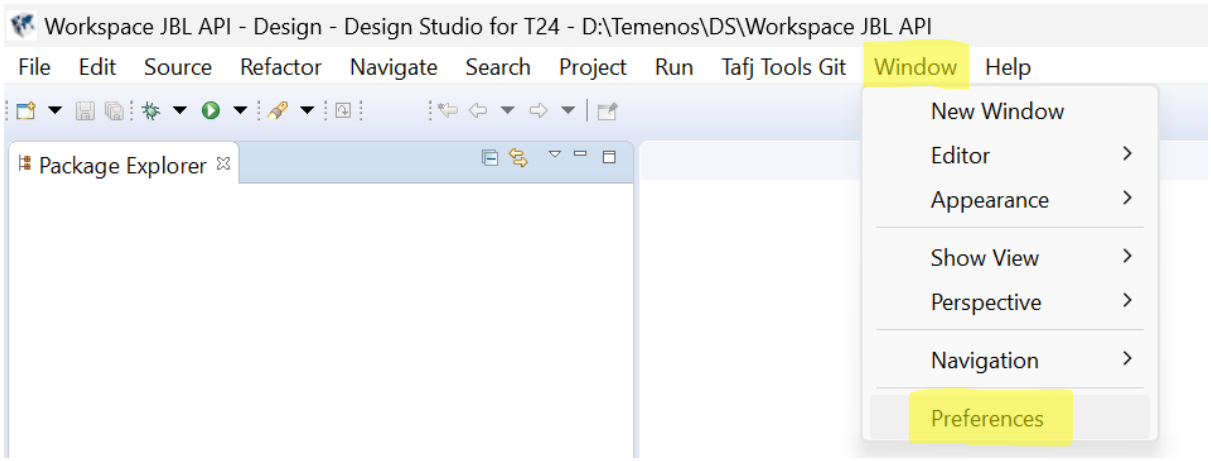
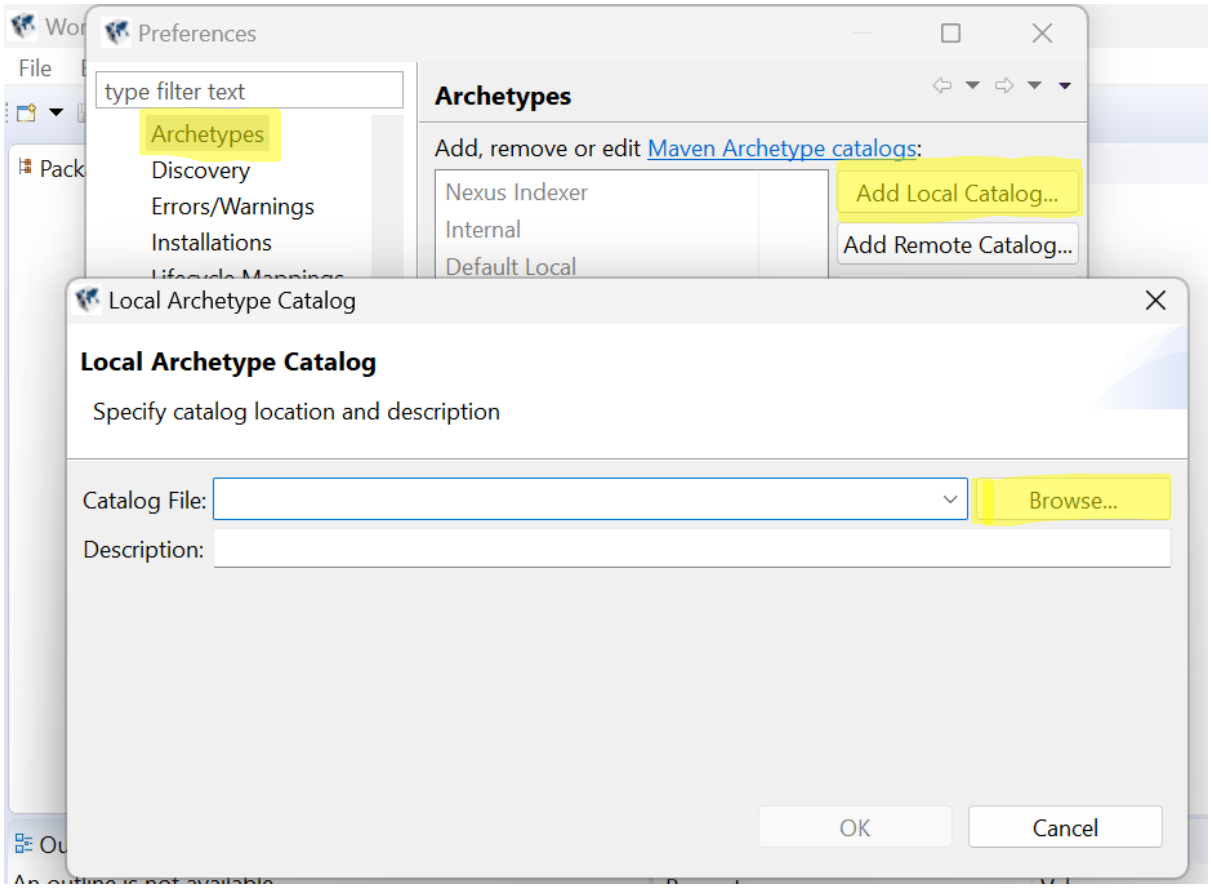
IRIS API

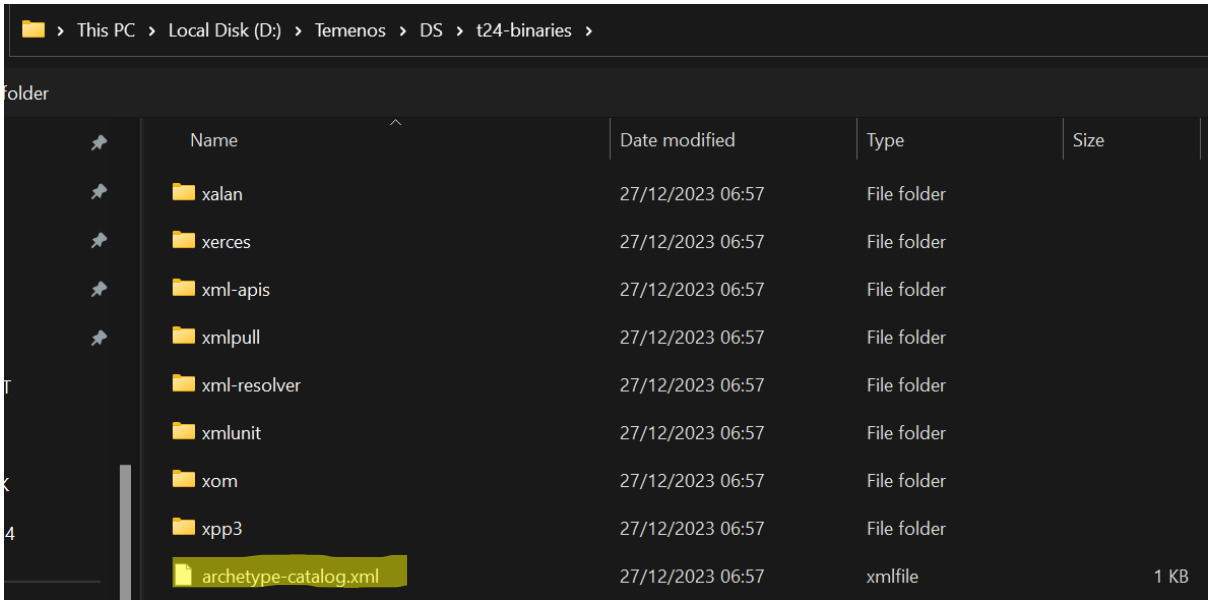
Adding Archetype in Design Studio:

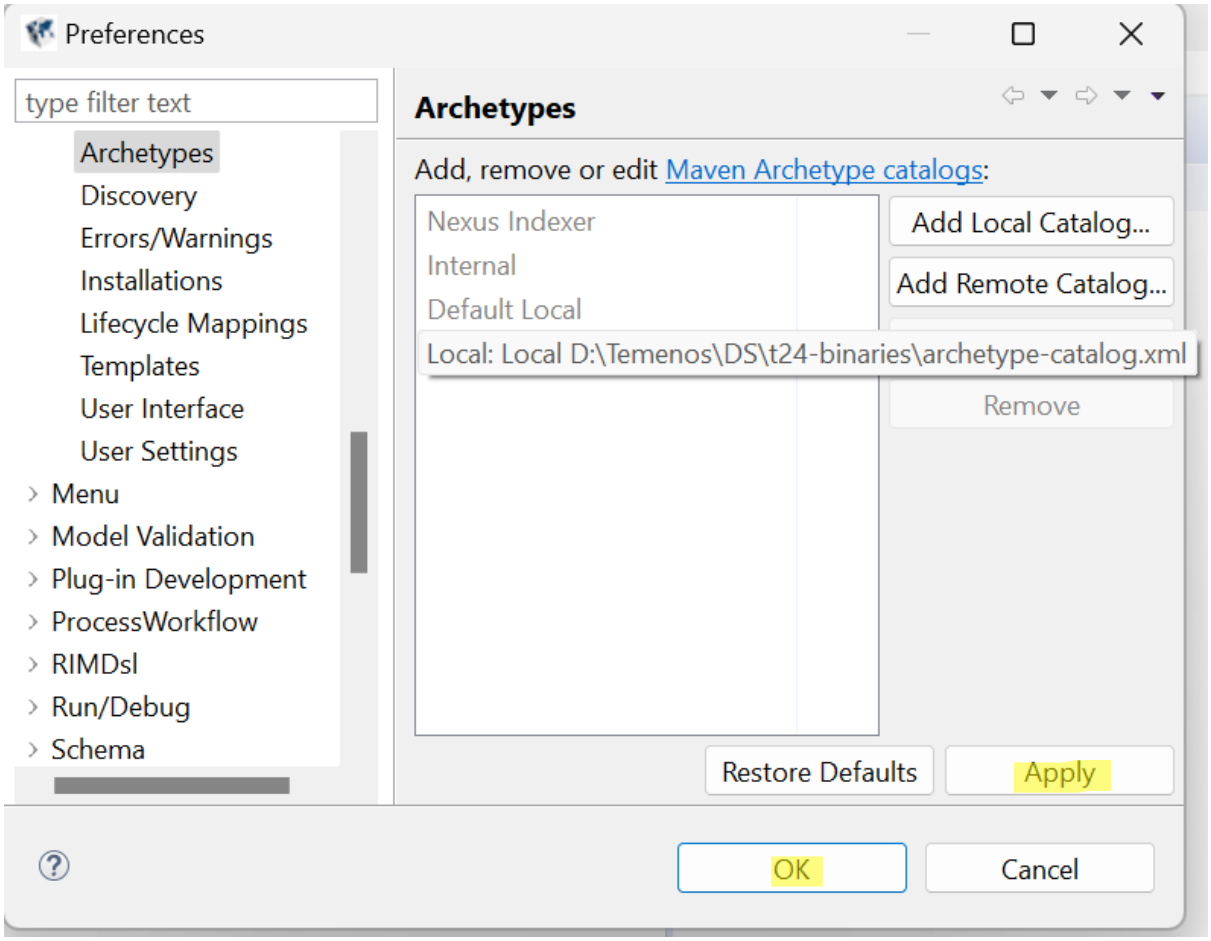


Window > Preferences > Archetypes > Add Local Catalog

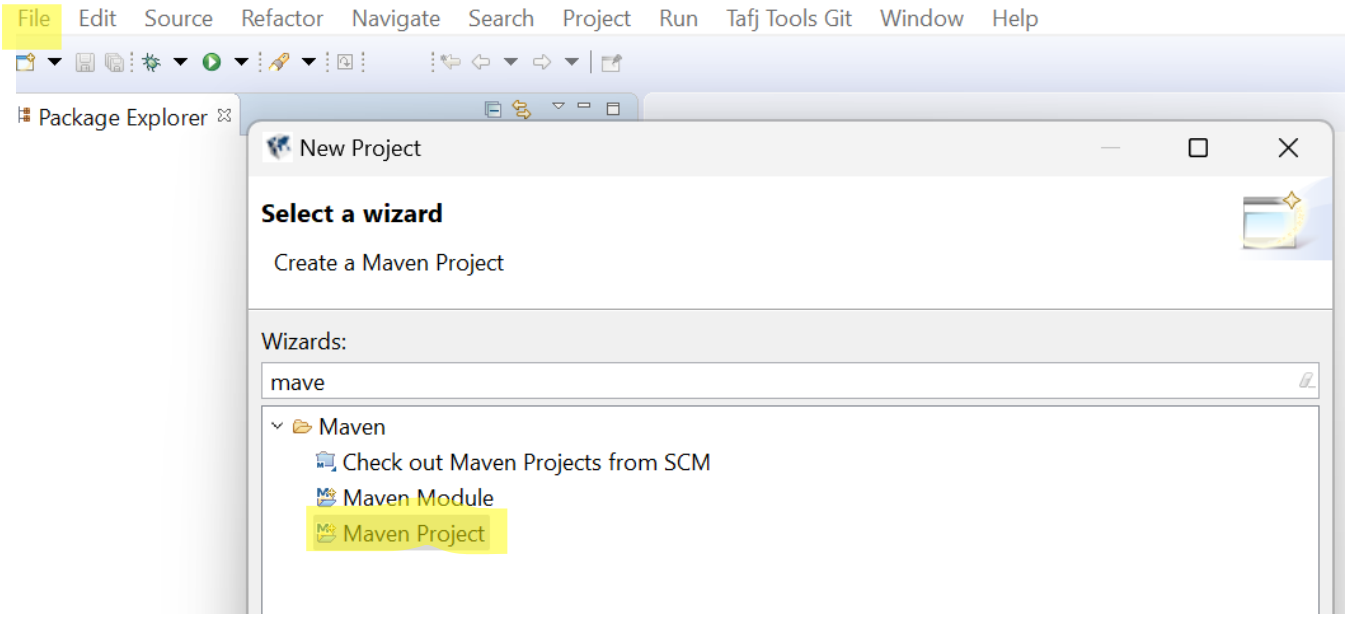


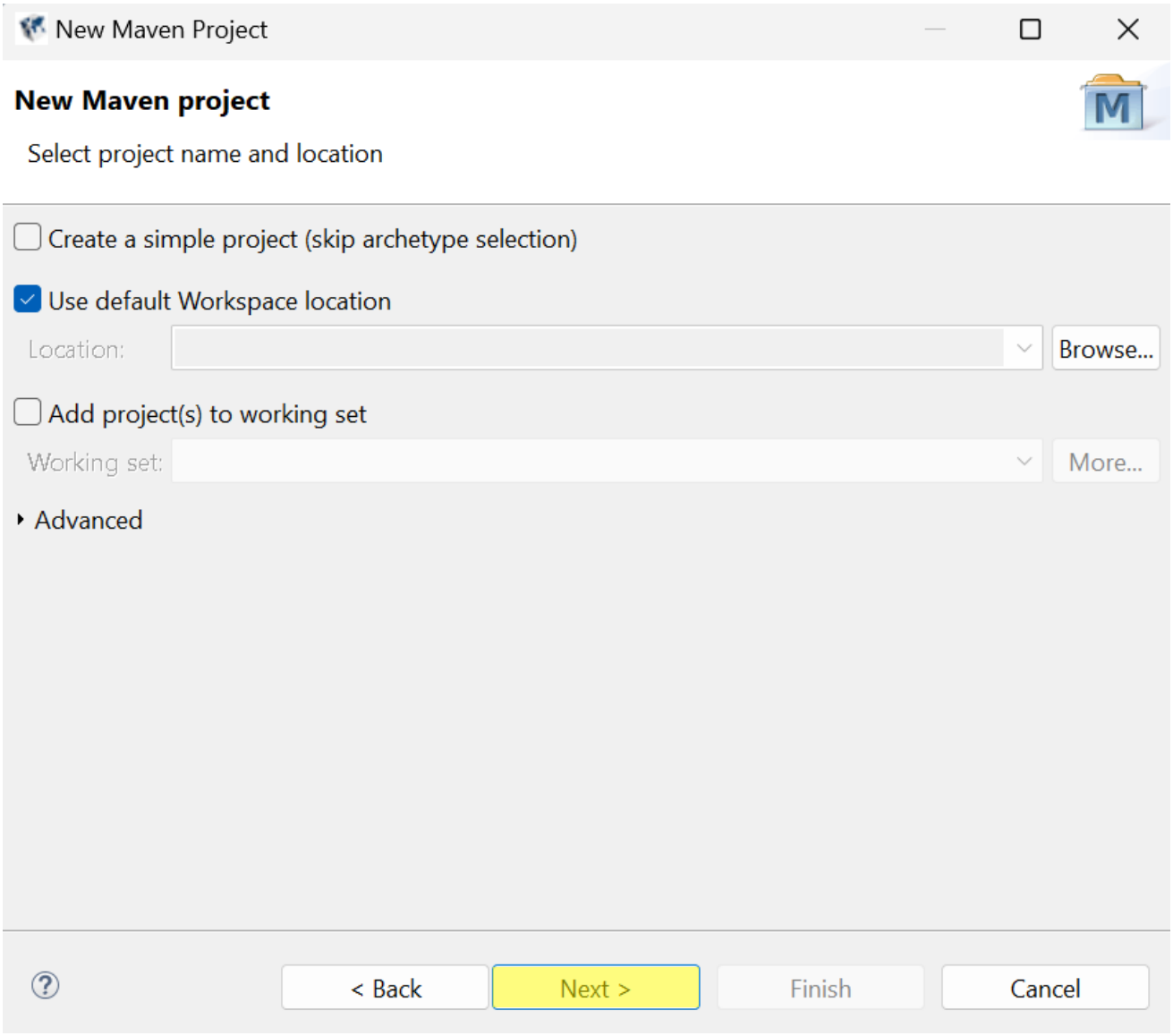
T24-binaries > D:\Temenos\DS\t24-binaries -- archetype-catalog.xml



