Rational. software

IBM

A First Look:
IBM Rational RobotJ

For release on March 25th, 2002 at the 2002 JavaOne Java Developers Conference.

:

Author Profile

Tom Arnold has been programming, managing and consulting on software test automation projects since 1991. In 1993, Tom co-founded Software Testing Laboratories (later renamed to ST Labs, and eventually purchased by Data Dimensions/LionBridge/Veritest), one of the software industry s first outsourced software testing firms. While at ST Labs, Tom added training & consulting to his repertoire when he began writing and teaching software test automation classes to Microsoft employees as he created ST Labs training group. Two years later, Arnold published the book, *Software Testing with Visual Test 4.0*.[1] Shortly after his book s publication, Tom managed the development team for Visual Test 4.0b for Microsoft Corporation; this version later became Rational Visual Test 4.0r. Arnold continued to run software test automation projects, consult for companies looking to establish an approach to testing, and speak at industry conferences about effective uses and practical approaches to automated testing. Tom continued managing software dev teams, including the programmers and test engineers that created Rational Visual Test 6.0 (released in November 1998). His 700-page book *Visual Test 6 Bible*[2] and 10-tape (10-hour) training video series *VT6 InDepth*[3] were both published in January 1999. Tom s current focus is on software project management, programming, test automation and writing papers & articles about software development. Tom plays an active role in the software industry and presents at such conferences as STAR (Software Testing Analysis & Review), Internet World, and RUC (Rational Users Conference). His Bachelors Degree in Computer Science comes from Purdue University.

Table of Contents

Please send comments to tom@xtenddevelopment.com

Introduction

Rational asked me to take a look at their new baby, Rational RobotJ. There were two reasons they asked me to do this. The obvious reason is that I've been in the test automation game for a long time and have seen a lot of ways people have used (and misused) these testing tools. The second reason is that I've never used Rational Robot or their Test Manager model; I would bring an automation background with a fresh set of eyes looking at how they're approaching software test automation solutions.

I received my Beta copy of Rational RobotJ and I was eager to get started exploring what Rational's latest tool brings to software testing. Having written Rational Visual Test automation scripts for years I was eager to see how this tool would define itself in the realm of testing web-based applications.

Within minutes of installing the tool, clicking around to see what was in the interface, and generally trying to figure out what s what, I decided to dive in and use the *Record new RobotJ script* button on the toolbar. What the heck, I thought to myself, let s see what kind of verbose script this thing generates as it tries to not only accurately record my actions, but keep things in sync so that the playback actually works. I m a bit cynical when it comes to record-and-playback tools.

I was eager to learn more about what this tool could do even though I m fonder of the good old-fashioned banging-out code approach. However, using a recorder is one of the best ways to learn about an automation tool and how that tool wants to be used. It turned out, as you ll soon see, that the recorder was (and is) not only a great tool for learning more about RobotJ, it s the way Rational expects people to create the majority of their test scripts. It does a great job of it, too.

Background

I got wind through the test automation grapevine that Rational was about to release a new product that would focus on testing HTML and Java-based web applications. Furthermore, it used Java as the programming language allowing for the typical benefits of object oriented programming. I wanted to see it as soon as possible. A new programming-related toy to play with and I couldn t wait to get my hands on it.

For years I ve been using and writing about software test automation with Rational Visual Test as my tool of choice. Its Visual Basic-like language made programming straightforward, and its advanced capabilities of working with pointers for complex data structures made it a real language. Visual Test even had functionality for testing web pages that was added in version 6.0. It worked, and still works, but it s a capability added to a testing tool that was originally designed for testing Microsoft® Windows programs. Visual Test was trying to be the end-all, be-all tool for software test automation, and, to be sure, it has held its own over the years.

I still use VT today, even for non-testing solutions. But when a new tool focusing solely on web application testing was about to be announced, I really wanted to see what it could do, and I was really excited about Java being used as its scripting language.

I ve always tried to teach good programming practices to my students and clients, as well as approaches to create object-oriented-like scripts, complete with data hiding, encapsulation and a simple form of inheritance, in a not-so object-oriented language. The methods worked well, but using an object oriented language enforces these practices instead of simply encouraging them. That s why I m happy that RobotJ is making use of Java as its language.

The Test

I figured I would start off with something simple: Click around on some links to see how well it handled the scrolling of the browser window to reach links that were off-screen and lag-time between page loads. But then I thought about it another way: when test-driving a new car that promises to have a big engine and racing suspension, would I sit there and verify the turn signals work and that I can find NPR on the radio? Well, yes, actually. But in addition, I d definitely want to take it out and find lots of curves to play in, even while listening to NPR s *Talk of the Nation*.

