Customizing Rational Functional Tester scripts for data-driven testing

Level: Intermediate

Pradosh Tarkar (pradosht@in.ibm.com), Software Engineer, IBM India Software Labs

24 Oct 2006

Learn about an approach involving the use of datapools for implementing data-driven tests and the necessary customization required in Rational Functional Tester 6.1 scripts. There are a number of ways automation engineers can implement data-driven tests in Functional Tester. The article will talk about how comma separated files (CSV) can be imported in Functional Tester as a datapool. The article will also demonstrate an example of a script modified to read data values from this CSV datapool.

Introduction

Data-driven testing is one of the most important aspects of functional automation. It's the most important technique to implement an actual usage scenario by providing unique data sets to the automation, making it more realistic and closer to the end user usage pattern. It's also useful in scenarios where millions of records are required to be supplied to the application under test for complete coverage. IBM® Rational® Functional Tester provides a framework which is useful for creating automated data-driven test scripts. However, automation engineers are required to customize their scripts to suit their application and specific requirements.

What is a datapool?

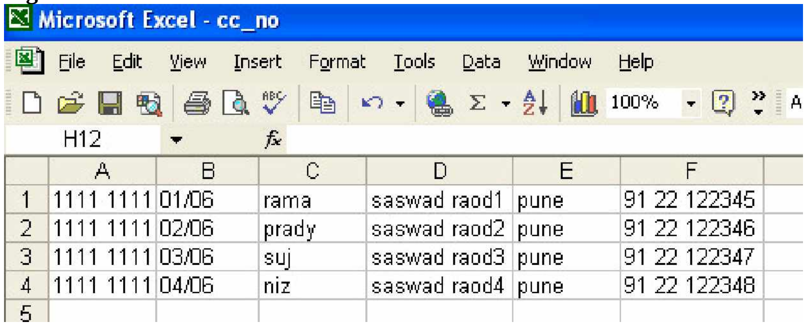
A datapool is nothing more than a test dataset. It's a collection of related data records which supplies data values to the variables in a test script during test script playback.

Creating a datapool

Automation engineers can use the following step-by-step procedure to import a CSV file to RFT.

1. Create a .CSV (comma separated values file) with the desired test data set. See Figure 1.

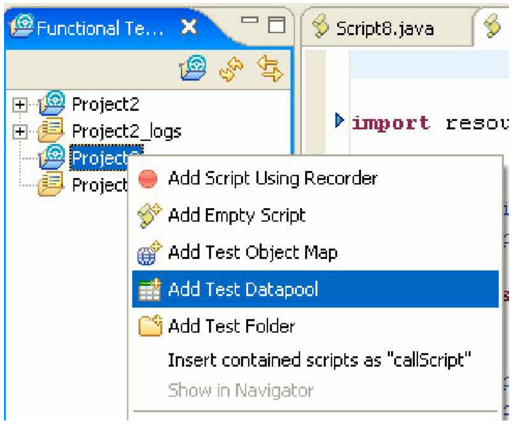
Figure 1. CSV file



	A	B	C	D	E	F
1	1111 1111	01/06	rama	saswad raod1	pune	91 22 122345
2	1111 1111	02/06	prady	saswad raod2	pune	91 22 122346
3	1111 1111	03/06	suj	saswad raod3	pune	91 22 122347
4	1111 1111	04/06	niz	saswad raod4	pune	91 22 122348
5						

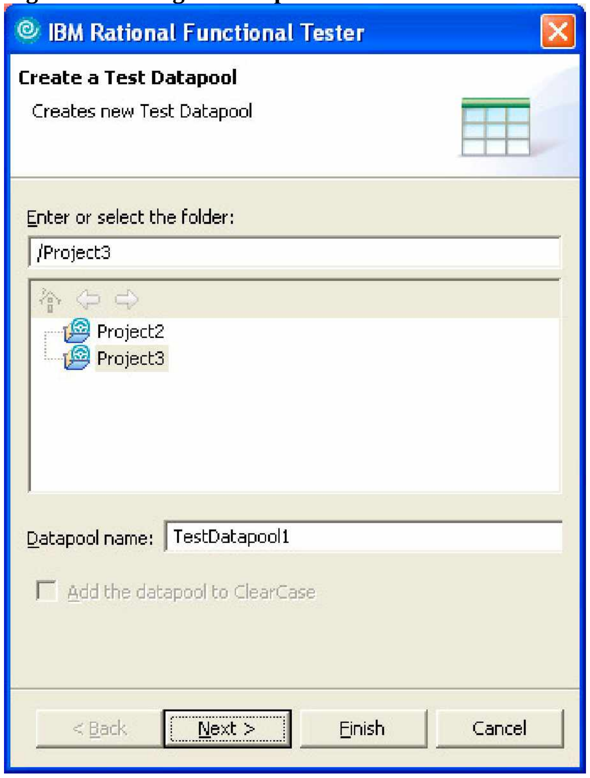
2. Open Functional Tester with the correct data store. Right click on **Project** and select **Add Test Datapool**.

