Oracle Application Testing Suite

OpenScript for Load Testing

Script Troubleshooting Hands-on Lab Exercise

Contents

Introduction

This document reviews the script debugging method that is used in the video training, "***OpenScript Load Testing Troubleshooting Training***" session. Click here to access the training in Oracle Learning Library.

The exercise enables you to familiarize with the common debugging methods in OpenScript Load testing module. When you execute a load testing script that is recorded against your application, playback may fail with script error. This exercise document walks you through to understand how to analyze the error messages, compare the content and header information between recording and playback, identify the root cause of the playback problem, and apply custom correlation to resolve the problem.

Setting up Lab Environment

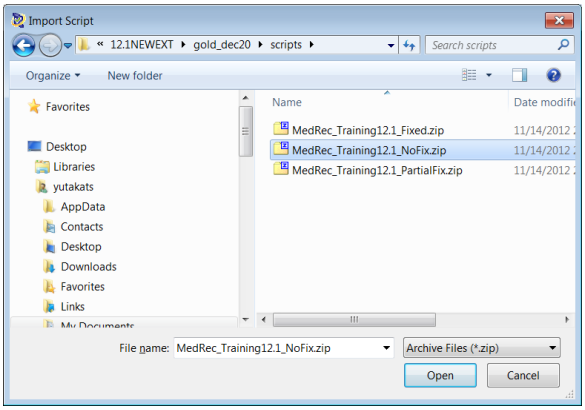
The HOL exercise comes with three sample scripts.

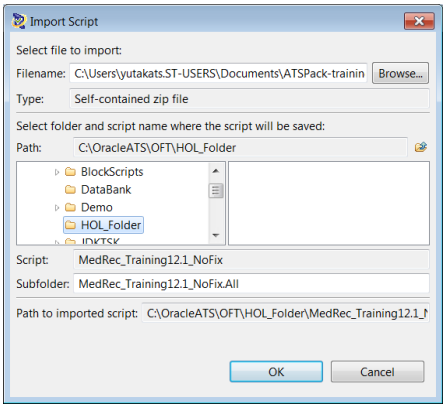
- **MedRec_Training12.1_NoFix.zip:** Script has no fix applied. When executed, it fails at step3.
- **MedRec_Training12.1_PartialFix.zip:** Script has a partial fix (same fix done in the Video training). Passes all steps, but has hidden problems in the content returned.
- **MedRec_Training12.1_Fixed.zip:** Script has the complete fix. When executed, it passes all steps with correct content loaded.

Please use script "***MedRec_Training12.1_NoFix***" for this exercise.

How to Import the script to your OpenScript:

OpenScript menu **File** -> **Import** and select the **MedRec_Training12.1_NoFix.zip** file from your local file systems.





The sample script runs against the Medical Record application that comes with your ATS installation. Please enable the sample application before running the scripts. The script requires OpenScript version 12.1 or higher.

How to enable the Sample Application:

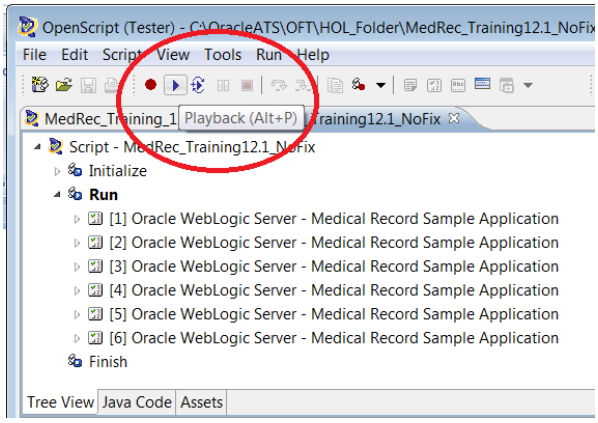
Windows **Start** Menu -> **ALL programs** -> **Oracle Application Testing Suites** -> **Samples** ->**Start Medrec Application**. This will start the Medrec application.

For more details on how to enable the sample application, please see "***OATSGettingStaredGuide***", Chapter 3.1 *Starting the Avitek Medical Records Sample Application***"**.

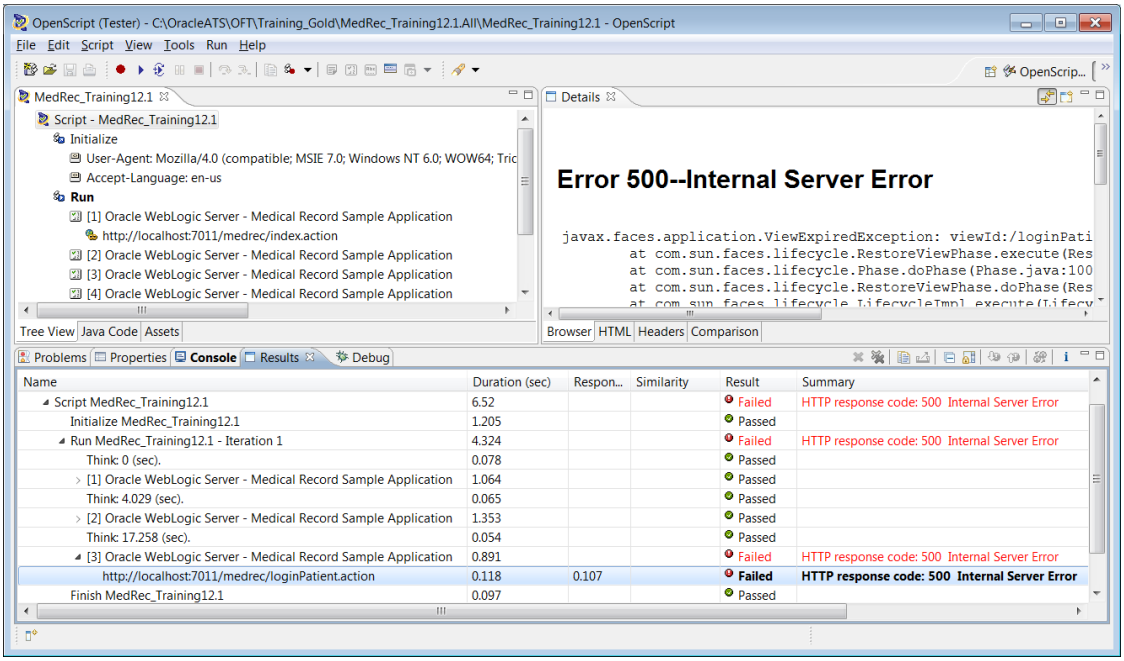
Note: The script runs against your localhost. In some cases, it may not playback if a proxy is set in your OpenScript configuration. When you see this problem, please try removing the proxy settings from **View** -> **OpenScript Preference** -> **Playback** -> **HTTP**-> **Proxy Settings.**

Script Debugging Review

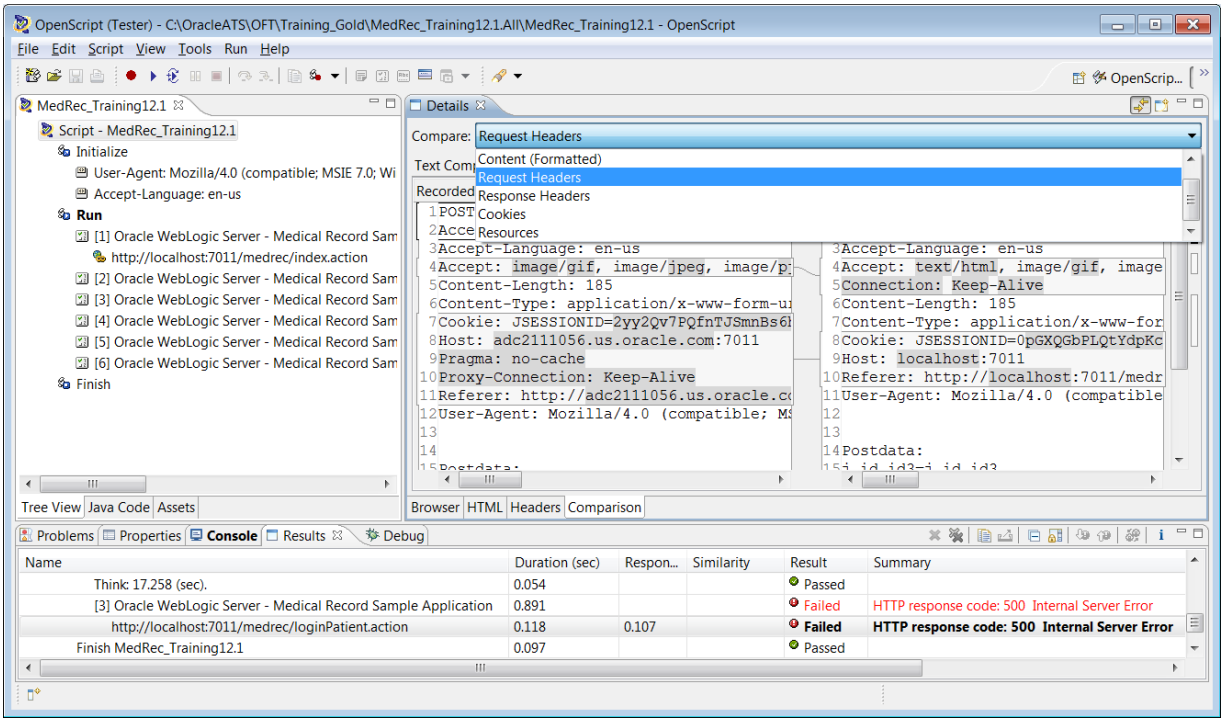
Once the script **MedRec_Training12.1_NoFix.zip** is imported to your OpenScript 12.x, and MedRec application in your local system is enabled, click the playback button on the toolbar to execute the script.



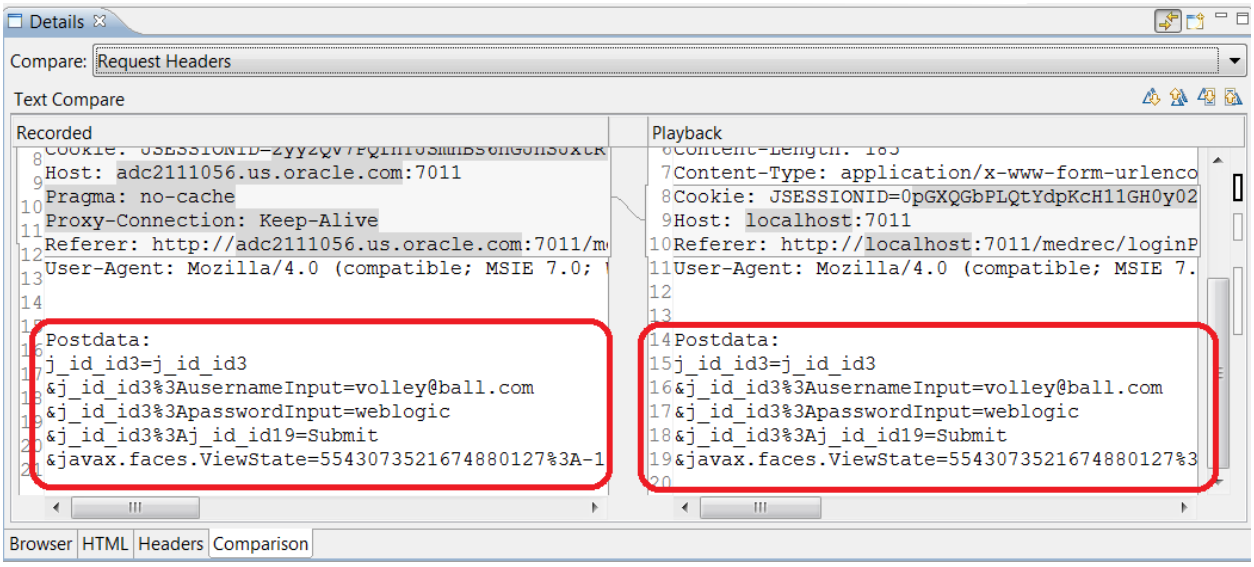
The script playback will execute the navigations and eventually fail with the error message "**HTTP response code 500 Internal Server Error**". This is the problem you will resolve in this exercise.



Select the Failed node in the **Result** view (bottom pane), and go to the **Details** view (upper right hand pane). Click the **Comparison** tab. This feature allows you to compare content, headers, cookies or resources information between recorded and playback, to help you identify the root cause of the navigation failure. Select **Request Headers** from the pull-down menu.