Admittedly, I didn t take it through too many curves on the first run because I didn t want to put it into the wall. I wanted to have the script I recorded actually work; I was slightly skeptical about a recorded script and whether or not it would work straight out of the gate. So I clicked the record button and went through the following steps:

1. I ran my browser-of-choice (Microsoft Internet Explorer 6.0)
2. Titled my test First_one (You ll see it referenced in the code)
3. Ran my application with associated URL (www.xtenddev.com)
4. Clicked a link to reach Xtend Development s search engine
5. Typed in the text I was looking for (Rational)
6. Navigated to the product I wanted (VT6 InDepth Videos)
7. Added an item to my shopping cart (buy.gif graphic)
8. Clicked the Check Out button to complete my transaction
9. Filled in my customer and billing information
10. Submitted the product order to our transaction server
11. Exited my browser

Expected Results

As a test automation programmer I had a number of preconceived notions of what to expect when using the RobotJ recorder. These notions are based on working with Visual Test and other test automation tools. I expected a number of issues, specifically:

> Compatibility: At first I was concerned about its compatibility with Microsoft Internet Explorer 6.0. Microsoft keeps pumping out updates of its browser at a fast pace and I was wondering if RobotJ was keeping up. Rational must be using the internal object model provided by the IE APIs, but so did the Visual Test team, and Microsoft s internal bits don t always work as expected and required a number of work-arounds by the dev team. I imagined the same was true for the RobotJ developers.
>
> Script Length: I was concerned that a very long script would be generated with a number of delay or sleep statements attempting to emulate my actual delays in typing and clicking. I was also expecting a number of commands verifying the size and position of my browser window, screen

resolution, and other things that can sometimes be overkill and create a long script.

Latency Issues: I expected the script to run only under perfect conditions where the server response was equal to or faster than when the recording was generated. In the event the server lagged, I fully expected the need to fine-tune the script with additional code to pause at specific points, such as when a link was clicked that navigated off the current page.

Actual Results

What I received, however, was a pleasant surprise:

Compatibility: Clicking on links, typing into multiple `<input type=text>` box controls, `<textarea>` controls, `<select><option>` drop-down boxes, `<input type=submit>` buttons, submitting forms with *post* and *get* methods, and more, all seemed to work without a problem. I ve not yet tried it on other browsers, but I was pleased that it correctly recognized and more importantly successfully found on playback, the many varied controls on the web page under test.

Compact Code: The code was compact and readable. By taking advantage of a Helper base class, all of the extraneous code was hidden from view leaving only the meat of the source that was of interest to me. As expected, comments were added to help guide the automation programmer through what was happening at each line in the code. Additional information that I didn t expect, however, was included as comments, specifically the information that was posted when the *Place Order Now* button was pressed. (Nice touch.)

Synchronicity: Because no sleeps or delays were explicitly added in the code, and the Helper base class handled all control recognition, action was only taken upon a control when it could be located. If it were there the script would fly through filling in the form and clicking the appropriate controls. It waited patiently, however, if the page was still loading or if it was still building an understanding of what controls and links existed on the page.

For you code junkies I ve tried to create a sample script within the first few pages of this document. The generated code is shown on the next page in **Listing 1**. (Note: Some minimal formatting was necessary to have it fit readably into this document, including removal of some of the auto-generated comments). I will spend the next few pages showing you what it looked like using RobotJ to generate this script, discuss some of the key aspects of the generated script, and then explore the results from running the script.

Listing 1

```
import resources.First_oneHelper;
import com.rational.test.ft.*;
import com.rational.test.ft.object.interfaces.*;
import com.rational.test.ft.script.*;
import com.rational.test.ft.value.*;
import com.rational.test.ft.vp.*;

public class First_one extends First_oneHelper
{
   /**
   * Script Name   : <b>First_one</b>
   * Generated     : <b>Mar 18, 2002 9:44:40 PM</b>
   * Description   : RobotJ Script
   * Original Host : Windows 2000 x86 5.0
   * Original Host : WinNT Version 5.0  Build 2195 (Service Pack 2)
   */

   public void testMain (Object[] args)
   {
   startApp("Xtend Development, Inc.");
   consultingsmjpg_textVP().performTest();

   // Document: Xtend Development, Inc.: http://www.xtenddev.com/
   Link_SiteMap().click();
   Link_SearchEngine().click();
   Text_query().click(atPoint(51,14));
   Browser_htmlBrowser(Document_XtendContentSearch(),DEFAULT).inputKeys("rational");
   Button_Submit().click();
   Link_XtendOnDemandVideoLibrary().click();
   Document_XtendOnDemandVideoLib().drag(atPoint(754,247),atPoint(753,450));
   Link_VisualTest6Indepth().click();
   Document_VT6InDepthVideos().drag(atPoint(755,137),atPoint(754,203));
   Image_buygif().click();
   Button_CheckOutsubmit().click();
   Text_name().click(atPoint(29,13));
   name_textVP().performTest();
   Text_name().click(atPoint(35,10));
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "Thomas Arnold");
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "{Tab}Xtend Development, Inc.{Tab}206-938-2370{Tab}206-932-8797");
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "{Tab}206-938-1191{Tab}tom@xtenddev.com{Tab}tom@xtenddev.com{Tab}");
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "4742 42nd Avenue SW{Tab}#621{Tab}Seayt{BKSP}{BKSP}ttle{Tab}WA");
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "{Tab}98116{Tab}{Tab}{Tab}{Tab}{Tab}{Tab}");
   Document_OrderConfirmationForm().drag(atPoint(754,178),atPoint(731,301));
   Text_CCNum().click(atPoint(14,9));
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "{Num4}{Num3}{Num8}{Num8}{Num5}{Num4}{Num3}{Num0}{Num2}{Num3}");
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "{Num7}{Num2}{Num0}{Num3}{Num5}{Num7}");
   List_Expiremonth().click();
   List_Expiremonth().click(atText("05"));
   Document_OrderConfirmationForm().drag(atPoint(752,364),atPoint(752,444));
   List_shippingid().click();
   List_shippingid().click(atText("United States (lower 48 states) (Overnight) $36.00"));
   Text_Comments().click(atPoint(154,58));
   Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys(
        "This is a test order. Delete it!{Enter}");
   Document_OrderConfirmationForm().drag(atPoint(752,356),atPoint(753,478));
   Button__PLACEORDERNOWsubmit().click();
   CloseWindow_textVP().performTest();

   // Document: Xtend / Purchase Completed:
   //http://www.xtenddev.com/purchase_completed.asp?name=Thomas+Arnold
   //&company=Xtend+Development%2C+Inc%2E&email1=tom%40xtenddev%2Ecom
   //&address1=4742+42nd+Avenue+SW&address2=%23621&city=Seattle&state=WA […some deleted…]
   Link_CloseWindow2().click();
   Dialog_HtmlDialogButtonYes().click();
   }
}
```

