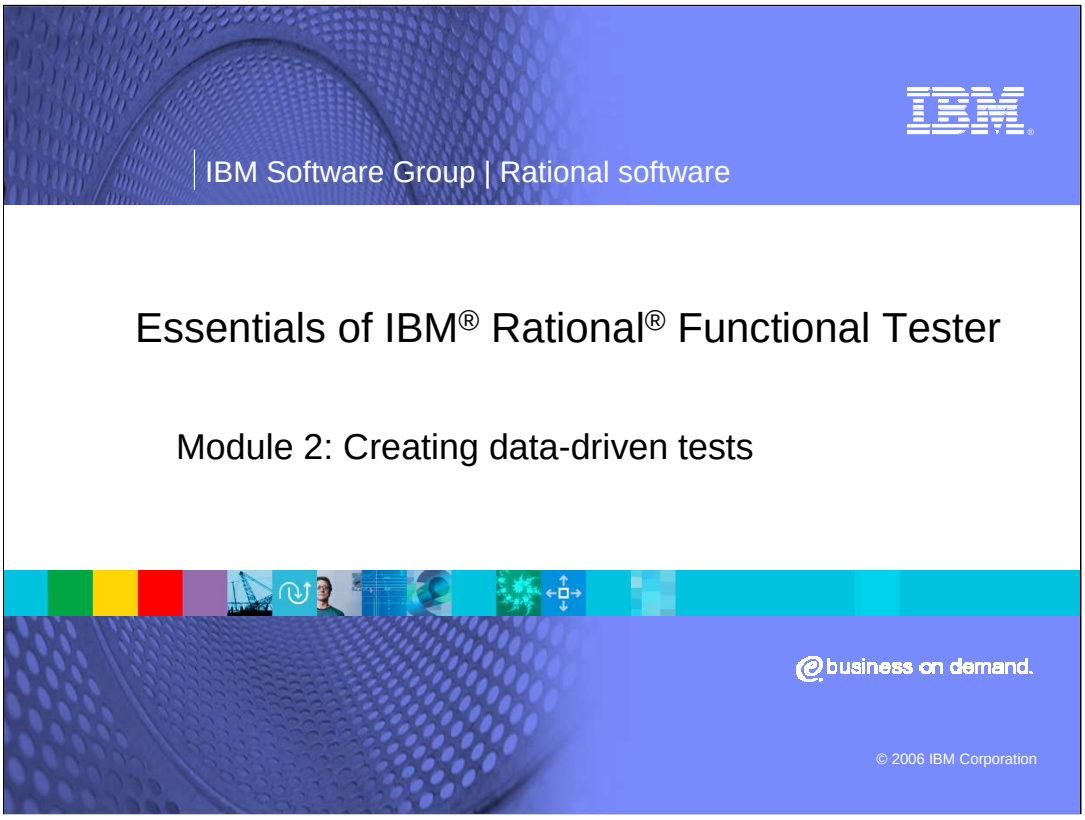
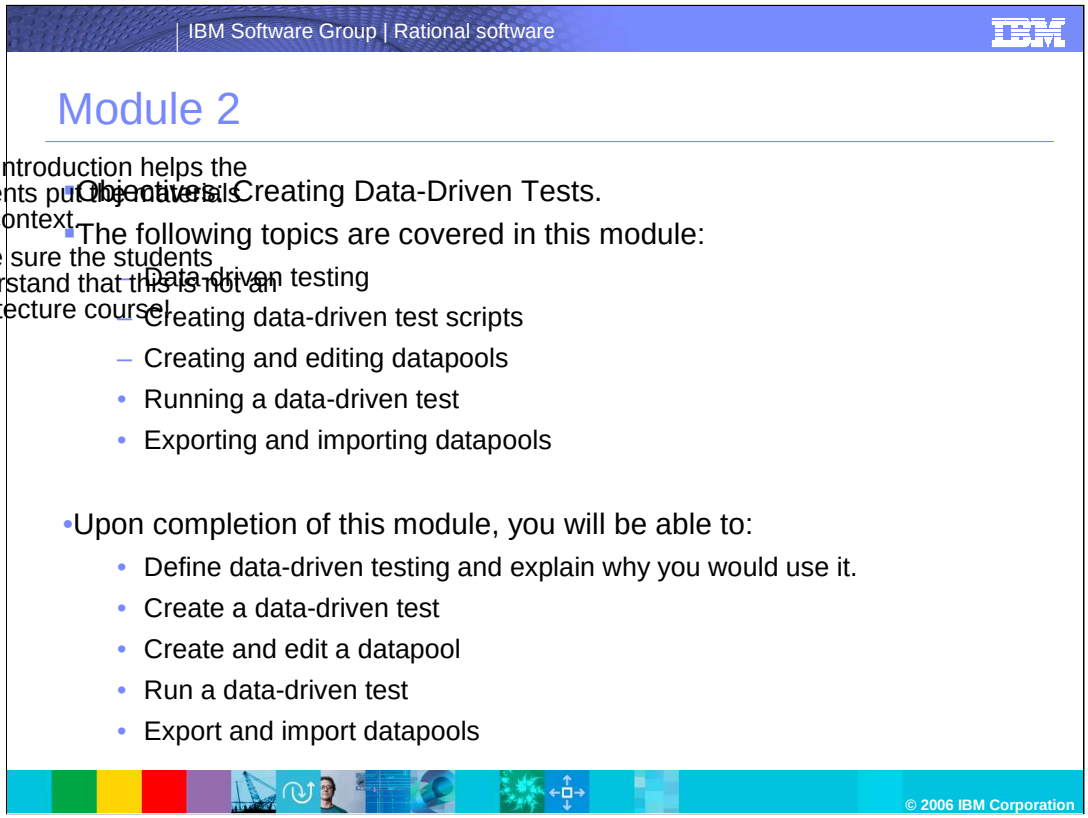


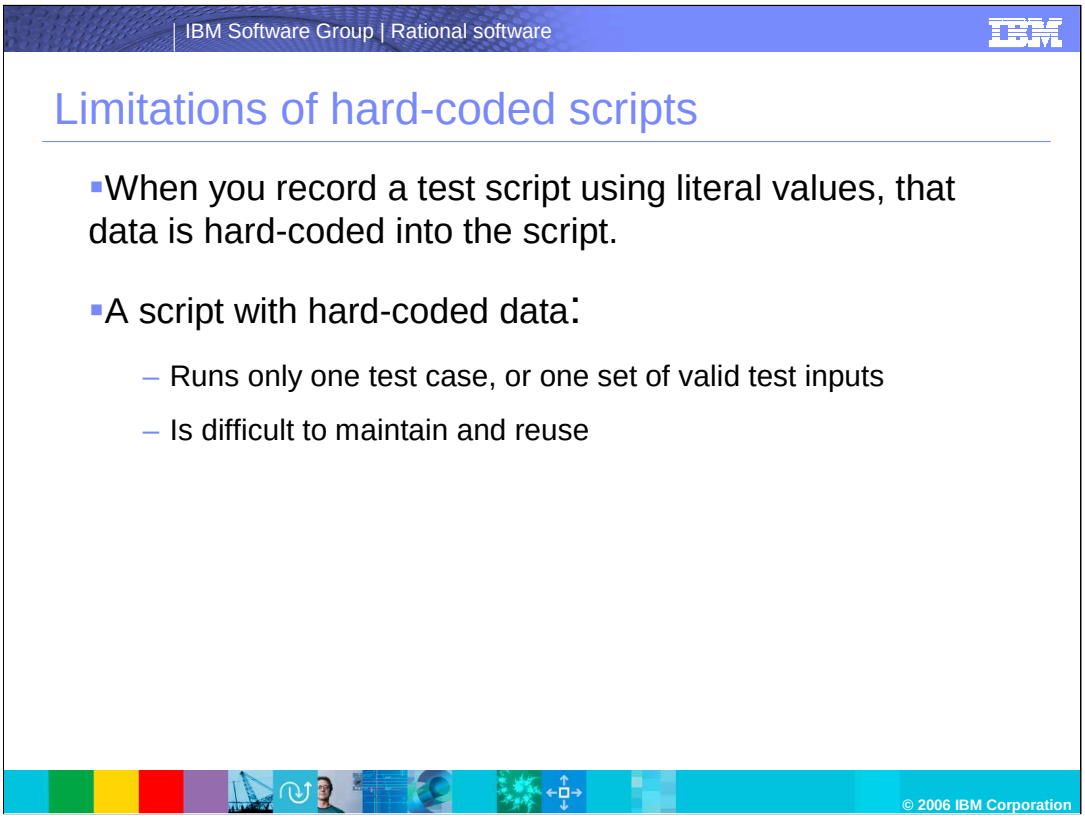
This presentation will provide an overview of scripting with Rational Functional Tester version 6.1 and describe the steps necessary to work with data-driven tests.