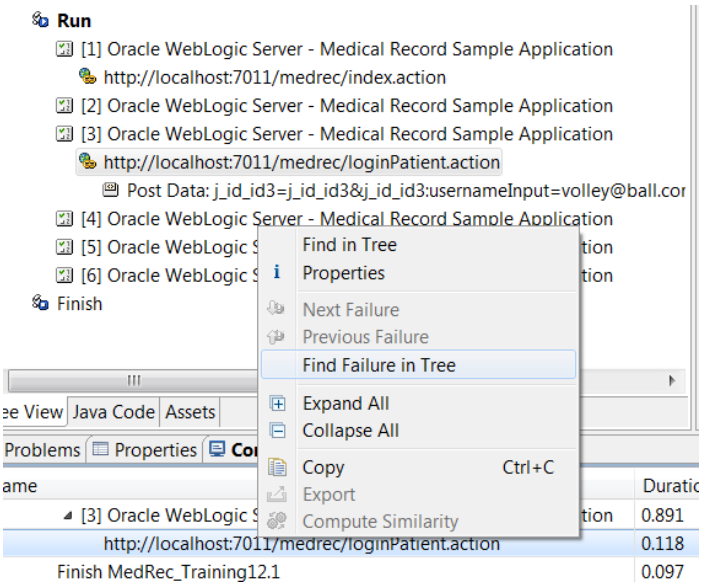


Scroll down to where you can see the Postdata parameters. Session IDs have the same values for recorded and playback. If the application expects different values for each playback for these session IDs, then the navigation may fail, as the values used in the script are no longer valid during playback.

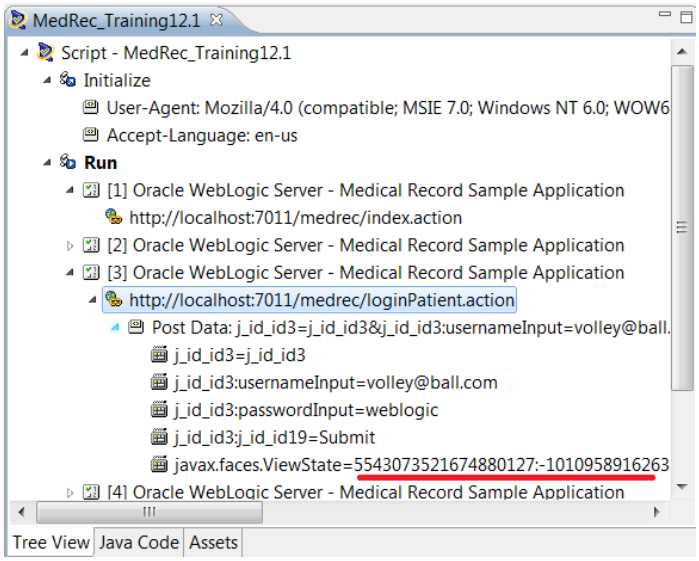


How to fix this problem? We will need to correlate the dynamic parameters so that they will use different values each time during playback, instead of the recorded data.

Once again select the failed node in the **Result** view, and right mouse click to select "**Find Failure in Tree**". This will highlight the corresponding navigation node in the **Script** view.



Expand the highlighted navigation in the **Script** view to display the PostData parameters. You will see the parameter values are hardcoded. The values may need to be correlated, in order to receive a correct response from the server.



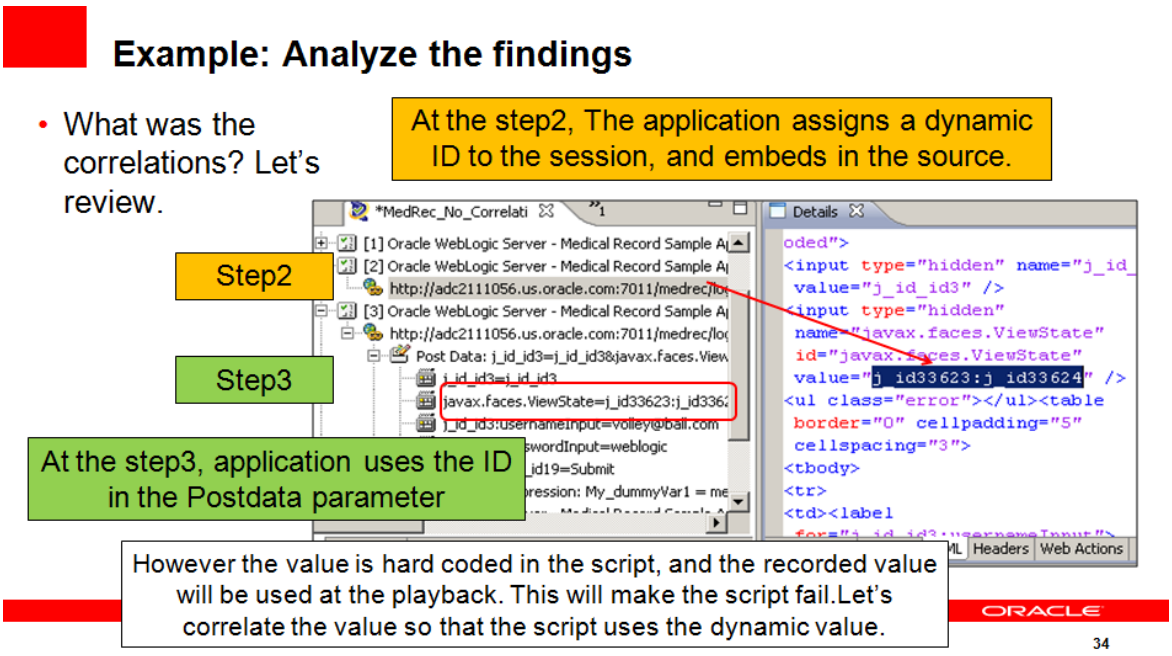
What are the Correlations in Load scripting? Let's review.

Correlations are required in OpenScript load scripts, when a server in AUT (application under test) exchanges dynamic session values with the browser. OpenScript tries to auto-correlate the session IDs. However, in some cases manual correlation is required.

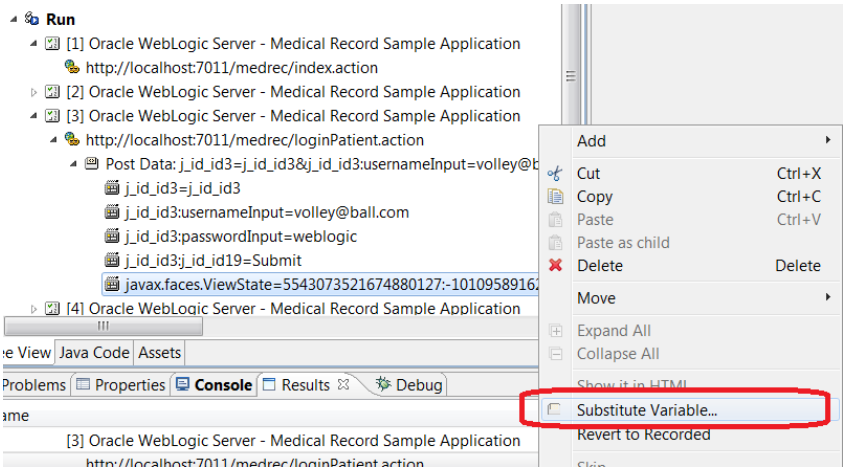
In the sample application, the server applies a dynamic session ID by embedding the value in an input type hidden field in the contents (step 2). The client then needs to send back the same session id to the server to notify who is sending the request, in the next navigation (step 3). The problem is that the session ID in step 3 is hard-coded, and uses the same value every time the script plays back.

As the server receives the WRONG value during playback, it cannot identify the sender of the request. That is why the script has a 500 Internal Server Error from the server.

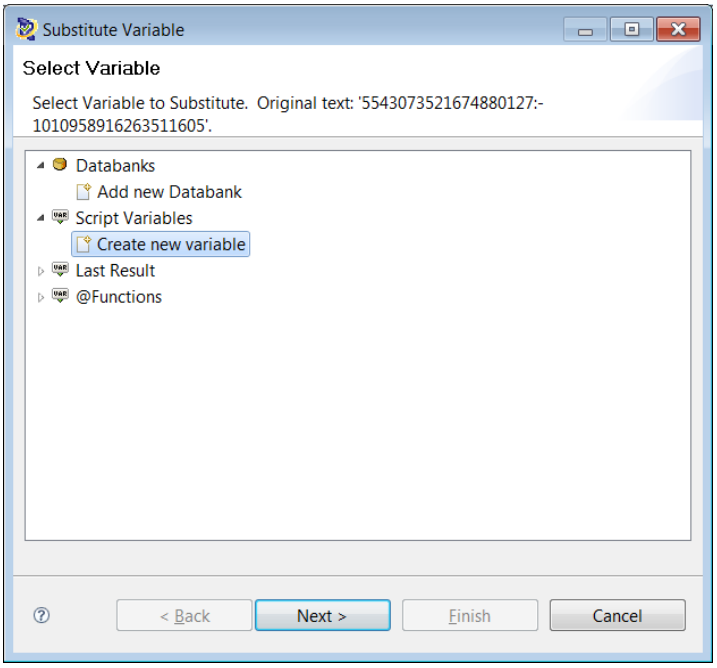
To fix this problem, we will need to manually correlate the parameter, so that it will pick up the dynamic value each time the script plays back.



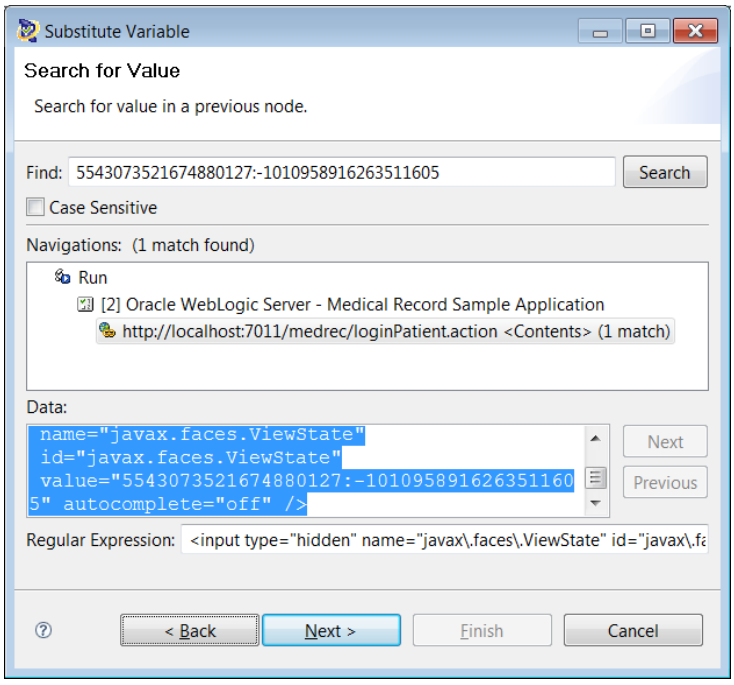
In the script view, select the PostData parameter you want to correlate. In this example, select the parameter "**javax.faces.ViewState**". Then right mouse click to select "**Substitute Variable**".



"**Substitute Variable**" dialog opens. Select "**Create New Variable**" and click **Next**.



"**Search for Value**" dialog opens. You can search the contents of the earlier navigations, and specify where in the source you want to extract the dynamic value that you will apply to a "solve" variable that will be created later in the steps. You can also specify regular expression to retrieve the value at run-time during the playback.



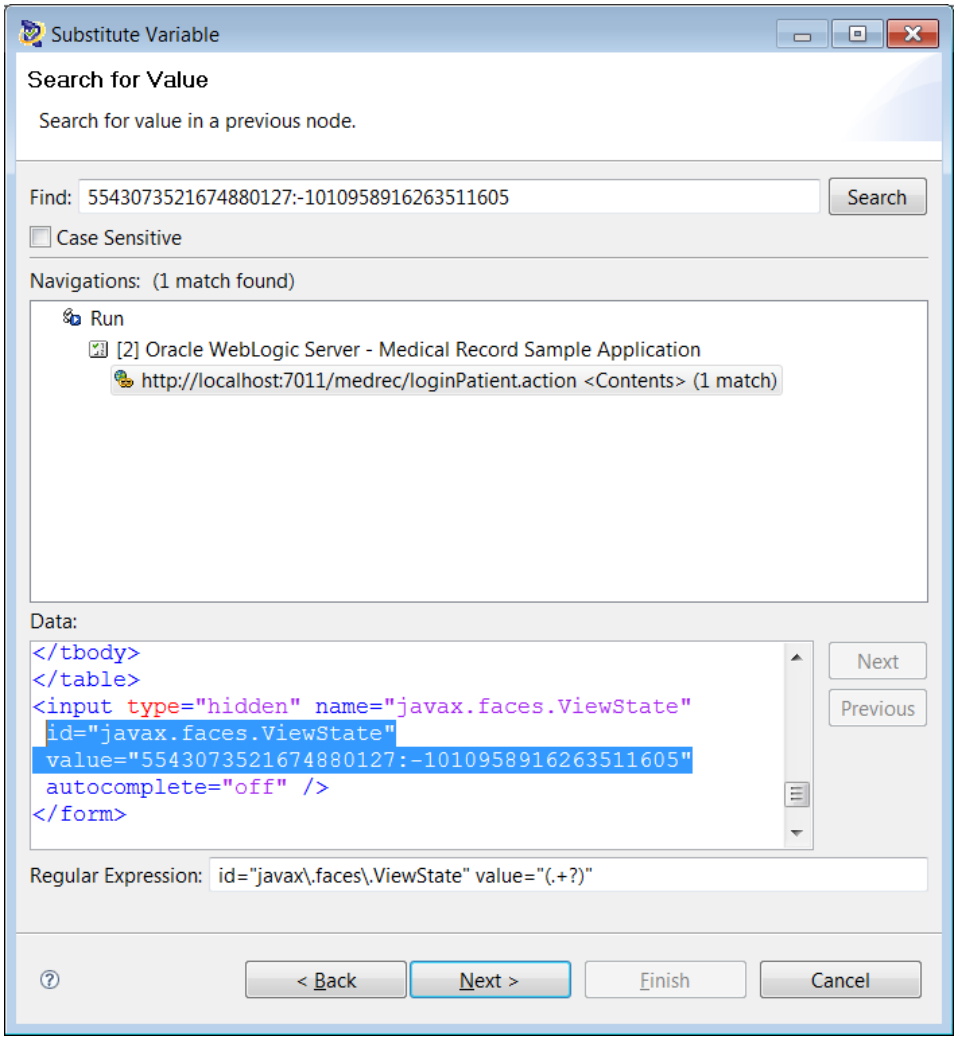
Don't know the regular expression syntax? Don't worry as OpenScirpt does auto-search and suggests a default regular expression for you. Here is what OpenScript suggested:

<input type="hidden" name="javax\.faces\.ViewState" id="javax\.faces\.ViewState" value="(.+?)"