How to Generate a Script

Now that you ve seen what a script looks like that goes through all of the steps I listed earlier (see Listing 1), I m going spend the rest of this document explaining how I generated the script as well as identify points-of-interest along the way.

Welcome to Rational RobotJ

As I ve already mentioned, RobotJ is Rational's newest solution to testing Java or Web application environments on Microsoft Windows and Unix platforms.[4] RobotJ plugs into IBM s open-sourced Integrated Development Environment (IDE) known as *Eclipse*. By embracing IBM s IDE, shown in Figure 1, RobotJ will sit along side other development tools created by Rational and other vendors allowing easy tool integration into a common interface.

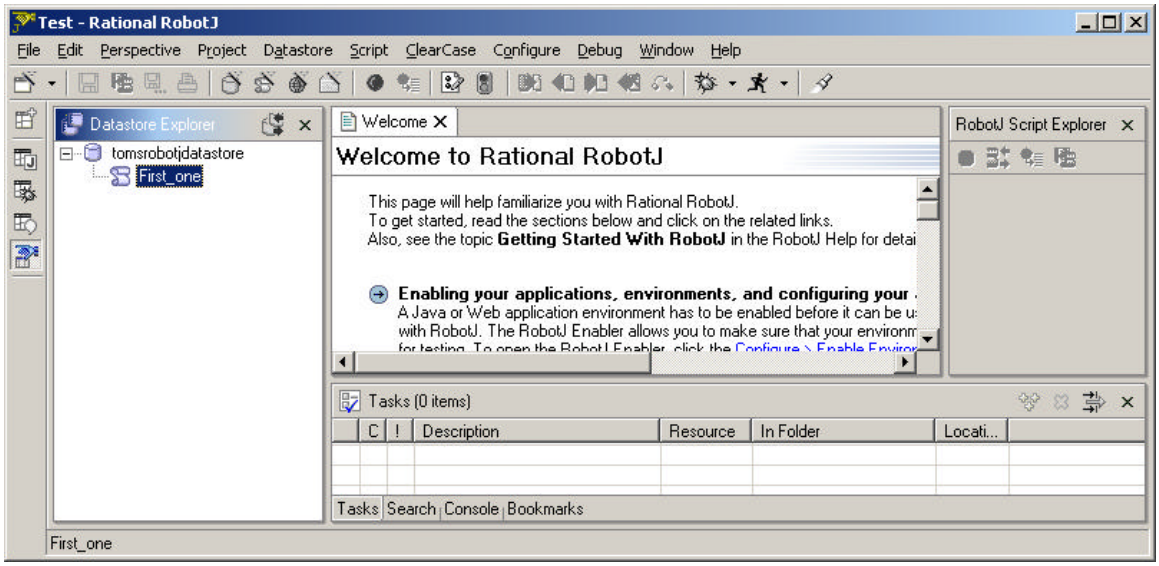

Figure 1*: RobotJ utilizes IBM s open-sourced IDE known as* Eclipse*.*

To begin creating a RobotJ test script it is necessary to first create a container where those scripts can be stored. This is referred to as a *Data Store*. Such a project file must be created before any headway can be made. In Figure 1, you can see that I ve created a data store called *tomsrobotjdatastore* (creative, eh?). Within that data store is my first recorded script, *First_one*, shown in Listing 1. I ve skipped writing a description of the steps for creating a data store because I want to get right into the meat of RobotJ.

Getting Ready to Record

The first thing I did in creating my automated script was to enter the *Configuration Editor*, shown in Figure 2 on the next page. Selecting the *Configure Applications for Testing* menu item found under the *Configure* menu in RobotJ accesses this dialog box. Once in the dialog, I clicked the *Add* button and was given the option of creating an entry for a Java application, HTML application or Windows executable.

---

[4] Java applications developed with Sun JDK/JRE 1.2.2 and greater, and IBM JRE 1.2.2 and greater are supported. GUI-based Java applications using or extending the AWT, Swing and SWT libraries are supported. There is HTML support for Microsoft Internet Explorer 4.0, 5.0, 5.5, 6.0 as well as Netscape Navigator 6.1.

For this example I created an entry for my website www.xtenddev.com. This would allow me to select my application from a list of applications maintained by the Configuration Editor once the recording interface was displayed.