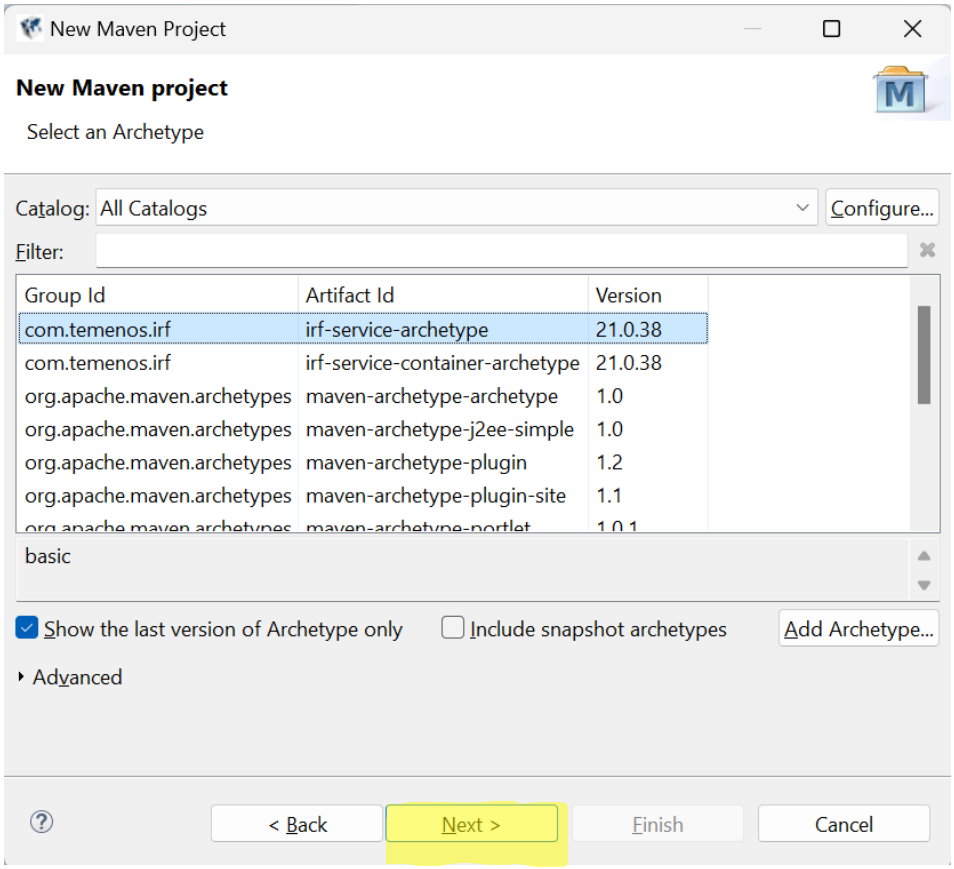


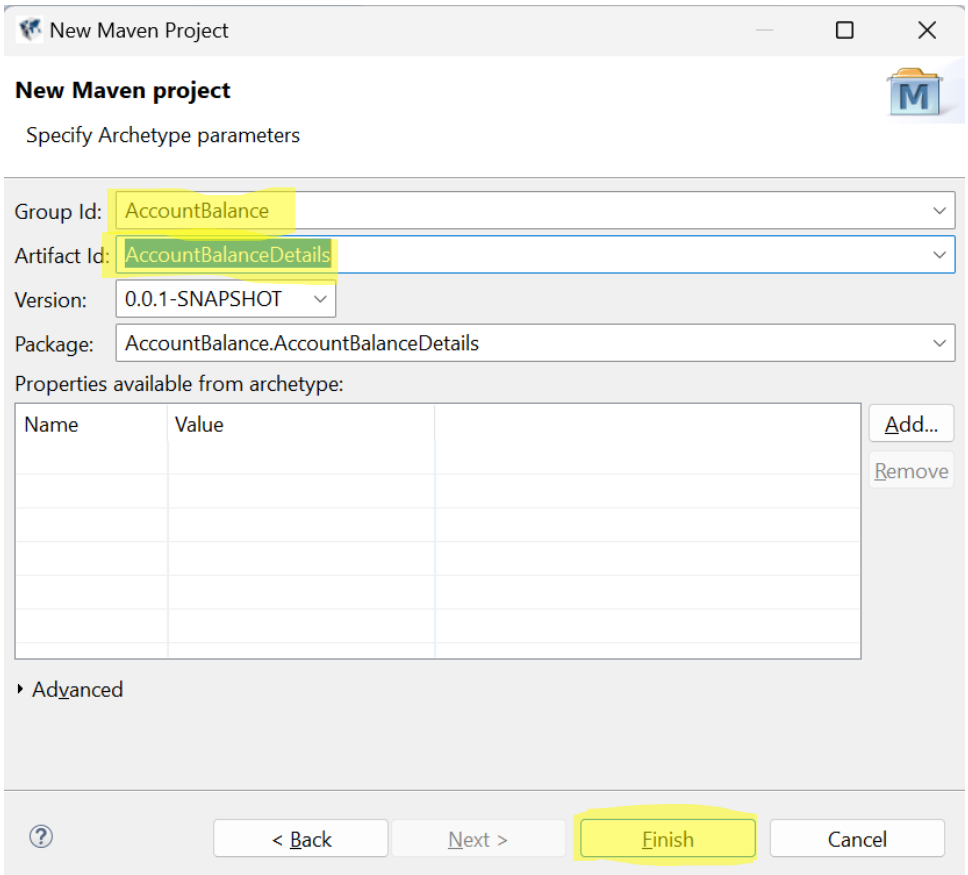
New Mavel Project from DS:

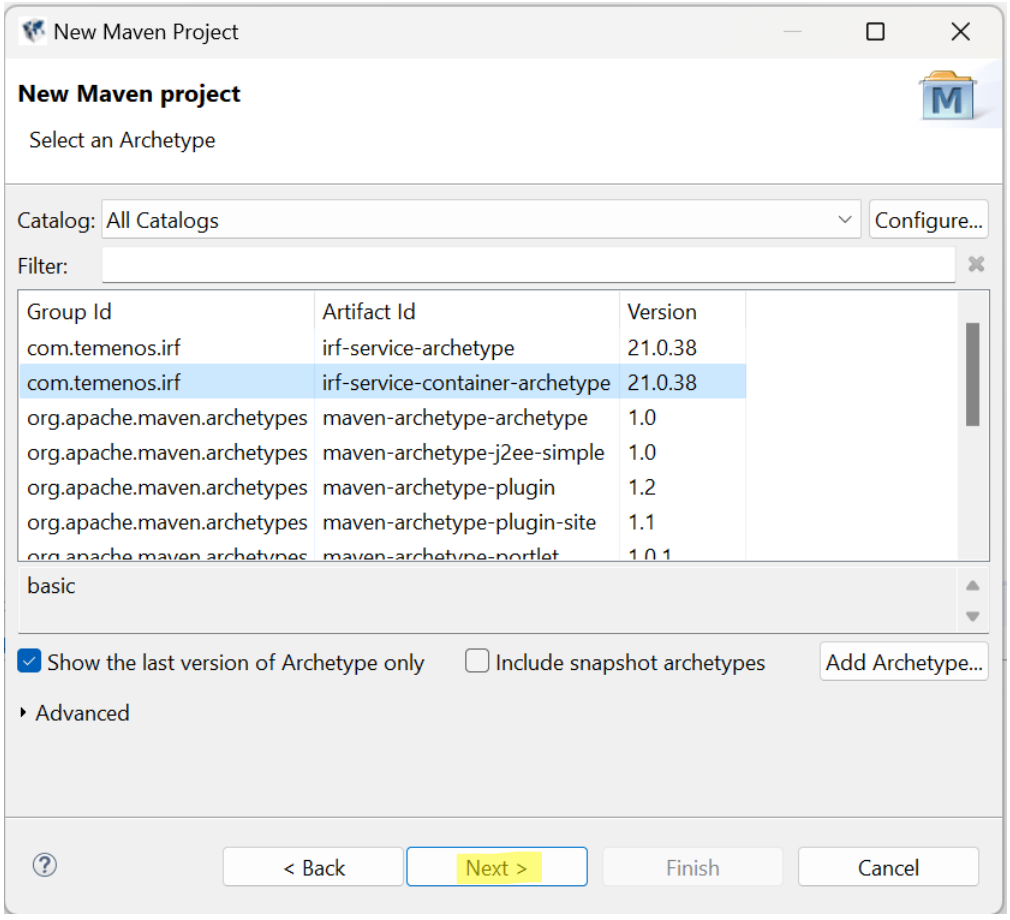


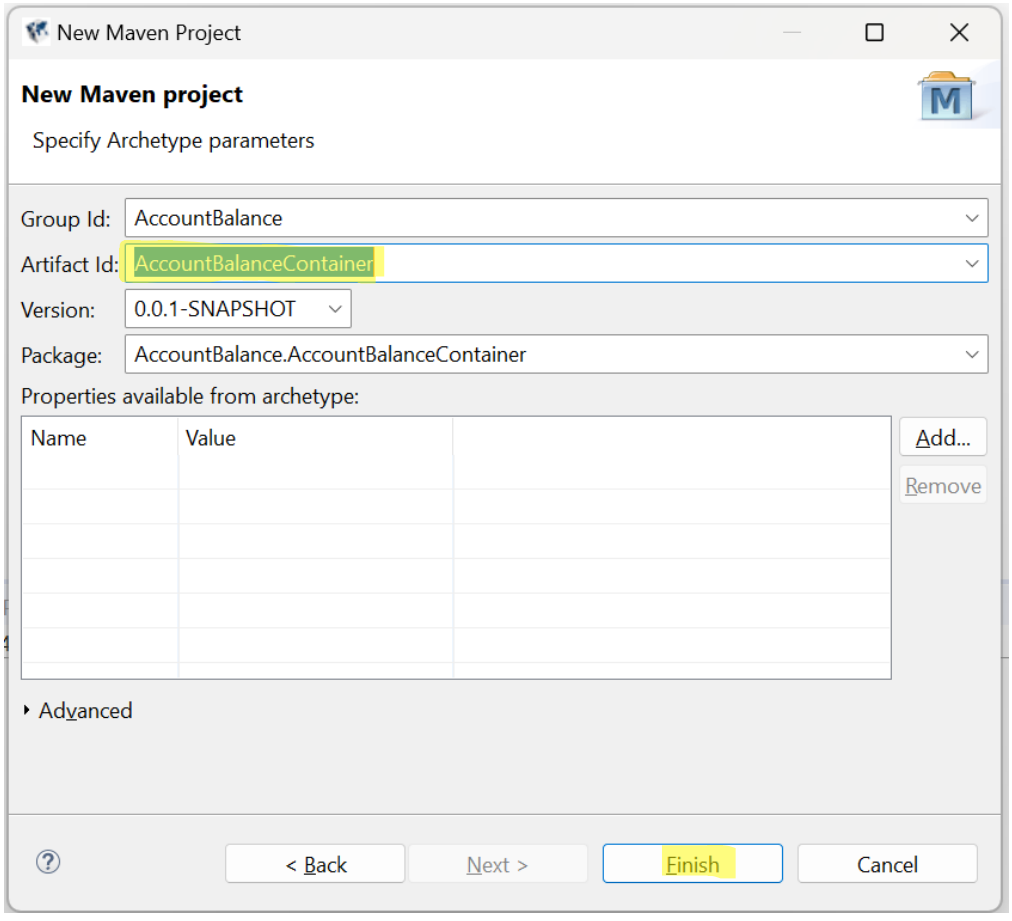


There must be two **Maven** Projects (irf-service-archetype & irf-service-container -archetype)



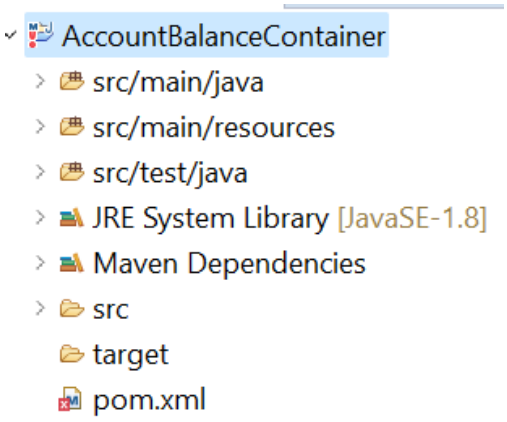


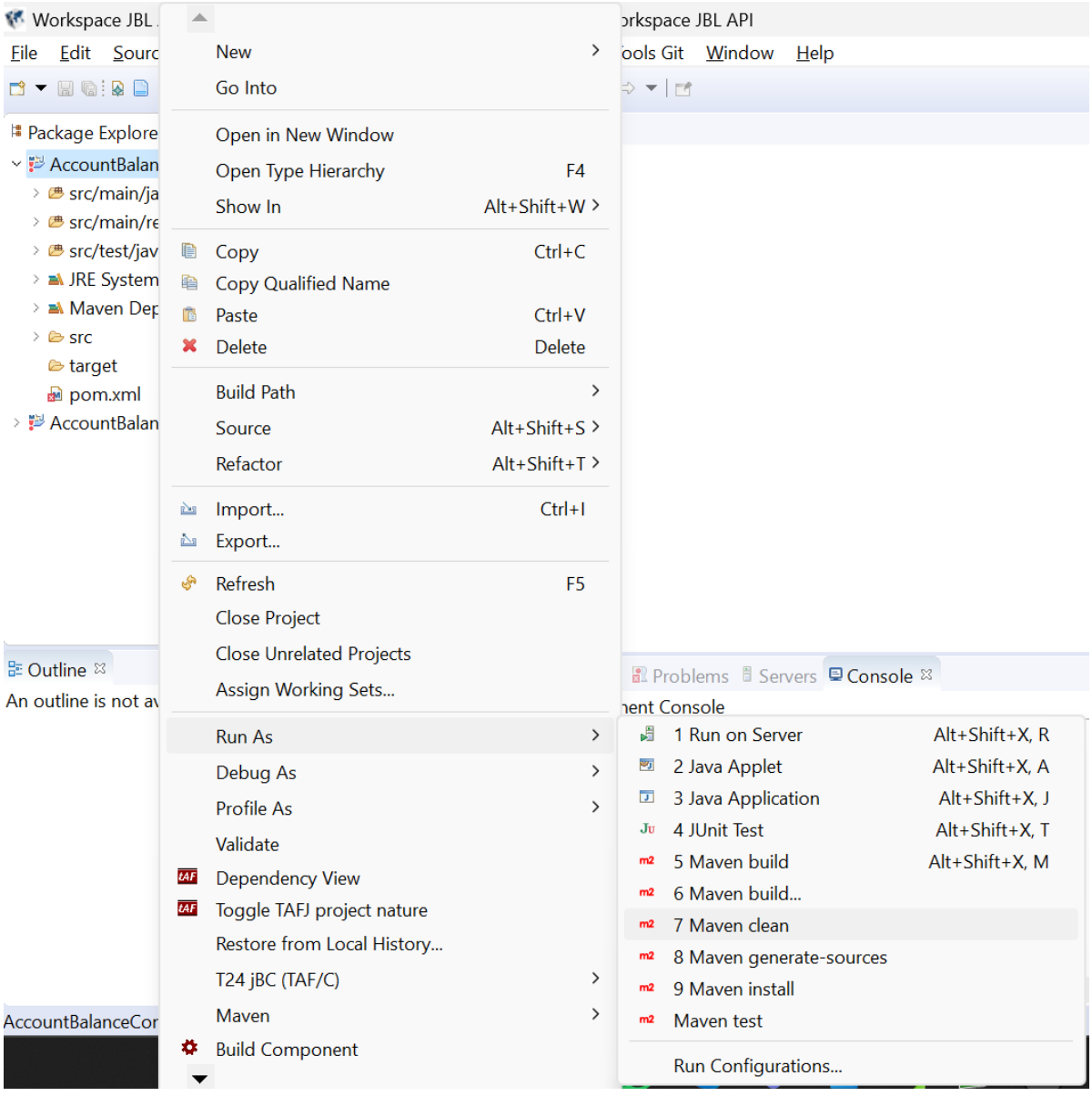

New Maven Project
New Maven project
Select an Archetype
Catalog: All Catalogs
Configure...
Filter:
Group Id
Artifact Id
Version
com.temenos.irf
irf-service-archetype
21.0.38
com.temenos.irf
irf-service-container-archetype
21.0.38
org.apache.maven.archetypes
maven-archetype-archetype
1.0
org.apache.maven.archetypes
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
basic
Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Finish
Cancel