I can accept the regular expression suggested, but I think it can be shorter to get the same result. So, I will modify the regular expression to the following.

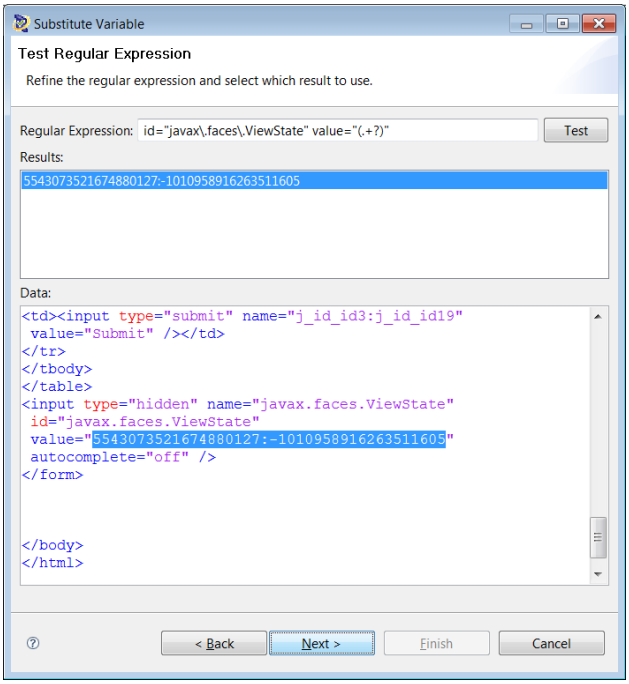
id="javax\.faces\.ViewState" value="(.+?)"

I can either edit the Regular Expression itself, OR to simply re-select the content in the "**Data**" window, to auto create a regular expression.

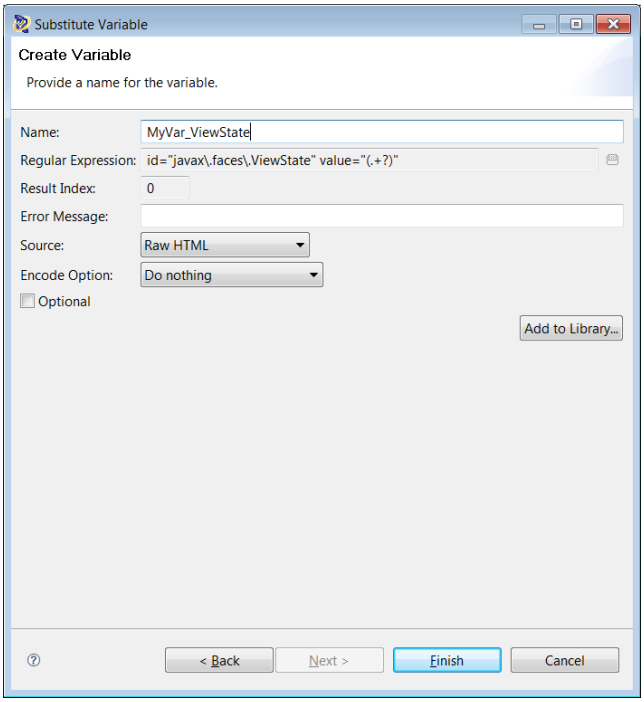


Click Next to go to the next dialog, "**Test Regular Expression**".

In the "**Test Regular Expression"** dialog, you can test out the Regular Expression you created in the previous step. Click the **Test** button, to see whether the results come up as a single correct value.

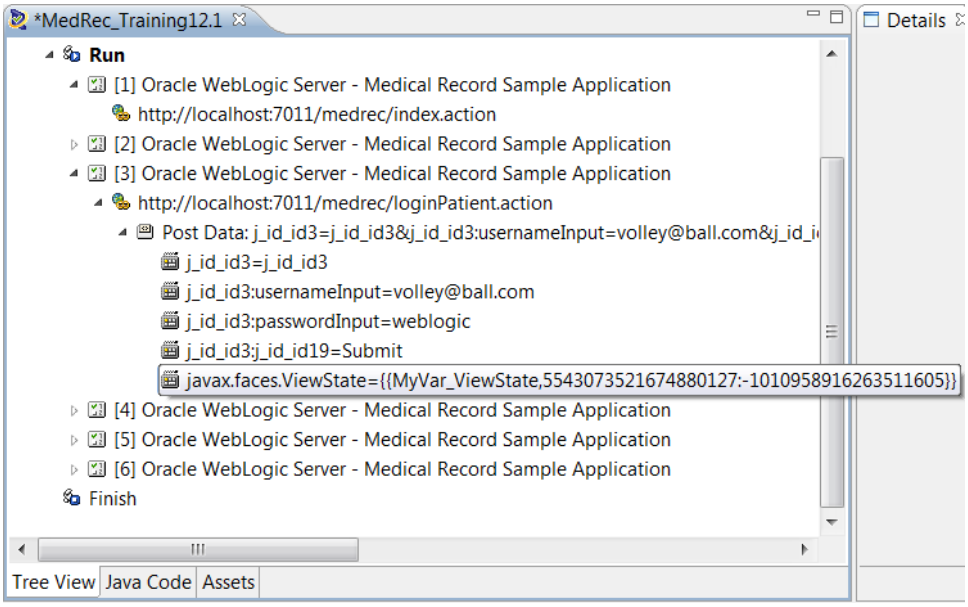


If you see multiple values with different results, then you will need to go back to the previous dialog to redefine the regular expression. This is because it was not robust enough to identify the value uniquely from the source. Make sure the test shows the satisfied result, and click **Next** to show "**Create Variable"** dialog.

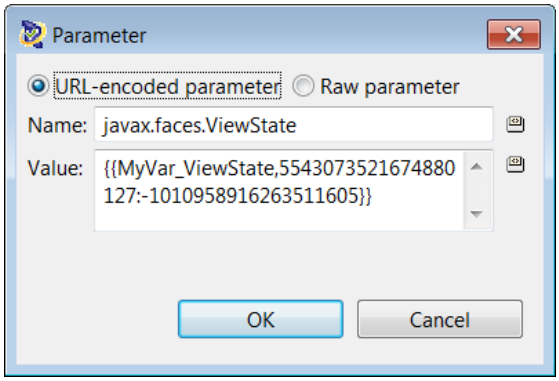


In this example, I will name the variable "**MyVar_ViewState**", and click **Finish**.

Now the variable is created and applied to the selected parameter. Please go to the **Script** view, and see the navigation in Step 3.



Double click the node to see the parameter details.



Value before the correlation:

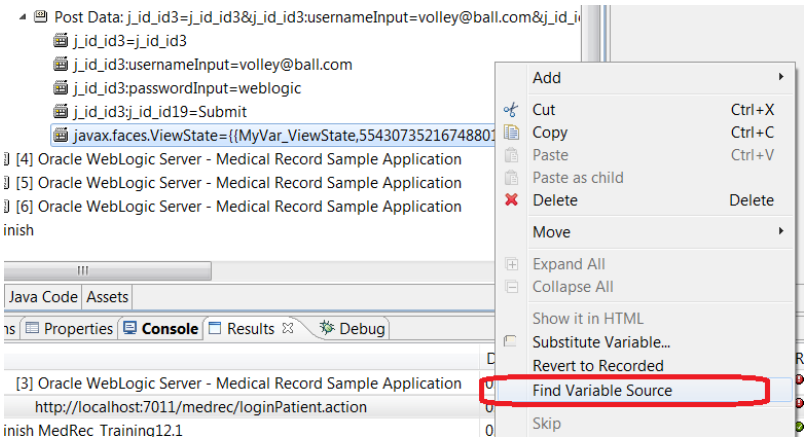
5543073521674880127:-1010958916263511605

Value after the correlation:

{{MyVar_ViewState,5543073521674880127:-1010958916263511605}}

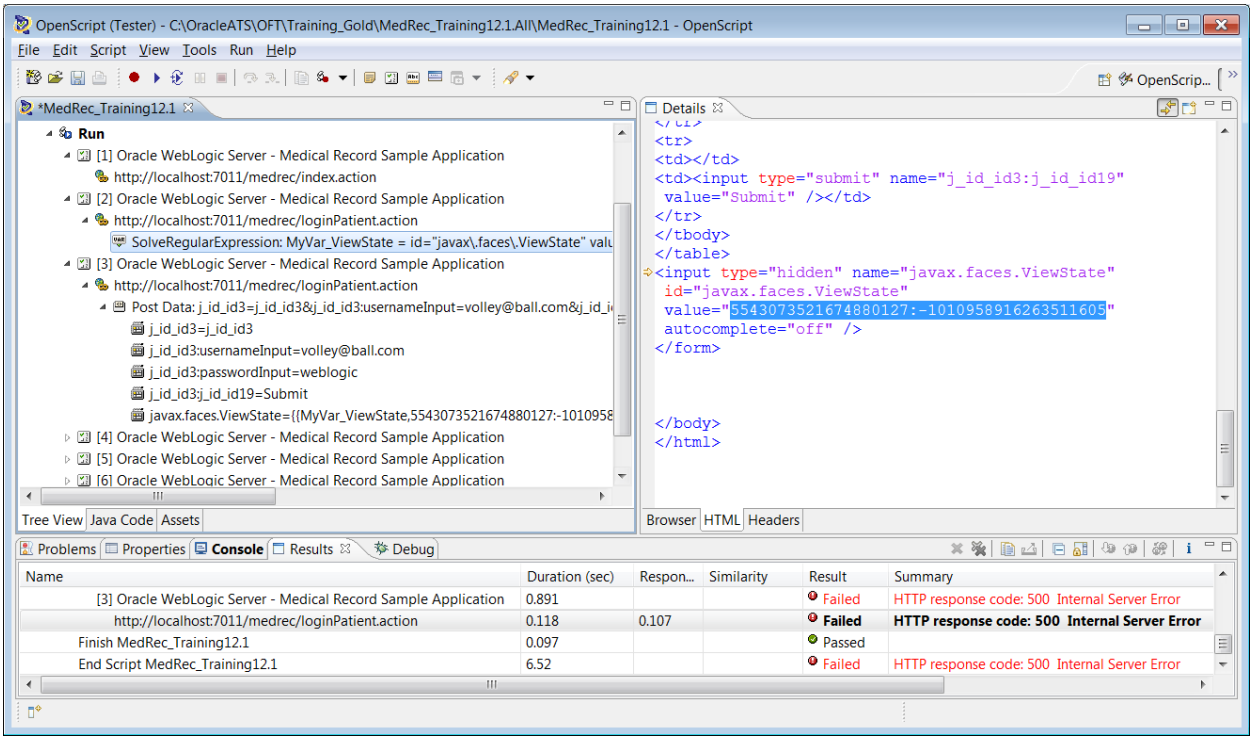
Curly brackets indicate that the value is a variable. The variable Name is **MyVar_ViewState**, and the value after comma is a recorded value. The next time this script is executed, it will not use the recorded value for this parameter, but will use the value that is retrieved from the source, using the regular expression you specified & tested in the earlier steps.