Figure 2. Adding a test datapool



3. Select the project and enter a name for the datapool. Click **Next**.

Figure 3. Naming the datapool


IBM Rational Functional Tester

Create a Test Datapool

Creates new Test Datapool

Enter or select the folder: /Project3

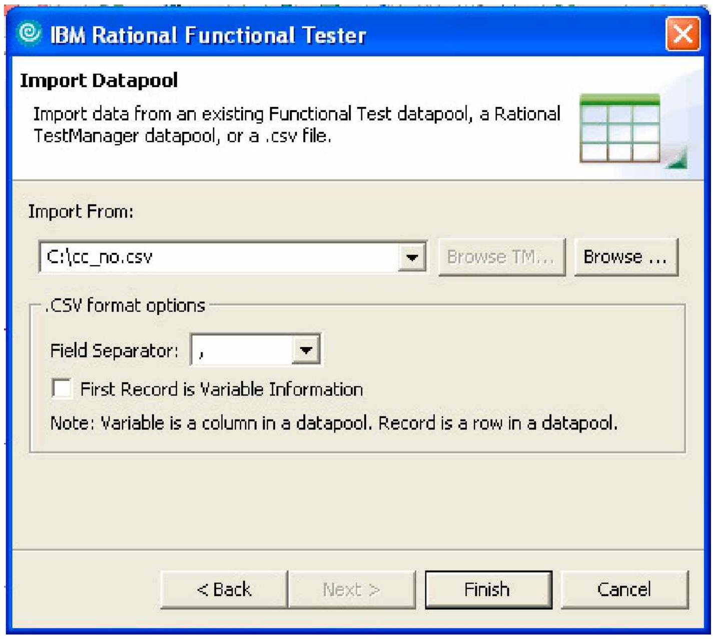
Project2
Project3

Datapool name: TestDatapool1

Add the datapool to ClearCase

4. Browse to the CSV file you wish to import, which contains the desired test data. Click **Finish**.

Figure 4. Finishing the import



IBM Rational Functional Tester

Import Datapool

Import data from an existing Functional Test datapool, a Rational TestManager datapool, or a .csv file.

Import From: C:\cc_no.csv

.CSV format options

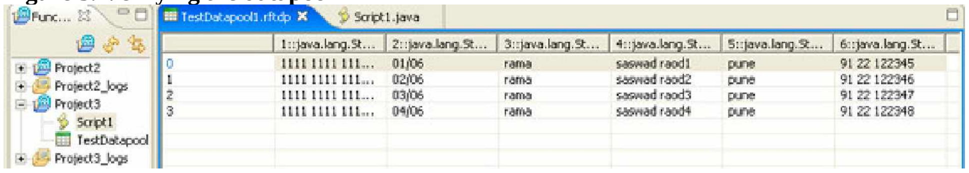
Field Separator: ,

☐ First Record is Variable Information

Note: Variable is a column in a datapool. Record is a row in a datapool.