This introduction helps the students put the materials into context.

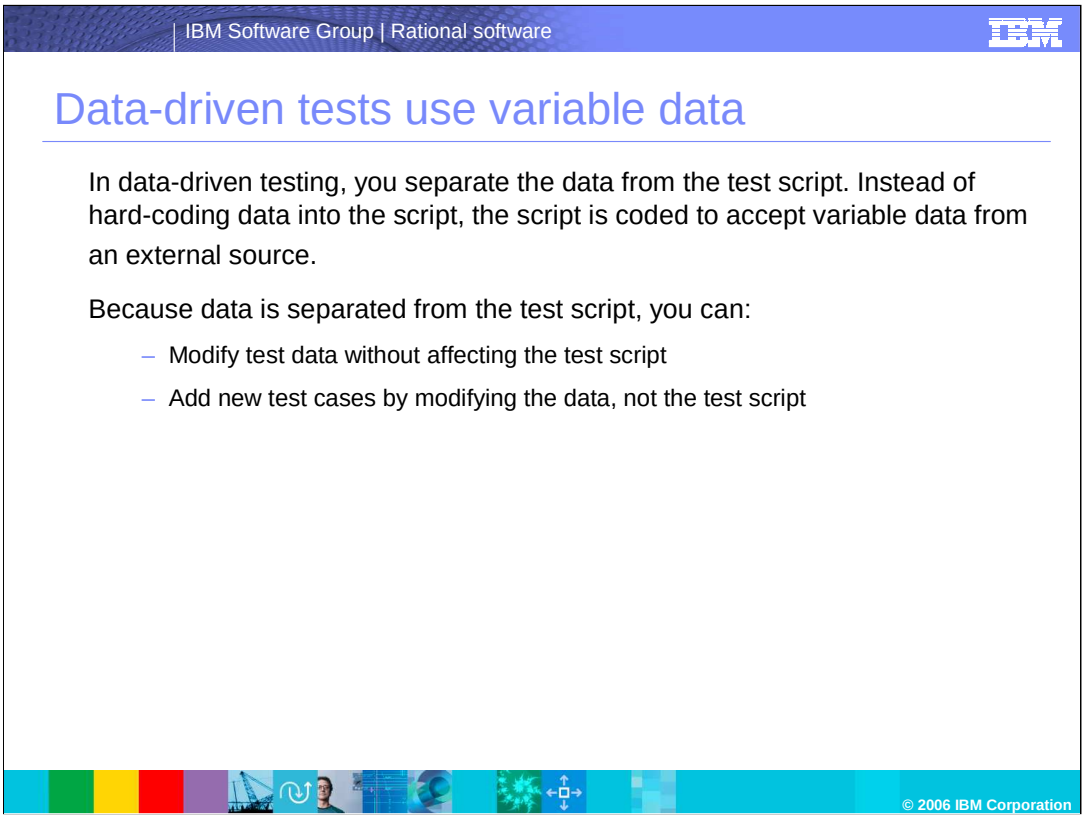
Make sure the students understand that this is not an architecture course!

The objective of this module is to introduce data-driven tests. Topics covered include data-driven testing, creating, editing, running, exporting, and importing datapools and data-driven test scripts.

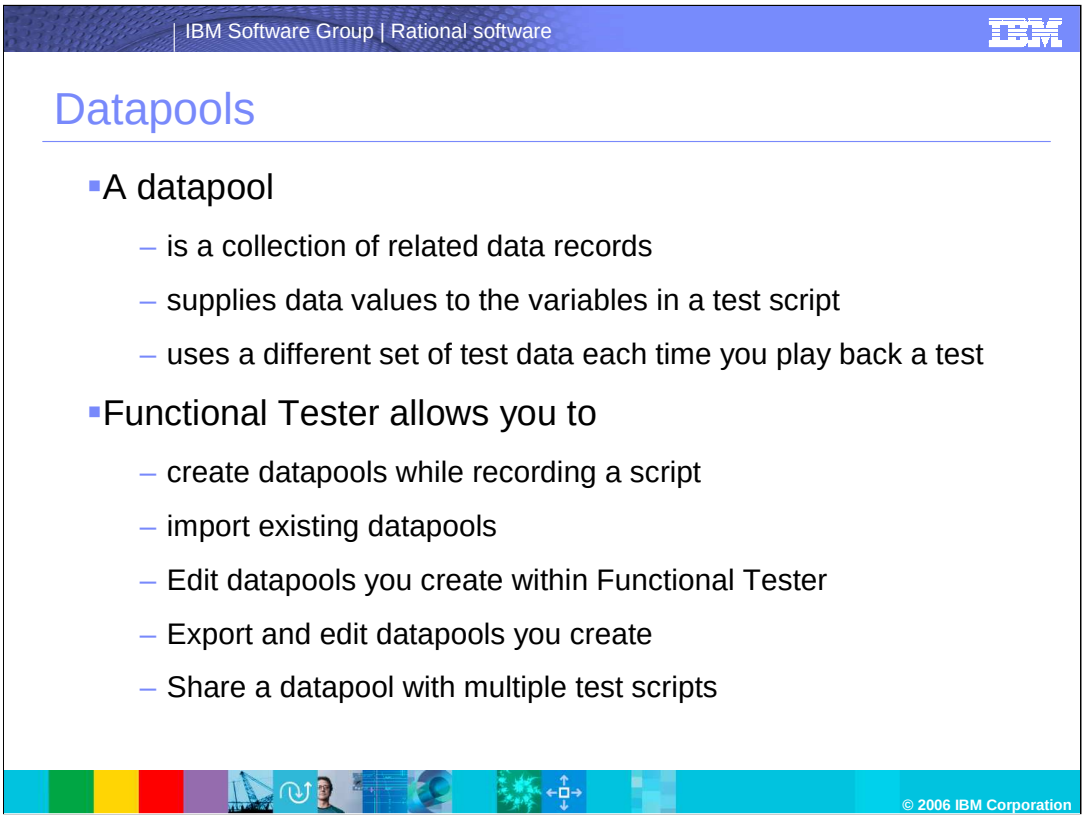


There are limitations to testing when using hard-coded scripts. This module will explore the benefits of data driven tests.

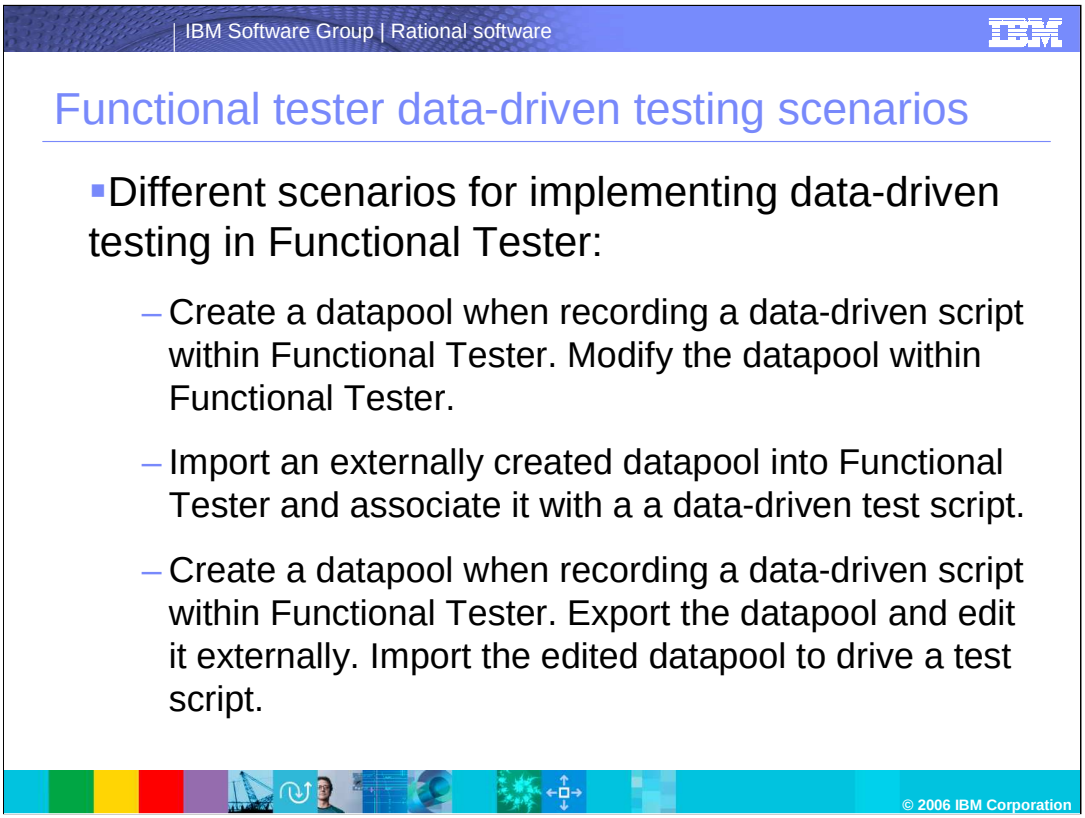
When recording a test script using literal values, that data is hard-coded into the script. A script with hard-coded data runs only one test case, or one set of valid test inputs, and is difficult to maintain and reuse.