Let's check where the value in the variable is coming from. Select the parameter node, right mouse click to select "**Find Variable Source**"

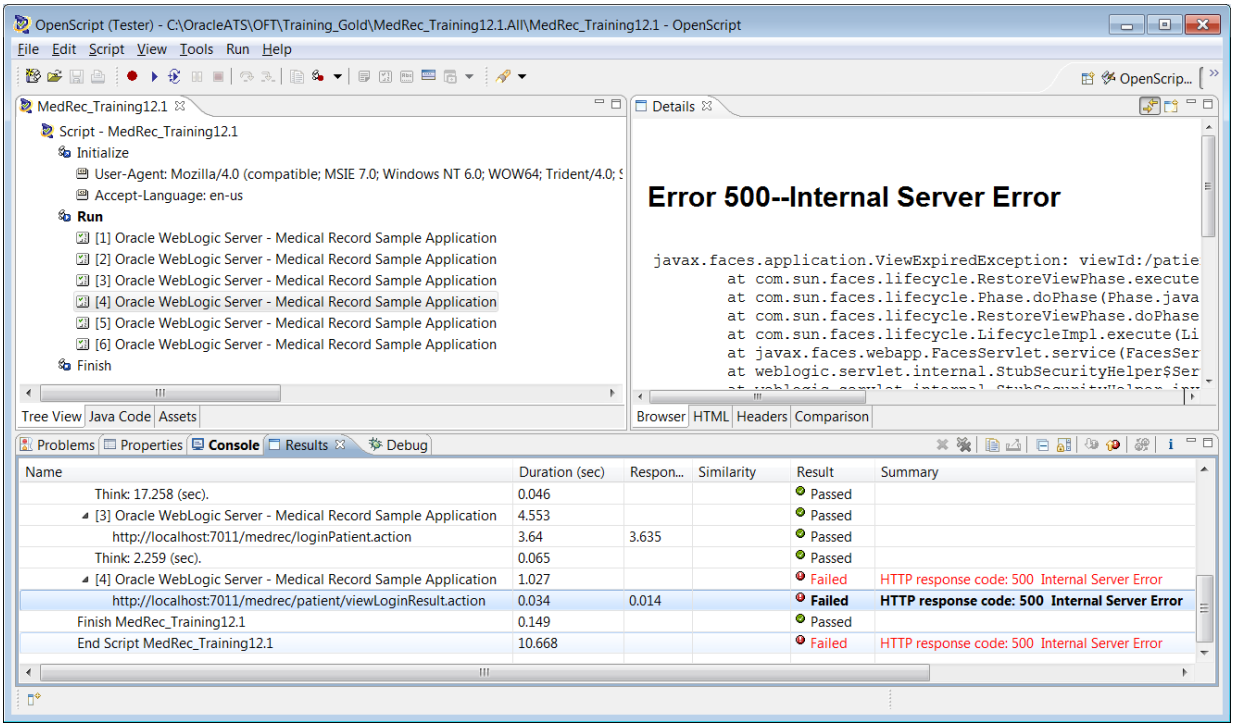


This will highlight the corresponding node in step2, which has a variable declaration that was applied to the parameter in step3. You will see a new SolveRegularExpression variable, **MyVar_ViewState**, is created under the navigation in step2.

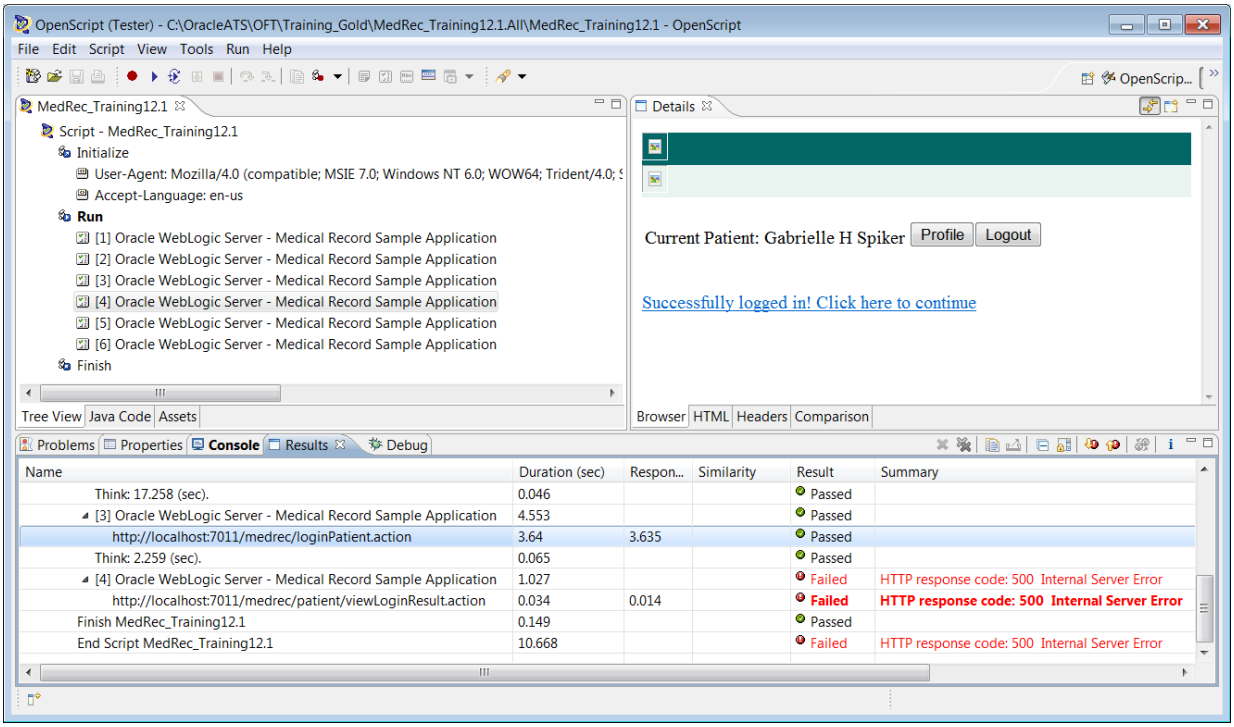
Select the **HTML** tab in the **Details** view, and re-select the SolveRegularExpression created in step2. The text highlighted in the HTML content is where the variable is retrieving the value from the source, and applying to the variable.



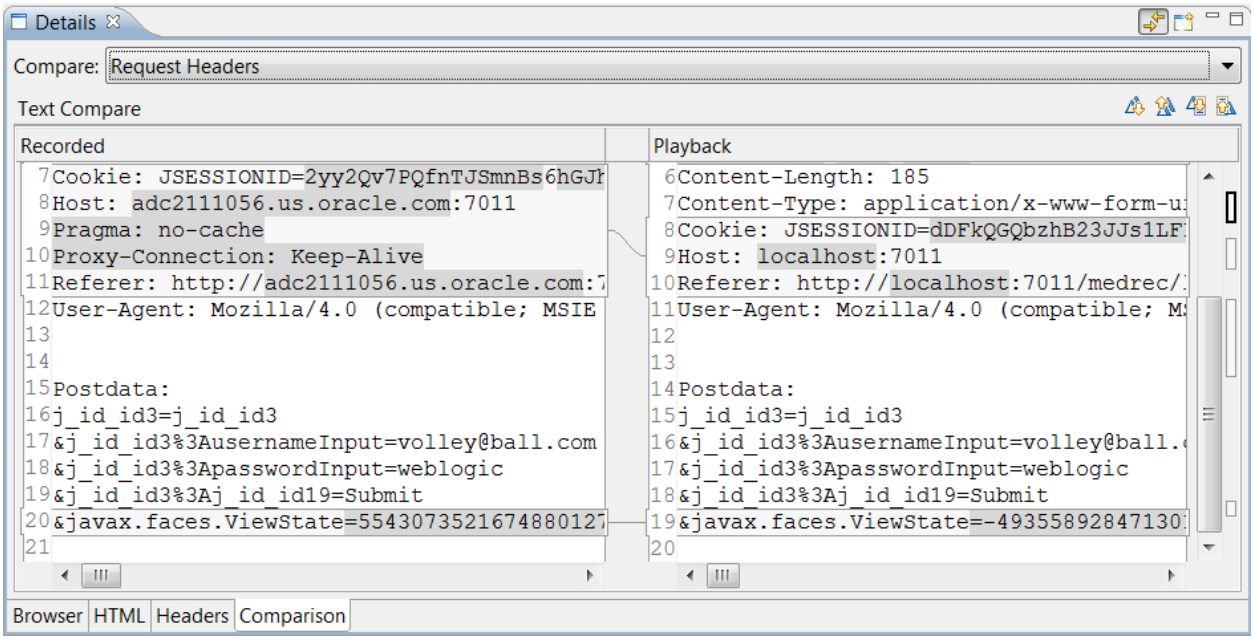
Go ahead and playback the script to verify the fix applied. The script will pass Step 3, which is good news, however it will fail at Step4 with the same error message we saw before. **HTTP response code 500, Internal Server Error**.



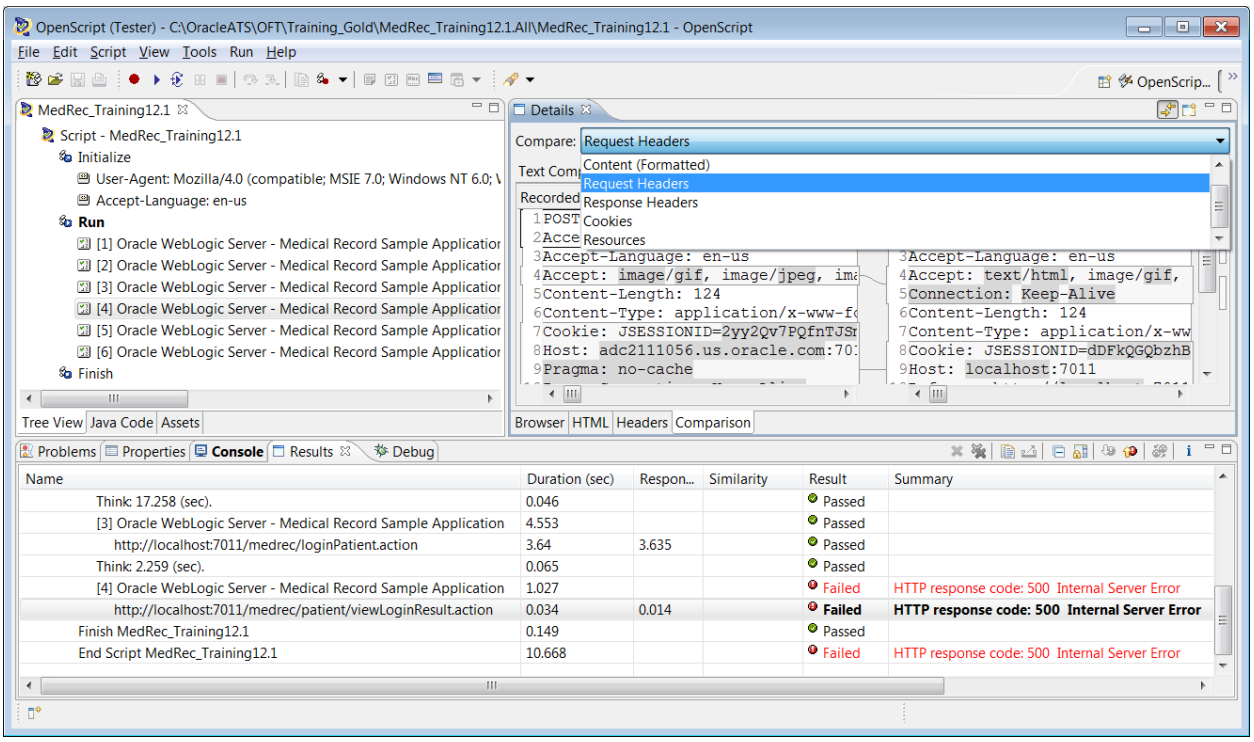
First let's check the result for Step 3. Select the navigation in Step3 and click the **Browser** tab in the **Details** view. Verify that the correct browser content is loaded.



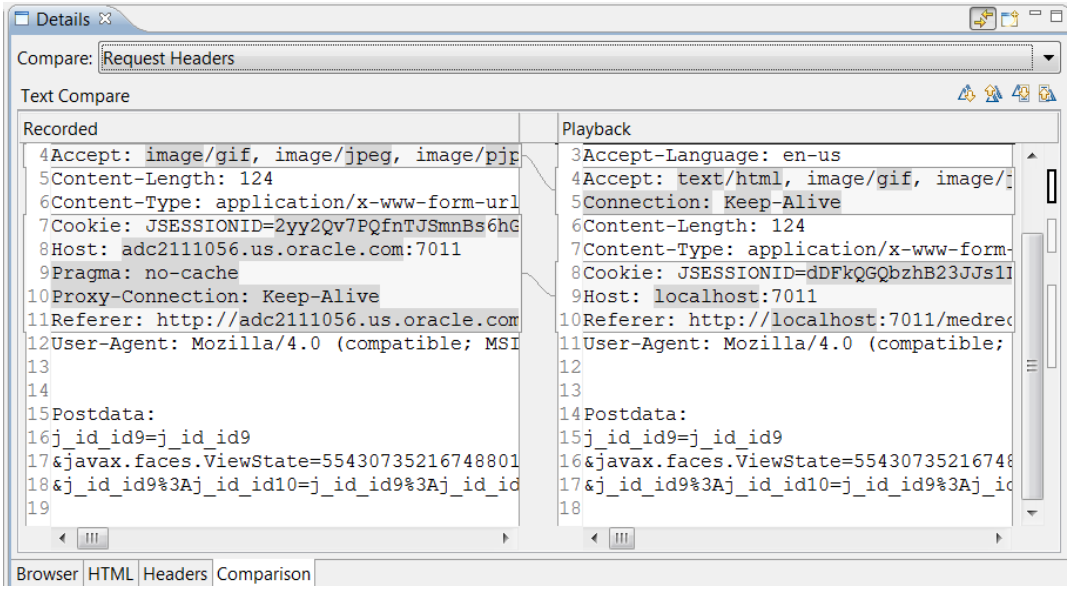
Then click the **Comparison** tab. Select **Request Headers** from the pulldown menu and scroll down to where it displays the Postdata parameters. Now, javax.faces.ViewState parameter is correlated. So we at least fixed the problem in step3.



