DevTest Solutions

Getting Started

Version 8.0

ca technologies

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: DevTest Portal

This section contains the following topics:

Open the DevTest Portal

You open the DevTest Portal from a web browser.

Note: For information about the server components that must be running, see Start the DevTest Processes or Services in *Installing*.

Follow these steps:

1. Complete one of the following actions:
   - Enter **http://localhost:1507/devtest** in a web browser. If the registry is on a remote computer, replace **localhost** with the name or IP address of the computer.
   - Select View, DevTest Portal from DevTest Workstation.
2. Enter your user name and password.
3. Click Log in.

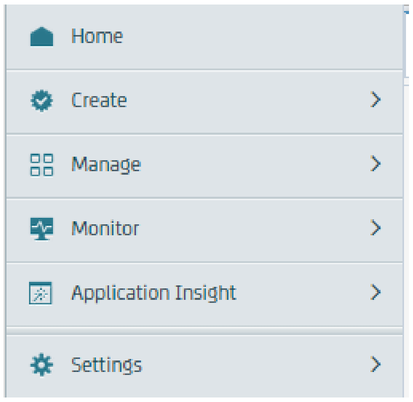
Navigating the DevTest Portal

Browse to the DevTest Portal. Substitute the host name of the DevTest Server on which the following processes or services are running: Registry, Portal, Broker, and VSE.

```
http://hostname:1507/devtest
```

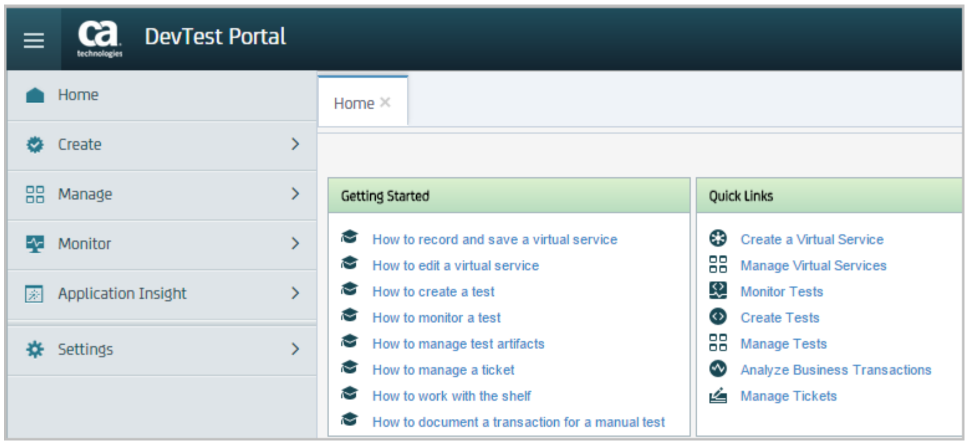
Log in to the DevTest Portal with your user ID and password. If you do not have one, log in as a standard user. See your administrator for details.

Use the DevTest Portal homepage to access pages for major activities and the corresponding documentation. You can expand any section of the navigation pane to display links to relevant tasks. Click a link to display the UI for a selected task, typically a tab on the Portal. Some links redirect you to the DevTest Console.



For links you will use most often, you can click the corresponding Quick Links option. For related documentation, click the corresponding Getting Started link. The Getting Started and Quick Links panes appear in the Home tab.

Note: The first time that you click a Getting Started link, a login dialog opens. Your login to view the documentation persists throughout your session on the DevTest Portal.

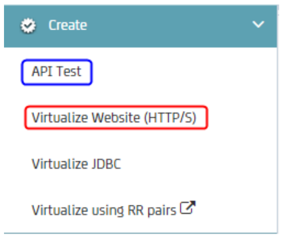


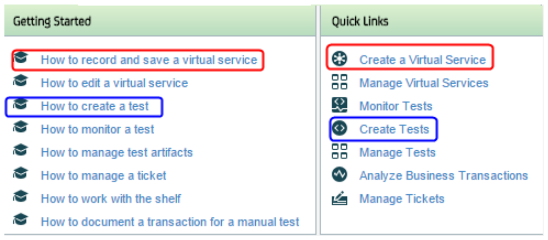
Details on the mapping of the Quick Links and Getting Started options to the navigation menu options follows.

Create

Perform common create actions.

Notice the icon following the Virtualize using RR pairs. This icon indicates that clicking the link redirects you to another UI.




Getting Started
How to record and save a virtual service
How to edit a virtual service
How to create a test
How to monitor a test
How to manage test artifacts
How to manage a ticket
How to work with the shelf
How to document a transaction for a manual test
Quick Links
Create a Virtual Service
Manage Virtual Services
Monitor Tests
Create Tests
Manage Tests
Analyze Business Transactions
Manage Tickets

API Test

Opens: API Test tab

Getting Started: How to create a test in *Using CA Application Test*

Quick Links: Create Tests

Virtualize Website (HTTP/S)

Opens: Virtualize Website (HTTP/S) tab

Getting Started: How to record and save a virtual service in *Using CA Service Virtualization*

Quick Links: Create a Virtual Service

Virtualize JDBC

Opens: Virtualize JDBC tab

Topic: JDBC in Using CA Service Virtualization

Virtualize RR Pairs

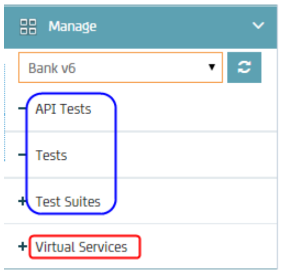
Redirect: DevTest Console

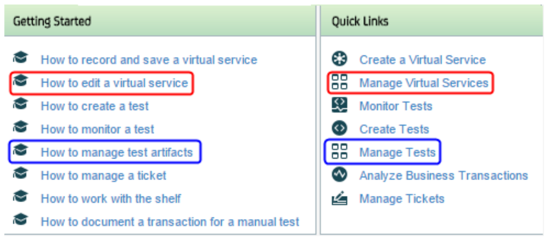
Opens: Create Virtual Service on VSEasy tab

Topic: Create and Deploy a Virtual Service with VSEasy in *Using CA Service Virtualization*

Manage

Manage test artifacts.




Getting Started
How to record and save a virtual service
How to edit a virtual service
How to create a test
How to monitor a test
How to manage test artifacts
How to manage a ticket
How to work with the shelf
How to document a transaction for a manual test
Quick Links
Create a Virtual Service
Manage Virtual Services
Monitor Tests
Create Tests
Manage Tests
Analyze Business Transactions
Manage Tickets

API Test

Manage tests that were created with the DevTest Portal.

Getting Started: How to manage test artifacts, Manage API Tests in *Using CA Application Test*

Quick Links: Manage Tests

Test: Current Status (Today)

Tests

View information about test cases and run test cases that were created with DevTest Workstation.

Getting Started: How to manage test artifacts, Manage Tests in *Using CA Application Test*

Quick Links: Manage Tests

Test: Current Status (Today)

Test Suites

View information about test suites and run test suites that were created with DevTest Workstation.

Getting Started: How to manage test artifacts, Manage Test Suites in *Using CA Application Test*

Quick Links: Manage Tests

Test: Current Status (Today)

Virtual Services

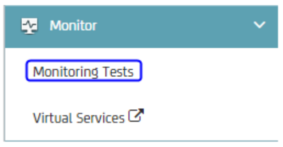
Opens tabs: http-rest, WebServicesModel, http

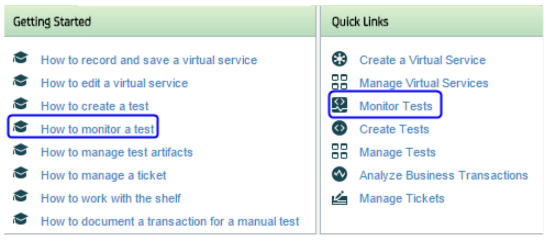
Getting Started: How to edit a virtual service in *Using CA Service Virtualization*

Quick Links: Manage Virtual Services

Monitor

Monitor tests or virtual services. When there are no tests or suites that meet selection criteria, you can click "Create a Test" or "Execute a Test"




Getting Started
How to record and save a virtual service
How to edit a virtual service
How to create a test
How to monitor a test
How to manage test artifacts
How to manage a ticket
How to work with the shelf
How to document a transaction for a manual test
Quick Links
Create a Virtual Service
Manage Virtual Services
Monitor Tests
Create Tests
Manage Tests
Analyze Business Transactions
Manage Tickets

Monitoring Tests

The Monitor Test window lets you see API tests, test cases, and test suites that have been run on CA Application Test.

Click the filter button to toggle the search options.

Opens: Monitoring Tests tab

Getting Started: How to monitor a test in *Using CA Application Test*

Quick Links: Monitor Tests

Virtual Services

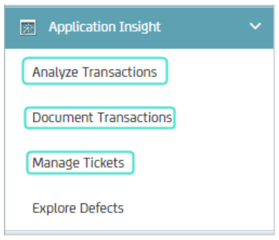
Redirect: DevTest Console

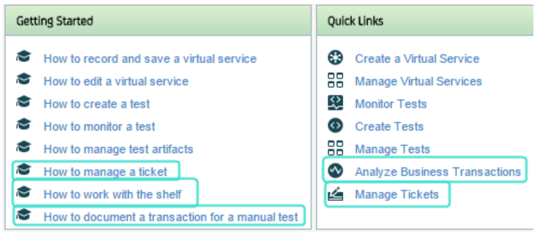
Opens: Server Console, DevTest Network, where you can click VSE

Topic: Create and Deploy a Virtual Service with VSEasy in *Using CA Service Virtualization*

Application Insight

Start common tasks with CA Continuous Application Insight.




Getting Started
How to record and save a virtual service
How to edit a virtual service
How to create a test
How to monitor a test
How to manage test artifacts
How to manage a ticket
How to work with the shelf
How to document a transaction for a manual test
Quick Links
Create a Virtual Service
Manage Virtual Services
Monitor Tests
Create Tests
Manage Tests
Analyze Business Transactions
Manage Tickets

Analyze Transactions

Opens: Analyze Transactions tab

Getting Started: How to work with the shelf in *Using CA Continuous Application Insight*

Quick Links: Analyze Business Transactions

Document Transactions

Opens: Document Transactions tab

Getting Started: How to document transactions for a manual test in *Using CA Continuous Application Insight*

Manage Tickets

Opens: Manage Tickets tab

Getting Started: How to manage a ticket in *Using CA Continuous Application Insight*

Quick Links: Manage Tickets

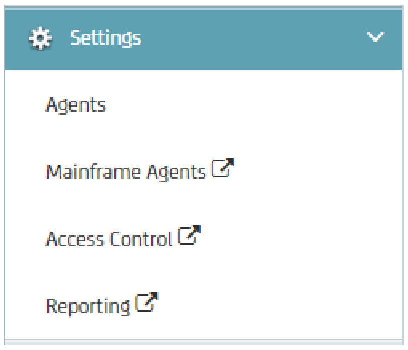
Explore Defects

Opens: Explore Defects tab

Topic: Working with Defects in *Using CA Continuous Application Insight*

Settings

The Settings tab has no Quick Links entries or Getting Started links.



Agents

Opens: Agents tab

Topics: See *Agents*.

Mainframe Agents

Redirects: DevTest Console, Server Console, DevTest Network

Topic: Mainframe Bridge in *Agents.*

Access Control

Redirects: DevTest Console, Server Console. Expand the Administration tab to display Security options.

Topic: Access control (ACL) in *Administering*.

Reporting

Redirects: DevTest Console, Reporting

Topics: Reports in *Using CA Application Test*.

DevTest Portal Home Page

The home page of DevTest Portal contains a collection of portlets that let you monitor the activity of your application. Each portlet displays information of a DevTest component that can be manually refreshed.

The home page contains the following portlets:

Getting Started

Displays links to documentation topics that describe key functions of the portal.

Quick Links

Displays links to DevTest Solutions operations.

Help

Displays a link to the DevTest documentation, to ask a question to the DevTest Community, and to report an issue.

Virtual Services: Current Status

Displays the current, total count of transactions, the number of virtual services that are deployed, and the number of virtual services that are recorded.

Test: Current Status

Displays the available test results. Clicking Show More... navigates you to the Monitoring Test window.

Agents: Current Status

Displays the status for an agent. Selecting an agent name or Show More... navigates you to the agent management operations in the Agents window. To view the current agent information, manually refresh the portlet.

New Ticket Alerts

Displays a list of new tickets and a count of new, identified, or closed tickets. The name, date, and time of the ticket displays in the list. Selecting the ticket, New, Identified, or Closed navigates you to Manage Tickets window. The Manage Tickets window lets you view tickets, update existing tickets, view a list of tickets, and search for tickets in the CA Continuous Application Insight database.

Path Alerts

Displays a list of paths with exception flags. Selecting a path navigates you to the Explore Defects window.

Points of Interest

Displays a list of all the points of interest that were pinned in Explore Defects. Selecting the name in the list of Points of Interest navigates you to the Explore Defects window.

Chapter 2: CA Application Test Tutorials

This section contains a series of tutorials that illustrate various aspects of CA Application Test. The tutorials are sequential. Complete them in the order presented.

The first few tutorials walk you through using DevTest Workstation to build simple test cases. You become familiar with basic concepts such as projects, properties, data sets, filters, and assertions.

The subsequent tutorials help you acquire deeper knowledge about how to set up test steps to interact with and test several common technologies. These technologies include Java objects, web pages, Enterprise JavaBeans (EJBs), web services, and databases. You also learn how to stage a quick test.

To perform the tutorials, you must have DevTest Workstation installed and you must have access to a registry.

Some tutorials use the Demo Server as the system under test. For information about installing the Demo Server, see *Installing.*

This section contains the following topics:

Tutorial 1 - Projects, Test Cases, and Properties

Tutorial Tasks

In this tutorial, you:

- Create a project
- Create a test case
- Add properties
- Add simple test steps
- Use the Interactive Test Run Utility

Prerequisites

- DevTest Workstation is installed and DevTest license credentials are entered.
- You have reviewed the Glossary of Terms (see page 139).

Step 1 - Start DevTest Workstation

Follow these steps:

1. Start the registry.
   - If your computer has DevTest Server installed:
   a. Start the registry by clicking Start Menu, All Programs, DevTest Solutions, EnterpriseDashboard. Wait until the "Enterprise Dashboard started" message appears.
   b. Start the registry by clicking Start Menu, All Programs, DevTest Solutions, Registry.
   - If your computer has DevTest Workstation installed, use a registry that is running on another computer.
2. Click Start, All Programs, DevTest Solutions, Workstation.
3. When the Set DevTest Registry dialog opens, select a registry and click OK.
4. The Login dialog opens. Enter a valid username and password and click Login.

Step 2 - Create a Project

The project that you create holds all the test case example files that are required for the tutorials.

Follow these steps:

1. From the DevTest Workstation main menu, select File, New, Project.

   The Create New Project dialog opens.

2. In the Project Name field, remove the default value and type **My Tutorials**.
3. Click Create.

   The My Tutorials project is created.

Step 3 - Create a Test Case

A test case is a specification of how to test a business component in the system under test.

Follow these steps:

1. In the Project panel, right-click the Tests folder and select Create New Test Case.
2. Set the file name to **tutorial1**.
3. Click Save.

   DevTest Workstation opens a new tab labeled tutorial1. The green arrow in the model editor represents the start of the test case.

Step 4 - Add a Property to the Project Configuration

In this step, you set a global property in the project configuration. You access this property later in the tutorial.

The default configuration has the name **project.config**, and is created automatically for a new project. The project.config file is located in the Configs folder in the Project panel. The file extension is not shown. You can add the properties to the project.config file and, if necessary can also create a configuration file.

Follow these steps:

1. In the Project panel, double-click **project** in the My Tutorials > Configs folder.

   The properties editor opens.

2. To add a row, click Add at the bottom of the properties editor.
3. In the Key field, type **config_prop**.
4. In the Value field, type **42**.
5. From the main toolbar, click Save.

Step 5 - Add a Test Step

A test case includes one or more test steps. In this procedure, you add an Output Log Message test step to write text to the log file.

Follow these steps:

1. Click the tutorial1 tab.
2. Click Add Step, select Utilities, and select Output Log Message.

   A step named Output Log Message is added to the model editor.

3. Right-click the Output Log Message step and select Rename.
4. Change the name to My Output Log Message.
5. Make sure that My Output Log Message is still selected. In the right pane, click the arrow next to Output Log Message.

   The Output Log Message tray opens.

Step 6 - Add a Log Message

With the log editor open, you add a log message that includes various properties.

The properties in the log message originate from several sources:

- The LISA_HOME property is automatically set.
- The java.version property is a system property.
- You added the config_prop property to the project configuration in Step 4.
- You create a property with the name **MyOutputLogMessage_step_prop** in the log message itself.

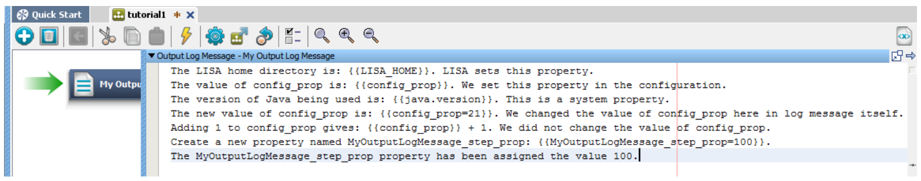
The syntax for a property is **{{property_name}}**.

Follow these steps:

1. In the log editor, delete the placeholder text.
2. Copy and paste the following text into the log editor:

```
The LISA home directory is: {{LISA_HOME}}. LISA sets this property.
The value of config_prop is: {{config_prop}}. We set this property
in the configuration.
The version of Java being used is: {{java.version}}. This is a
system property.
The new value of config_prop is: {{config_prop=21}}. We changed the
value of config_prop here in log message itself.
Adding 1 to config_prop gives: {{config_prop}} + 1. We did not
change the value of config_prop.
Create a new property named MyOutputLogMessage_step_prop:
{{MyOutputLogMessage_step_prop=100}}.
The MyOutputLogMessage_step_prop property has been assigned the
value 100.
```

The log editor looks like the following graphic.



```
The LISA home directory is: {{LISA_HOME}}. LISA sets this property.
The value of config_prop is: {{config_prop}}. We set this property in the configuration.
The version of Java being used is: {{java.version}}. This is a system property.
The new value of config_prop is: {{config_prop=21}}. We changed the value of config_prop here in log message itself.
Adding 1 to config_prop gives: {{config_prop}} + 1. We did not change the value of config_prop.
Create a new property named MyOutputLogMessage_step_prop: {{MyOutputLogMessage_step_prop=100}}.
The MyOutputLogMessage_step_prop property has been assigned the value 100.
```

Step 7 - Add a Second Log Message

The second test step in the test case writes a different message to the log file.

Follow these steps:

1. Close the first log message step by clicking the arrow at the upper left corner of the window.
2. Click Add Step, select Utilities, and select Output Log Message.

   A step with the name **Output Log Message** is added to the model editor and the Output Log Message tray opens.
3. In the log editor, delete the placeholder text.
4. Copy and paste the following text into the log editor:
   ```
   The current value of config_prop is: {{config_prop}}.
   The current value of MyOutputLogMessage_step_prop:
   {{MyOutputLogMessage_step_prop}}.
   ```
   Note: The log message does not change the values of **config_prop** or **MyOutputLogMessage_step_prop**.
5. Close the second log message step by clicking the arrow at the upper left corner of the window.
6. From the main application toolbar, click Save Save, or select File, Save, tutorial1.

Step 8 - Run the My Output Log Message Step

The Interactive Test Run (ITR) utility enables you to walk through and verify a test case.

Follow these steps:

1. From the toolbar, click the Start ITR.

   The ITR opens. The ITR contains an Execution History pane on the left and a set of tabs on the right.
2. In the Execution History pane, click Execute Next Step.

   The My Output Log Message step is run. The Response tab displays the response from the My Output Log Message step. The actual values replace the properties.

Step 9 - Observe Property Values

The ITR also lets you observe how the properties are created and modified.

Follow these steps:

1. Click the Properties tab in the ITR.

   The Properties tab displays the value of each property before and after the execution of the My Output LogMessage step. A value that the step created is highlighted in green. A value that was modified in the step is highlighted in yellow. Notice that the value of config_prop was changed from 42 to 21.

2. Compare these values with the response in Step 8.

Step 10 - Run the Output Log Message Step

In this procedure, you use the ITR to run the second step in the test case.

Follow these steps:

1. In the ITR, click Execute Next Step to run the Output Log Message step.
2. To view the response, click the Response tab. Although you set config_prop to 42 in the project.config file, you changed the value to 21 in the My Output Log Message step, and the value did not change in the Output Log Message step. The value of the **MyOutputLogMessage_step_prop** property also carried over from the My Output Log Message step to the Output Log Message step.
3. To view the current and previous property values, click the Properties tab.
4. When you are done, close the tutorial1 and project tabs.

Review

In this tutorial, you took a first look at properties. You saw that properties are denoted by using a special syntax, {{property_name}}. You can set properties by using a variation of this syntax; {{property_name=value}}. After you set a property, use or modify it in subsequent steps in a test case.

In this tutorial, you:

- Learned how to create and save a test case
- Learned how to add a simple test step (Output Log Message)
- Used a configuration to store properties
- Saw a brief glimpse of the Interactive Test Run utility

Tutorial 2 - Data Sets

In this tutorial, you learn how to create and use a simple data set. You also learn how to provide the data in a data set to a test case.

Tutorial Tasks

In this tutorial, you will:

- Create a simple data set
- Use the data set in various ways
- Iterate through a series of test steps using a data set

Prerequisites

- You have completed Tutorial 1 - Properties.
- DevTest Workstation is open.

Step 1 - Create a Data Set

In this tutorial, you use a comma-delimited text file as the data set. This option is only one of several options available to create a data set. After you create the text file, you import it into the My Tutorials project.

Follow these steps:

1. In a text editor such as Notepad, create a text file.
2. Copy and paste the following properties and values into the text editor. Do not use spaces in the text file.

   ```
   month,day,year
   3,2,1956
   4,7,2007
   1,3,2010
   5,8,{{yearglobal}}
   8,10,2004
   12,11,{{yearglobal}}
   10,12,2007
   3,5,2011
   ```

   The first row specifies the names of the properties to which this data is assigned (month, day, year). The remaining rows specify the data that is read and used in the test case. Two of the rows include a property with the name **yearglobal**.
3. Save the file as dates.txt.
4. In the Project panel, right-click the Data folder in the My Tutorials project and select Import Files.
5. Navigate to the folder where you saved the dates.txt file and select the file name.
6. Click Open. The dates.txt file now appears in the Data folder.

Step 2 - Create a Test Case

You add a test case to the My Tutorials project.

Follow these steps:

1. In the Project panel, right-click the Tests folder and select Create New Test Case.
2. Make the file name **tutorial2**.
3. Click Save.

Step 3 - Add a Property to the Project Configuration

The dates.txt file includes a property with the name **yearglobal**. In this procedure, you add the yearglobal property to the project configuration.

Follow these steps:

1. In the Project panel, double-click project.config.
2. Click Add to add a row.
3. In the Key field, enter yearglobal.
4. In the Value field, enter 1999.
5. Click Save.

Step 4 - Add a Test Step for Output Log Message

To write text out to the log, use a test step, the Output Log Message step.

Follow these steps:

1. Click the tutorial2 tab.
2. Click Add.

   The Add step menu is displayed.
3. Select Utilities and select Output Log Message.

   A step with the name **Output Log Message** is added to the model editor.
4. Right-click Output Log Message and select Rename. Change the name to DSstep1.
5. In the right pane, click the arrow next to Output Log Message.

   The Output Log Message tray opens.
6. Delete the placeholder text.
7. Enter the following log message:
   ```
   Date is: {{month}}/{{day}}/{{year}}
   ```
   Note: The curly brackets are important. The test case runs correctly only if they are included.
8. To close the Output Log Message tray, click anywhere in the model editor.
9. Click Save.

Step 5 - Create Another Output Log Message Step

Create another test step similar to the DSstep1 test step.

Follow these steps:

1. Click Add.

   The Add step menu is displayed.
2. Select Utilities and select Output Log Message.

   A step with the name **Output Log Message** is added to the model editor.
3. Right-click Output Log Message and select Rename. Change the name to DSstep2.
4. In the right pane, click the arrow next to Output Log Message.

   The Output Log Message tray opens.
5. Delete the placeholder text.
6. Enter the following log message:
   ```
   Date is: {{month}}/{{day}}/{{year}}
   ```
7. To close the Output Log Message tray, click anywhere in the model editor.
8. Click Save Save.

Step 6 - Execute the Test

To execute the test and see what happens, use the Interactive Test Run (ITR).

Follow these steps:

1. From the toolbar, click Start ITR.

   The ITR opens.
2. In the Execution History pane, click Automatically execute test .
3. When the test is complete, click OK.
4. In the Execution History pane, click DSstep1 and DSstep2.

   Notice that the month, day, and year properties have not been replaced with actual values. This result is expected, because you have not added the data set to the test case.

Step 7 - Add the Data Set

You now add the dates.txt data set to the DSstep1 test step.

Follow these steps:

1. In the model editor, select DSstep1.
2. In the right pane, double-click the Data Sets step tab.
3. Click Add below the Data Sets element.
4. From the Common DataSets list, select Read Rows from a Delimited Data File.

   The data set is added to the test step.

   The data set editor opens in the right pane.
5. In the data set editor, set the name to DatesDS.
6. Click File Location, then navigate to and select the dates.txt file in the LISA_HOME\Projects\My Tutorials\Data directory.
7. Click the Test and Keep button.

   If the test is successful, the Data Set Editor window returns a "Test successful" message.
8. Click OK.
9. From the toolbar, click Start ITR, then select Start new ITR.
10. In the Execution History pane, click Automatically execute test .
11. When the test is complete, click OK.
12. In the Execution History pane, click DSstep1 and DSstep2.

    The first row of data in the data set is displayed in the Response tab. Both step responses display the same date because we read only from the data set in DSstep1.

Step 8 - Change the Data Set Behavior

You now modify the data set so that it loops through the test step until all the rows in the data set are read.

Follow these steps:

1. In the model editor, select the DSstep1 test step.
2. In the step elements panel of DSstep1, click the arrow next to DatesDS under the Data Sets element.

   The data set editor opens.
3. In the At end of data field, select the Execute option.
4. Click the drop-down arrow on the Execute field and select End the Test from the list of choices that appear.

   This setting causes the test to end when all the data rows have been read.
5. Click the Test and Keep button.
6. Click OK to close the test successful message.
7. In the model editor, select the DSstep2 test step.
8. In the Step Information tab, set the Next drop-down list to DSstep1.

   This setting causes the two test steps to loop. The arrows in the model editor show the order of execution: DSstep1, followed by DSstep2, followed by DatesDS.
9. From the toolbar, click Start ITR, then select Start new ITR.
10. In the ITR, click Automatically execute test .

    The test case runs in a loop until there are no more data rows in the data set.
11. When the test is complete, click OK.
12. Click Save.

Tutorial 2 - Review

In this tutorial, you:

- Created a comma-delimited data set
- Used the data set for running a simple test case
- Learned how a test step accesses the data in the data set

Tutorial 3 - Filters and Assertions

In this tutorial, you modify the test case that was created in Tutorial 2 to include a filter and an assertion.

For an introduction to filters and assertions, see Filters and Assertions in *Using CA Application Test.*

Tutorial Tasks

In this tutorial, you:

- Save an existing test case with a new name
- Add an assertion to a test step
- Add a filter to a test step

Prerequisites

- You have completed Tutorial 2 - Data Sets.
- DevTest Workstation is open.

Step 1 - Create a Test Case from an Existing Test Case

In this step, you open tutorial2.tst and save it as tutorial3.tst.

Follow these steps:

1. Open the tutorial2.tst test case in the My Tutorials project.
2. From the menu bar, select File, Save As.
3. In the File name field, enter **tutorial3**.
4. Click Save.

   The tutorial3 test case is created and saved under the My Tutorials project.

Step 2 - Change Action of Test Step

Change the Next Steps action of both test steps so that DSstep1 is the next step. Only the first step reads from the data set.

Follow these steps:

1. In the model editor, select DSstep1.
2. In the Step Information tab, change the Next step to **DSstep1**.

   With this action, the output goes back to the same step DSStep1. For the time being, alert icons appear next to DSStep2.
3. In the model editor, double-click DSstep2 and change the Output Log Message as follows:
   ```
   Date contains 1999. It is: {{month}}/{{day}}/{{year}}.
   ```
   Note: The curly brackets are important. The test case runs correctly only if they are included.
4. Click Save Save.

Step 3 - Add an Assertion

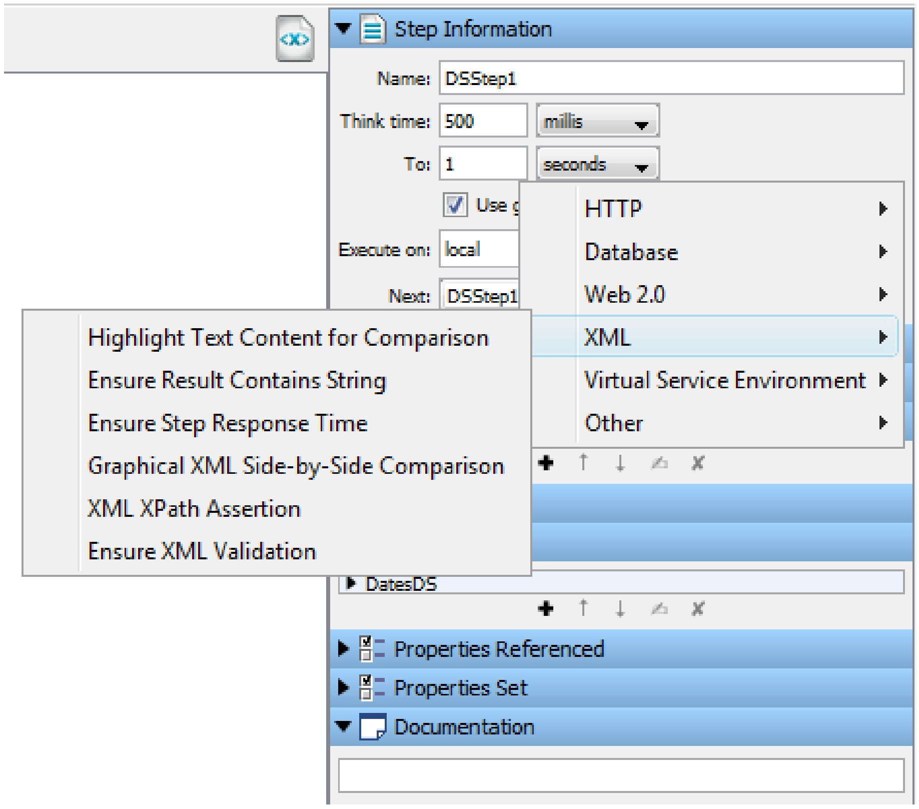
You can add various types of assertions to a test case. In this procedure, you add an XML assertion with the name **Ensure Result Contains String**.

The assertion logic is:

- If the response contains the string 1999, then the DSstep2 step is run next.
- If the response does not contain the string 1999, then the DSstep1 step is run next.

Follow these steps:

1. In the model editor, select DSstep1.
2. Open the Assertions tab.
3. Click ✚ Add.
4. From the XML submenu, select Ensure Result Contains String.



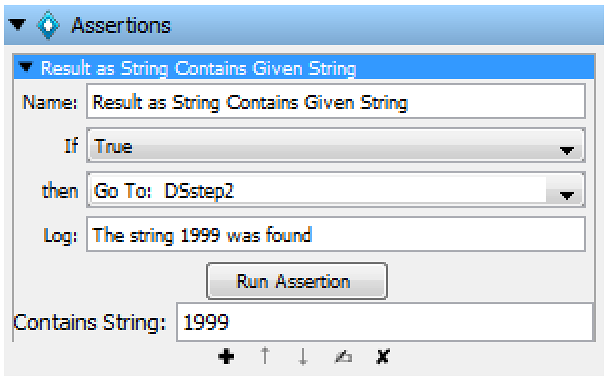
5. The new assertion that is applied to DSstep1 is added to the Assertions tab.

   The assertion editor opens.
6. In the assertion editor, do the following:
   a. In the If list, select True.

b. In the then list, select Go To: DSstep2.

c. In the Log field, enter The string 1999 was found.

d. In the Contains String field, enter 1999.



Name: Result as String Contains Given String

If True

then Go To: DSstep2

Log: The string 1999 was found

Contains String: 1999

7. Click Save Save.

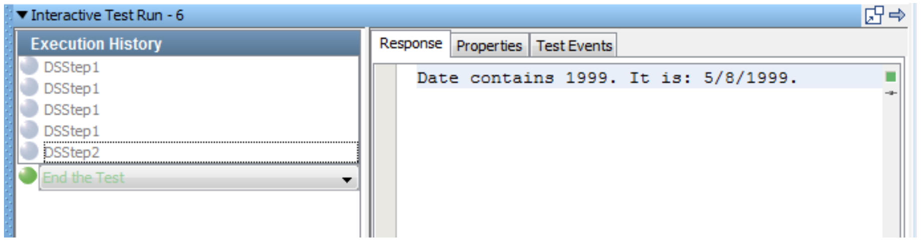
Step 4 - Test the Assertion

To determine whether the assertion works as expected, use the Interactive Test Run (ITR) utility.

Follow these steps:

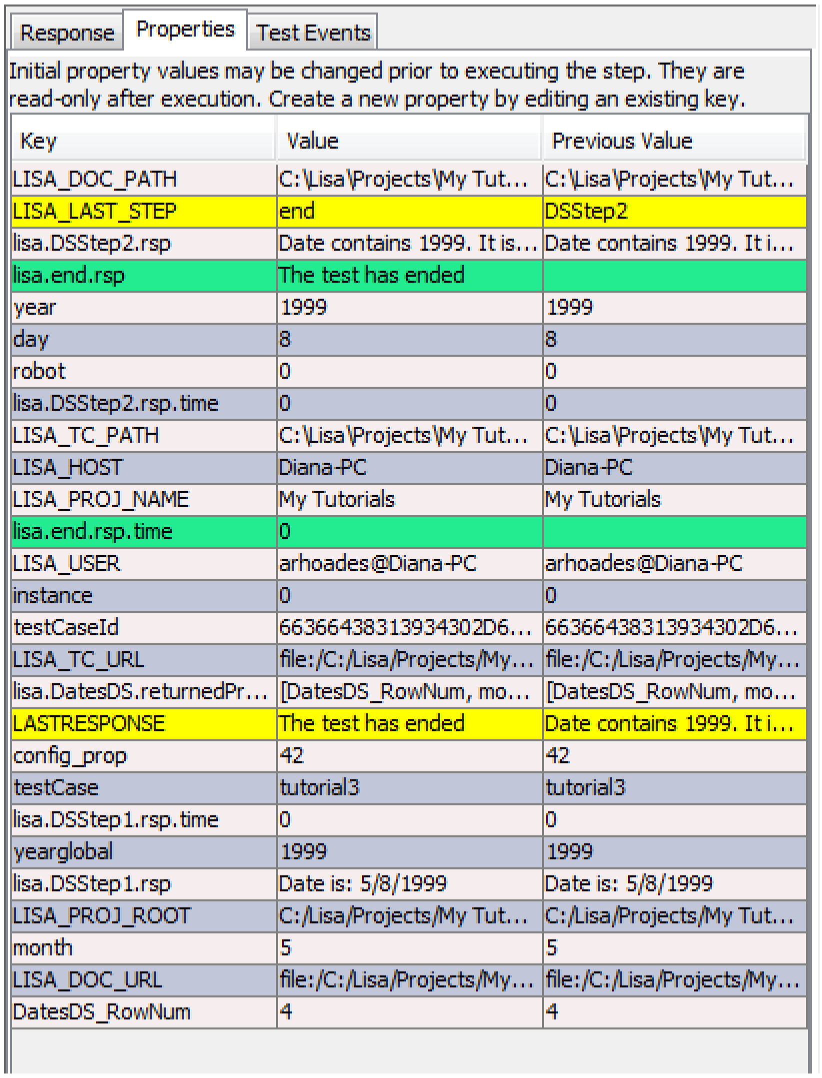
1. Start a new ITR session.
2. In the Execution History pane, click Automatically execute test .
3. When the test is complete, click OK.
4. Review the Response tab.

