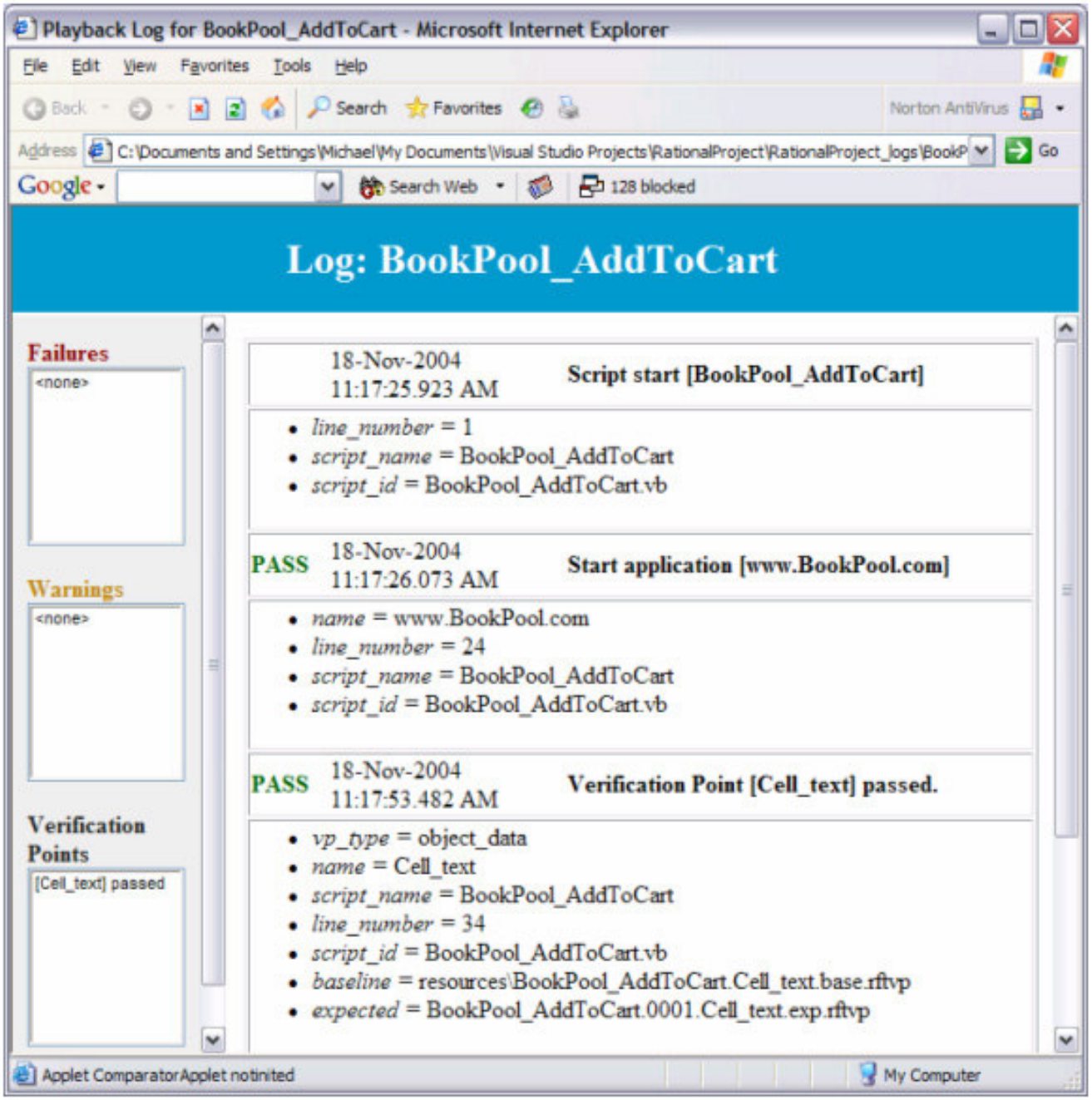
1. With the script open, click the  **Run Functional Test Script** button in the toolbar. This will open the Select Log window.



