Article
Evgeny Shvarov · May 7, 2020 4m read

Open Exchange

Unit Testing with ZPM

Hi Developers!

I want to share with you the approach you can use working with Unit Testing which I personally find very convenient and robust.

Will not talk much here about Unit Testing, we have documentation, enough good articles on the topic not only on the Internet but here on Developers Community too, e.g. this one.

How can you manage the Unit Testing of your ZPM module?

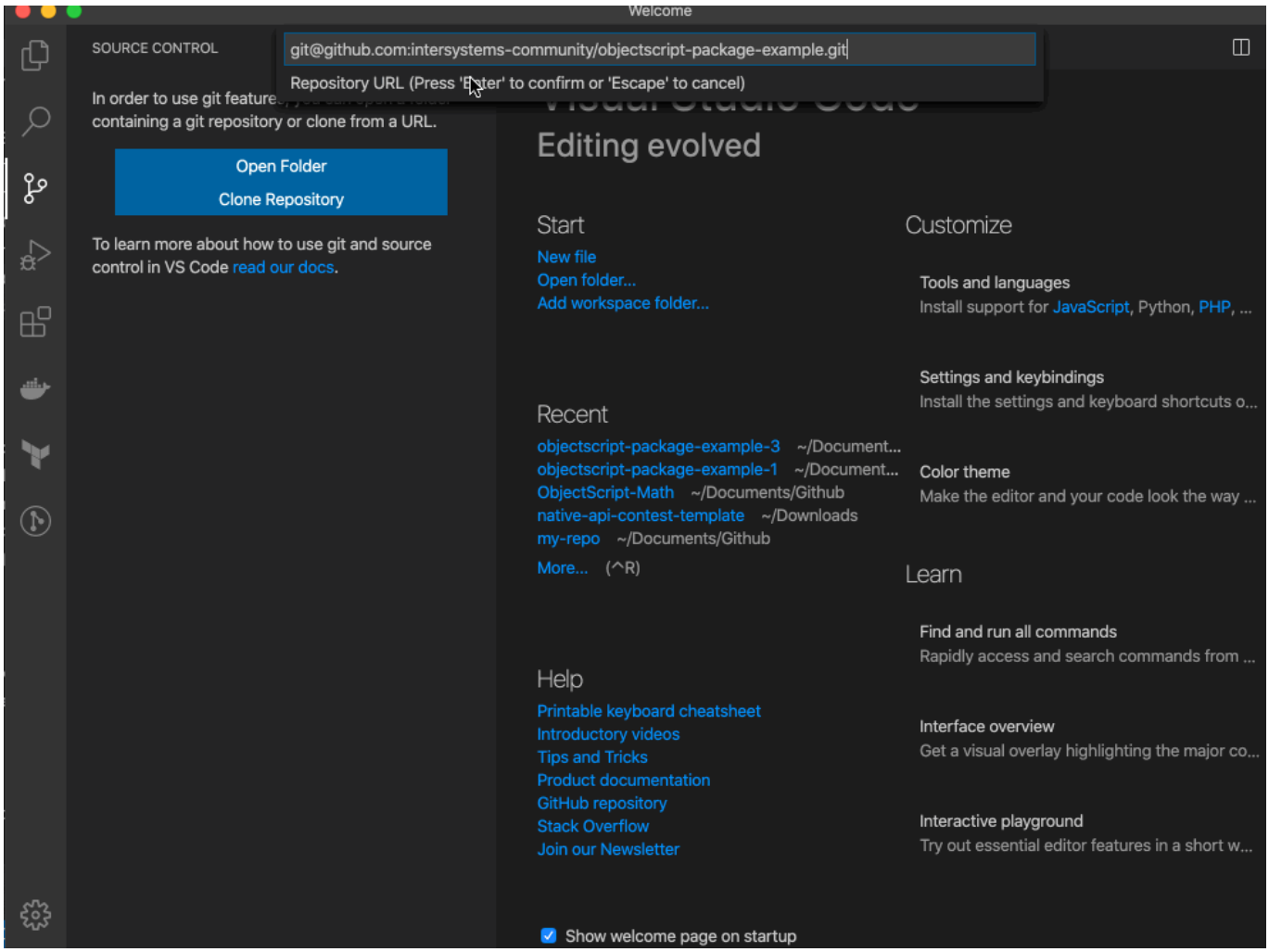
This is pretty handy with ZPM. It has a special command "test" which runs all the tests you mentioned in the module. You can execute it as shown below:

```
zpm:USER>module-name test
```