   Note: The DSstep2 step is executed next when DSstep1 encounters a date in which the year is 1999.

   ```
   Date contains 1999. It is: 5/8/1999.
   ```

5. Click the Properties tab and review the behavior of the properties.

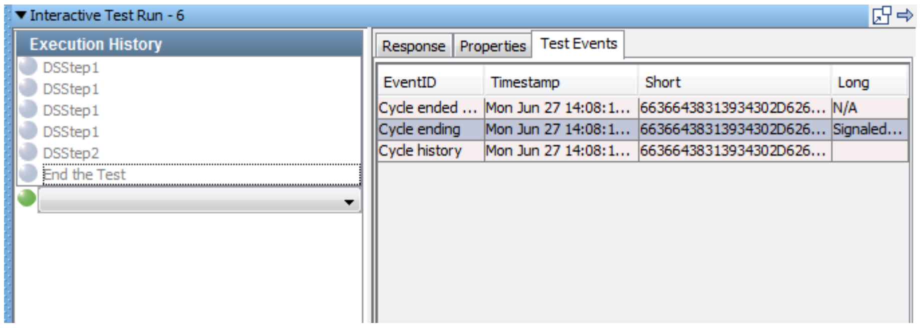


Response | Properties | Test Events

Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
LISA_DOC_PATH	C:\Lisa\Projects\My Tut...	C:\Lisa\Projects\My Tut...
LISA_LAST_STEP	end	DSStep2
lisa.DSStep2.rsp	Date contains 1999. It is...	Date contains 1999. It i...
lisa.end.rsp	The test has ended	
year	1999	1999
day	8	8
robot	0	0
lisa.DSStep2.rsp.time	0	0
LISA_TC_PATH	C:\Lisa\Projects\My Tut...	C:\Lisa\Projects\My Tut...
LISA_HOST	Diana-PC	Diana-PC
LISA_PROJ_NAME	My Tutorials	My Tutorials
lisa.end.rsp.time	0	
LISA_USER	arhoades@Diana-PC	arhoades@Diana-PC
instance	0	0
testCaseId	6636643831393430 2D6...	6636643831393430 2D6...
LISA_TC_URL	file:/C:/Lisa/Projects/My...	file:/C:/Lisa/Projects/My...
lisa.DatesDS.returnedPr...	[DatesDS_RowNum, mo...	[DatesDS_RowNum, mo...
LASTRESPONSE	The test has ended	Date contains 1999. It i...
config_prop	42	42
testCase	tutorial3	tutorial3
lisa.DSStep1.rsp.time	0	0
yearglobal	1999	1999
lisa.DSStep1.rsp	Date is: 5/8/1999	Date is: 5/8/1999
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tut...	C:/Lisa/Projects/My Tut...
month	5	5
LISA_DOC_URL	file:/C:/Lisa/Projects/My...	file:/C:/Lisa/Projects/My...
DatesDS_RowNum	4	4

6. Click the Test Events tab and review the events that were generated.



EventID	Timestamp	Short	Long
Cycle ended ...	Mon Jun 27 14:08:1...	66366438313934302D626...	N/A
Cycle ending	Mon Jun 27 14:08:1...	66366438313934302D626...	Signaled...
Cycle history	Mon Jun 27 14:08:1...	66366438313934302D626...	

Step 5 - Add a Filter

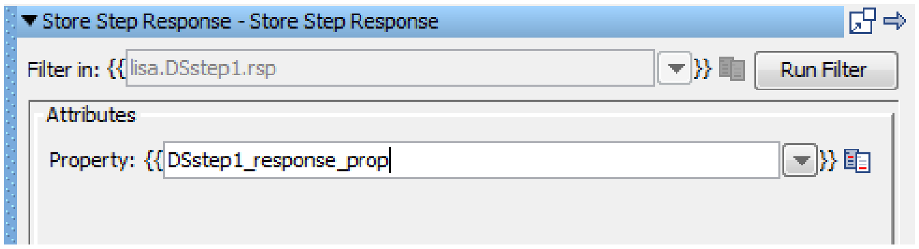
You can add various types of filters to a test case. In this procedure, you add a utility filter with the name **Store Step Response**. This type of filter lets you save the step response as a property.

Follow these steps:

1. In the model editor, select DSstep1.
2. Open the Filters tab.
3. Click ✚ Add.
4. From the Utility Filters submenu, select Store Step Response.

   The filter editor opens.
5. In the filter editor, set the property name to **DSstep1_response_prop**.

   This property is where the step response is stored.



lisa.DSstep1.rsp

DSstep1_response_prop

6. In the model editor, double-click DSstep2 and add the following text to the end of the output log message:
   `The value of DSstep1_response_prop is: {{DSstep1_response_prop}}.`
7. Click Save Save.

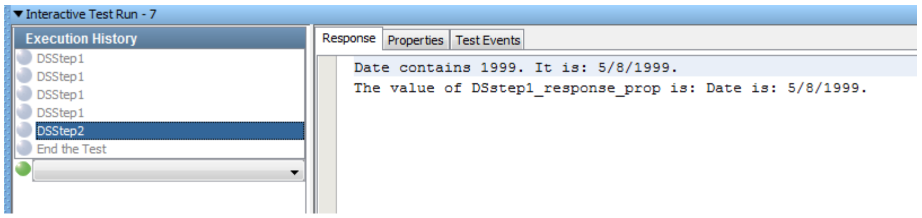
Step 6 - Test the Filter

To determine whether the filter works as expected, use the Interactive Test Run (ITR) utility.

Follow these steps:

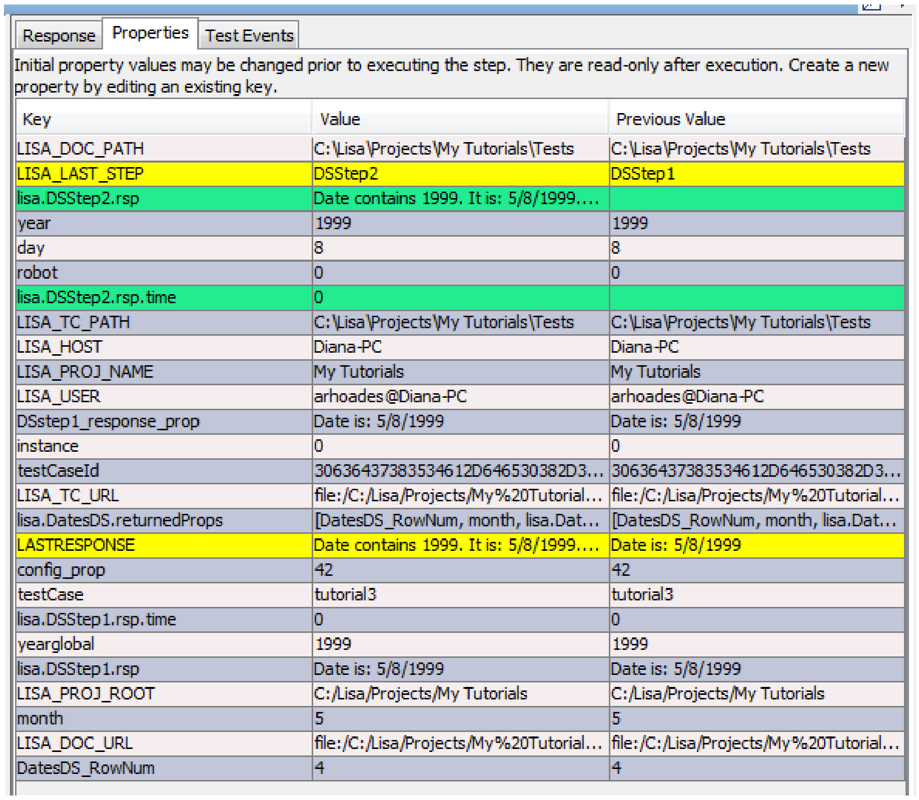
1. Start a new ITR session.
2. In the Execution History pane, click Automatically execute test .
3. When the test is complete, click OK.
4. Review the Response tab.

   The DSstep2 test step now displays the additional text that you added to the output log message.

   ```
   Date contains 1999. It is: 5/8/1999.
   The value of DSstep1_response_prop is: Date is: 5/8/1999.
   ```

5. Click the Properties tab and observe where the DSstep1_response_prop property is created and modified.

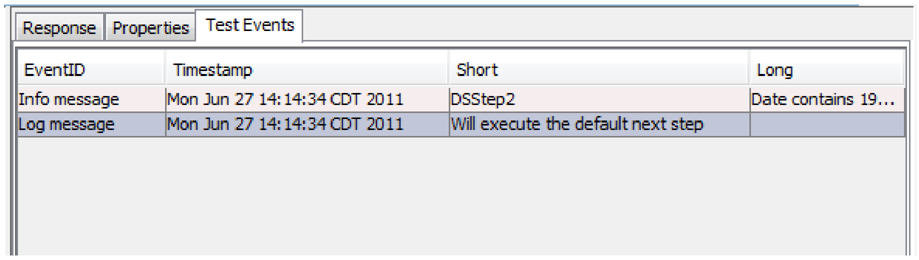


Response | Properties | Test Events

Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
LISA_DOC_PATH	C:\Lisa\Projects\My Tutorials\Tests	C:\Lisa\Projects\My Tutorials\Tests
LISA_LAST_STEP	DSStep2	DSStep1
lisa.DSStep2.rsp	Date contains 1999. It is: 5/8/1999....	
year	1999	1999
day	8	8
robot	0	0
lisa.DSStep2.rsp.time	0	
LISA_TC_PATH	C:\Lisa\Projects\My Tutorials\Tests	C:\Lisa\Projects\My Tutorials\Tests
LISA_HOST	Diana-PC	Diana-PC
LISA_PROJ_NAME	My Tutorials	My Tutorials
LISA_USER	arhoades@Diana-PC	arhoades@Diana-PC
DSstep1_response_prop	Date is: 5/8/1999	Date is: 5/8/1999
instance	0	0
testCaseId	30636437383534612D646530382D3...	30636437383534612D646530382D3...
LISA_TC_URL	file:/C:/Lisa/Projects/My%20Tutorial...	file:/C:/Lisa/Projects/My%20Tutorial...
lisa.DatesDS.returnedProps	[DatesDS_RowNum, month, lisa.Dat...	[DatesDS_RowNum, month, lisa.Dat...
LASTRESPONSE	Date contains 1999. It is: 5/8/1999....	Date is: 5/8/1999
config_prop	42	42
testCase	tutorial3	tutorial3
lisa.DSStep1.rsp.time	0	0
yearglobal	1999	1999
lisa.DSStep1.rsp	Date is: 5/8/1999	Date is: 5/8/1999
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tutorials	C:/Lisa/Projects/My Tutorials
month	5	5
LISA_DOC_URL	file:/C:/Lisa/Projects/My%20Tutorial...	file:/C:/Lisa/Projects/My%20Tutorial...
DatesDS_RowNum	4	4

6. Click the Test Events tab and review the events that were generated.



Response | Properties | Test Events

EventID	Timestamp	Short	Long
Info message	Mon Jun 27 14:14:34 CDT 2011	DSStep2	Date contains 19...
Log message	Mon Jun 27 14:14:34 CDT 2011	Will execute the default next step	

Tutorial 3 - Review

In this tutorial, you:

- Took a first look at filters and assertions.
- Opened and modified an existing test case.
- Learned how to add a simple assertion.
- Learned how to add a simple filter.
- Used the Interactive Test Run utility to validate that the assertion and filter worked as expected.

More Information

DevTest provides filters and assertions to cover most of the situations that you encounter in your test case development. If no appropriate filter exists, DevTest provides a mechanism for developing custom filters and assertions through the Software Developers Kit (SDK). See *Using the SDK* for more information.

Tutorial 4 - Manipulate Java Objects (POJOs)

In this tutorial, you create and manipulate a simple Java object and use the **java.util.Date** class to create a date object.

First, you construct the object and review how to call methods on the object. Then you incorporate the object into a simple DevTest model editor.

Tutorial Tasks

In this tutorial, you:

- Use the Dynamic Java Execution test step
- Use the Complex Object Editor for simple objects
- Use inline filters and save the results into a property

Prerequisites

- You have completed Tutorial 3 - Filters and Assertions.
- DevTest Workstation is open.

Step 1 - Create a Test Case

To create a test case:

- In the My Tutorials project, create a test case with the name **tutorial4**.

Step 2 - Create a Dynamic Java Execution Test Step

The Dynamic Java Execution test step lets you create a Java object from a class in the DevTest classpath. In the following procedure, you use the **java.util.Date** class.

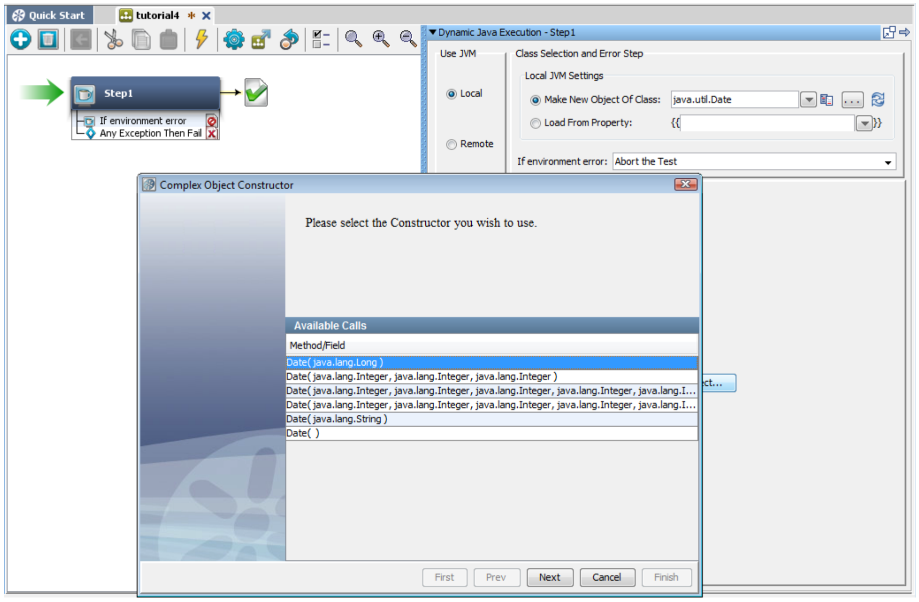
Follow these steps:

1. Click ➕ Add Step.

   The Add step menu is displayed.

2. Select Java/J2EE and then select Dynamic Java Execution.

   The Dynamic Java Execution editor opens.



[✓ Local] Remote

[✓ Make New Object Of Class:] java.util.Date

Load From Property: {{ }}

If environment error: Abort the Test

Please select the Constructor you wish to use.

```
Date( java.lang.Long )
Date( java.lang.Integer, java.lang.Integer, java.lang.Integer )
Date( java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.I...
Date( java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.I...
Date( java.lang.String )
Date( )
```

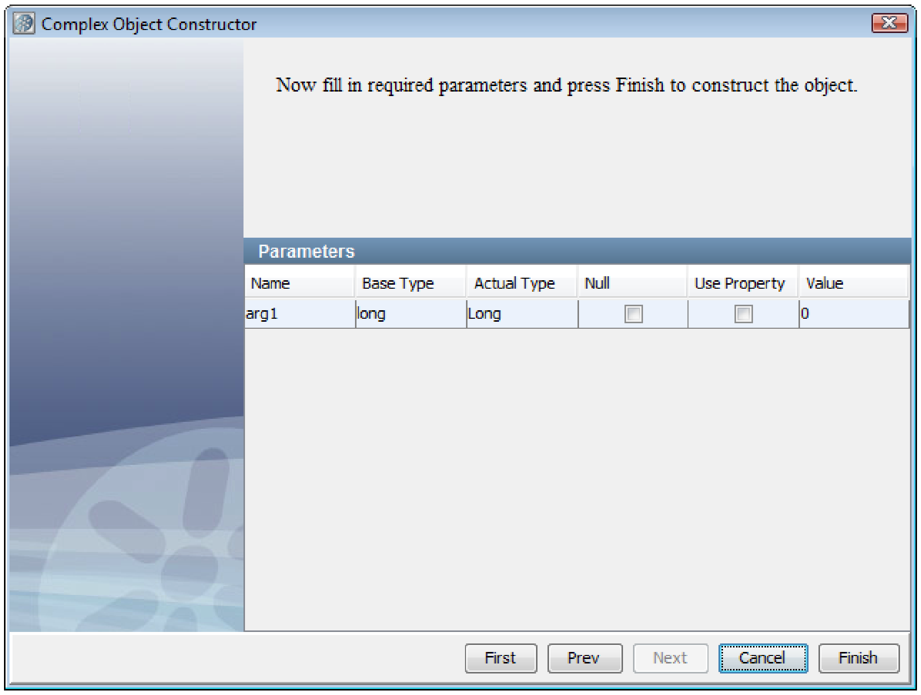
3. In the Local JVM Settings area, ensure that Make New Object of Class is selected.

4. In the field to the right of Make New Object of Class, type **java.util.Date**.

5. Click Construct/Load Object.

   The Complex Object Constructor wizard appears. The first step shows the available constructors.

6. Select the Date(java.lang.Long) constructor.



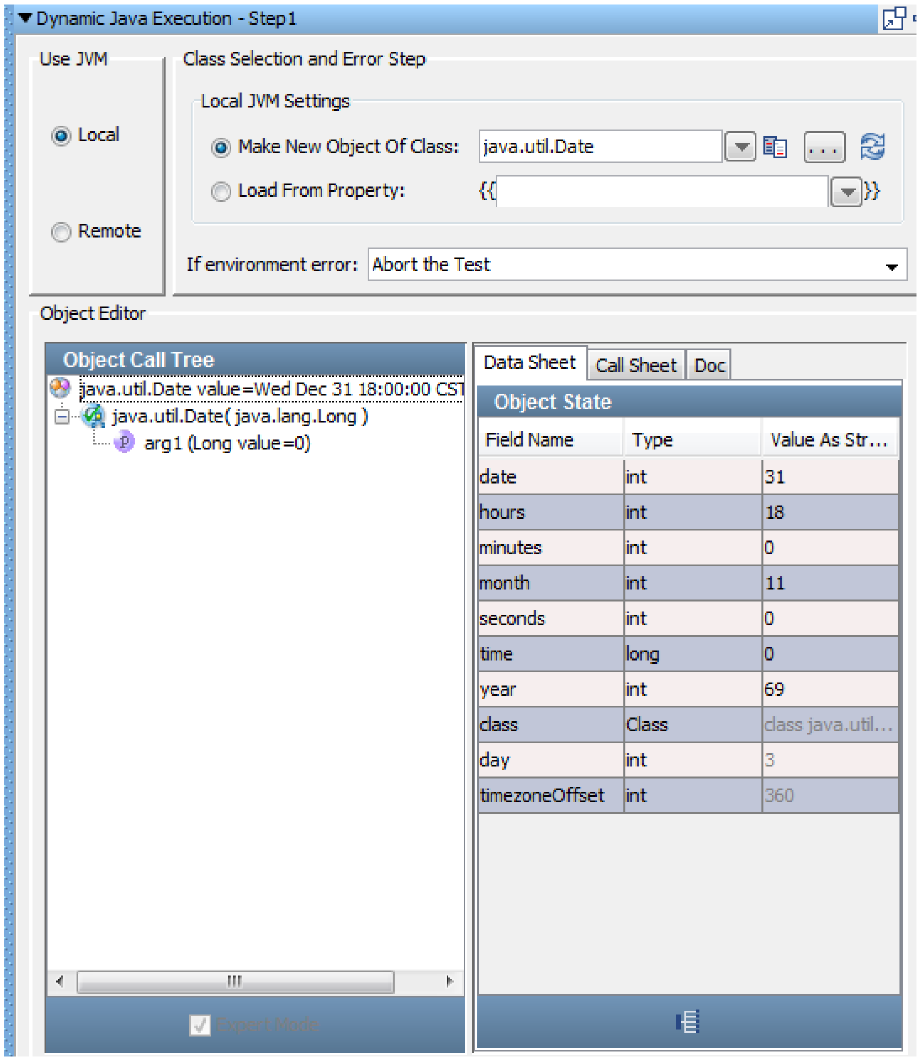
Now fill in required parameters and press Finish to construct the object.

Parameters

Name	Base Type	Actual Type	Null	Use Property	Value
arg1	long	Long	☐	☐	0

7. Click Next.
8. Click Finish.

   The Complex Object Editor opens.



Local JVM Settings

(● Local) ○ Remote

● Make New Object Of Class: java.util.Date

○ Load From Property: {{ }}

If environment error: Abort the Test

java.util.Date value=Wed Dec 31 18:00:00 CST
java.util.Date(java.lang.Long)
arg1 (Long value=0)

Field Name	Type	Value As Str...
date	int	31
hours	int	18
minutes	int	0
month	int	11
seconds	int	0
time	long	0
year	int	69
class	Class	class java.util...
day	int	3
timezoneOffset	int	360

Now you have a Java object to manipulate in the Complex Object Editor.

Step 3 - Make a Call on the Java Object

The Complex Object Editor is divided into two panels. The left panel contains the Object Call Tree, which tracks method invocations and their input parameters and return values. The following icons are used to identify the branches in the object call tree:

The type (class) of the currently loaded object, followed by the response from calling the 'toString' method of the object.

The Constructor that was called. This icon is shown if multiple constructors exist.

A method call that has not been executed.

A method call that has been executed.

The input parameters (type and current value) for the enclosing method.

The return value (current value if the call has been executed) for the enclosing method.

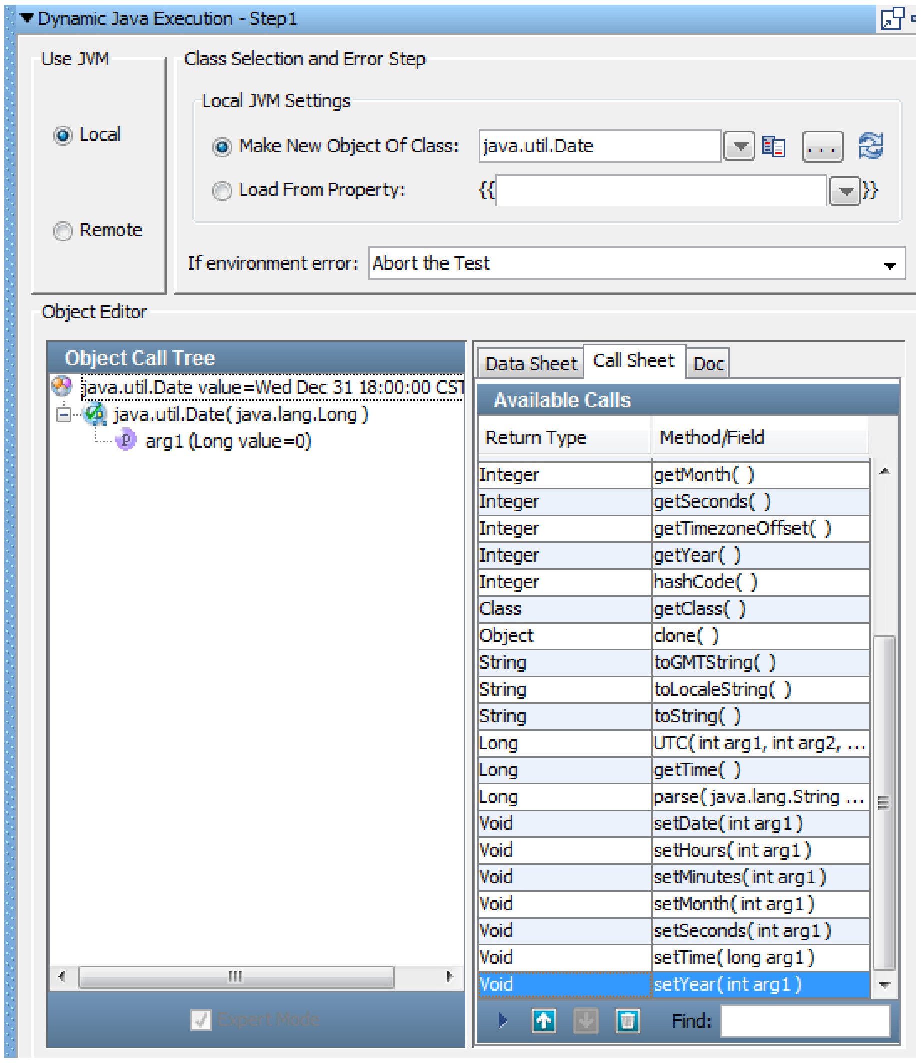
The contents of the right panel vary depending on what is selected in the left panel.

To call the Java object:

1. In the right panel of the Complex Object Editor, click the Call Sheet tab.

   The Call Sheet tab shows the available methods that you can call.

2. Double-click the setYear() method. Or, you can select the setYear() method and click Add selected method to Object Call Tree.



Make New Object Of Class: java.util.Date

If environment error: Abort the Test

Object Call Tree:

```
java.util.Date value=Wed Dec 31 18:00:00 CST
  java.util.Date( java.lang.Long )
    arg1 (Long value=0)
```

Available Calls:

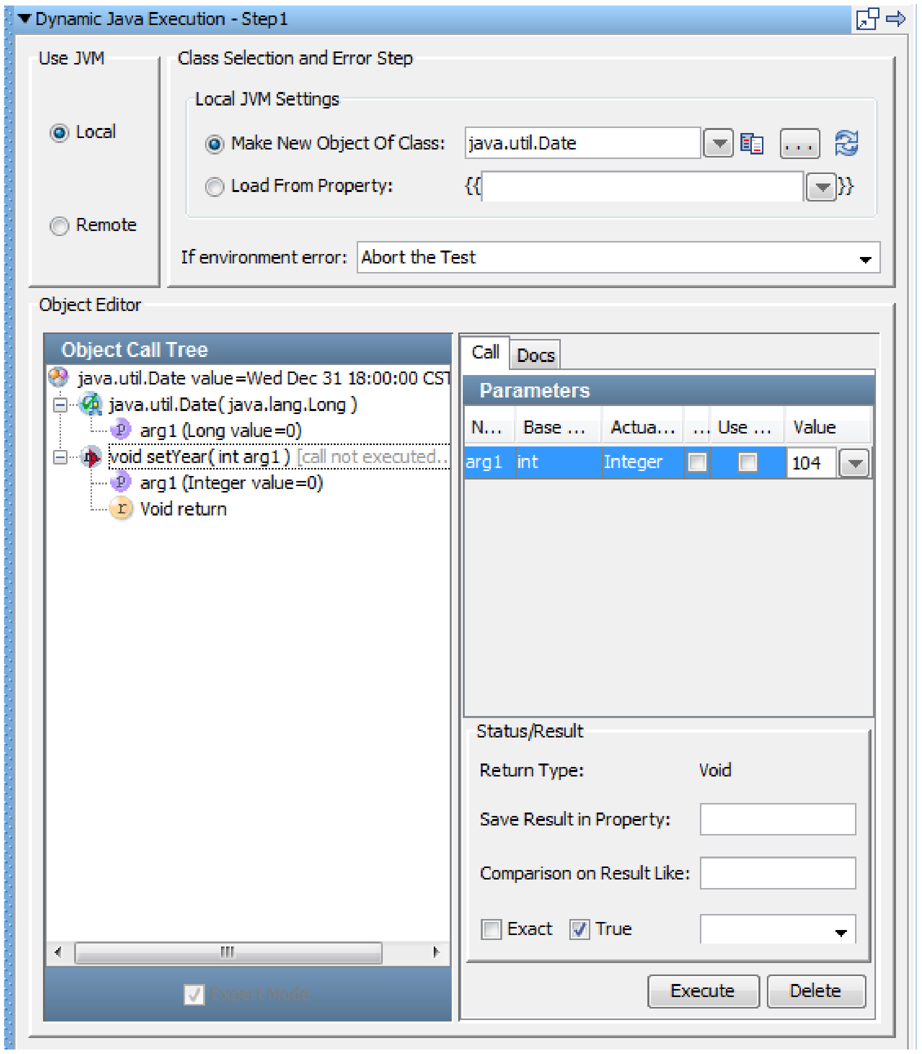
Return Type	Method/Field
Integer	getMonth( )
Integer	getSeconds( )
Integer	getTimezoneOffset( )
Integer	getYear( )
Integer	hashCode( )
Class	getClass( )
Object	clone( )
String	toGMTString( )
String	toLocaleString( )
String	toString( )
Long	UTC(int arg1, int arg2, ...
Long	getTime( )
Long	parse(java.lang.String ...
Void	setDate(int arg1)
Void	setHours(int arg1)
Void	setMinutes(int arg1)
Void	setMonth(int arg1)
Void	setSeconds(int arg1)
Void	setTime(long arg1)
Void	setYear(int arg1)

The setYear() method is added to the Object Call Tree. The right panel now displays the Call tab and Docs tab.

The Call tab lists the argument information.

3. In the Value field for arg1, enter **104**.

4. Click Execute.


Dynamic Java Execution - Step1

Use JVM
[✓ Local] Remote

Class Selection and Error Step
Local JVM Settings
[✓ Make New Object Of Class:] java.util.Date
Load From Property: {{ }}
If environment error: Abort the Test

Object Editor

Object Call Tree
java.util.Date value=Wed Dec 31 18:00:00 CST
java.util.Date(java.lang.Long)
arg1 (Long value=0)
void setYear(int arg1) [call not executed..
arg1 (Integer value=0)
Void return

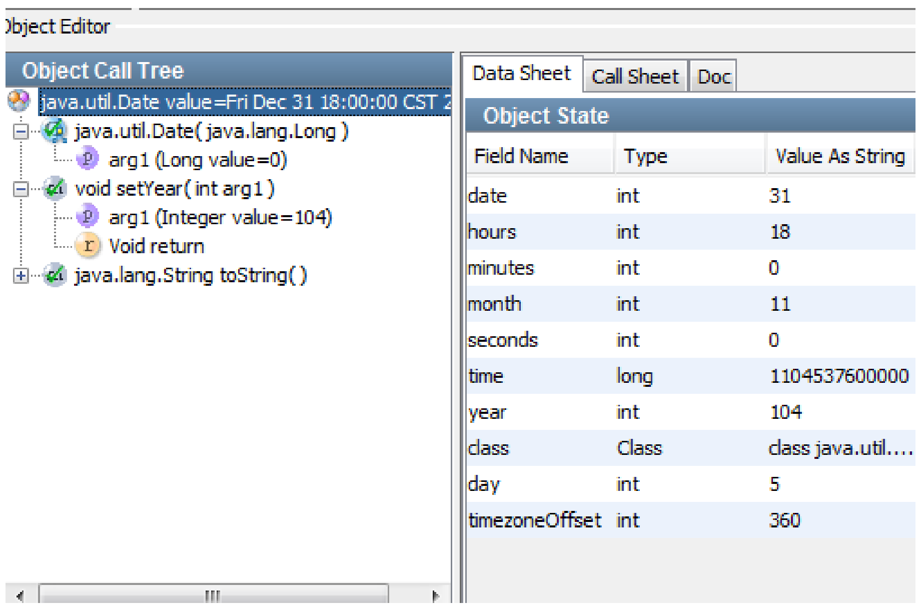
Call | Docs

Parameters

N...	Base ...	Actua...	...	Use ...	Value
arg1	int	Integer	☐	☐	104

Status/Result
Return Type: Void
Save Result in Property:
Comparison on Result Like:
☐ Exact ☑ True

5. In the Object Call Tree, select the **java.util.Date** object.
6. In the Data Sheet tab, verify that the year field is now set to 104.



```
java.util.Date value=Fri Dec 31 18:00:00 CST 2
  java.util.Date( java.lang.Long )
    arg1 (Long value=0)
  void setYear( int arg1 )
    arg1 (Integer value=104)
    Void return
  java.lang.String toString( )
```

Field Name	Type	Value As String
date	int	31
hours	int	18
minutes	int	0
month	int	11
seconds	int	0
time	long	1104537600000
year	int	104
class	Class	class java.util....
day	int	5
timezoneOffset	int	360

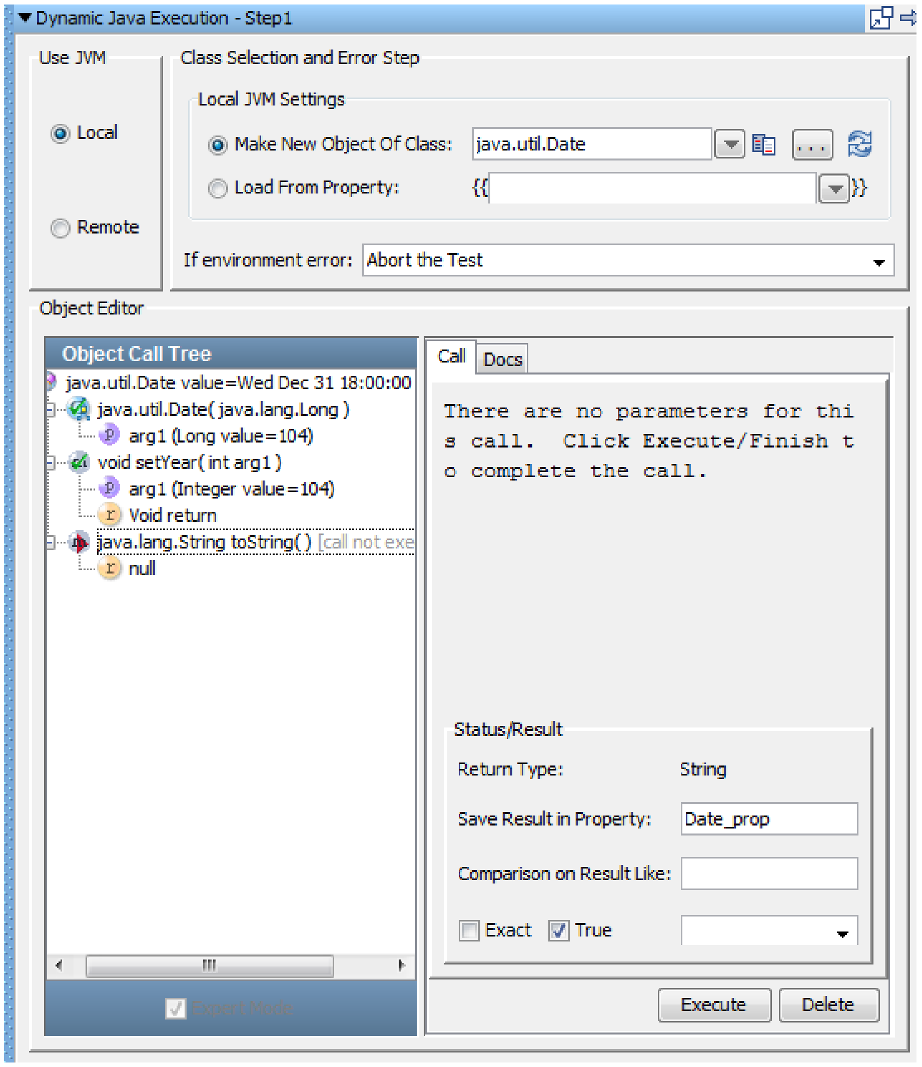
Step 4 - Add an Inline Filter

You can add an inline filter from the Complex Object Editor. Inline filters (and assertions) do not result in a filter being added to the test step in the elements panel. Inline filter management is always done in the Complex Object Editor.