New Maven Project
New Maven project
Specify Archetype parameters
Group Id: AccountBalance
Artifact Id: AccountBalanceContainer
Version: 0.0.1-SNAPSHOT
Package: AccountBalance.AccountBalanceContainer
Properties available from archetype:
Name
Value
Add...
Remove
Advanced
< Back
Next >
Finish
Cancel

Clean the both container and Service project by Selecting,

Right click on Project → Run As → Maven Clean.





```
Properties  Problems  Servers  Console
<terminated> D:\Temenos\DS\jdk\bin\javaw.exe (Jan 29, 2024, 10:53:08 AM)
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building AccountBalanceContainer 21.0.38
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ AccountBalanceContainer ---
[INFO] Deleting D:\Temenos\DS\Workspace JBL API\AccountBalanceContainer\target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.278 s
[INFO] Finished at: 2024-01-29T10:53:09+06:00
[INFO] Final Memory: 15M/487M
[INFO] ------------------------------------------------------------------------
```

```
Properties  Problems  Servers  Console
<terminated> D:\Temenos\DS\jdk\bin\javaw.exe (Jan 29, 2024, 10:54:06 AM)
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building AccountBalanceDetails 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.0.0:clean (default-clean) @ AccountBalanceDetails ---
[INFO] Deleting D:\Temenos\DS\Workspace JBL API\AccountBalanceDetails\target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.353 s
[INFO] Finished at: 2024-01-29T10:54:07+06:00
[INFO] Final Memory: 16M/487M
[INFO] ------------------------------------------------------------------------
```

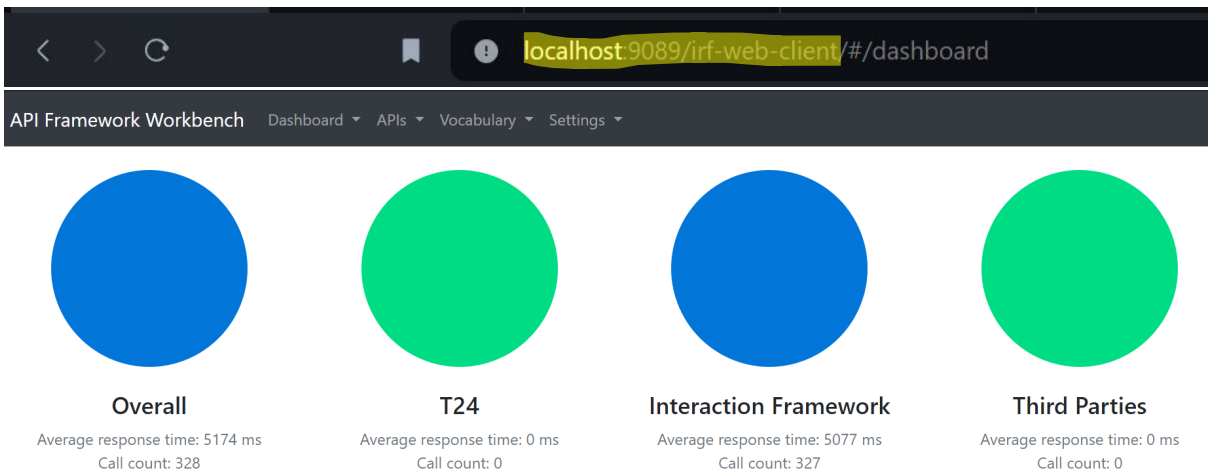
IRIS 2.0 war File Generation

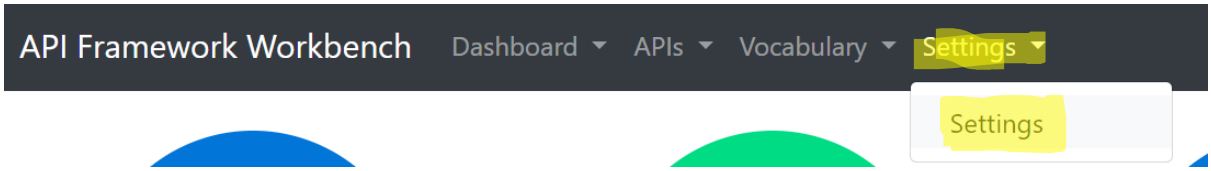
Deploy the 2 war files which is given by temenos in your JBoss

irf-web-client - Workbench

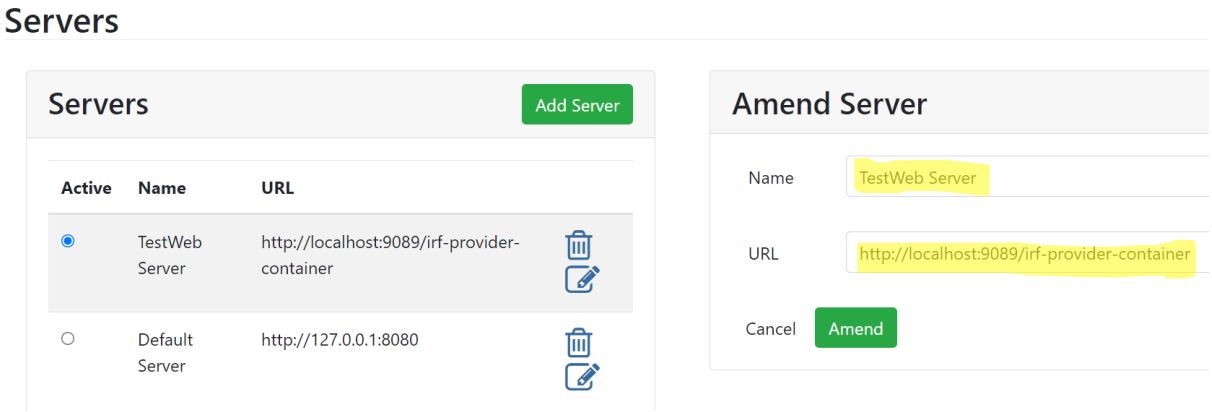
irf-test-web - To test the IRIS Services

Give the base URL in the browser-> Workbench gets opened…





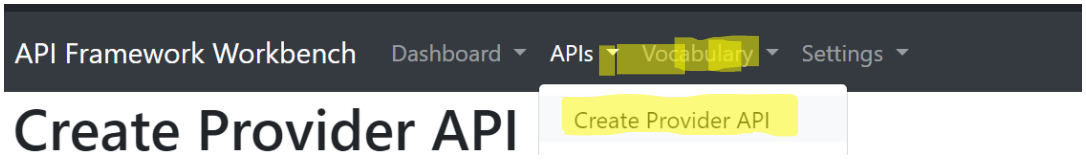
Select settings tab → click settings → Then the servers page gets opened.



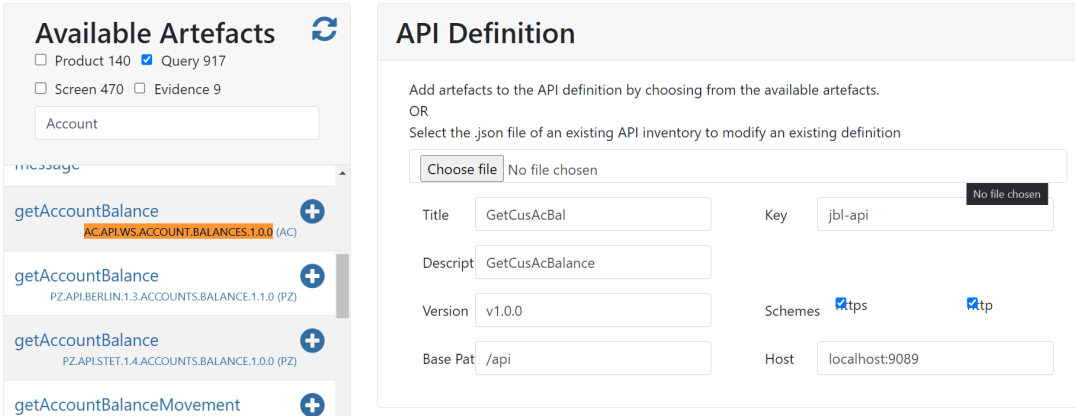
API Creation from Workbench

API Creation:
First We should create a Version or Enquiry in below format.
Product.API.verb.version -----EX: ST.API.CUSTOMERS.1.0.0

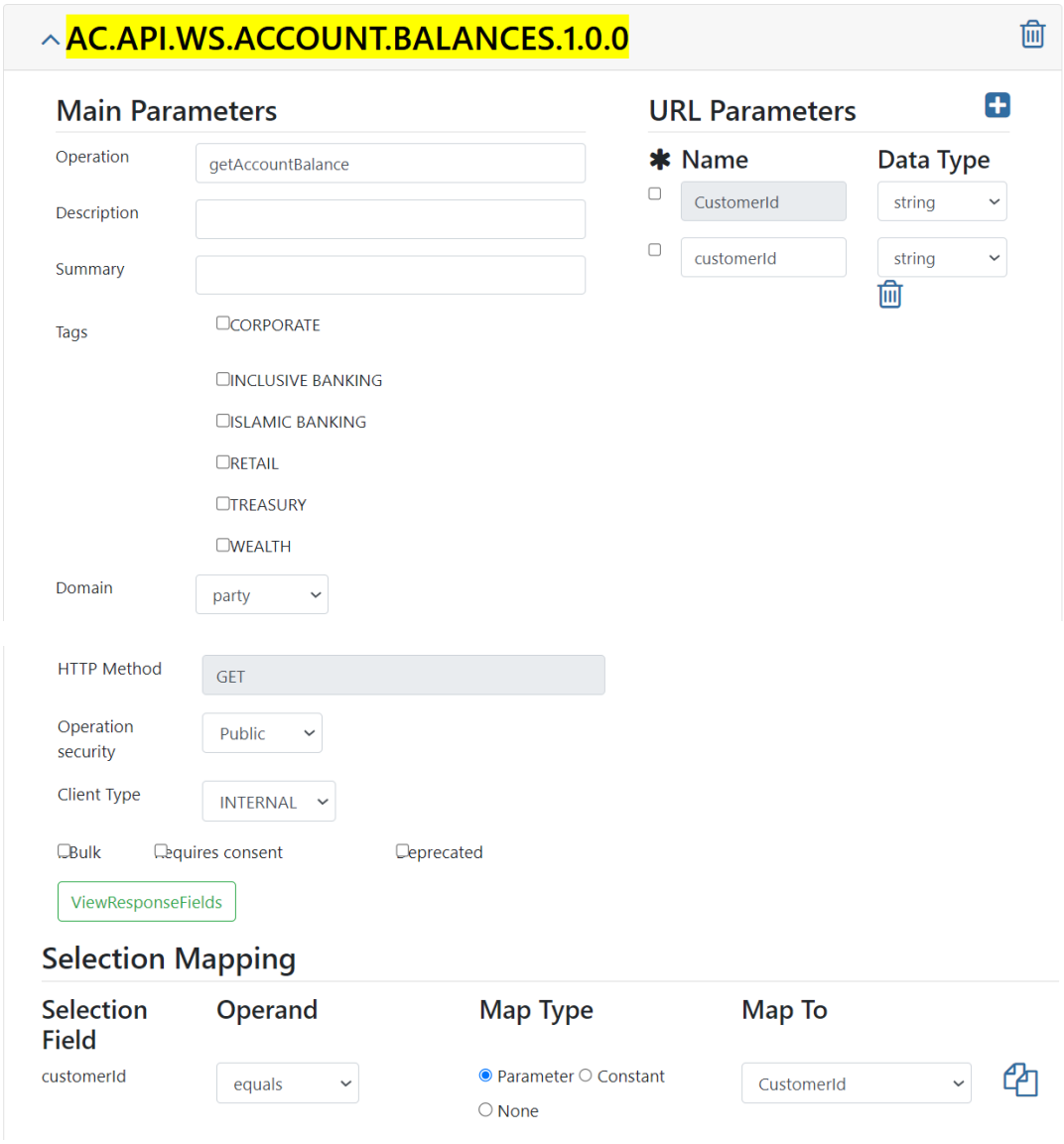
go to localhost:9089/irf-web-client
Click API>create provider API.



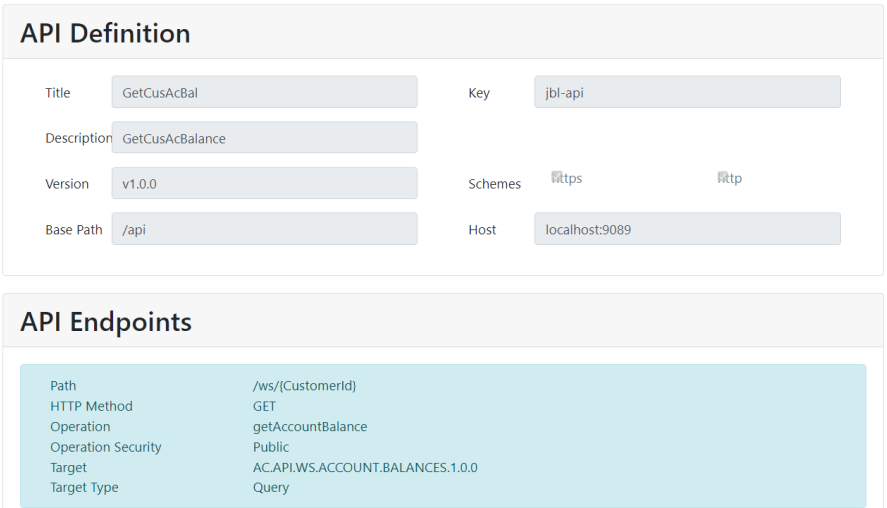
The Available Artefacts can be displayed. The Artefacts can be Versions(Screen), Enquiries(Query),AA Product(Product) and so on.
Select the data's which you want to import.
The API definition page will be displayed.
Select the required artefacts from available artefacts.

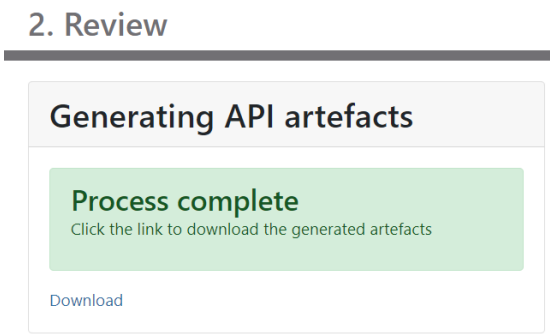


Provide key , url, summary, description. (**give request and response payload versions If it is AA API**)



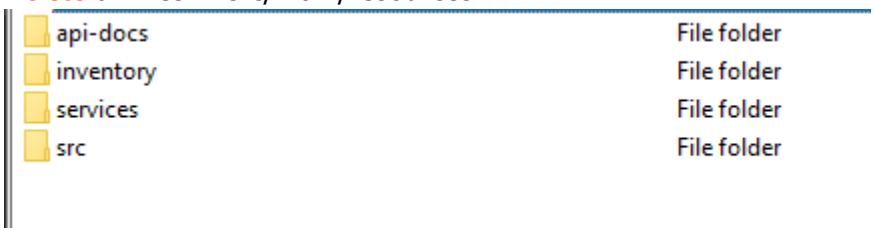
click **NEXT** on top of the screen>**FINISH**



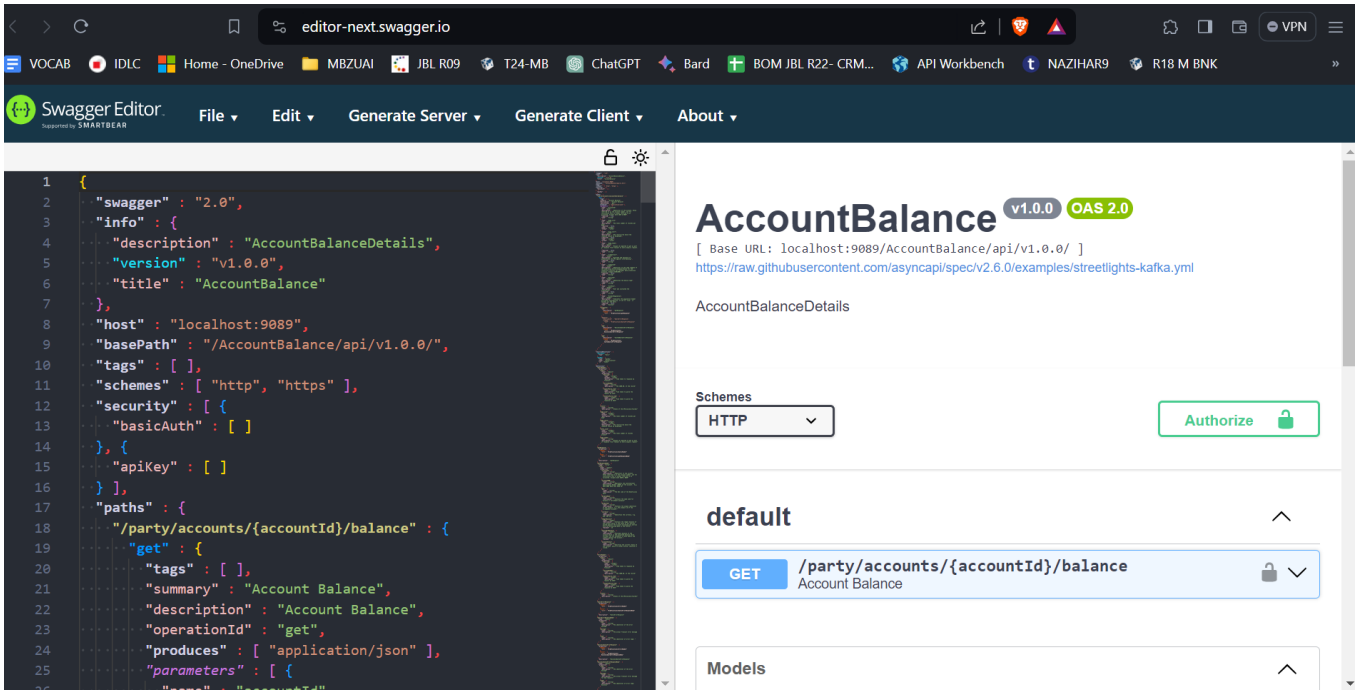


IRIS Workbench will generate a zip based on selected transact artefacts. Download it. That zip contains 3 folders. **i.e. api-docs, inventory, services**