5. Open the datapool to verify it.

Figure 5. Verifying the datapool

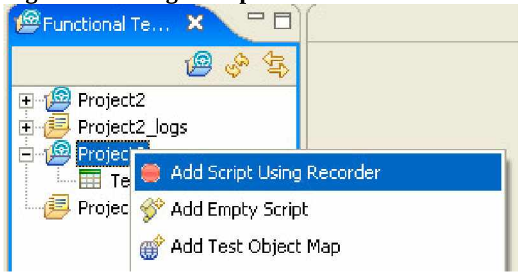


	1::java.lang.St...	2::java.lang.St...	3::java.lang.St...	4::java.lang.St...	5::java.lang.St...	6::java.lang.St...
0	1111 1111 111...	01/06	rama	saswad raod1	pune	91 22 122345
1	1111 1111 111...	02/06	rama	saswad raod2	pune	91 22 122346
2	1111 1111 111...	03/06	rama	saswad raod3	pune	91 22 122347
3	1111 1111 111...	04/06	rama	saswad raod4	pune	91 22 122348

Using the imported CSV file as a datapool

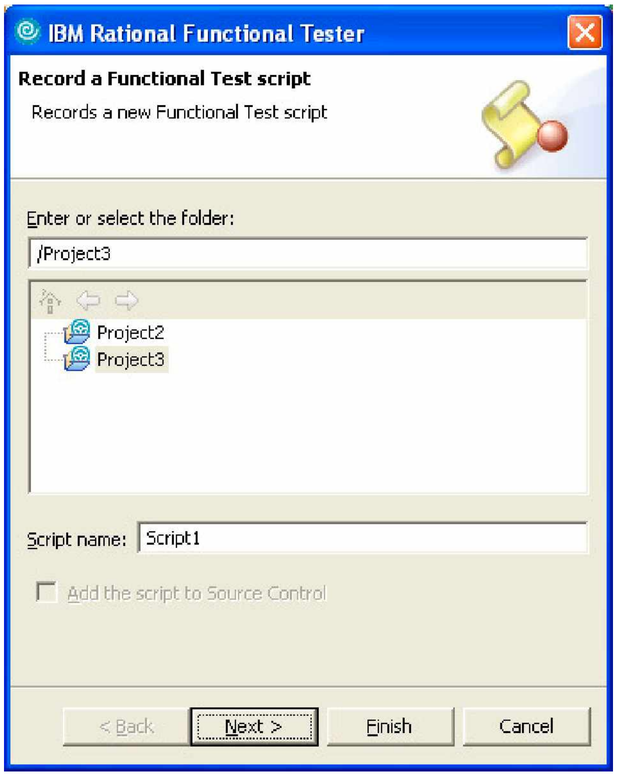
1. Right-click on the **Project** and select **Add Script Using Recorder**.

Figure 6. Adding a script



2. Select the project and enter a script name. Cick on **Next**.

Figure 7. Naming the script



Record a Functional Test script

Records a new Functional Test script

Enter or select the folder: /Project3

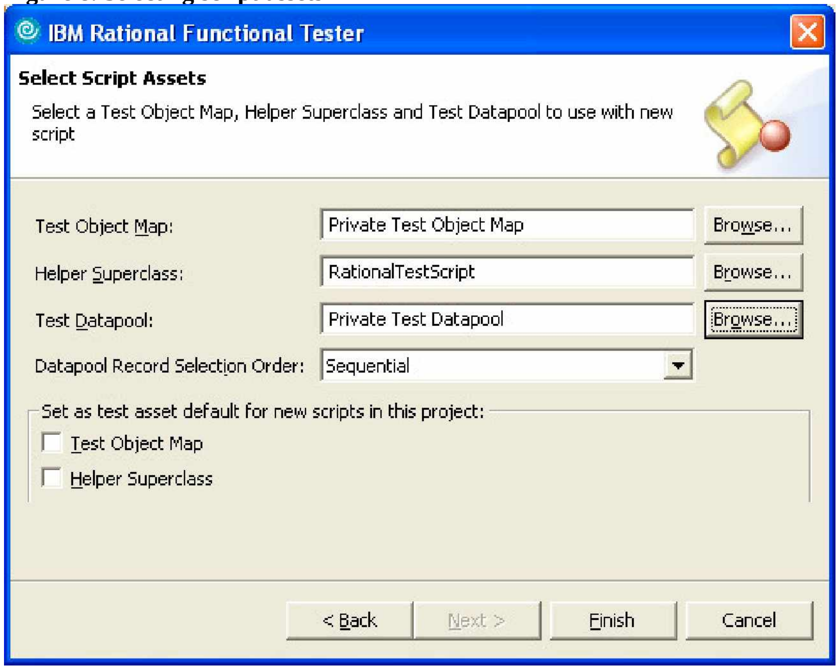
Project2
Project3

Script name: Script1

☐ Add the script to Source Control

3. Update the script assets Window pop-up with necessary details.

Figure 8. Selecting script assets



Select Script Assets

Select a Test Object Map, Helper Superclass and Test Datapool to use with new script