Follow these steps:

1. With the **java.util.Date** object selected, click the Call Sheet tab.
2. To retrieve the date to be placed in a property, invoke the toString() method.

   The right panel now displays the Call tab and Docs tab.
3. To add an inline filter, enter **Date_prop** as the property name in the Save Result in Property field in the Status/Result area on the Call tab.



Make New Object Of Class: java.util.Date

If environment error: Abort the Test

```
java.util.Date value=Wed Dec 31 18:00:00
  java.util.Date( java.lang.Long )
    arg1 (Long value=104)
  void setYear( int arg1 )
    arg1 (Integer value=104)
    Void return
  java.lang.String toString( ) [call not exe
    null
```

```
There are no parameters for thi
s call.  Click Execute/Finish t
o complete the call.
```

Status/Result

Return Type: String

Save Result in Property: Date_prop

Comparison on Result Like:

Exact [✓ True]

4. Click Execute.

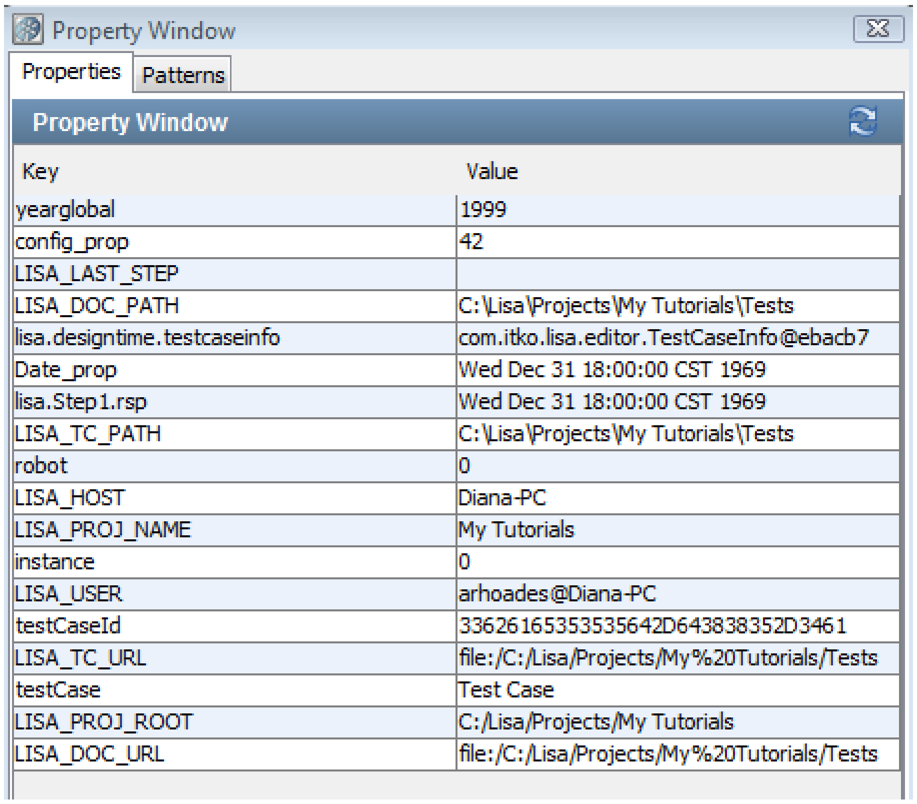
5. Click Save Save.

Step 5 - Verify the Property Created

You can display the Property Window to verify that the **Date_prop** property was created.

Follow these steps:

1. Click Show model properties on the test case toolbar. Or, you can select Help, View Properties from the main menu.
2. Click Refresh .
3. Locate the **Date_prop** property.



Key	Value
yearglobal	1999
config_prop	42
LISA_LAST_STEP	
LISA_DOC_PATH	C:\Lisa\Projects\My Tutorials\Tests
lisa.designtime.testcaseinfo	com.itko.lisa.editor.TestCaseInfo@ebacb7
Date_prop	Wed Dec 31 18:00:00 CST 1969
lisa.Step1.rsp	Wed Dec 31 18:00:00 CST 1969
LISA_TC_PATH	C:\Lisa\Projects\My Tutorials\Tests
robot	0
LISA_HOST	Diana-PC
LISA_PROJ_NAME	My Tutorials
instance	0
LISA_USER	arhoades@Diana-PC
testCaseId	336261653535642D643838352D3461
LISA_TC_URL	file:/C:/Lisa/Projects/My%20Tutorials/Tests
testCase	Test Case
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tutorials
LISA_DOC_URL	file:/C:/Lisa/Projects/My%20Tutorials/Tests

4. Click Close.

Tutorial 4 - Review

In this tutorial, you:

- Created a test step to manipulate a Java object of **java.util.Date** type.
- Used the Complex Object Editor to manipulate the Java object.
- Learned how to add inline filters to objects and save results into a property.

Tutorial 5 - Run a Demo Server Web Application

In this tutorial, you step through a simple web application that accompanies DevTest.

The LISA Bank application is a simple front end that is connected to a database table containing financial account information. The application business logic consists of Enterprise JavaBeans and web services. From the web application, you can view the profile of the user, create an account, add addresses, and so on.

The goal of this tutorial is for you to become familiar with the application. This application is used in subsequent tutorials as the system under test.

Prerequisites

- The Demo Server is running.

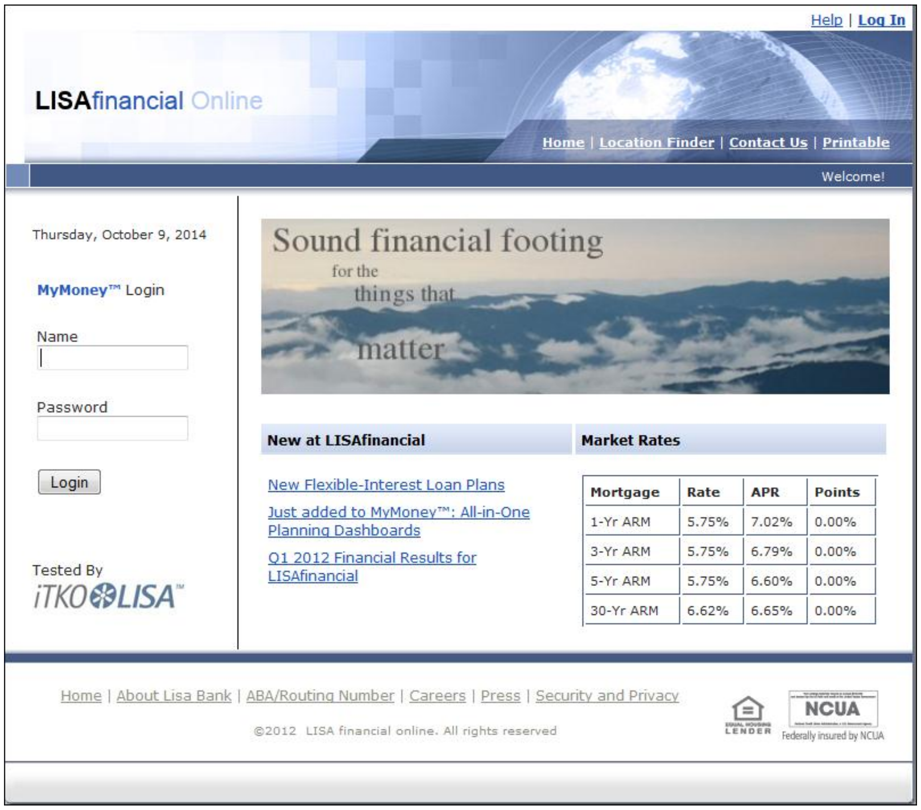
Step 1 - Start the Web Application

Follow these steps:

1. Open a new browser window.
2. Enter the following URL, where localhost is the path with the IP address for your computer.

   `http://localhost:8080/lisabank/`

   The login page appears.


Help | Log In
LISAfinancial Online
Home | Location Finder | Contact Us | Printable
Welcome!
Thursday, October 9, 2014
MyMoney™ Login
Name
Password
Login
Tested By
iTKO LISA
Sound financial footing for the things that matter
New at LISAfinancial
New Flexible-Interest Loan Plans
Just added to MyMoney™: All-in-One Planning Dashboards
Q1 2012 Financial Results for LISAfinancial
Market Rates
Home | About Lisa Bank | ABA/Routing Number | Careers | Press | Security and Privacy
©2012 LISA financial online. All rights reserved
NCUA
Federally insured by NCUA

Mortgage	Rate	APR	Points
1-Yr ARM	5.75%	7.02%	0.00%
3-Yr ARM	5.75%	6.79%	0.00%
5-Yr ARM	5.75%	6.60%	0.00%
30-Yr ARM	6.62%	6.65%	0.00%

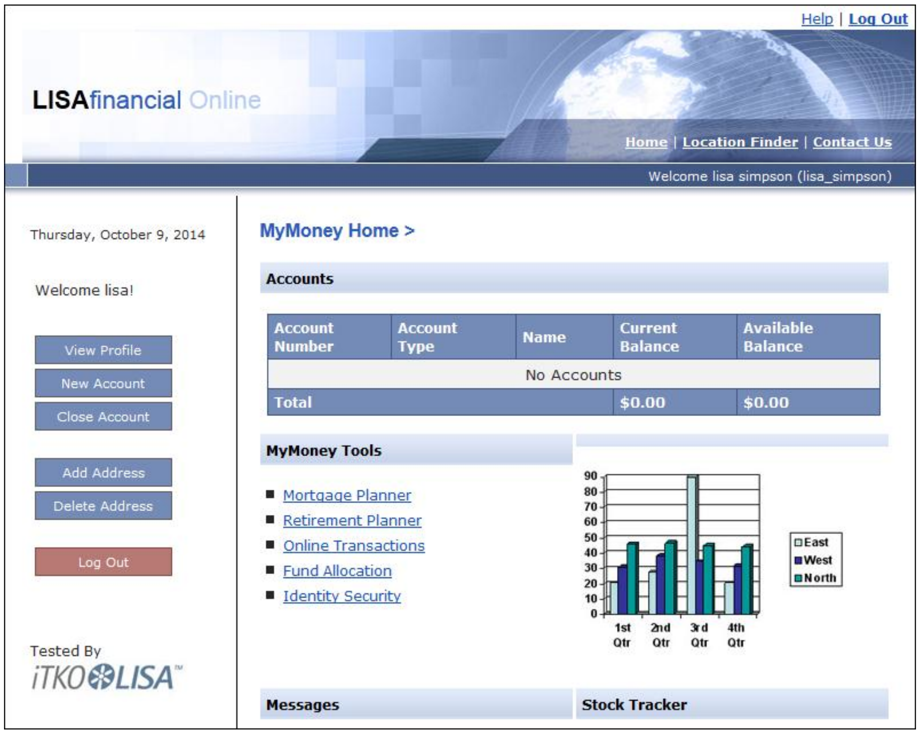
Step 2 - Log in to the Web Application

You use the predefined user name lisa_simpson.

Follow these steps:

1. In the Name field, enter **lisa_simpson**.
2. In the Password field, enter **golisa**.
3. Click Login.

   The welcome page appears. The left side contains buttons for various actions that you can perform: View Profile, New Account, Close Account, Add Address, Delete Address, and Log Out.



Welcome lisa simpson (lisa_simpson)

Thursday, October 9, 2014

Welcome lisa!

MyMoney Home >

Accounts

Account Number	Account Type	Name	Current Balance	Available Balance
No Accounts				
Total			$0.00	$0.00

MyMoney Tools

- Mortgage Planner
- Retirement Planner
- Online Transactions
- Fund Allocation
- Identity Security

Messages

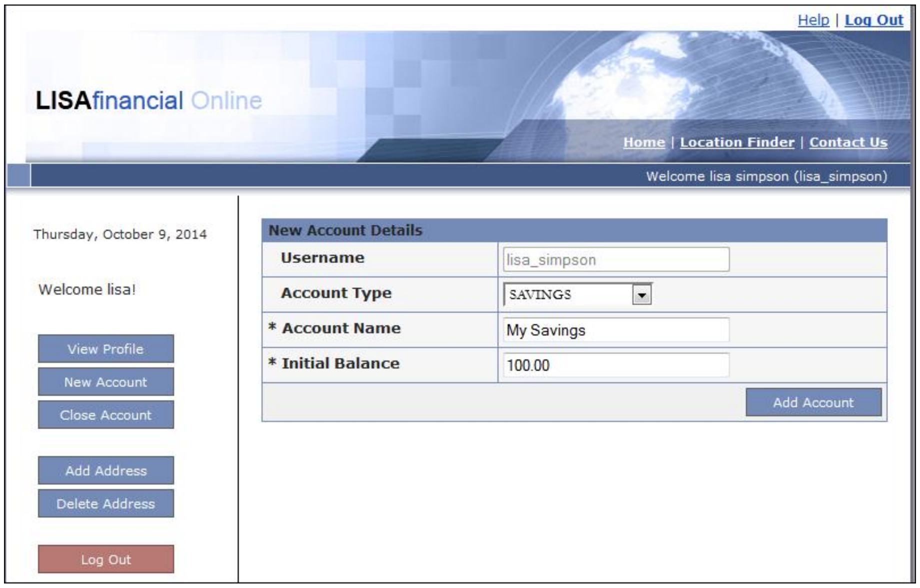
Stock Tracker

Step 3 - Create an Account

Notice that the current user does not have any accounts.

Follow these steps:

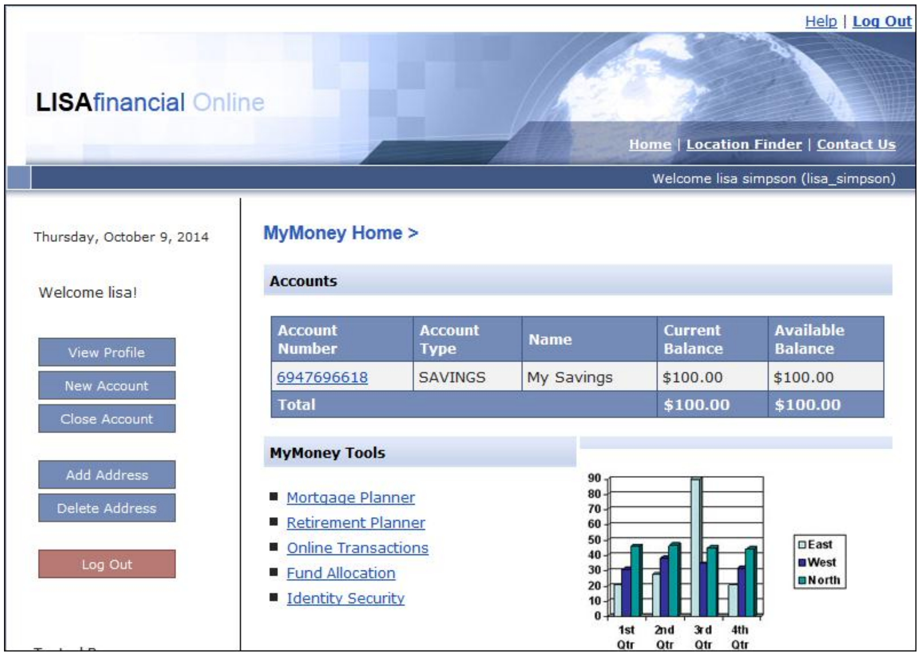
1. Click New Account.
2. From the Account Type list, select SAVINGS.
3. In the Account Name field, enter **My Savings**.
4. In the Initial Balance field, enter **100.00**.
5. Click Add Account.



New Account Details

Username	lisa_simpson
Account Type	SAVINGS
* Account Name	My Savings
* Initial Balance	100.00

The new savings account is added to the Accounts section.

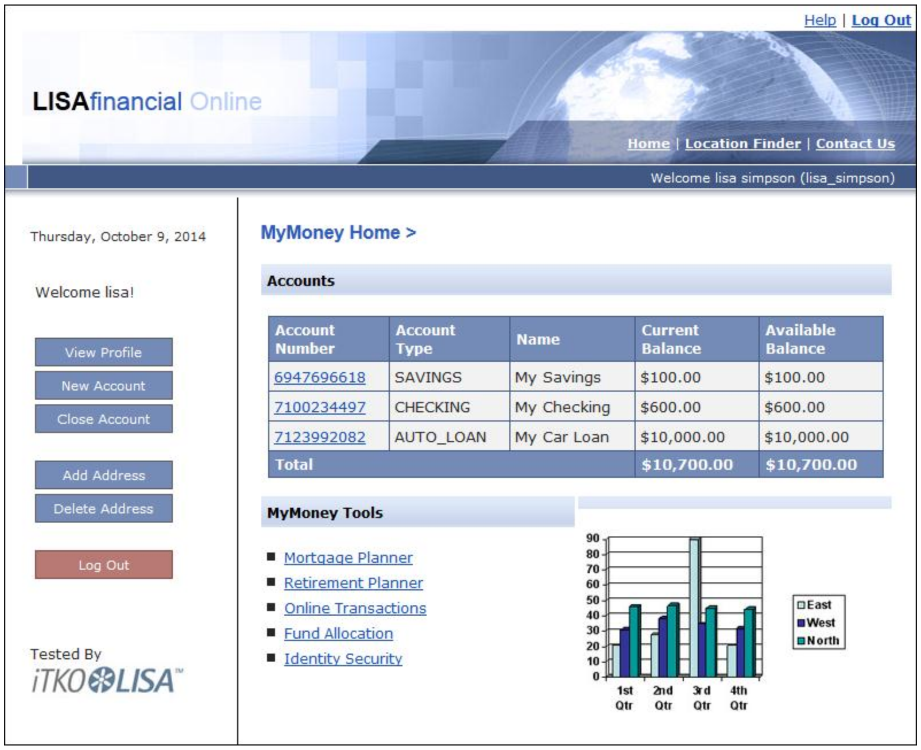


Accounts

Account Number	Account Type	Name	Current Balance	Available Balance
6947696618	SAVINGS	My Savings	$100.00	$100.00
Total			$100.00	$100.00

6. To create two more accounts: Checking and Auto_Loan, repeat the preceding steps. Set the initial balance of the checking account to $600.00. Set the initial balance of the auto loan to $10000.00.

   Do not use commas in the Initial Balance field.



Accounts

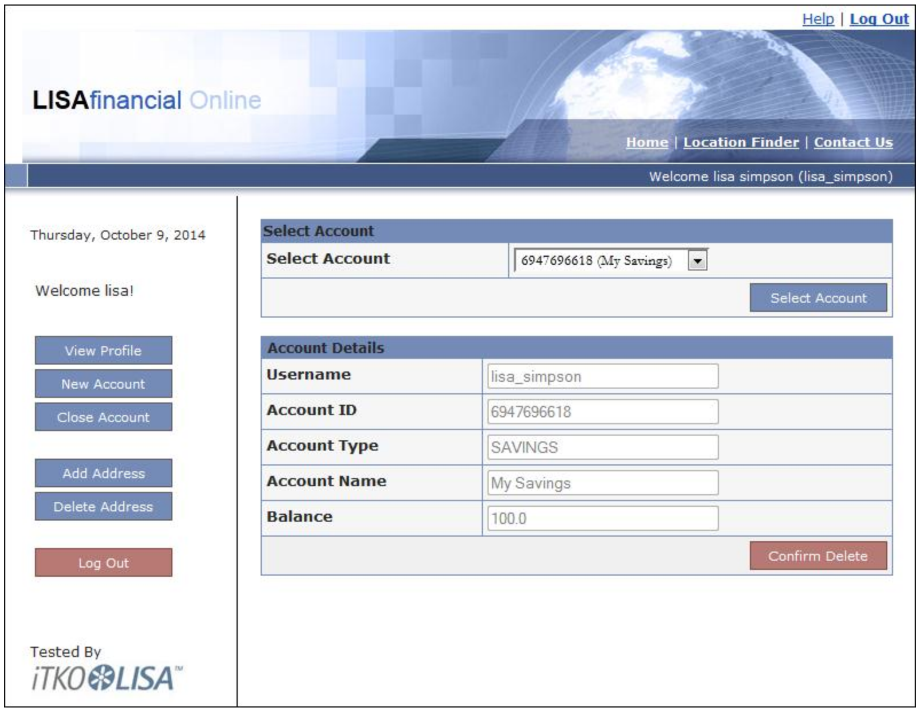
Account Number	Account Type	Name	Current Balance	Available Balance
7100234497	CHECKING	My Checking	$600.00	$600.00
6947696618	SAVINGS	My Savings	$100.00	$100.00
7123992082	AUTO_LOAN	My Car Loan	$10,000.00	$10,000.00
Total			$10,700.00	$10,700.00

Step 4 - Close an Account

The application lets you close accounts.

Follow these steps:

1. Click Close Account.
2. Select **My Savings** from the list and click Select Account.



Welcome lisa simpson (lisa_simpson)

Thursday, October 9, 2014

Welcome lisa!

Select Account

Select Account	6947696618 (My Savings)

Account Details

Username	lisa_simpson
Account ID	6947696618
Account Type	SAVINGS
Account Name	My Savings
Balance	100.0

3. Click Confirm Delete.

   The **My Savings** account is removed from the Accounts section.

Step 5 - Log Out from the Web Application

To log out from the web application:

1. Click Log Out.

Tutorial 5 - Review

In this tutorial, you:

- Logged in to the LISA Bank application.
- Created new accounts.
- Closed an account.

Tutorial 6 - Test a Website

In this tutorial, you use the web recorder to record the path through a website. You then create test steps of HTTP/HTML Request for each HTTP request/response pair.

The HTTP/HTML Request test step enables you to make requests of a web server and receive results in a test case. To verify that the pages work as expected, test a simple website.

Tutorial Tasks

In this tutorial, you:

- Create a test case that contains HTTP/HTML Request steps with the web recorder
- Edit and run the test case that the recorder produces
- Add a data set to the test case

Prerequisites

- You have completed Tutorial 5 - Run a Demo Server Web Application.
- DevTest Workstation is open.
- You have access to the demo server.

Tutorial Parts

- Tutorial 6 - Part A - Record the Test Case (see page 57)
- Tutorial 6 - Part B - Run the Test Case (see page 65)
- Tutorial 6 - Part C - Modify Request Test Steps (see page 66)

Tutorial 6 - Part A - Record the Test Case

With the web recorder, create a test case that contains HTTP/HTML Request steps.

Step 1 - Create a Test Case

Follow these steps:

1. In the My Tutorials project, create a test case with the name **tutorial6a** in the Tests subfolder.

   The model editor opens.

Step 2 - Start the Web Recorder

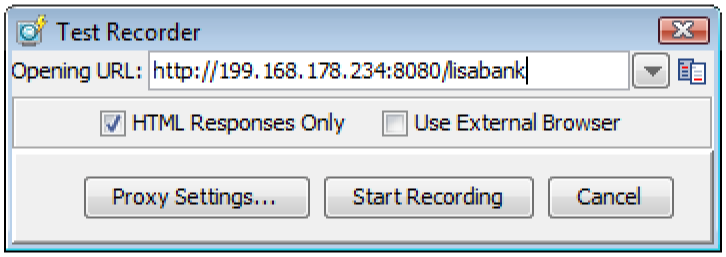
In this tutorial, you record a website through HTTP proxy.

Follow these steps:

1. Select Actions, Record Test Case for User Interface, Web Recorder (HTTP proxy) from the main menu.

   The Test Recorder dialog opens.

2. In the Opening URL field, enter the following URL. Replace the IP address in the path with the IP address for your computer.



Test Recorder

Opening URL: http://199.168.178.234:8080/lisabank

[✓] HTML Responses Only [ ] Use External Browser

3. Click Start Recording.

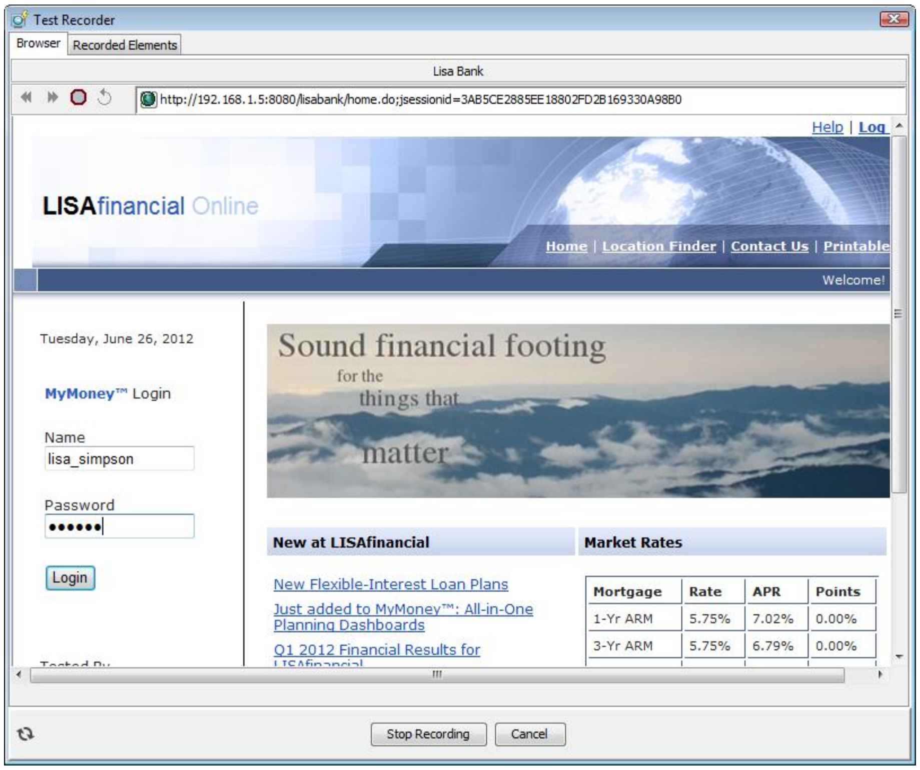
   The Test Recorder window opens. The Test Recorder window contains two tabs: Browser and Recorded Elements. The login page of the LISA Bank application appears in the Browser tab.

Step 3 - Record the LISA Bank Application

While you perform actions in the LISA Bank application, the request and response information for each page visited are recorded.

Follow these steps:

1. In the Name field, enter **lisa_simpson**. In the Password field, enter **golisa**.



http://192.168.1.5:8080/lisabank/home.do;jsessionid=3AB5CE2885EE18802FD2B169330A98B0

Tuesday, June 26, 2012

MyMoney™ Login

Name: lisa_simpson

Password: ••••••

New at LISAfinancial

New Flexible-Interest Loan Plans
Just added to MyMoney™: All-in-One Planning Dashboards
Q1 2012 Financial Results for LISAfinancial

Market Rates

Mortgage	Rate	APR	Points
1-Yr ARM	5.75%	7.02%	0.00%
3-Yr ARM	5.75%	6.79%	0.00%

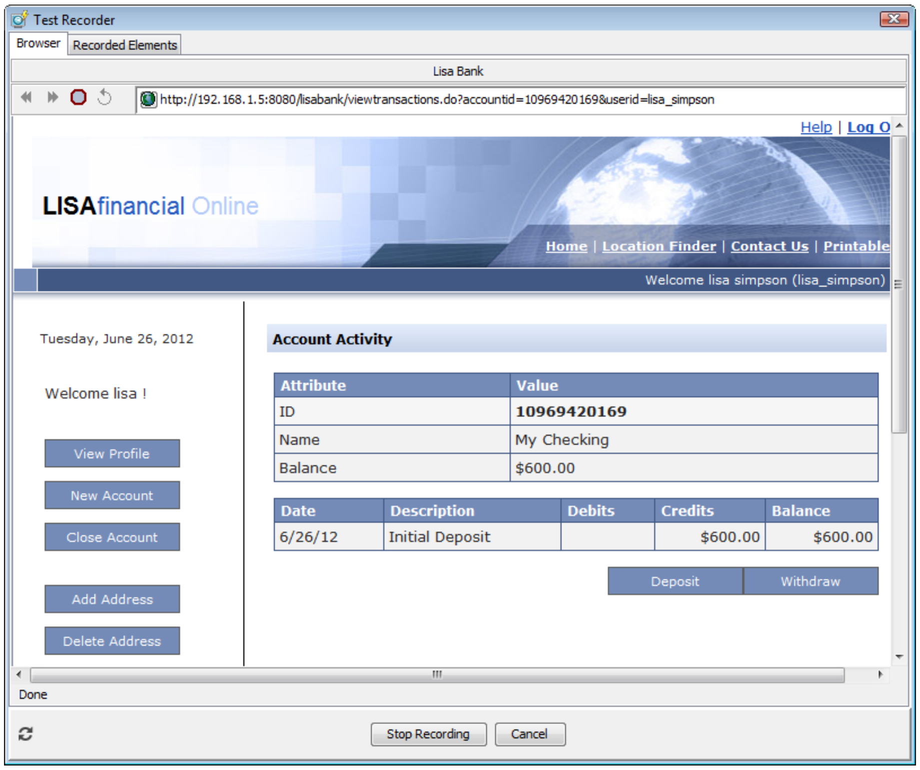
2. Click Login.

   The welcome page appears.

3. In the Accounts section, click the account number of the checking account.

   The Account Activity window appears.

4. Click Deposit.



Tuesday, June 26, 2012

Welcome lisa !

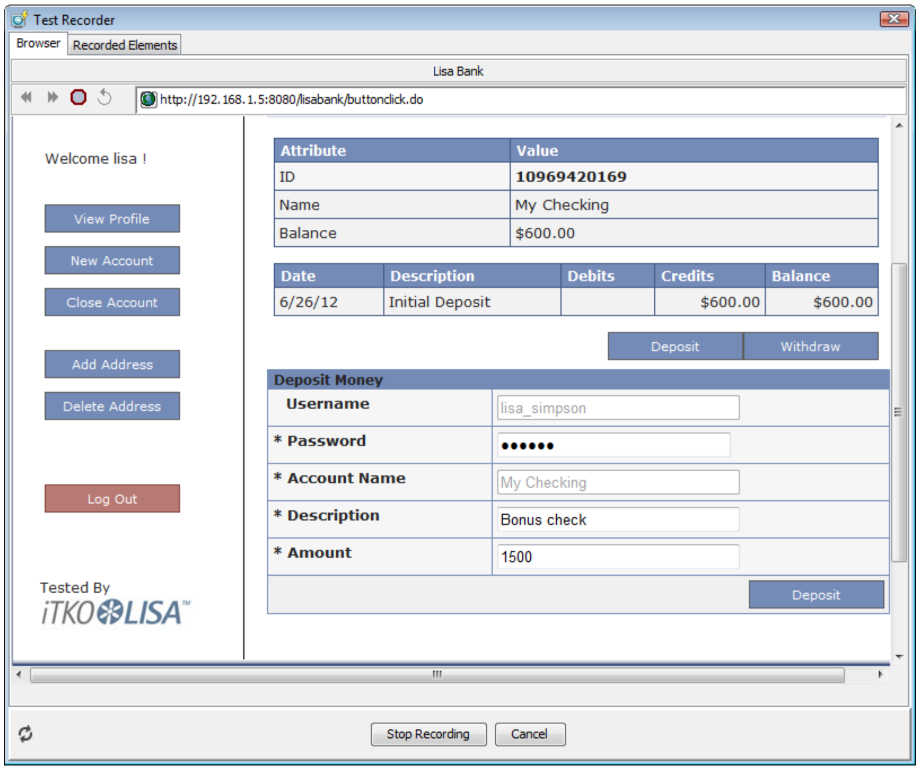
Account Activity

Attribute	Value
ID	**10969420169**
Name	My Checking
Balance	$600.00

Date	Description	Debits	Credits	Balance
6/26/12	Initial Deposit		$600.00	$600.00

5. In the Deposit Money area, enter the password **golisa**, a description of the transaction, and the amount for this deposit.

6. Click Deposit.



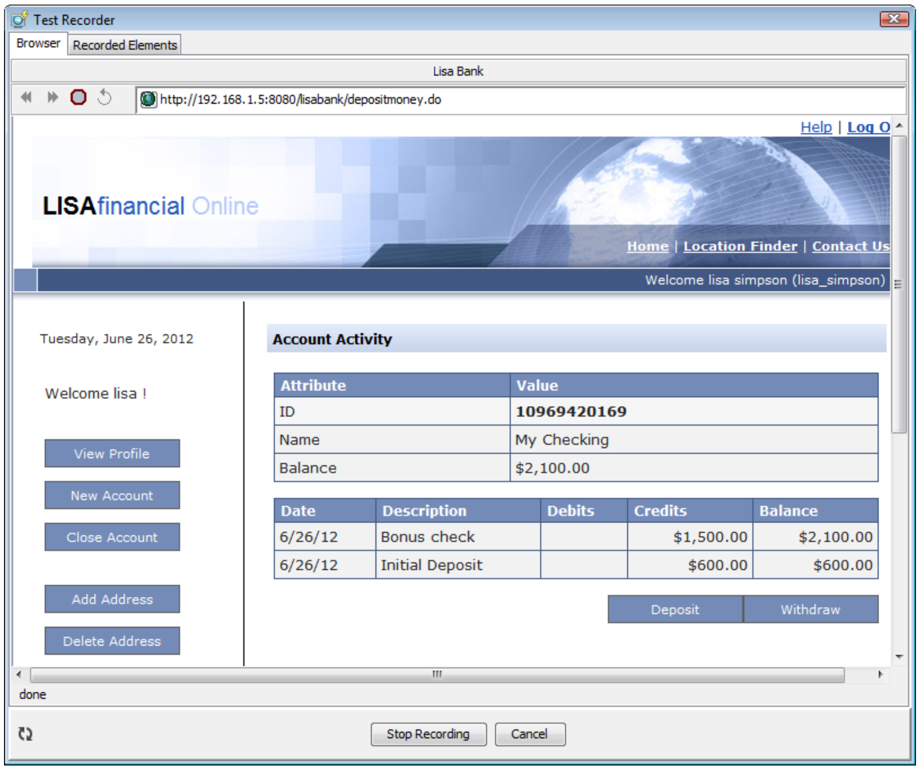
Attribute	Value
ID	**10969420169**
Name	My Checking
Balance	$600.00

Date	Description	Debits	Credits	Balance
6/26/12	Initial Deposit		$600.00	$600.00

Deposit Money

Username	lisa_simpson
* **Password**	••••••
* **Account Name**	My Checking
* **Description**	Bonus check
* **Amount**	1500

The Account Activity window shows the updated balance and a record of the deposit.



Tuesday, June 26, 2012

Welcome lisa !

