

IRIS R18





Contents

Prerequisites	3
Setup	3
Create Service Project.....	7
Create Service Container Project	10
Creation of Artefacts in T24	14
Creation of Provider API	15
Build Service Project	20
Build Container Project	24
Testing.....	27
Deployment options	30
Deployment and Testing.....	34
Swagger Specification	35



Introduction

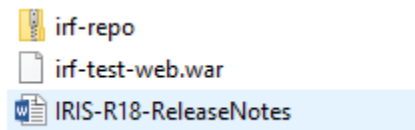
IRIS R18 is a lightweight, REST standards based solution that uses OFS message format to communicate with T24. This document helps to configure and deploy IRIS R18 Provider APIs.

Prerequisites

- Apache Camel
- Maven
- Spring
- Open API Spec, fka Swagger
- Postman
- Eclipse
- Junit
- Java 8
- T24 Application

Setup

- 1) Request IRIS R18 Package from Distribution, The package will have the below components.



- 2) Unzip the irf-repo.zip and copy the contents into DesignStudio -> t24binaries as given in the below sample,



Temenos > R18 > Env > Slot01 > Products > DesignStudio > t24-binaries

Name	Date modified	Type
activation	6/13/2018 6:39 PM	File folder
ant	6/13/2018 6:39 PM	File folder
ant-contrib	6/13/2018 6:39 PM	File folder
antlr	6/13/2018 6:39 PM	File folder
aopalliance	6/13/2018 6:39 PM	File folder
asm	6/13/2018 6:39 PM	File folder
avalon-framework	6/13/2018 6:39 PM	File folder
backport-util-concurrent	6/13/2018 6:39 PM	File folder
bouncycastle	6/13/2018 6:39 PM	File folder
bsh	6/13/2018 6:39 PM	File folder
cglib	6/13/2018 6:39 PM	File folder
ch	6/13/2018 6:39 PM	File folder
checkstyle	6/13/2018 6:39 PM	File folder
classworlds	6/13/2018 6:39 PM	File folder
com	6/13/2018 6:40 PM	File folder
commons-beanutils	6/13/2018 6:39 PM	File folder
commons-chain	6/13/2018 6:39 PM	File folder