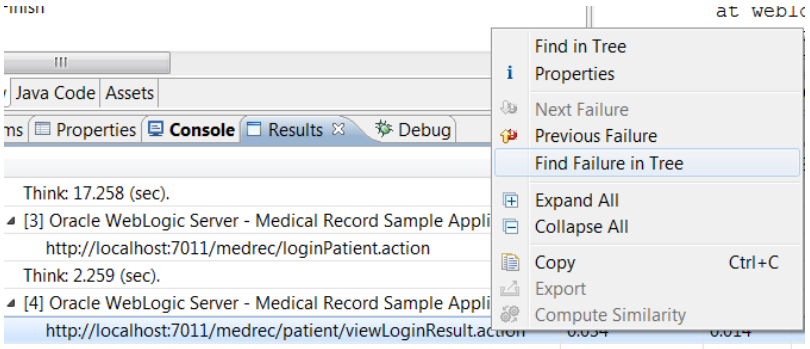
Now let's check what went wrong with Step4. What we need to do is the same practice. Select the Failed node in the **Result** view, and check the **Comparison** tab in the **Details** view.



Scroll down to see the Postdata parameters. OK, you see what is the problem here. The ViewState parameter is again hardcoded. We will need to apply the same solution as we did to step3.

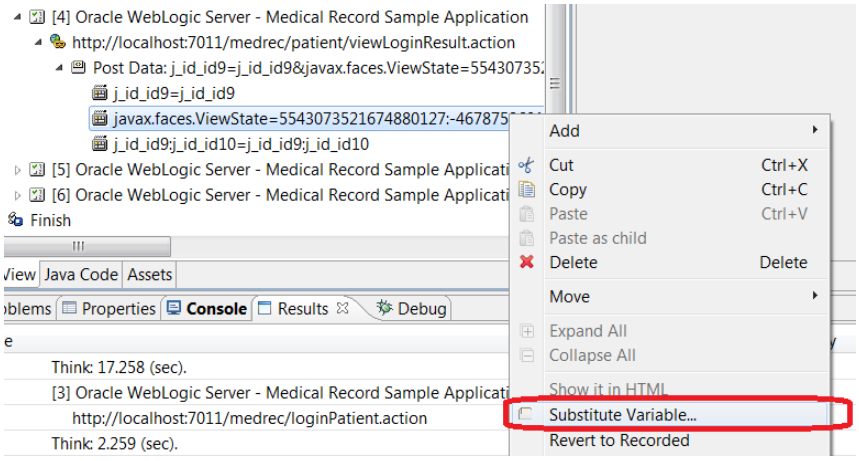


Select the failed node in the **Result** view, right mouse click and select "**Find Failure in Tree**". This will highlight the corresponding navigation node in Step 4 of the **Script** view.

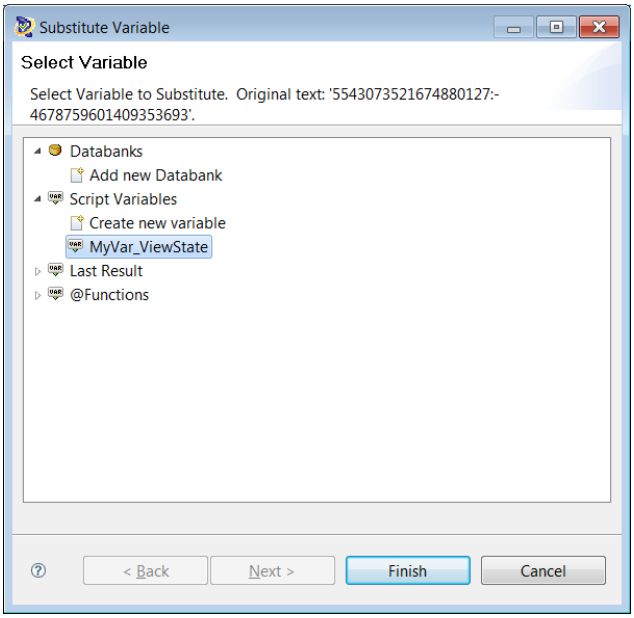


Expand the navigation to display the Postdata parameters. As expected, session IDs are not correlated.

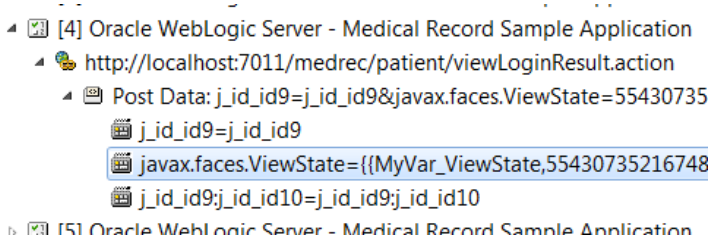
We will need to apply the same solution for ViewState parameter as we did for Step 3. Select the parameter node, right mouse click to select "**Substitute Variable**".



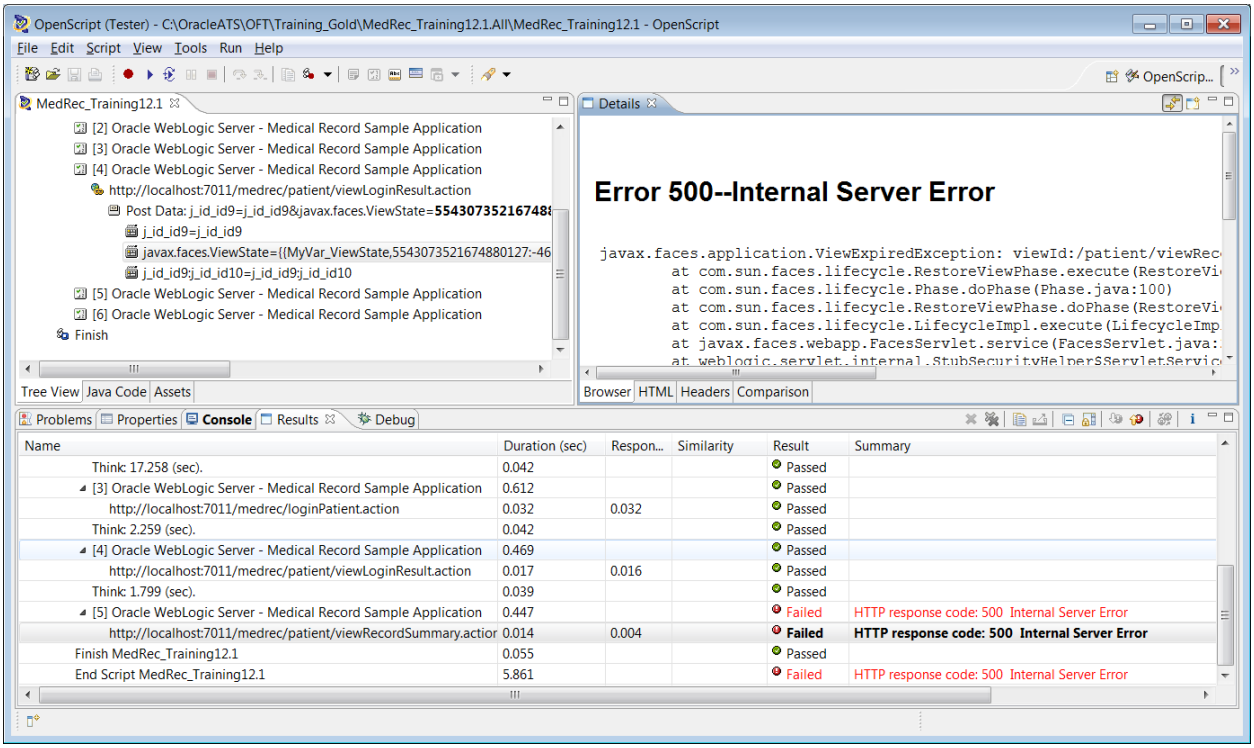
"Substitute Variable" dialog opens. You can create a new variable, but at this time we already have created a variable for ViewState parameter. You can find a custom parameter "**MyVar_ViewState**" shows up in the treeview.



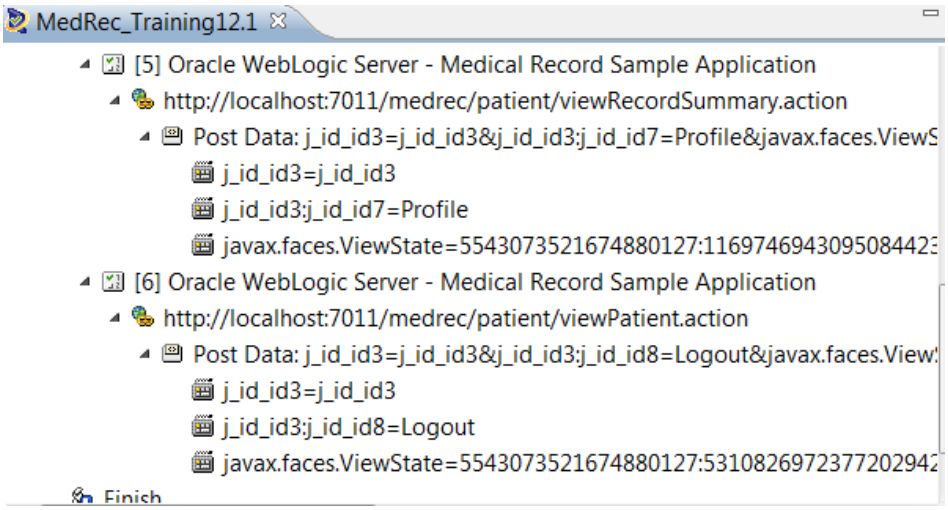
I "think" I can use this variable, so I will select the "**MyVar_ViewState**" node and click **Finish** to close the dialog. Confirm the parameter is correlated in the script view.



Playback the script again, and verify now it passes Step 4… however it fails in Step 5 this time.

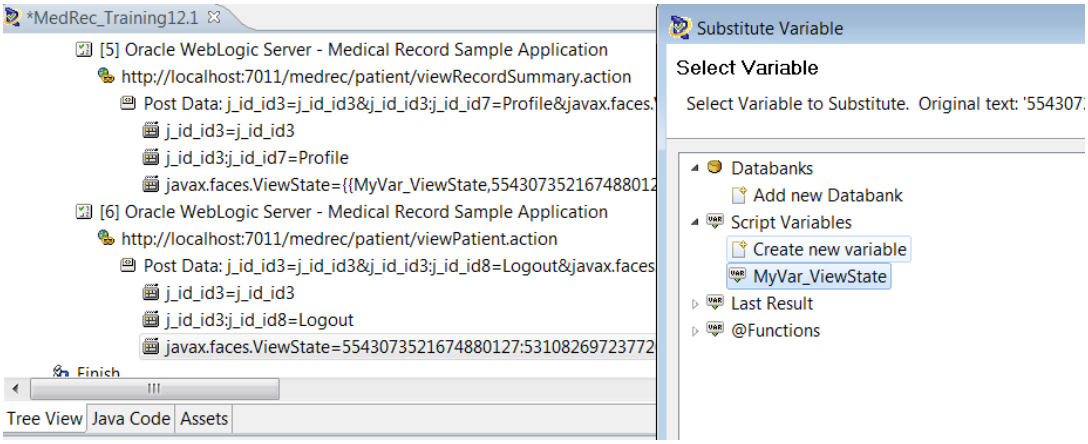


Expand the nodes in Step 5. The ViewState parameter is not correlated. Expand Step 6 to see that it has the same situation here as well.