Account Activity

Attribute	Value
ID	**10969420169**
Name	My Checking
Balance	$2,100.00

Date	Description	Debits	Credits	Balance
6/26/12	Bonus check		$1,500.00	$2,100.00
6/26/12	Initial Deposit		$600.00	$600.00

7. From the left navigation pane, click Log Out.

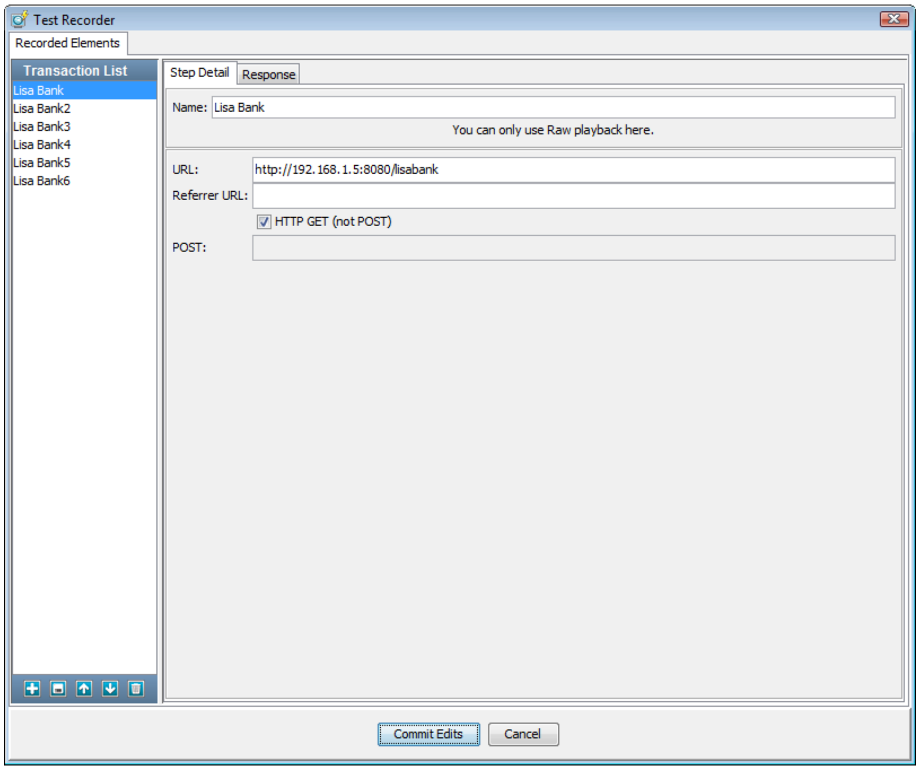
Step 4 - Stop the Web Recorder

After you stop recording, you can view details about the transactions.

Follow these steps:

1. At the bottom of the Test Recorder window, click Stop Recording.

   Filters and properties are automatically created for the web page references. The form fields for the deposit are also displayed. The left pane shows a list of transactions (steps). The right pane shows the step detail and response for the selected transaction.



Name: Lisa Bank

You can only use Raw playback here.

URL: http://192.168.1.5:8080/lisabank

Referrer URL:

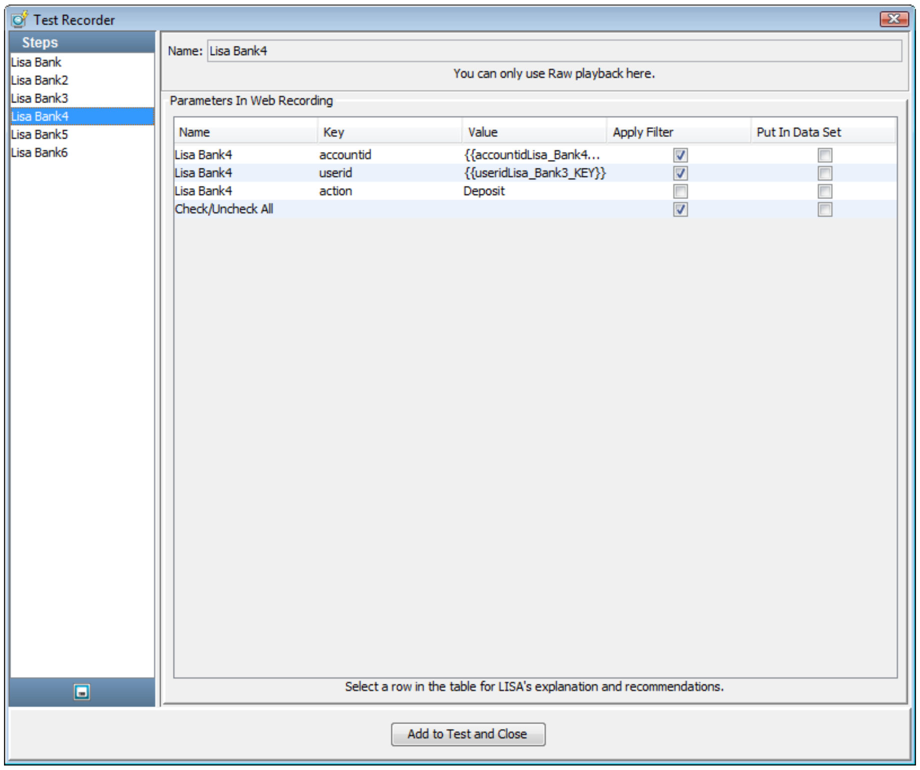
[✓] HTTP GET (not POST)

POST:

2. Click Commit Edits.

   The parameters page appears.

3. Click Add to Test and Close.


Test Recorder

Steps
- Lisa Bank
- Lisa Bank2
- Lisa Bank3
- Lisa Bank4
- Lisa Bank5
- Lisa Bank6

Name: Lisa Bank4

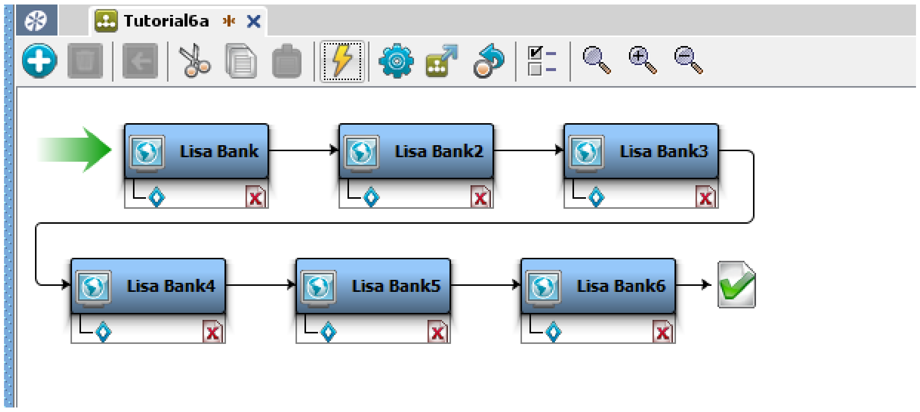
You can only use Raw playback here.

Parameters In Web Recording

Name	Key	Value	Apply Filter	Put In Data Set
Lisa Bank4	accountid	{{accountidLisa_Bank4...	✓	☐
Lisa Bank4	userid	{{useridLisa_Bank3_KEY}}	✓	☐
Lisa Bank4	action	Deposit	☐	☐
Check/Uncheck All			✓	☐

Select a row in the table for LISA's explanation and recommendations.

The model editor is populated with a new test case, having all the transaction information from the saved recording. Each test step in the test case represents an HTTP request.



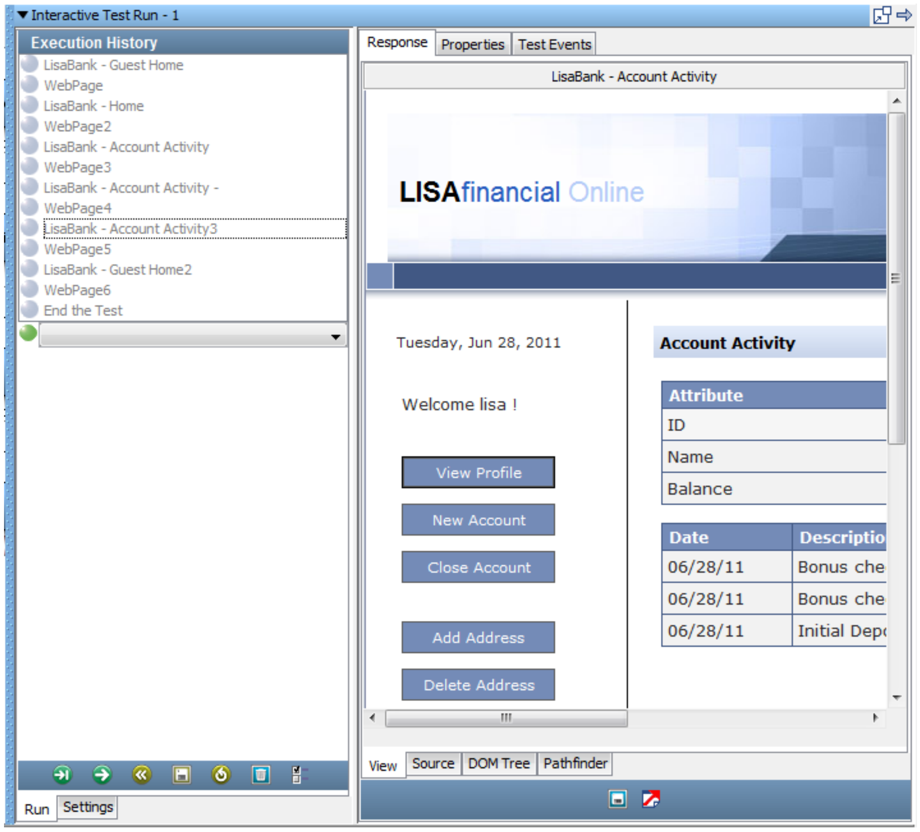
4. Save the test case.

Tutorial 6 - Part B - Run the Test Case

In this section, continue from Part A to run the saved test case in the Interactive Test Run (ITR) utility and view the results.

Follow these steps:

1. Start a new ITR session.
2. In the Execution History pane, click Automatically execute test.
3. When the test is complete, click OK.



Interactive Test Run - 1

Execution History
- LisaBank - Guest Home
- WebPage
- LisaBank - Home
- WebPage2
- LisaBank - Account Activity
- WebPage3
- LisaBank - Account Activity -
- WebPage4
- LisaBank - Account Activity3
- WebPage5
- LisaBank - Guest Home2
- WebPage6
- End the Test

LisaBank - Account Activity

LISAfinancial Online

Tuesday, Jun 28, 2011

Welcome lisa !

Account Activity

Attribute
ID
Name
Balance

Date	Descriptio
06/28/11	Bonus che
06/28/11	Bonus che
06/28/11	Initial Dep

The View tab shows the rendered pages while the ITR replays the deposit into the checking account.

The Source tab shows the HTML code for the page that is captured in the step.

DevTest acts as the browser and sends the same HTTP requests to the web server.

4. Close the ITR utility.

Tutorial 6 - Part C - Modify Request Test Steps

The web recorder produces an HTTP/HTML Request test step for each request.

You can edit and modify these test steps like the other test steps in DevTest. The recorder uses the parameters that you entered during the recording as values for the Post and Get parameters in the request.

To generalize your LISA Bank test, replace these hard-coded description and deposit amount values (for example, "cash" and "1000.00") with properties from a data set. You previously worked with data sets in Tutorial 2 - Data Sets.

In the LISA Bank5 test step from the recording results, the Host Name and Port parameters are parameterized and added to the configuration. The values for description and amount are hard-coded.

In this part of the tutorial, we use a numeric counting data set to parameterize the test case so it deposits different amounts of money. When you then run the test case, it uses deposit values different from the ones recorded.

Step 1 - Copy a Test Case

Follow these steps:

1. Ensure that the tutorial6a test case is open in the model editor.
2. Select File, Save As from the menu bar.
3. In the File name field, enter tutorial6c.
4. Click Save.

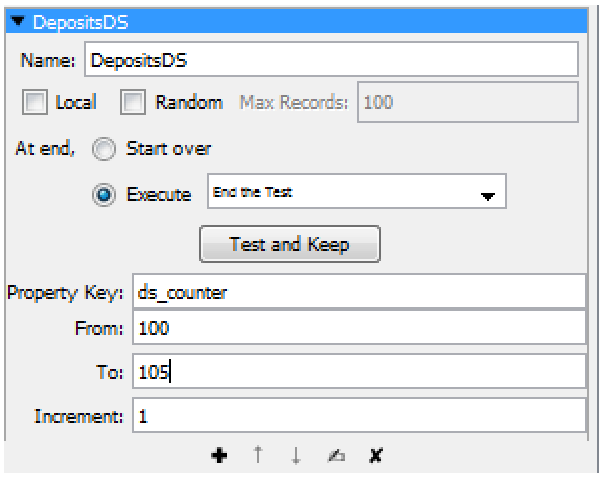
Step 2 - Add a Data Set

In the following procedure, you add a numeric counting data set. This type of data set enables you to assign a number to a property. You can change the number by a fixed value each time the data set is used.

Follow these steps:

1. In the model editor, select the first test step.
2. In the right pane, double-click the Data Sets step tab.
3. Click ✚ Add below the Data Sets element.
4. From the Common Datasets list, select Create a Numeric Counting Data Set.

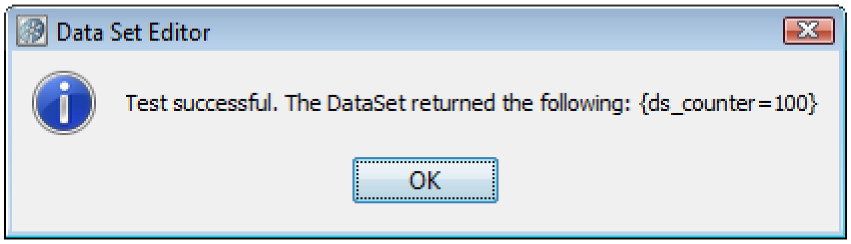
   The data set editor opens in the right pane.
5. Enter the following values:
   a. In the Name field, enter DepositsDS.
   b. In the At end field, select the Execute option and select End the Test from the list.
   c. In the Property Key field, enter ds_counter.
   d. In the From field, enter 100.
   e. In the To field, enter 105.
   f. In the Increment field, enter 1.


DepositsDS

Name:	DepositsDS
Local	☐
Random	☐
Max Records:	100
At end,	Start over [✓ Execute] End the Test
Property Key:	ds_counter
From:	100
To:	105
Increment:	1

6. Click the Test and Keep button to test the data set.

   You see a success message that shows the first row of data in the data set:



Test successful. The DataSet returned the following: {ds_counter=100}

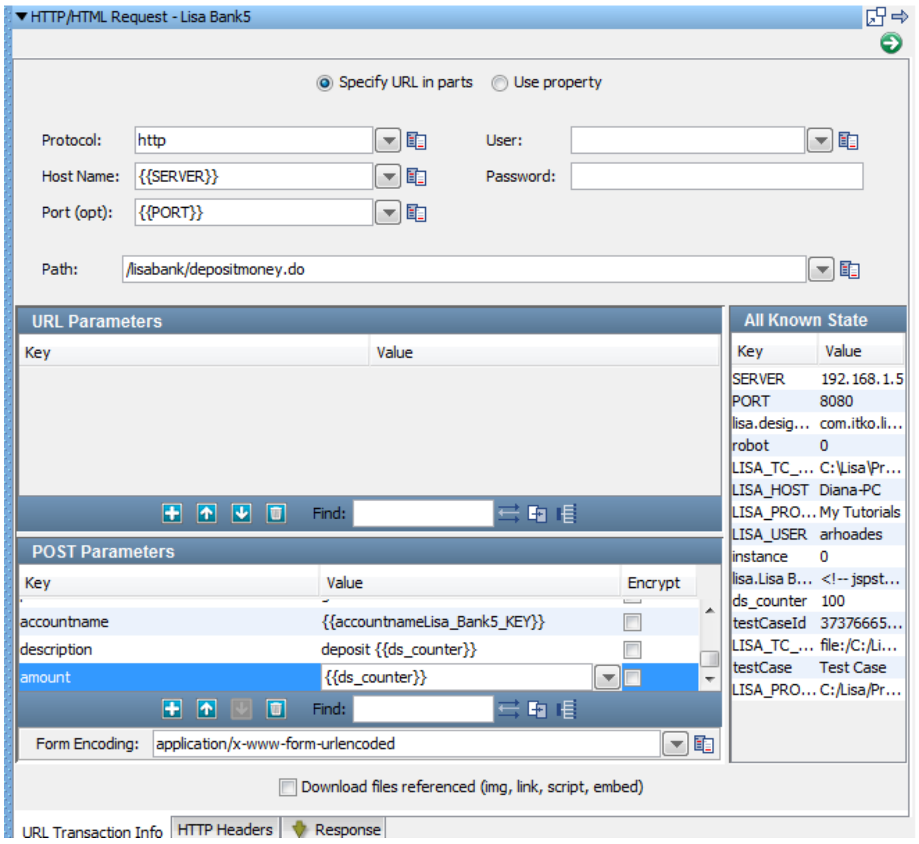
7. Click OK.

Step 3 - Modify the POST Parameters for the Recorded Deposit

You now use the ds_counter property (which you created in the data set) to specify varying amounts of money for the deposit.

Follow these steps:

1. In the model editor, double-click the LISA Bank5 step.
2. In the POST Parameters area, change the value of the description key to **deposit {{ds_counter}}**.
3. Change the value of the amount key to **{{ds_counter}}**.



(● Specify URL in parts) ○ Use property

Protocol: http
Host Name: {{SERVER}}
Port (opt): {{PORT}}
User:
Password:
Path: /lisabank/depositmoney.do

Key	Value	Encrypt
accountname	{{accountnameLisa_Bank5_KEY}}	☐
description	deposit {{ds_counter}}	☐
amount	{{ds_counter}}	☐

Form Encoding: application/x-www-form-urlencoded

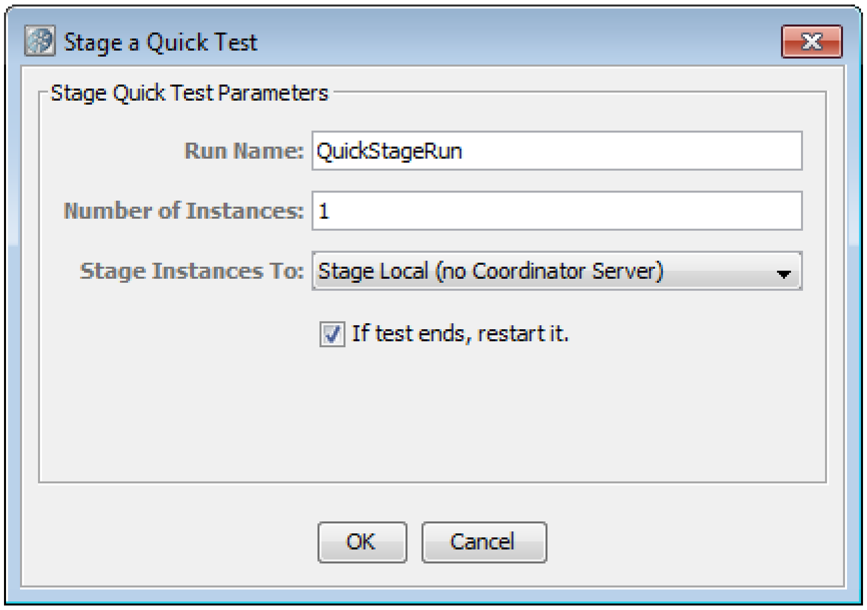
Key	Value
SERVER	192.168.1.5
PORT	8080
lisa.desig…	com.itko.li…
robot	0
LISA_TC_…	C:\Lisa\Pr…
LISA_HOST	Diana-PC
LISA_PRO…	My Tutorials
LISA_USER	arhoades
instance	0
lisa.Lisa B…	<!-- jspst…
ds_counter	100
testCaseId	37376665…
LISA_TC_…	file:/C:/Li…
testCase	Test Case
LISA_PRO…	C:/Lisa/Pr…

4. Save the test case.

Step 4 - Stage the Test Case

To stage (or run) a Quick Test:

1. From the toolbar, click Stage a quick test.
2. In the Stage a Quick Test window, ensure that **If test ends, restart it** is selected.



Run Name: QuickStageRun

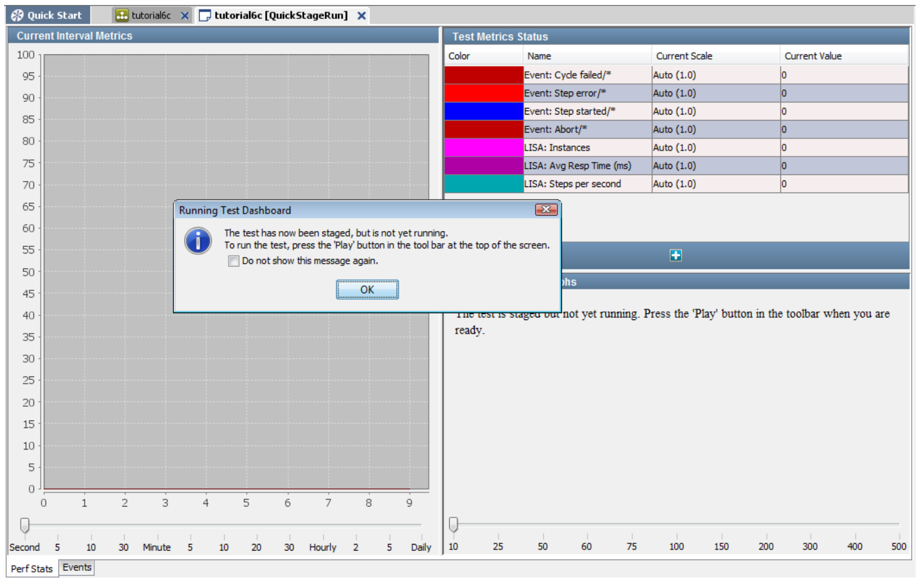
Number of Instances: 1

Stage Instances To: Stage Local (no Coordinator Server)

☑ If test ends, restart it.

3. Click OK.

   The Test Monitor opens, but the test has not been started yet.



Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	0
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	0
🟪	LISA: Avg Resp Time (ms)	Auto (1.0)	0
🟦	LISA: Steps per second	Auto (1.0)	0

The test has now been staged, but is not yet running.
To run the test, press the 'Play' button in the tool bar at the top of the screen.

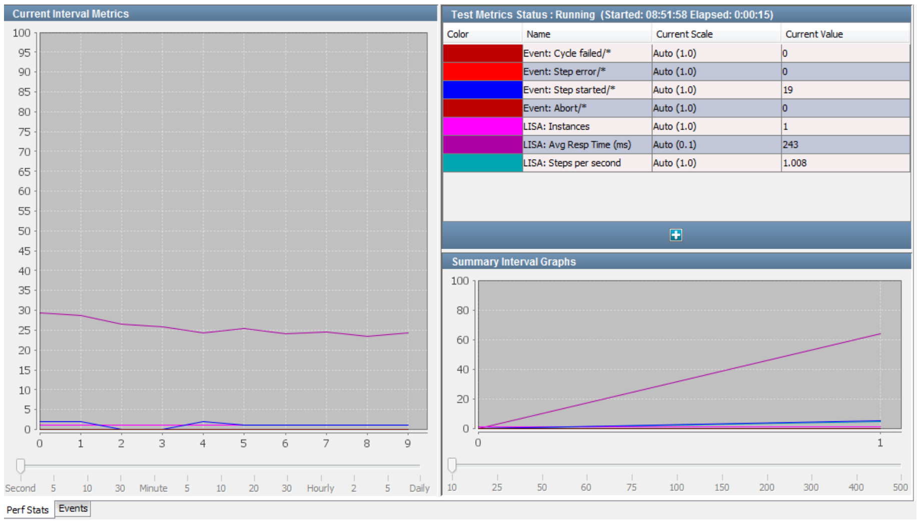
☐ Do not show this message again.

The test is staged but not yet running. Press the 'Play' button in the toolbar when you are ready.

4. Click OK.

5. From the toolbar, click Play.

   The line graphs show the progress of the test.



Test Metrics Status : Running (Started: 08:51:58 Elapsed: 0:00:15)

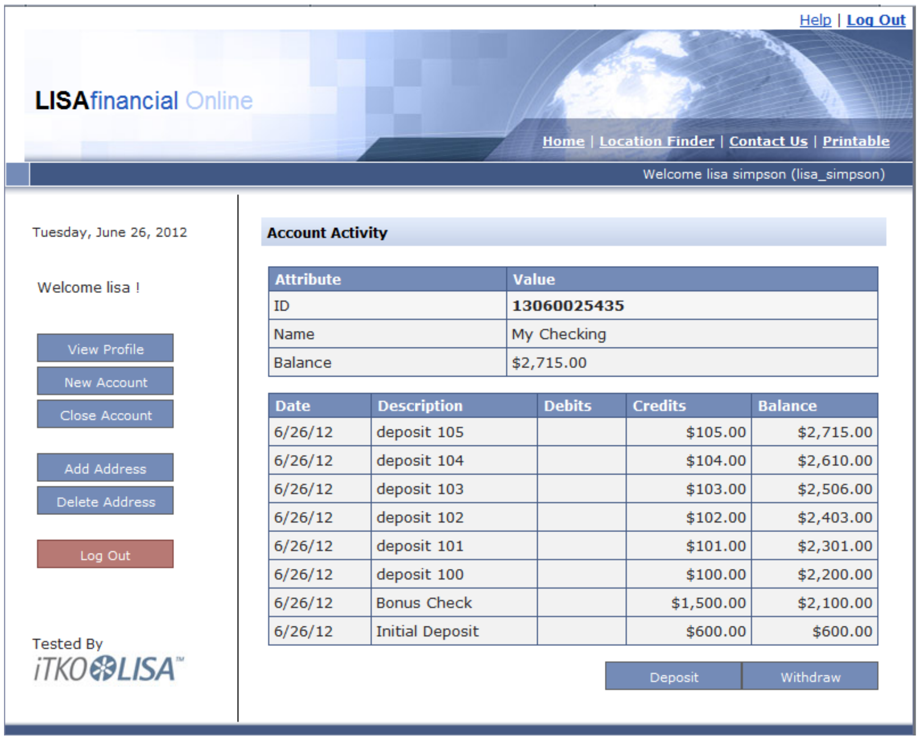
Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	19
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	1
🟪	LISA: Avg Resp Time (ms)	Auto (0.1)	243
🟦	LISA: Steps per second	Auto (1.0)	1.008

Step 5 - View the New Deposits in LISA Bank

Follow these steps:

1. Log in to the LISA Bank application again with the user lisa_simpson and password *golisa*.
2. To view the deposits, click the account number link for the checking account.

   Notice how the deposits start with 100 and increase by 1 until the amount 105 is reached.



Welcome lisa simpson (lisa_simpson)

Tuesday, June 26, 2012

Welcome lisa !

Account Activity

Attribute	Value
ID	13060025435
Name	My Checking
Balance	$2,715.00

Date	Description	Debits	Credits	Balance
6/26/12	deposit 105		$105.00	$2,715.00
6/26/12	deposit 104		$104.00	$2,610.00
6/26/12	deposit 103		$103.00	$2,506.00
6/26/12	deposit 102		$102.00	$2,403.00
6/26/12	deposit 101		$101.00	$2,301.00
6/26/12	deposit 100		$100.00	$2,200.00
6/26/12	Bonus Check		$1,500.00	$2,100.00
6/26/12	Initial Deposit		$600.00	$600.00

Tutorial 6 - Review

In this tutorial, you used a numeric counting data set to provide input to the recorded test.

You:

- Copied a test case and added a numeric counting data set.
- Modified the POST Parameters for the recorded deposit.
- Staged a quick test.

Tutorial 7 - Test an Enterprise JavaBean (EJB)

The LISA Bank application provides a full set of EJBs to interact with an account, get the user and account information from the Java interface.

This tutorial uses the Enterprise JavaBean Execution test step to call EJB methods in a test case and test the response with an assertion. You test a simple EJB to verify that the addUser and deleteUser methods work as expected.

Tutorial Tasks

In this tutorial, you:

- Use the Enterprise Java Bean Execution step.
- Use the Complex Object Editor with EJB objects.

Prerequisites

- You have completed Tutorial 6 - Test a Website.
- DevTest Workstation is open.
- You have access to the demo server.

Step 1 - Create a Test Case

Follow these steps:

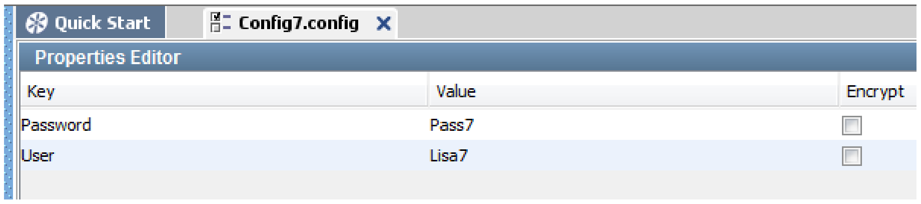
1. In the Project pane, right-click the Tests folder and select Create New Test Case.
2. Set the file name to **tutorial7**.
3. Click Save.

Step 2 - Create a Configuration

You previously worked with configurations in Tutorial 2 - Data Sets.

Follow these steps:

1. Open the project.config file.
2. If the configuration does not contain the User and Password properties, add these properties. You do not need to set the values.
3. Create a configuration with the name **config7**.
4. Add the User property to the config7 configuration and set the value to **Lisa7**.
5. Add the Password property to the config7 configuration and set the value to **Pass7**.



Key	Value	Encrypt
Password	Pass7	☐
User	Lisa7	☐

6. Click Save.
7. In the Project pane, right-click the config7 configuration and select Make Active.

   The configuration now appears in purple.

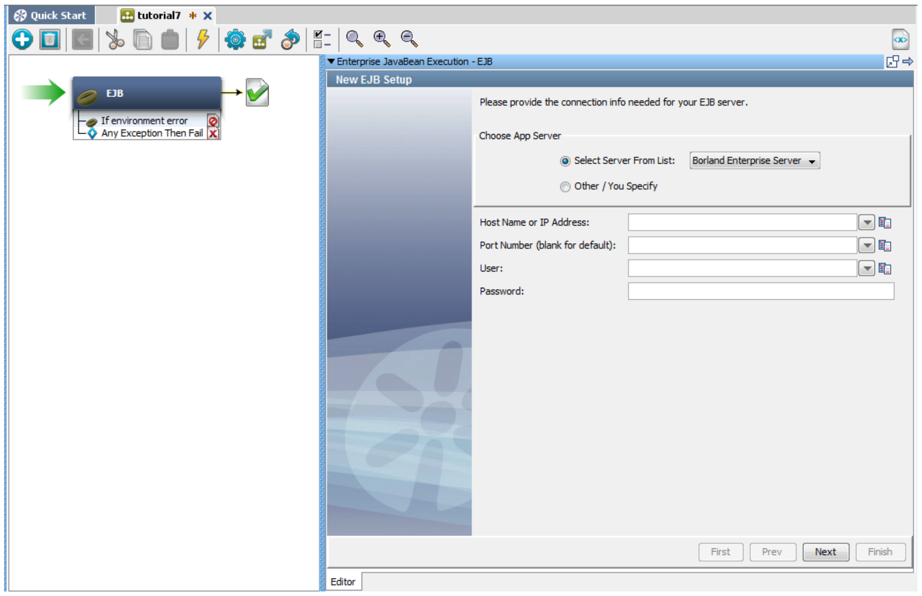
Step 3 - Add an EJB Test Step

The Enterprise JavaBean Execution test step enables you to make calls on a running EJB.

Follow these steps:

1. Click the tutorial7 tab.
2. Click Add Step.
3. Select Java/J2EE and select Enterprise JavaBean Execution.

   The New EJB Setup wizard appears.



Please provide the connection info needed for your EJB server.

Choose App Server

(●) Select Server From List: Borland Enterprise Server

( ) Other / You Specify

Host Name or IP Address:

Port Number (blank for default):

User:

Password:

Step 4 - Connect to the Server

The New EJB Setup wizard prompts you to specify the connection information for the EJB server.

Follow these steps:

1. From the Select Server From List drop-down list, select JBoss 3.2/4.0.
2. In the Host Name or IP Address field, enter **localhost**.
3. Click Next.

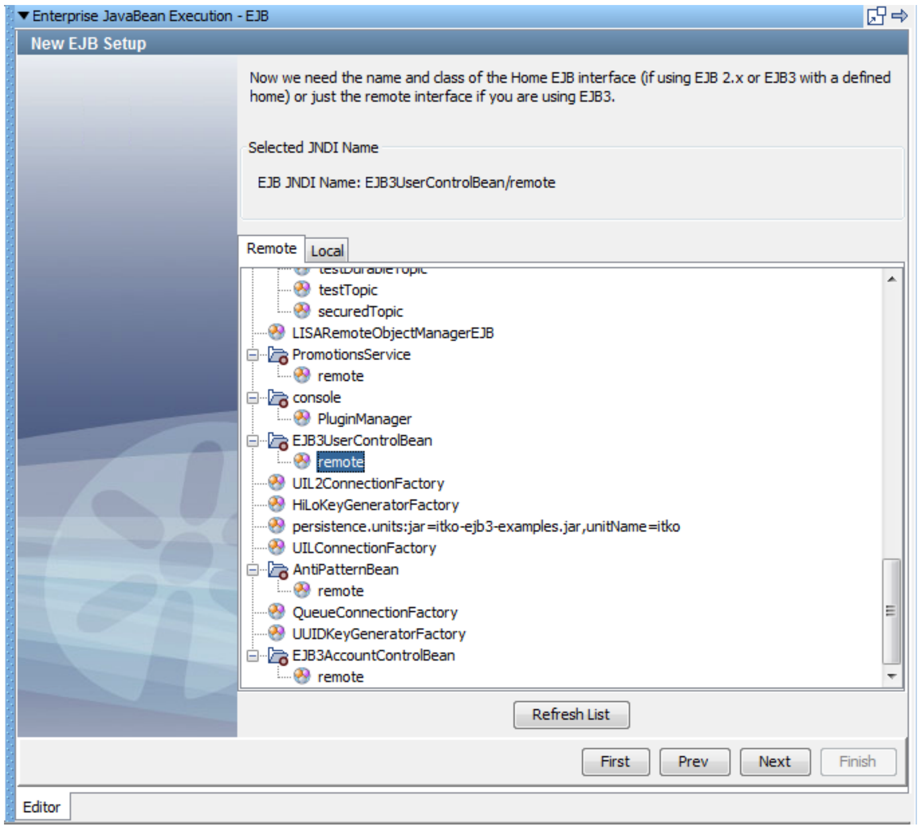
   The list of JNDI names is retrieved from the EJB server.

Step 5 - Locate the EJB Interface

The New EJB Setup wizard prompts you to specify the name of the EJB interface.

