

Test practice

GeneXus™ 16



September 2019

Copyright ♥ GeneXus S.A. 1988-2019.

All rights reserved. This document may not be reproduced by any means without the express permission of GeneXus S.A. The information contained herein is intended for personal use only.

Registered Trademarks:

GeneXus is trademark or registered trademark of GeneXus S.A. All other trademarks mentioned herein are the property of their respective owners.

CONTENTS

Contents	2
Goal	3
Unit Test creation	3
Jenkins configuration	7
1- Add step for test execution	7
2- Add step for publishing results	8
3- Commit tests and see test results	10

GOAL

After we have configured the build process, we will configure the testing process of the application. For that, we will create a unit test and we will add a task in Jenkins by which the unit test will be executed unattended.

So, we will add steps in the configuration of our Jenkins project in which we have been working using DevOps.

UNIT TEST CREATION

Create a unit test for “GetSessionTitle” procedure, which receives an input a parameter called “&SessionId” and returns as output a parameter called “&SessionTitle”.

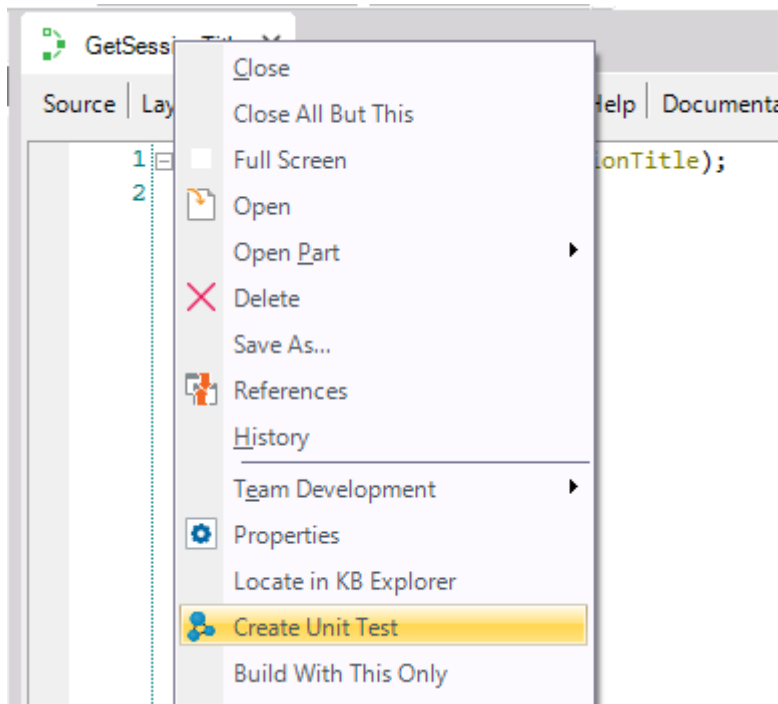
The screenshot shows the GeneXus KB Explorer interface. On the left, the 'Open:' field contains 'GetSessionTitle'. The tree view shows the project structure: DevOpsGXmeetingJuan > Main Programs > Root Module > DM_GeneXusMeeting > WebSite > Procedures > GetSessionTitle. The right pane shows the 'Rules' tab for the 'GetSessionTitle' procedure, with the following code:

```

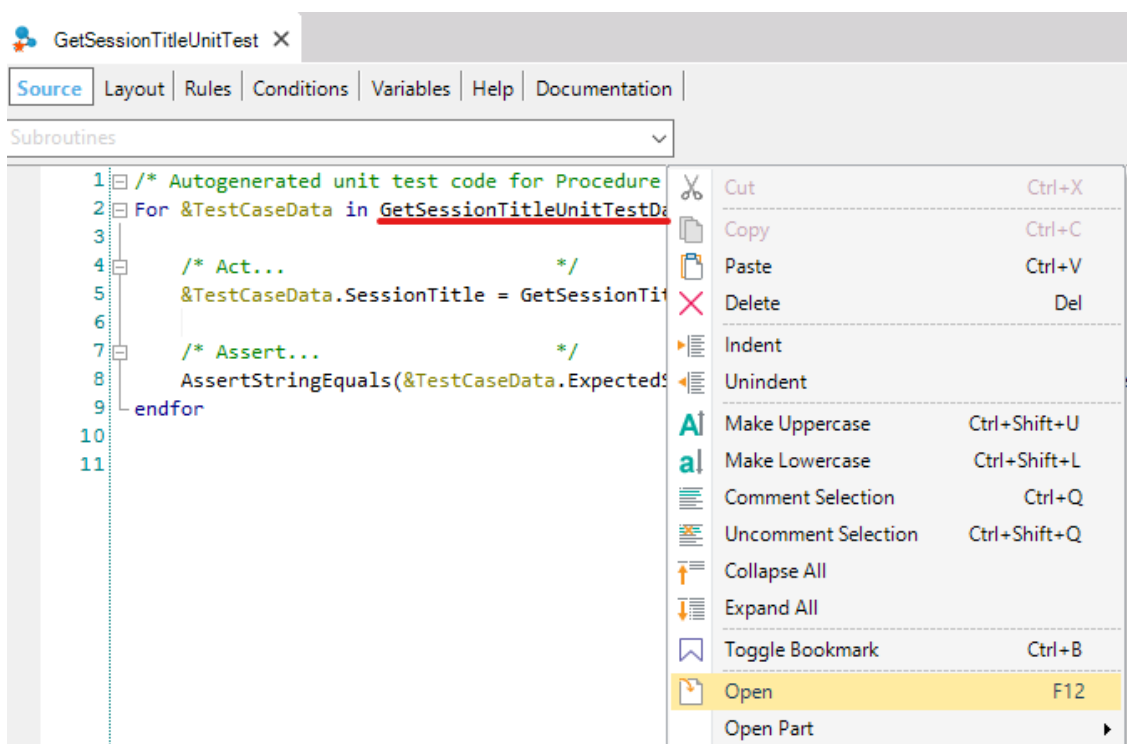
1 parm(in:&SessionId, out:&SessionTitle);
2