After creating a **Maven** project (**AccountBalanceDetails**),
Delete all files in src/main/resources

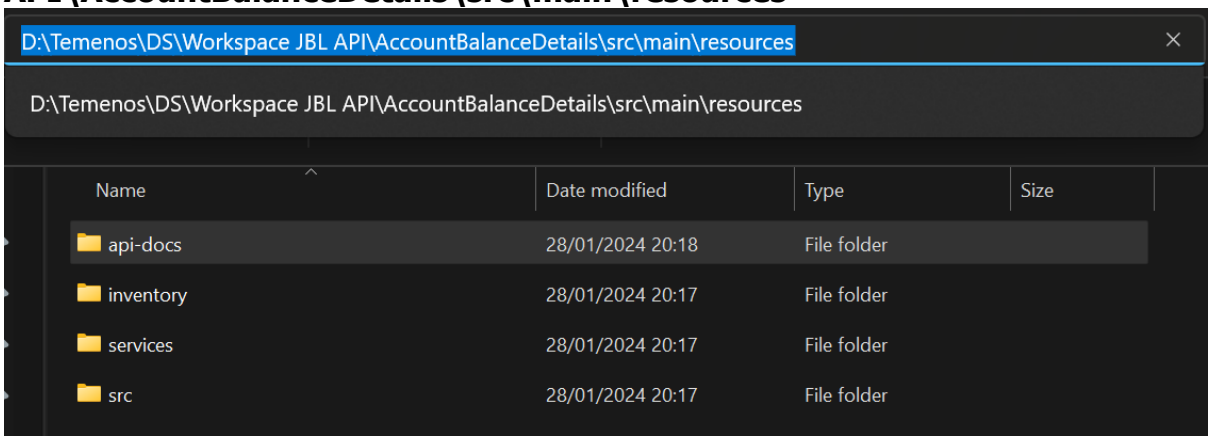


api-docs – This folder contains swagger which will show the specification of our API
Ex:

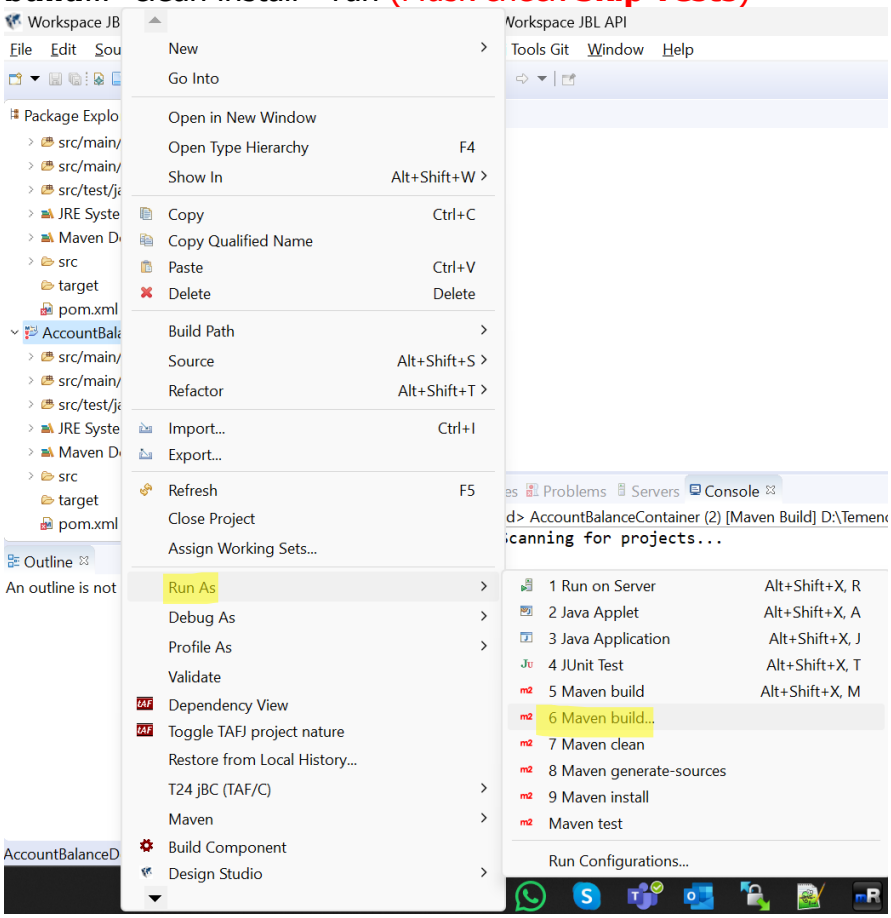


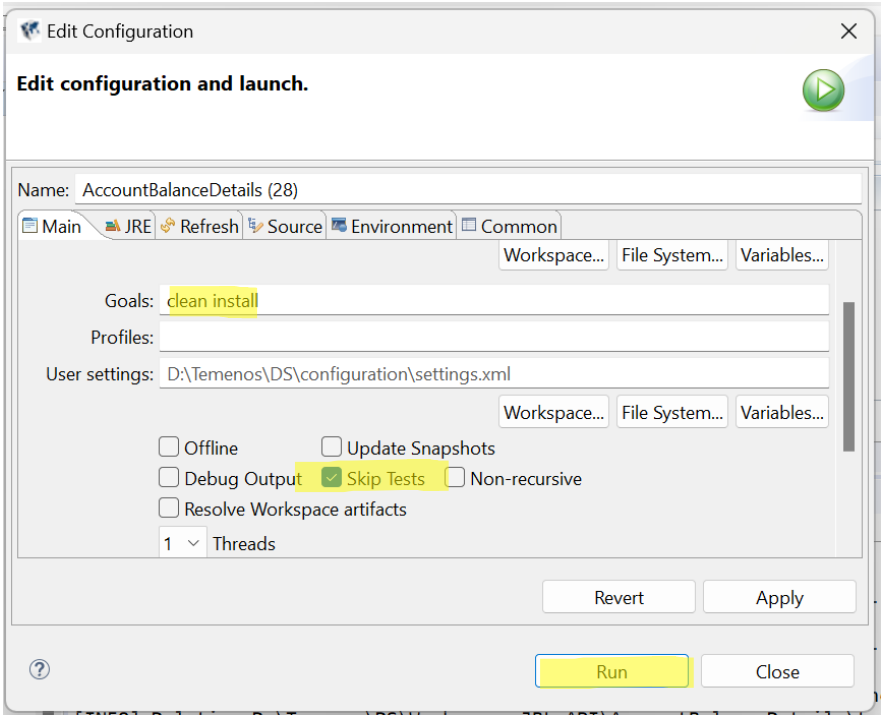
Inventory – This folder contains details of the generated artefacts/APIs. Which is useful to load existing APIs to the workbench rather than selecting artefacts from scratch.
Services- This folder contains the xml routes for routing the API to appropriate transact artefact.
This file alone is fine to deploy in a container to get our API work.

Paste the **Downloaded** files from the **Workbench** in the Below path, **D:\Temenos\DS\Workspace JBL API\AccountBalanceDetails\src\main\resources**



Right click on the main project(`AccountBalanceDetails`)>Run as> **Maven build**...>clean install> run (Musk check **Skip Tests**)

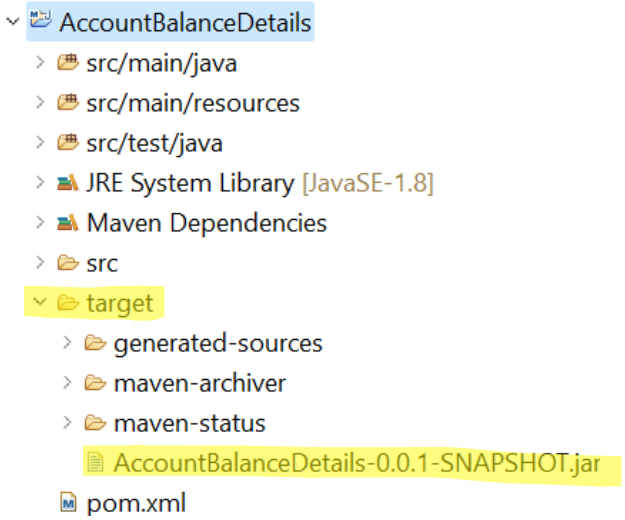




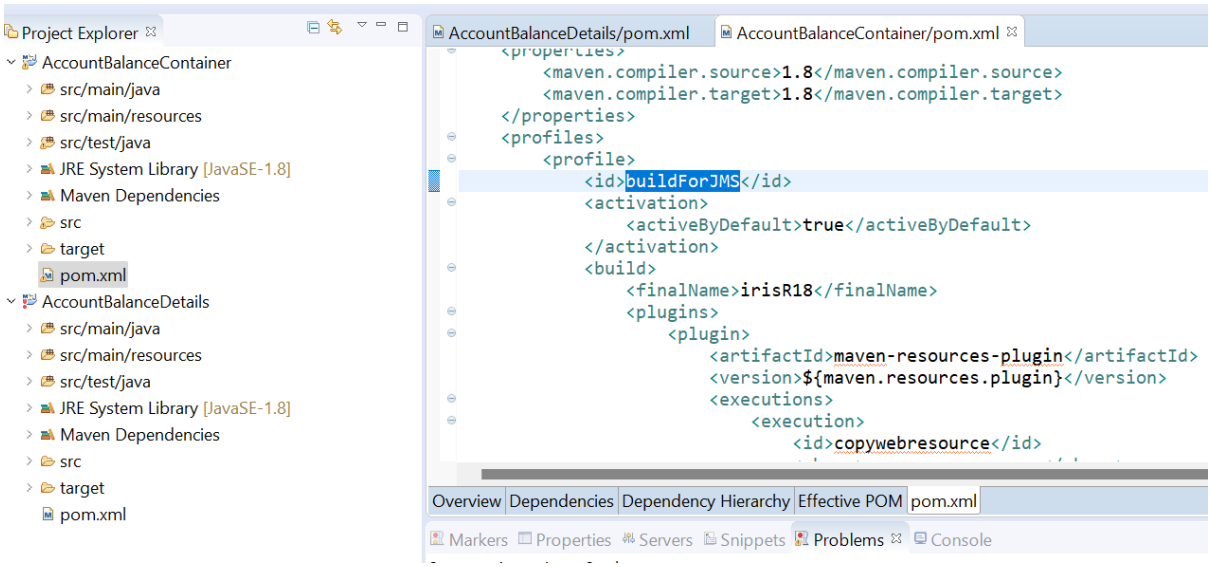
If the BUILD is FAILED then need to download the .jar json-20170516.jar file and paste it inside the **t24 binaries** directory like this,
D:\Temenos\DS\t24-binaries\org\json\json\20170516

We also need to replace each http to https in **archetype-catalog.xml** archetype-catalog.xml file.

If the **BUILD** is **SUCCESSFUL** then the **jar** file will be created here.

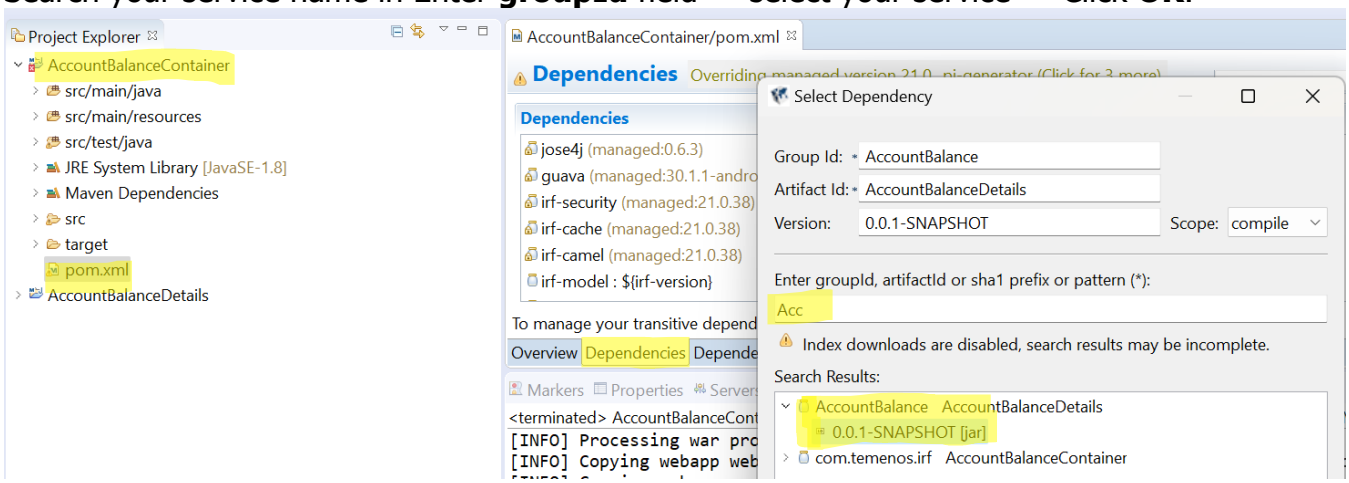


Go to the **container** project >pom.xml
Go to **dependencies** tab and add your jars (created in main project)
Go to **pom.xml** tab
buildforJMS > true ; Not Required/Optional for Model Bank.

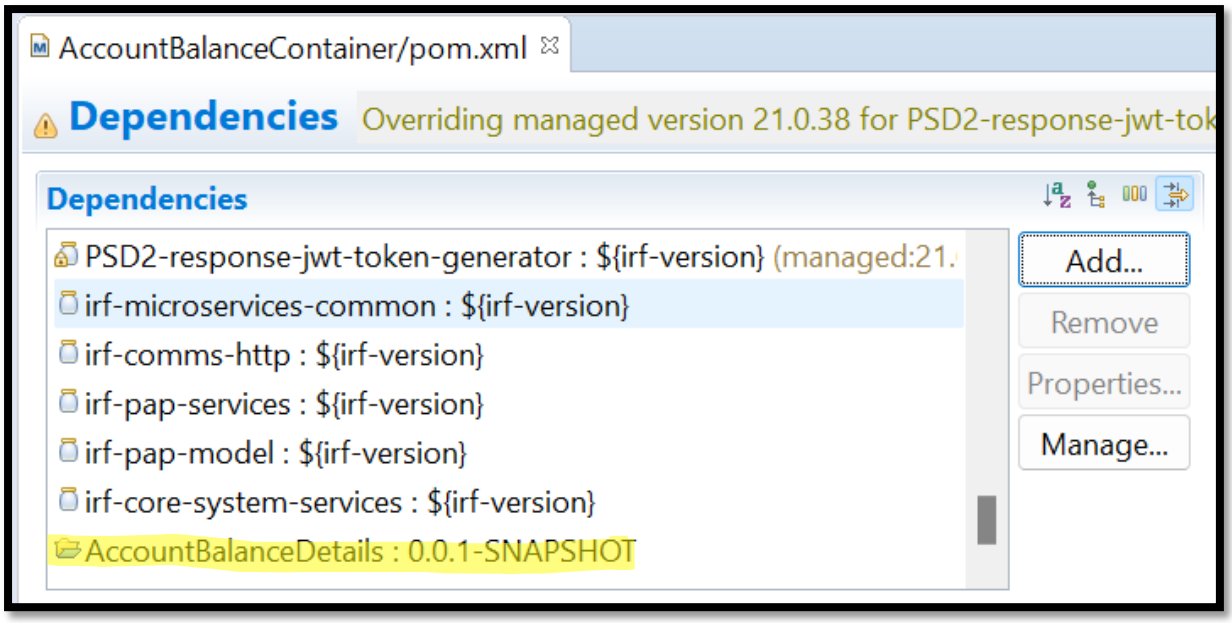


Adding Dependencies:

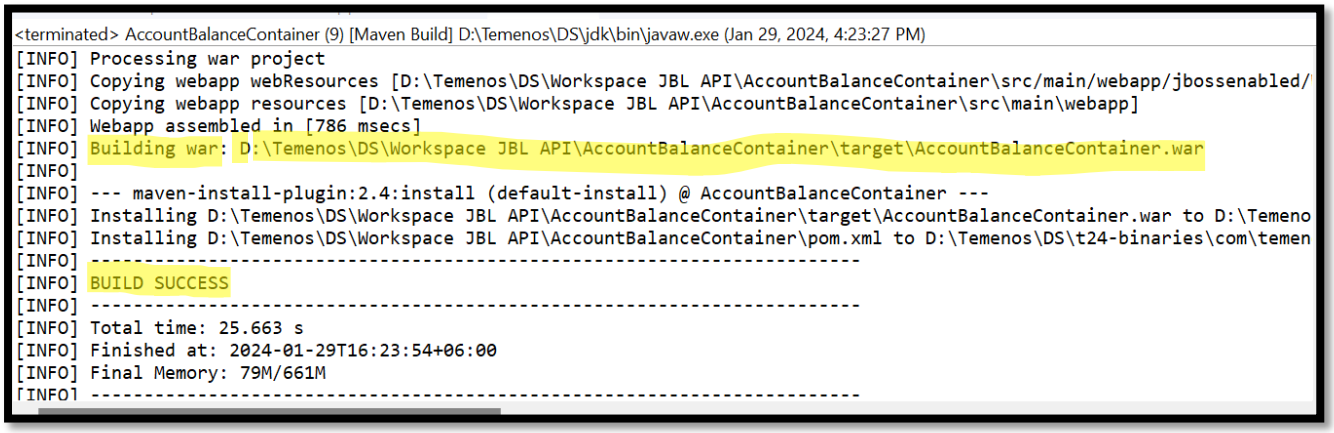
Special Note: We must build the main project first then the container project.
Double click on pom.xml → select Dependencies tab → click Add.
The **container** will have the main project **jar file** as a Dependency.
Search your service name in Enter **groupId** field → select your service → Click **OK**.