2. Enter the log name and click **Finish**.

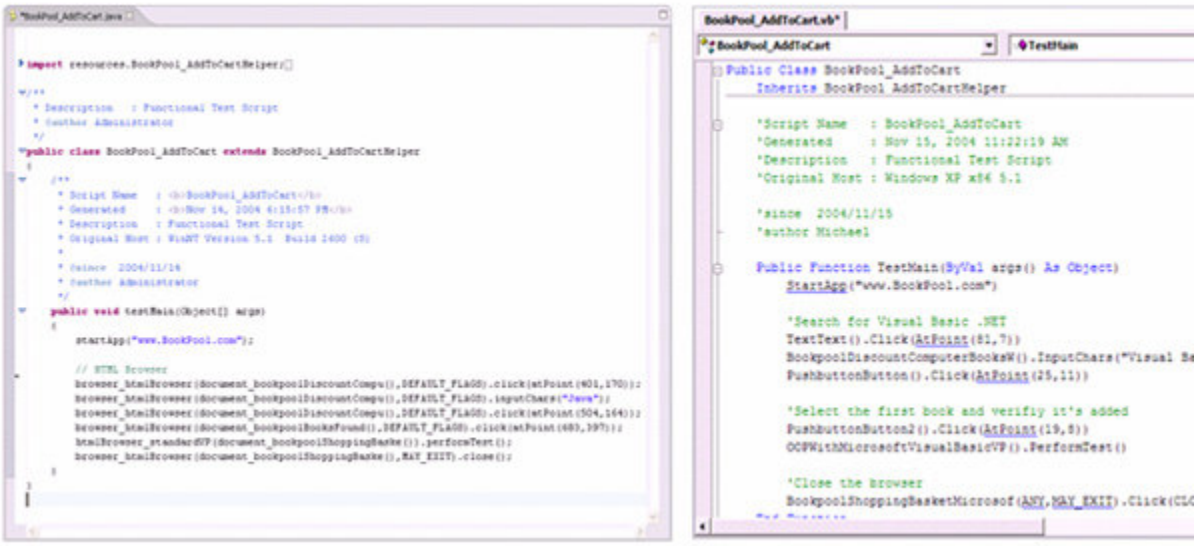
You should see the Playback window while the script is executing

When your script finishes executing, a browser should open on your desktop displaying the results of the test run.



Comparing the Java and Visual Basic .NET scripts

As I mentioned earlier, Rational Functional Tester generates (roughly) the same thing in both the Java/Eclipse and Visual Studio .NET environments. To demonstrate that, let's take a look at the scripts we just recorded, side by side. The Java script is shown on the left, Visual Basic .NET on the right:



Note these things:

- Both scripts are the same size.
- All of the objects (buttons, menus, windows, and so forth) were recognized using the exact same names.
- All of the commands are almost identical, with a few variations in each case.

Also noteworthy is that in both environments, once the Rational Recording window opened, all of the windows and wizards encountered were exactly the same.

I won't place the logs side by side like I did the script code, but if you glance at them you can tell that they display the same information in the same format. Again, what this means is that any investment you make in one version transfers to the other. So if you liked the idea of parsing the logs to results, you can parse both logs using the same function.

We haven't looked at datapools and generated object maps and all of the other things that go along with this, but basically you'll find the same thing with those features as well. If you take the time to learn about a feature of the tool in one of the environments, it will be available in the other and it will be trivial to switch (if there's any difference at all in the feature).

What next?

At this point, you may be saying, "Great! Sold! But where do I start? I don't know Java or Visual Basic .NET." If that's the case, read on. It can be difficult (and intimidating) learning this stuff. I think the best thing to keep in mind is that this is real code and real software development. Just like for any project, start small, learn a little, try something out, and build on small successes.