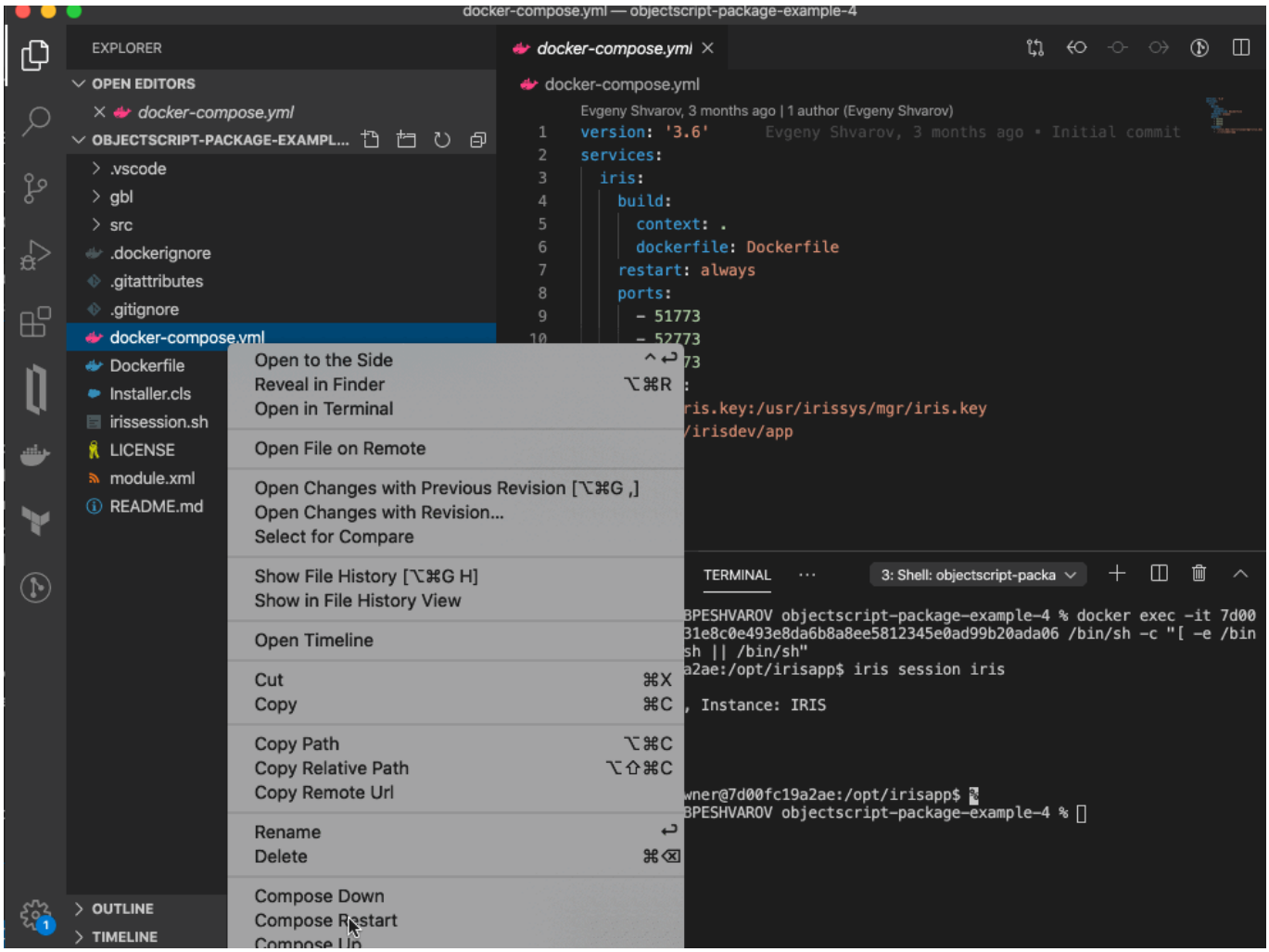
And this command runs all the tests you mentioned in module.xml.

Let's see how it works.

Clone and open package example repo in VSCode.



Run docker container with IRIS and preinstalled ZPM-client:



As you see in the module.xml the name of the module is "objectscript-package-example".