Follow these steps:

1. In the Remote tab, select EJB3UserControlBean/remote.



Enterprise JavaBean Execution - EJB

New EJB Setup

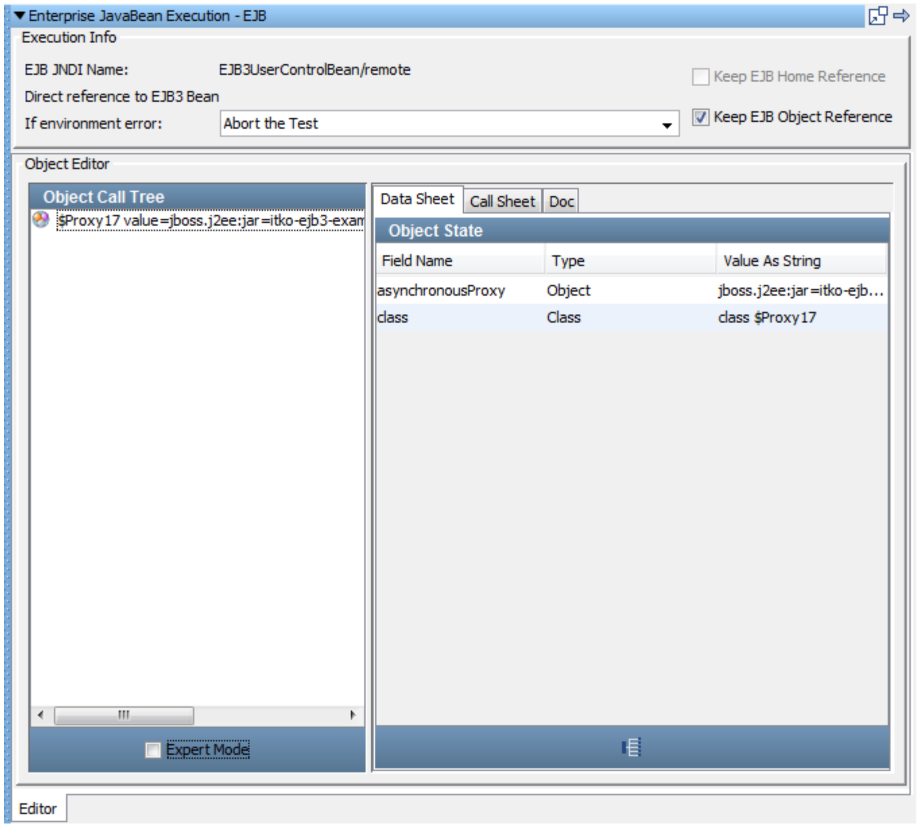
Now we need the name and class of the Home EJB interface (if using EJB 2.x or EJB3 with a defined home) or just the remote interface if you are using EJB3.

Selected JNDI Name

EJB JNDI Name: EJB3UserControlBean/remote

2. Click Next.

   The Complex Object Editor opens.



EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

Keep EJB Home Reference

[✓ Keep EJB Object Reference]

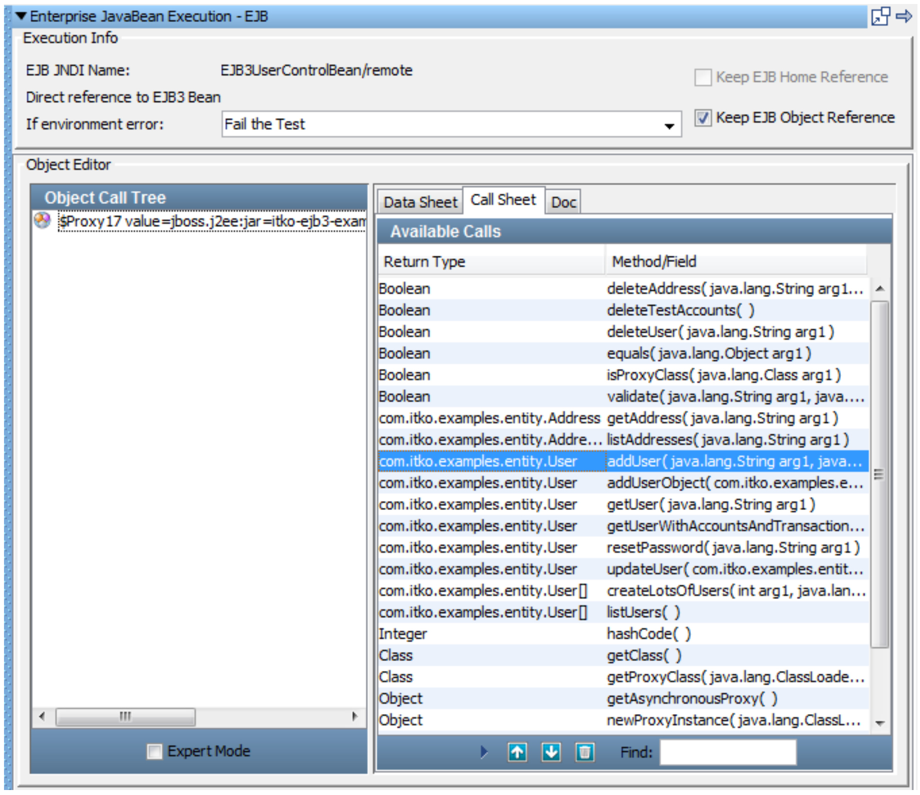
$Proxy17 value=jboss.j2ee:jar=itko-ejb3-exam

Field Name	Type	Value As String
asynchronousProxy	Object	jboss.j2ee:jar=itko-ejb...
class	Class	class $Proxy17

Step 6 - Configure the EJB

Follow these steps:

1. If you use the same EJB object repeatedly, and the Keep EJB Object Reference check box is not already selected, select the check box.
2. Set the If environment error field to the step to execute if an exception occurs while executing this EJB step. Select Fail the Test from the list.
3. In the Object Editor area, select the Call Sheet tab and select the addUser method.
4. Click Add selected method to Object Call Tree.



EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

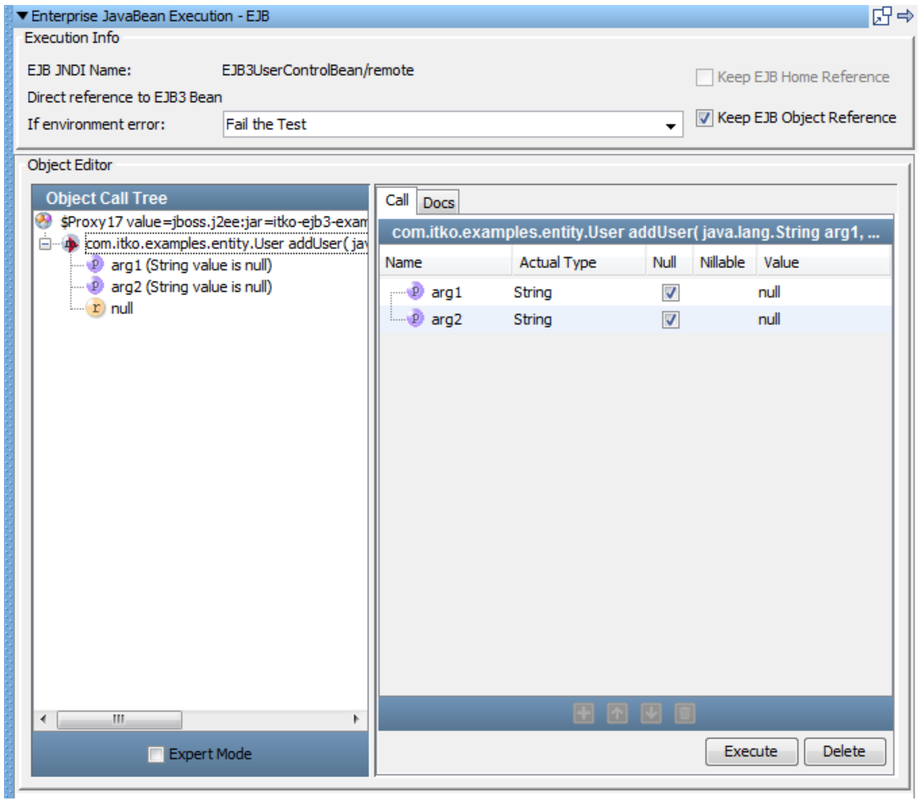
☐ Keep EJB Home Reference

☑ Keep EJB Object Reference

$Proxy17 value=jboss.j2ee:jar=itko-ejb3-exam

Return Type	Method/Field
Boolean	deleteAddress(java.lang.String arg1...
Boolean	deleteTestAccounts( )
Boolean	deleteUser(java.lang.String arg1)
Boolean	equals(java.lang.Object arg1)
Boolean	isProxyClass(java.lang.Class arg1)
Boolean	validate(java.lang.String arg1, java....
com.itko.examples.entity.Address	getAddress(java.lang.String arg1)
com.itko.examples.entity.Addre...	listAddresses(java.lang.String arg1)
com.itko.examples.entity.User	addUser(java.lang.String arg1, java...
com.itko.examples.entity.User	addUserObject(com.itko.examples.e...
com.itko.examples.entity.User	getUser(java.lang.String arg1)
com.itko.examples.entity.User	getUserWithAccountsAndTransaction...
com.itko.examples.entity.User	resetPassword(java.lang.String arg1)
com.itko.examples.entity.User	updateUser(com.itko.examples.entit...
com.itko.examples.entity.User[]	createLotsOfUsers(int arg1, java.lan...
com.itko.examples.entity.User[]	listUsers( )
Integer	hashCode( )
Class	getClass( )
Class	getProxyClass(java.lang.ClassLoade...
Object	getAsynchronousProxy( )
Object	newProxyInstance(java.lang.ClassL...

The Object Call Tree now displays the addUser method.



EJB JNDI Name: EJB3UserControlBean/remote
Direct reference to EJB3 Bean
If environment error: Fail the Test
[ ] Keep EJB Home Reference
[✓] Keep EJB Object Reference

$Proxy17 value=jboss.j2ee:jar=itko-ejb3-exam
com.itko.examples.entity.User addUser(ja
arg1 (String value is null)
arg2 (String value is null)
null

com.itko.examples.entity.User addUser(java.lang.String arg1, ...

Name	Actual Type	Null	Nillable	Value
arg1	String	✓		null
arg2	String	✓		null

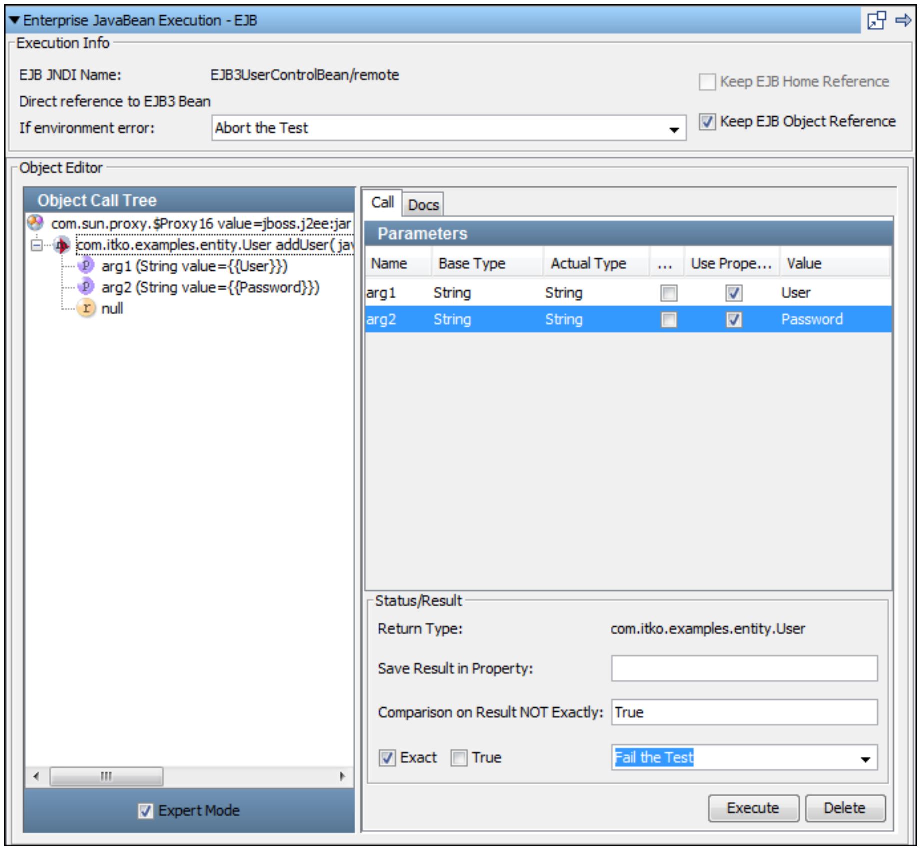
Important! You can only add User and Password once. To execute this tutorial more than once, change the values that are associated with User and Password.

Step 7 - Add an Assertion

To enter the method parameters and add an inline assertion:

1. In the Object Call Tree pane, select Expert Mode if it is not already enabled.
2. Select the Use Property check boxes for each argument.

   Use Property is a column heading in the Parameters area.
3. In the Value column for arg1, select **User** from the Defaults list of properties.
4. In the Value column for arg2, select **Password** from the Defaults list of properties.
5. In the Status/Result area, add the inline assertion by selecting Exact and clearing the True check box.
6. In the Comparison on Result NOT Exactly field, enter **True**.
7. From the Exact drop-down, select Fail the Test.
8. Click Execute.


Enterprise JavaBean Execution - EJB

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

☐ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

com.sun.proxy.$Proxy16 value=jboss.j2ee:jar

com.itko.examples.entity.User addUser(ja

arg1 (String value={{User}})

arg2 (String value={{Password}})

null

☑ Expert Mode

Call | Docs

Parameters

Name	Base Type	Actual Type	...	Use Prope...	Value
arg1	String	String	☐	☑	User
arg2	String	String	☐	☑	Password

Status/Result

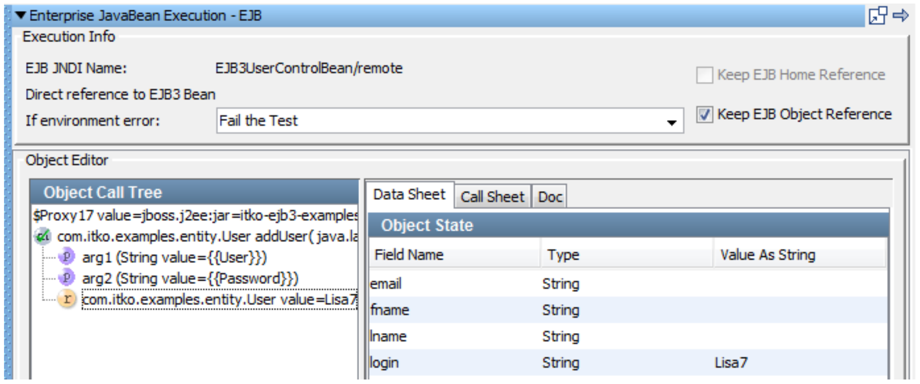
Return Type: com.itko.examples.entity.User

Save Result in Property:

Comparison on Result NOT Exactly: True

☑ Exact ☐ True Fail the Test

The parameters to the method are displayed in the Object Call Tree, next to the Input Parameter icon. The return value of this method is the value of the User field, "Lisa7," in the Object Call Tree.



Enterprise JavaBean Execution - EJB

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

☐ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

```
$Proxy17 value=jboss.j2ee:jar=itko-ejb3-examples
com.itko.examples.entity.User addUser( java.la
   arg1 (String value={{User}})
   arg2 (String value={{Password}})
   com.itko.examples.entity.User value=Lisa7
```

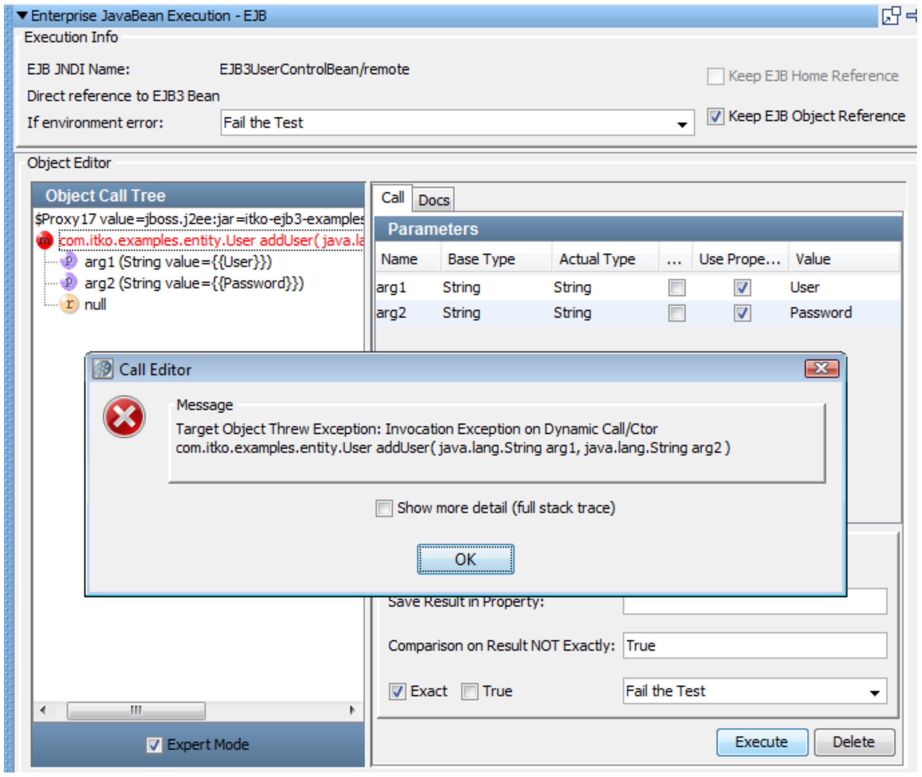
Data Sheet | Call Sheet | Doc

Object State

Field Name	Type	Value As String
email	String	
fname	String	
lname	String	
login	String	Lisa7

9. Test the addUser method again by clicking Execute.

10. The return r changes to null and an error is produced because the user has already been added.



Enterprise JavaBean Execution - EJB

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

☐ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

```
$Proxy17 value=jboss.j2ee:jar=itko-ejb3-examples
com.itko.examples.entity.User addUser( java.la
   arg1 (String value={{User}})
   arg2 (String value={{Password}})
   null
```

Call | Docs

Parameters

Name	Base Type	Actual Type	...	Use Prope...	Value
arg1	String	String	☐	☑	User
arg2	String	String	☐	☑	Password

Call Editor

Message

```
Target Object Threw Exception: Invocation Exception on Dynamic Call/Ctor
com.itko.examples.entity.User addUser( java.lang.String arg1, java.lang.String arg2 )
```

☐ Show more detail (full stack trace)

OK

Save Result in Property:

Comparison on Result NOT Exactly: True

☑ Exact ☐ True Fail the Test

☑ Expert Mode

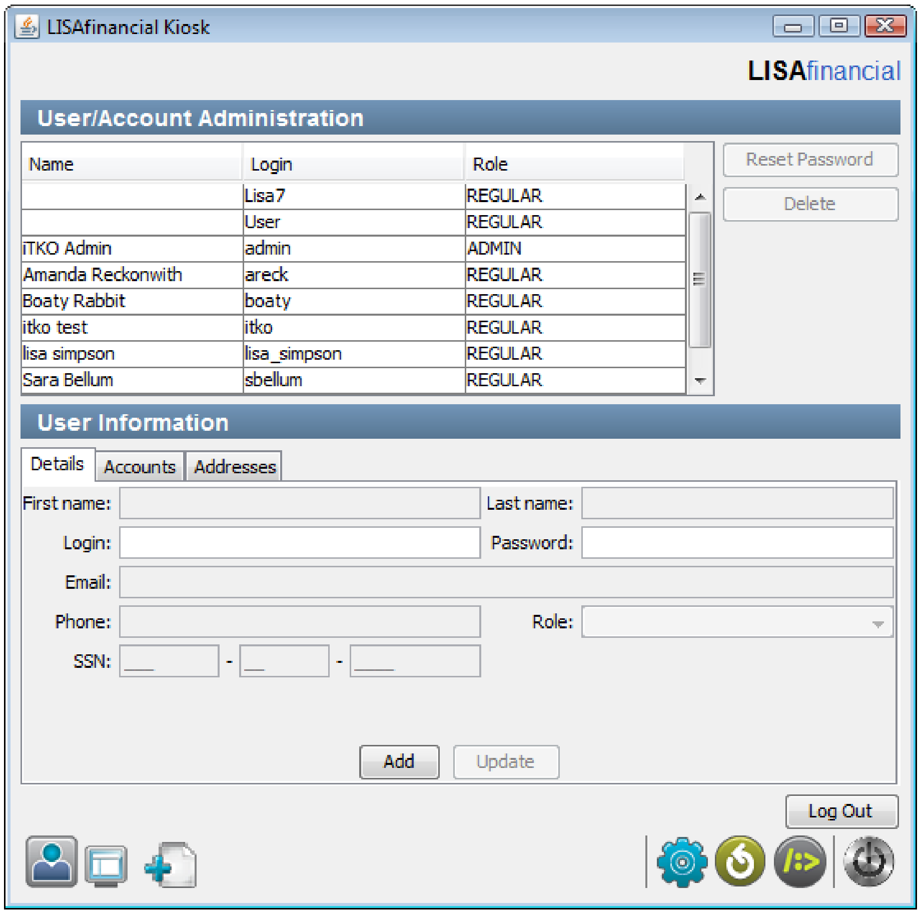
Execute Delete

Step 8 - Verify the Method Execution

From the LISA Bank application, you can verify that the user was added.

Follow these steps:

1. Go to the LISA Bank application.
2. Log in as user **admin** with the password **admin**.
3. To confirm that Lisa7 was added, view the list of users.



Name	Login	Role
	Lisa7	REGULAR
	User	REGULAR
iTKO Admin	admin	ADMIN
Amanda Reckonwith	areck	REGULAR
Boaty Rabbit	boaty	REGULAR
itko test	itko	REGULAR
lisa simpson	lisa_simpson	REGULAR
Sara Bellum	sbellum	REGULAR

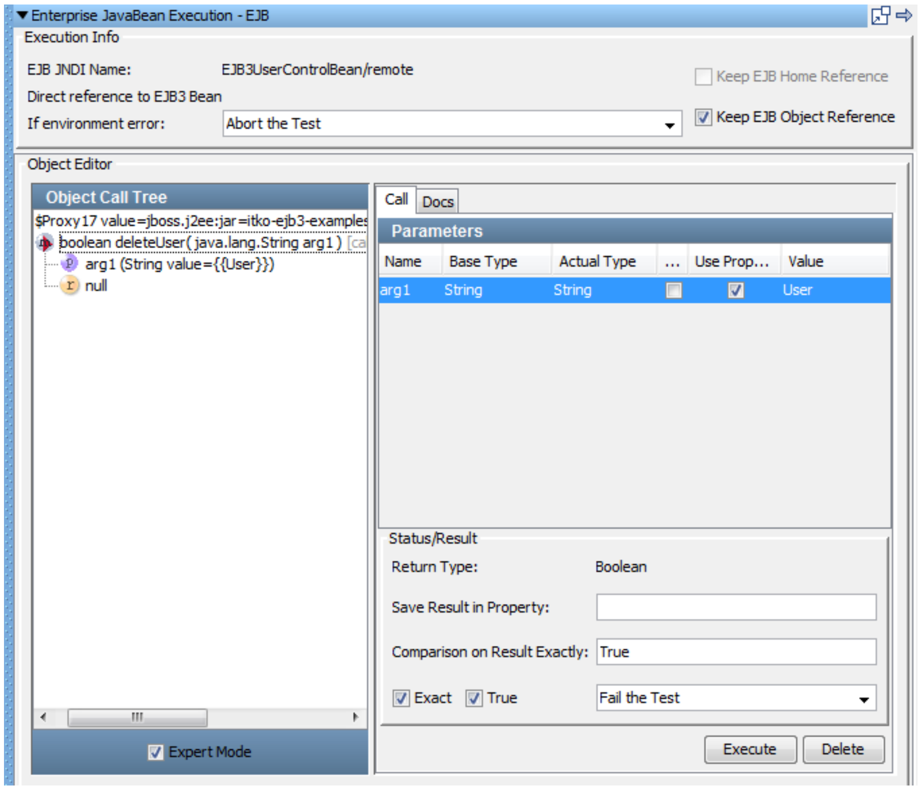
First name:
Last name:
Login:
Password:
Email:
Phone:
Role:
SSN: ___ - __ - ____

Step 9 - Add Another EJB Test Step

Now try the preceding steps again to invoke the deleteUser method.

Follow these steps:

1. Repeat the tutorial beginning with Step 3 to add an EJB step with the name **DeleteUser**.
2. Use the method parameter property User.



Enterprise JavaBean Execution - EJB

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

Keep EJB Home Reference: unchecked

Keep EJB Object Reference: checked

Object Editor

Object Call Tree

```
$Proxy17 value=jboss.j2ee:jar=itko-ejb3-examples
boolean deleteUser( java.lang.String arg1 )
    arg1 (String value={{User}})
    null
```

Parameters

Name	Base Type	Actual Type	...	Use Prop...	Value
arg1	String	String	☐	✓	User

Status/Result

Return Type: Boolean

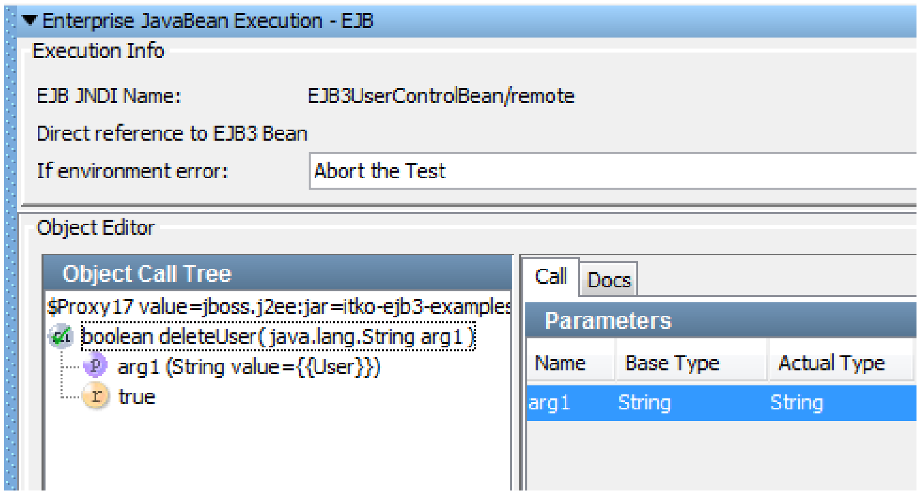
Save Result in Property:

Comparison on Result Exactly: True

Exact: checked, True: checked, Fail the Test

Expert Mode: checked

3. Click Execute to execute this method and get results.



EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

```
$Proxy17 value=jboss.j2ee:jar=itko-ejb3-examples
boolean deleteUser( java.lang.String arg1 )
    arg1 (String value={{User}})
    true
```

Name	Base Type	Actual Type
arg1	String	String

The return r is true, indicating the user has been deleted.

4. Click Save.

Tutorial 7 - Review

In this tutorial, you:

- Created a test case consisting of two EJB test steps.

  The EJB object was loaded from the example application on the demo server.
- Created an EJB test step and loaded an EJB.
- Used the Complex Object Editor to manipulate EJB objects.

Tutorial 8 - Test a Web Service

In this tutorial, you use the Web Service Execution (XML) test step to call web service operations in a test case. You then test the request and response. These web service operations provide the same functionality as the equivalent method calls in the EJB used in Tutorial 7.

Tutorial Tasks

In this tutorial, you:

- Add the Web Service Execution (XML) test step.
- Execute a web service operation.

Prerequisites

- You have completed Tutorial 5.
- DevTest Workstation is open.
- You have access to the demo server.

Step 1 - Create a Test Case

Follow these steps:

1. Right-click on the Tests folder in the Project panel, and select Create New Test Case.
2. Set the file name to **tutorial8**.
3. Click Save.
4. In the Project panel, right-click on project.config and select Make active.

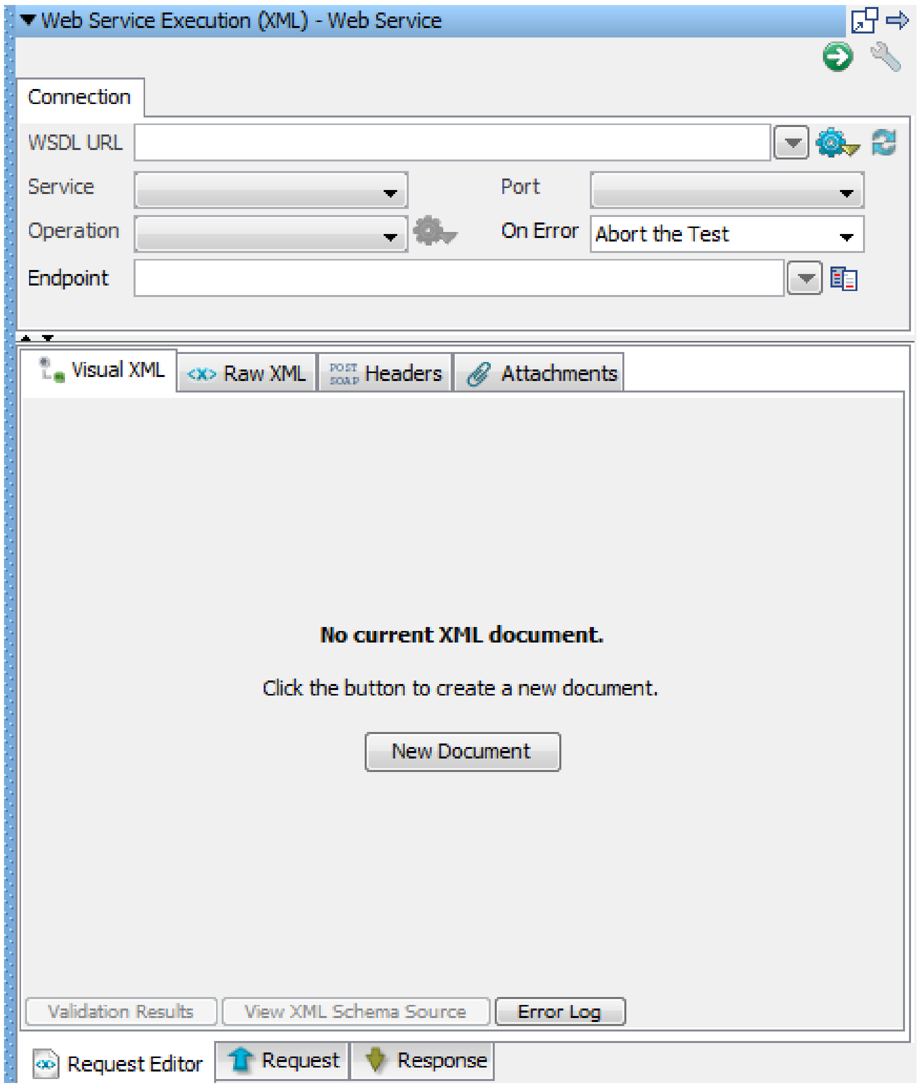
Step 2 - Add a Web Service Execution (XML) Test Step

The Web Service Execution (XML) test step enables you to execute an operation on a SOAP-based web service.

Follow these steps:

1. Click the tutorial8a tab.
2. Click Add Step.
3. Select Web/Web Services and select Web Service Execution (XML).

   A Web Service step is added to the model editor.
4. To open the Web Service Execution (XML) editor, double-click the Web Service step.



5. Click New Document.

Step 3 - Create a Web Service Client

Now specify the operation to be called, and create a SOAP message to send to the operation.

Follow these steps:

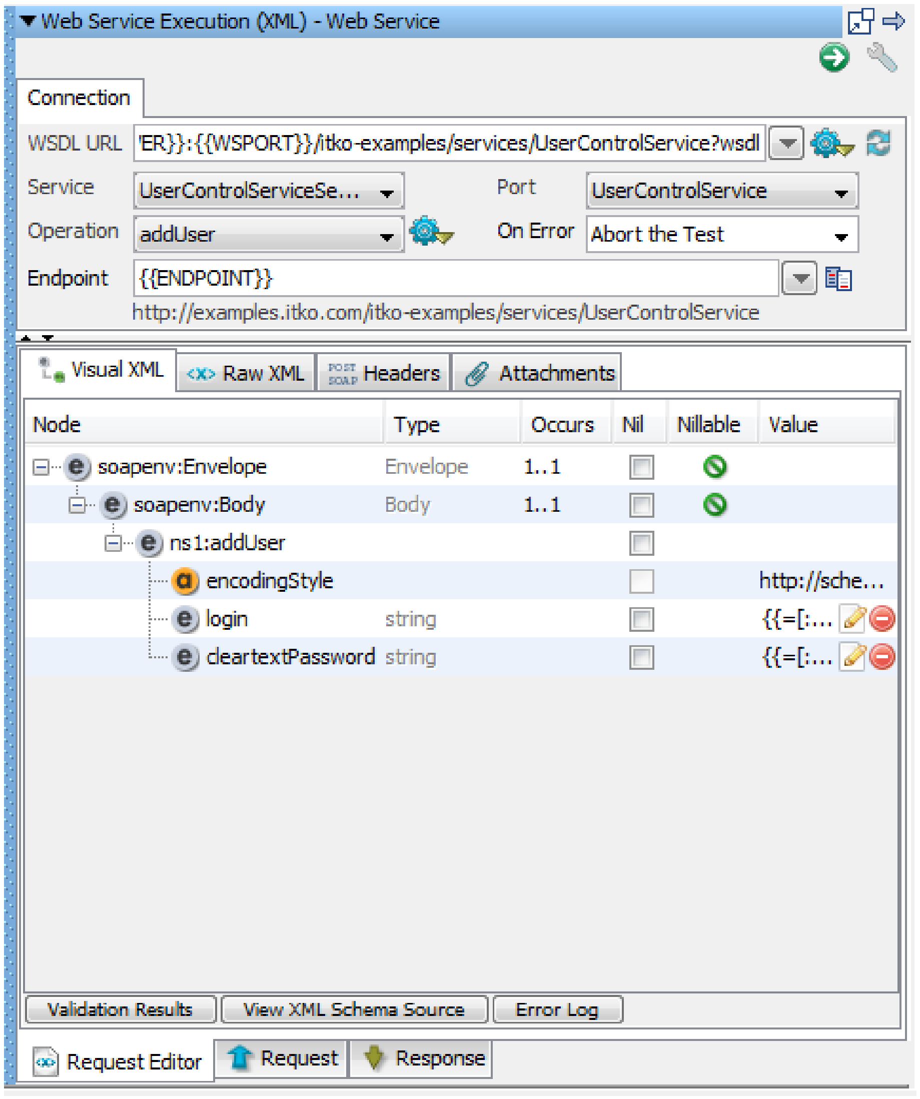
1. In the WSDL URL field, enter the following location.

   Note: The WSSERVER and WSPORT properties represent the server and port.

   ```
   http://localhost:8080/itko-examples/services/UserControlService
   ?wsdl
   ```

2. In the Service field, select UserControlServiceService.
3. In the Port field, select UserControlService.
4. In the Operation field, select the addUser operation.
5. In the On Error field, select Abort the Test.

   DevTest uses this criteria to build the web service client. The Visual XML editor shows a graphical view of the SOAP message.



▼ Web Service Execution (XML) - Web Service

Connection

WSDL URL 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl

Service UserControlServiceSe... Port UserControlService

Operation addUser On Error Abort the Test

Endpoint {{ENDPOINT}}

http://examples.itko.com/itko-examples/services/UserControlService

Visual XML | Raw XML | Headers | Attachments

Node	Type	Occurs	Nil	Nillable	Value
soapenv:Envelope	Envelope	1..1	☐	⊘	
soapenv:Body	Body	1..1	☐	⊘	
ns1:addUser			☐		
encodingStyle			☐		http://sche...
login	string		☐		{{=[:...
cleartextPassword	string		☐		{{=[:...