For example, use record and playback for a while and get a feel for the tool and the code it generates. (I'm *not* advocating record and playback as a viable test scripting option, only as a valuable learning tool!) Once you can follow the scripts (something a bit more complex than the example shown in this article), start learning ways to optimize the recorded code. There are several effective ways to do this:

- **Buddy up**. It's easier to learn stuff like this if you have someone to bounce ideas off of. If you have other people on your team learning the tool at the same time, this won't be so difficult. However, if you're on your own, try to find someone else locally who's learning as well. If you work at a large company, this could be a person on the next project; if you work at a smaller

company, you may need to find someone at your local user group. Sharing ideas like this will give you immediate support and feedback, and will help keep you motivated to try new things with the tools. The idea here is to build synergy with your efforts. Unlike talking with a developer (see below) about technical questions, discovering functionality with another tester will help you discover more effective test techniques faster.

- **Use your developers**. Often, if you simply have a language question, you can go find a developer - they're usually more than happy to help. In fact, most of the developers I've worked with really enjoy it. Not only do most people genuinely want to be helpful and share something they've learned, but many developers feel that testers could better test their software if they better understood the technology. While I don't necessarily agree with that philosophy, it does make developers very willing to help, and this makes your job of learning a new language that much easier.
- **Take some classes**. I'm sure IBM Rational Software will be offering classes on these tools in the future, both online and classroom style. And there are likely introductory classes on Java and Visual Basic .NET taking place at local colleges in your area (not to mention the hundreds of online classes you can now take from some good to not-so-good online colleges). The idea here is, if you learn effectively in the classroom setting, it can be relatively cheap and convenient to get training in these two languages. For most automated testing, you wouldn't need more than an introductory course.
- **Get a good book**. My learning tool of choice is books, and there are plenty of books on Java and Visual Basic .NET out there. If you're unsure what to look for, have your developers recommend a good introductory book, or as you're looking at classes make a list of the books they require and consider those. Regardless of what you end up choosing, at the bare minimum get yourself a desk reference you can keep nearby.
- **Go online**. Finally, and this goes without saying, go online. There are tons of Java, .NET, and testing Web sites that will have relevant information. Keep in mind that most articles written on Rational XDE will still be applicable, so there's a wealth of information there. Also, hit the online forums with questions you may have.

If you've been using some other tool in the past and are looking at expanding your suite of tools to include Rational Functional Tester, I'd suggest picking a subset of your automation effort and implementing it in Rational Functional Tester. See what the tool can do and what it can't. Good luck!

Resources

- Learn about the IBM Software Development Platform and why Rational products are an important component of it.
- Connect with other IBM Rational software users and learn more about Rational User Groups in the Rational Software Global User Group Community.
- Find online and classroom courses on Java and Visual Basic .NET offered by IBM Rational Software Training.
- Start learning about Java in Head First Java by Kathy Sierra and Bert Bates (O'Reilly, 2003) and in Core Java 2, Volume 1: Fundamentals, 7th edition, by Cay S. Horstmann and Gary Cornell (Prentice-Hall, 2004). Get a head start on Visual Basic .NET by reading Learning Visual Basic .NET by Jesse Liberty (O'Reilly, 2002) and referring to VB.NET Language Pocket Reference by Steven Roman, Ron Petrusha, and Paul Lomax (O'Reilly, 2002). Purchase other books at discounted prices in the Rational section of the Developer Bookstore.
- Visit these Web sites for information on Java, .NET, and testing: IBM developerWorks®, Java Ranch ("a friendly place for Java greenhorns"), Sun's Java Technology site, Microsoft Visual Basic Developer Center, and Visual Basic Tutorial (a free online tutorial for beginning to intermediate users).
- Share your questions and views on the topics covered in this article with the author and other readers in the Rational discussion forums. See also the JavaRanch Big Moose Saloon, Xtreme .NET Talk, VBForums.com, and QAForums.com.

About the author

Mike Kelly is currently a senior SQA specialist at CTI Group. He's had experience managing software automated-testing teams and has been working with the Rational tools since 1999. His primary areas of interest are software development life cycles, software test automation, and project management. Mike can be reached by e-mail.