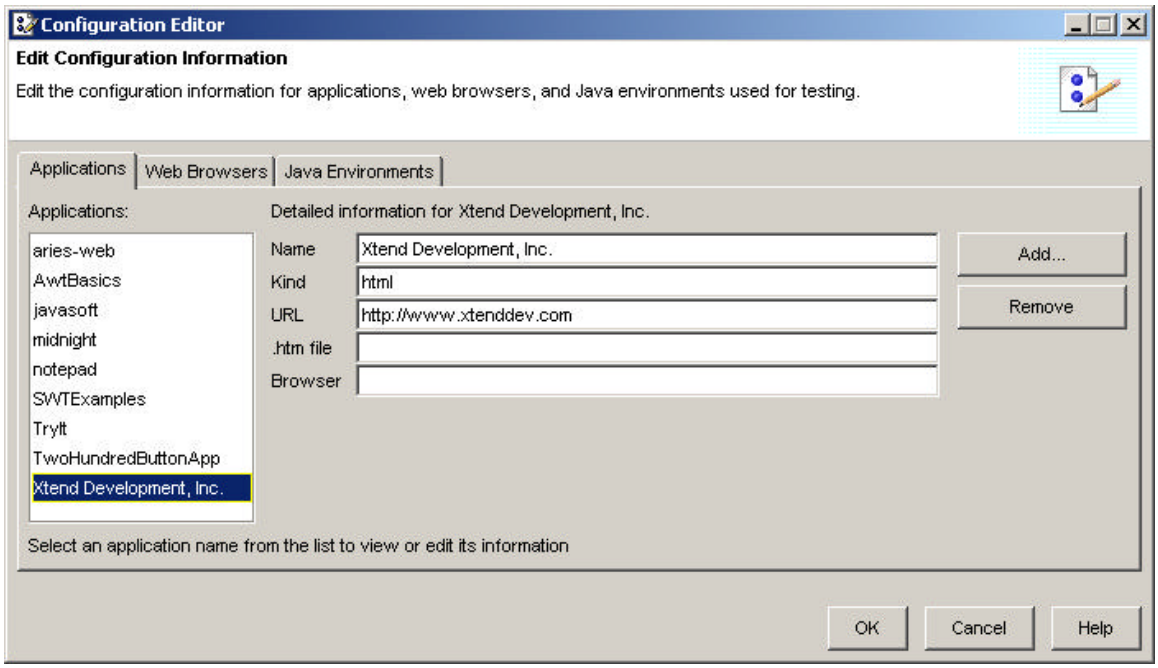

***Figure 2**: It is necessary to enter start-up information about the application that will be tested by RobotJ. This is used by the Recorder and allows you to specify which application to test.*

Record a New RobotJ Script

With some of the initial settings out of the way, I was ready to begin automating a test on my website. The goals, mentioned earlier, were to navigate to my website, search for product information, add an item to the e-commerce shopping cart, advance to check-out where customer shipping & billing information can be entered, submit the order and shut down the browser.

***Figure 3**: The (red) button on the toolbar just below the Script menu activates the recorder.*

I began the process by clicking the *Record New RobotJ* toolbar button found just below the *Script* menu, as shown in Figure 3. This resulted in a dialog box being displayed prompting for the name of the script. For my example I typed in First_one (no spaces or non-alphanumeric characters are allowed) and clicked the *Finish* button to begin the recording session.

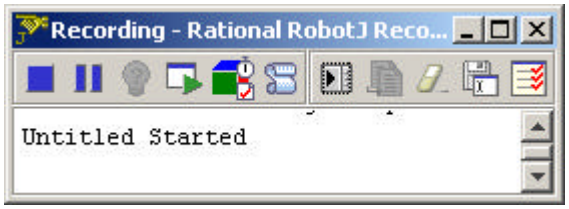

***Figure 4**: The Recording window is displayed while the RobotJ recorder is active.*

An always-on-top window with *Recording* as its caption, along with a toolbar chock-full of fun-looking buttons was displayed, as shown in Figure 4. This window allowed me to

control key aspects of how my script was recorded.

I clicked the fourth button from the left in the Recording window shown in Figure 4. This was the *Start Application* button that displayed the (no surprise here) *Start Application* dialog box. This dialog box listed the application I had configured earlier using the Configuration Editor. As shown in Figure 5, I selected the application I wanted to test and clicked the *Ok* button.

If I stopped the recorder at this point the generated script would be what is shown in Listing 2:

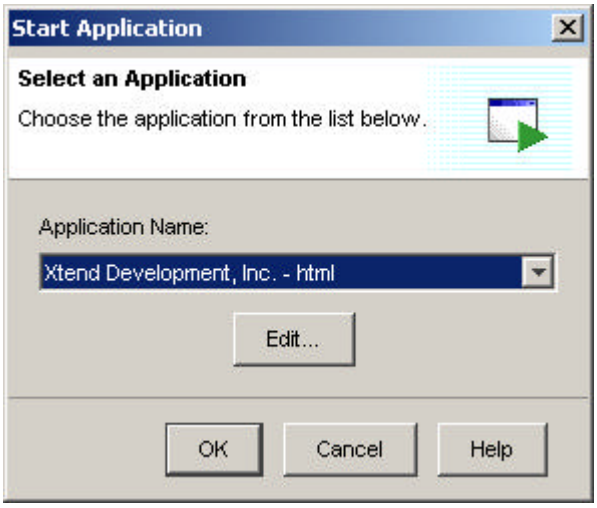

***Figure 5**: The* Start Application *dialog can be displayed using the* Start Application *button on the Recording window s toolbar.*

Listing 2

```
import resources.First_oneHelper;
import com.rational.test.ft.*;
import com.rational.test.ft.object.interfaces.*;
import com.rational.test.ft.script.*;
import com.rational.test.ft.value.*;
import com.rational.test.ft.vp.*;

public class First_one extends First_oneHelper
{
      public void testMain (Object[] args)
      {
        startApp("Xtend Development, Inc.");
```

The first six lines make RobotJ s secret sauce available to us in the form of Java packages, which are groupings of related APIs (Application Programmers Interface). Each package and what it provides is listed below in Table 1.