- 3) Copy the archetype-catalog.xml provided along with the release notes to the root of your local maven repository (in our scenario it is t24binaries directory). Below given is the sample archetype-catalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<archetype-catalog
  xsi:schemaLocation="http://maven.apache.org/plugins/maven-archetype-
plugin/archetype-catalog/1.0.0 http://maven.apache.org/xsd/archetype-catalog-
1.0.0.xsd"
  xmlns="http://maven.apache.org/plugins/maven-archetype-plugin/archetype-
catalog/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <archetypes>
    <archetype>
      <groupId>com.temenos.irf</groupId>
      <artifactId>irf-service-archetype</artifactId>
      <version>18.0.2</version>
    </archetype>
    <archetype>
      <groupId>com.temenos.irf</groupId>
      <artifactId>irf-service-container-archetype</artifactId>
      <version>18.0.2</version>
    </archetype>
  </archetypes>
</archetype-catalog>
```



- 4) Workbench binaries are available as irf-web-client-release.zip bundle in as irf-web-client-release-x-x-x.jar.

IRISR18_V1.0.0 > irf-repo > com > temenos > irf > irf-test-webclient > 18.0.2

Name	Date modified	Type	Size
irf-test-webclient-18.0.2	5/30/2018 6:33 AM	Executable Jar File	1,849 KB
irf-test-webclient-18.0.2.pom	5/30/2018 6:33 AM	POM File	2 KB

C:\Sri\Temenos-Docs\DOCS\IRIS\irisR18\IRISR18_V1.0.0\irf-repo\com\temenos\irf\irf-test-webclient\18.0.2\irf-test-webclient-18.0.2.jar\

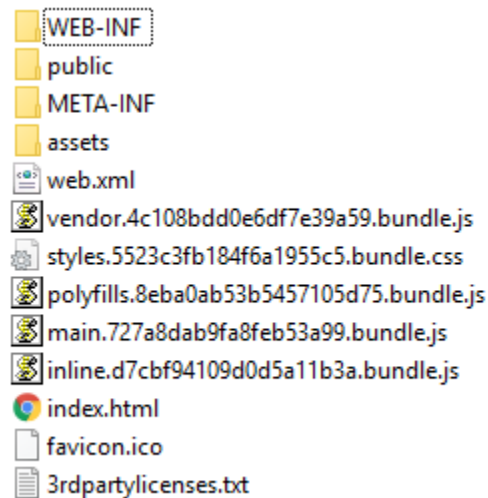
File Edit View Favorites Tools Help

Add Extract Test Copy Move Delete Info

C:\Sri\Temenos-Docs\DOCS\IRIS\irisR18\IRISR18_V1.0.0\irf-repo\com\temenos\irf\irf-test-webclient\18.0.2\irf-test-webclient-18.0.2

Name	Size	Pi
META-INF	1 785	
irf-web-client-release.zip	1 901 043	

Extract the contents of this zip file and copy it into <Apache HTTP Server Installation Home >\ht-docs folder



<Apache HTTP Server Installation Home >\ht-docs folder



My Data (C:) > Apache24 > htdocs			
Name	Date modified	Type	Size
assets	6/15/2018 11:12 AM	File folder	
META-INF	3/13/2018 5:36 PM	File folder	
public	6/15/2018 11:12 AM	File folder	
WEB-INF	6/15/2018 11:12 AM	File folder	
.htaccess	2/14/2018 1:27 PM	HTACCESS File	1 KB
3rdpartylicenses	4/12/2018 2:45 PM	Text Document	5 KB
favicon	4/12/2018 2:45 PM	Icon	5 KB
index	4/12/2018 2:45 PM	Chrome HTML Do...	2 KB
inline.2d37535c255ca915ce93.bundle	3/20/2018 8:04 PM	JavaScript File	2 KB
inline.30d72e4e61fe5ff066f4.bundle	3/13/2018 5:36 PM	JavaScript File	2 KB
inline.0374aac3194fb1af1798.bundle	3/23/2018 3:39 PM	JavaScript File	2 KB
inline.6274e9e109cb0e0f042f.bundle	3/13/2018 5:49 PM	JavaScript File	2 KB
inline.d7cbf94109d0d5a11b3a.bundle	4/12/2018 2:45 PM	JavaScript File	2 KB
main.22a3b753127d6e0facb7.bundle	3/20/2018 8:04 PM	JavaScript File	325 KB
main.589df908f12039d6a377.bundle	3/23/2018 3:39 PM	JavaScript File	325 KB
main.727a8dab9fa8feb53a99.bundle	4/12/2018 2:45 PM	JavaScript File	354 KB
main.ac1722b1e2ba81029a72.bundle	3/13/2018 5:49 PM	JavaScript File	310 KB
main.b5d2cb55da6711da90af.bundle	3/13/2018 5:36 PM	JavaScript File	310 KB
polyfills.8eba0ab53b5457105d75.bundle	4/12/2018 2:45 PM	JavaScript File	65 KB
styles.5523c3fb184f6a1955c5.bundle	4/12/2018 2:45 PM	Cascading Style S...	1 KB
styles.ae0274caf099dd786b37.bundle	3/23/2018 3:39 PM	Cascading Style S...	1 KB
vendor.4c108bdd0e6df7e39a59.bundle	4/12/2018 2:45 PM	JavaScript File	615 KB
web	3/23/2018 2:18 PM	XML Document	2 KB

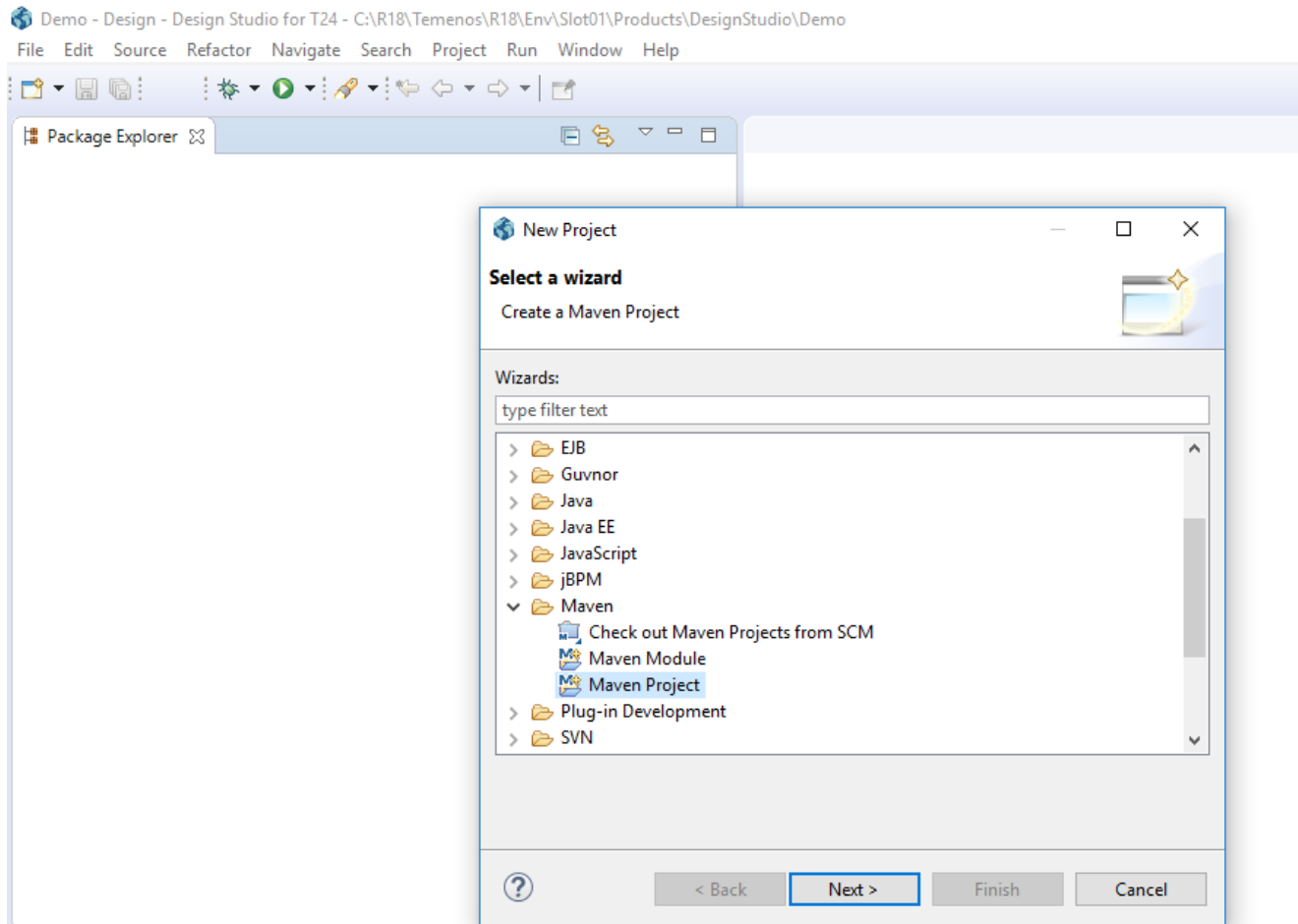
5) With this, all the setup related to IRIS R18 Build and Design time is done.



Create Service Project