When hard-coding data into the script, the script is coded to accept variable data from an external source. Because data is separated from the test script, you can modify test data without affecting the test script, and add new test cases by modifying the data, not the test script.



A datapool is a collection of related data records. A datapool supplies data values to the variables in a test script during script playback. Datapools automatically pump a different set of test data to a test script each time you play back a test. Rational Functional Tester makes it easy for you to create datapools while you are recording a script. Alternatively, you can import existing datapools. Rational Functional Tester also allows you to edit datapools you create within Functional Tester, export and edit datapools you create, and share a datapool with multiple test scripts.



There are different scenarios for implementing data-driven testing in Functional Tester, including:

Creating a datapool when recording a data-driven script within Functional Tester.

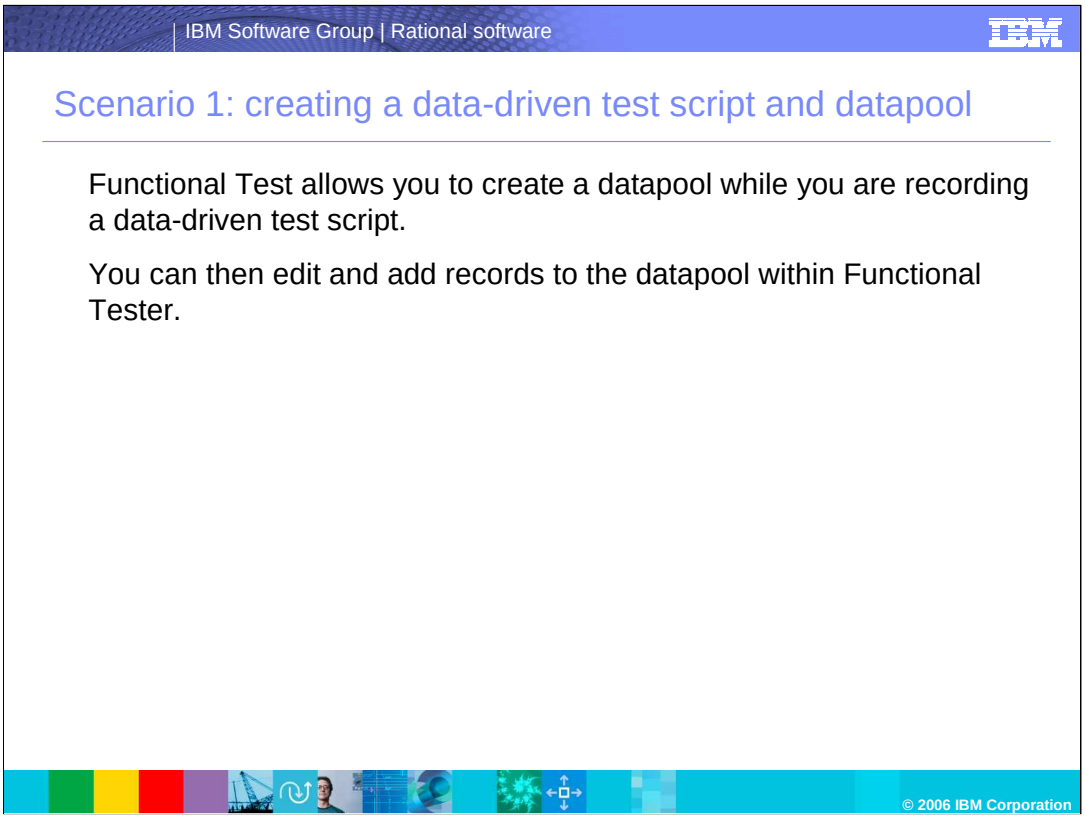
Modifying the datapool within Functional Tester

Importing an externally created datapool into Functional Tester and associating it with a data-driven test script

Creating a datapool when recording a data-driven script within Functional Tester.

Exporting the datapool and editing it externally

And, Importing the edited datapool to drive a test script.

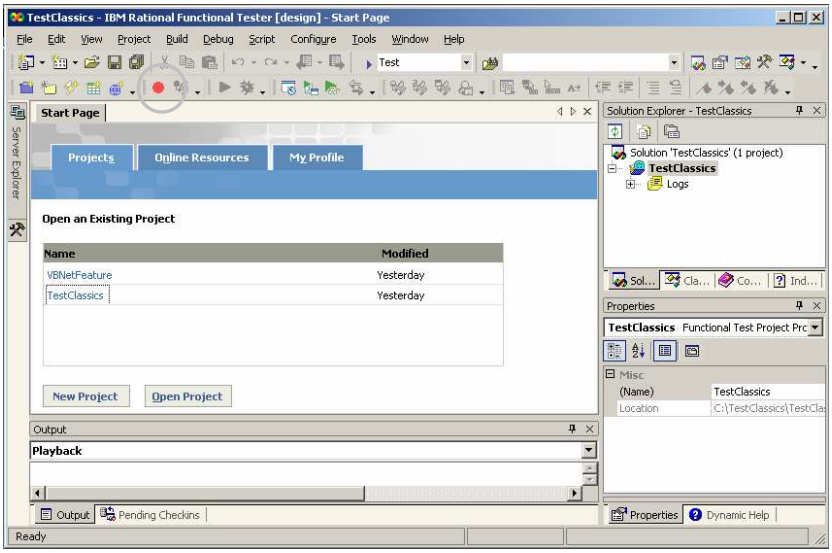


Scenario 1: Creating a data-driven test script and datapool.

This scenario takes you through the process of creating a datapool while you are recording a test script. You can then edit and add records to the datapool within Rational Functional Tester.

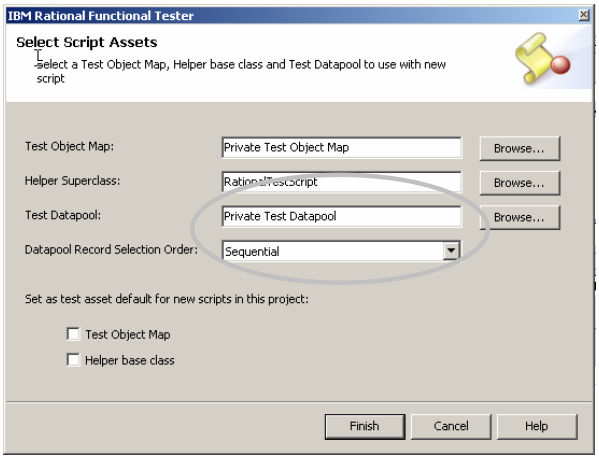
Recording a data-driven test script

Begin recording a test script by clicking the Record Test Script button on the Functional Test toolbar.



To begin recording a test script, click the Record Test Script button on the Functional Test toolbar.

Recording a data-driven test script (cont.)



A **Private Test Datapool** is associated with only one script.

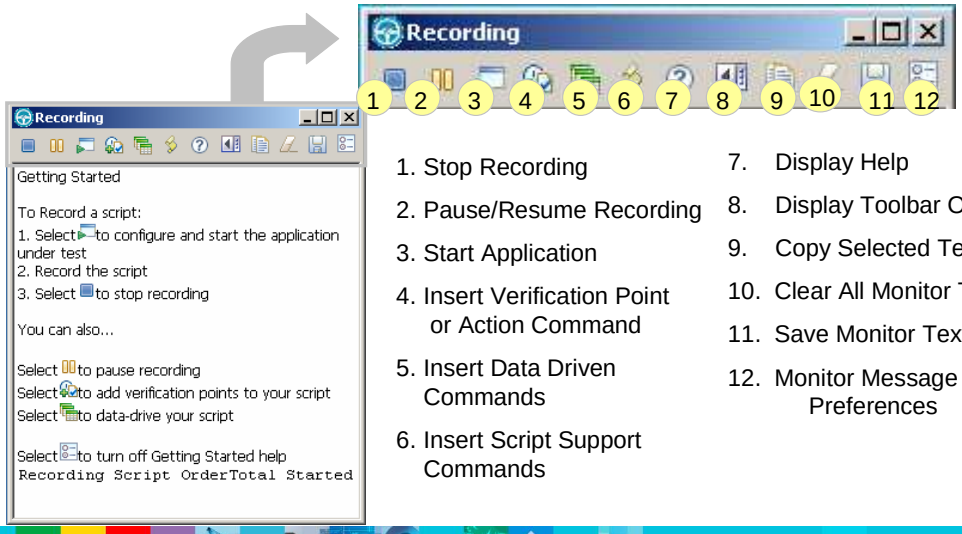
A **Shared Test Datapool** may be shared by many scripts.