Rational RobotJ Package	Description
`Resources.First_oneHelper`	Auto-generated class based on test app s UI
`Com.rational.test.ft`	Base exception classes used by RobotJ
`Com.rational.test.ft.object.interfaces`	Classes that interact w/ software under test
`Com.rational.test.ft.script`	Classes the manage the test script
`Com.rational.test.ft.value`	Interfaces to access and manage values
`Com.rational.test.ft.vp`	Interfaces that represent Verification Points

***Table 1**: Core packages included in scripts generated by the Recorder functionality.*

The very first item is the `Resources.First_oneHelper` class. This class is created when the recorder analyzes the application being tested. For each control that exists in the user interface, a method is created using the name of that control. This method can then be used to interact with the control. Because RobotJ is using object-oriented programming the way it was meant to be used, each control becomes a black box and any changes made to the class has little or no effect on the method being used. The other packages can be found in the extensive on-line help provided with RobotJ.

Referring again to Listing 2, just beyond the `import` statements, are the lines:

```
public class First_one extends First_oneHelper
{
      public void testMain (Object[] args)
      {
                  startApp("Xtend Development, Inc.");
```

The first line derives a new class (`First_one`) based on the auto-generated `First_oneHelper` class. The `First_oneHelper` class was created when RobotJ analyzed the application being tested and created a number of methods to provide easy access to the application s controls. (`First_oneHelper` is, itself, derived from RobotJ s `Helper` base class). Within this new `First_one` class a method called `testMain()` is created. The first command in the `testMain()` function is a call to `startApp()` with a parameter that matches the entry I created in the Configuration Editor (“`Xtend Development, Inc.`”). Now if I go and change where the application is located (if the URL changes, for example), I need only change the values in the Configuration Editor and my script continues to run, unaffected.

The next line in the script finds the browser that was created when `startApp()` was called. (I clicked on the window to verify it had focus):

```
Browser_htmlBrowser(Document_XtendDevelopmentInc(),DEFAULT).click(
                                          atPoint(382,11));
```

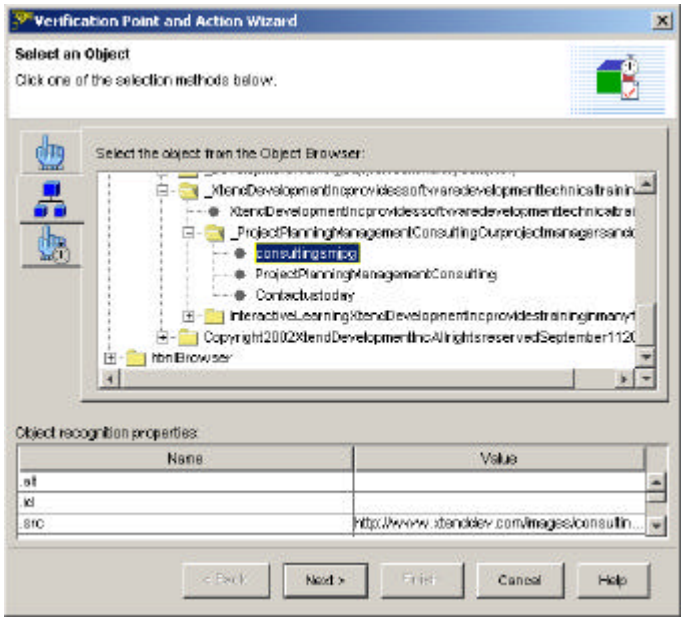

***Figure 6**: Verification Points help you ensure you are where you* think *you are.*

With the application up and running, I needed to verify I was where I expected to be.

Adding Verification Points

To verify a script is still synchronized, a *Verification Point* (VP) can be added. This is done using the Recording window from Figure 4 (fifth button from the left). The VP button displays the dialog in Figure 6.

On this dialog box there are three tabs. The first tab allows for clicking and dragging a

pointer to an object to be used in the verification process.
The second tab, in Figure 6, displays each object hierarchically as RobotJ has mapped them in the browser. To verify that I reached the Xtend Development home page, I selected the `consultingsm.jpg` graphic, which is unique to that web page. If RobotJ sees that graphic at this point in the script, it s a good bet things are on track.

The third tab a pointing finger with a stopwatch allows a delay to be added to the script. This delay can be configured to wait until a specific control appears. If the control does not appear, and the specified period of time goes by, an exception is thrown resulting in a failure logged by RobotJ.

Clicking the *Next* button on the dialog box shown in Figure 6 displays a final dialog allowing fine-tuning of the VP, shown in Figure 7.