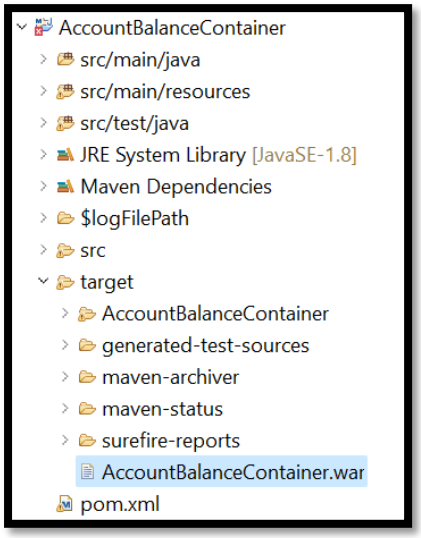
Then **Save** it. → Ctrl +S



If the jboss is a remote server then,
Provide the jboss environment: username and password.
Java.naming.security.principal = username
Java.naming.security.credentials = Password

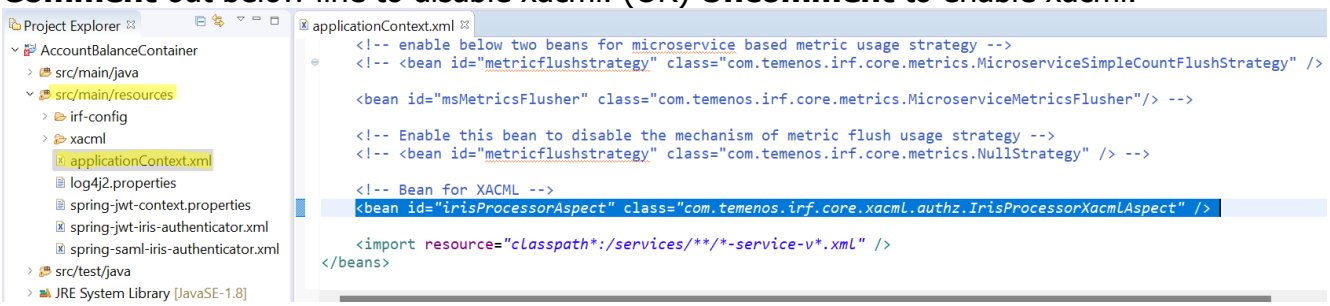


D:\Temenos\DS\Workspace JBL API\AccountBalanceContainer\target



To Enable/disable xacml:

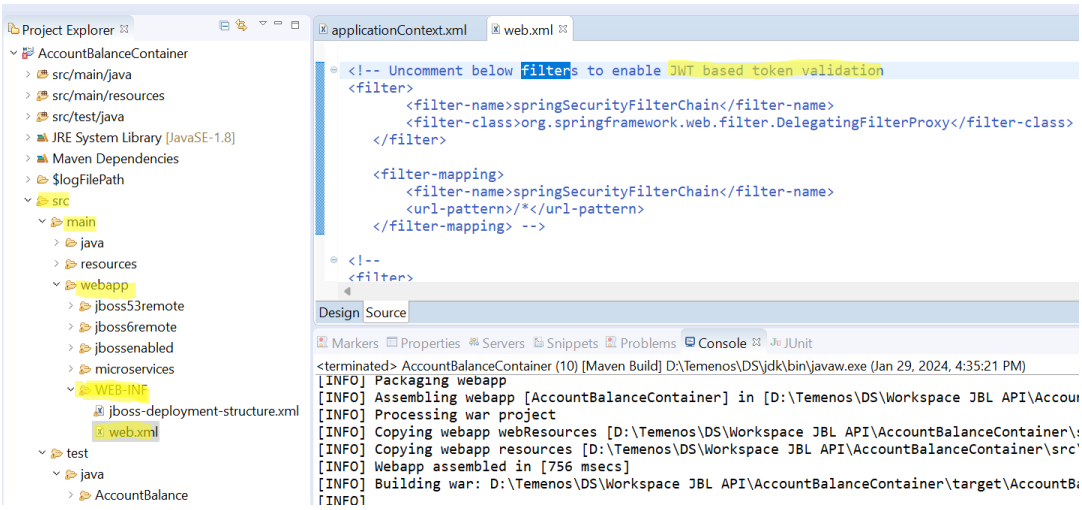
Comment out below line to disable xacml. (OR) **Uncomment** to enable xacml.



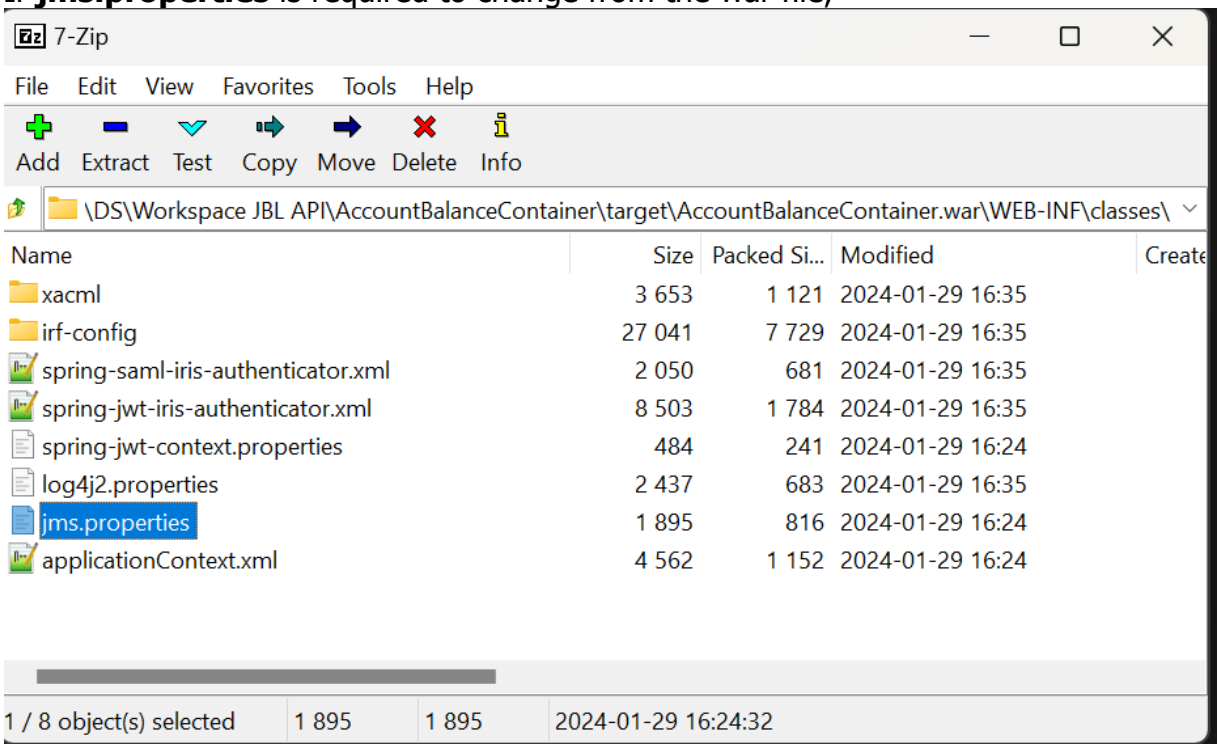
To Enable/disable JWT:

Comment out below set of lines to disable JWT (OR)

UnComment below set of lines to enable JWT.

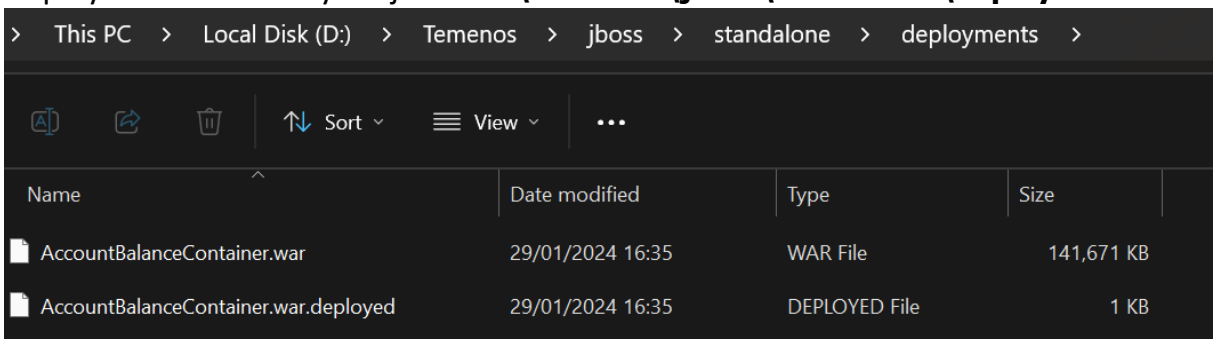


If **jms.properties** is required to change from the war file,



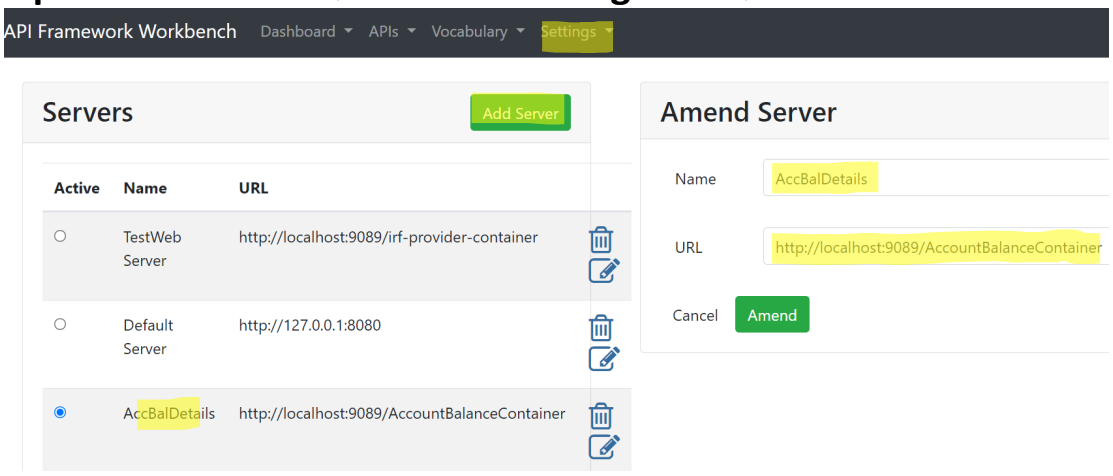
Deployment :

Deploy the .war file in your jboss. **D:\Temenos\jboss\standalone\deployments**

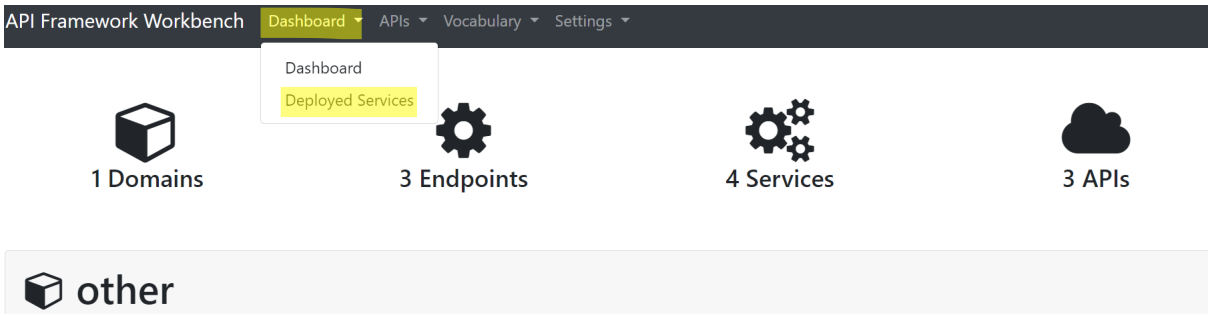


Adding server to view the API :

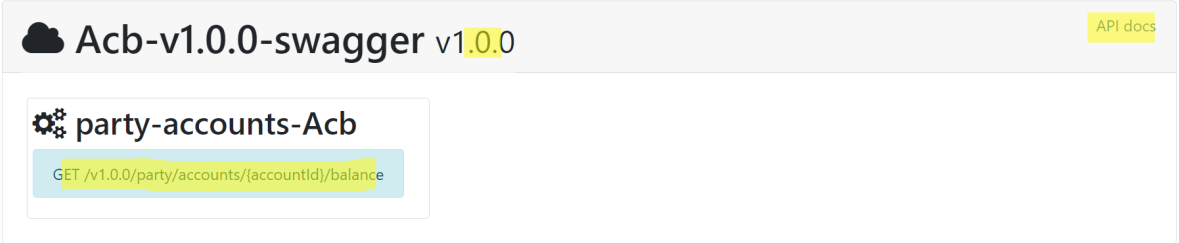
Open workbench → click on Settings tab → click Add Server button.