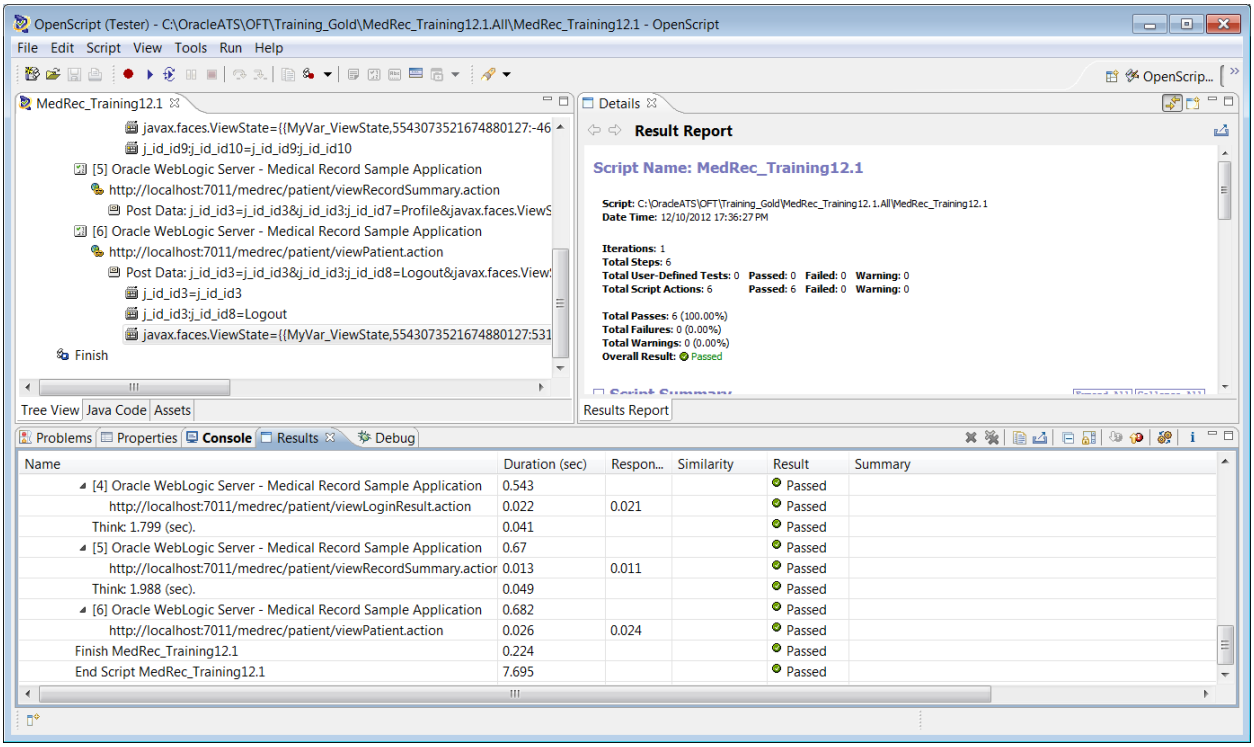


Now you have an idea what needs to be done next. We will need to correlate ViewState parameters for Step 5 and Step 6 too!

Select the parameter node in Step 5, right mouse click to select "**Substitute Variable**" to open the dialog. Select the custom parameter "**MyVar_ViewState**" and click **Finish**. This will correlate the ViewState parameter in Step 5. Repeat the same steps for Step 6.



Playback the script. Now all steps have passed status. Finally, the script execution was successful.

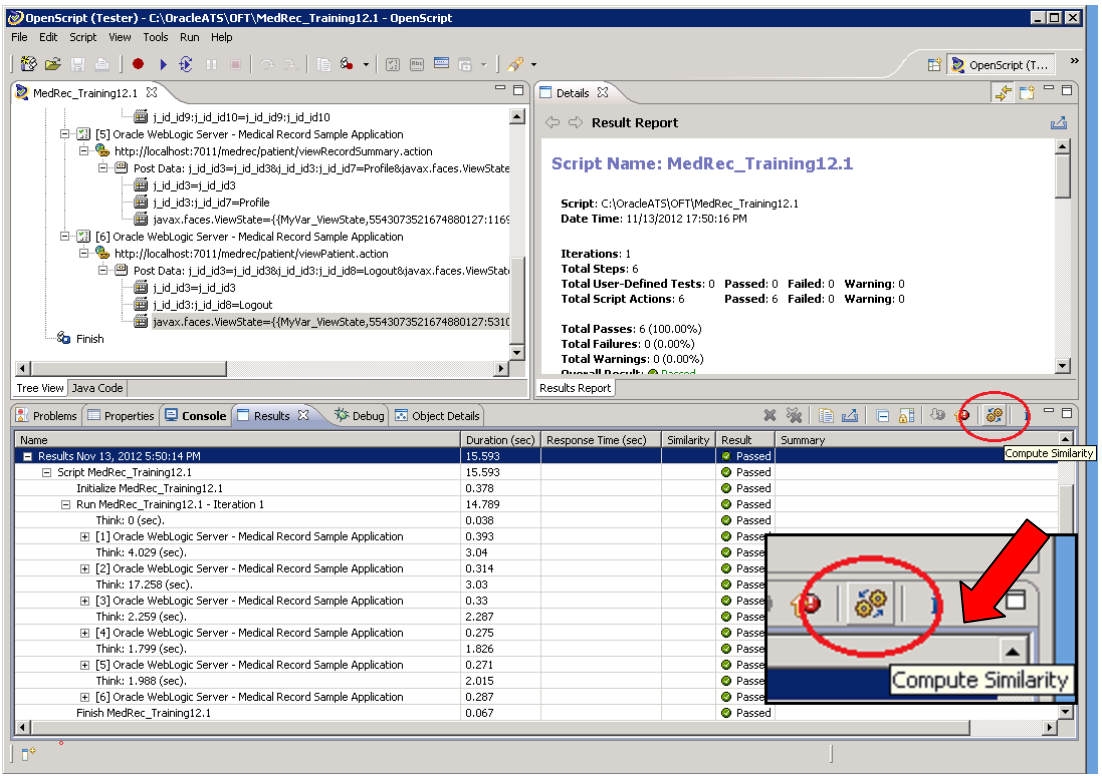


However, the correlation problem in the script has not been completely fixed yet. Although the playback result has "Passed" result for all navigations, there are still some hidden problems remaining in the script. Please see the next section "**Additional Script Debugging Method**" to work with this problem and fix your script completely.

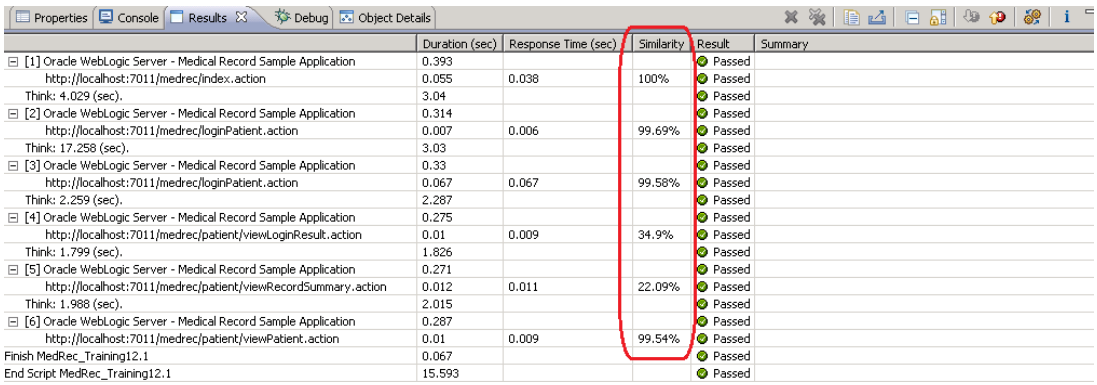
Additional Script Debugging Method

In fact, the solution applied in previous section, (and in the video training) did not completely resolve the correlation problem for the sample script yet. Although the script result has "**Passed**" status for all navigations, some hidden problems still need to be manually fixed. If you are starting from this section, please use script "***MedRec_Training12.1_PartialFix***" for this exercise.

In the sample script, select the top node in the playback result for the passed session, and click "**Compute Similarity**" button from the tool bar in the Results view to verify the similarities between the recorded and playback contents returned from the server. This feature provides additional verification to make sure the pages with passed result are truly passed.

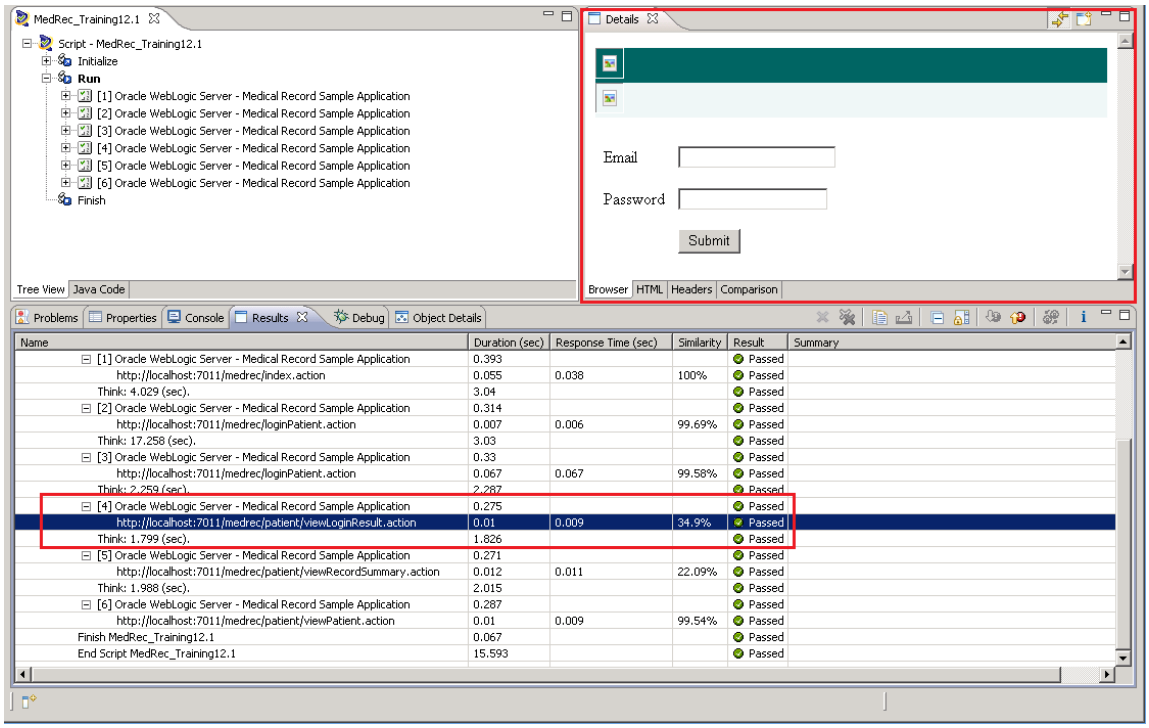


Similarities will be displayed for each of the playback navigations. In the sample script, navigations in steps 4 and 5 are showing large differences between recording and playback.

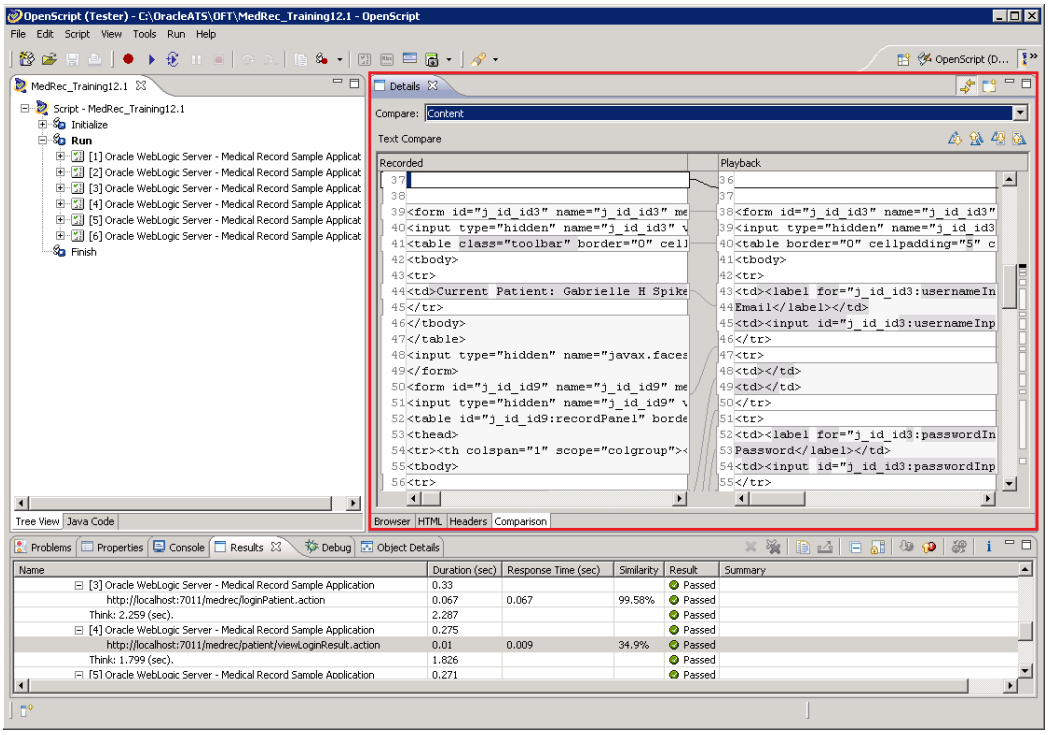


	Duration (sec)	Response Time (sec)	Similarity	Result	Summary
[1] Oracle WebLogic Server - Medical Record Sample Application	0.393			Passed	
http://localhost:7011/medrec/index.action	0.055	0.038	100%	Passed	
Think: 4.029 (sec).	3.04			Passed	
[2] Oracle WebLogic Server - Medical Record Sample Application	0.314			Passed	
http://localhost:7011/medrec/loginPatient.action	0.007	0.006	99.69%	Passed	
Think: 17.258 (sec).	3.03			Passed	
[3] Oracle WebLogic Server - Medical Record Sample Application	0.33			Passed	
http://localhost:7011/medrec/loginPatient.action	0.067	0.067	99.58%	Passed	
Think: 2.259 (sec).	2.287			Passed	
[4] Oracle WebLogic Server - Medical Record Sample Application	0.275			Passed	
http://localhost:7011/medrec/patient/viewLoginResult.action	0.01	0.009	34.9%	Passed	
Think: 1.799 (sec).	1.826			Passed	
[5] Oracle WebLogic Server - Medical Record Sample Application	0.271			Passed	
http://localhost:7011/medrec/patient/viewRecordSummary.action	0.012	0.011	22.09%	Passed	
Think: 1.988 (sec).	2.015			Passed	
[6] Oracle WebLogic Server - Medical Record Sample Application	0.287			Passed	
http://localhost:7011/medrec/patient/viewPatient.action	0.01	0.009	99.54%	Passed	
Finish MedRec_Training12.1	0.067			Passed	
End Script MedRec_Training12.1	15.593			Passed	

