Article
Lorenzo Scalese · Apr 15, 2021 6m read

Open Exchange

Combine config-api with ZPM client

Hi Developers,

In the previous article, we describe how to use config-api to configure IRIS.

Now, let's try to combine the library with the ZPM client.
The goal is to load a configuration document during zpm install at the configure phase.

For this exercise, a template repository is available here (this is based on objectscript-docker-template).

We attempt to :

- Create a database MYAPPDATA.
- Set Globals mapping for dc.PackageSample.*.
- Add a user named SQLUserRO with read-only SQL role access.
- Add an SSL Configuration named SSLAppDefault.
- Create a REST application /rest/myapp.

The configuration file iris-config.json is located in the config-api subdirectory. This is an empty template and you can fill with the following content:

```
{
    "Security.Roles":{
        "MyRoleRO" : {
            "Descripion" : "SQL Read Only Role for dc_PackageSample schema.",
            "Resources" : "%Service_SQL:U",
            "GrantedRoles" : "%SQL"
        }
    },
    "Security.SQLPrivileges": [{
        "Grantee": "MyRoleRO",
        "PrivList" : "s",
        "SQLObject" : "1,dc_PackageSample.*"
    }],
    "Security.SQLAdminPrivilegeSet" : {
        "${namespace}": [

        ]
    },
    "Security.Users": {
        "${sqlusr}": {
            "Name":"${sqlusr}",
            "Password":"$usrpwd$",
            "AccountNeverExpires":true,
            "AutheEnabled":0,
            "ChangePassword":false,
            "Comment":"Demo SQLUserRO",
            "EmailAddress":"",
```

```
            "Enabled":true,
            "ExpirationDate":"",
            "FullName":"Demo SQLUserRO",
            "NameSpace":"${namespace}",
            "PasswordNeverExpires":false,
            "PhoneNumber":"",
            "PhoneProvider":"",
            "Roles":"MyRoleRO",
            "Routine":""
        }
    },
    "Security.SSLConfigs": {
        "SSLAppDefault":{}
    },
    "Security.Applications" : {
        "/rest/myapp": {
            "NameSpace" : "${namespace}",
            "Enabled" : 1,
            "DispatchClass" : "Your.Dispatch.class",
            "CSPZENEnabled" : 1,
            "AutheEnabled": 32
        }
    },
    "SYS.Databases":{
        "${mgrdir}myappdata" : {
            "ExpansionSize":64
        }
    },
    "Databases":{
        "MYAPPDATA" : {
            "Directory" : "${mgrdir}myappdata"
        }
    },
    "MapGlobals":{
        "${namespace}": [{
            "Name" : "dc.PackageSample.*",
            "Database" : "MYAPPDATA"
        }]
    },
    "Library.SQLConnection": {

    }
}
```

You can see the usage of ${namespace}, ${mgrdir}. This is pre-defined variables and It will be evaluated at runtime with the current namespace and IRIS mgr directory. Others pre-defined variables are :

- ${cspdir} : csp subdirectory in iris install directory.
- ${bindir} : bin directory $SYSTEM.Util.BinaryDirectory()
- ${libdir}: lib subdirectory in iris install directory.
- ${username}: current username $USERNAME
- ${roles}: current granted roles $ROLES

Also, we used ${sqlusr}, It's not a pre-defined variable and we must set the value from module.xml. We don't really need it, but it's just to demonstrate how developers can define and set variables from modules.xml and pass them to the configuration file.

To help you to create your own configuration file, a cheat sheet config-api.md is also available in the template repository.

The configuration file is now ready, next step the module.xml file.

We have to :

- Add the config-api module as a dependency.
- Invoke a method to load our configuration file ./config-api/iris-config.json with ${sqlusr} argument.

```
<?xml version="1.0" encoding="UTF-8"?>
<Export generator="Cache" version="25">
  <Document name="objectscript-template-with-config-api.ZPM">
    <Module>
      <Name>objectscript-template-with-config-api</Name>
      <Version>0.0.1</Version>
      <Packaging>module</Packaging>

      <FileCopy Name="config-api/" InstallDirectory="${libdir}config-api/"/>
      <Invoke Class="Api.Config.Services.Loader" Method="LoadFromInvoke">
        <Arg>${root}config-api/iris-config.json</Arg>
        <Arg>sqlusr</Arg>
        <Arg>SQLUserRO</Arg>
      </Invoke>

      <SourcesRoot>src</SourcesRoot>
      <Resource Name="dc.PackageSample.PKG"/>

      <Dependencies>
        <ModuleReference>
          <Name>config-api</Name>
          <Version>1.0.0</Version>
        </ModuleReference>
      </Dependencies>

    </Module>
  </Document>
</Export>
```

Take a look at the Invoke tag :

```
<Invoke Class="Api.Config.Services.Loader" Method="LoadFromInvoke">
        <Arg>${root}config-api/iris-config.json</Arg>
        <Arg>sqlusr</Arg>
        <Arg>SQLUserRO</Arg>
</Invoke>
```

The first argument is the path to the configuration file, ${root} contains the module directory used by zpm during the deployment.
After that, all others arguments must be paired. Firstly a string with the variable name followed by the related value.
In our example sqlusr for the variable ${sqlusr} followed by the value SQLUserRO.
You can pass many paired arguments as you need.