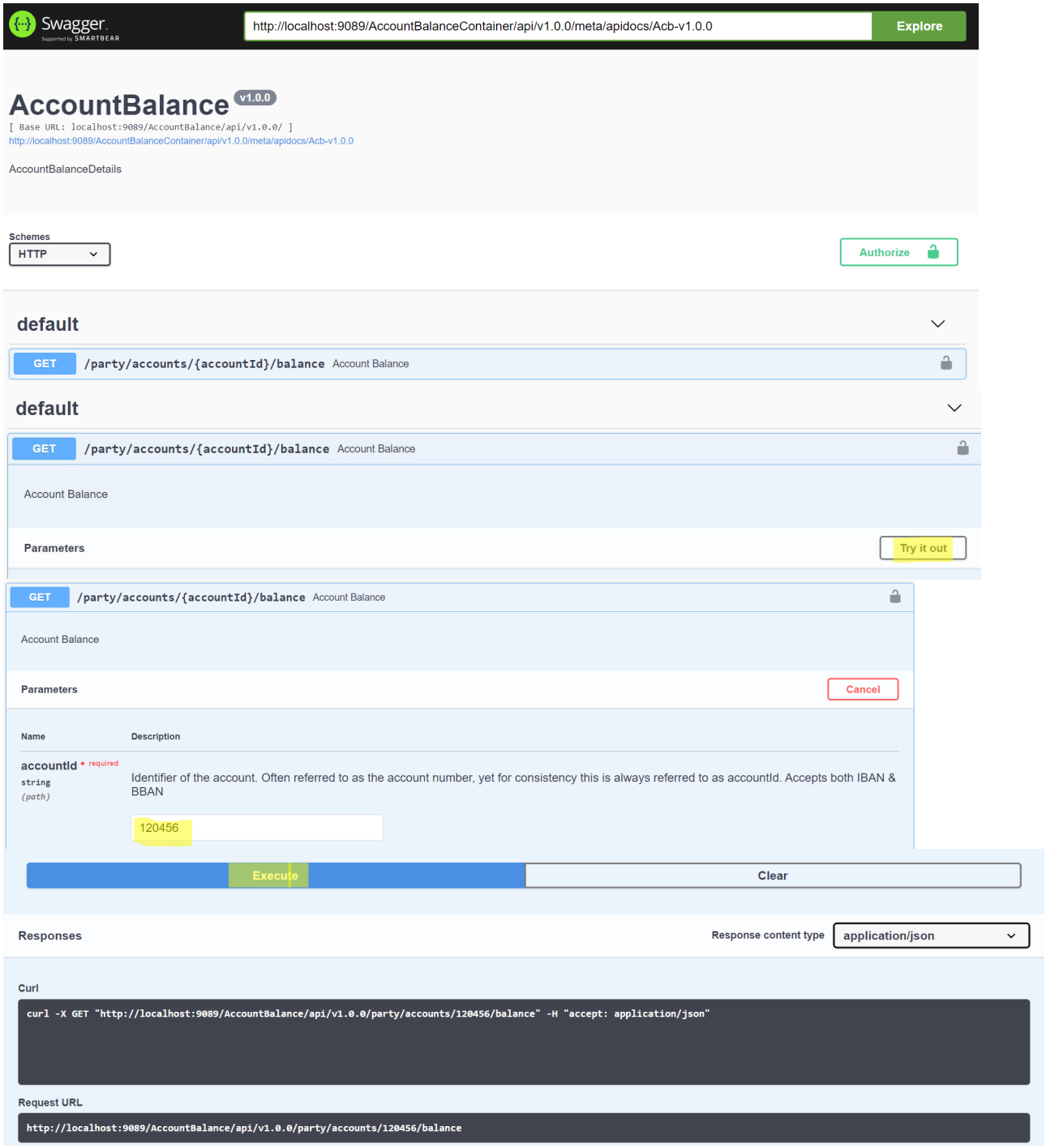


Checking the Deployed Services

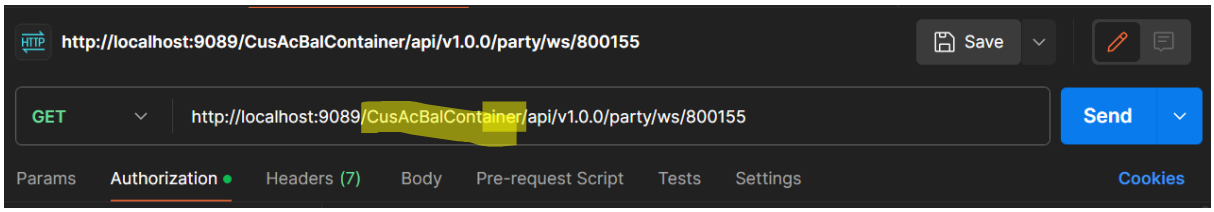


CLICK - API docs





Checking API in Postman:
Run postman → click '+' button → paste your API link and add War filename in the link.
Click Send → Response will be shown.



If authentication(Basic Auth) error is found then **applicationContext.xml file** need to be changed.

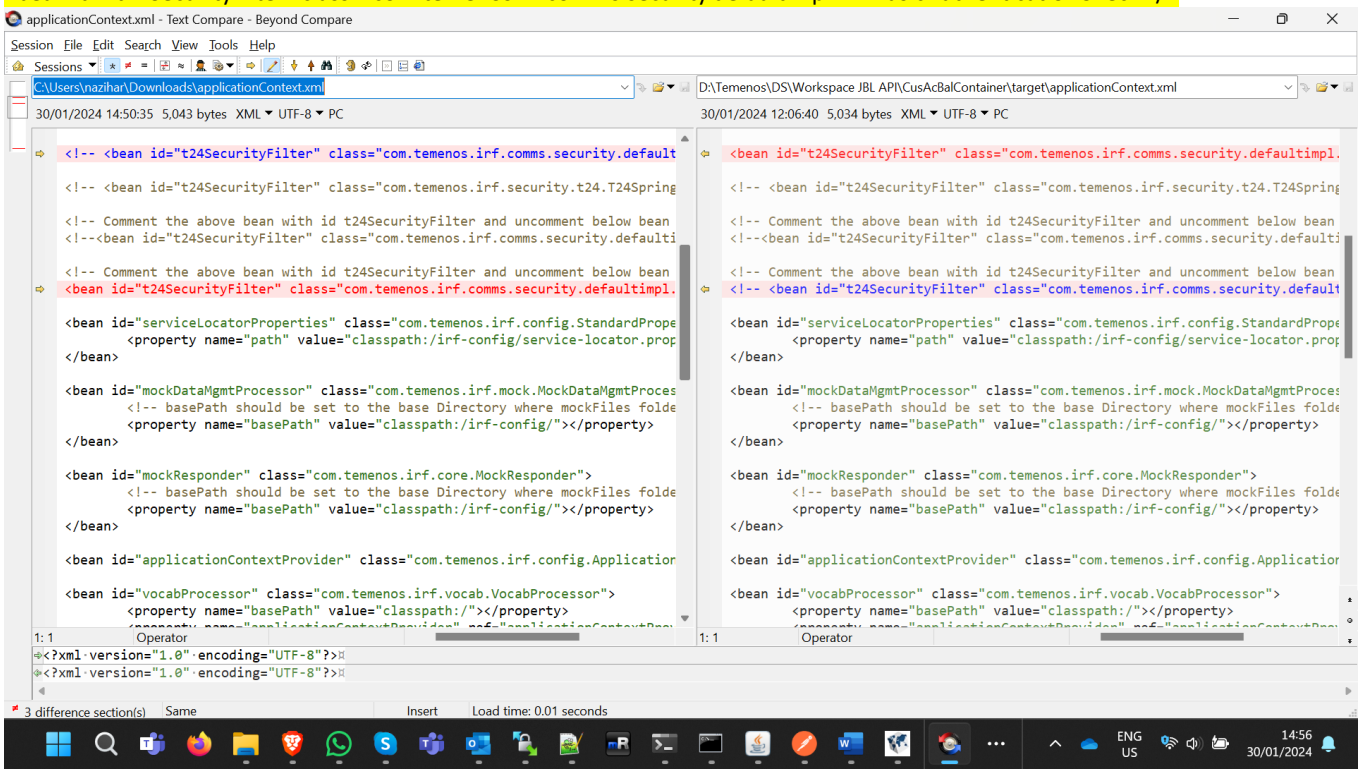
Commend out these lines:

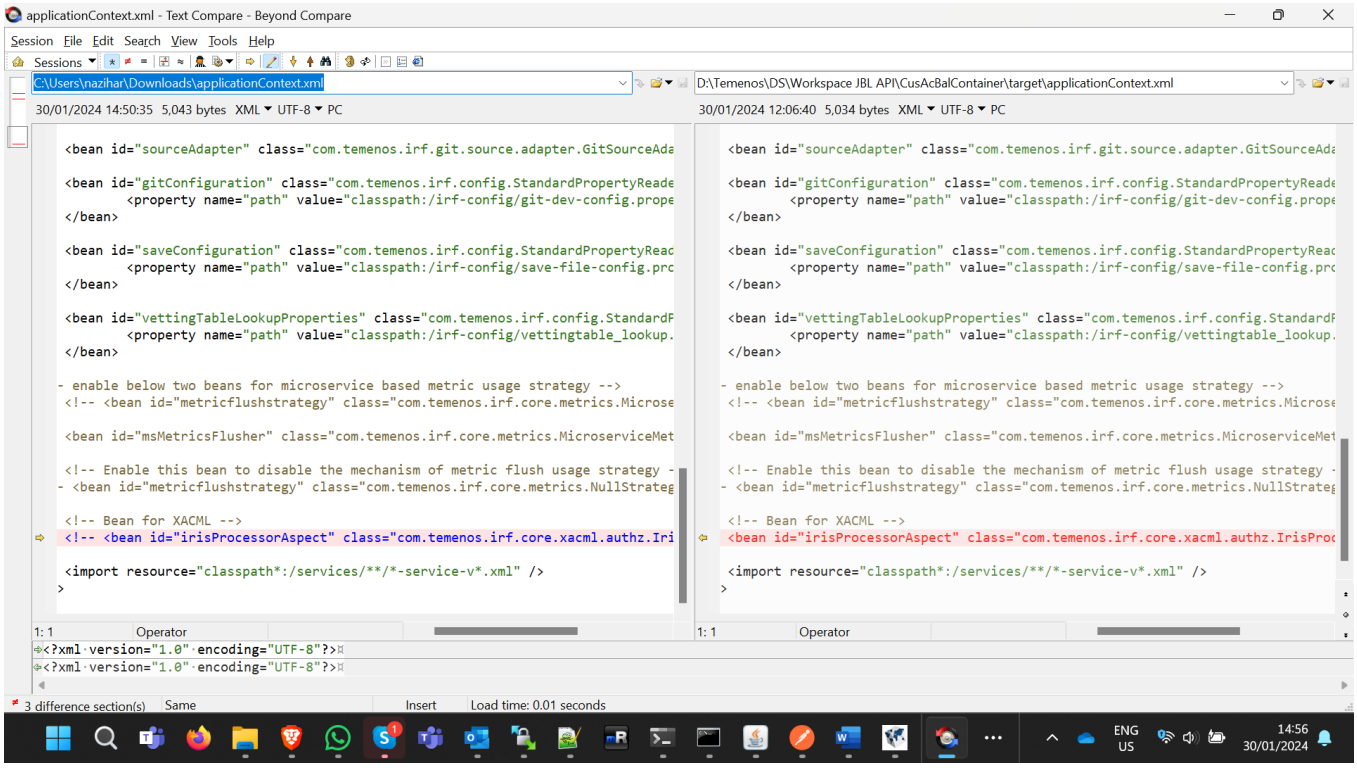
<!-- <bean id="t24SecurityFilter" class="com.temenos.irf.comms.security.defaultimpl.NullBean" /> -->

<!-- <bean id="irisProcessorAspect" class="com.temenos.irf.core.xacml.authz.IrisProcessorXacmlAspect" /> -->

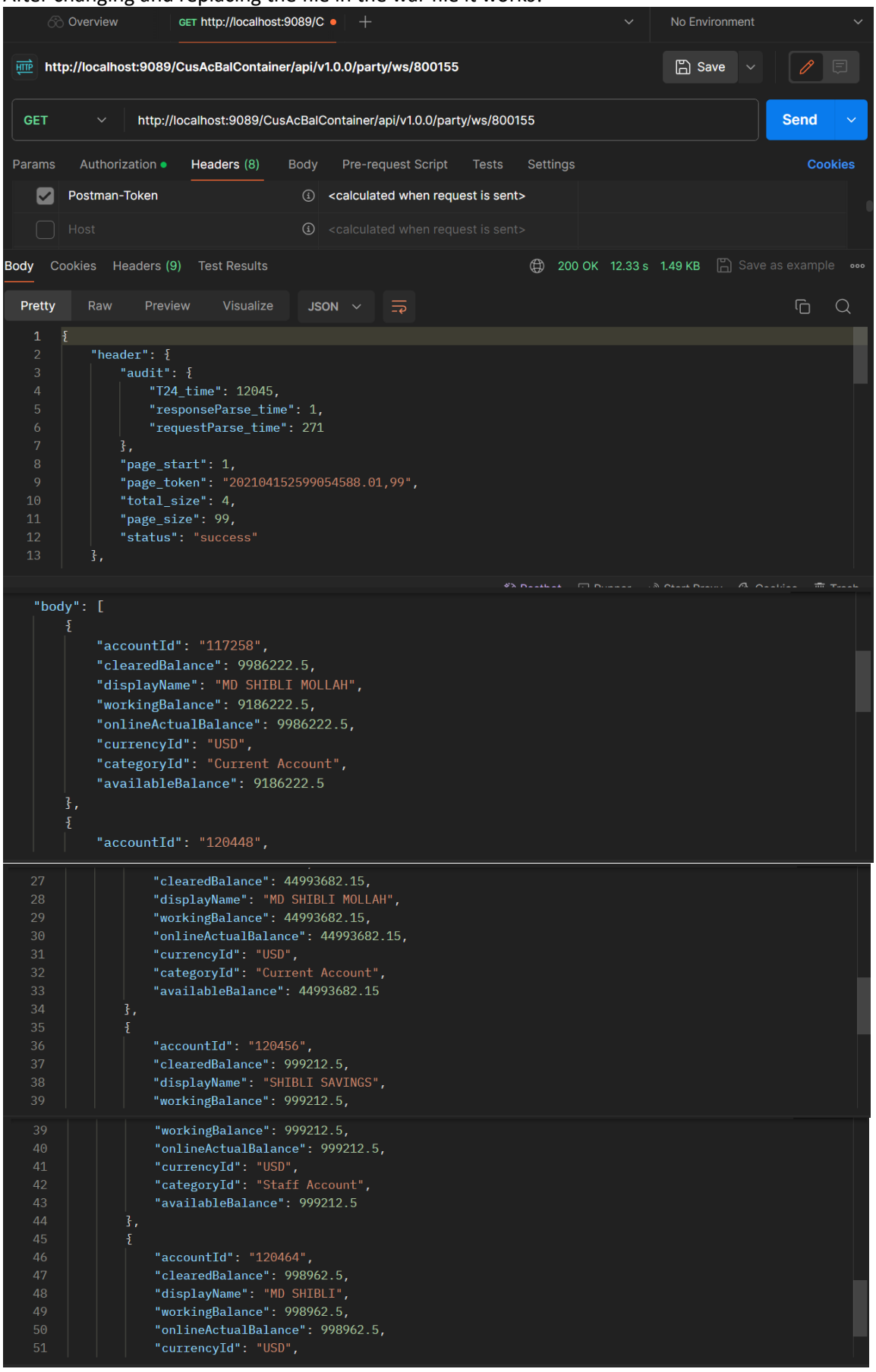
Uncomment:

<bean id="t24SecurityFilter" class="com.temenos.irf.comms.security.defaultimpl.T24BasicAuthenticationCheck" />





After changing and replacing the file in the war file it works!



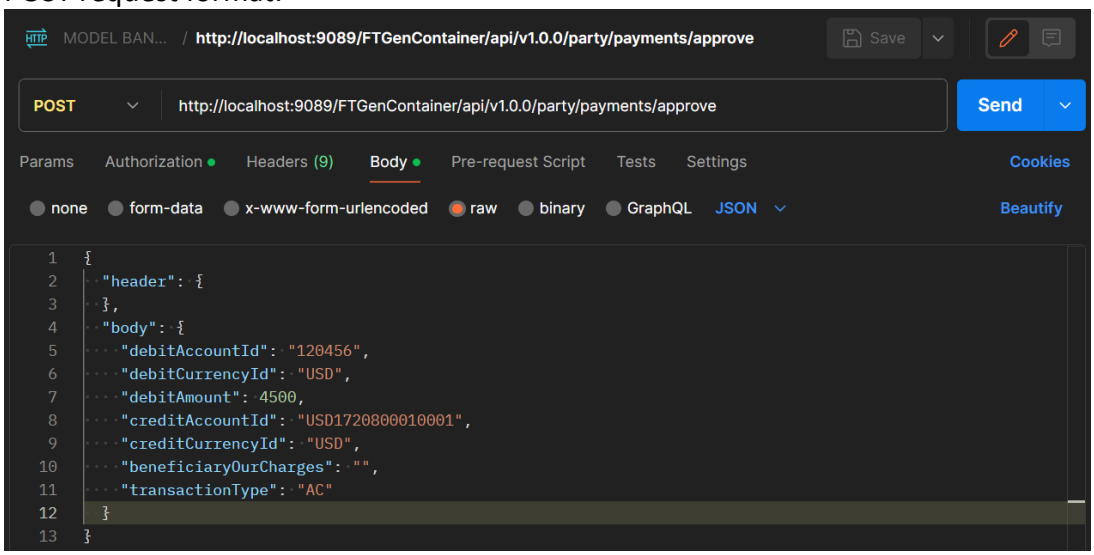
Response:


```
{
  "header": {
    "audit": {
      "T24_time": 12045,
      "responseParse_time": 1,
      "requestParse_time": 271
    },
    "page_start": 1,
    "page_token": "202104152599054588.01,99",
    "total_size": 4,
    "page_size": 99,
    "status": "success"
  },
  "body": [
    {
      "accountId": "117258",
      "clearedBalance": 9986222.5,
      "displayName": "MD SHIBLI MOLLAH",
      "workingBalance": 9186222.5,
      "onlineActualBalance": 9986222.5,
      "currencyId": "USD",
      "categoryId": "Current Account",
      "availableBalance": 9186222.5
    },
    {
      "accountId": "120448",
      "clearedBalance": 44993682.15,
      "displayName": "MD SHIBLI MOLLAH",
      "workingBalance": 44993682.15,
      "onlineActualBalance": 44993682.15,
      "currencyId": "USD",
      "categoryId": "Current Account",
      "availableBalance": 44993682.15
    },
    {
      "accountId": "120456",
      "clearedBalance": 999212.5,
      "displayName": "SHIBLI SAVINGS",
      "workingBalance": 999212.5,
      "onlineActualBalance": 999212.5,
      "currencyId": "USD",
      "categoryId": "Staff Account",
      "availableBalance": 999212.5
    },
    {
      "accountId": "120464",
      "clearedBalance": 998962.5,
      "displayName": "MD SHIBLI",
      "workingBalance": 998962.5,
      "onlineActualBalance": 998962.5,
      "currencyId": "USD",
      "categoryId": "Savings Acct",
      "availableBalance": 998962.5
    }
  ]
}
```

VERSION - POST

Api creation and war file creation is almost same as the process for ENQUIRY.