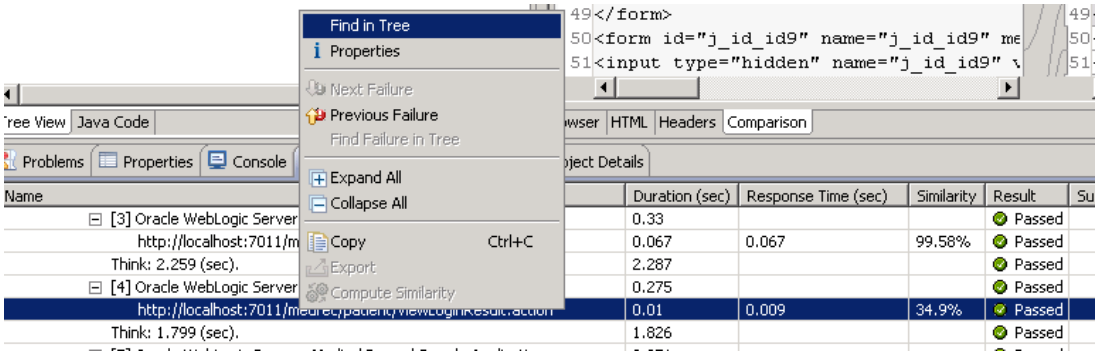
In the Result view, select the navigation that has differences, and check the Details view. The browser tab is showing a Login Page. This is what the server returned during playback. Is this the expected page..?



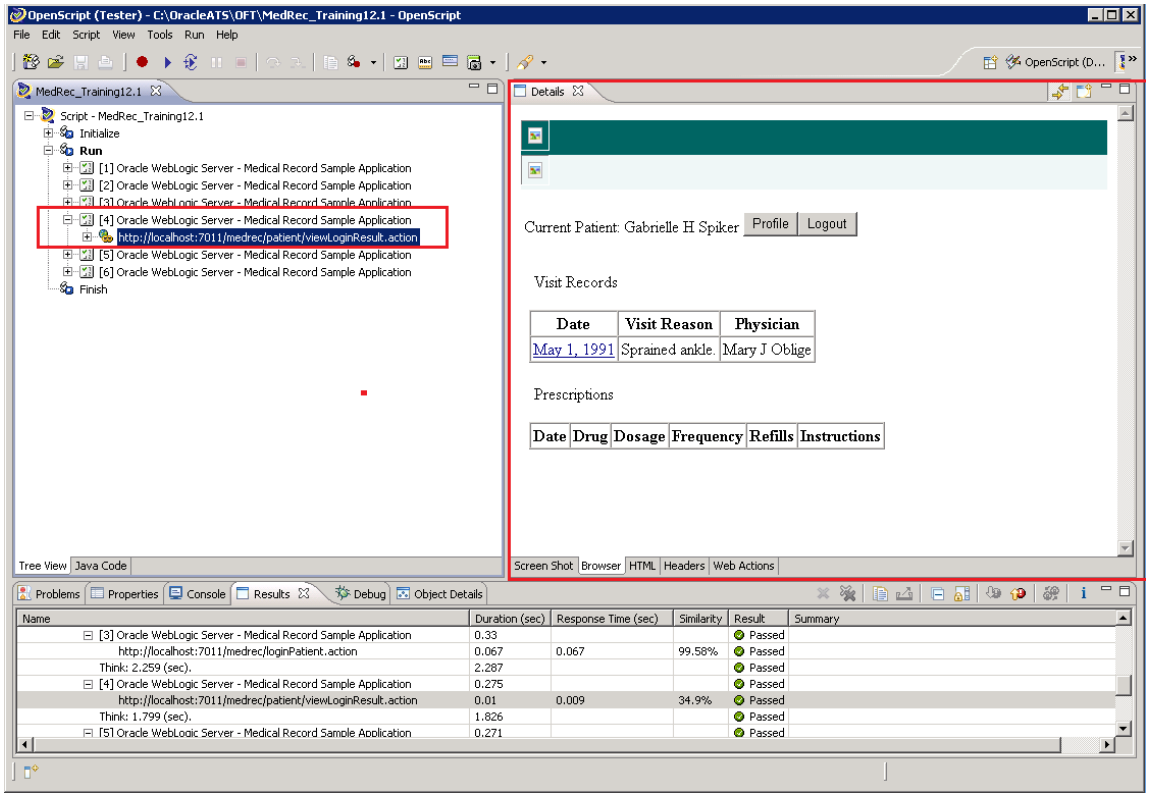
Click the Comparison Tab, and select "**Content**" from the pulldown menu. The differences can be seen between the recorded and playback sessions.



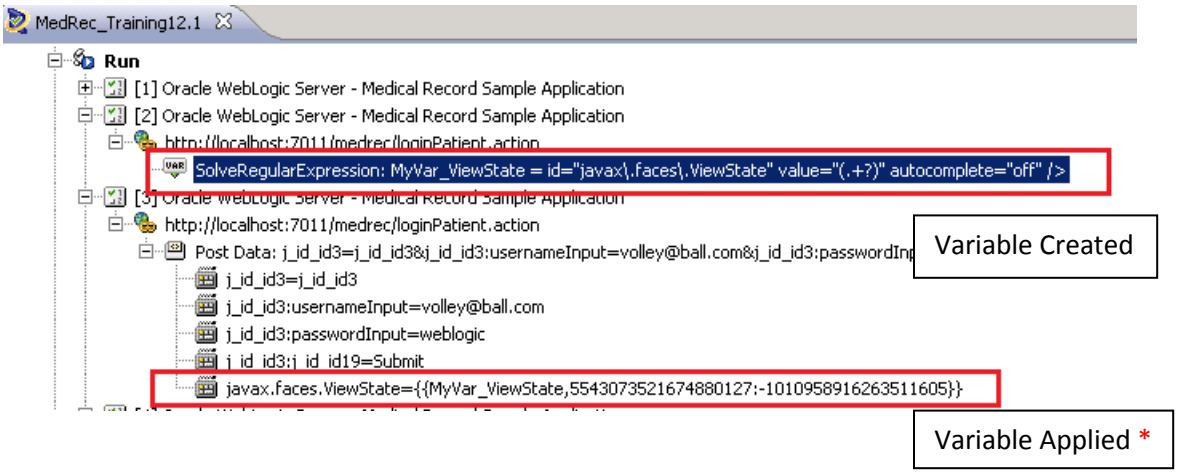
Next, let's check what we were supposed to receive, by looking at the recorded data. Select the node in the playback result, right mouse click to select "**Find in Tree**". This will highlight the corresponding navigation in the script view.



Select the node in the script view, and look at the Details view's Browser's tab. This is what we were supposed to see during playback as well. The page displayed in below screenshot looks quite different from what we had in the playback. Although the result status of the script playback is "**Passed**", problems may be still exist in this script.

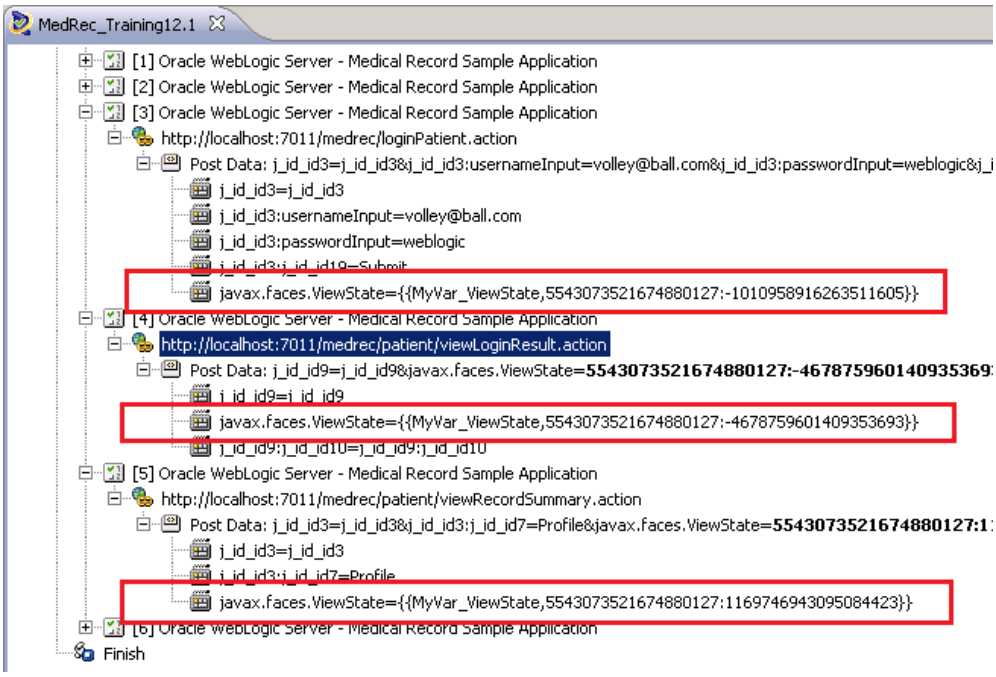


Let's review what we did to fix this script. We created a correlation variable for the post data parameter, "**ViewState**", at step 2, and applied the same variable for the rest of the steps: 3, 4, 5 and 6.



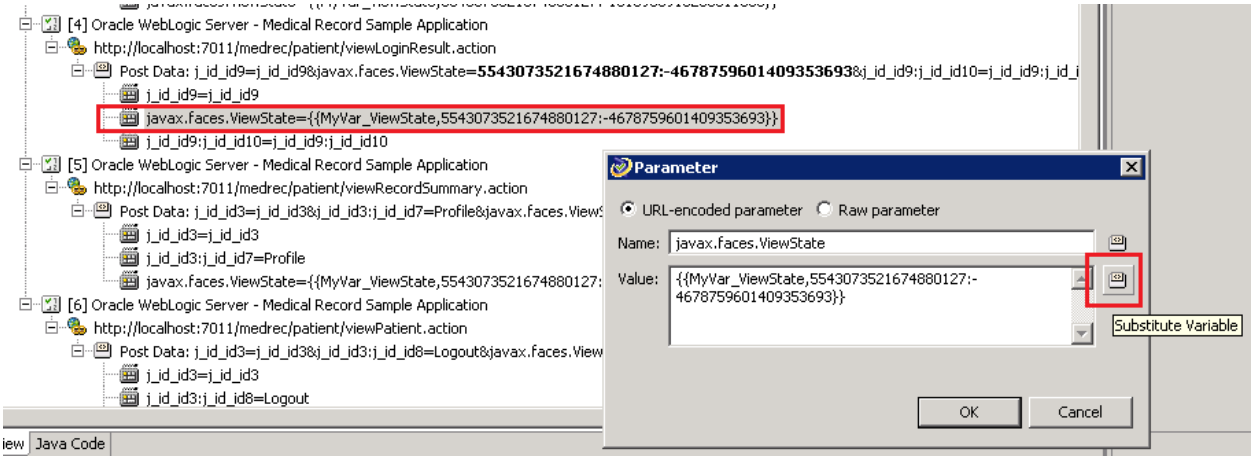
However, looking closely at the recorded value for the VIewState parameter, they have different values for each navigation. *Note: Display format of the variable is: {{variable_name, recorded_value}}

- Step3: 5543073521674880127:**-1010958916263511605**
- Step4: 5543073521674880127:**-4678759601409353693**
- Step5: 5543073521674880127:**1169746943095084423**
- Step6: 5543073521674880127:**5310826972377202942**

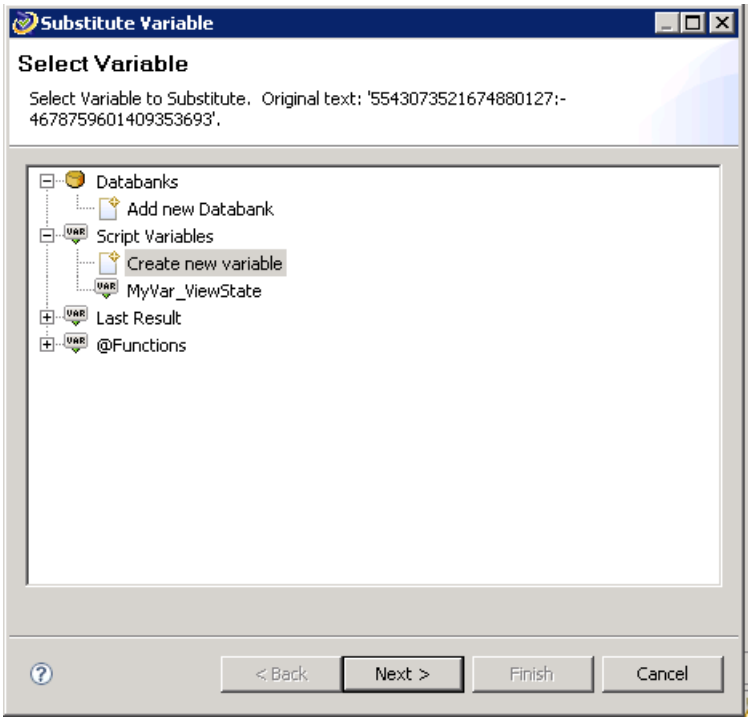


The problem occurred because we applied the same variable (which has the same value) to all the navigations. To fix this, we will need to create and apply different variables for each navigation. Step 3 has a high similarity percentage (>99%). The browser page displayed is the same between recording and playback, therefore we know the variable substitution for ViewState parameter is correct. The ViewState parameter, however, would need to be fixed for Steps 4, 5, and 6.