6. Save the test case.

Step 4 - Execute the Web Service Request

Follow these steps:

1. Click Execute WS Request.

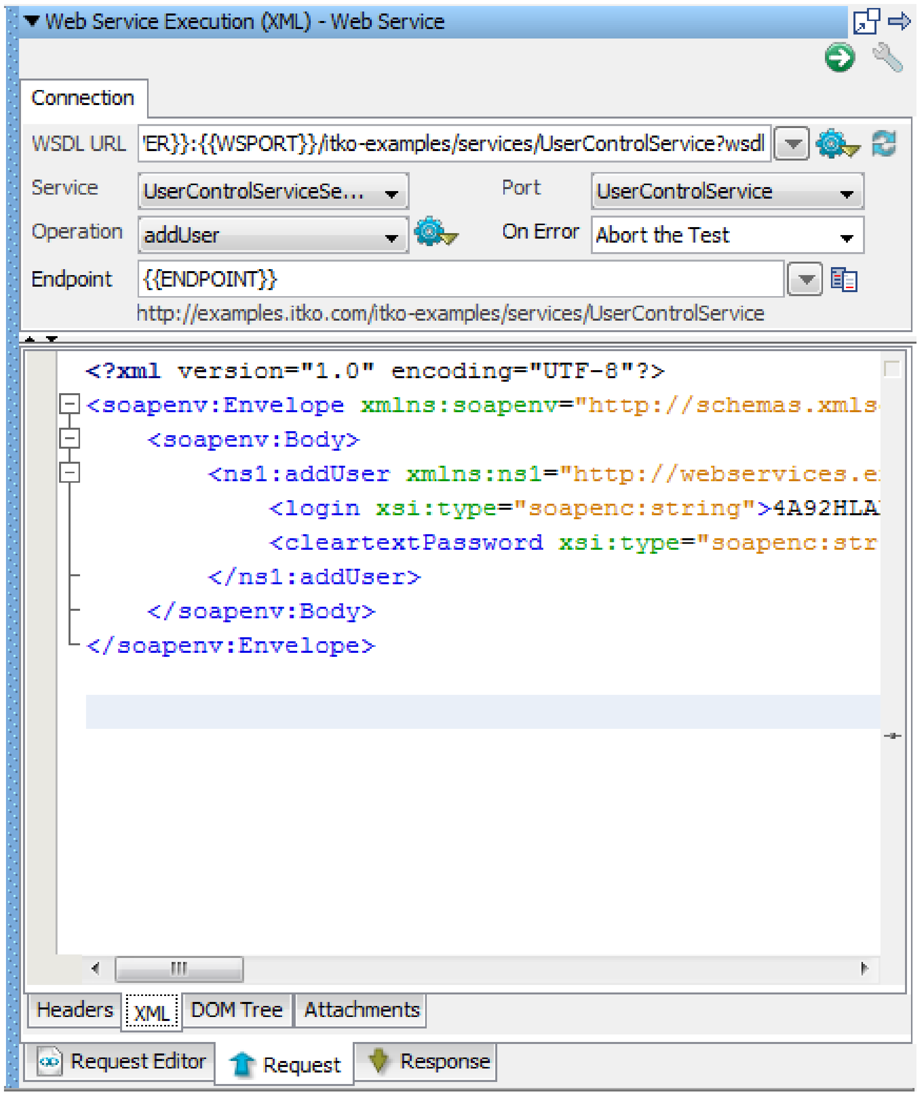
   The test is executed.

Step 5 - View the Request and Response

The Request tab shows the resulting request data that was sent after any post processing (for example, substituting DevTest properties). The Response tab shows the resulting response data that was received.

Follow these steps:

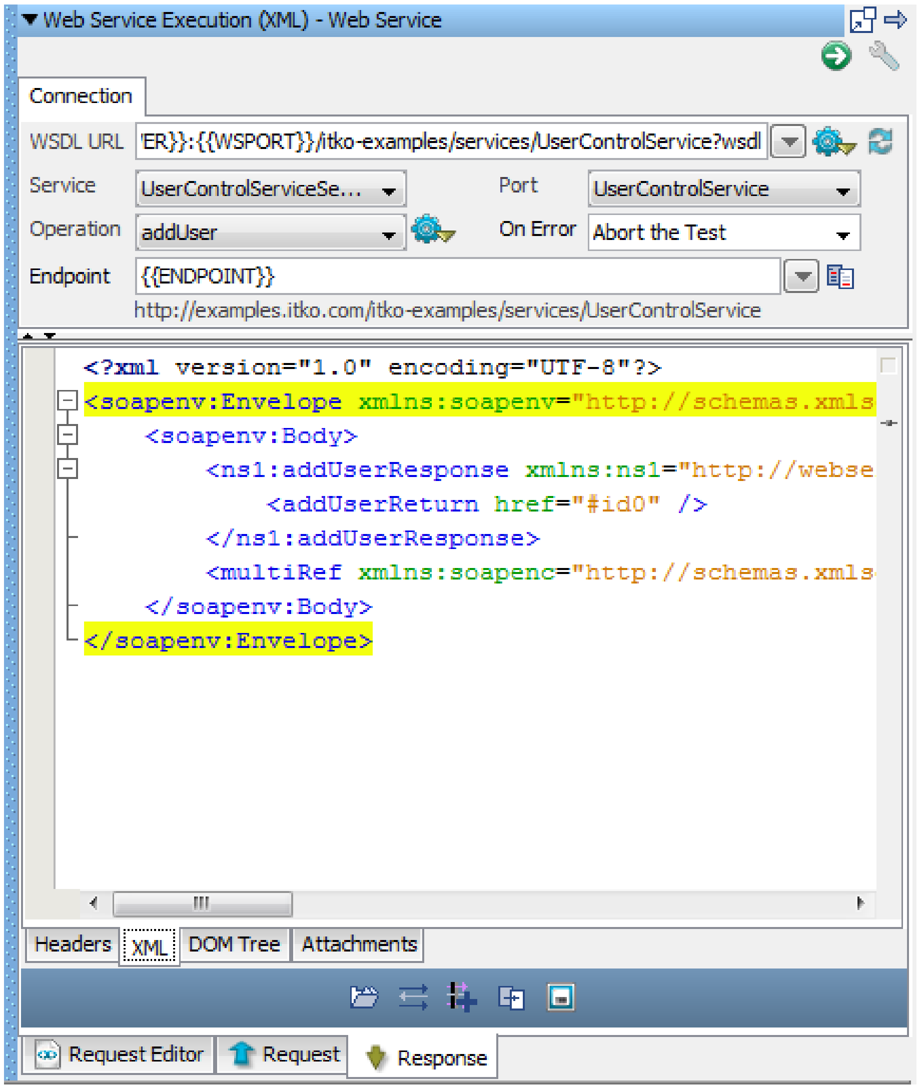
1. To view the request upon execution, click the Request tab.



WSDL URL 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl
Service UserControlServiceSe...
Port UserControlService
Operation addUser
On Error Abort the Test
Endpoint {{ENDPOINT}}
http://examples.itko.com/itko-examples/services/UserControlService

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
    <soapenv:Body>
        <ns1:addUser xmlns:ns1="http://webservices.e
            <login xsi:type="soapenc:string">4A92HLA
            <cleartextPassword xsi:type="soapenc:str
        </ns1:addUser>
    </soapenv:Body>
</soapenv:Envelope>
```

2. To view the response upon execution, select the Response tab.



WSDL URL 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl

Service UserControlServiceSe... Port UserControlService

Operation addUser On Error Abort the Test

Endpoint {{ENDPOINT}}

http://examples.itko.com/itko-examples/services/UserControlService

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
    <soapenv:Body>
        <ns1:addUserResponse xmlns:ns1="http://webse
            <addUserReturn href="#id0" />
        </ns1:addUserResponse>
        <multiRef xmlns:soapenc="http://schemas.xmls
    </soapenv:Body>
</soapenv:Envelope>
```

Tutorial 8 - Review

In this tutorial, you:

- Created a test case with the Web Service Execution (XML) test step.
- Executed the addUser operation.
- Viewed the request and response for this operation.

Tutorial 9 - Examine and Test a Database

In this tutorial, you examine and test a database table that is part of the web application in Tutorial 5.

You use the SQL Database Execution (JDBC) step to interact with a database in a test case and test the response with an assertion. You examine the Users table from a Derby database that is part of the application.

Tutorial Tasks

In this tutorial, you:

- Use the SQL Database Execution (JDBC) step.
- Store application properties in a configuration.
- Add and modify an assertion.
- Add a filter.

Prerequisites

- You have completed Tutorial 5.
- DevTest Workstation is open.
- You have access to the demo server.

Step 1 - Create a Test Case

Follow these steps:

1. In the Project pane, right-click on the Tests folder and select Create New Test Case.
2. Set the file name to **tutorial9**.
3. Click Save.

Step 2 - Add Database Properties to the Configuration

Store the properties that are necessary to connect to the database in the configuration. This practice is a standard DevTest practice that increases the portability of test cases.

Follow these steps:

1. If project.config is not the active configuration, then right-click project.config in the Project pane and select Make Active.
2. Open the project.config configuration.
3. Add the following properties:

   DBDriver

   org.apache.derby.jdbc.ClientDriver

   DBConnect

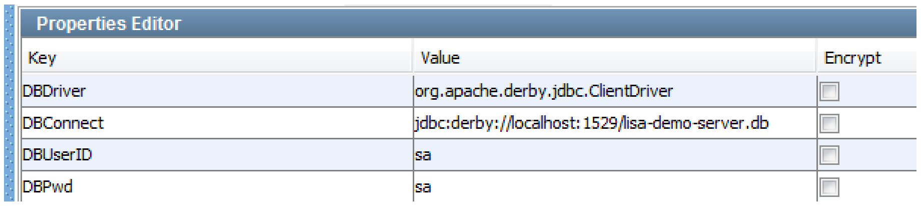
   jdbc:derby://localhost:1529/lisa-demo-server.db

   DBUserID

   sa

   DBPwd

   sa



Properties Editor

Key	Value	Encrypt
DBDriver	org.apache.derby.jdbc.ClientDriver	☐
DBConnect	jdbc:derby://localhost:1529/lisa-demo-server.db	☐
DBUserID	sa	☐
DBPwd	sa	☐

4. Click Save.

Step 3 - Add a SQL Database Execution (JDBC) Test Step

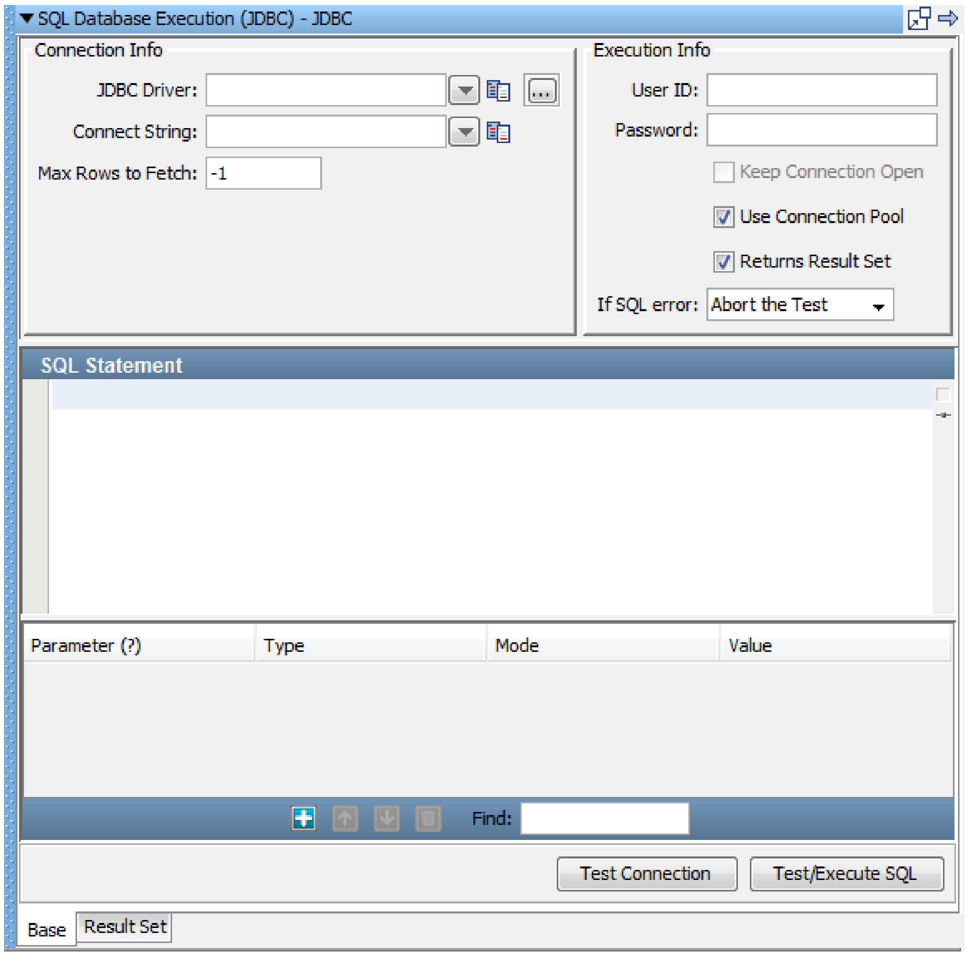
The SQL Database Execution (JDBC) test step enables you to connect to a database using JDBC and make SQL queries on the database.

Follow these steps:

1. Click the tutorial9 tab.
2. Click ➕ Add Step.
3. Select Other Transactions and select SQL Database Execution (JDBC).

   JDBC is added to the model editor.

To open the step editor, double-click the JDBC step.



SQL Database Execution (JDBC) - JDBC

Connection Info

JDBC Driver:	
Connect String:	
Max Rows to Fetch:	-1

Execution Info

User ID:	
Password:	
Keep Connection Open	☐
Use Connection Pool	✓
Returns Result Set	✓
If SQL error:	Abort the Test

SQL Statement

Parameter (?)	Type	Mode	Value

Step 4 - Connect to the Database

To provide the connection information, use the properties that you added to the project.config configuration.

Follow these steps:

1. Enter the following values in the Connection Info and Execution Info areas of the step editor. Notice that when you enter the password, the value is masked.

   JDBC Driver

   {{DBDriver}}

   Connect String

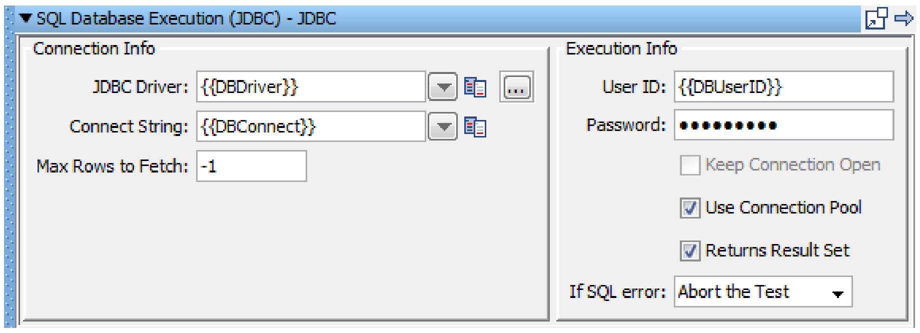
   {{DBConnect}}

   User ID

   {{DBUserID}}

   Password

   {{DBPwd}}

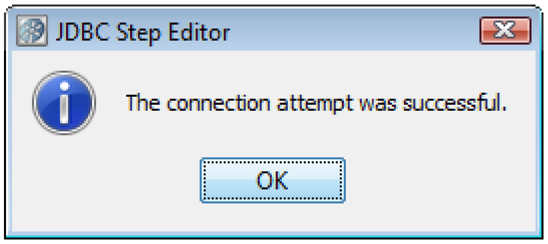


JDBC Driver: {{DBDriver}}
Connect String: {{DBConnect}}
Max Rows to Fetch: -1

User ID: {{DBUserID}}
Password: ●●●●●●●●●
Keep Connection Open
☑ Use Connection Pool
☑ Returns Result Set
If SQL error: Abort the Test

2. Click the Test Connection button at the bottom of the step editor.

   A message indicates that the connection is valid.



The connection attempt was successful.

3. Click OK.

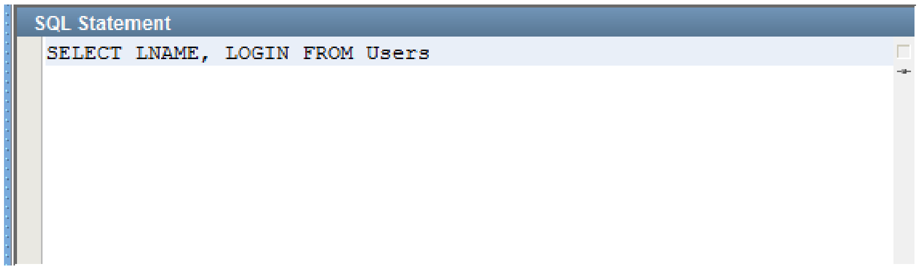
Step 5 - Execute a SQL Query

Now specify and run a SQL statement that retrieves data from the Users table.

Follow these steps:

1. In the SQL Statement pane, enter the following statement:

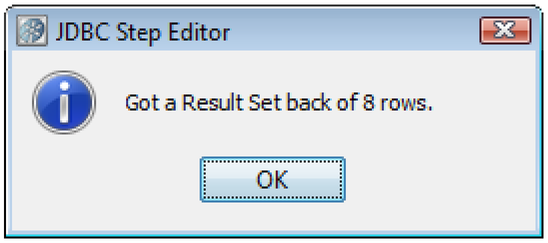
   `SELECT LNAME, LOGIN FROM Users`

   ```
   SELECT LNAME, LOGIN FROM Users
   ```

2. Click the Test/Execute SQL button at the bottom of the step editor.

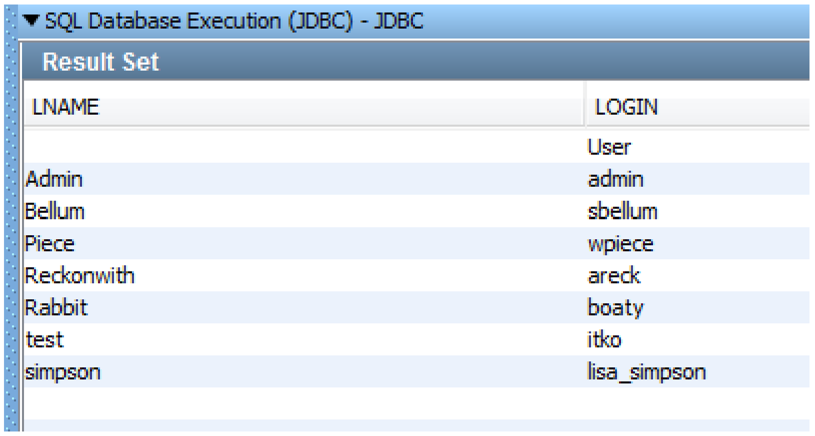
   A message confirms a valid query and displays the number of rows returned.

   Got a Result Set back of 8 rows.

3. Click OK.

   The Result Set tab is displayed.

LNAME	LOGIN
	User
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	areck
Rabbit	boaty
test	itko
simpson	lisa_simpson

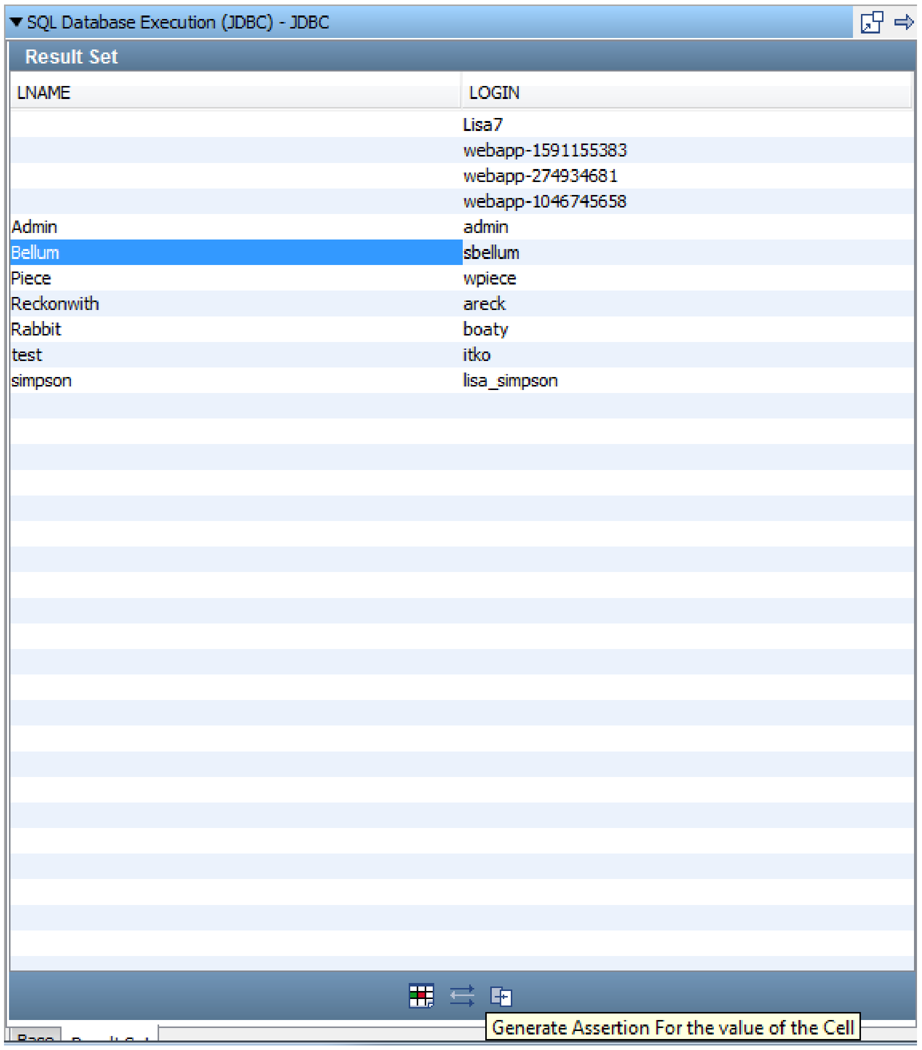
Step 6 - Add an Assertion

Add an assertion that tests for the presence of a specific last name in the result set.

Follow these steps:

1. In the Result Set tab, select a cell in the LNAME column.
2. Click Generate Assertion for the Value of a Cell.

   The Generate JDBC Result Set Value Assertion dialog opens.



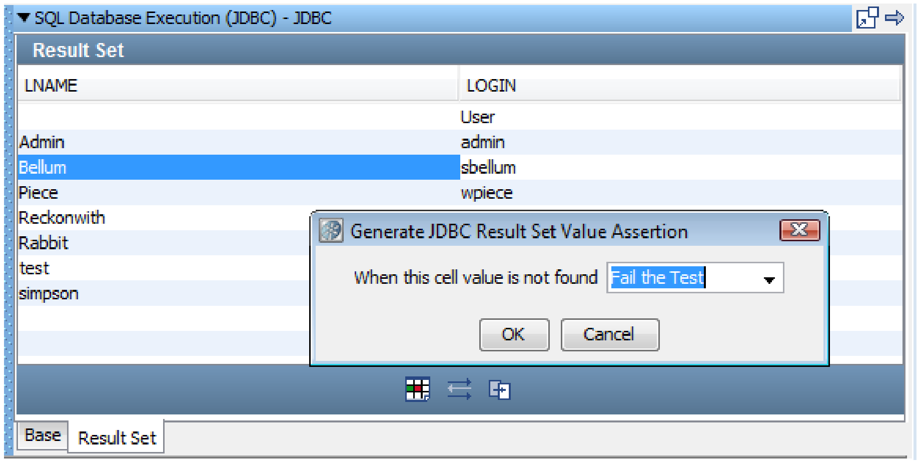
SQL Database Execution (JDBC) - JDBC

Result Set

LNAME	LOGIN
	Lisa7
	webapp-1591155383
	webapp-274934681
	webapp-1046745658
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	areck
Rabbit	boaty
test	itko
simpson	lisa_simpson

3. From the drop-down list, select the Fail the Test option.

   If the last name that you selected is not found, then the test fails.


SQL Database Execution (JDBC) - JDBC
Result Set
Generate JDBC Result Set Value Assertion
When this cell value is not found
Fail the Test
OK
Cancel
Base
Result Set

LNAME	LOGIN
	User
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	
Rabbit	
test	
simpson	

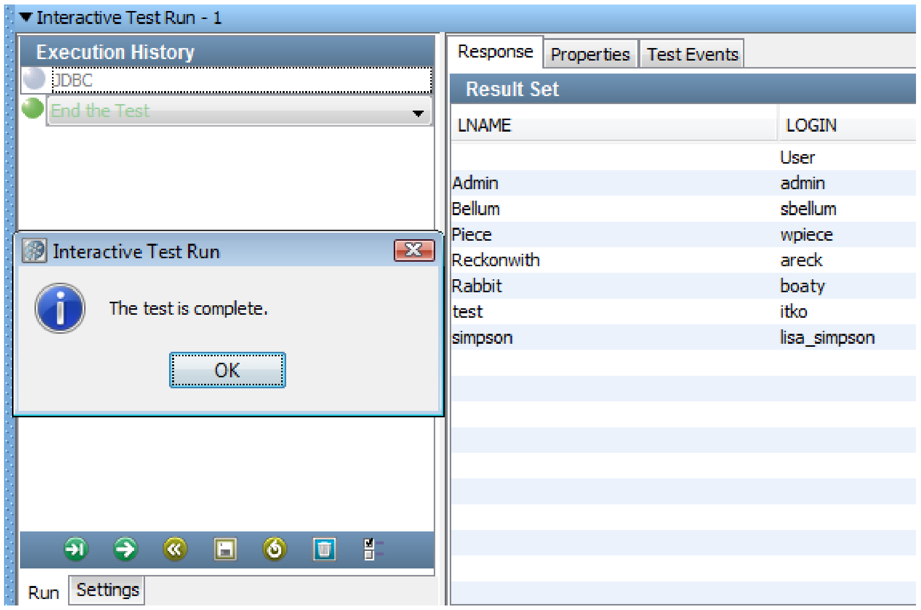
4. Click OK.
5. Click Save.

Step 7 - Run the Test Case

Follow these steps:

1. From the toolbar, click Start ITR.
2. Click Execute Next Step.

   The test runs successfully. The result set is shown in the Response tab.



LNAME	LOGIN
	User
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	areck
Rabbit	boaty
test	itko
simpson	lisa_simpson

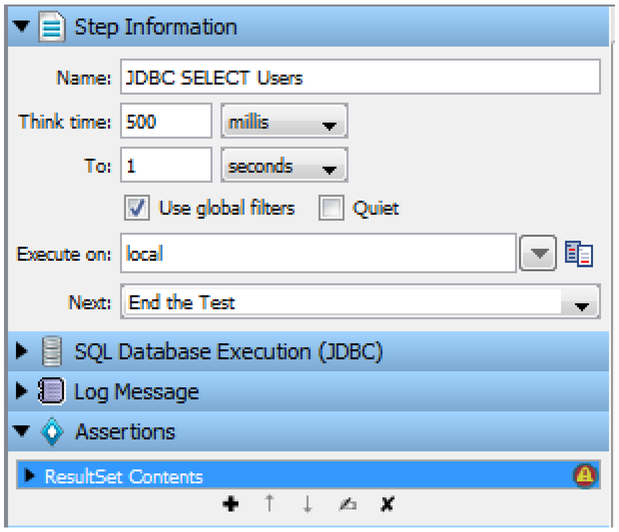
3. Retract the ITR tray.

Step 8 - Change the Assertion

You now modify the assertion to cause the test to fail.

Follow these steps:

1. In the model editor, click the JDBC SELECT Users test step.
2. Open the Assertions tab in the Element Tree.

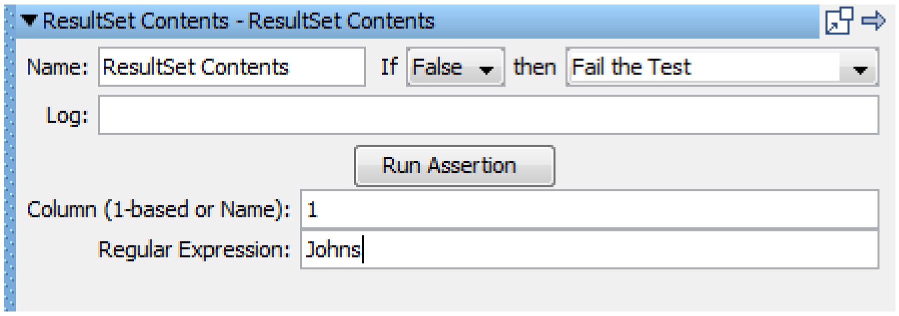


Name: JDBC SELECT Users
Think time: 500 millis
To: 1 seconds
Use global filters [✓] Quiet [ ]
Execute on: local
Next: End the Test

3. Double-click the assertion that you created earlier.

   The assertion editor is opened. The lower portion indicates that the assertion checks the first column of the result set for the specified value.

4. Change the value of the Regular Expression field to Johns.



Name: ResultSet Contents If False then Fail the Test
Log:

Column (1-based or Name): 1
Regular Expression: Johns

5. Start a new ITR and run the test case again.

   The test fails.

6. Retract the ITR tray.

Step 9 - Add a Filter

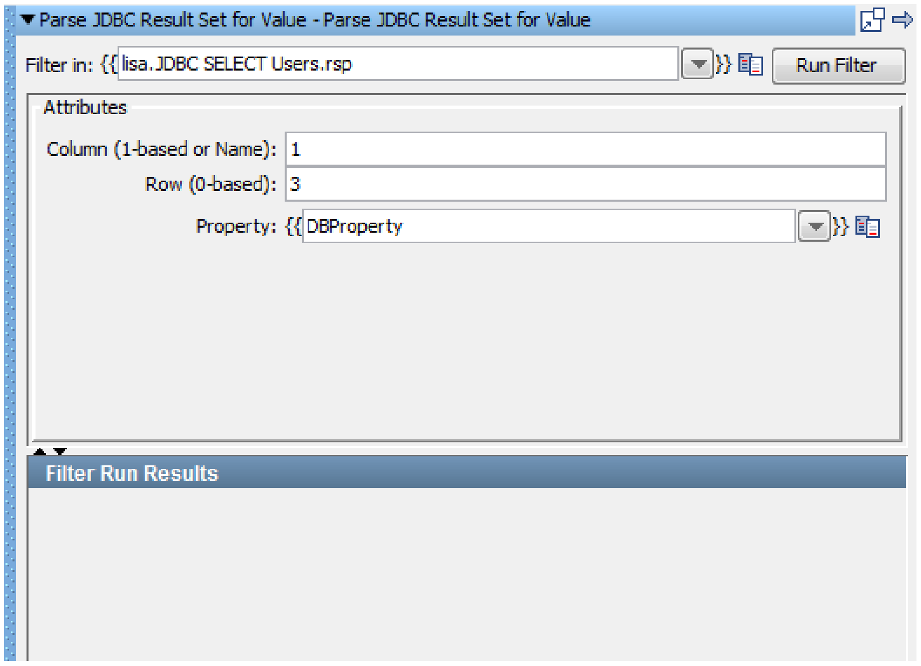
Add a database filter that captures the value in the first column and fourth row of the result set. The value is stored in a property.

Follow these steps:

1. In the model editor, select the JDBC SELECT Users test step.
2. Open the Filters tab in the Element Tree.
3. Click ✚ Add.
4. From the Database Filters submenu, select Extract Value from JDBC Result Set.

   The filter editor opens.
5. In the Column field, enter *1.* Or, you can enter the actual column name, which is *LNAME.*
6. In the Row field, enter *3.*

   This field is zero-based. Therefore, the value 3 refers to the fourth row.
7. In the Property field, enter *DBProperty.*


Parse JDBC Result Set for Value - Parse JDBC Result Set for Value
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.JDBC SELECT Users.rsp}}

Column (1-based or Name): 1

Row (0-based): 3

Property: {{DBProperty}}

8. Click Save.

Step 10 - Test the Filter and Assertion

Follow these steps:

1. Start a new ITR and run the test case again.

   The test fails because Johns was not found in the result set.

2. Click the Test Events tab.

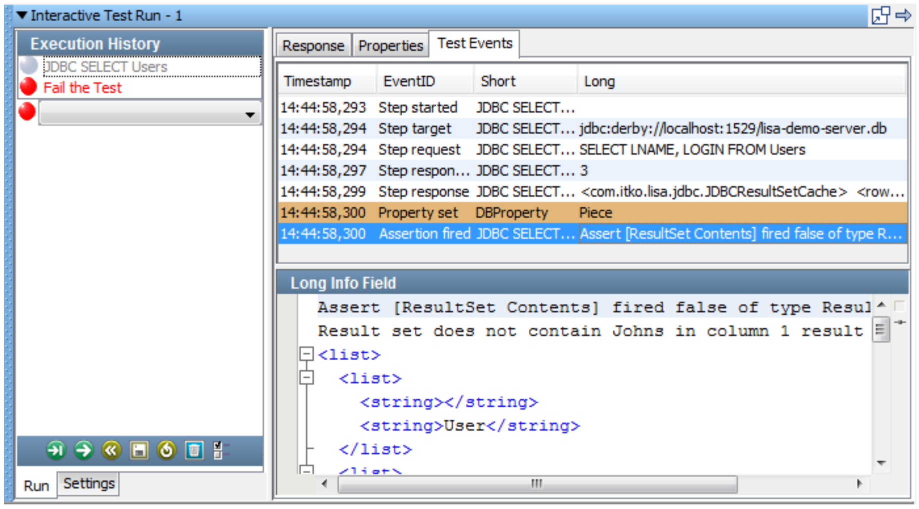
3. Click the Property set event.

   Notice that DBProperty was set to the value specified by the filter.

4. Click the Assertion fired event.

   The Long Info Field area indicates that the assertion fired because the first column of the result set did not contain the value Johns.