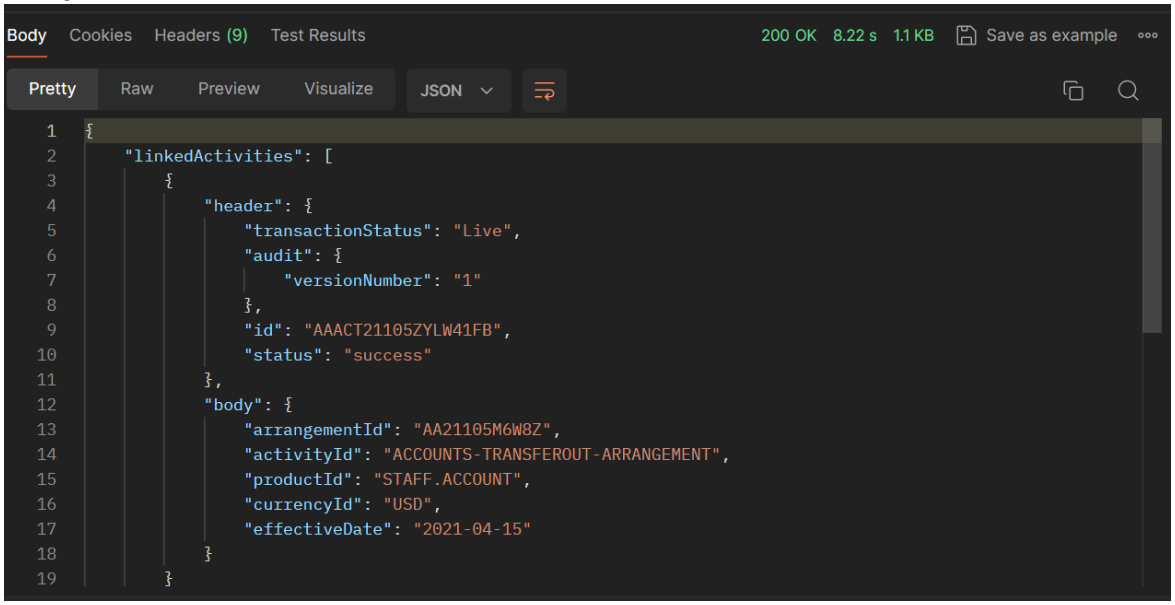
POST request format:



Body:

```
{
  "header": {
  },
  "body": {
    "debitAccountId": "120456",
    "debitCurrencyId": "USD",
    "debitAmount": 4500,
    "creditAccountId": "USD1720800010001",
    "creditCurrencyId": "USD",
    "beneficiaryOurCharges": "",
    "transactionType": "AC"
  }
}
```

Response:



```
],
"header": {
    "transactionStatus": "Live",
    "audit": {
        "T24_time": 8209,
        "responseParse_time": 4,
        "requestParse_time": 2,
        "versionNumber": "1"
    },
    "id": "FT21105RT0MY",
    "status": "success"
},
"body": {
    "transactionType": "AC",
    "debitCurrencyId": "USD",
    "processingDate": "2021-04-15",
    "creditCurrencyId": "USD",
      "debitAccountId": "120456",
      "debitAmount": 4500,
      "creditAccountId": "USD1720800010001"
  }
}
```


Orchestration

Question: Creating orchestration for FT and Account

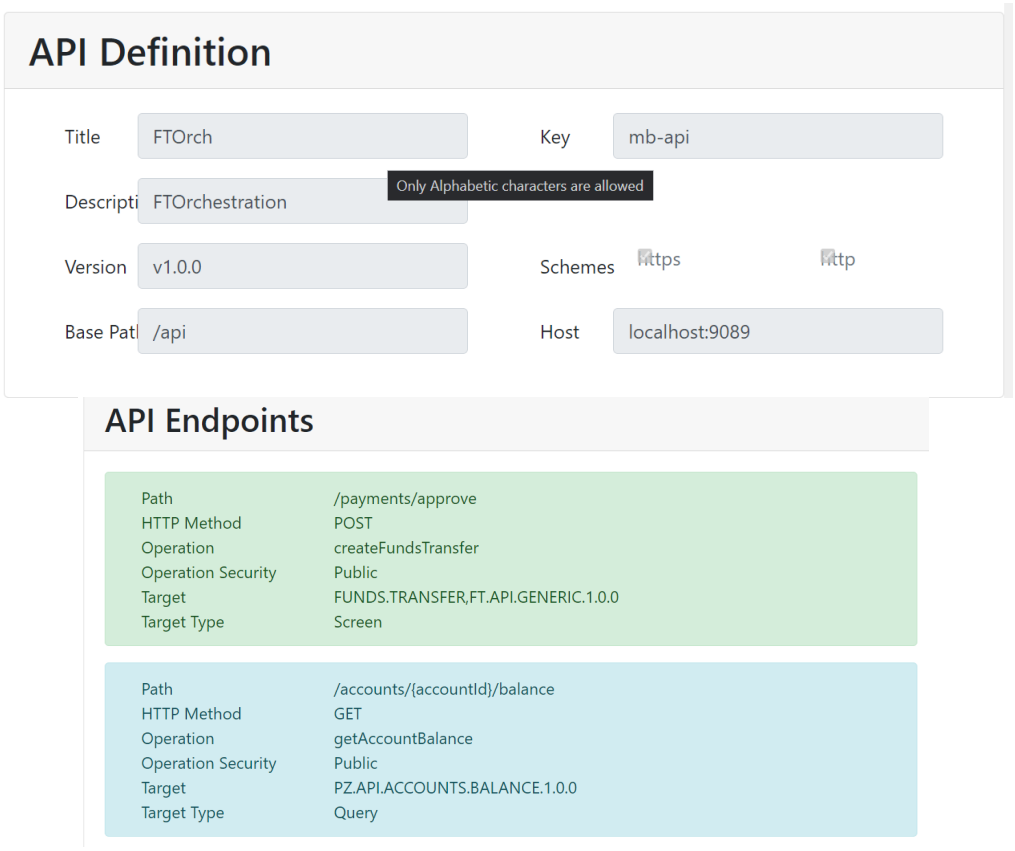
Requirement : After Authorization of FT(version), The enquiry which shows the debit Account Id's Available balance

Step 1:

Choose the artefacts in the workbench and generate the zip file.

Enquiry : FUNDS.TRANSFER,FT.API.GENERIC.1.0.0

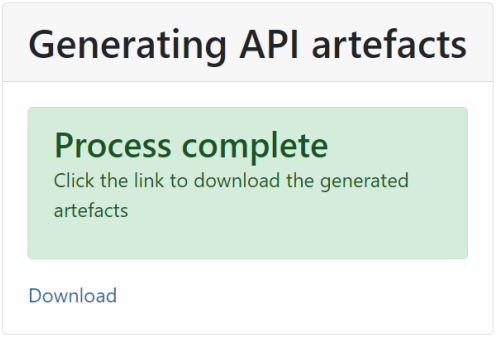
Version : PZ.API.ACCOUNTS.BALANCE.1.0.0



Inventory

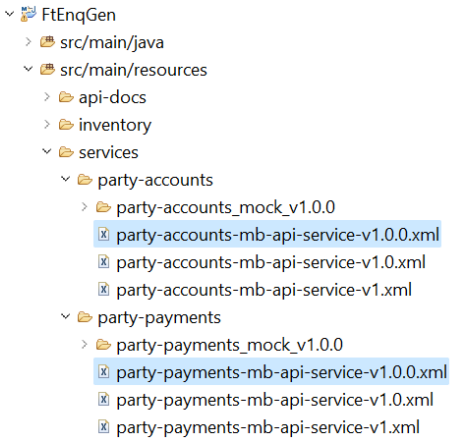

```
{
 "paths": [
   {
     "method": "POST",
     "url": "/party/payments/approve",
     "function": "input",
     "tags": [],
     "operationId": "createFundsTransfer",
     "operationSecurity": "Public",
     "resources": [
       {
         "key": "FUNDS.TRANSFER,FT.API.GENERIC.1.0.0",
         "resourceType": "Screen",
         "consentManaged": false
       }
     ],
     "properties": {
       "clientType": "INTERNAL",
       "isBulk": false,
       "deprecated": false
     }
   },
   {
     "method": "GET",
     "url": "/party/accounts/{accountId}/balance",
     "tags": [],
     "operationId": "getAccountBalance",
     "operationSecurity": "Public",
     "resources": [
       {
         "key": "PZ.API.ACCOUNTS.BALANCE.1.0.0",
         "resourceType": "Query",
         "consentManaged": false,
         "selections": [
           {
             "field": "ACCOUNTREFERENCE",
             "param": "accountId",
             "operand": "EQ",
             "required": "",
             "type": "string"
           }
         ]
       }
     ],
     "properties": {
       "clientType": "INTERNAL",
       "isBulk": false,
       "deprecated": false
     }
   }
 ],
 "version": "v1.0.0",
 "title": "FTOrch",
```

```
  "description": "FTOrchestration",
  "key": "mb-api",
  "schemes": [
    "http",
    "https"
  ],
  "basepath": "/api",
  "host": "localhost:9089"
}
```



Step 2:
Extract the zip file and place it in the new service project.
src -> main -> resources

Step 3:
Open the service xml file in the project.



Step 4:
Open the **xml file** and write the below code under the **t24 VERSION** processor line in the xml file.

```
<choice>
        <when>
            <simple>${headers.CamelHttpResponseCode} == 200</simple>
            <setHeader headerName = "debitAccountId" >
                <camel:jsonpath>$.body.debitAccountId</camel:jsonpath>
            </setHeader>
            <to uri=" direct-vm:party-accounts.v1.0.0.getAccountBalance" />
        </when>
</choice>
```

Above xml file is for **VERSION**. Here I am passing debitAccountId field to the **ENQUIRY** as selection field.

Open the ENQUIRY xml file:
Here we are Mapping the **debitAccountId** to the selection field. (Here we passing **headerName** which is written in the version xml file)

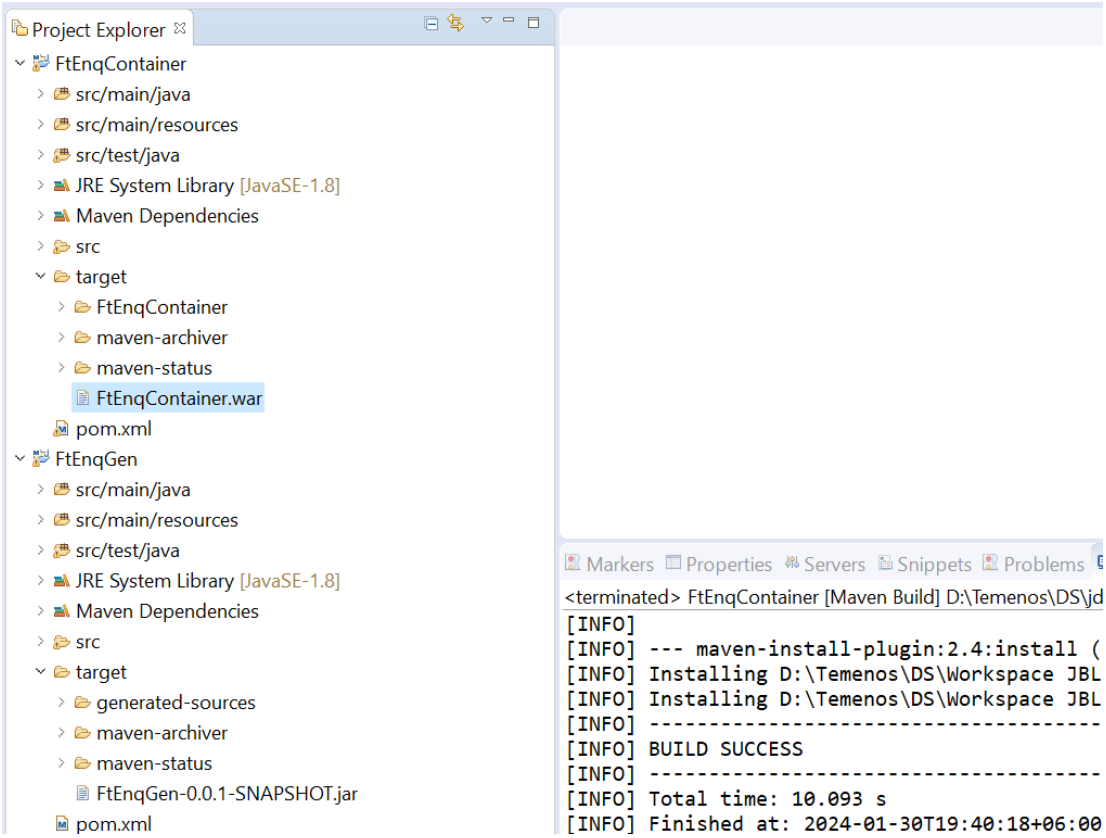
Edit this line: `<constant>ACCOUNTREFERENCE EQ {debitAccountId}</constant>`

party-accounts-mb-api-service-v1.0.0.xml | party-payments-mb-api-service-v1.0.0.xml

```
        <from uri="direct-vm:party-accounts.v1.0.0.getAccountBalance"/>
        <setProperty propertyName="operationSecurity">
            <constant>Public</constant>
        </setProperty>
        <setProperty propertyName="isBulk">
            <constant>false</constant>
        </setProperty>
        <setProperty propertyName="target">
            <constant>PZ.API.ACCOUNTS.BALANCE.1.0.0</constant>
        </setProperty>
        <setProperty propertyName="selection1">
            <constant>ACCOUNTREFERENCE EQ {debitAccountId}</constant>
        </setProperty>
        <process ref="t24EnquiryProcessor"/>
    </route>
    <route id="direct.mockResponder">
        <from uri="direct:mockResponder"/>
        <process ref="mockResponder"/>
    </route>
</camelContext>

</beans>
```

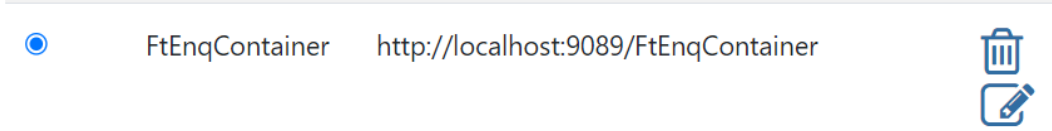
Step 5:
Right click on both project and Container **RunAs**-> Maven clean & Install.
Jar file is created in the **service project** & **war file** is created in the **container project**.

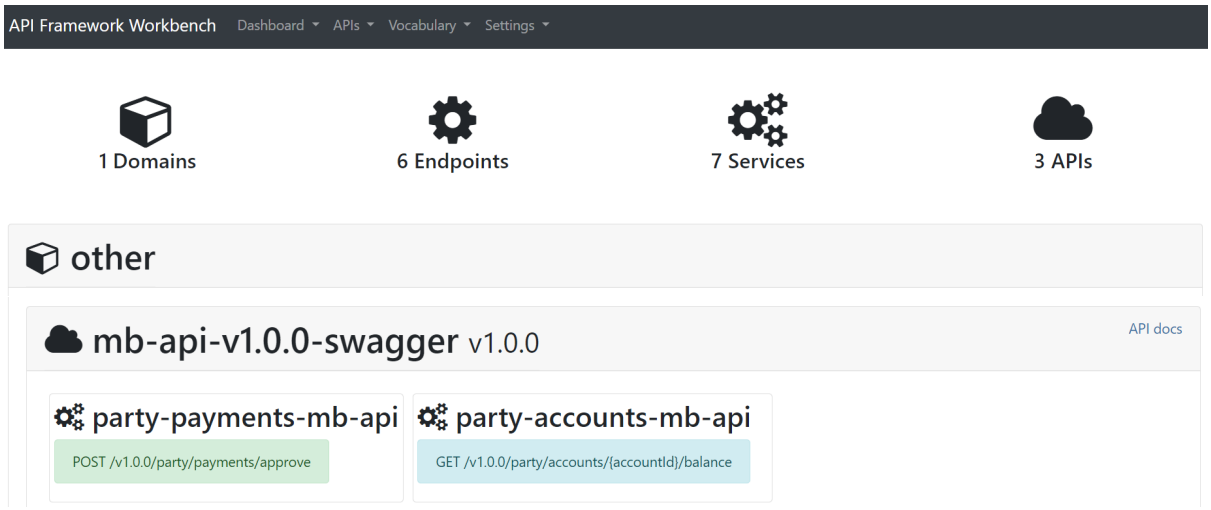


Step 6:

Deploy the war file in JBoss.

Add the server in the workbench.