Let's test it in IRIS terminal.

```
USER>zn "IRISAPP"

zpm:IRISAPP>load objectscript-package-example
```

And we can test it:

```
zpm:IRISAPP>objectscript-package-example test

[objectscript-package-example]  Reload START
[objectscript-package-example]  Reload SUCCESS
[objectscript-package-example]  Module object refreshed.
[objectscript-package-example]  Validate START
[objectscript-package-example]  Validate SUCCESS
[objectscript-package-example]  Compile START
[objectscript-package-example]  Compile SUCCESS
[objectscript-package-example]  Activate START
[objectscript-package-example]  Configure START
[objectscript-package-example]  Configure SUCCESS
[objectscript-package-example]  Activate SUCCESS
[objectscript-package-example]  Test START
Use the following URL to view the result:
```

```
http://192.168.80.2:52773/csp/sys/%25UnitTest.Portal.Indices.cls?Index=11&$NAMESPACE=
IRISAPP
All PASSED
[objectscript-package-example]  Test SUCCESS
```

As we see it passed all the tests.

We can see more details if we pass "-v" clause:

Let's see how it all works.

The module.xml has the one line which describes unit tests:

```
<UnitTest Name="src" Package="UnitTests" Phase="test"/>
```

This line means that ZPM will look in */src* folder of the repo and will run all the tests of all the classes which are included in UnitTests package.

Here we have two test classes, so both will be executed.

Let's change something in the source code to let the test fail.

In the TestClassExample test we have method:

```
Method TestFortyTwo()

{

Set tResults=##class(community.objectscript.ClassExample).FortyTwo()

Set tExpected=42

Do $$$AssertEquals(tResults,tExpected,tExpected_" = "_tResults)

}
```

This test method covers the related method FortyTwo in objectscript.ClassExample:

```
ClassMethod FortyTwo() As %Integer
{
    return 42
}
```

Let's change it to make it return 43 and Save:

```
ClassMethod FortyTwo() As %Integer
{
    return 43
}
```

And run tests again:

```
zpm:IRISAPP>objectscript-package-example test -v
...
[objectscript-package-example]  Test START
...
  UnitTests.TestClassExample begins ...
      TestFortyTwo() begins ...
AssertEquals:42 = 43 (failed)  <<==== **FAILED**
(root):UnitTests.TestClassExample:TestFortyTwo:
        LogMessage:Duration of execution: .000028 sec.
...
zpm: IRISAPP>
```

So this test failed as planned. Perfect!

But it's a very artificial example. Let's fail the test which checks something closer to the real world.

Check another test case:

Here is the related test method in another test class UnitTests.TestPersistentClass:

```
Method TestCreateRecord()

{

do $$$AssertStatusOK(##class(community.objectscript.PersistentClass).CreateRecord()," 
Create Record test")

}
```

Which covers the method CreateRecord in objectscript.PersitentClass:

```
ClassMethod CreateRecord() As %Status

{

s objPC=..%New()

s objPC.Test="Test string"

return objPC.%Save()

}
```

As you see in source of the PersistentClass it has one property Test with MAXLEN limit of 12 symbols:

```
Property Test As %String(MAXLEN = 12);
```

Let's exceed the length to fail the save record method and change the code to the following:

```
ClassMethod CreateRecord() As %Status

{

s objPC=..%New()
```

```
s objPC.Test="Test very long string"

return objPC.%Save()

}
```

Save it and then run the test again:

```
zpm:IRISAPP>objectscript-package-example test -v
...
[objectscript-package-example] Test START
...
    UnitTests.TestPersistentClass begins ...
      TestCreateRecord() begins ...
AssertStatusOK:Create Record => ERROR #7201: Datatype value 'Test very long string' l
ength longer than MAXLEN allowed of 12
  > ERROR #5802: Datatype validation failed on property 'community.objectscript.Persi
stentClass:Test', with value equal to "Test very long string" (failed)  <<==== **FAIL
ED**
(root):UnitTests.TestPersistentClass:TestCreateRecord:
        LogMessage:Duration of execution: .001794 sec.
      TestCreateRecord failed
    UnitTests.TestPersistentClass failed
```

As you see our second TestCreateREcord test failed.

That's it!

I want to add, that you can use this approach not only to create packages but also to simplify and automate your testing and deployment procedures!

Also, you may find interesting @Timothy.Leavitt's article on how you can check the test coverage of your project with ZPM and Mocking Framework article by @AndreClaude Gendron .

Happy coding!

#ObjectScript #InterSystems Package Manager (IPM) #Testing #InterSystems IRIS
Check the related application on InterSystems Open Exchange