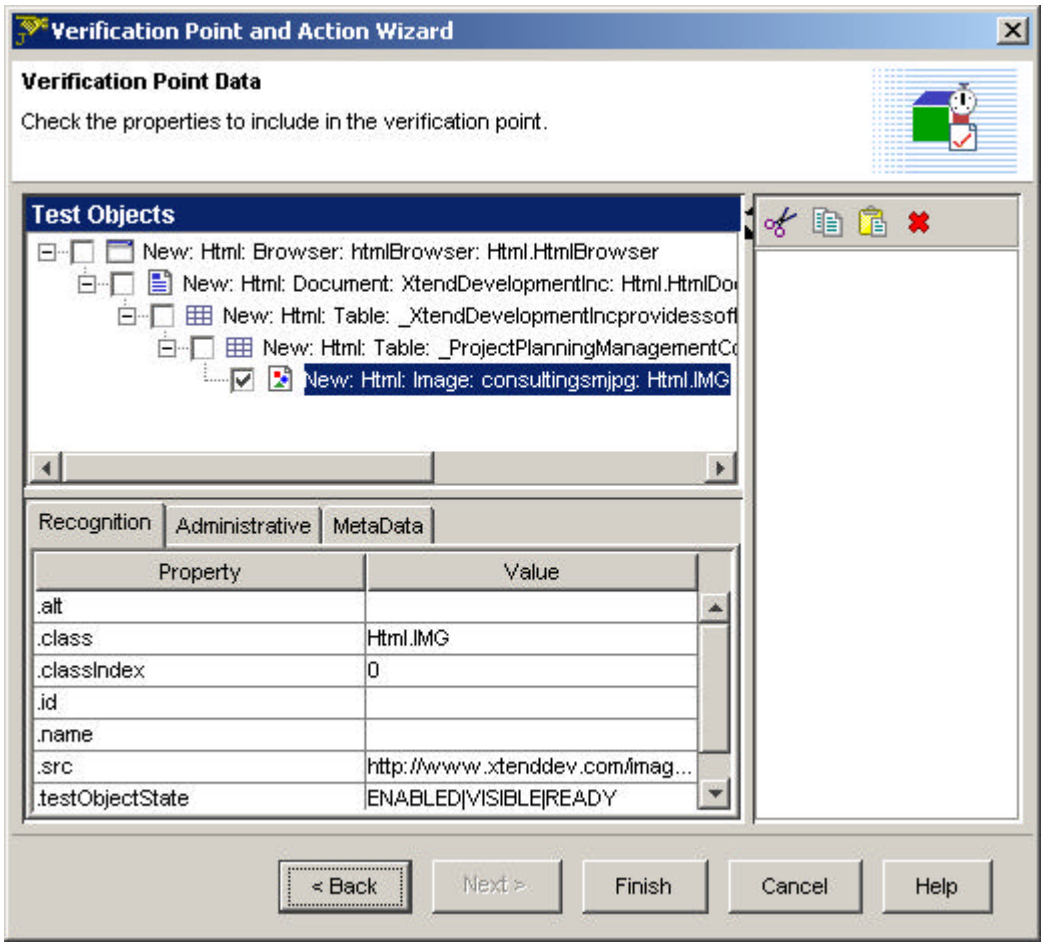

Figure 7*: Final step in adding a Verification Point (VP) to the recorded script.*

Tweaking the values and clicking the *Finish* button inserts a VP call into the script and resumes recording:

```
consultingsmjpg_textVP().performTest();
```

(Notice that the name of the object used in the verification is used to create an object `consultingsmjpg_textVP()` with an inherited method `performTest()`.) There is neither a right nor a wrong place to add a VP. They are typically added when a key navigation takes place so that execution can be stopped the minute it becomes apparent the train has gone off the tracks.

Test Object Maps

RobotJ s Recorder keeps a running tally of all objects acted upon during a recording session. Most of these objects are placed into a group known as a *Test Object Map*.

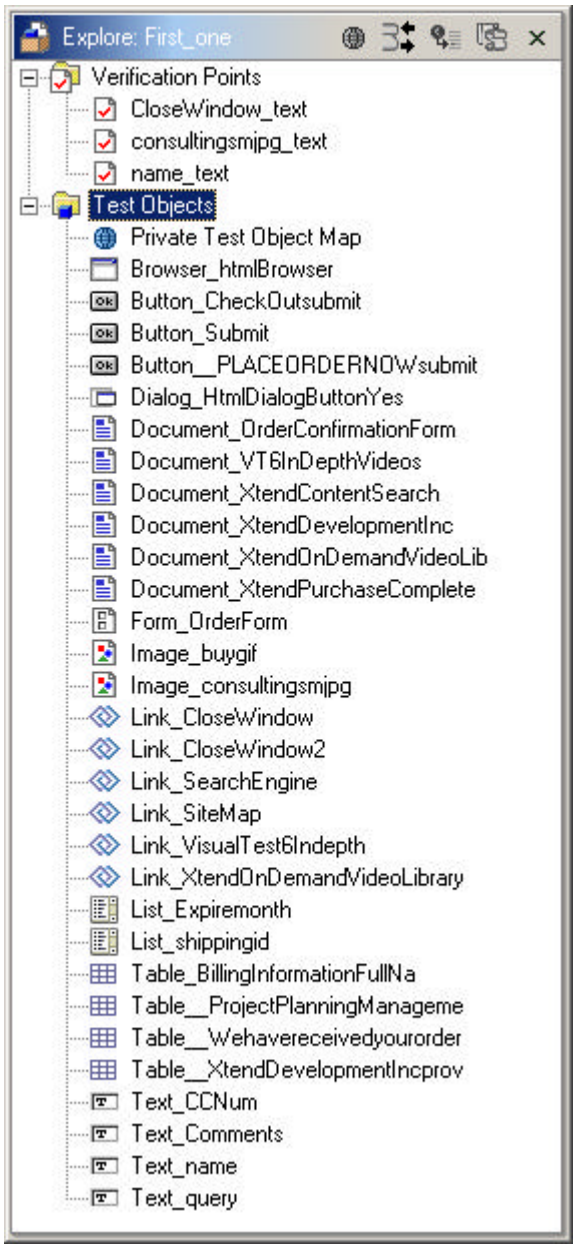

***Figure 8**: All objects on an application being tested are mapped to a function or method called within the automated script.*

Other items such as those created through the process of defining a Verification Point are added to the *Verification Points* section.