When Sequential is selected, the script pulls records from the datapool in order. When random is selected, the script randomly accesses every record in the datapool once.

A **Private Test Datapool** is associated with only one script, while a **Shared Test Datapool** may be shared by many scripts. When the Sequential Order option is selected, the script pulls records from the datapool in order, whereas when the Random Order option is selected, the script randomly accesses every record in the datapool once. Make sure you choose the correct datapool you want to use in the Test Datapool field and the correct datapool selection order.

Recording a data-driven test script (cont.)

Once you have selected an application to record the script against, the Recording toolbar appears.

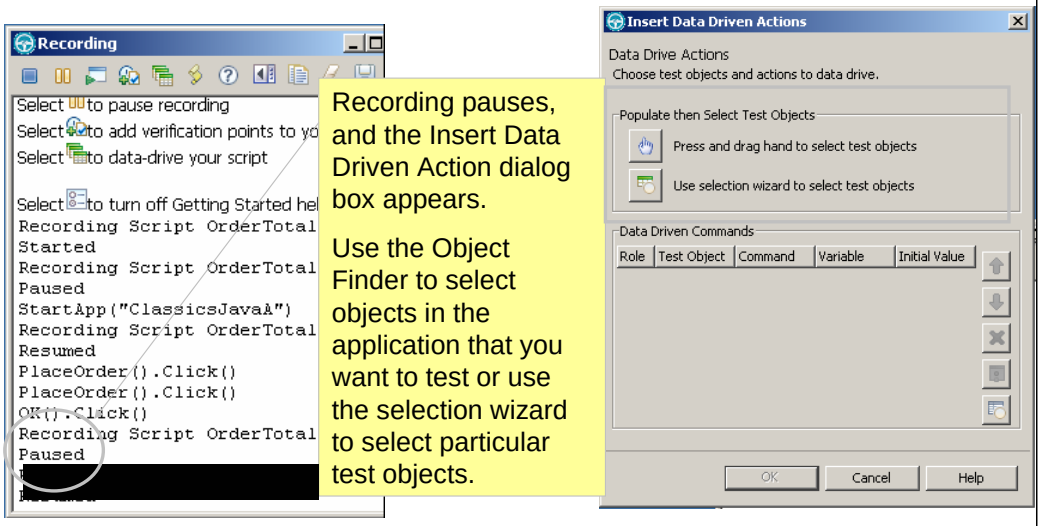


1. Stop Recording
2. Pause/Resume Recording
3. Start Application
4. Insert Verification Point or Action Command
5. Insert Data Driven Commands
6. Insert Script Support Commands
7. Display Help
8. Display Toolbar Only
9. Copy Selected Text
10. Clear All Monitor Text
11. Save Monitor Text As
12. Monitor Message Preferences

Once you have selected an application to record the script against, the Recording toolbar appears. You can perform a number of functions as indicated in the toolbar above such as starting and stopping, and inserting verification points.

Inserting data-driven commands

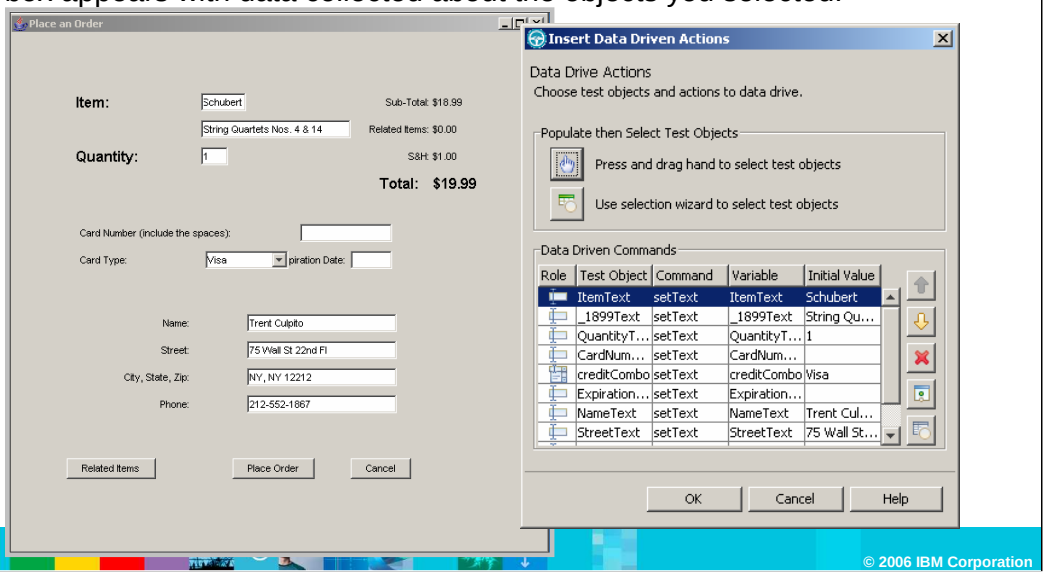
When you are ready to data-drive your script, click the **Insert Data Driven Commands** button on the Recording toolbar.



When you are ready to data-drive your script, click the **Insert Data Driven Commands** button on the Recording toolbar (shown on the right).

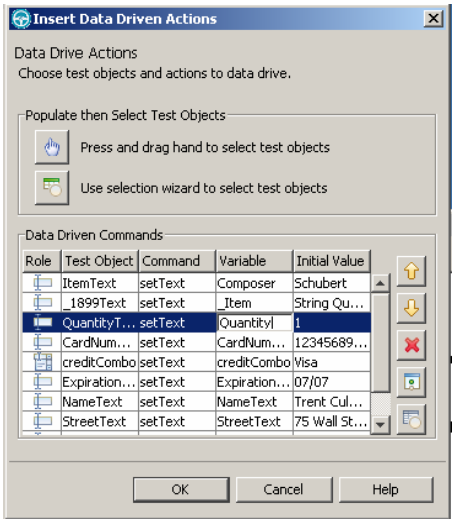
Inserting data-driven commands (cont.)

An object selected by the Object selector, is outlined in red. When you have selected the objects, release the mouse. The Insert Driven Actions dialog box appears with data collected about the objects you selected.



An object selected by the Object selector, is outlined in red. When you have selected the objects, release the mouse. The Insert Driven Actions dialog box appears with data collected about the objects you selected.

Editing data-driven commands

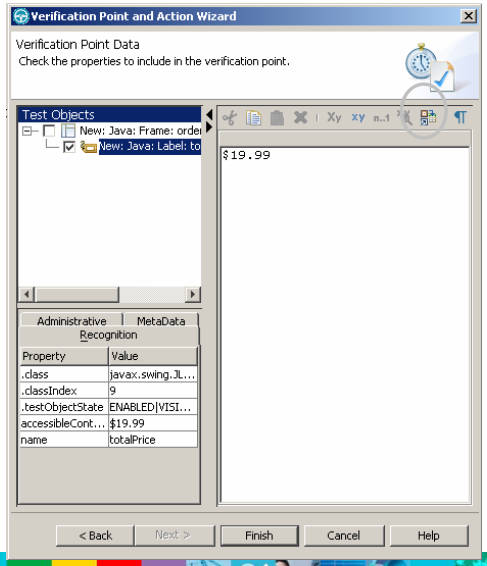


You can add descriptive headings to the data in the datapool. Meaningful headings make it easier to later add data to the datapool.

Once you create the command, you can add descriptive headings to the data in the datapool. Meaningful headings make it easier to later add data to the datapool.

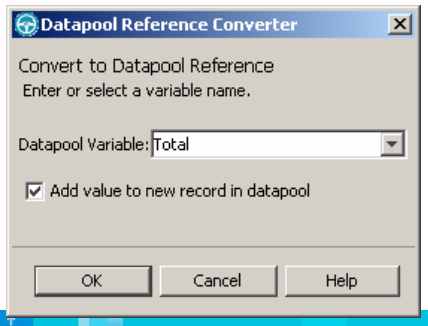
Inserting a verification point that references the datapool

When you insert a verification point in a data-driven script, you can create a datapool reference instead of a literal value.



