Article
Evgeny Shvarov · Jan 15, 2020 3m read

Open Exchange

Using Invoke Element to Call Class Methods in ObjectScript Packages

Hi Developers!

Often when we install a code package we want to make some post-install settings, e.g. call to a method, set up a configuration file.

This article describes how to do this with the ObjectScript Package Manager.

To make any post-install calls you need to add <Invoke> elements into <Invokes> tag to the module.xml. Each <Invoke> element can have nested <Arg> elements if you want to pass params to the method:

```
<Invokes>

<Invoke Class="Class.Name1" Method="MethodName1">

<Arg>Sting Value</Arg>

<Arg>123</Arg>
</Invoke>

</Invkoes>
```

Here is an example module.xml which has the calls for one method without params which creates a record in a persistent class, and another call with parameters which sets the data into global:

```
<Invokes>
        <Invoke Class="community.objectscript.PersistentClass" Method="CreateRecord">
</Invoke>
        <Invoke Class="community.objectscript.ClassExample" Method="SetToTheGlobal">
          <Arg>42</Arg>
          <Arg>Text Data</Arg>
        </Invoke>
      </Invokes>
```

According to module.xml this package 'objectscript-package-template' after it imports all the code will call two methods.

First: CreateRecord of community.objectscript.PersistentClass class - which is equivalent to the call:

```
do ##class(community.objectscript.PersistentClass).CreateRecord()
```

Second: SetToTheGlobal of community.objectscript.ClassExample class and passes two parameters: 42 and "Test Data", which is equivalent to the call:

```
do ##class(community.objectscript.ClassExample).SetToTheGlobal(42,"Text Data")
```

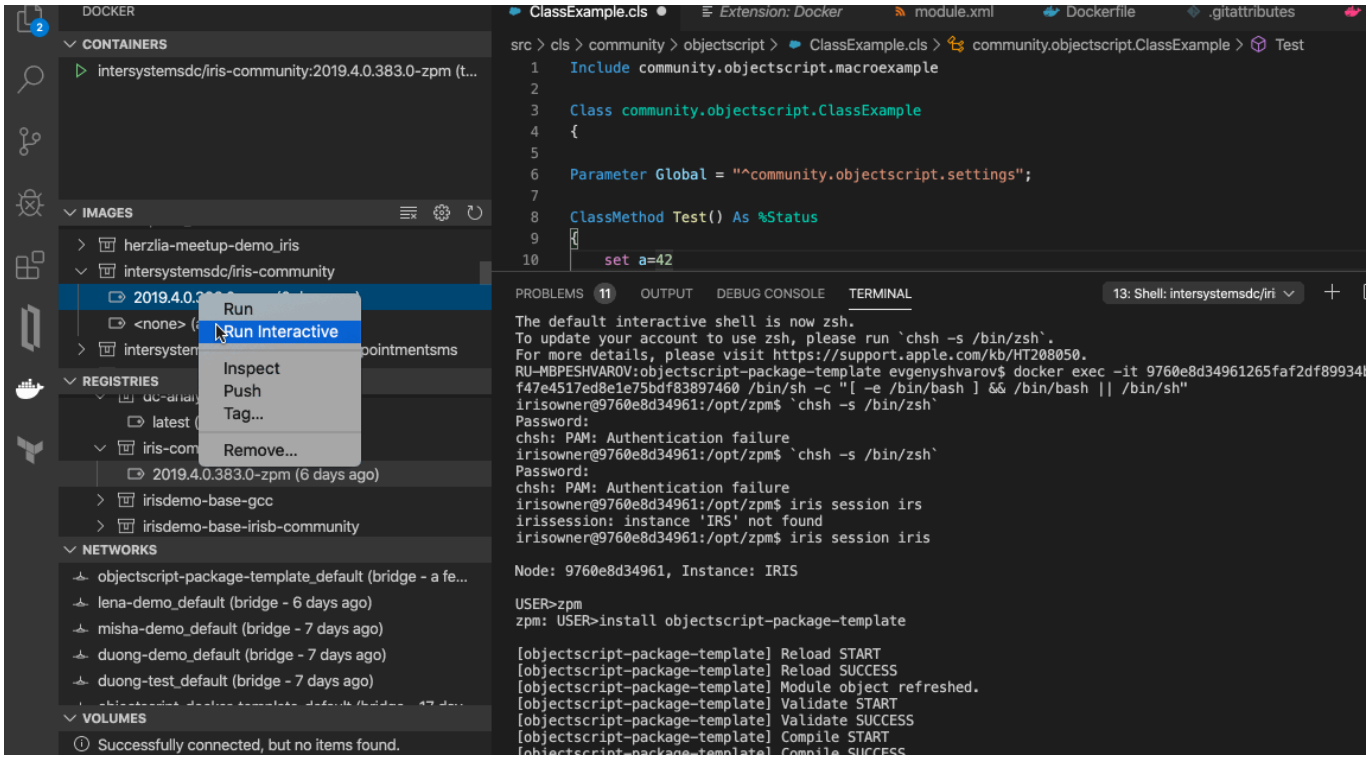
Let's test if this works. Run the docker container with IRIS 2019.4 and ZPM client on board (which you can pull from InterSystems Developers Community repository):

```
docker run --rm -d intersystemsdc/iris-community:2019.4.0.383.0-zpm
```