Test Object Map: Private Test Object Map

Helper Superclass: RationalTestScript

Test Datapool: Private Test Datapool

Datapool Record Selection Order: Sequential

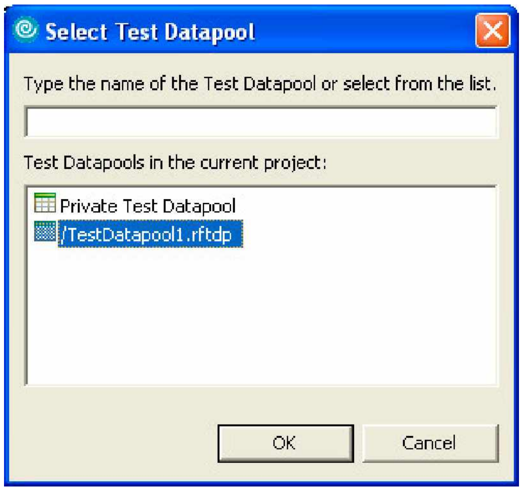
Set as test asset default for new scripts in this project:

☐ Test Object Map

☐ Helper Superclass

4. To select the test datapool, click on **Browse** and select the newly imported CSV datapool. Click **OK**.

Figure 9. Selecting the datapool


Select Test Datapool

Type the name of the Test Datapool or select from the list.

Test Datapools in the current project:

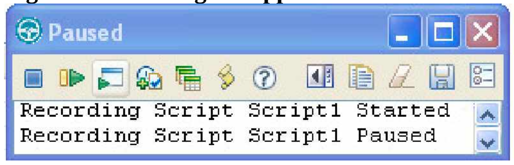
Private Test Datapool
/TestDatapool1.rftdp

5. Click **Finish**.
6. Open the application.

Figure 10. The application

7. Select the application and click **OK**. In this case, we're using the ClassicJavaB sample application.

Figure 11. Selecting the application



```
Recording Script Script1 Started
Recording Script Script1 Paused
```

8. Start recoding a script for the data input operations you wish to automate. We clicked on the **Place Order** button, selected the **New Customer** radio button and then clicked **OK**. On the next screen the application opens up a **Place an order** form. We entered values in **Card number**, **Expiration Date**, **Name**, **Street**, **City, State, zip** and **Phone** before clicking the **Place order** button. Finally we clicked **OK** for order confirmation.
9. The above script recording will generate the following recorded script:

Listing 1. The recorded script

```
import resources.Script1Helper;
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 Script1 extends Script1Helper
{
		public void testMain(Object[] args)
		{
				// Start Application
				startApp("ClassicsJavaB");

				// Click on Place Order
				placeOrder().click();

				// Click on New Customer radio button
				newCustomer().click();
				ok().click();

				// Enter the values
				cardNumberIncludeTheSpacesText().click(atPoint(43,10));
				placeAnOrder().inputKeys("1111 1111 1111 1111");
				expirationDateText().click(atPoint(33,9));
				placeAnOrder().inputChars("01/06");
				nameText().click(atPoint(95,6));
				placeAnOrder().inputChars("rama");
				streetText().click(atPoint(65,13));
				placeAnOrder().inputChars("saswad raod");
				cityStateZipText().click(atPoint(44,9));
				placeAnOrder().inputChars("pune");
				phoneText().click(atPoint(31,14));
```

```
                    placeAnOrder().inputChars("91 22 122345");
                    placeOrder().click();

                    //
                    ok().click();

                    // Frame: ClassicsCD
                    classicsJava(ANY,MAY_EXIT).close();
          }
}
```

10. This recorded script is useful for re-entering the static records which got hard coded in the script. For data-driven testing, automation engineers are required to provide unique records when the script gets executed. The above recorded script needs to be modified in order to provide the necessary data set from the datapool at runtime. The test script shown below is a customized script for that purpose. The modified part of the script is marked in bold.