The Test Object Map tracks all aspects of an object, including its name, type, parent, siblings, and other properties. The name of the item is also used to create functions or methods that are used by the script to interact with the object.

Because RobotJ does not pay attention to any one attribute of an object, if a property changes through the course of development, the script will likely continue to run without error. However, warnings will be logged allowing the automation engineer to re-map and object and return RobotJ s match to 100%.

Figure 8 shows the object map that was created as a result of the script that I generated. Note that although there were hundreds of objects on the pages that I stepped through when creating the script, only those objects that I acted upon were included in the mapping.

Not shown here is the ability to control the weights applied to the different properties of a control when RobotJ is searching for a match. These weights allow the scriptwriter to specify what RobotJ must pay attention to, and what is less important when searching for a matching control during the script s playback. This is *very* cool: scripts break less often.

The Rest of the Code

The rest of the source code generated by the Recorder is fairly straight forward, now that you understand where the function names come from. The recorded script is essentially self-documenting, even though RobotJ is kind enough to include helpful comments throughout. See Listing 3 for the final lines (some comments have been removed to save space).

Listing 3

```
Link_SiteMap().click();
Link_SearchEngine().click();
Text_query().click(atPoint(51,14));

Browser_htmlBrowser(Document_XtendContentSearch(),DEFAULT).inputKeys("rational");
Button_Submit().click();
Link_XtendOnDemandVideoLibrary().click();
Document_XtendOnDemandVideoLib().drag(atPoint(754,247), atPoint(753,450));
Link_VisualTest6Indepth().click();
Document_VT6InDepthVideos().drag(atPoint(755,137),atPoint(754,203));
Image_buygif().click();
Button_CheckOutsubmit().click();
Text_name().click(atPoint(29,13));
name_textVP().performTest();
Text_name().click(atPoint(35,10));
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("Thomas
Arnold");
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("{Tab}Xtend
Development, Inc.{Tab}206-938-2370{Tab}206-932-8797");
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("{Tab}206-938-
1191{Tab}tom@xtenddev.com{Tab}tom@xtenddev.com{Tab}");
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("4742 42nd
Avenue SW{Tab}#621{Tab}Seayt{BKSP}{BKSP}ttle{Tab}WA");
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("{Tab}98116{Ta
b}{Tab}{Tab}{Tab}{Tab}");
Document_OrderConfirmationForm().drag(atPoint(754,178),atPoint(731,301));
Text_CCNum().click(atPoint(14,9));
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("{Num4}{Num3}{
Num8}{Num8}{Num5}{Num4}{Num3}{Num0}{Num2}{Num3}");
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("{Num7}{Num7}{
Num0}{Num3}{Num5}{Num7}");
List_Expiremonth().click();
List_Expiremonth().click(atText("05"));
Document_OrderConfirmationForm().drag(atPoint(752,364),atPoint(752,444));
List_shippingid().click();
List_shippingid().click(atText("United States (lower 48 states) (Overnight) $36.00"));
Text_Comments().click(atPoint(154,58));
Browser_htmlBrowser(Document_OrderConfirmationForm(),DEFAULT).inputKeys("This is a
test order. Delete it!{Enter}");
Document_OrderConfirmationForm().drag(atPoint(752,356),atPoint(753,478));
Button__PLACEORDERNOWsubmit().click();
CloseWindow_textVP().performTest();
// Document: Xtend / Purchase Completed:
http://www.xtenddev.com/purchase_completed.asp?name=Thomas+Arnold&company=Xtend+Develo
pment%2C+Inc%2E&email1=tom%40xtenddev%2Ecom&address1=4742+42nd+Avenue+SW&address2=%236
21&city=Seattle&state=WA&zip=98116&country=United+States&fax=206%2D938%2D1191&homePhon
e=206%2D932%2D8797&workPhone=206%2D938%2D2370&orderID=615514&cardType=&shipname=Thomas
+Arnold&shipaddress1=4742+42nd+Avenue+SW&shipaddress2=%23621&shipcity=Seattle&shipstat
e=WA&shipzip=98116&shipcountry=United+States&sku1=BDL%2DVT6%2DSET&product1=Videos+%2D+
Bundle+%2D+VT6+Videos+%2B+Book+%26+CD&quantity1=1&option1=&optionA1=&optionB1=&price1=
749%2E00&total=749%2E00&tax=65%2E91&shippingMethod=United+States+%28lower+48+states%29
+%28Overnight%29&shippingAmount=36%2E00&gst=0%2E00&grandTotal=850%2E91
Link_CloseWindow2().click();

Dialog_HtmlDialogButtonYes().click();
```

I did leave one line of comments in place in Listing 3 to show you that RobotJ s Recorder keeps track of the HTML headers returned after a form is submitted. The large comment block shows the information returned back to my scripted page (in this case an .ASP page, but this would work for a .JSP page as well). This type of comment would be very helpful during testing to understand values being returned that don t appear on the URL line (and are hidden in the HTTP headers). (Note: The comment block at the end of Listing 3 is one long line that wrapped in my word processor. That is why the // comment marks are used instead of /* and */.)