Launch terminal to the container. It will be another container_id in your case but you can use a handy menu in VSCode Docker plugin:

```
docker exec -it 9760e8d34961265faf2df89934b4c049510c2f47e4517ed8e1e75bdf83897460 /bin
/sh -c "[ -e /bin/bash ] && /bin/bash || /bin/sh"
```

This is how you can make it in VSCode with Docker Plugin:



Or you can just import ZPM client into any Namespace from Community Registry:

Open IRIS and install objectscript-package-template module:

```
irisowner@9760e8d34961:/opt/zpm$ iris session iris

Node: 9760e8d34961, Instance: IRIS

USER>zpm
zpm: USER>install objectscript-package-template

[objectscript-package-template] Reload START
[objectscript-package-template] Reload SUCCESS
[objectscript-package-template] Module object refreshed.
[objectscript-package-template] Validate START
[objectscript-package-template] Validate SUCCESS
[objectscript-package-template] Compile START
[objectscript-package-template] Compile SUCCESS
[objectscript-package-template] Activate START
```

```
[objectscript-package-template] Configure START
[objectscript-package-template] Configure SUCCESS
[objectscript-package-template] Activate SUCCESS
zpm: USER>
```

The module has been installed. Let's test if it created a record first, and call the method which writes all the records to the terminal:

```
USER>d ##class(community.objectscript.PersistentClass).WriteAllRecords()
ID=1 Test=Test string
```

Yes, it created a record. Ok, let's test if the data is in the global:

```
USER>d ##class(community.objectscript.ClassExample).OutputGlobal()
^community.objectscript.settings("IRIS Version")="IRIS for UNIX (Ubuntu Server LTS fo
r x86-64 Containers) 2019.4 (Build 383U) Fri Dec 6 2019 08:49:54 EST"
^community.objectscript.settings("Mode")="Package Manager"
^community.objectscript.settings("Parameter")=42
^community.objectscript.settings("integer")=42
^community.objectscript.settings("string")="Text Data"
^community.objectscript.settings("today")="15 Jan 2020"
```

Notice, that ^community.objectscript.settings global contains not only the data which we introduced with the call to 'SetToTheGlobal' method but also the data in the ^community.objectscript.settings global we installed with package import.

I hope ZPM invoke functionality will help you use ZPM more and deliver handy and useful InterSystems ObjectScript libraries and solutions!

Your questions are very welcome in the comments below! Stay tuned!

#ObjectScript #ObjectScript Package Manager (ZPM) #Tutorial #Open Exchange
Check the related application on InterSystems Open Exchange

Source URL: https://community.intersystems.com/post/using-invoke-element-call-class-methods-objectscript-packages