Create a service project using the irf-service-archetype. It contains the default directories and set of model data. It basically acts as a platform where developers can build and run the required APIs

- 1) Open the Design Studio and Click on File -> New -> Maven Project





New Maven Project

Select project name and location

☐ Create a simple project (skip archetype selection)

☒ Use default Workspace location

Location: Browse...

☐ Add project(s) to working set

Working set: More...

► **Advanced**

- 2) Select irf-service-archetype and the appropriate version delivered (in this example 18.0.2), if com.temenos.irf is not displayed then configure it by clicking on the configure option displayed in the wizard and select the archetype-catalog.xml (available in t24binaries directory).

New Maven Project

Select an Archetype

Catalog: Configure...

Filter:

Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	18.0.2
com.temenos.irf	irf-service-container-archetype	18.0.2
org.apache.maven.archetypes	maven-archetype-archetype	1.0
org.apache.maven.archetypes	maven-archetype-j2ee-simple	1.0
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1

☒ Show the last version of Archetype only ☐ Include snapshot archetypes Add Archetype...

► **Advanced**

< Back Next > Finish Cancel



Enter Group Id, Artifact Id, Version and Package, an example is given below,

New Maven Project

New Maven project
Specify Archetype parameters

Group Id:

Artifact Id:

Version:

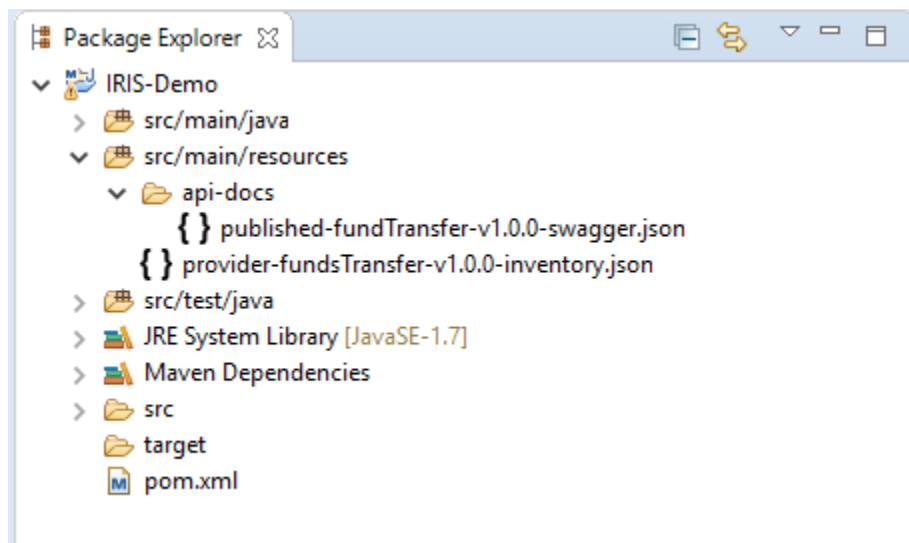
Package:

Properties available from archetype:

Name	Value

► Advanced

6) Click on Finish. It will generate the folder structure as shown below,

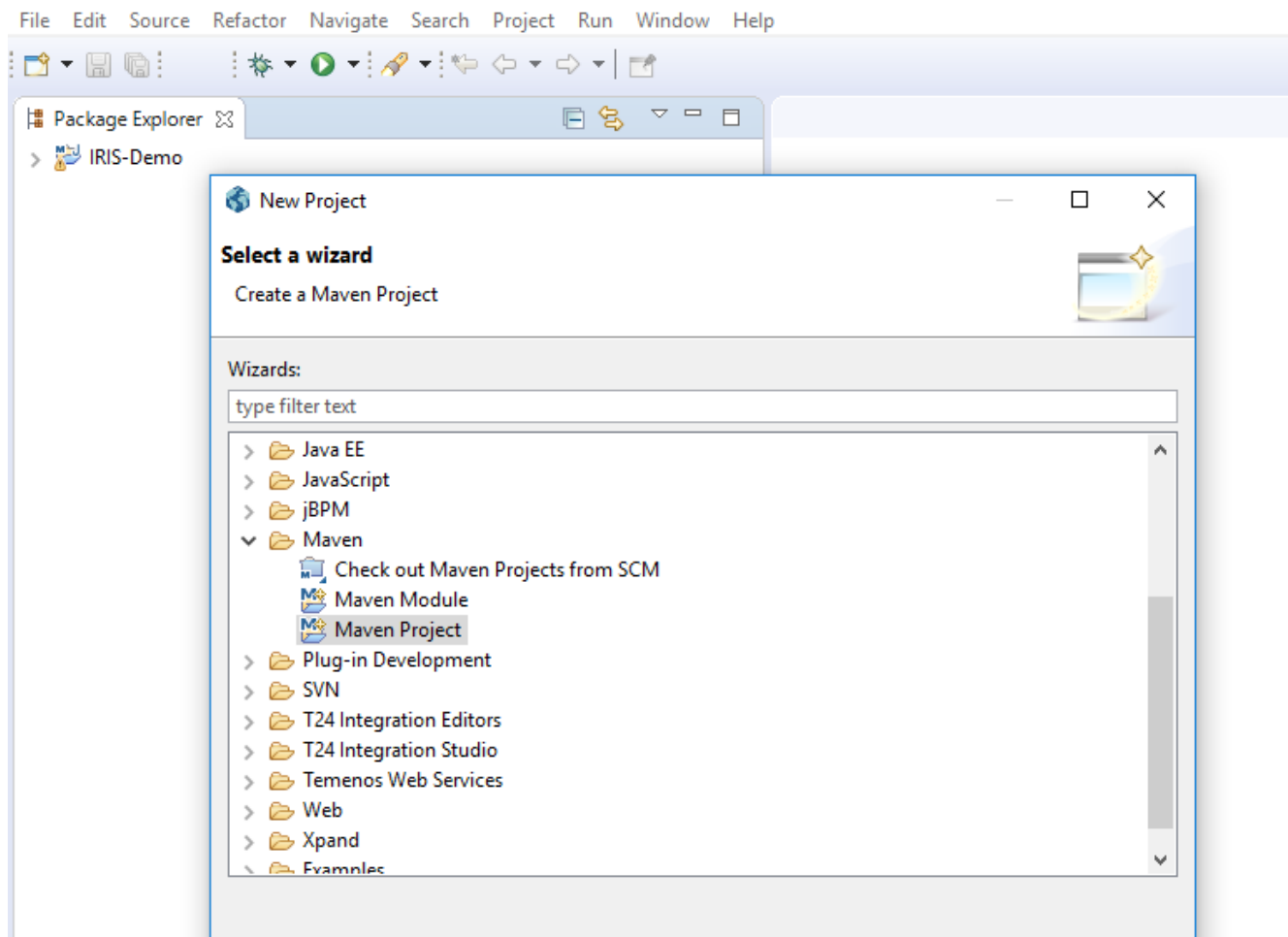




Create Service Container Project

In order to run any API projects, you should have a service container. A service container hosts one or more service projects. It is also a runtime container that hosts the camel runtime and configuration required to connect to downstream systems, i.e.T24

- 1) Open the Design Studio and Click on File -> New -> Maven Project





New Maven Project

Select project name and location

☐ Create a simple project (skip archetype selection)

☒ Use default Workspace location

Location:

☐ Add project(s) to working set

Working set:

▶ Advanced

- 3) Select irf-service-container-archetype and the appropriate version delivered (in this example 18.0.2). if com.temenos.irf is not displayed then configure it by clicking on the configure option displayed in the wizard and select the archetype-catalog.xml (available in t24binaries directory).

New Maven Project

Select an Archetype

Catalog:

Filter:

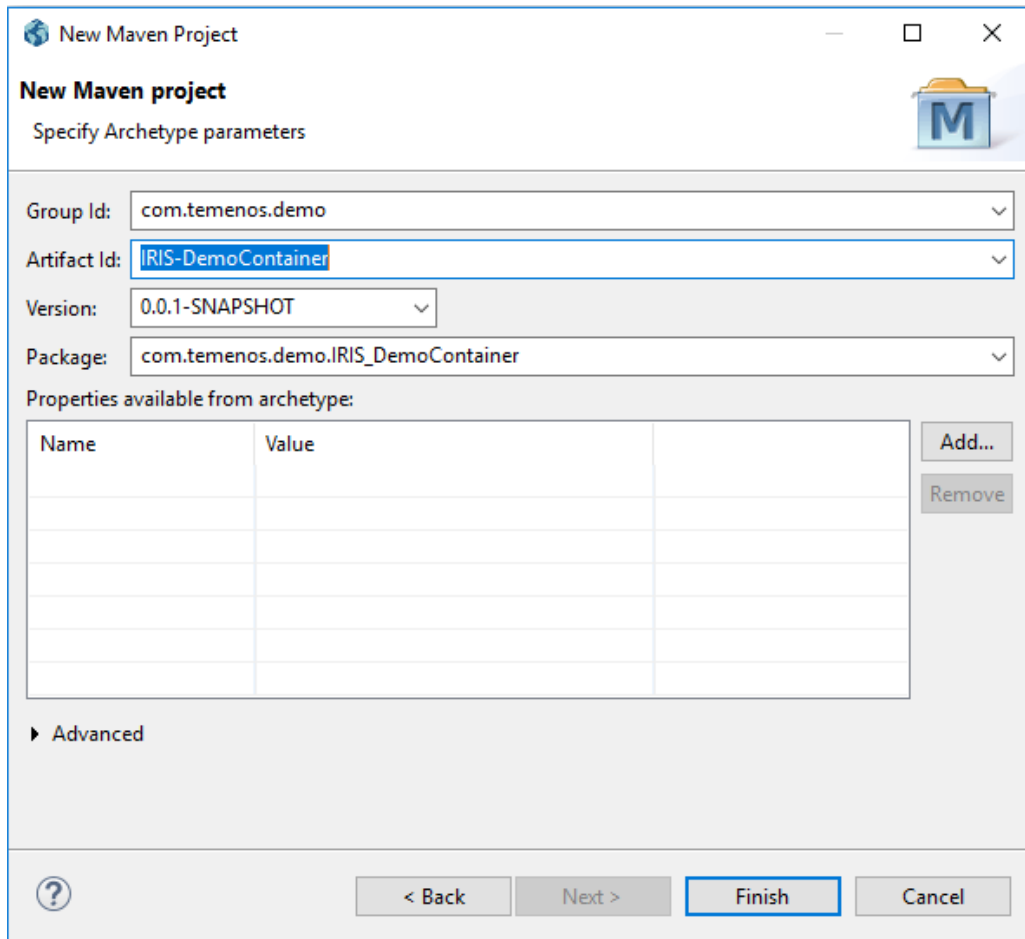
Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	18.0.2
com.temenos.irf	irf-service-container-archetype	18.0.2
org.apache.maven.archetypes	maven-archetype-archetype	1.0
org.apache.maven.archetypes	maven-archetype-j2ee-simple	1.0
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1

☒ Show the last version of Archetype only ☐ Include snapshot archetypes

▶ Advanced



Fill in the Group Id, Artifact Id and Version fields, and click the Finish button.



New Maven Project

Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

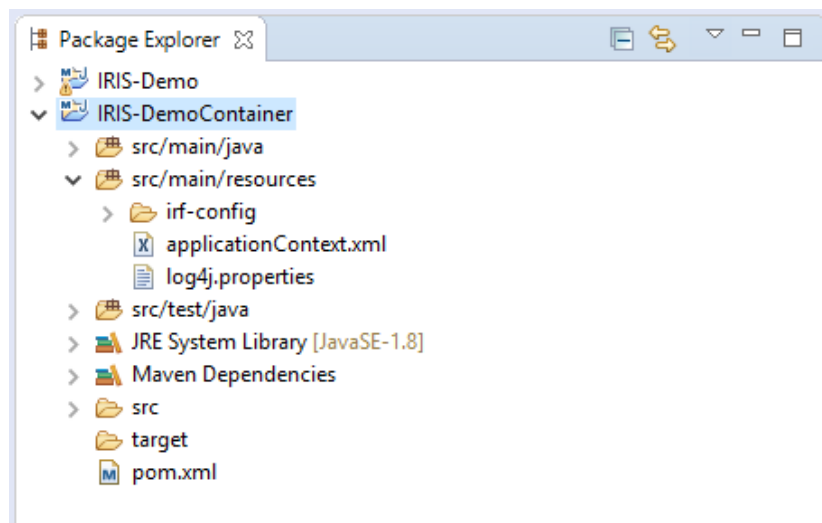
Properties available from archetype:

Name	Value

Advanced

< Back Next > **Finish** Cancel

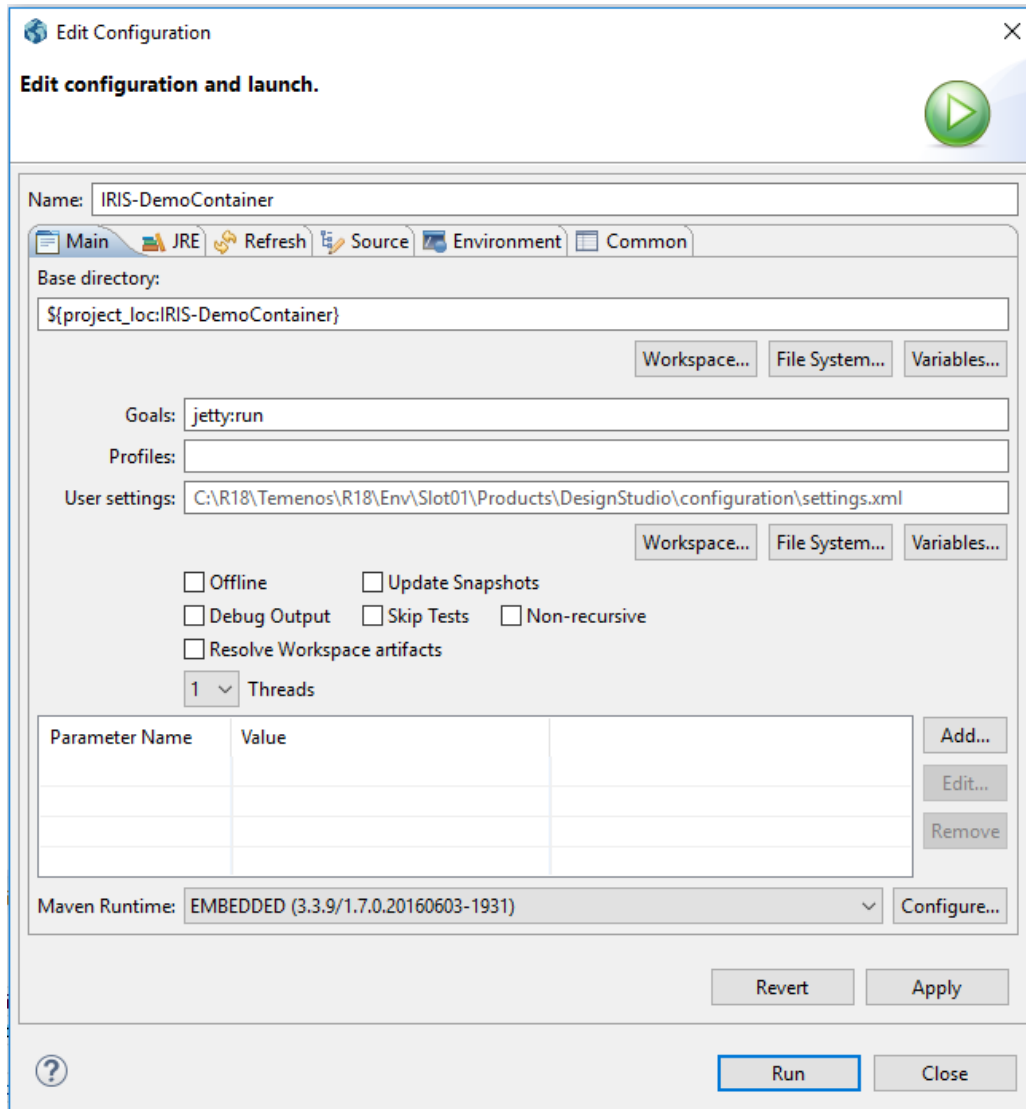
4) You should see a project created with the following structure:





IRIS R18 – Step by Step Guide

5) Right Click and Run As -> Maven Build and set the Goals as jetty:run



From the Design Studio Console, you would see the message as shown below, Started Jetty server,