Execution & Results

Now that the script has been generated, the only thing left to do is run it and look at the results. Script execution starts when the Running Man icon is clicked (this button is found on the application s toolbar). This causes RobotJ s IDE to minimize itself and display a separate window that shows the status of the executing script.

***Figure 9**: The Running Man icon on the toolbar is clicked to start execution of the RobotJ test script.*

When the script completes its run, a separate application is launched that displays the results log and any errors it has encountered. Figure 10 shows a typical results file after successfully running the test script.

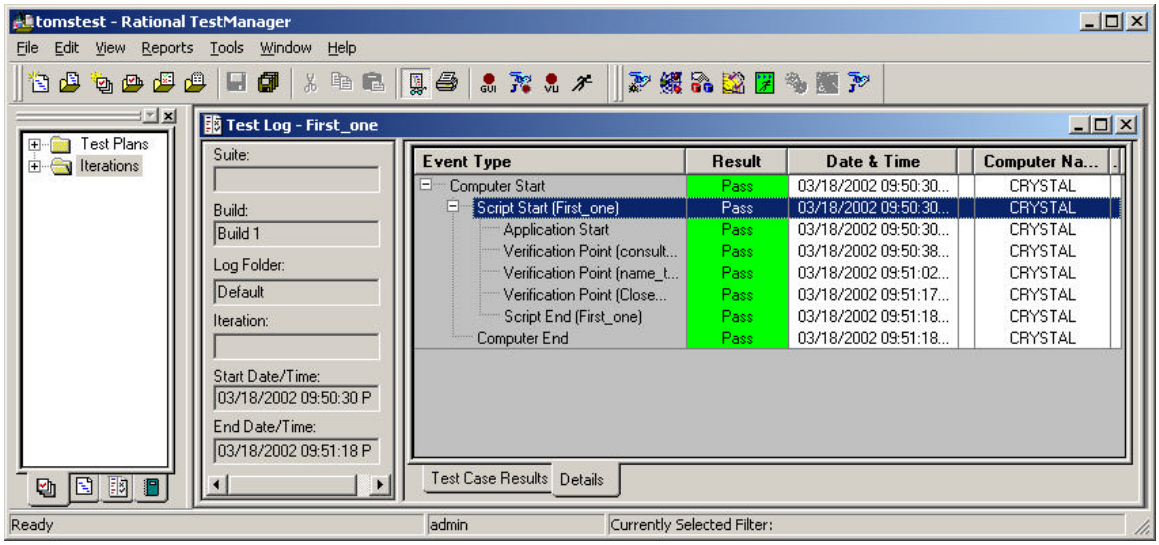

***Figure 10**: Rational Test Manager displays the results of RobotJ s test execution.*

Summary

In this document I showed you a scenario to be automated, displayed the final script first to show you want could be generated in only minutes, then stepped through the configuration, recording, and execution of that script.

Although there is much, much more to cover in using Rational's latest automation tool, I hope this document gives you a good introduction to what it has to offer. The robustness of the Recorder makes script generation a breeze, provides excellent examples for those who like to bang out code and avoid recorders, and uses a model that proves flexible as a test application evolves.

Thank you for giving me the sneak peek!

IBM software integrated solutions
IBM Rational supports a wealth of other offerings from IBM software. IBM software solutions can give you the power to achieve your priority business and IT goals.

- *DB2® software helps you leverage information with solutions for data enablement, data management, and data distribution.*
- *Lotus® software helps your staff be productive with solutions for authoring, managing, communicating, and sharing knowledge.*
- *Tivoli® software helps you manage the technology that runs your e-business infrastructure.*
- *WebSphere® software helps you extend your existing business-critical processes to the Web.*
- *Rational® software helps you improve your software development capability with tools, services, and best practices.*

Rational software from IBM
Rational software from IBM helps organizations create business value by improving their software development capability. The Rational software development platform integrates software engineering best practices, tools, and services. With it, organizations thrive in an on demand world by being more responsive, resilient, and focused. Rational's standards-based, cross-platform solution helps software development teams create and extend business applications, embedded systems and software products. Ninety-eight of the Fortune 100 rely on Rational tools to build better software, faster. Additional information is available at www.rational.com and www.therationaledge.com, the monthly e-zine for the Rational community.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Printed in the United States of America
01-03 All Rights Reserved.
Made in the U.S.A.

The IBM home page on the Internet can be found at **ibm.com**

---

[1] Software Testing with Visual Test 4.0, ISBN 0-7645-8000-0, IDG Books Worldwide, Developers Press Division, 500 pages, 3 / floppy disk.
[2] Visual Test 6 Bible, ISBN 0-7645-3255-3, Hungry Minds, Developers Press Division, 700 pages, CD-ROM. http://www.amazon.com/exec/obidos/ASIN/0764532553
[3] VT6 InDepth, produced and distributed by Xtend Development, Inc. (formerly InDepth Productions). http://www.visualtest.com/