You notice that there is a FileCopy tag : <FileCopy Name="config-api/" InstallDirectory="${libdir}config-api/"/>
This tag exists only to force zpm to package the configuration file. We would add an Invoke tag to delete the

directory to clean up like this :

```
<Invoke Class="%File" Method="RemoveDirectoryTree">
    <Arg>${libdir}config-api/</Arg>
</Invoke>
```

Module.xml is ready, it's time to test. Build and start the container docker-compose up -d --build.
You should see these logs during the configuration phase:

```
[objectscript-template-with-config-api] Configure START
2021-04-08 19:24:43 Load from file /opt/irisbuild/config-api/iris-config.json ...
2021-04-08 19:24:43 Start load configuration
2021-04-08 19:24:43 {
  "Security.Roles":{
    "MyRoleRO":{
      "Descripion":"SQL Read Only Role for dc_PackageSample schema.",
      "Resources":"%Service_SQL:U",
      "GrantedRoles":"%SQL"
    }
  },
  "Security.SQLPrivileges":[
    {
      "Grantee":"MyRoleRO",
      "PrivList":"s",
      "SQLObject":"1,dc_PackageSample.*"
    }
  ],
  "Security.SQLAdminPrivilegeSet":{
    "USER":[
    ]
  },
  "Security.Users":{
    "SQLUserRO":{
      "Name":"SQLUserRO",
      "Password":"$usrpwd$",
      "AccountNeverExpires":true,
      "AutheEnabled":0,
      "ChangePassword":false,
      "Comment":"Demo SQLUserRO",
      "EmailAddress":"",
      "Enabled":true,
      "ExpirationDate":"",
      "FullName":"Demo SQLUserRO",
      "NameSpace":"USER",
      "PasswordNeverExpires":false,
      "PhoneNumber":"",
      "PhoneProvider":"",
      "Roles":"MyRoleRO",
      "Routine":""
    }
  },
  "Security.SSLConfigs":{
    "SSLAppDefault":{
    }
  },
  "Security.Applications":{
```

```
    "/rest/myapp":{
      "NameSpace":"USER",
      "Enabled":1,
      "DispatchClass":"Your.Dispatch.class",
      "CSPZENEnabled":1,
      "AutheEnabled":32
    }
  },
  "SYS.Databases":{
    "/usr/irissys/mgr/myappdata":{
      "ExpansionSize":64
    }
  },
  "Databases":{
    "MYAPPDATA":{
      "Directory":"/usr/irissys/mgr/myappdata"
    }
  },
  "MapGlobals":{
    "USER":[
      {
        "Name":"dc.PackageSample.*",
        "Database":"MYAPPDATA"
      }
    ]
  },
  "Library.SQLConnection":{
  }
}
2021-04-08 19:24:43  * Security.Roles
2021-04-08 19:24:43    + Create MyRoleRO ... OK
2021-04-08 19:24:43  * Security.SQLPrivileges
2021-04-08 19:24:43    + Create {"Grantee":"MyRoleRO","PrivList":"s","SQLObject":"1,d
c_PackageSample.*"} ... OK
2021-04-08 19:24:43  * Security.SQLAdminPrivilegeSet
2021-04-08 19:24:43  * Security.Users
2021-04-08 19:24:43    + Create SQLUserRO ... OK
2021-04-08 19:24:43  * Security.SSLConfigs
2021-04-08 19:24:43    + Create SSLAppDefault ... OK
2021-04-08 19:24:43  * Security.Applications
2021-04-08 19:24:43    + Create /api/config ... OK
2021-04-08 19:24:43  * SYS.Databases
2021-04-08 19:24:43    + Create /usr/irissys/mgr/myappdata ... OK
2021-04-08 19:24:43  * Databases
2021-04-08 19:24:43    + Create MYAPPDATA ... OK
2021-04-08 19:24:44  * MapGlobals
2021-04-08 19:24:44    + Create USER dc.PackageSample.* ... OK
2021-04-08 19:24:44  * Library.SQLConnection
[objectscript-template-with-config-api] Configure SUCCESS
```

Your configuration has been applied and the configuration file will be packaged on publishing.
You can check this in the management portal.

Thanks for reading.

#Deployment #DevOps #InterSystems IRIS
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/combine-config-api-zpm-client