Listing 2. Customized script

```
                    import org.eclipse.hyades.execution.runtime.datapool.IDatapoolIterator;

import resources.Script1Helper;

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 Script1 extends Script1Helper
{


          public void testMain(Object[] args)
          {
          // DatapoolScriptSupport provides methods for accessing rows in an associated datapool.
|-------10--------20--------30--------40--------50--------60--------70--------80--------9|
|-------- XML error:  The previous line is longer than the max of 90 characters ---------|
          DatapoolScriptSupport dpss = new DatapoolScriptSupport();

          //Declare "dp" as an object for IDatapool
               (org.eclipse.hyades.execution.runtime.datapool.IDatapool)
org.eclipse.hyades.execution.runtime.datapool.IDatapool dp;

          //Create a file object with complete TestDatapool file path
java.io.File dpfile = new java.io.File("D:\\Documents and Settings\\
pradosht\\IBM\\rationalsdp6.0\\workspace\\Project3\\TestDatapool1.rftdp");

          //Load the Testdatapool
          dp = dpss.dpFactory().load(dpfile,true);

          //Open the Test Datapool
          IDatapoolIterator dpitr = dpss.dpFactory().open(dp,"");

          //Initialize Test Datapool
          dpitr.dpInitialize(dp);

//Starting application
          startApp("ClassicsJavaB");


          while(!dpitr.dpDone())
          {
                    // Click on Place order
                    placeOrder().click();

                    // Click on New Customer
                    newCustomer().click();
                    ok().click();

                    //Get the current record & store it in record object
                    IDatapoolRecord dprec = dpitr.dpCurrent();

                    // Frame: Place an Order - Click on Card Number
                    cardNumberIncludeTheSpacesText().click();

                    //By accessing the corresponding cell in the current record from excel sheet
|-------10--------20--------30--------40--------50--------60--------70--------80--------9|
|-------- XML error:  The previous line is longer than the max of 90 characters ---------|
                    placeAnOrder().inputKeys(dprec.getCell(0).getStringValue());

                    // Frame: Place an Order - Click on Date
                    expirationDateText().click();
                    //By accessing the corresponding cell in the current record from excel sheet
|-------10--------20--------30--------40--------50--------60--------70--------80--------9|
|-------- XML error:  The previous line is longer than the max of 90 characters ---------|
                    placeAnOrder().inputChars(dprec.getCell(1).getStringValue());

                    // Frame: Place an Order - Click on Name
                    nameText().click();

//By accessing the corresponding cell in the current record from excel sheet
                    placeAnOrder().inputChars(dprec.getCell(2).getStringValue());

                    // Frame: Place an Order - Click on Street
                    streetText().click();

//By accessing the corresponding cell in the current record from excel sheet
                    placeAnOrder().inputChars(dprec.getCell(3).getStringValue());

                    // Frame: Place an Order - Click on CityStateZip
                    cityStateZipText().click();

//By accessing the corresponding cell in the current record from excel sheet
                    placeAnOrder().inputChars(dprec.getCell(4).getStringValue());

                    // Frame: Place an Order - Click on Phone
                    phoneText().click();

//By accessing the corresponding cell in the current record from excel sheet
                    placeAnOrder().inputChars(dprec.getCell(5).getStringValue());
|-------10--------20--------30--------40--------50--------60--------70--------80--------9|
|-------- XML error:  The previous line is longer than the max of 90 characters ---------|
                    // Frame: Place an Order
                    placeOrder2().click();

                    //Order Confirmation
                    ok2().click();

                    //Iterating to the Next Record
                    dpitr.dpNext();

             }

                    // Close Application
                    classicsJava(ANY,MAY_EXIT).close();
          }
}
```

In this article you learned how to use datapools for implementing data-driven tests and the necessary customization required in Rational Functional Tester 6.1 scripts. We hope you found it helpful.

Resources

- The Rational products' trial downloads area is a great way to evaluate products.
- In the developerWorks Rational Performance Tester product area you'll find technical documentation, how-to articles, education, downloads, product information, and more.
- Participate in the developerWorks Rational forums to get involved in the developerWorks community.

About the author

Pradosh Tarkar is a System Software Engineer working in the IBM Workplace Component Designer team. He is based in India at the Software Labs (ISL) Pune and has expertise in software testing. He holds an Engineering degree in Electronics & a MBA in Systems. He also has Sun Certified Java Programmer 1.4 certification. He has experience in BVT, FVT, Globalization Verification Testing, accessibility testing, and automation tools like Rational Robot and Rational Functional Tester.