Click the Convert Value to Datapool Reference button.

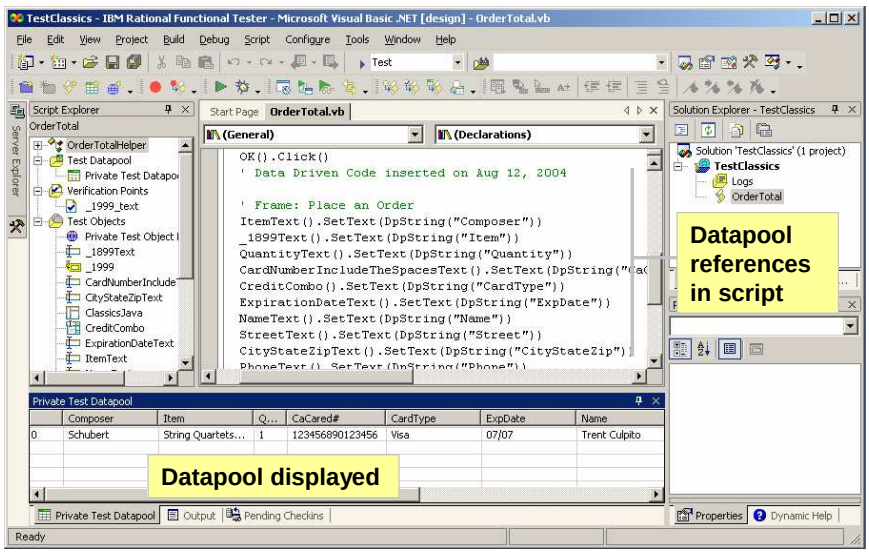
Type a name for the variable. Check the Add value to new record in datapool checkbox to add the variable to your datapool.



When you insert a verification point in a data-driven script, you can create a datapool reference instead of a literal value. Remember, verification points are the components that can be used to check the differences in the actual values against expected values.

Datapool display

When you stop recording, the script and datapool information display.



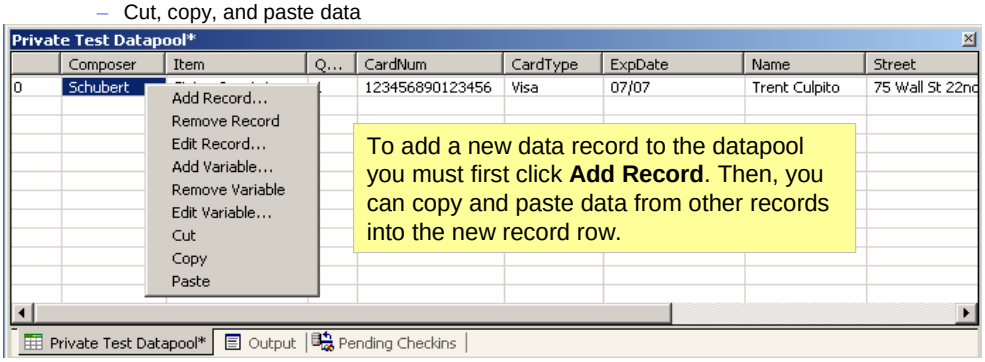
When you stop recording, the script and datapool information is shown here.

Editing a datapool

Once you have created a datapool you can edit the data you input or add additional data.

Double-click the datapool title bar to begin editing the datapool. From the shortcut menu you can:

- Edit, add, and remove records
- Edit, add, and remove variables
- Cut, copy, and paste data

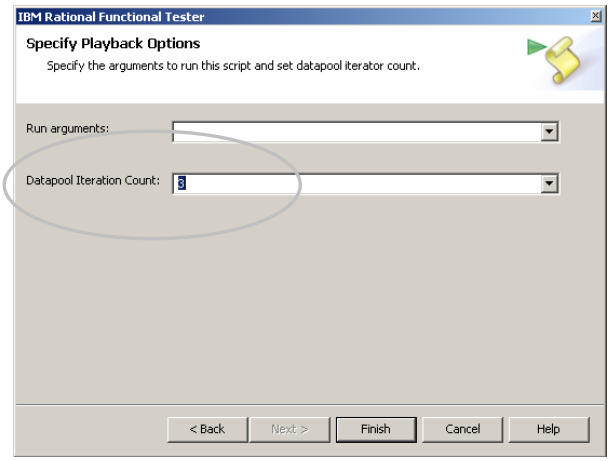


Once you have created a datapool you can edit the data you input or add additional data. Remember, to add a new data record, you must first click Add Record. You will then be able to copy and paste data from other records into the new record row.

Data-driving a test script

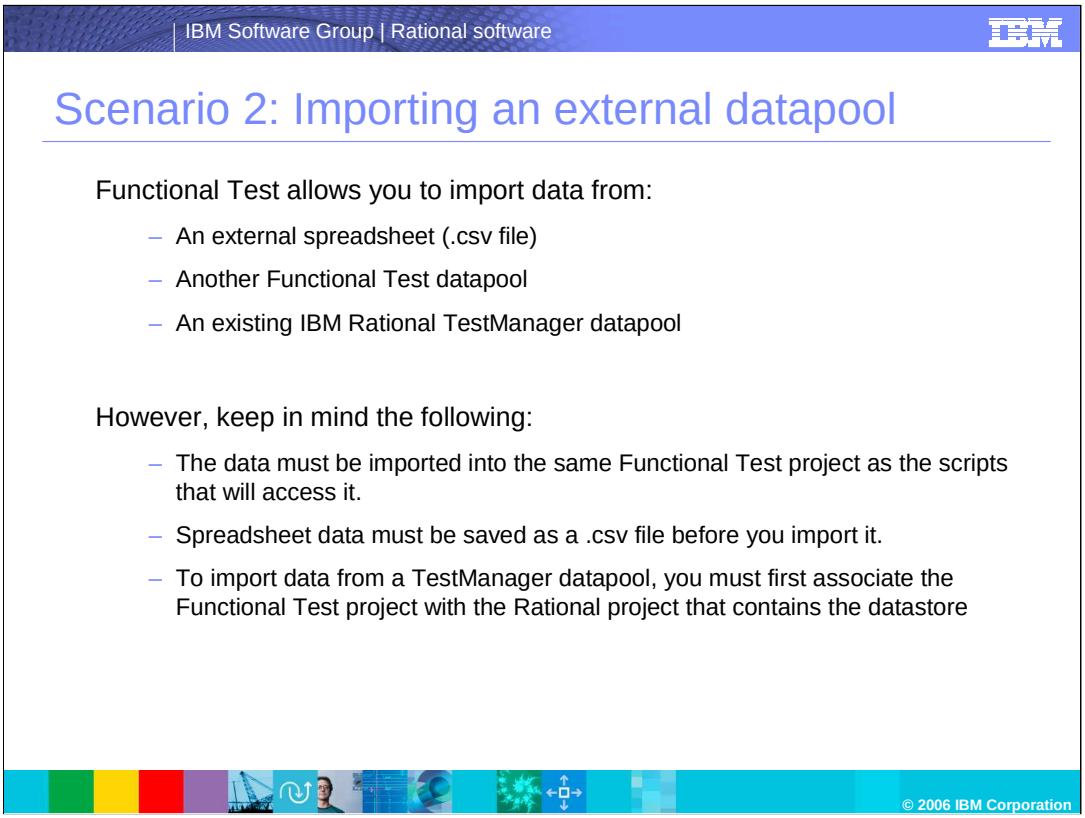
When you play back the script, you set the datapool iteration count. Each time the script runs, it pulls a different record from the datapool.

Or, you can add a parameter to CallScript to iterate through the entire datapool:



CallScript ("myScript", DEFAULT_ARGS, DP_ALL)

You can add a parameter to CallScript to iterate through a datapool. When you play back the script, you set the datapool iteration count. Each time the script runs, it pulls a different record from the datapool.