```

- Right click on the object and select “Create Unit Test”.



- Open the object “GetSessionTitleUnitTestData”.



- Copy the test cases data into the Data Provider.

```
GetSessionTitleUnitTestSDT
```

```
{  
    SessionId = 4448  
    ExpectedSessionTitle = "Tips & trends for smart devices"  
    ErrorMsgSessionTitle = ""  
}
```

```
GetSessionTitleUnitTestSDT
```

```
{  
    SessionId = 4619  
    ExpectedSessionTitle = "GX28 Mobile: GeneXus 16 en su máxima potencia"  
    ErrorMsgSessionTitle = ""  
}
```

```
GetSessionTitleUnitTestSDT
```

```
{  
    SessionId = 4597  
    ExpectedSessionTitle = "Next-Gen Trends: NOSQL and Serverless Apps in the Cloud"  
    ErrorMsgSessionTitle = ""  
}
```

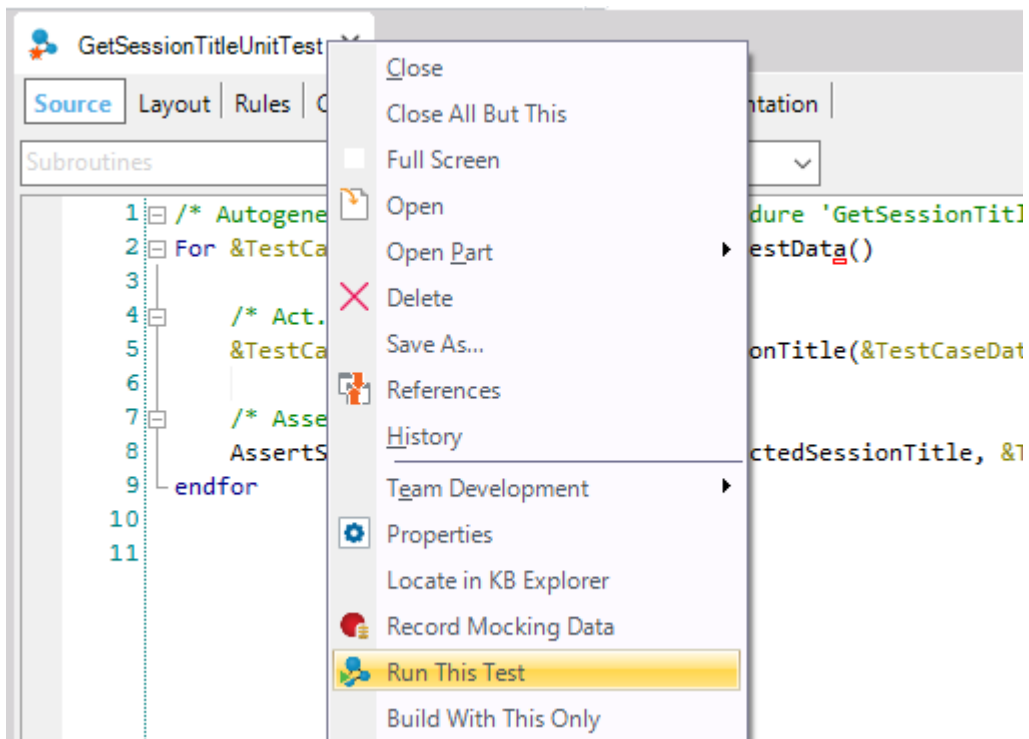
```
GetSessionTitleUnitTestSDT
```

```
{  
    SessionId = 4568  
    ExpectedSessionTitle = "El estado del Mapa GeneXus"  
    ErrorMsgSessionTitle = ""  
}
```

```
GetSessionTitleUnitTestSDT
```

```
{
    SessionId = 4591
    ExpectedSessionTitle = "Café con Startups"
    ErrorMsgSessionTitle = ""
}
```

- Right click over the test “*GetSessionTitleUnitTest*” and select the “*Run This Test*” option.