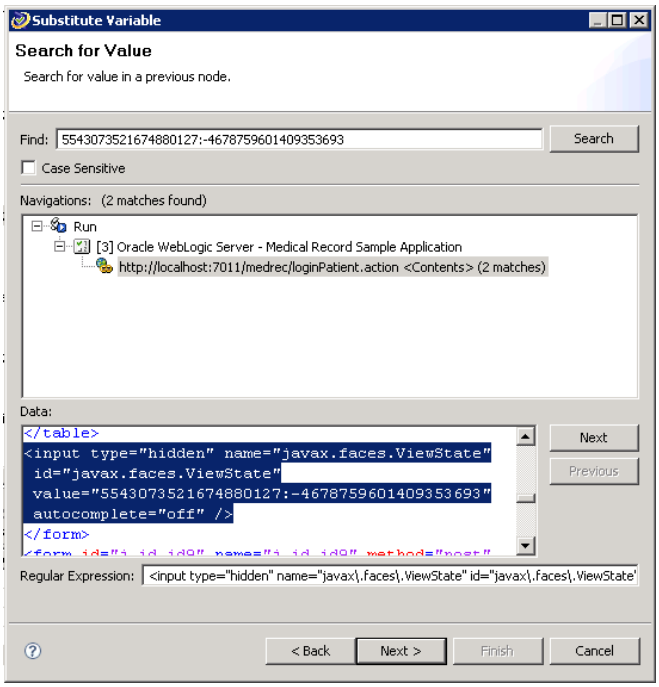
From the script view, select the navigation in Step 4, double click to open Properties dialog, and click "**Substitute Variable**" icon.

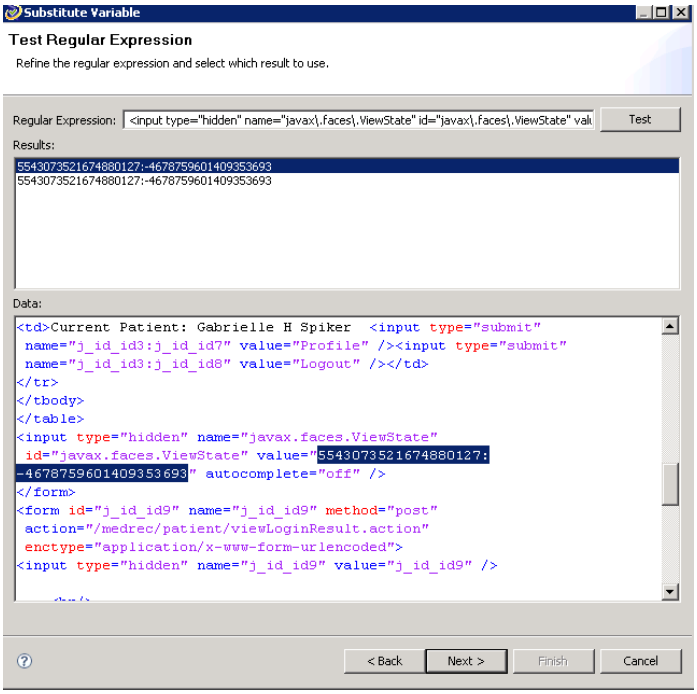


Select "**Create new Variable**" from the Script Variables node. Click Next.

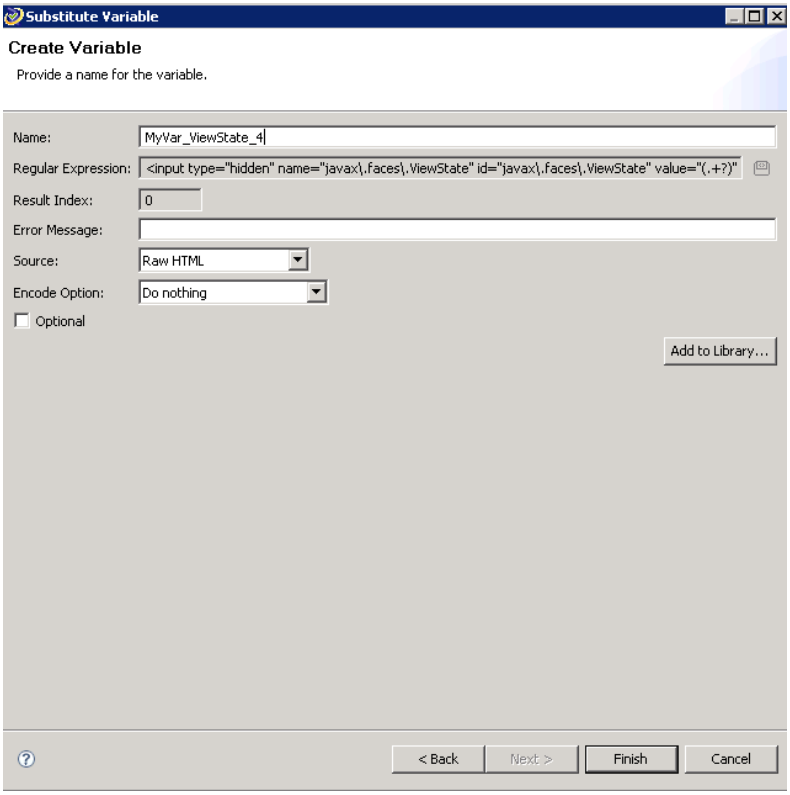


In the “**Search for a Value**” dialog, accept the default Regular expression OpenScript suggested, and click Next. Make sure the regular expression returns the expected value in the “**Test Regular Expression”** dialog that opens next.

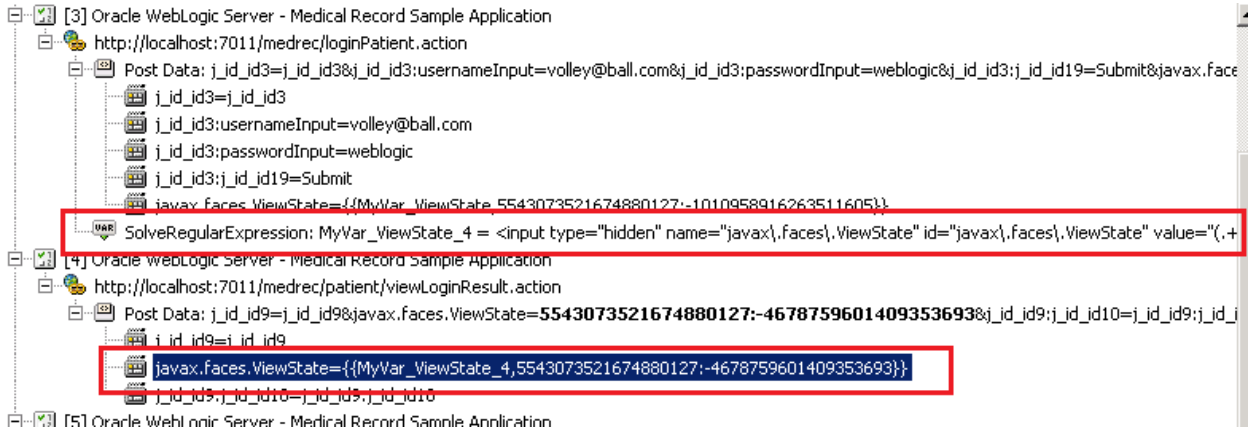




Name the Variable. I will name “**MyVar_ViewState_4**” in this example. Click Finish.

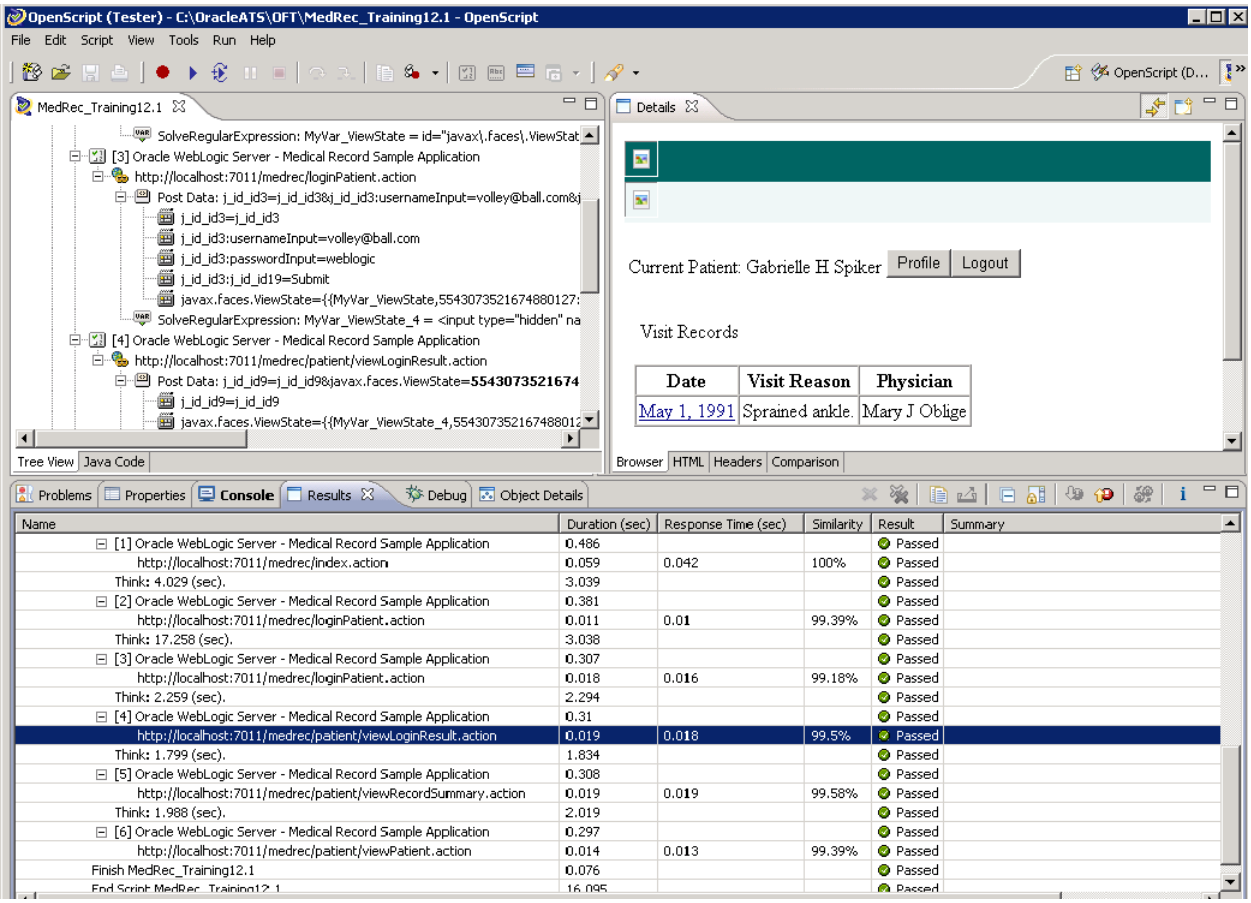


New variable is created. The value is retrieved from the contents from step 3, and applied to the Post data navigation in Step 4.



You will need to repeat the same procedure to create and apply variables for the ViewState parameters in the Steps 5, and 6.

Playback the script, and see the playback results. Similarities are more than 99% for all navigations, and correct content is returned from the server. Now the problem in the script is completely fixed.



Please visit **Oracle Application Testing Suite 12.x Video Series** at the following URL:
http://apex.oracle.com/pls/apex/f?p=44785:24:0::::P24_CONTENT_ID,P24_PREV_PAGE:6587,1

Oracle Application Testing Suite

OpenScript for Load Testing

Script Troubleshooting

Hands-on Lab Exercise