Scenario 2: Importing an external datapool

Functional Test allows you to import data from:

An external spreadsheet (.csv file)

Another Functional Test datapool or

An existing IBM Rational TestManager datapool

However, keep in mind that:

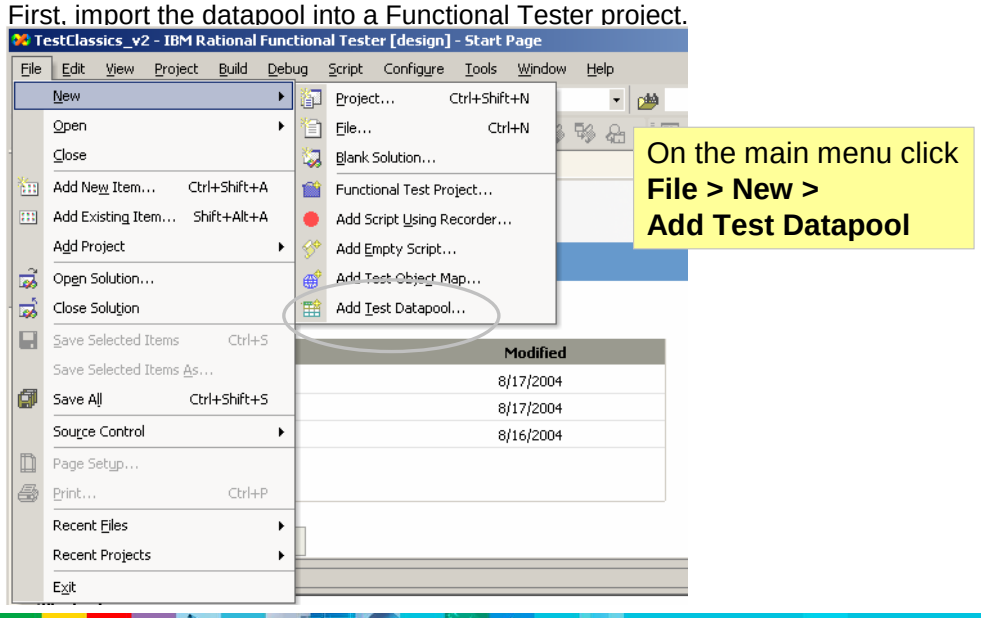
The data must be imported into the same Functional Test project as the scripts that will access it.

Spreadsheet data must be saved as a .csv file before you import it.

To import data from a TestManager datapool, you must first associate the Functional Test project with the Rational project that contains the datastore.

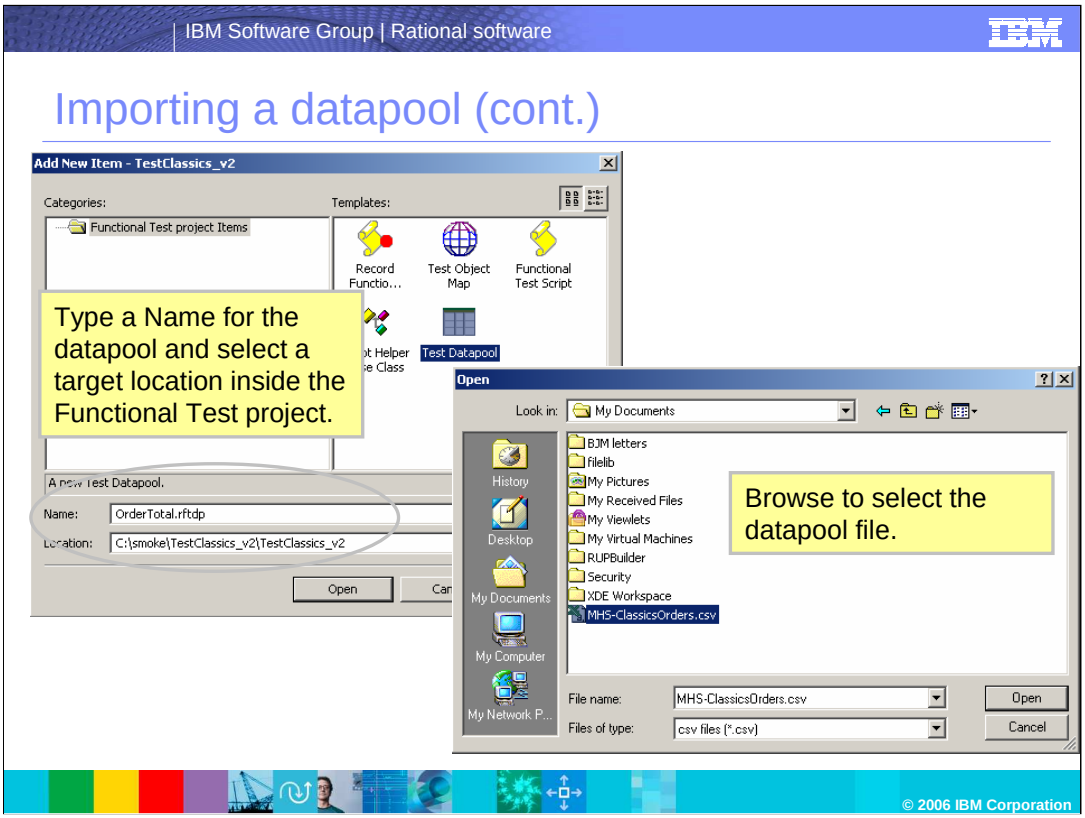
Importing an external datapool

First, import the datapool into a Functional Tester project.



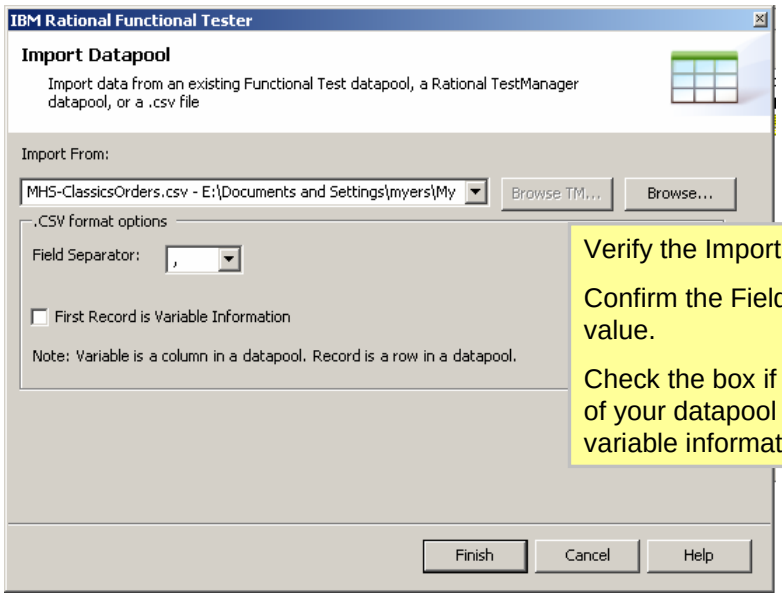
In this scenario, you import an external datapool (.csv file) into a Functional Test project and then associate it with a test script.

Importing a datapool (cont.)



This window shows how to import from the UI. Follow the directions above to import a datapool.

Importing a datapool (cont.)

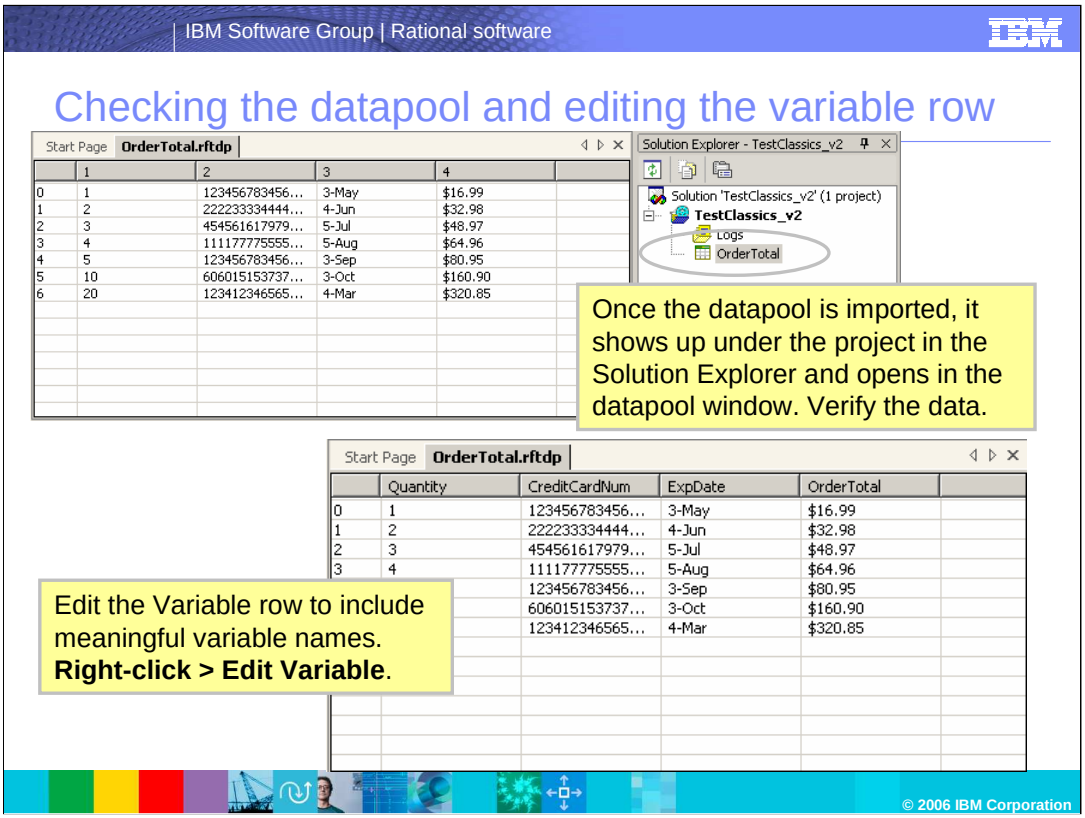


Verify the Import From location.