- In "Tests results" panel, you can see the results obtained for the execution.

☰ Tests Results
🔍 ✕

Tests list

Name	Started at	Elapsed
Passed (1)		
<input checked="" type="checkbox"/> ✔ WebSite.GetSessionTitl...	00:38:40	133 ms

Execution detail

Export Clear Results

✔ [WebSite.GetSessionTitleUnit Test](#)
 Started at: Wednesday, September 25, 2019 12:38:40 AM
 Elapsed time: 00:00:00.133

	Expected	Obtained	Error message	
✔	Tips & trends for s...	Tips & trends for smart devices		+
✔	GX28 Mobile: Gen...	GX28 Mobile: GeneXus 16 en...		+
✔	Next-Gen Trends: ...	Next-Gen Trends: NOSQL an...		+
✔	El estado del Map...	El estado del Mapa GeneXus		+
✔	Café con Startups	Café con Startups		+

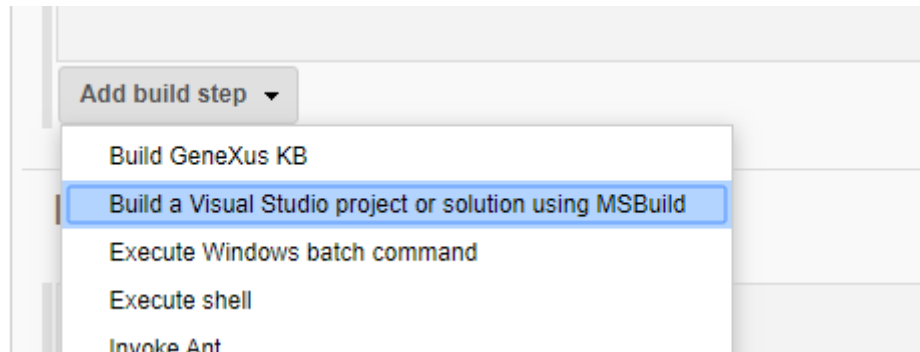
JENKINS CONFIGURATION

Now, we will configure the test step in Jenkins. This step will execute the unit test that we've just created.

For this, it is necessary to add a step in our pipeline. Follow the next points to achieve it:

1- ADD STEP FOR TEST EXECUTION

In the MsBuild “*Integration*” project, select the “*Configure*” menu, go to the “*Build*” section of the project tabs and select the “*Add build step*” option of type “*Build a Visual Studio project or solution using MSBuild*”.



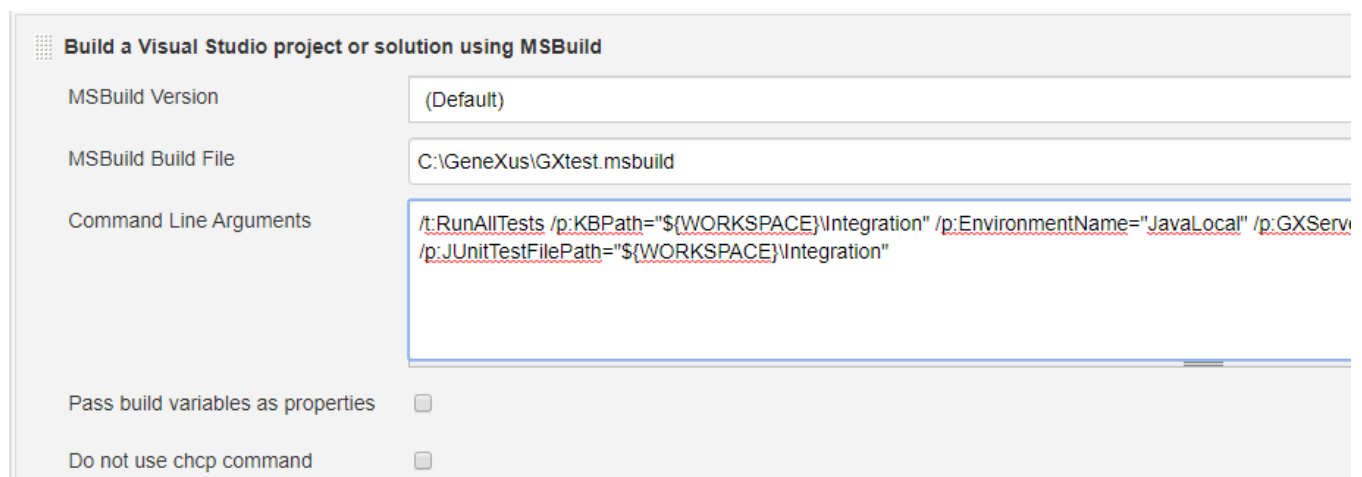
Complete with the following data:

MSBuild Build File

“C:\GeneXus\GXtest.msbuild”

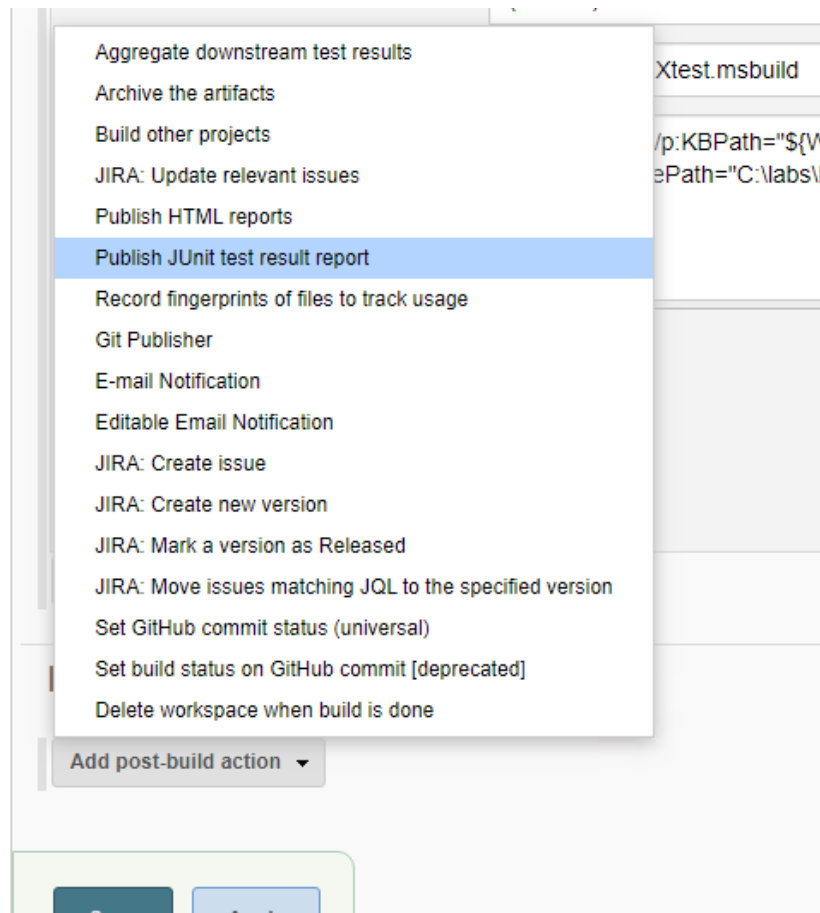
Command Line Arguments

```
/t:RunAllTests /p:KBPath="{WORKSPACE}\Integration" /p:EnvironmentName="JavaLocal"
/p:GXServerUser="local\builder" /p:GXServerPass="builder123"
/p:JUnitTestFilePath="{WORKSPACE}\Integration"
```



2- ADD STEP FOR PUBLISHING RESULTS

Go to section “Post-build Actions”, expand “Add post-build action” and then select “Publish JUnit test result report”



Fill with value “`${WORKSPACE}\Integration\TestResults*.xml`” the field “Test report XMLs” and save changes.

Post-build Actions

Post-build Actions
X
?

Publish JUnit test result report

Test report XMLs

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is [the workspace root](#).

Retain long standard output/error ?

Health report amplification factor ?

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Allow empty results Do not fail the build on empty test results ?

Add post-build action ▼

Save
Apply

3- COMMIT TESTS AND SEE TEST RESULTS

Finally, commit the unit test and view the results of the pipeline execution.

When pipeline execution is finished, test results can be viewed in section “*Test Result*” inside the corresponding pipeline menu.

The screenshot shows the Jenkins web interface for a test report. The browser address bar indicates the URL is localhost:8082/job/Integration/9/testReport/. The Jenkins header shows the user 'Bob The Builder' is logged in. The breadcrumb trail is 'Jenkins > Integration > #9 > Test Results'. The main content area is titled 'Test Result' and displays '0 failures (-1)' with a blue progress bar. Summary statistics show '1 tests (±0)' and 'Took 0.5 sec.' with an 'add description' link. Below this is an 'All Tests' table with the following data:

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
(root)	0.5 sec	0 -1	0	1 +1	1

GeneXus™ 16

GXtest