```
Properties Problems Servers Console
IRIS-DemoContainer (1) [Maven Build] C:\R18\Temenos\R18\Env\Slot01\Products\DesignStudio\jdk\bin\javaw.exe (Jun 15, 2018, 8:51:56 PM)
20:52:17,062 INFO main SpringCamelContext:4026 - Route: direct.uploadCreateProcessor started and consuming from: direct://uploadCreateProcessor
20:52:17,062 INFO main SpringCamelContext:4026 - Route: route16 started and consuming from: servlet:/v1.0.0/api-creator/%7BserviceId%7D?httpMethodRestrict=POST%2COPTIONS&optionsEnabled=true
20:52:17,062 INFO main SpringCamelContext:4026 - Route: route17 started and consuming from: servlet:/v1.0.0/api-creator/%7BzipId%7D?httpMethodRestrict=GET%2COPTIONS&optionsEnabled=true
20:52:17,062 INFO main SpringCamelContext:3195 - Total 4 routes, of which 4 are started
20:52:17,062 INFO main SpringCamelContext:3207 - Apache Camel 2.20.1 (CamelContext: api-creator-service-v1.0.0) started in 0.063 seconds
20:52:17,062 INFO main SpringCamelContext:3146 - Apache Camel 2.20.1 (CamelContext: meta-oa-service-v1.0.0) is starting
20:52:17,062 INFO main ManagedManagementStrategy:205 - JMX is enabled
20:52:17,078 INFO main DefaultTypeConverter:66 - Type converters loaded (core: 192, classpath: 15)
20:52:17,101 INFO main SpringCamelContext:3466 - StreamCaching is not in use. If using streams then its recommended to enable stream caching. See more details at http://camel.apache.org/stream-caching.htm
20:52:17,121 INFO main SpringCamelContext:4026 - Route: direct.meta.oa.getOriginationDefinition started and consuming from: direct://getOriginationDefinition
20:52:17,123 INFO main SpringCamelContext:4026 - Route: getOriginationDefinition started and consuming from: servlet:/v1.0.0/meta/originationDefinitions/%7Btarget%7D?httpMethodRestrict=GET%2COPTIONS&optionsEnabled=true
20:52:17,124 INFO main SpringCamelContext:3195 - Total 2 routes, of which 2 are started
20:52:17,124 INFO main SpringCamelContext:3207 - Apache Camel 2.20.1 (CamelContext: meta-oa-service-v1.0.0) started in 0.062 seconds
20:52:17,129 INFO main ContextLoader:345 - Root WebApplicationContext: initialization completed in 9268 ms
20:52:17,357 INFO main CamelHttpTransportServlet:77 - Initialized CamelHttpTransportServlet[name=CamelServlet, contextPath=]
[INFO] Started o.e.j.m.p.JettyWebAppContext@1ac45389(/, file:///C:/R18/Temenos/R18/Env/Slot01/Products/DesignStudio/Demo/IRIS-DemoContainer/src/main/webapp/,AVAILABLE){file:///C:/R18/Temenos/R18/Env/Slot01/
[INFO] Started ServerConnector@7c4697fc(HTTP/1.1,[http/1.1]){0.0.0.0:9080}
[INFO] Started Jetty Server
```



Creation of Artefacts in T24

Artefacts need to be created to expose the core banking features to external parties. In IRIS R18, T24 Versions and Enquiries are used as base artefacts to expose the core banking features through API development.

Note: If you have existing artefacts, then you can skip this step.

In order for T24 Version and Enquiry artefacts to be used in the Interaction Framework, certain rules must be followed. These rules provide a simple and effective governance framework that,

- 1) Clearly identifies that a given T24 artefact is used in an API
- 2) Allows simple versioning control to be applied to the T24 artefact following semantic versioning
- 3) Provides data type meta data to ensure clean conversion of data from T24 to internet standard data types.
- 4) Provides meaningful operation names

The following rules must be followed:

ID Naming Convention

For Enquiry, the following ID naming convention must be followed:

XX.API.ACCOUNT.BALANCE.Major.Minor.Patch

API is a mandatory keyword in the name of the T24 Enquiry or Version.

Example : EB.API.ACCT.BAL.1.0.0

For Version the following ID naming convention must be followed:

APPLICATION,XX.API.VERB.RESOURCEID.Major.Minor.Patch

Example: FUNDS.TRANSFER, AC.API.CREATE.TRANSFER.1.0.0

Field data typing

Each Enquiry column and Version field must have a data type defined. For Enquiry, this is the

FIELD.DISP.TYPE field, for Version the TOOL.TIP field is used. Fields are considered to be alphanumeric unless they are set as Date or Amount. Amount and date fields must be unformatted, e.g. 20170123 and



1234.56

Description

Each Enquiry and Version must have a description. For Enquiry, this is the DESCRIPT field, for Version the

DESCRIPTION field is used.

Restrictions on Enquiry

Header fields should not be used in API enquiries.

Naming should be in standard vocabulary

The standard Temenos vocabulary exists to bring consistent naming across all API initiatives.
Field / main

resource / sub resource names should exist in the standard vocabulary.

In this example, we create the enquiry **BIL.APL.ACCT.BAL.1.0.0**

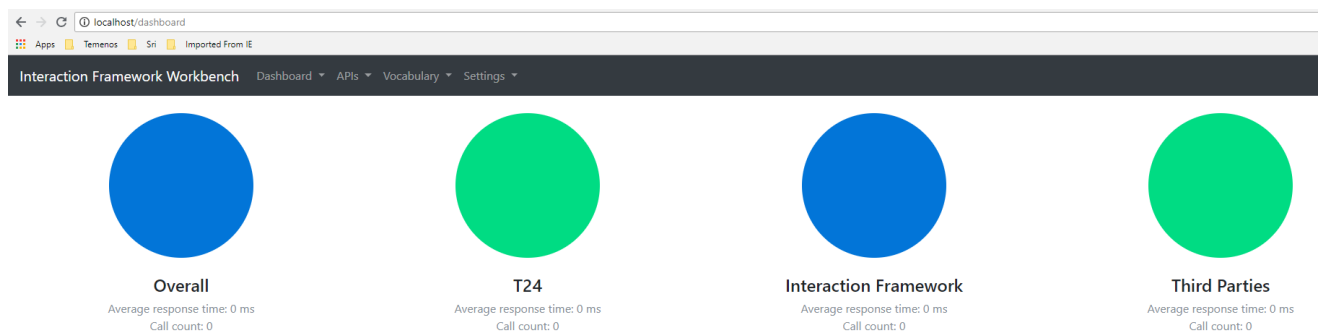
Creation of Provider API

Provider APIs expose core-banking capabilities as RESTful APIs. The key concept is that each Provider API is driven from an inventory that defines the contents of the API in core banking terms. The inventory is used to create the Swagger specification of the API. Together, the inventory file and the generated swagger specification are used to create the service implementation.

Creating an Inventory using Interaction Framework Workbench

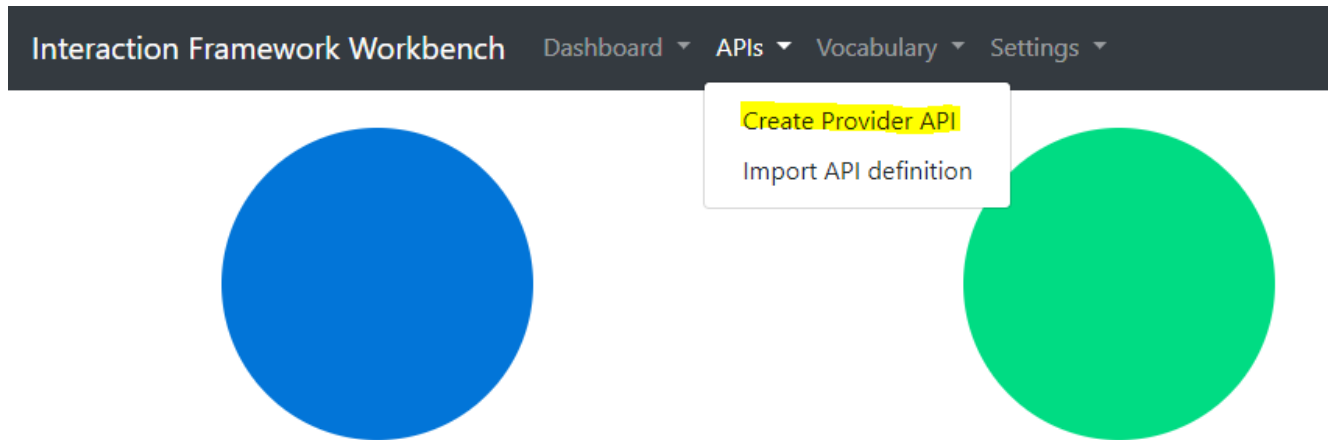
The easiest mechanism to create a Provider API is to use the Interaction Framework Workbench.

- 1) Run the Apache server and Open the Interaction Framework Workbench using the URL <http://localhost> or <http://localhost:8080/dashboard>

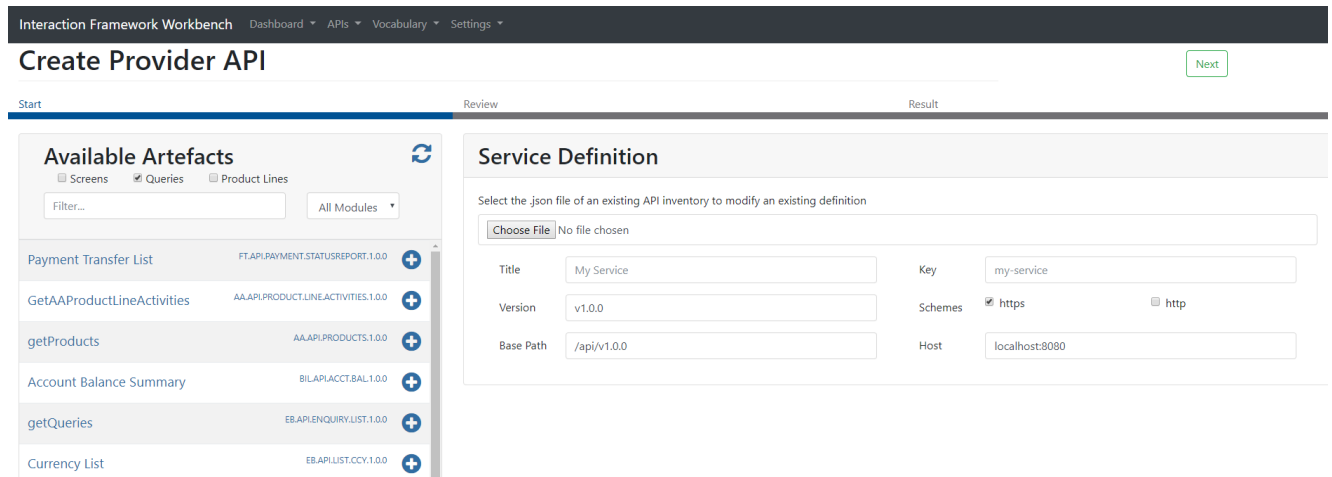




2) Select APIs -> Create Provider API



This will connect to T24 using Jetty server and list the artefacts created in T24 for provider APIs as shown below.



3) Select the artefact, in this example we have used Account Balance Summary BIL.API.ACCT.BAL.1.0.0, (created under the section [Creation of Artefacts in T24](#))



Fill the Title, Key, Version, schemes, Base Path and Host

Interaction Framework Workbench Dashboard APIs Vocabulary Settings

Create Provider API

Start Review Result

Available Artefacts

Screens Queries Product Lines

Filter... All Modules

- Payment Transfer List FT.APLPAYMENT.STATUSREPORT.1.0.0
- GetAAPProductLineActivities AA.APLPRODUCT.LINE.ACTIVITIES.1.0.0
- getProducts AA.APLPRODUCTS.1.0.0
- Account Balance Summary BIL.APLACCT.BAL.1.0.0**
- getQueries EB.APLINQUIRY.LIST.1.0.0
- Currency List EB.APLLIST.CCY.1.0.0
- getScreens EB.APLVERSION.LIST.1.0.0
- Payment Initiation Response FT.APLPAYMENT.INITIATION.RESPONSE.1.0.0
- GetAAPProductLines AA.APLPRODUCT.LINES.1.0.0
- Payment Transfer Status FT.APLPAYMENT.TRANSFER.STATUS.1.0.0

Service Definition

Select the json file of an existing API inventory to modify an existing definition

Choose File No file chosen

Title AccountBalanceSummary Key AccountBalanceSummary

Version v1.0.0 Schemes ☒ https ☒ http

Base Path /api/v1.0.0 Host localhost:8080

^ BIL.APL.ACCT.BAL.1.0.0

Main Parameters

Operation Account Balance Summary

Domain party

URL /acct/{id}/bal

HTTP Method GET

Operation security Public

URL Query Parameters

Name	Data Type
id	string

4) Fill the Main Parameters,

- Operation
- Domain
- URL

URL Query Parameters,

- Name
- Data Type

Do the Selection field Mapping with the field names provided in URL Query Parameters (these are the field names used by consumer) against the T24 selection fields.



Note: Check all the values against the Vocabulary defined in the workbench. Goto Settings and disable the Vocabulary, if it is not needed.

Interaction Framework Workbench Dashboard ▾ APIs ▾ Vocabulary ▾ Settings ▾

Settings

Connecting to: **default**@**http://127.0.0.1:8080** using **api.server.com** as default API host and **/api** as default base path.

Validating APIs against vocabulary is ON

Servers

Servers Add Server

Active	Name	URL
<input checked="" type="radio"/>	default	http://127.0.0.1:8080

Defaults

Defaults Modify

Base Path	/api
Host	api.server.com
Validate APIs	true

Modify Defaults

Base Path

/api

Host

api.server.com

☒ Validate provider API definition against vocabulary

Cancel Save

If T24 VERSION is selected, only the method is required to be set. The payload will be configured in Swagger directly from the T24 Model. POST should be used to create new resources, PUT to amend an existing resource, GET for viewing a resource and DELETE for removing a resource.

If T24 ENQUIRY is selected, the URL parameters need to be mapped to the ENQUIRY selection fields.



^ BIL.API.ACCT.BAL.1.0.0

Main Parameters

Operation

getAccountBalanceSummary

Domain

holdings

URL

/acct/

HTTP Method

GET

Operation security

Public

URL Query Parameters

Name	Data Type
accountNo	string
customerNo	string

Selection Mapping

Selection Field	Operand	Parameter	Constant
Account.No	equals	accountNo	
Customer	equals	customerNo	

In the above example, the T24 Selection fields Account.No is mapped against accountNo and the field Customer is mapped against customerNo.

- Check the box in URL Query Parameters, if you want to make the field as Mandatory field.

Click Next

Inventory