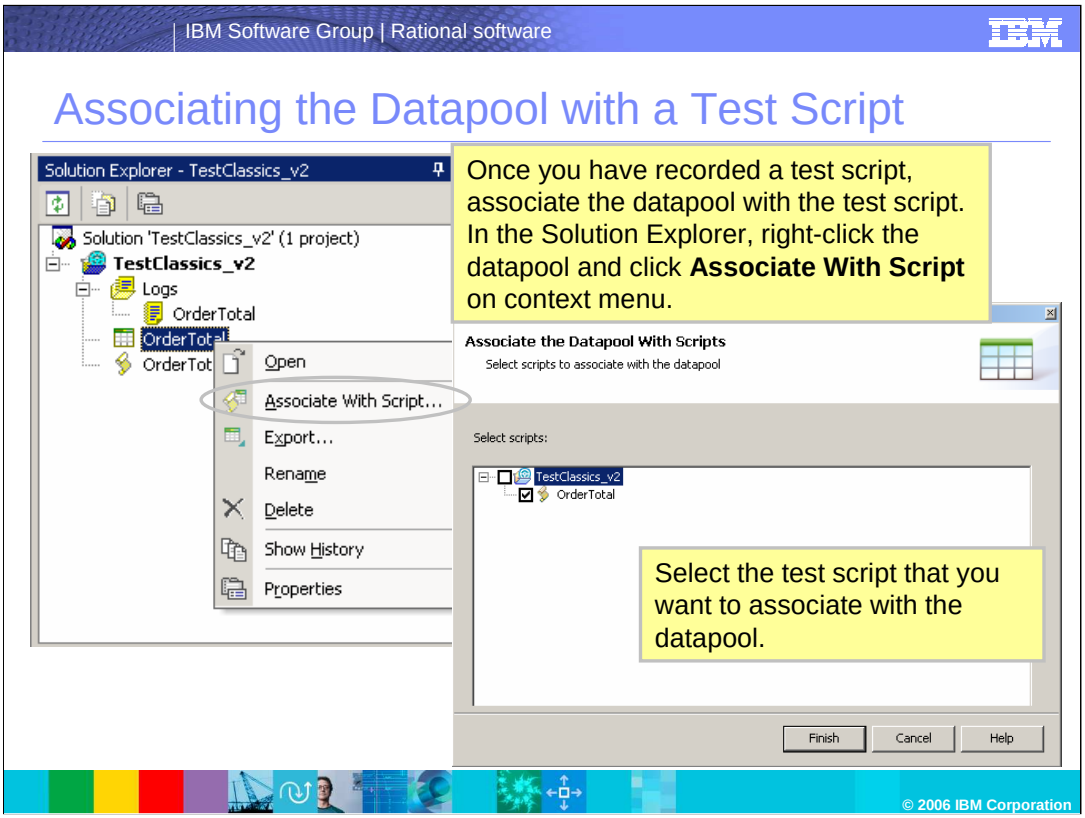
Confirm the Field Separator value.

Check the box if the first record of your datapool contains variable information.

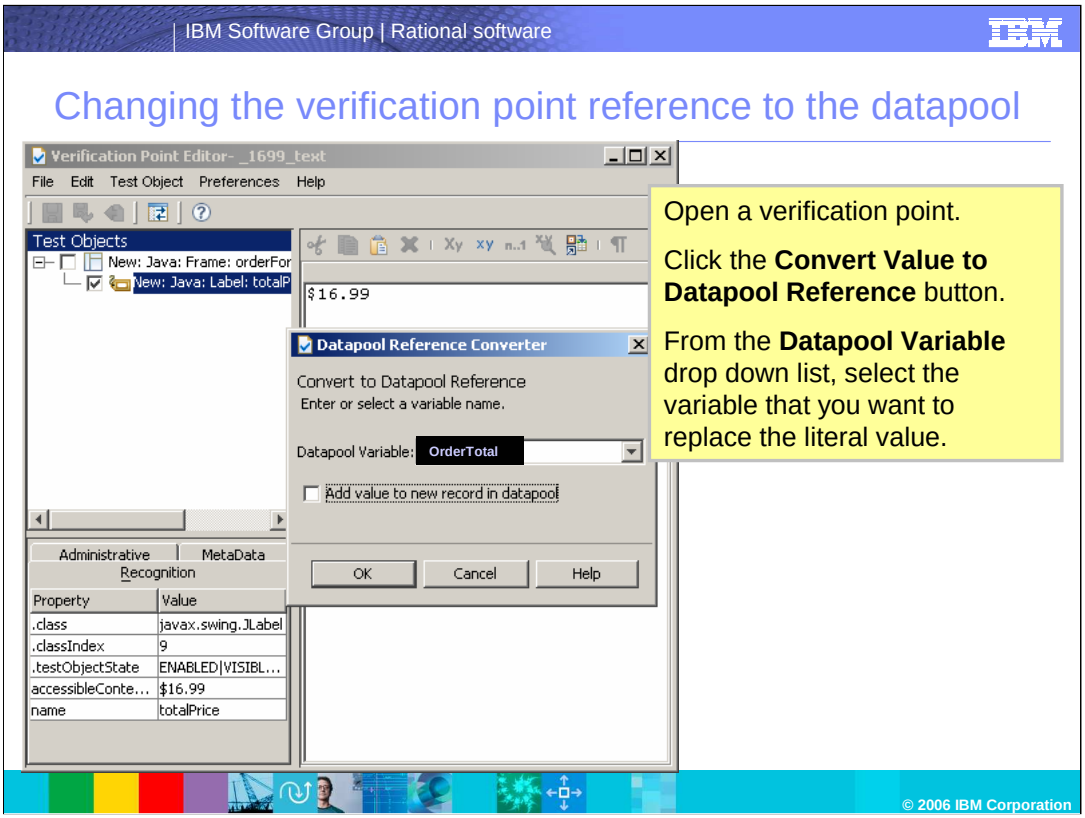
Verify the Import From location and other information on this window.



Once the datapool is imported, it shows up under the project in the Solution Explorer and opens in the datapool window. Verify the data shown. You can also edit the variable row to include meaningful variable names.

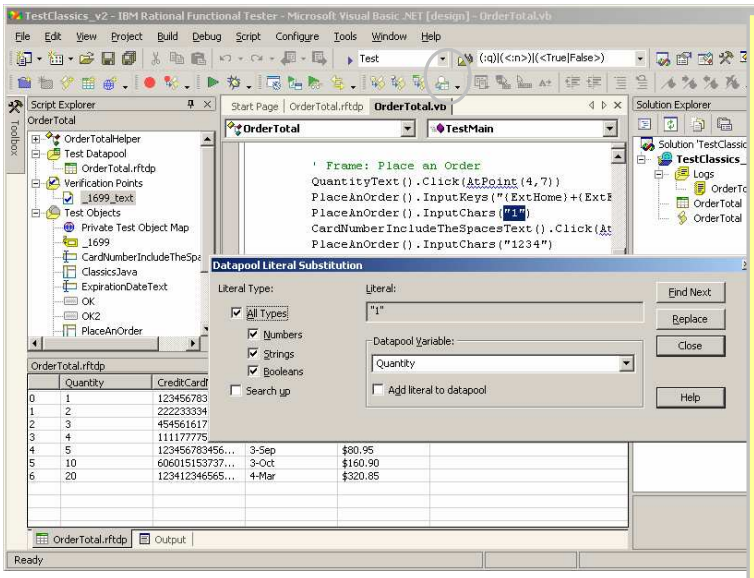


Once you have recorded a test script, associate the datapool with the test script. In the Solution Explorer, right-click the datapool and click **Associate With Script** on context menu.



You can also change the Verification Point Reference to the Datapool here.