Figure 1: Screenshot of ITR Long Info Field for Tutorial 9

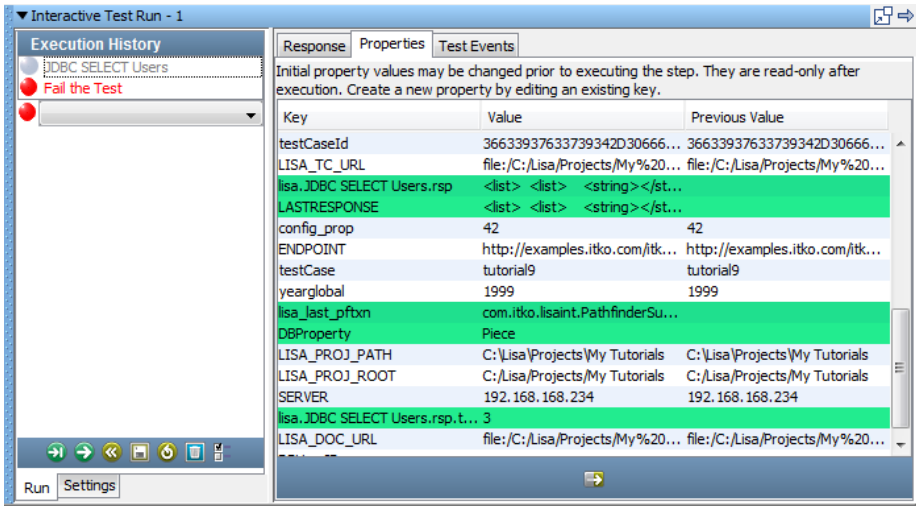


Timestamp	EventID	Short	Long
14:44:58,293	Step started	JDBC SELECT...	
14:44:58,294	Step target	JDBC SELECT...	jdbc:derby://localhost:1529/lisa-demo-server.db
14:44:58,294	Step request	JDBC SELECT...	SELECT LNAME, LOGIN FROM Users
14:44:58,297	Step respon...	JDBC SELECT...	3
14:44:58,299	Step response	JDBC SELECT...	<com.itko.lisa.jdbc.JDBCResultSetCache> <row...
14:44:58,300	Property set	DBProperty	Piece
14:44:58,300	Assertion fired	JDBC SELECT...	Assert [ResultSet Contents] fired false of type R...

```
Assert [ResultSet Contents] fired false of type Resul
Result set does not contain Johns in column 1 result
<list>
  <list>
    <string></string>
    <string>User</string>
  </list>
```

5. Click the Properties tab.
6. Locate and review the DBProperty row.

Figure 2: Screenshot for ITR results for Tutorial 9



Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
testCaseId	36633937633739342D30666...	36633937633739342D30666...
LISA_TC_URL	file:/C:/Lisa/Projects/My%20...	file:/C:/Lisa/Projects/My%20...
lisa.JDBC SELECT Users.rsp	<list> <list> <string></st...	
LASTRESPONSE	<list> <list> <string></st...	
config_prop	42	42
ENDPOINT	http://examples.itko.com/itk...	http://examples.itko.com/itk...
testCase	tutorial9	tutorial9
yearglobal	1999	1999
lisa_last_pftxn	com.itko.lisaint.PathfinderSu...	
DBProperty	Piece	
LISA_PROJ_PATH	C:\Lisa\Projects\My Tutorials	C:\Lisa\Projects\My Tutorials
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tutorials	C:/Lisa/Projects/My Tutorials
SERVER	192.168.168.234	192.168.168.234
lisa.JDBC SELECT Users.rsp.t...	3	
LISA_DOC_URL	file:/C:/Lisa/Projects/My%20...	file:/C:/Lisa/Projects/My%20...

Tutorial 9 - Review

In this tutorial, you created a test case to query a database. You used the Users table from the Apache Derby database that accompanies the applications on the demo server. You learned how to:

- Connect to the database.
- Execute a SQL query against the database.
- Add assertions and filters.

Tutorial 10 - Stage a Quick Test

In this tutorial, you use the quick staging option (quick test) to learn how to stage tests and read subsequent reports. You run the quick test on the multi-tier-combo example that accompanies DevTest. Running a quick test is the simplest way to stage a test.

Tutorial Tasks

In this tutorial, you:

- Use the multi-tier-combo test case.
- Use the quick test feature.
- Select and format reports.

Prerequisites

- You have completed Tutorials 5 through 9.
- DevTest Workstation is open.
- You have access to the demo server.

Step 1 - Open the Test Case

Open a test case from the examples project.

Follow these steps:

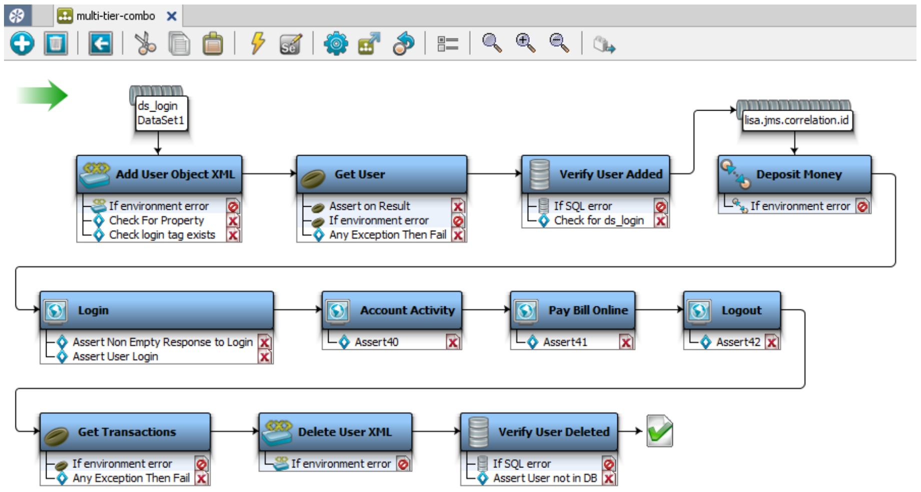
1. Select File, Open, Test Case, File System from the main menu.
2. Navigate to the LISA_HOME\examples\Tests folder.
3. Select multi-tier-combo and click Open.

   The multi-tier-combo test case opens in the model editor.

Step 2 - Review the Test Case

Review the various types of test steps in this test case. For example:

- Add User is a Web Service Execution (Legacy) step.
- Get User is an Enterprise JavaBean Execution step.
- Verify User Added is a SQL Database Execution (JDBC) step.
- Deposit Money is a JMS Messaging (JNDI) step.



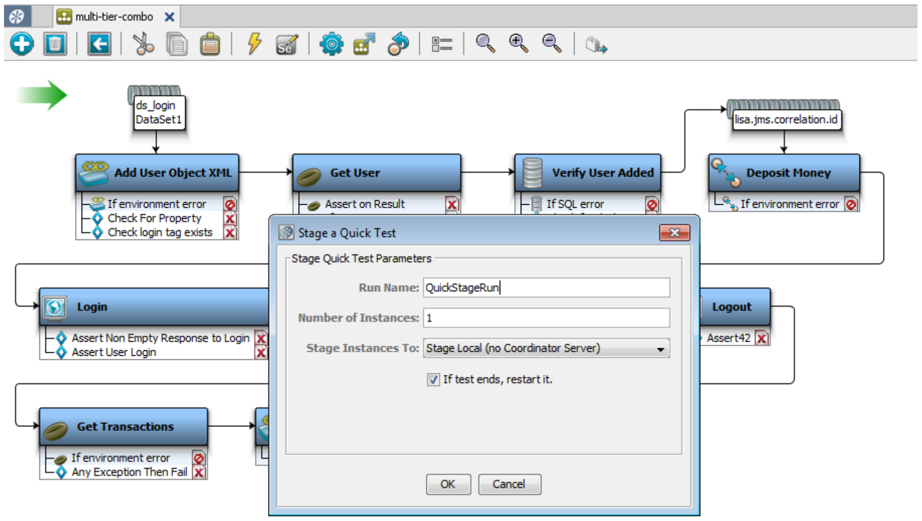
You used many of these steps in tutorials 6 through 9. In this tutorial, you use all the test steps to build a more realistic test case involving several layers of the application.

Step 2 - Part A - Run a Quick Test

Follow these steps:

1. From the menu bar, click Stage a Quick Test on the test case toolbar.

   To stage a quick test, the example test case can be open in the model editor. Or, you can right-click on the test in the Project panel and can enter the parameters to stage a quick test from there.



Stage a Quick Test

Stage Quick Test Parameters

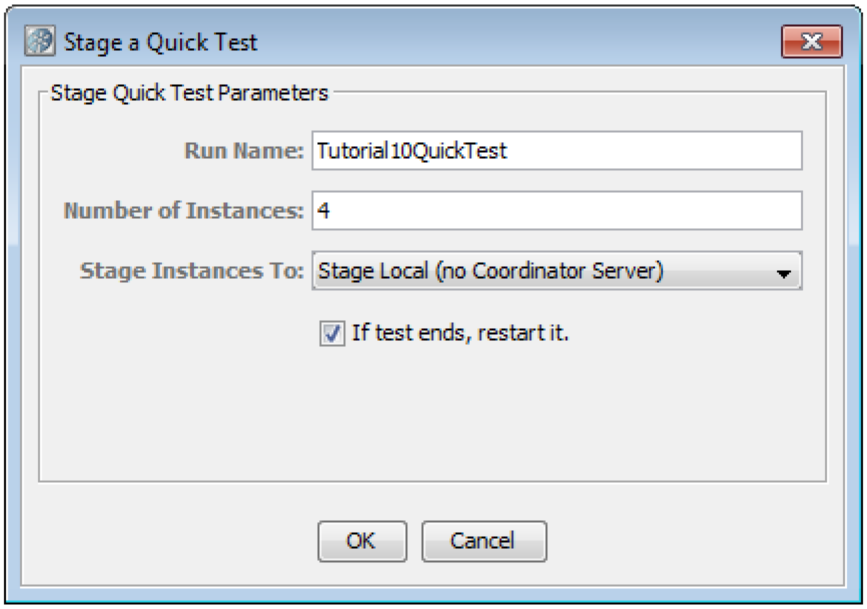
Run Name: QuickStageRun

Number of Instances: 1

Stage Instances To: Stage Local (no Coordinator Server)

☑ If test ends, restart it.

2. In the Stage a Quick Test dialog, complete the following required information:
   - Run Name: Enter a unique name (Tutorial10QuickTest).
   - Number of Instances: Enter a number of users to run the test concurrently (4).
   - Stage Instances To: Select the name of the coordinator server or stage it locally.
   - To restart the test, select If test ends, restart it.



Run Name: Tutorial10QuickTest

Number of Instances: 4

Stage Instances To: Stage Local (no Coordinator Server)

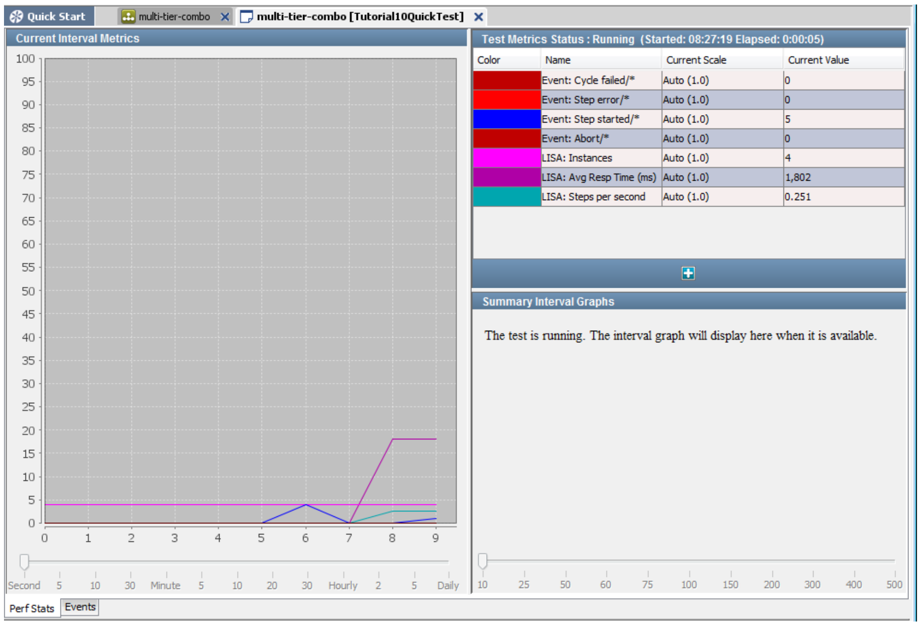
[✓] If test ends, restart it.

3. Click OK.

   The Test Run window opens, but the test has not started yet.

4. To start the test running, click ➔ from the main toolbar.

   The test begins, and the graph immediately plots results.

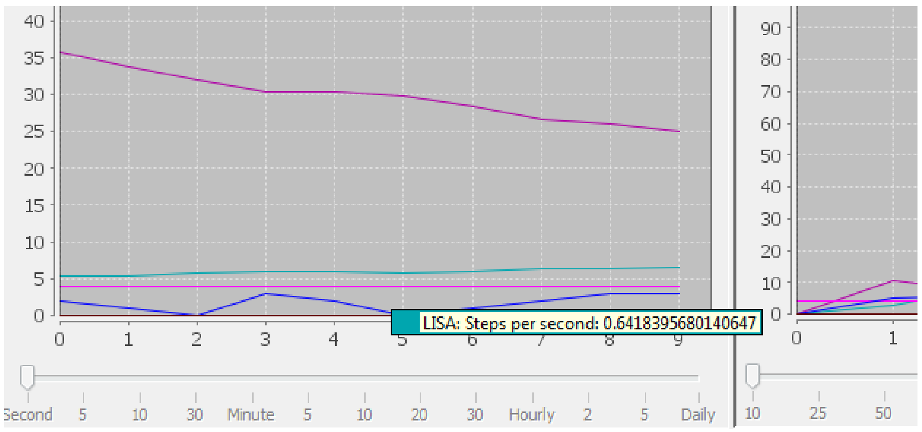


Test Metrics Status : Running (Started: 08:27:19 Elapsed: 0:00:05)

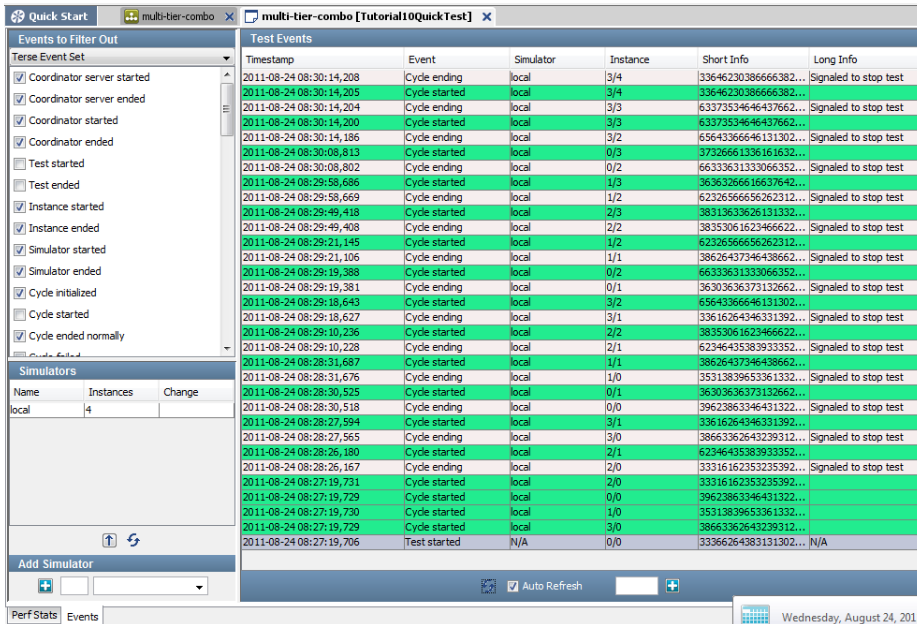
Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	5
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	4
🟪	LISA: Avg Resp Time (ms)	Auto (1.0)	1,802
🟦	LISA: Steps per second	Auto (1.0)	0.251

The test is running. The interval graph will display here when it is available.

You can roll over the graph lines to view descriptions.



5. To select which events to display, select the Events tab.



Events to Filter Out: Terse Event Set

- ✓ Coordinator server started
- ✓ Coordinator server ended
- ✓ Coordinator started
- ✓ Coordinator ended
- ☐ Test started
- ☐ Test ended
- ✓ Instance started
- ✓ Instance ended
- ✓ Simulator started
- ✓ Simulator ended
- ✓ Cycle initialized
- ☐ Cycle started
- ✓ Cycle ended normally

Simulators

Name	Instances	Change
local	4	

Test Events

Timestamp	Event	Simulator	Instance	Short Info	Long Info
2011-08-24 08:30:14,208	Cycle ending	local	3/4	33646230386666382...	Signaled to stop test
2011-08-24 08:30:14,205	Cycle started	local	3/4	33646230386666382...	
2011-08-24 08:30:14,204	Cycle ending	local	3/3	63373534646437662...	Signaled to stop test
2011-08-24 08:30:14,200	Cycle started	local	3/3	63373534646437662...	
2011-08-24 08:30:14,186	Cycle ending	local	3/2	65643366646131302...	Signaled to stop test
2011-08-24 08:30:08,813	Cycle started	local	0/3	37326661336161632...	
2011-08-24 08:30:08,802	Cycle ending	local	0/2	66333631333066352...	Signaled to stop test
2011-08-24 08:29:58,686	Cycle started	local	1/3	36363266616637642...	
2011-08-24 08:29:58,669	Cycle ending	local	1/2	62326566656262312...	Signaled to stop test
2011-08-24 08:29:49,418	Cycle started	local	2/3	38313633626131332...	
2011-08-24 08:29:49,408	Cycle ending	local	2/2	38353061623466622...	Signaled to stop test
2011-08-24 08:29:21,145	Cycle started	local	1/2	62326566656262312...	
2011-08-24 08:29:21,106	Cycle ending	local	1/1	38626437346438662...	Signaled to stop test
2011-08-24 08:29:19,388	Cycle started	local	0/2	66333631333066352...	
2011-08-24 08:29:19,381	Cycle ending	local	0/1	36303636373132662...	Signaled to stop test
2011-08-24 08:29:18,643	Cycle started	local	3/2	65643366646131302...	
2011-08-24 08:29:18,627	Cycle ending	local	3/1	33616264346331392...	Signaled to stop test
2011-08-24 08:29:10,236	Cycle started	local	2/2	38353061623466622...	
2011-08-24 08:29:10,228	Cycle ending	local	2/1	62346435383933352...	Signaled to stop test
2011-08-24 08:28:31,687	Cycle started	local	1/1	38626437346438662...	
2011-08-24 08:28:31,676	Cycle ending	local	1/0	35313839653361332...	Signaled to stop test
2011-08-24 08:28:30,525	Cycle started	local	0/1	36303636373132662...	
2011-08-24 08:28:30,518	Cycle ending	local	0/0	39623863346431322...	Signaled to stop test
2011-08-24 08:28:27,594	Cycle started	local	3/1	33616264346331392...	
2011-08-24 08:28:27,565	Cycle ending	local	3/0	38663362643239312...	Signaled to stop test
2011-08-24 08:28:26,180	Cycle started	local	2/1	62346435383933352...	
2011-08-24 08:28:26,167	Cycle ending	local	2/0	33316162353235392...	Signaled to stop test
2011-08-24 08:27:19,731	Cycle started	local	2/0	33316162353235392...	
2011-08-24 08:27:19,729	Cycle started	local	0/0	39623863346431322...	
2011-08-24 08:27:19,730	Cycle started	local	1/0	35313839653361332...	
2011-08-24 08:27:19,729	Cycle started	local	3/0	38663362643239312...	
2011-08-24 08:27:19,706	Test started	N/A	0/0	33366264383131302...	N/A

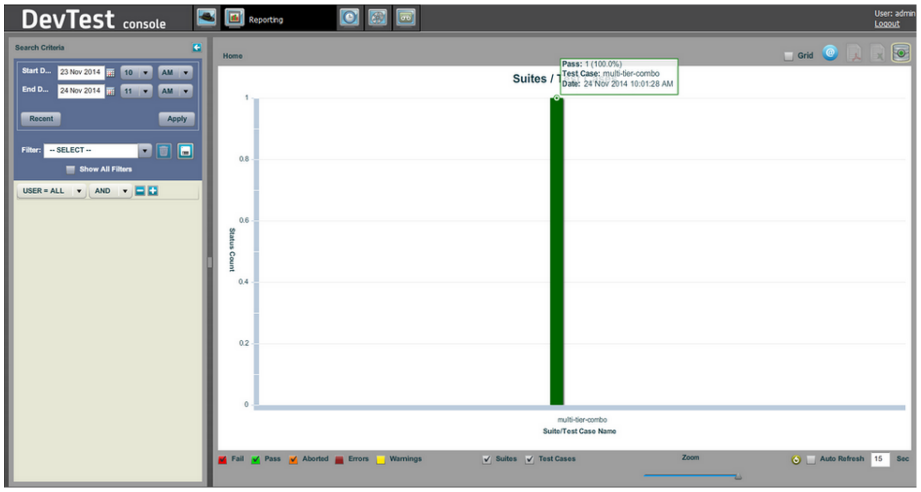
Step 2 - Part B - View Generated Reports

DevTest provides a report viewer to view reports.

Follow these steps:

1. Click View, Reporting Console from the main menu, or click Reports on the toolbar.

   The Report Viewer opens.



2. To see only recent activity, click the Recent button.

   The graph in the previous graphic shows that all of the tests in this test case passed.

You can right-click the graph for different menus. For more information about reports, see the Reports section in *Using CA Application Test.*

Tutorial 10 - Review

In this tutorial, you tested the multi-tier-combo example using a quick test. You learned how to:

- Review a test case containing several types of test steps.
- Configure and run a test in the quick test feature.
- Examine a report that is generated from the test run.

Chapter 3: Mobile Testing Tutorials

This section contains tutorials that illustrate various aspects of mobile testing.

Prerequisites for each tutorial

- You have installed DevTest Workstation and entered your DevTest Solutions license credentials.
- You have completed all of the tasks that are described in Setting Up the Mobile Testing Environment.
- For *Tutorial 1 - Record an iOS Test Case*, download the UICatalog.app.zip tutorial application to your local computer and unzip it.
- For *Tutorial 2 - Record an Android Test Case*, download the ApiDemos.apk tutorial application to your local computer.

This section contains the following topics:

Tutorial 1 - Record an iOS Test Case

This tutorial illustrates each of the steps that are required to record a test case using an iOS simulator.

In this tutorial, you perform the following tasks:

- Create an asset
- Record a mobile test case
- Run the test case in the ITR

Step 1 - Start DevTest Workstation

Follow these steps:

1. Start the registry.
   - For DevTest Server:
     a. Click Start, All Programs, DevTest Solutions, Registry.
     b. Wait until the DevTest Registry Ready message appears.
   - For DevTest Workstation, use a registry that is running on another computer.
2. Click Start, All Programs, DevTest Solutions, DevTest Workstation.

   The Set DevTest Registry dialog opens.
3. Select a registry and click OK.

Step 2 - Create a Project

The project that you create holds all the test case example files that are required for the mobile tutorials.

Note: If you created the MyMobileTutorials project in a previous tutorial, skip to Step 3, Create a Config File for an Android Simulator (see page 116).

Follow these steps:

1. From the DevTest Workstation main menu, select File, New, Project.

   The Create New Project dialog appears.
2. In the Project Name field, type **MyMobileTutorials**.
3. Accept the default setting to create the project in the LISA_HOME directory.
4. Click Create.

   The MyMobileTutorials project is created.

Step 3 - Create a Config File for an iOS Simulator

The default configuration is named project.config, and is created automatically for a new project. The project.config file is located in the Configs folder in the Project panel. You can create a configuration file that contains asset information specific to using an iOS simulator.

Follow these steps:

1. Right-click the Configs folder in the Project panel and select Create New Config.
2. Enter the name of the new configuration: **MobileiOSSimulator.**
3. Click OK.

   The MobileiOSSimulator config file opens in the properties editor.
4. Right-click the MobileiOSSimulator config file in the Project panel and select Make Active.

Step 4 - Create an iOS Simulator Asset

A Simulator Session asset that is associated with a specific config lets you specify the mobile simulator that is used for testing.

Follow these steps:

1. Open the MobileiOSSimulator configuration file that you created in Step 3 - Create a Config File for an iOS Simulator (see page 111) (if it is not already open).
2. In the Asset Browser, click Add at the bottom of the pane.
3. Click Mobile Session.

   The Mobile Session dialog displays.

   Note: If you have already defined values for any of these fields, you can select the value from the list for that field. To add or remove values from the list, select Maintain List.
4. Enter the following fields:

   Name

   Enter **iOSSimulator**.

   Description

   Enter **iOS Simulator for use with Mobile Tutorial 1**.

   Platform

   Select **iOS** from the drop-down menu.

   Application

   Navigate to and select the **UICatalog.app** file that you downloaded.

   Family

   Select **iPhone only** from the drop-down menu.

   Target

   Select **Simulator** from the drop-down menu.

   Simulator

   Navigate to and select the iOS simulator on your machine.

   iOS Version

   Select 6.1.
5. (Optional) To verify the asset before saving, click .
6. Click OK.

   The new iOS Simulator asset appears in the Asset Browser.
7. Click Save.

Step 5 - Record a Test Case

Follow these steps:

1. Ensure the MobileiOSSimulator config file is active.
2. Create a test case named **Tutorial_iOS_Simulator**:
   a. Right-click the Tests folder in the Project panel and click Create New Test Case.
   b. In the dialog, browse to the directory where you plan to store the test case.
   c. Enter the name of the new test case.
   d. Click Save.
   e. Click Create steps by recording or templating ⚡.
3. Click Record Test Case for User Interface, Mobile Recorder.

   The mobile test recorder and the mobile simulator windows open.
4. If you defined multiple mobile assets, select an asset to connect to using the Choose Mobile asset drop-down list, then click OK.
5. Click Start Recording at the bottom of the recorder window.

   You are now recording.
6. Perform the actions that you want to record for your test case in the recorder window.

   Note: Do not perform these actions in the mobile simulator directly. All actions that are performed in the recorder window are sent to the simulator automatically.
7. Click Stop Recording when you have captured all of the actions for your test.

   The recorder window and the mobile simulator close. Your new test case is populated with test steps that represent the mobile actions that are captured during the recording. Each test step represents the actions that you performed on a specific screen in the application.
8. To view the details of each step:
   a. Click the test step to review.
   b. Click Mobile testing step in the element tree on the right to expand the details for the step.

      The Mobile Testing Step tab opens and shows a screenshot of the test application. The Actions section at the top of the tab shows the individual actions that are performed in the test step.
   c. To view the screenshot associated with a specific action, click the action in the Actions section.

      For more information about modifying a recorded test step, see Modify Mobile Test Steps.

For more information about adding assertions to your test step, see Add an Assertion to a Mobile Test Step.

Step 6 - Run the Test Case in the ITR

Follow these steps:

1. Open the test case from Step 5 - Record a Test Case (see page 113) (if it is not already open).
2. Click the ITR icon on the toolbar.

   Note: Select whether to start a new ITR run or open a previous ITR run (if you have run the ITR before).

   The Interactive Test Run window opens.
3. Click Execute at the bottom of the ITR window.

   The mobile simulator window opens and DevTest runs the test steps. The mobile application is visible on the simulator while the test is running.

   When the test is complete, a message opens indicating the test is complete. The simulator window closes.
4. Click OK.

   For more information about using the ITR, see Running Test Cases and Suites.

Tutorial 2 - Record an Android Test Case

This tutorial illustrates each of the steps that are required to record a test case using an Android simulator.

In this tutorial, you perform the following tasks:

- Create an asset
- Record a mobile test case
- Run the test case in the ITR

Step 1 - Start DevTest Workstation

Follow these steps:

1. Start the registry.
   - For DevTest Server:
     a. Click Start, All Programs, DevTest Solutions, Registry.
     b. Wait until the DevTest Registry Ready message appears.
   - For DevTest Workstation, use a registry that is running on another computer.
2. Click Start, All Programs, DevTest Solutions, DevTest Workstation.

   The Set DevTest Registry dialog opens.
3. Select a registry and click OK.

Step 2 - Create a Project

The project that you create holds all the test case example files that are required for the mobile tutorials.

Note: If you created the MyMobileTutorials project in a previous tutorial, skip to Step 3, Create a Config File for an Android Simulator (see page 116).

Follow these steps:

1. From the DevTest Workstation main menu, select File, New, Project.

   The Create New Project dialog appears.
2. In the Project Name field, type **MyMobileTutorials**.
3. Accept the default setting to create the project in the LISA_HOME directory.
4. Click Create.

   The MyMobileTutorials project is created.

Step 3 - Create a Config File for an Android Simulator

The default configuration is named project.config, and is created automatically for a new project. The project.config file is located in the Configs folder in the Project panel. You can create a configuration file that contains asset information specific to using an iOS simulator.

Follow these steps:

1. Right-click the Configs folder in the Project panel and select Create New Config.
2. Enter the name of the new configuration: **MobileAndroidSimulator.**
3. Click OK.

   The MobileAndroidSimulator config file opens in the properties editor.
4. Right-click the MobileAndroidSimulator config file in the Project panel and select Make Active.

Step 4 - Create an Android Emulator Asset

A Simulator Session asset that is associated with a specific config lets you specify the mobile simulator that is used for testing.

Follow these steps:

1. Open the MobileAndroidSimulator configuration file that you created in Step 3 - Create a Config File for an Android Simulator (see page 116) (if it is not already open).
2. In the Asset Browser, click Add at the bottom of the pane.
3. Click Mobile Session.

   The Mobile Session dialog displays.
4. Define the following fields:

   Name

   Enter **AndroidSimulator**.

   Description

   Enter **Android Simulator for use with Mobile Tutorial 2**.

   Platform

   Select **Android** from the drop-down menu.

   Application

   Navigate to and select the **ApiDemos.apk** file that you downloaded.

   Target

   Select **Emulator** from the drop-down menu.

   AVD

   Enter the name of the Android AVD that you defined when configuring mobile testing.

   SDK Version

   Select **4.2.2.**
5. Click to verify the asset.
6. Click OK.

   The new Android Simulator asset appears in the Asset Browser.
7. Click Save.

Step 5 - Record a Test Case

Follow these steps:

1. Ensure the MobileAndroidSimulator config file is active.
2. Create a test case named **Tutorial_Android_Simulator**.
   a. Right-click the Tests folder in the Project panel and click Create New Test Case.
   b. In the dialog, browse to the directory where you plan to store the test case.
   c. Enter the name of the new test case.
   d. Click Save.
3. Click Create steps by recording or templating ⚡.
4. Click Record Test Case for User Interface, Mobile Recorder.

   The mobile test recorder and the mobile simulator windows open.
5. If you defined multiple mobile assets, select an asset to connect to using the Choose Mobile asset drop-down list, then click OK.
6. Click Start Recording at the bottom of the recorder window.

   You are now recording.
7. Perform the actions that you want to record for your test case in the recorder window.

   Note: Do not perform these actions in the mobile simulator directly. All actions that are performed in the recorder window are sent to the simulator automatically.
8. Click Stop Recording when you have captured all of the actions for your test.

   The recorder window and the mobile simulator close. Your new test case is populated with test steps that represent the mobile actions that are captured during the recording. Each test step represents the actions that you performed on a specific screen in the application.
9. To view the details of each step:
   a. Click the test step to review.
   b. Click Mobile testing step in the element tree on the right to expand the details for the step.

      The Mobile Testing Step tab opens and shows a screenshot of the test application. The Actions section at the top of the tab shows the individual actions that are performed in the test step.
   c. To view the screenshot associated with a specific action, click the action in the Actions section.

      For more information about modifying a recorded test step, see Modify Mobile Test Steps.

For more information about adding assertions to your test step, see Add an Assertion to a Mobile Test Step.

Step 6 - Run the Test Case in the ITR

Follow these steps:

1. Open the test case from Step 5 - Record a Test Case (see page 113) (if it is not already open).
2. Click the ITR icon on the toolbar.

   Note: Select whether to start a new ITR run or open a previous ITR run (if you have run the ITR before).

   The Interactive Test Run window opens.
3. Click Execute at the bottom of the ITR window.

   The mobile simulator window opens and DevTest runs the test steps. The mobile application is visible on the simulator while the test is running.

   When the test is complete, a message opens indicating the test is complete. The simulator window closes.
4. Click OK.

   For more information about using the ITR, see Running Test Cases and Suites.

Chapter 4: CA Service Virtualization Tutorial

In this tutorial, you:

- Create and activate a configuration file
- Configure the VSE Recorder
- Record a test case
- Deploy a virtual service model
- Test against a virtual service model

This section contains the following topics:

Prerequisites

The following steps are prerequisites to completing this tutorial:

- DevTest Workstation and VSE are installed.
- You have reviewed the following topics:
  - Glossary (see page 139)
  - Introduction to Virtualization
  - Understanding CA Service Virtualization

Process for Creating and Testing a VSI

After you complete the prerequisites, complete the following steps:

- Step 1 - Start DevTest Workstation (see page 122)
- Step 2 - Start VSE (see page 123)
- Step 3 - Start Demo Server (see page 123)
- Step 4 - Run a Test Case (see page 123)
- Step 5 - Create a Configuration File (see page 124)
- Step 6 - Activate the Configuration File (see page 125)
- Step 7 - Configure the VSE Recorder (see page 126)
- Step 8 - Record the Test Case (see page 128)
- Step 9 - Deploy the Virtual Service Model (see page 130)
- Step 10 - Test Against the Virtual Service Model (see page 131)
- Review the Tutorial (see page 132)

Step 1 - Start DevTest Workstation

Follow these steps:

1. Ensure that the registry is running.
   - If your computer has DevTest Server installed:

   a. Start the registry by clicking Start Menu, All Programs, DevTest Solutions, EnterpriseDashboard. Wait until the "Enterprise Dashboard started" message appears.

   b. Start the registry by clicking Start Menu, All Programs, DevTest Solutions, Registry.

   - If your computer has DevTest Workstation installed, use a registry that is running on another computer.
2. Click Start, All Programs, DevTest Solutions, Workstation.

   The Set DevTest Registry dialog opens.
3. Select a registry and click OK.
4. The Login dialog opens. Enter a valid username and password and click Login.
5. Continue with "Step 2 - Start VSE (see page 123)".

Step 2 - Start VSE

Follow these steps:

1. Click Start, All Programs, DevTest Solutions, Virtual Service Environment.

   When the "Virtual service environment is now ready" line appears, VSE has initialized.

2. Continue with "Step 3 - Start Demo Server (see page 123)".

Step 3 - Start Demo Server

When DevTest Workstation and VSE are both running, you can start the demo server that you use to complete this tutorial.

Follow these steps:

1. Click Start, All Programs, DevTest Solutions, DevTest Demo Server.

   When the "Started in XXs:YYms" line appears, the demo server has started.

2. Continue with "Step 4 - Run a Test Case (see page 123)".

Step 4 - Run a Test Case

To verify that LISA Bank is available, run a test case in the Interactive Test Run (ITR).

Follow these steps:

1. Double-click **webservices.tst** in the Project panel.

   The webservices test case opens in a tab.

2. Click ITR ⚙ to open the ITR.

3. To run the test case on the demo server, click Automatically Execute Test ➔.

   When the test completes successfully, you know that the demo server is available and the test case runs correctly.

4. Close the ITR window.

5. Close the test case by clicking X on the editor tab.

6. Continue with "Step 5 - Create a Config File (see page 124)".

Step 5 - Create a Config File

The **webservices** test case is the source application driver. You use the VSE Recorder in DevTest Workstation to listen and record the transactions of this test case. Insert the VSE Recorder between the source client application and the live system. The live LISA Bank system web service is on localhost port 8080. Configure your endpoint so that it uses the listen port for the VSE Recorder, 8001. The recorder intercepts the request and response on the way to and from the live web service application on port 8080.

Each of the steps in your test case contains a field where you can specify an endpoint for the step. By default, DevTest uses the {{ENDPOINT1}} property to define this endpoint. By using a property and not hard-coding the endpoint, you can quickly change the end point for each step by changing the project configuration for the {{ENDPOINT1}} value.

The default configuration is named project.config, and is created automatically for a new project. The project.config file is located in the Configs folder in the Project panel. You can also create a configuration file.

Follow these steps:

1. Right-click the Configs folder in the Project panel and select Create New Config.
2. Enter the name of the new configuration: **VSRecorder**.
3. Click OK.

   The properties editor opens the VSRecorder config file.

To add a property to the configuration file:

1. To add a row, click Add ➕ at the bottom of the properties editor.
2. Select ENDPOINT1 from the Key field.
3. Enter the following value in the Value field:

   ```
   http://localhost:8001/itkoExamples/EJB3UserControlBean?wsdl
   ```

   Note: The port from the test case (8080) is changed to the listen port for the VSE Recorder (8001). The recorder can intercept the request and response on the way to and from the live web service application on port 8080.
4. Click Save on the main toolbar.
5. To close the config file, click X on the editor tab.
6. Continue with "Step 6 - Activate Config File (see page 125)".