```

{
  "paths": [
    {
      "method": "GET",
      "url": "/holdings/acct/",
      "operationId": "getAccountBalanceSummary",
      "operationSecurity": "Public",
      "resources": [
        {
          "key": "BIL.API.ACCT.BAL.1.0.0",
          "resourceType": "Query",
          "selections": [
            {
              "field": "ACCOUNT.NUMBER",
              "param": "accountNo",
              "operand": "EQ",
              "required": "",
              "type": "string"
            },
            {
              "field": "CUSTOMER",
              "param": "customerNo",
              "operand": "EQ",
              "required": "",
              "type": "string"
            }
          ]
        }
      ]
    }
  ],
  "version": "v1.0.0",
  "title": "AccountBalanceSummary",
  "key": "AccountBalanceSummary",
  "schemes": [
    "http",
    "https"
  ],
  "basepath": "/api/v1.0.0",
  "host": "localhost:8080"
}

```

Service Definition

Title	AccountBalanceSummary	Key	AccountBalanceSummary
Version	v1.0.0	Schemes	<input checked="" type="checkbox"/> https <input checked="" type="checkbox"/> http
Base Path	/api/v1.0.0	Host	localhost:8080

Service Endpoints

Path	/acct/
HTTP Method	GET
Operation	getAccountBalanceSummary
Operation Security	Public
Target	BIL.API.ACCT.BAL.1.0.0
Target Type	Query

- Verify the details in Service Definition and Service Endpoints.
- Click Finish to submit the provider API. The result provides a link to download a zip file containing the swagger spec, Camel routes, mappings and mock responses.



Interaction Framework Workbench
Dashboard
APIs
Vocabulary
Settings

Create Provider API
Previous

Start
Review
Result

Upload complete.

Click the link to download a zip containing the generated artefacts

Download

Build Service Project

- 1) Extract the components (which was created and downloaded in the previous step), you would see the 3 folders as shown below,

Name

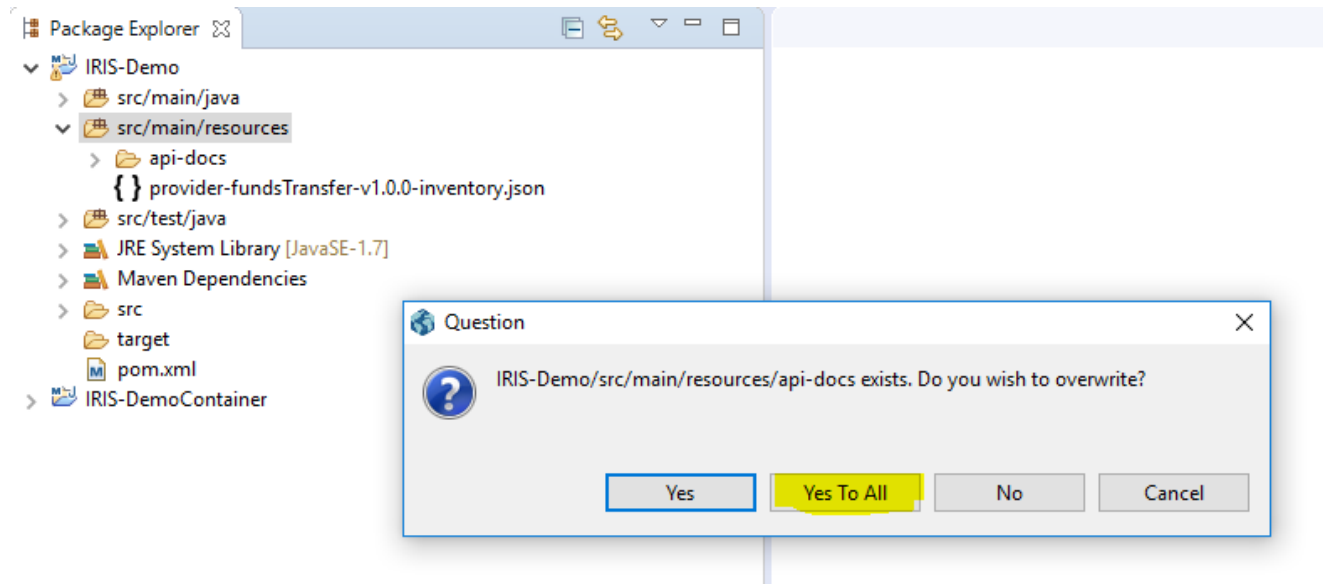
- api-docs
- inventory
- services

- 2) Copy/Drag these folders into the service project src/main/resources created above (created under the section [Create Service Project](#)). In this example, the service project is IRIS-Demo

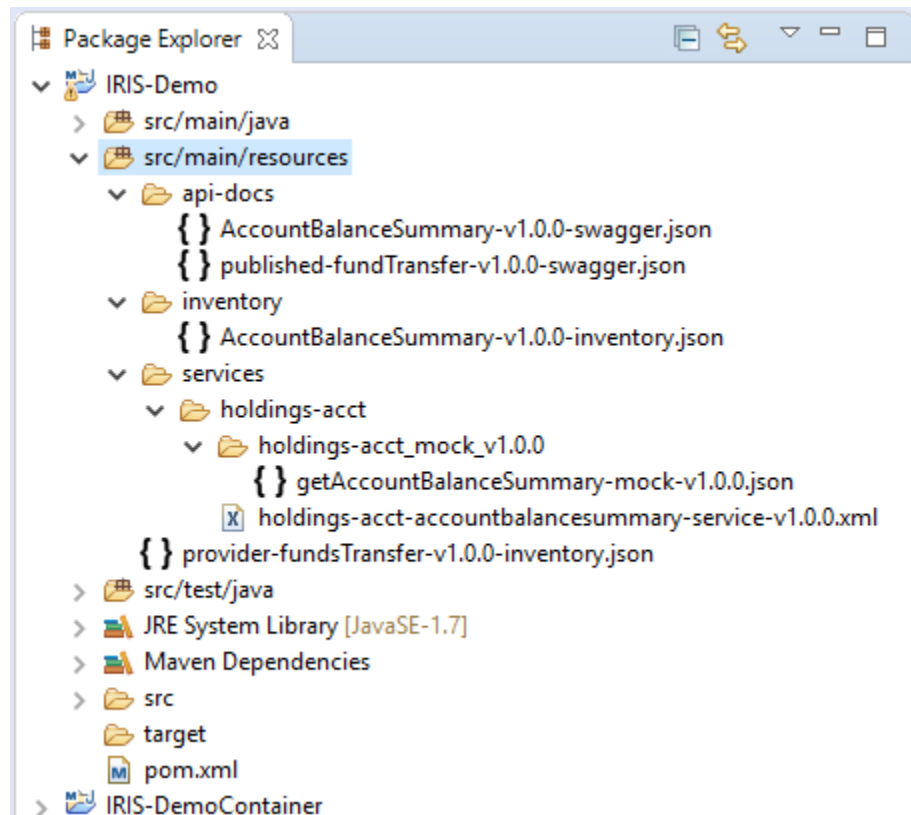
Package Explorer
IRIS-Demo
src/main/java
src/main/resources
api-docs
provider-fundsTransfer-v1.0.0-inventory.json
src/test/java
JRE System Library [JavaSE-1.7]
Maven Dependencies
src
target
pom.xml
IRIS-DemoContainer

File and Folder Operation
Select how files and folders should be imported into the project:
Copy files and folders
Link to files and folders
Link to files and recreate folder structure with virtual folders