Replacing literals in the test script with variables

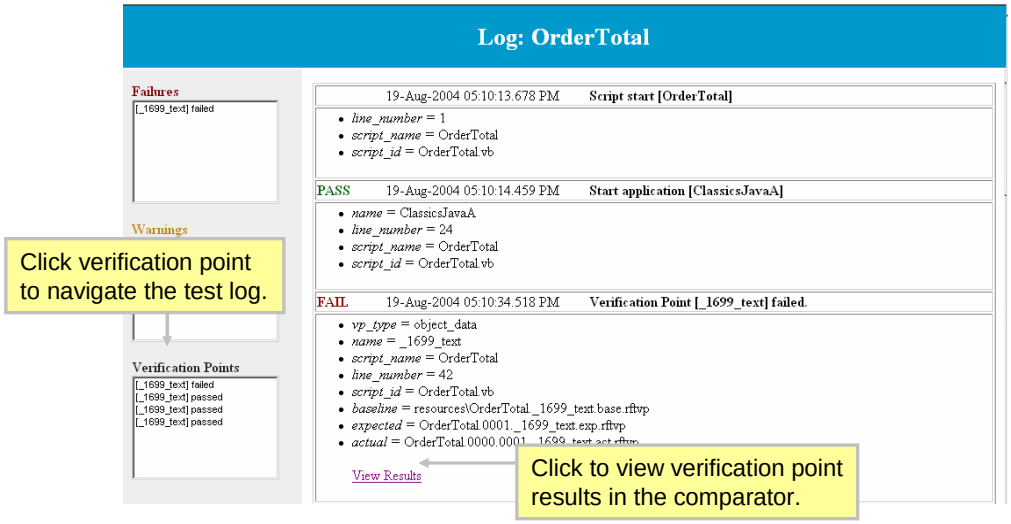


Click the **Find Literals and Replace with Datapool Reference** button to replace literal values in the script with variables.

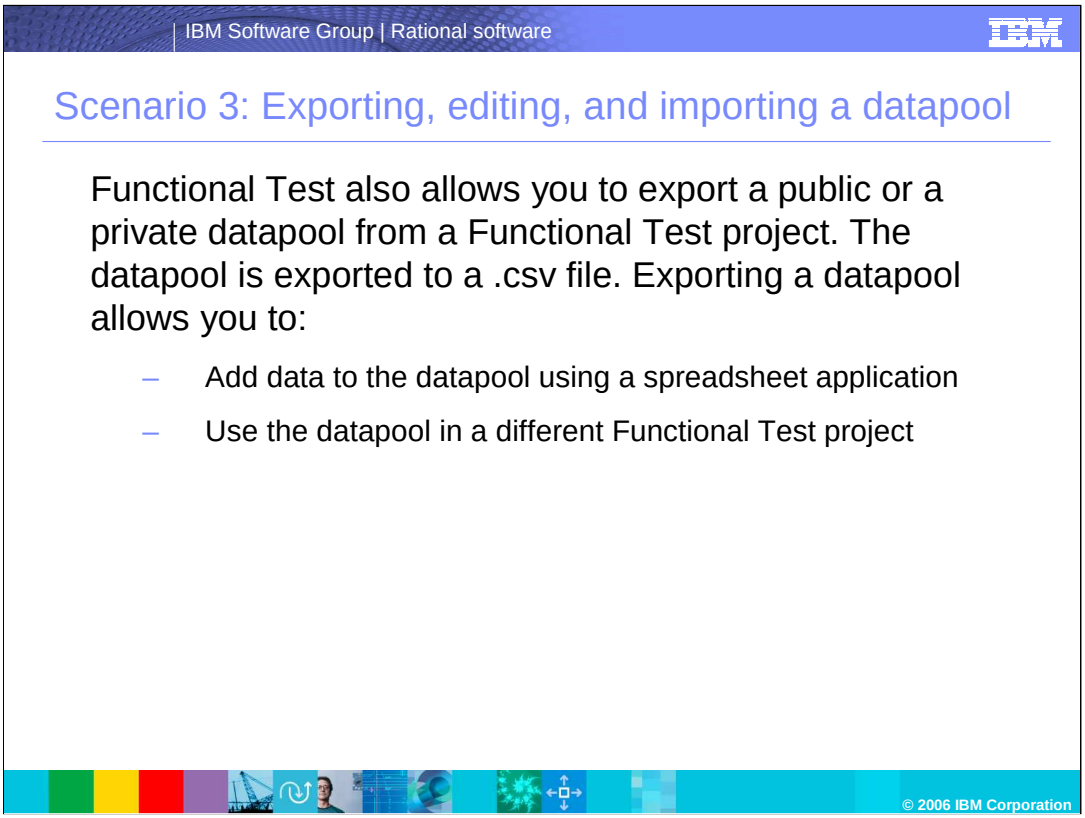
In the Datapool Literal Substitution box, click **Find Next** to move to the next literal. To replace a literal, select a variable to from the **Datapool Variable** drop down list.

You can replace literals in the Replacing Literals in the Test Script with Variables. This feature allows you to check verification points dynamically.

Playing back the test script to test the datapool



After you playback the script, the log will be displayed, showing the results in the comparator.

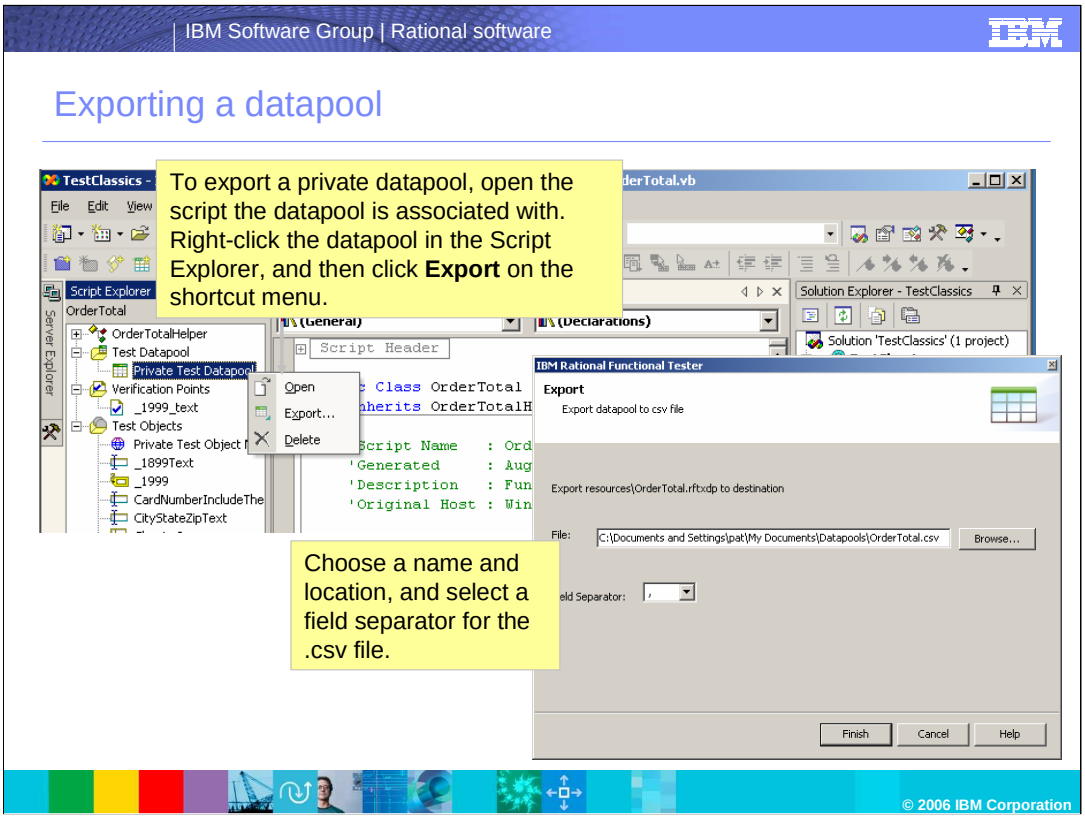


Scenario 3: Exporting, editing, and importing a datapool

Functional Test also allows you to export a public or a private datapool from a Functional Test project. The datapool is exported to a .csv file. Exporting a datapool allows you to:

Add data to the datapool using a spreadsheet application

Use the datapool in a different Functional Test project



This window explains how to export the datapool. Follow the steps above to complete the export. For more information on exporting, editing, and importing a datapool, review the tutorials from the dynamic help menu.

Trademarks, copyrights and disclaimers

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

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

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

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

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

Linux is a registered trademark of Linus Torvalds.

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

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

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

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.