Next Step (see page 125).

Step 6 - Activate Config File

For DevTest to use the newly created config file as the primary or active config file, activate it.

To activate the config file:

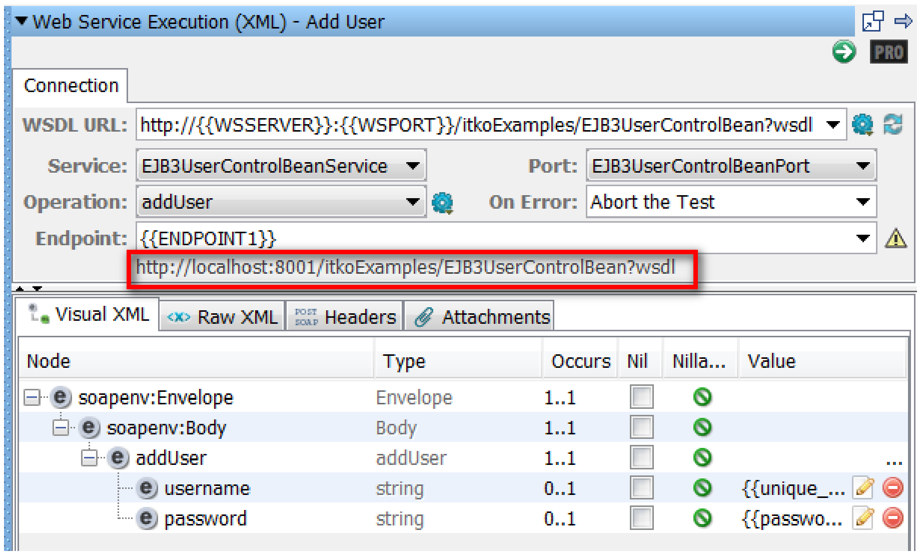
1. In the Project panel, right-click the VSRecorder config file.
2. Select Make Active.

To confirm that the ENDPOINT1 value from the VSRecorder config file is being used:

1. Double-click the **webservices** test case in the Project panel.

   The editor opens.

2. Double-click the **Add User** test step and verify that the Endpoint is on port 8001.



Web Service Execution (XML) - Add User

Connection

WSDL URL: http://{{WSSERVER}}:{{WSPORT}}/itkoExamples/EJB3UserControlBean?wsdl

Service: EJB3UserControlBeanService   Port: EJB3UserControlBeanPort

Operation: addUser   On Error: Abort the Test

Endpoint: {{ENDPOINT1}}

http://localhost:8001/itkoExamples/EJB3UserControlBean?wsdl

Visual XML | Raw XML | Headers | Attachments

Node	Type	Occurs	Nil	Nilla...	Value
soapenv:Envelope	Envelope	1..1	☐		
soapenv:Body	Body	1..1	☐		
addUser	addUser	1..1	☐		...
username	string	0..1	☐		{{unique_...
password	string	0..1	☐		{{passwo...

3. Continue with "Step 7 - Configure the VSE Recorder (see page 126)".

Step 7 - Configure the VSE Recorder

With the endpoint set, you are now ready to configure the VSE Recorder.

Follow these steps:

1. Click VSE Recorder in the main toolbar.

   On the Basics tab, configure the settings for the virtual service image, which stores the data. Configure the setting for the virtual service model, which creates the transaction workflow model.

2. Replace the default service image name, *newimage.vsi*, in the Write image to field with *VSETutorial.vsi*.

3. Leave the Import traffic field blank.

   For this tutorial, you will not import an existing traffic file.

4. Select the transport protocol HTTP/S.

   For this tutorial, you do not desensitize the data because you do not pass sensitive data (such as Social Security numbers or bank account numbers). Continue to the Export to field, which is used to export a raw traffic file of the recording. The exported file serves as raw backup of the entire recording session.

5. Click Browse, name the file **rt_VSETutorial**, and save it in the VServices folder.

   The "rt_" prefix indicates this file is a raw traffic file.

6. Click Browse to create the virtual service model file.

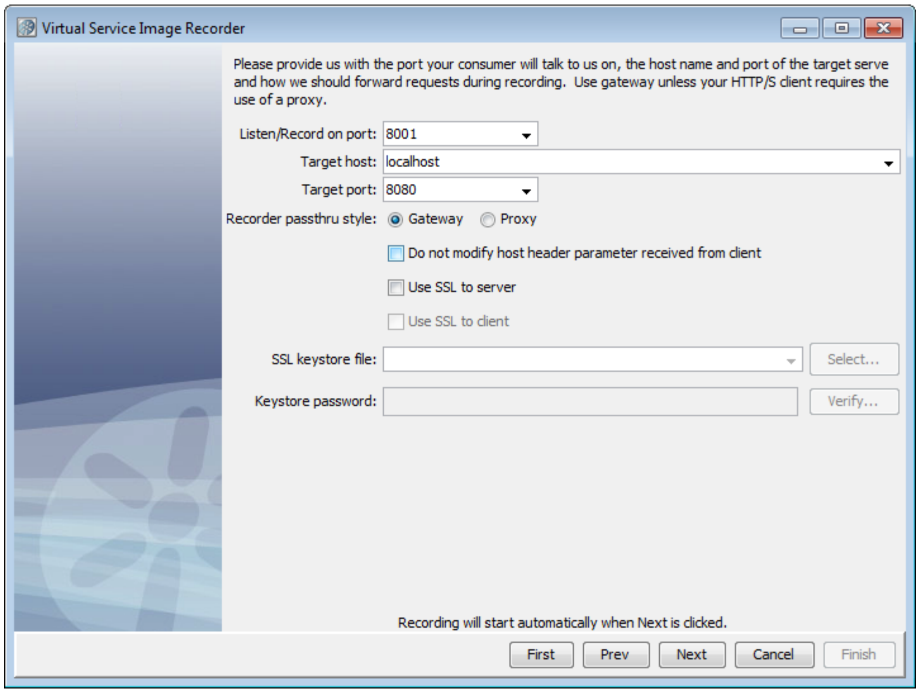
7. Enter the name **VSETutorial** and click **Save**.

8. Leave the VS Model style selections as they are, and click Next.

   On the next window, you designate the ports for the recorder.

9. By default, the recorder listens and records on port 8001, which you set as the endpoint in the VSRecorder config file. Add **localhost** to the Target host field.

10. Update the Target port, or the port the live LISA Bank system transactions are on, to **8080**.



Please provide us with the port your consumer will talk to us on, the host name and port of the target serve and how we should forward requests during recording. Use gateway unless your HTTP/S client requires the use of a proxy.

Listen/Record on port: 8001

Target host: localhost

Target port: 8080

Recorder passthru style: [✓ Gateway] Proxy

☐ Do not modify host header parameter received from client

☐ Use SSL to server

☐ Use SSL to client

SSL keystore file:

Keystore password:

Recording will start automatically when Next is clicked.

11. Select the Gateway option for the Recorder passthru style.

    With these settings completed, you can start recording.

12. Click Next.
13. Continue with "Step 8 - Record the Test Case (see page 128)".

Step 8 - Record the Test Case

Follow these steps:

1. Click Next to start recording.

   As the window shows, the recorder is listening on port 8001 to the target port 8080. The window also shows a session and transaction count. Because the test case has not been run, the count is at 0. When the test case finishes running, this window has three transactions that are counted.

2. In DevTest Workstation, click the test case tab, open the ITR, and execute the steps automatically.

3. Navigate to the recorder and verify that the three transactions were successfully recorded.

   With the test case recorded, you can end the recording session by clicking Next. DevTest processes the recording and presents a list of transactions. For more advanced virtual service transactions, modify the post-recording model and data on the next window. For our test case, we are ready to continue, so click Next.

   On the next window, the base path, binding, and logic settings are correct.

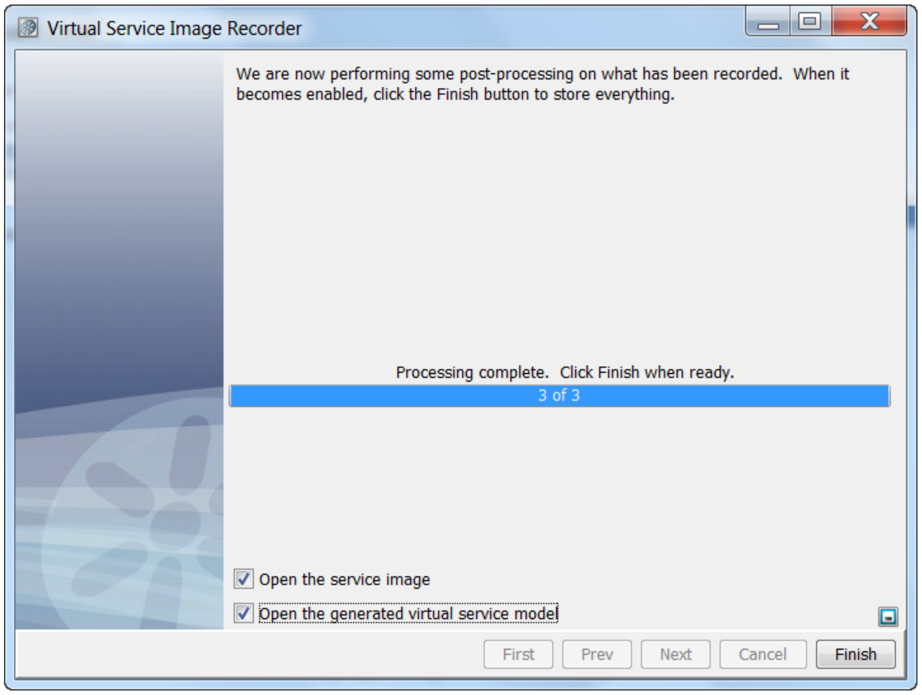
4. Click Next.

   The next window indicates that a Web Services (SOAP) data protocol handler has been selected for the request side.

5. Leave this selection unchanged, and click Next.

6. Make no changes on the next window and click Enter.

   DevTest processes the recording a final time and creates the virtual service model and the virtual service image files.


Virtual Service Image Recorder

We are now performing some post-processing on what has been recorded. When it becomes enabled, click the Finish button to store everything.

Processing complete. Click Finish when ready.

3 of 3

☑ Open the service image

☑ Open the generated virtual service model

You can open and edit each of these files in DevTest Workstation.

7. To see the virtual service model and virtual service image files, select the Open the service image and Open the generated virtual service model check boxes.

8. Click Finish.

   DevTest closes the recorder and opens the selected files in DevTest Workstation.

9. Continue with "Step 9 - Deploy the VSM (see page 130)".

Step 9 - Deploy the VSM

To deploy the Virtual Service Model (VSM):

1. To close the Demo Server window, click **X** in the upper right corner.

   Closing this window shuts down the demo server, which hosts LISA Bank. This shutdown leaves the test case without a system under test and allows you to use the virtual service you created.

2. Open DevTest Workstation and open the Server Console, which provides the VSE UI for managing virtual services.

3. Click VSE in the left navigation pane.

4. In DevTest Workstation, right-click VSETutorial virtual service model in the Project panel and select Deploy/Redeploy to VSE.

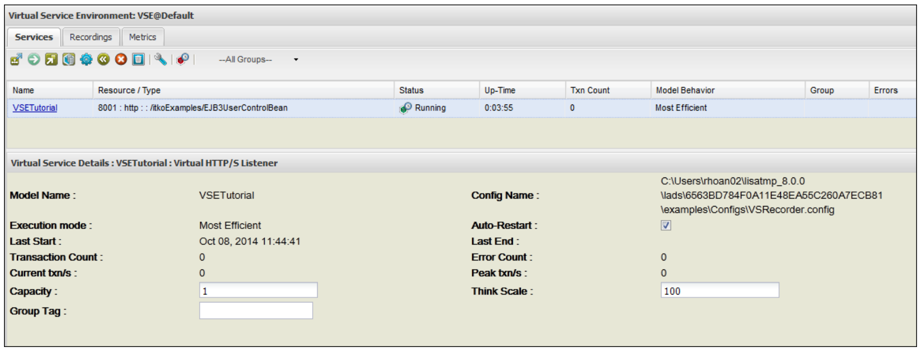
   On the Deploy Virtual Service dialog, the virtual service model and the VSRecorder config file are selected. The Group Tag is blank. The Concurrent capacity is set to 1, Think time scale is at 100%, and If service ends, automatically restart it is selected. Because the Start the service on deployment check box is selected, the virtual service starts when you deploy it.

   To share this model and the configurations, or to deploy it remotely, click the Save as MAR button to create a MAR file and distribute the model as necessary. All the prepopulated configurations are correct, so click Deploy, and the virtual service is deployed and displays in the VSE dashboard.

   The virtual service is deployed to VSE.

5. To return to the VSE Console, click the DevTest Console tab in your browser.

   The VSETutorial virtual service model appears in the Services tab.

Name	Resource / Type	Status	Up-Time	Txn Count	Model Behavior	Group	Errors
VSETutorial	8001 : http : : /itkoExamples/EJB3UserControlBean	Running	0:03:55	0	Most Efficient		

   Virtual Service Details : VSETutorial : Virtual HTTP/S Listener

Model Name :	VSETutorial	**Config Name :**	C:\Users\rhoan02\lisatmp_8.0.0 \lads\6563BD784F0A11E48EA55C260A7ECB81 \examples\Configs\VSRecorder.config
Execution mode :	Most Efficient	**Auto-Restart :**	✓
Last Start :	Oct 08, 2014 11:44:41	**Last End :**	
Transaction Count :	0	**Error Count :**	0
Current txn/s :	0	**Peak txn/s :**	0
Capacity :	1	**Think Scale :**	100
Group Tag :			

   The virtual service is started and ready to process transactions, which are tracked in the Txn Count column. The service initially shows that it has processed 0 transactions.

6. Run the test case and return to this window.

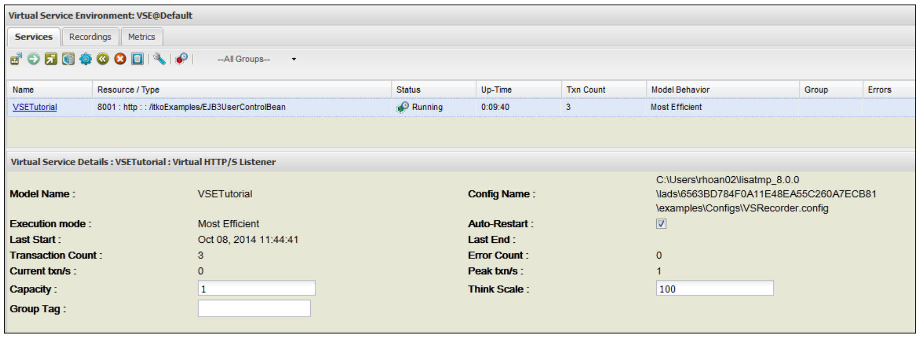
7. Continue with "Step 10 - Test Against the VSM (see page 131)".

Step 10 - Test Against the VSM

To test against the Virtual Service Model (VSM):

1. Open the **webservices** test case and run it in the ITR.
2. Return to the virtual service when the ITR completes.

   The transaction count is now at 3, confirming you are testing against the virtual service.



Virtual Service Environment: VSE@Default

Name	Resource / Type	Status	Up-Time	Txn Count	Model Behavior	Group	Errors
VSETutorial	8001 : http : : /itkoExamples/EJB3UserControlBean	Running	0:09:40	3	Most Efficient		

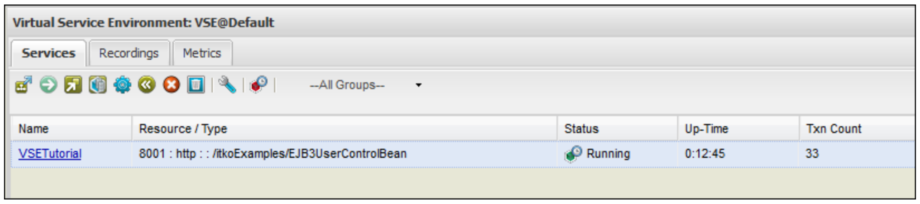
Virtual Service Details : VSETutorial : Virtual HTTP/S Listener

Model Name :	VSETutorial	**Config Name :**	C:\Users\rhoan02\lisatmp_8.0.0 \lads\6563BD784F0A11E48EA55C260A7ECB81 \examples\Configs\VSRecorder.config
Execution mode :	Most Efficient	**Auto-Restart :**	✓
Last Start :	Oct 08, 2014 11:44:41	**Last End :**	
Transaction Count :	3	**Error Count :**	0
Current txn/s :	0	**Peak txn/s :**	1
Capacity :	1	**Think Scale :**	100
Group Tag :			

Note: You can also stage and execute a test case or suite against the virtual service.

3. Right-click the **webservices** test case in the Project pane and select Stage a Quick Test.
4. Enter **10** for the Number of Instances.
5. Clear the If test ends, restart it check box, and click OK.
6. Click Play with the Test Monitor window open and let the test run.

   When you return to the VSE Dashboard after running the test case with 10 instances, you see the transaction count is 33.



Virtual Service Environment: VSE@Default

Name	Resource / Type	Status	Up-Time	Txn Count
VSETutorial	8001 : http : : /itkoExamples/EJB3UserControlBean	Running	0:12:45	33

7. Continue with "Review the Tutorial (see page 132)".

Review the Tutorial

In this tutorial, you created and tested a virtual service to review the basic functionality of VSE.

In the tutorial, you:

- Created and activated a configuration file
- Configured the VSE Recorder
- Recorded a test case
- Deployed a virtual service model
- Used the ITR and staged a test to test against a virtual service model

Chapter 5: CA Continuous Application Insight Tutorial

In this tutorial, you generate transactions by using the LISA Bank demo application. You then view the transactions in the Analyze Transactions window of the DevTest Portal.

Prerequisites

- The following components are running: Enterprise Dashboard, registry, demo server, broker, and portal.

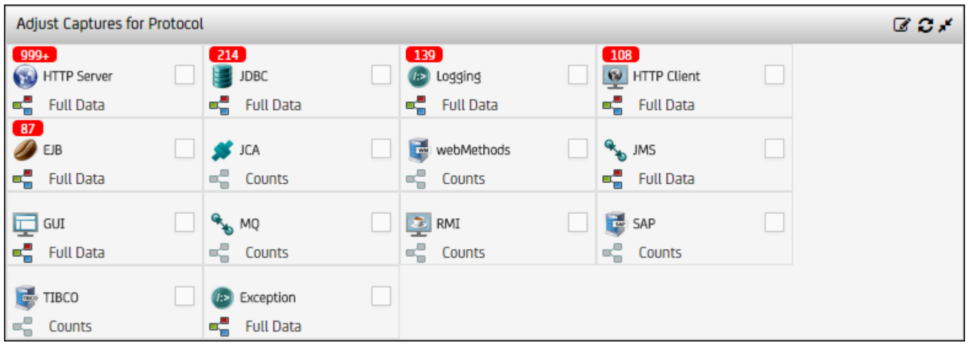
This section contains the following topics:

Step 1 - Increase the Default Capture Levels

Each protocol that the DevTest Java Agent can capture has a capture level.

The default capture level is Counts. To view the transactions later in this tutorial, you must set the capture level for the relevant protocols to Full Data.

The following graphic shows the Adjust Captures for Protocol pane in the Agents window.



Adjust Captures for Protocol

999+ HTTP Server Full Data	214 JDBC Full Data	139 Logging Full Data	108 HTTP Client Full Data
87 EJB Full Data	JCA Counts	webMethods Counts	JMS Full Data
GUI Full Data	MQ Counts	RMI Counts	SAP Counts
TIBCO Counts	Exception Full Data		

Follow these steps:

1. Open the DevTest Portal. If the portal is running on a local computer, the URL is **http://localhost:1507/devtest**.
2. Select Settings, Agents from the left navigation menu.

   The Agents window opens.
3. In the Agents pane, select the **JBoss_LISABank** agent.
4. If the Adjust Captures for Protocol pane is collapsed, expand the pane.
5. If some of the protocols do not appear, click Edit and select Show All Protocols.
6. Ensure that the capture levels for the following protocols are set to Full Data:
   - HTTP Server
   - JDBC
   - EJB
   - Logging

   To change a capture level, select the check box of the protocol and select Full Data from the capture level drop-down list.

Step 2 - Generate Transactions from the Demo Application

DevTest Solutions includes a demo application named LISA Bank.

In this procedure, you log in to the demo application, create an account, and log out.

Follow these steps:

1. In a web browser, enter **http://localhost:8080/lisabank**. If the demo server is running on another computer, replace **localhost** with the host name or IP address of the remote computer.

   The login page opens.

2. In the Name field, type **lisa_simpson**.
3. In the Password field, type **golisa**.
4. Click Login.

   The welcome page opens. The left side contains buttons for various actions that you can perform.

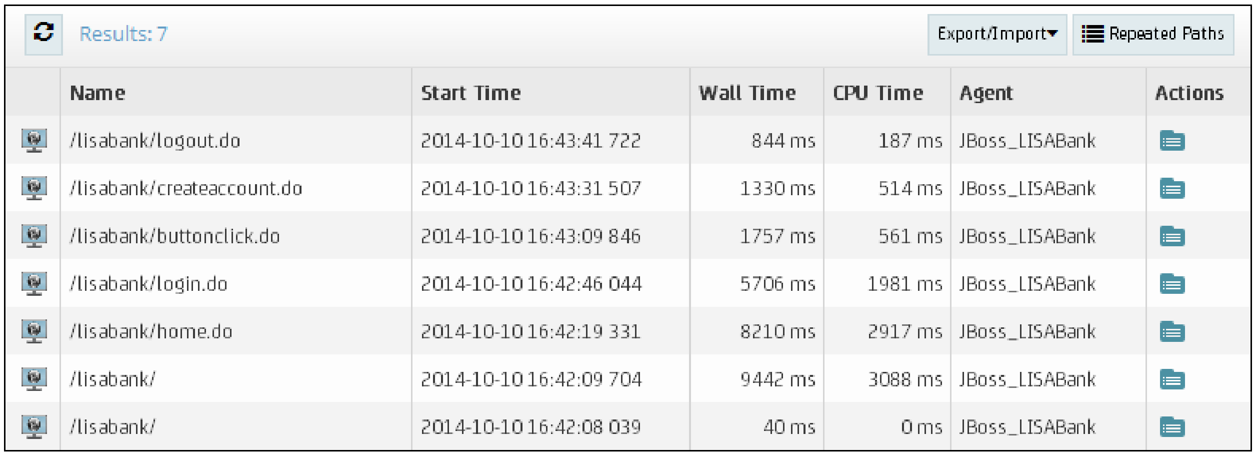
5. Create an account by performing these steps:
   a. Click New Account.
   b. In the Account Name field, type **My Checking**.
   c. In the Initial Balance field, replace the default value with **500**.
   d. Click Add Account.
6. Click Log Out.

Step 3 - View the Transactions in the Analyze Transactions Window

The Analyze Transactions window lets you do the following tasks:

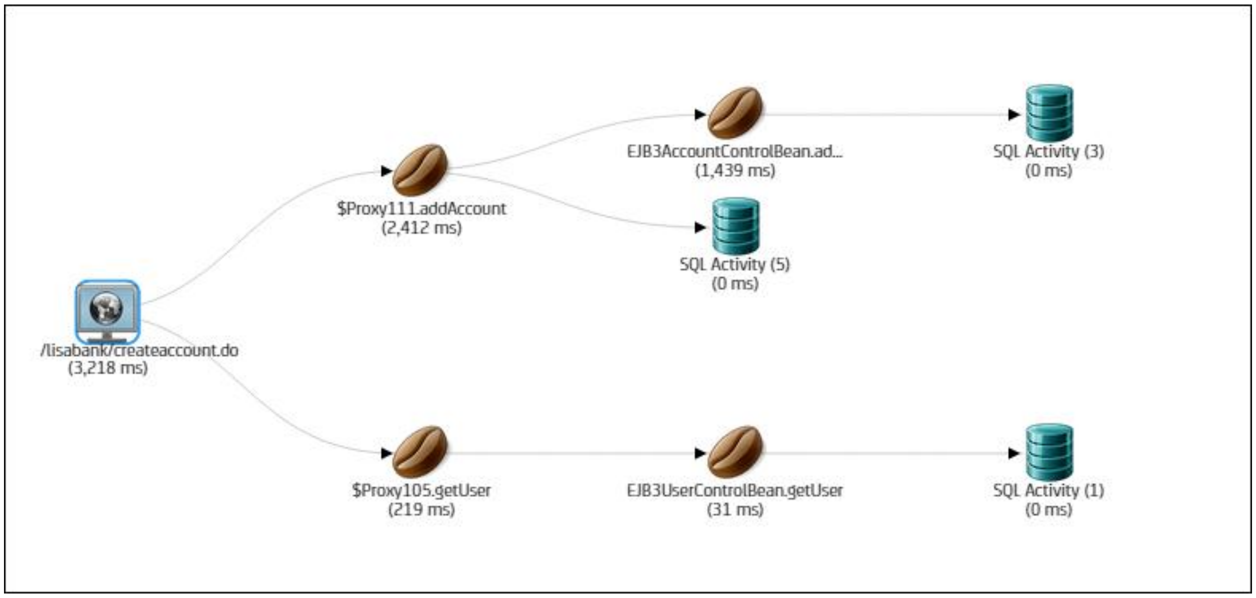
- View transactions
- Search and filter transactions
- View component details

The following graphic shows a set of transactions in the Analyze Transactions window. The transactions are displayed in the List view. The transactions appear in reverse chronological order.



	Name	Start Time	Wall Time	CPU Time	Agent	Actions
	/lisabank/logout.do	2014-10-10 16:43:41 722	844 ms	187 ms	JBoss_LISABank	
	/lisabank/createaccount.do	2014-10-10 16:43:31 507	1330 ms	514 ms	JBoss_LISABank	
	/lisabank/buttonclick.do	2014-10-10 16:43:09 846	1757 ms	561 ms	JBoss_LISABank	
	/lisabank/login.do	2014-10-10 16:42:46 044	5706 ms	1981 ms	JBoss_LISABank	
	/lisabank/home.do	2014-10-10 16:42:19 331	8210 ms	2917 ms	JBoss_LISABank	
	/lisabank/	2014-10-10 16:42:09 704	9442 ms	3088 ms	JBoss_LISABank	
	/lisabank/	2014-10-10 16:42:08 039	40 ms	0 ms	JBoss_LISABank	

The following graphic shows a path graph for one of the transactions.



Follow these steps:

1. Return to the DevTest Portal.
2. Select Application Insight, Analyze Transactions from the left navigation menu.

   The Analyze Transactions window opens.
3. Locate the transaction that has the name **/lisabank/createaccount.do**.
4. Click the Open transaction details button in the Actions column.

   The transactions details dialog opens. The upper area contains the path graph. The lower area contains information about the selected node.
5. Click an EJB node in the path graph.
6. Review the information that appears in each of the tabs.
7. Close the dialog.

Glossary

assertion

An *assertion* is an element that runs after a step and all its filters have run. An assertion verifies that the results from running the step match the expectations. An assertion is typically used to change the flow of a test case or virtual service model. Global assertions apply to each step in a test case or virtual service model. For more information, see Assertions in *Using CA Application Test.*

asset

An *asset* is a set of configuration properties that are grouped into a logical unit. For more information, see Assets in *Using CA Application Test*.

audit document

An *audit document* lets you set success criteria for a test, or for a set of tests in a suite. For more information, see Building Audit Documents in *Using CA Application Test.*

companion

A *companion* is an element that runs before and after every test case execution. Companions can be understood as filters that apply to the entire test case instead of to single test steps. Companions are used to configure global (to the test case) behavior in the test case. For more information, see Companions in *Using CA Application Test.*

configuration

A *configuration* is a named collection of properties that usually specify environment-specific values for the system under test. Removing hard-coded environment data enables you to run a test case or virtual service model in different environments simply by changing configurations. The default configuration in a project is named project.config. A project can have many configurations, but only one configuration is active at a time. For more information, see Configurations in *Using CA Application Test.*

Continuous Service Validation (CVS) Dashboard

The *Continuous Validation Service (CVS) Dashboard* lets you schedule test cases and test suites to run regularly, over an extended time period. For more information, see Continuous Validation Service (CVS) in *Using CA Application Test.*

conversation tree

A *conversation tree* is a set of linked nodes that represent conversation paths for the stateful transactions in a virtual service image. Each node is labeled with an operation name, such as withdrawMoney. An example of a conversation path for a banking system is getNewToken, getAccount, withdrawMoney, deleteToken. For more information, see *Using CA Service Virtualization.*

coordinator

A *coordinator* receives the test run information as documents, and coordinates the tests that are run on one or more simulator servers. For more information, see Coordinator Server in *Using CA Application Test.*

data protocol

A *data protocol* is also known as a data handler. In CA Service Virtualization, it is responsible for handling the parsing of requests. Some transport protocols allow (or require) a data protocol to which the job of creating requests is delegated. As a result, the protocol has to know the request payload. For more information, see Using Data Protocols in *Using CA Service Virtualization.*

data set

A *data set* is a collection of values that can be used to set properties in a test case or virtual service model at run time. Data sets provide a mechanism to introduce external test data into a test case or virtual service model. Data sets can be created internal to DevTest, or externally (for example, in a file or a database table). For more information, see Data Sets in *Using CA Application Test.*

desensitize

Desensitizing is used to convert sensitive data to user-defined substitutes. Credit card numbers and Social Security numbers are examples of sensitive data. For more information, see Desensitizing Data in *Using CA Service Virtualization.*

event

An *event* is a message about an action that has occurred. You can configure events at the test case or virtual service model level. For more information, see Understanding Events in *Using CA Application Test.*

filter

A *filter* is an element that runs before and after a step. A filter gives you the opportunity to process the data in the result, or store values in properties. Global filters apply to each step in a test case or virtual service model. For more information, see Filters in *Using CA Application Test.*

group

A *group*, or a *virtual service group,* is a collection of virtual services that have been tagged with the same group tag so they can be monitored together in the VSE Console.

Interactive Test Run (ITR)

The *Interactive Test Run (ITR)* utility lets you run a test case or virtual service model step by step. You can change the test case or virtual service model at run time and rerun to verify the results. For more information, see Using the Interactive Test Run (ITR) Utility in *Using CA Application Test.*

lab

A *lab* is a logical container for one or more lab members. For more information, see Labs and Lab Members in *Using CA Application Test.*

magic date

During a recording, a date parser scans requests and responses. A value matching a wide definition of date formats is translated to a *magic date*. Magic dates are used to verify that the virtual service model provides meaningful date values in responses. An example of a magic date is {{=doDateDeltaFromCurrent("yyyy-MM-dd","10");/*2012-08-14*/}. For more information, see Magic Strings and Dates in *Using CA Service Virtualization.*

magic string

A *magic string* is a string that is generated during the creation of a service image. A magic string is used to verify that the virtual service model provides meaningful string values in the responses. An example of a magic string is {{=request_fname;/chris/}}. For more information, see Magic Strings and Dates in *Using CA Service Virtualization.*

match tolerance

Match tolerance is a setting that controls how CA Service Virtualization compares an incoming request with the requests in a service image. The options are EXACT, SIGNATURE, and OPERATION. For more information, see Match Tolerance in *Using CA Service Virtualization.*

metrics

Metrics let you apply quantitative methods and measurements to the performance and functional aspects of your tests, and the system under test. For more information, see Generating Metrics in *Using CA Application Test.*

Model Archive (MAR)

A *Model Archive (MAR)* is the main deployment artifact in DevTest Solutions. MAR files contain a primary asset, all secondary files that are required to run the primary asset, an info file, and an audit file. For more information, see Working with Model Archives (MARs) in *Using CA Application Test.*

Model Archive (MAR) Info

A *Model Archive (MAR) Info* file is a file that contains information that is required to create a MAR. For more information, see Working with Model Archives (MARs) in *Using CA Application Test.*

navigation tolerance

Navigation tolerance is a setting that controls how CA Service Virtualization searches a conversation tree for the next transaction. The options are CLOSE, WIDE, and LOOSE. For more information, see Navigation Tolerance in *Using CA Service Virtualization.*

network graph

The network graph is an area of the Server Console that displays a graphical representation of the DevTest Cloud Manager and the associated labs. For more information, see Start a Lab in *Using CA Application Test*.

node

Internal to DevTest, a test step can also be referred to as a *node*, explaining why some events have node in the EventID.

path

A *path* contains information about a transaction that the Java Agent captured. For more information, see *Using CA Continuous Application Insight.*

path graph

A *path graph* contains a graphical representation of a path and its frames. For more information, see Path Graph in *Using CA Continuous Application Insight.*

project

A *project* is a collection of related DevTest files. The files can include test cases, suites, virtual service models, service images, configurations, audit documents, staging documents, data sets, monitors, and MAR info files. For more information, see Project Panel in *Using CA Application Test.*

property

A *property* is a key/value pair that can be used as a run-time variable. Properties can store many different types of data. Some common properties include LISA_HOME, LISA_PROJ_ROOT, and LISA_PROJ_NAME. A configuration is a named collection of properties. For more information, see Properties in *Using CA Application Test.*

quick test

The *quick test* feature lets you run a test case with minimal setup. For more information, see Stage a Quick Test in *Using CA Application Test.*

registry

The *registry* provides a central location for the registration of all DevTest Server and DevTest Workstation components. For more information, see Registry in *Using CA Application Test.*

service image (SI)

A *service image* is a normalized version of transactions that have been recorded in CA Service Virtualization. Each transaction can be stateful (conversational) or stateless. One way to create a service image is by using the Virtual Service Image Recorder. Service images are stored in a project. A service image is also referred to as a *virtual service image* (VSI). For more information, see Service Images in *Using CA Service Virtualization.*

simulator

A *simulator* runs the tests under the supervision of the coordinator server. For more information, see Simulator Server in *Using CA Application Test.*

staging document

A *staging document* contains information about how to run a test case. For more information, see Building Staging Documents in *Using CA Application Test.*

subprocess

A *subprocess* is a test case that another test case calls. For more information, see Building Subprocesses in *Using CA Application Test.*

test case

A *test case* is a specification of how to test a business component in the system under test. Each test case contains one or more test steps. For more information, see Building Test Cases in *Using CA Application Test.*

test step

A *test step* is an element in the test case workflow that represents a single test action to be performed. Examples of test steps include Web Services, JavaBeans, JDBC, and JMS Messaging. A test step can have DevTest elements, such as filters, assertions, and data sets, attached to it. For more information, see Building Test Steps in *Using CA Application Test.*

test suite

A *test suite* is a group of test cases, other test suites, or both that are scheduled to execute one after other. A suite document specifies the contents of the suite, the reports to generate, and the metrics to collect. For more information, see Building Test Suites in *Using CA Application Test.*

think time

Think time is how long a test case waits before executing a test step. For more information, see Add a Test Step (example) and Staging Document Editor - Base Tab in *Using CA Application Test.*

transaction frame

A *transaction frame* encapsulates data about a method call that the DevTest Java Agent or a CAI Agent Light intercepted. For more information, see Business Transactions and Transaction Frames in *Using CA Continuous Application Insight.*

Virtual Service Environment (VSE)

The *Virtual Service Environment (VSE)* is a DevTest Server application that you use to deploy and run virtual service models. VSE is also known as CA Service Virtualization. For more information, see *Using CA Service Virtualization.*

virtual service model (VSM)

A *virtual service model* receives service requests and responds to them in the absence of the actual service provider. For more information, see Virtual Service Model (VSM) in *Using CA Service Virtualization.*