Create link locations relative to: PROJECT_LOC
Configure Drag and Drop Settings...
OK
Cancel

- 3) Select Ok

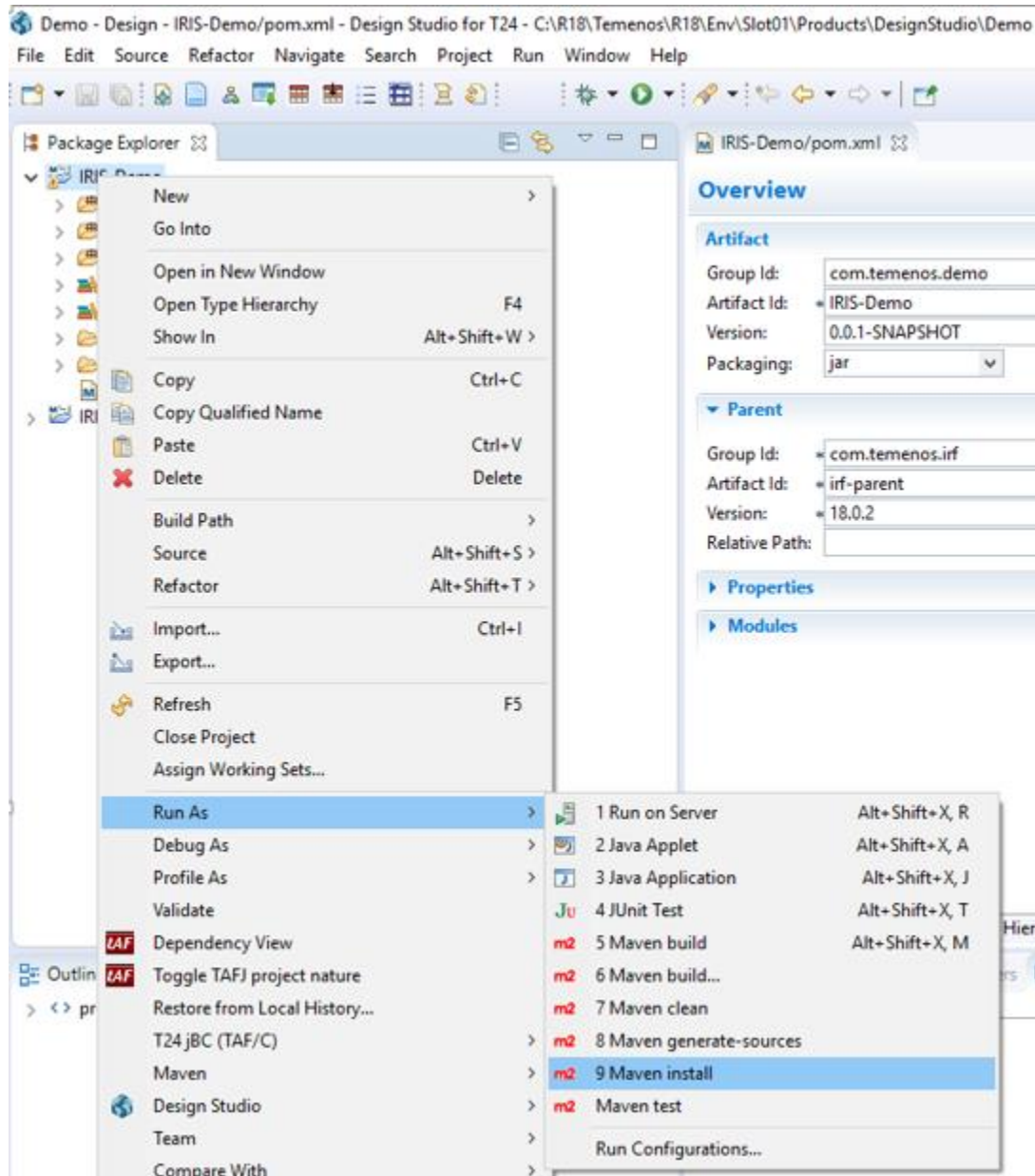


- 4) Click on Yes To All. You should see the project with the following structure, in this example - the AccountBalanceSummary can be seen in api-docs, inventory and services.



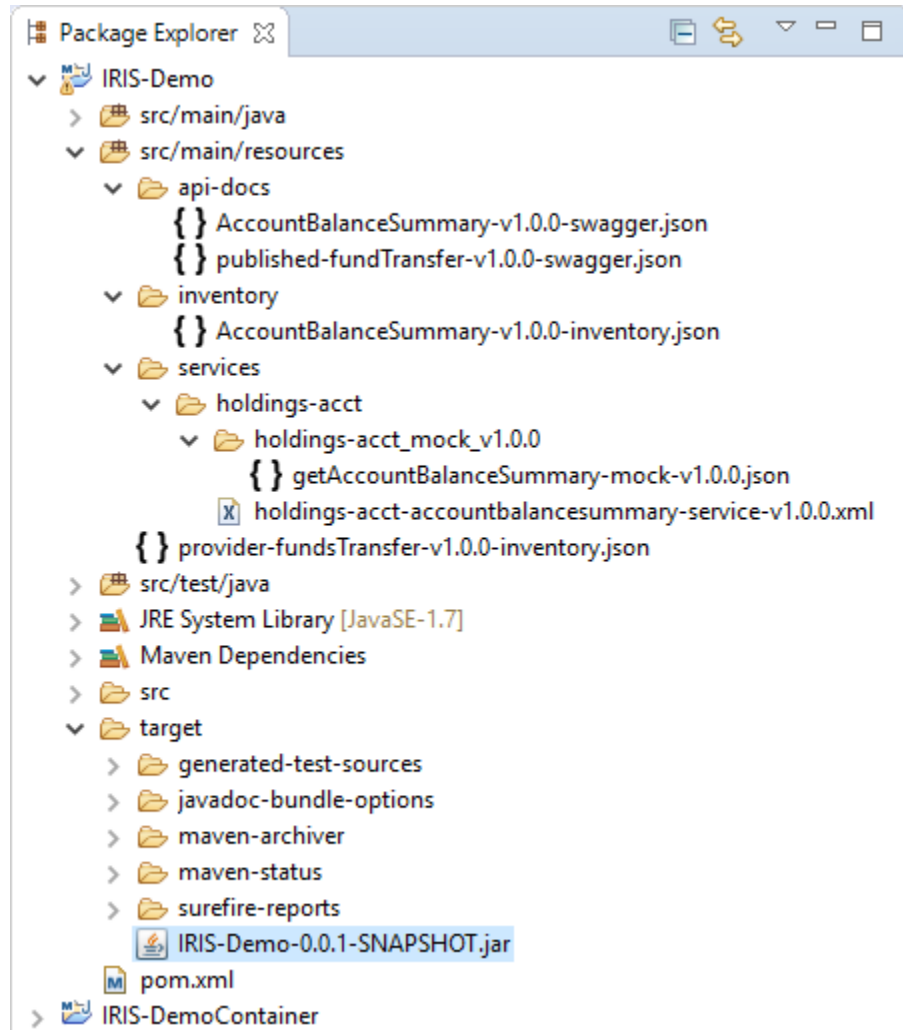


- 5) Right click on the Service Project (in this example – IRIS-Demo) and select Run As -> Maven Install.





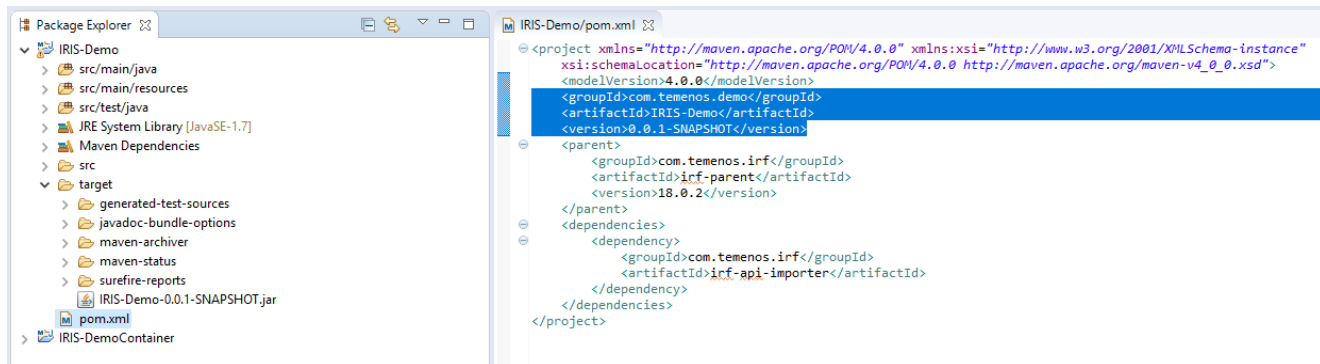
You should see the target folder with jar file as shown below,





Build Container Project

- 1) Open the pom.xml of the Service project (IRIS-Demo) as shown below,

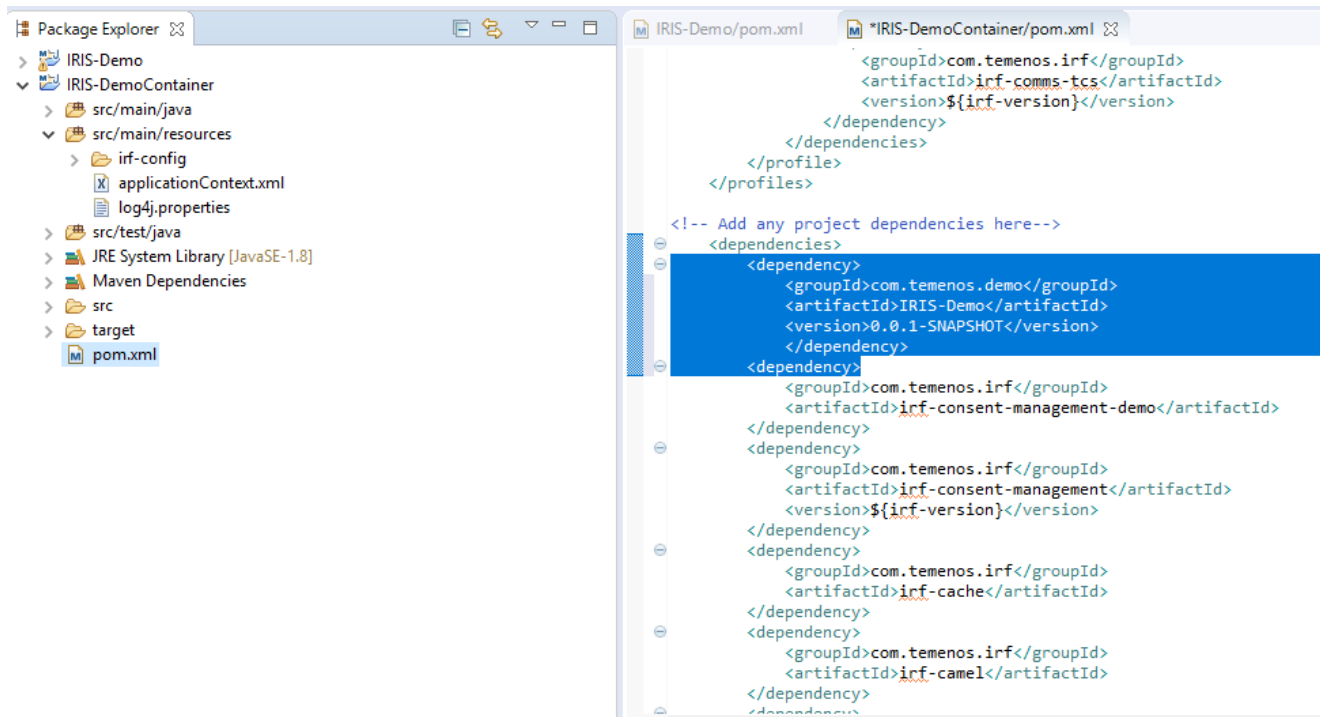


- 2) Copy the 3 lines groupId, artifactId and version as shown above and paste it into the <Dependencies> section in pom.xml (enclosed between <Dependency> tag) of the container project created earlier under the section [Create Service Container Project](#) (in this example – IRIS-DemoContainer).

Note: There are several dependencies tags in the pom.xml file - this dependency needs to be added to the

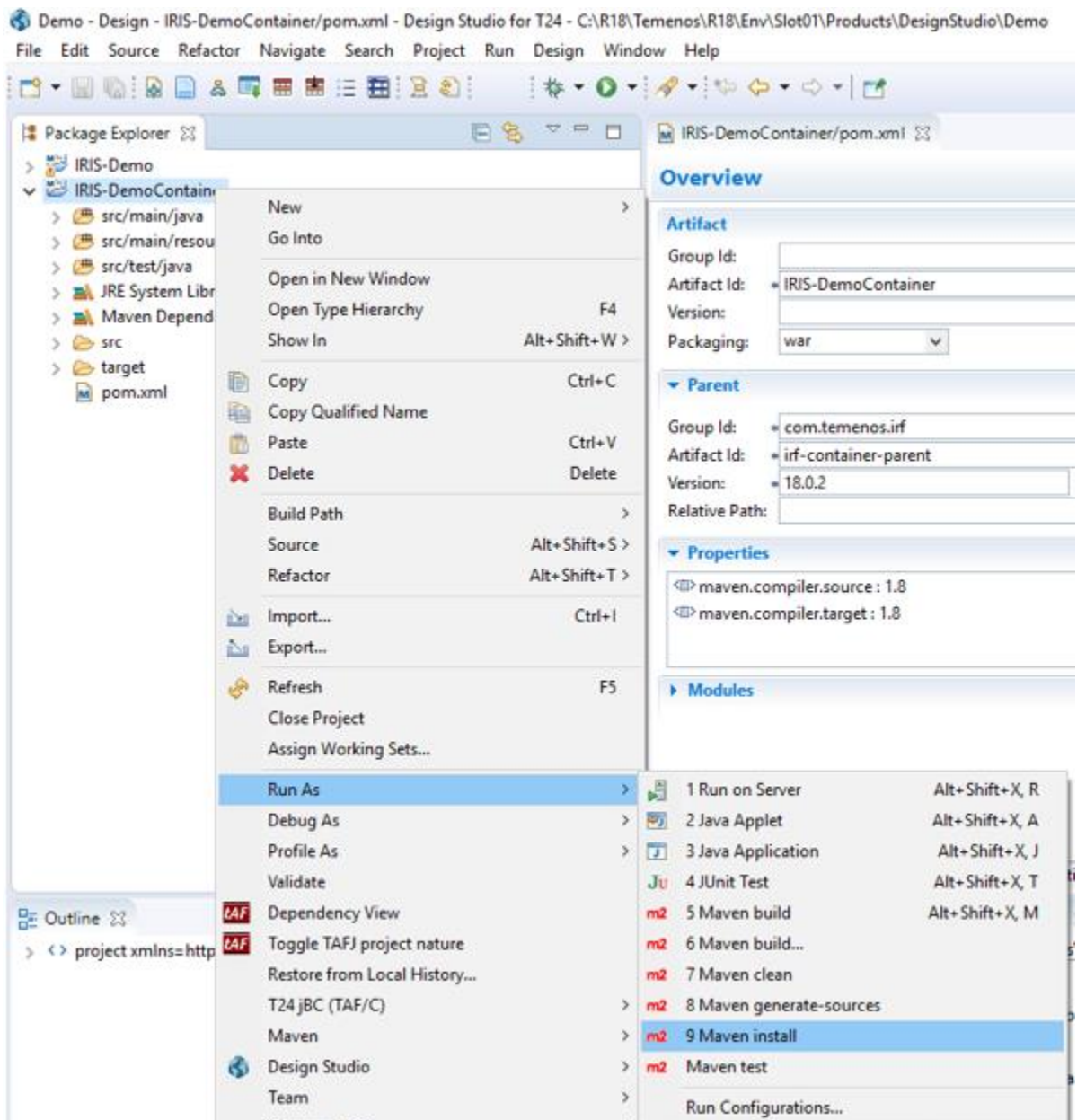
MAIN dependency, below the comment:

<!-- Add any project dependencies here-->





- 3) Right Click the Service Container Project (in this example, IRIS-DemoContainer) and select Run As -> Maven Install





- 4) Start the Jetty server by Right clicking on the container project and select Run As -> Maven Build and provide the Goal as jetty:run

Edit Configuration
 ✕

Edit configuration and launch.

Name: IRIS-DemoContainer (3)

Main JRE Refresh Source Environment Common

Base directory:

\${project_loc:IRIS-DemoContainer}
 Workspace... File System... Variables...

Goals: jetty:run

Profiles:

User settings: C:\R18\Temenos\R18\Env\Slot01\Products\DesignStudio\configuration\settings.xml

Workspace... File System... Variables...

☐ Offline
 ☐ Update Snapshots

☐ Debug Output
 ☐ Skip Tests
 ☐ Non-recursive

☐ Resolve Workspace artifacts

1 Threads

Parameter Name	Value

Add... Edit... Remove

Maven Runtime: EMBEDDED (3.3.9/1.7.0.20160603-1931)

Configure...

Revert Apply

Run Close



Testing

Test the Provider API created using the above steps using Postman.

- 1) Use the below URL in postman to get the meta data of APIs,

<http://localhost:8080/api/v1.0.0/meta/apis>

You should get the response as shown below, which should have your service

The screenshot shows a Postman interface with a GET request to `http://localhost:8080/api/v1.0.0/meta/apis`. The response is a JSON object with the following structure:

```
215 {
216   "method": "GET",
217   "uri": "/api/v1.0.0/reference/currencies/{currencyId}"
218 },
219 {
220   "method": "GET",
221   "uri": "/api/v1.0.0/reference/currencies/"
222 }
223 ],
224 "key": "reference-service-v1.0.0-swagger"
225 },
226 {
227   "services": [
228     {
229       "service": "holdings-acct.service.v1.0.0",
230       "endPoints": [
231         {
232           "method": "GET",
233           "uri": "/api/v1.0.0/holdings/acct/"
234         }
235       ]
236     }
237   ]
238 },
239 "key": "AccountBalanceSummary-v1.0.0-swagger"
240 },
241 {
242   "services": [
243     {
244       "service": "consent.service.v1.0.0",
245       "endPoints": [
246         {
247           "method": "DELETE",
248           "uri": "/api/v1.0.0/party/consents/{consentId}"
249         },
250         {
251           "method": "POST",
252           "uri": "/api/v1.0.0/party/consents"
253         }
254       ]
255     }
256   ]
257 }
```



- 2) Use Postman to test the api created, (in this example – Account Balance Summary created under the section Creation of Provider API),

The URL can be constructed using the values provided for the below details, the values can be identified based on the input in [Creation of Provider API](#)

- Host
- base path
- Domain
- URL

As per the values provided in [Creation of Provider API](#)

- Host – <http://localhost:8080>
- Base path - /api/v1.0.0
- Domain – holdings
- URL - /acct/

<http://localhost:8080/api/v1.0.0/holdings/acct/?accountNo=14017>



You should the JSON response as shown below.

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/v1.0.0/holdings/acct/?accountNo=14017`. The response is displayed in the 'Body' tab, formatted as JSON. The response structure includes a 'header' object with audit information and a 'body' array containing account details.

```
1 {
2   "header": {
3     "audit": {
4       "T24_time": 1938,
5       "parse_time": 1
6     },
7     "page_start": 0,
8     "page_token": "a223056d-196f-4a5a-a677-34940c4bcbf7",
9     "total_size": 1,
10    "page_size": 50
11  },
12  "body": [
13    {
14      "Cleared Bal": "21,004,720.77",
15      "Ledger Bal": "21,004,720.77",
16      "Locked Amount": "0",
17      "Acct No": "14017",
18      "Useable Bal": "21,004,720.77",
19      "Ccy": "USD",
20      "Working Bal": "21,004,720.77",
21      "Product": "Current Account",
22      "Name": "CITIGROUP"
23    }
24  ]
25 }
```



Deployment options

- 1) Once the service project is build, the user can deploy it using either of the following application servers:
 - Jboss
 - Weblogic
 - Websphere
- 2) You can configure the IRIS R18 solution using TAFJ (JMS Connector) or TAFC (TCS and TOCFEE Connectors).

Set the value as **false** as shown below for the profile **buildForStandalone** in pom.xml of the container project.

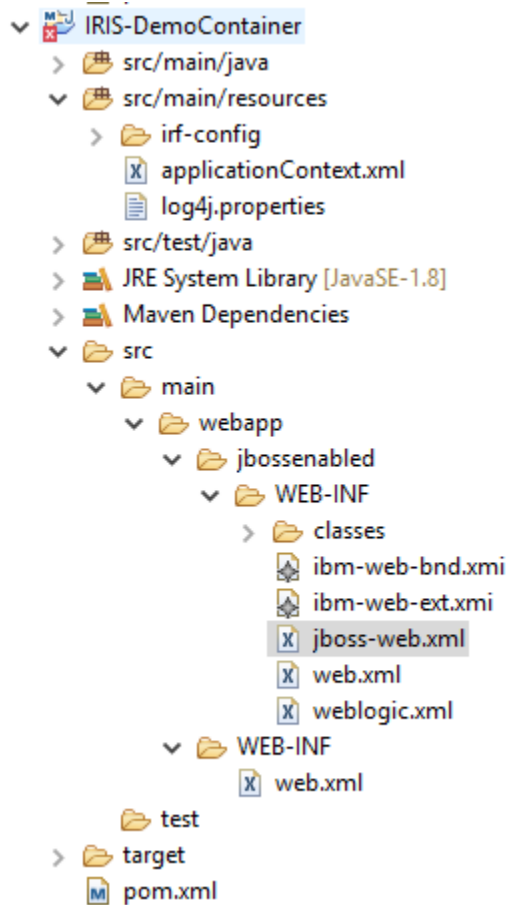
```
<profile>
  <id>buildForStandalone</id>
  <activation>
    <activeByDefault>>false</activeByDefault>
  </activation>
```

Set the value as **true** as shown below for the profile **buildForJMS** in pom.xml of the container project, the name provided in the tag <FinalName> is the name of the war file.

```
<profile>
  <id>buildForJMS</id>
  <activation>
    <activeByDefault>true</activeByDefault>
  </activation>
  <build>
    <finalName>DemoirisR18</finalName>
```

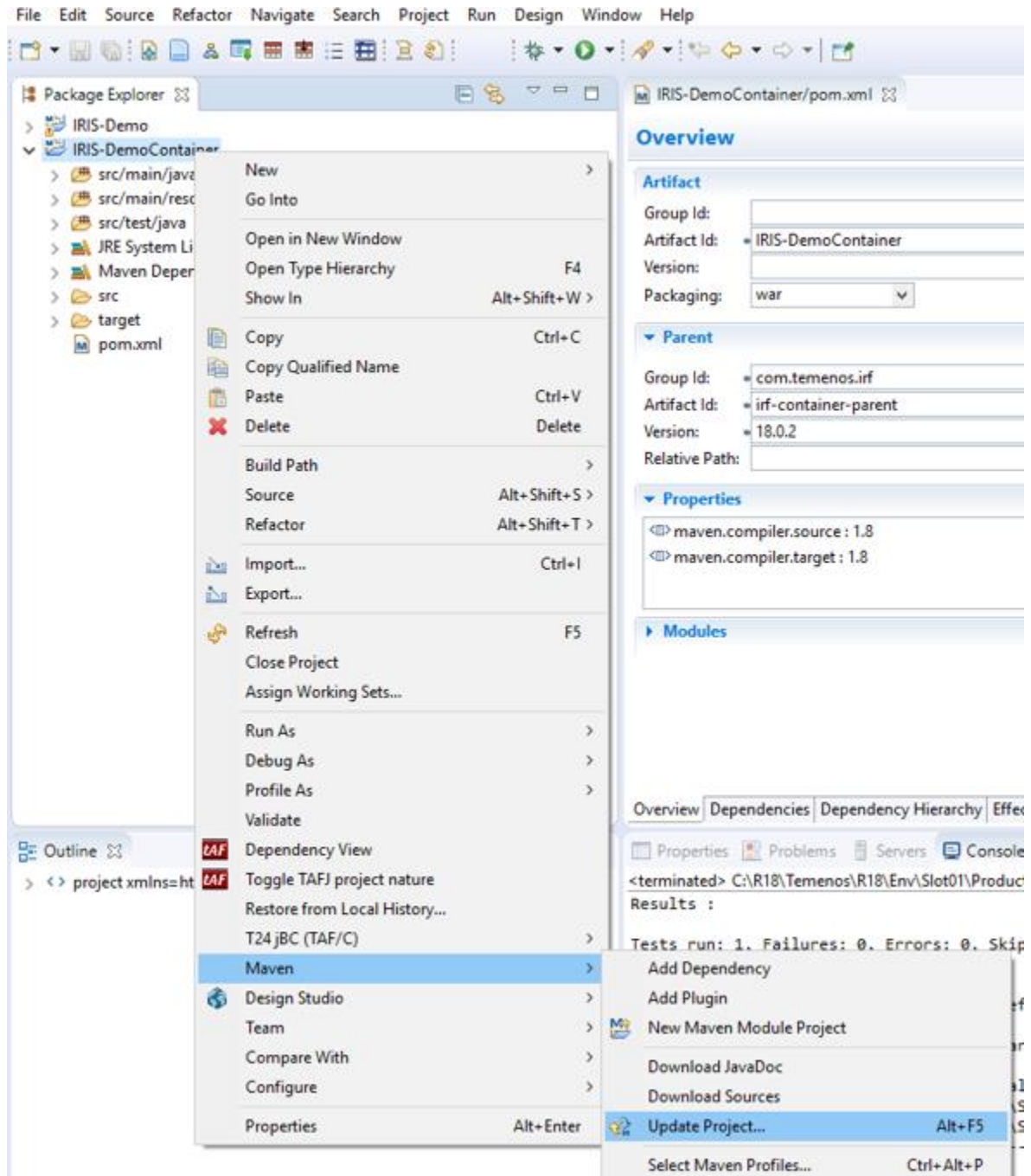


- 3) Check the JMS resources configuration under Container project -> src -> main -> webapp -> jbossenabled -> WEB-INF -> classes -> <<the respective web server file>> before building the final war file.





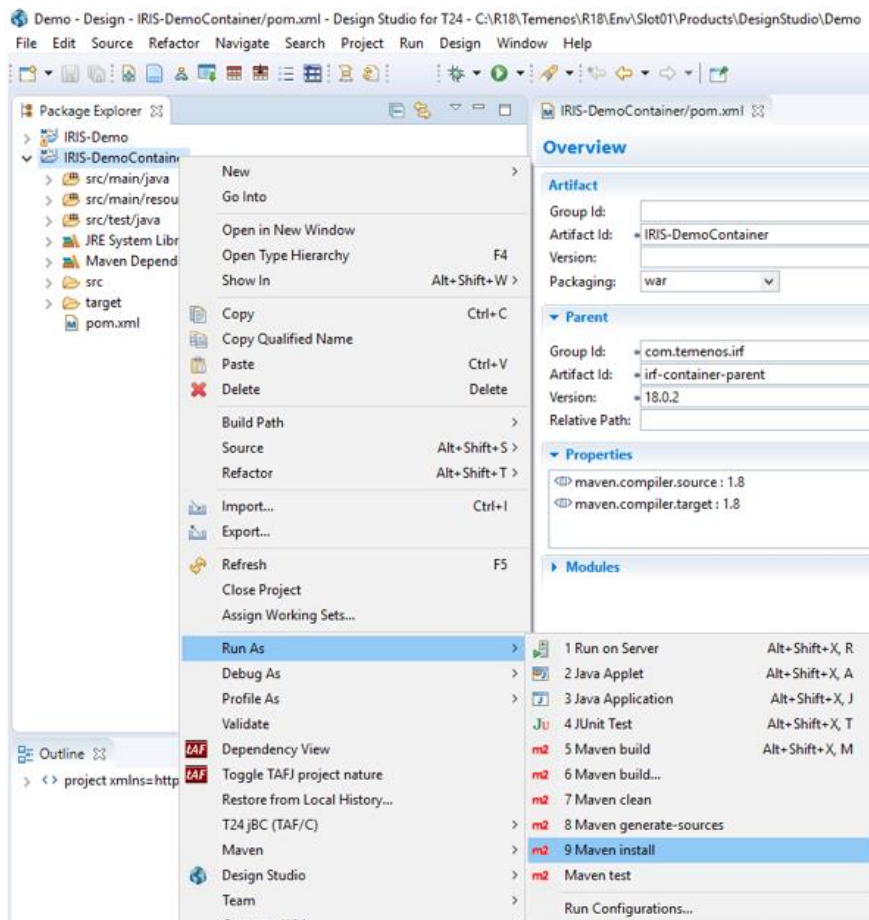
- 4) Right click on the Container project (in this example, IRIS-DemoContainer) and select Maven -> Update Project.



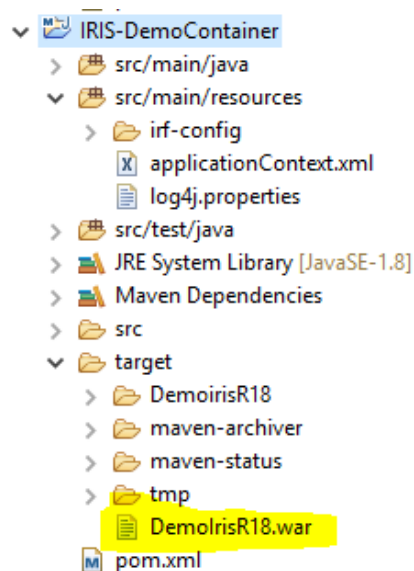


IRIS R18 – Step by Step Guide

- 5) Right click on the Container project (in this example, IRIS-DemoContainer) and select Run As -> Maven Install.



War file will be created under the target folder with the name provided in the tag <finalName>.

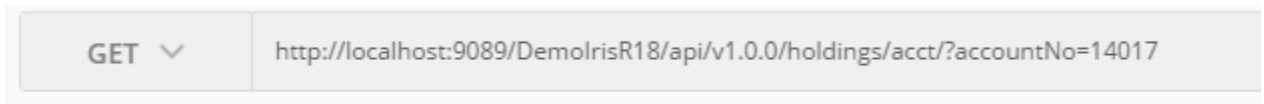




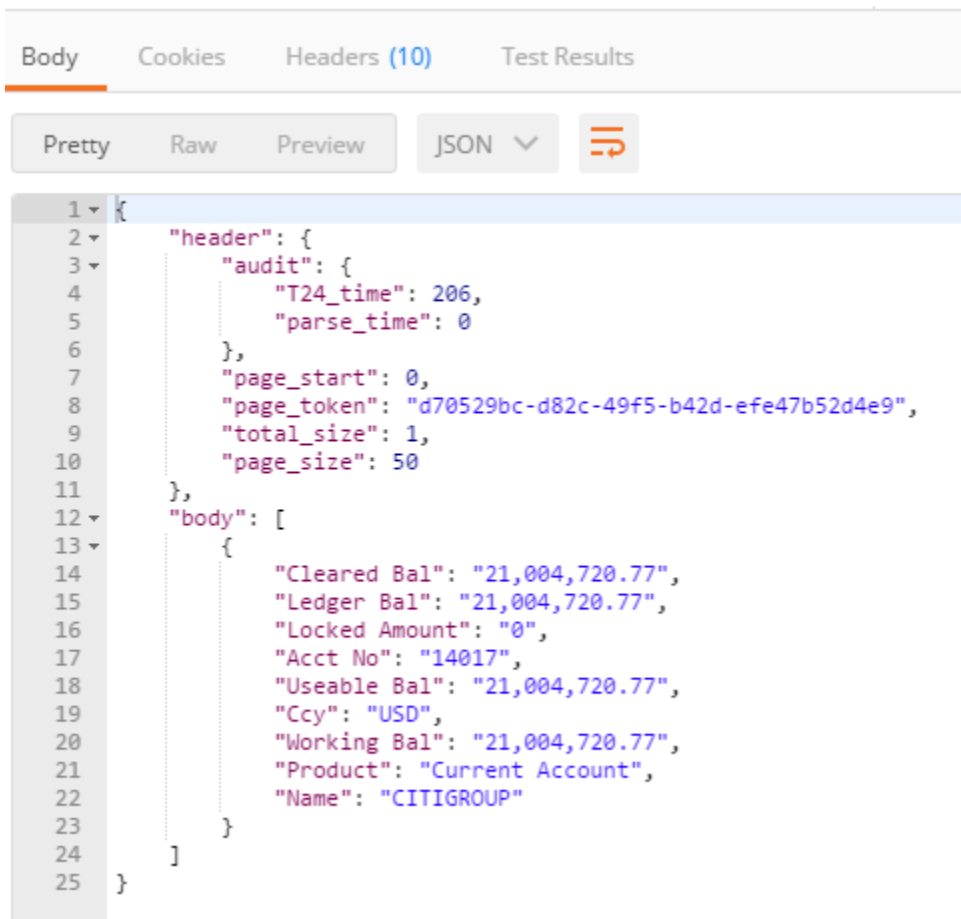
Deployment and Testing

- 1) Deploy the war generated in the above step in any application server (jboss, websphere etc...) and use postman to test,

Request:



Response:





Swagger Specification

The below URL will help the users to get the swagger documentation url for all the services supported in this deployment.

Request:

<http://localhost:9089/DemoIrisR18/api/v1.0.0/meta/apidocs>

Response:

```

1 {
2   "header": {},
3   "body": [
4     {
5       "title": "AccountBalanceSummary",
6       "key": "AccountBalanceSummary-v1.0.0",
7       "url": "http://localhost:9089/DemoIrisR18/api/v1.0.0/meta/apidocs/AccountBalanceSummary-v1.0.0"
8     },
9     {
10      "title": "Reference - System Data Service",
11      "key": "reference-service-v1.0.0",
12      "url": "http://localhost:9089/DemoIrisR18/api/v1.0.0/meta/apidocs/reference-service-v1.0.0"
13    },
14    {
15      "title": "Product Service",
16      "key": "product-service-v1.0.0",
17      "url": "http://localhost:9089/DemoIrisR18/api/v1.0.0/meta/apidocs/product-service-v1.0.0"
18    },
19    {
20      "title": "Funds Transfer Published API's",
21      "key": "published-fundTransfer-v1.0.0",
22      "url": "http://localhost:9089/DemoIrisR18/api/v1.0.0/meta/apidocs/published-fundTransfer-v1.0.0"
23    },
24    {
25      "title": "Meta - APIs about APIs",
26      "key": "meta-t24-service-v1.0.0",
27      "url": "http://localhost:9089/DemoIrisR18/api/v1.0.0/meta/apidocs/meta-t24-service-v1.0.0"
28    }
29  ]
30 }

```

The url available in the response of API Docs request can be used to get the swagger api documentation of the provider apis.

As per the above example,

<http://localhost:9089/DemoIrisR18/api/v1.0.0/meta/apidocs/AccountBalanceSummary-v1.0.0>



```

1 {
2   "swagger": "2.0",
3   "info": {
4     "version": "v1.0.0",
5     "title": "AccountBalanceSummary"
6   },
7   "host": "localhost:8080",
8   "basePath": "/api/v1.0.0",
9   "schemes": [
10    "http",
11    "https"
12  ],
13  "paths": {
14    "/holdings/acct/": {
15      "get": {
16        "operationId": "getAccountBalanceSummary",
17        "produces": [
18          "application/json"
19        ],
20        "parameters": [
21          {
22            "name": "accountNo",
23            "in": "query",
24            "required": false,
25            "type": "string"
26          },
27          {
28            "name": "customerNo",
29            "in": "query",
30            "required": false,
31            "type": "string"
32          },
33          {
34            "name": "page_size",
35            "in": "query",
36            "description": "The total number of records per page",
37            "required": false,
38            "type": "number"
39          },
40          {
41            "name": "page_start",
42            "in": "query"

```