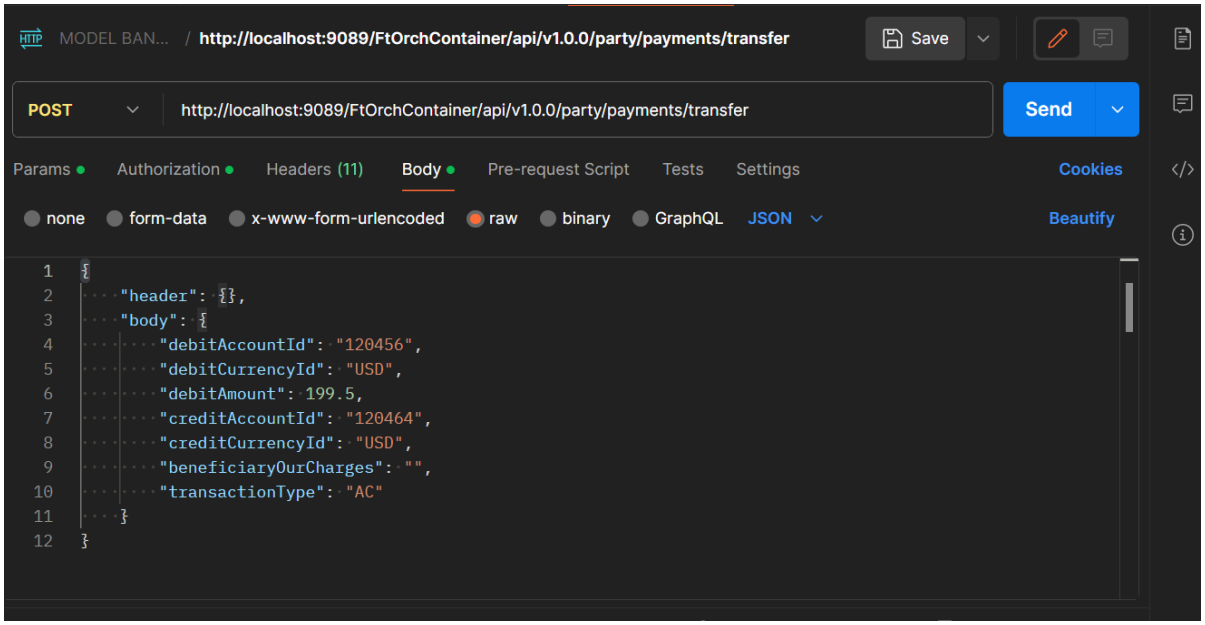
http://localhost:9089/FtEnqContainer/api/v1.0.0/party/payments/approve

Special Note: If Authorization issue occurs then we need to update the service xml

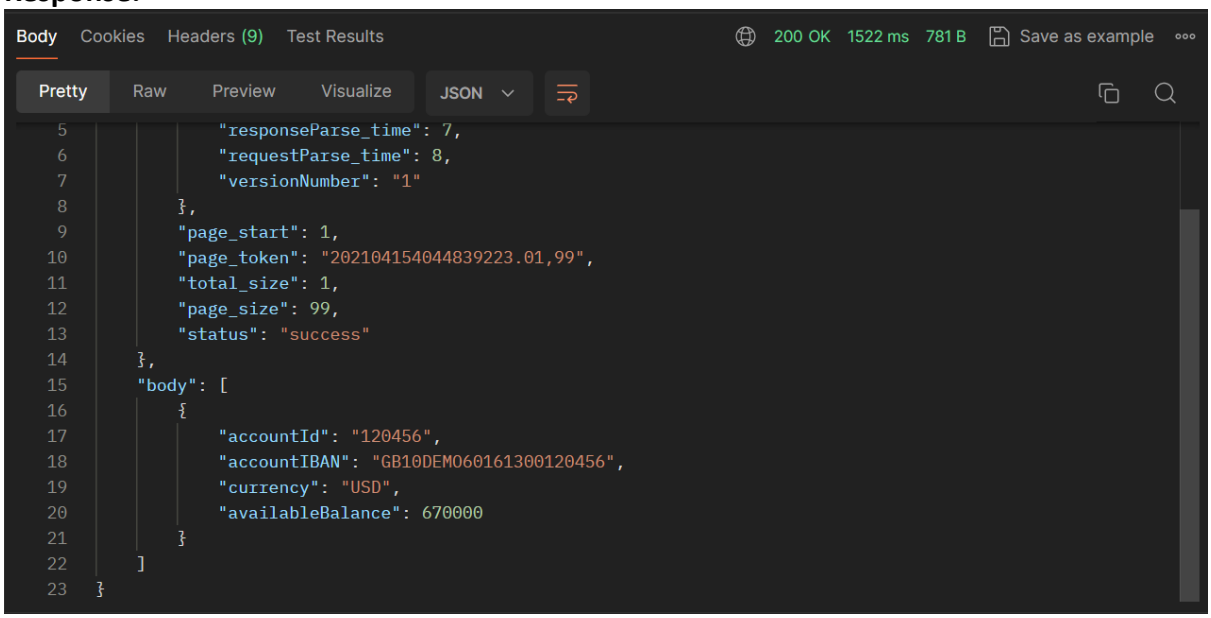
LATESTparty-payments-orc-api-service-v1.0.0.xml

```
<setProperty propertyName="validate_only">
        <header>true</header>
      </setProperty>
                  <setProperty propertyName="userDetails">
        <header>Authorization</header>
      </setProperty>
                  <setProperty propertyName="function">
        <constant>validate</constant>
      </setProperty>
     <process ref="t24VersionProcessor"/>
      <choice>
           <when>
             <simple>${headers.CamelHttpResponseCode} == 200</simple>
              <setHeader headerName="accountId">
                 <camel:jsonpath suppressExceptions="true">$.body.debitAccountId</camel:jsonpath>
               </setHeader>
               <setHeader headerName="Authorization">
                 <camel:exchangeProperty>userDetails</camel:exchangeProperty>
               </setHeader>
             <to uri="direct-vm:party-accounts.v1.0.0.getAcBalance" />
           </when>
        </choice>
       </route>
```

Step 7: Check it in POSTMAN



Response:



XACML/ JWT Token

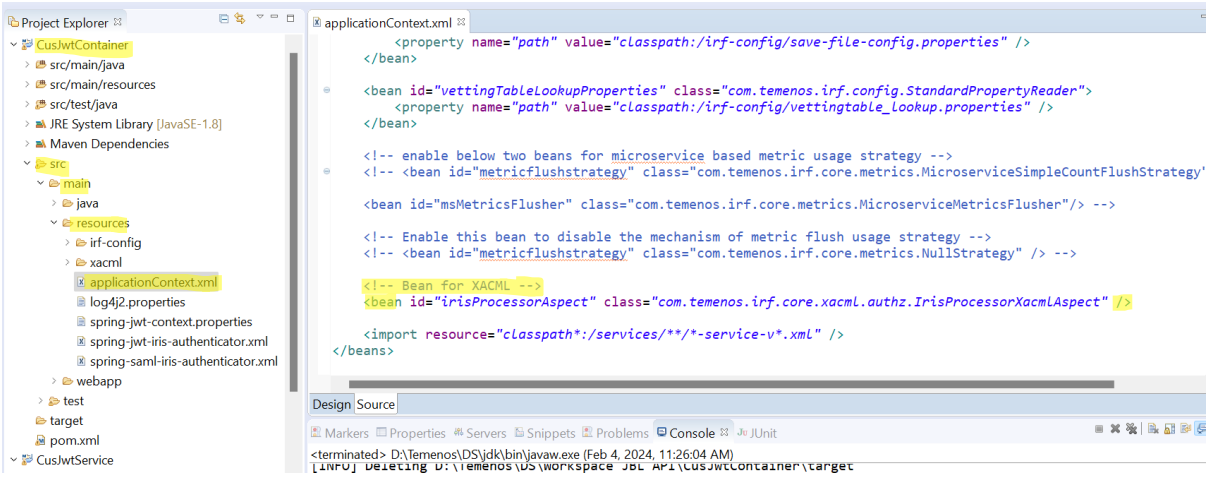
What is Xacml?
Extensible Access Control Markup Language – Which is used for attribute based authorization.

For example, An API request must contain “**roleId**” as “**ADMIN**”. This **roleId** we can pass through headers or **JWT Token**.

Xacml policy file will be available inside the container.

C:\Users\user\Downloads\irf-provider-container.war\WEB-INF\classes\xacml

To Enable xacml: NO NEED TO ENABLE XACML if it's JWT Only. *Keep the* ***applicationContext.xml*** *as it is.*
Uncommon below line to enable **xacml**.



IRIS Auth token Generation(JWT Only)

The functionality required in IRIS is to provide capability in IRIS to **validate the credentials** passed in **header**(**Basic Authentication**) against T24 and send back the JWT token for successful authentication.

Deploy the war file **irf-auth-token-generation-container.war** in JBoss.

Check it in the postman by the below URL

http://<host>:<port>/IrisAuthTokenGenerator/api/v1.0.0/generateauthtoken

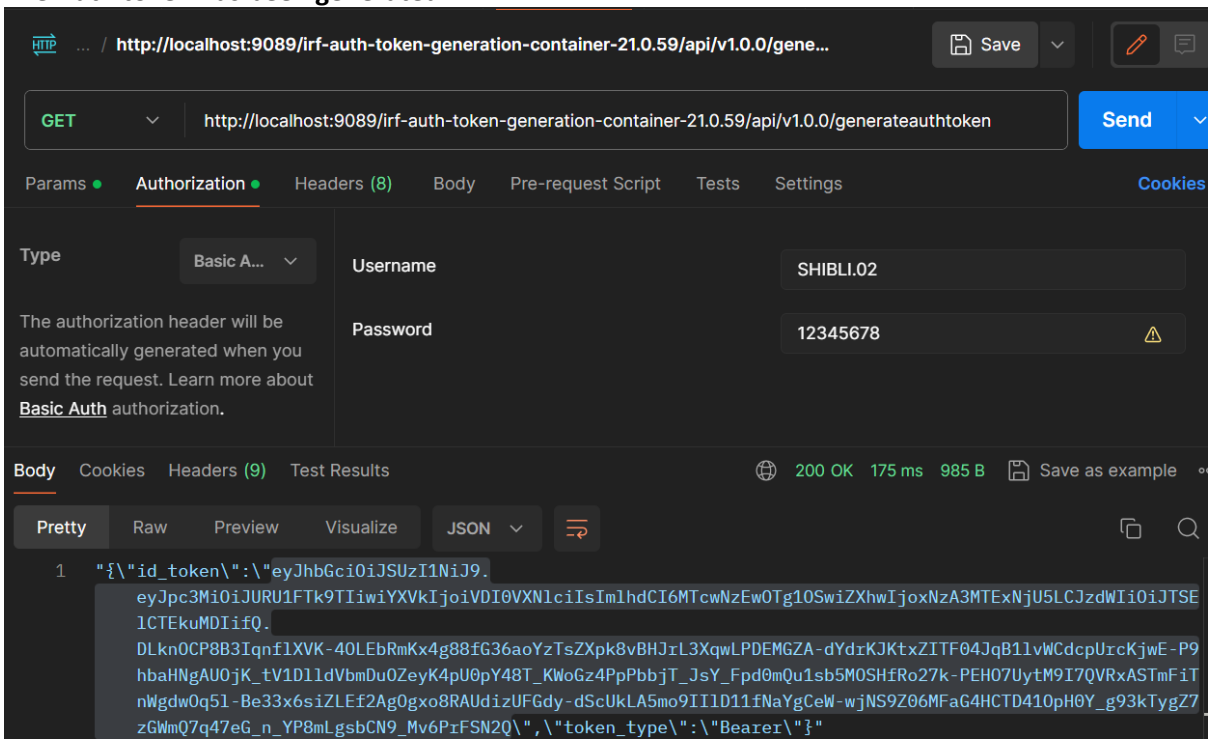
irf-auth-token-generation-container-21.0.59.war

http://<**host**>:<**port**>/**IrisAuthTokenGenerator**/api/v1.0.0/generateauthtoken

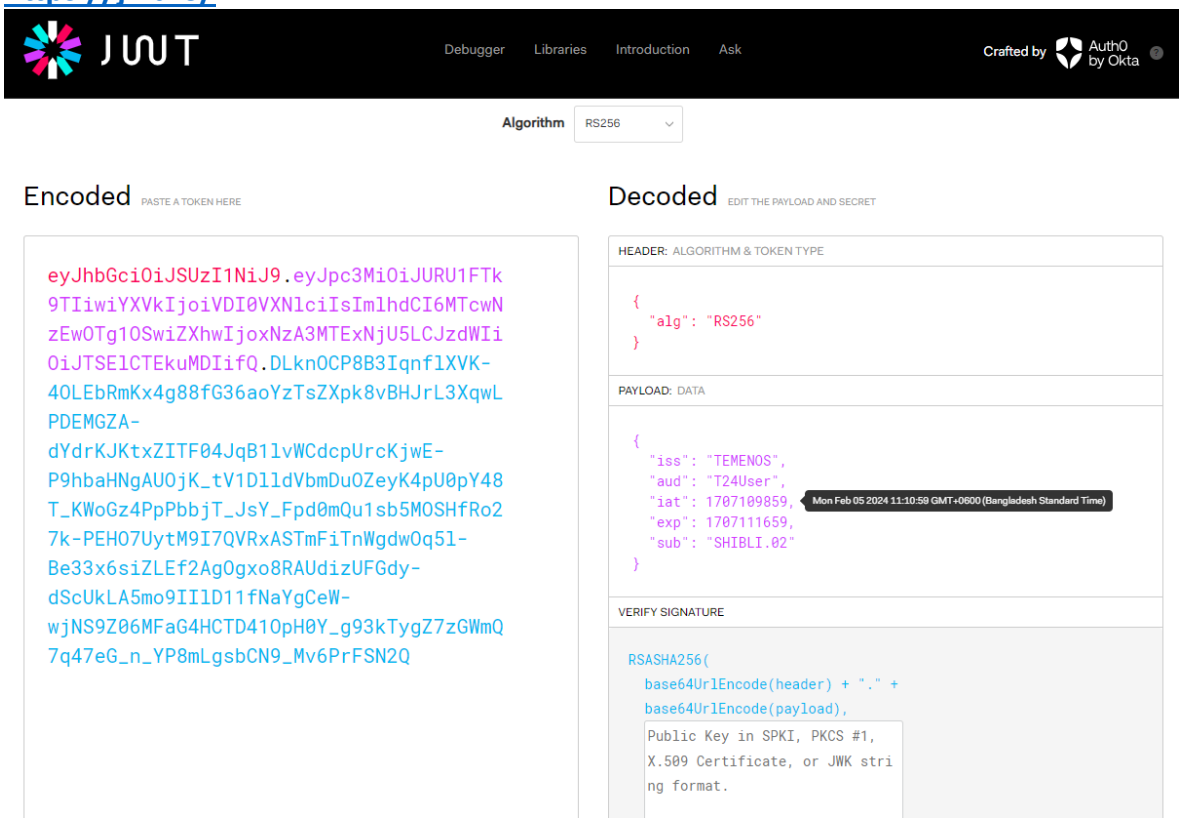
http://localhost:9089/irf-auth-token-generation-container-21.0.59/api/v1.0.0/generateauthtoken

pass with credentials,

The Auth token has been generated.

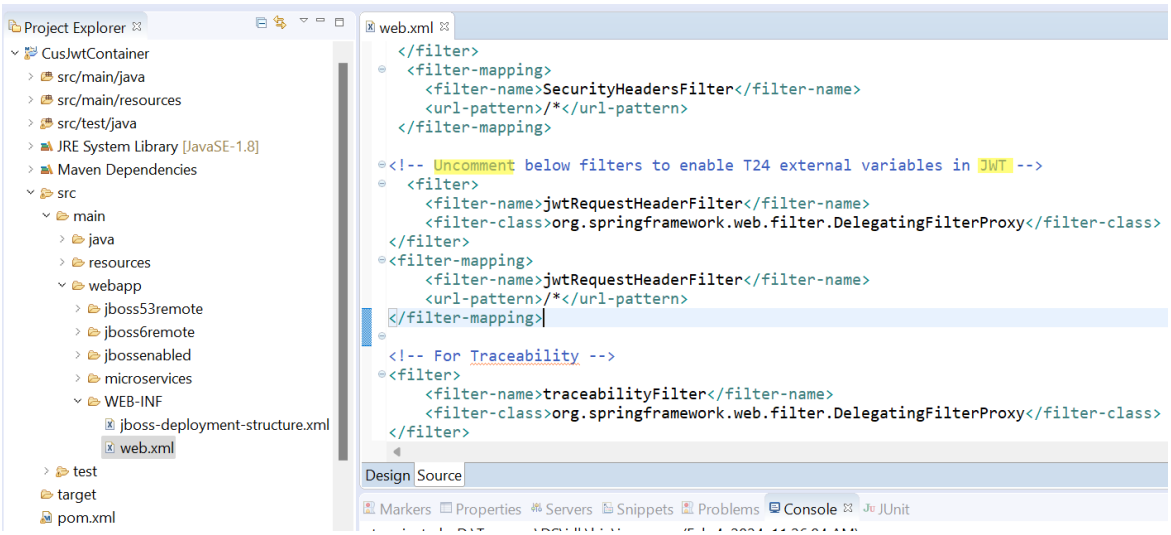


https://jwt.io/



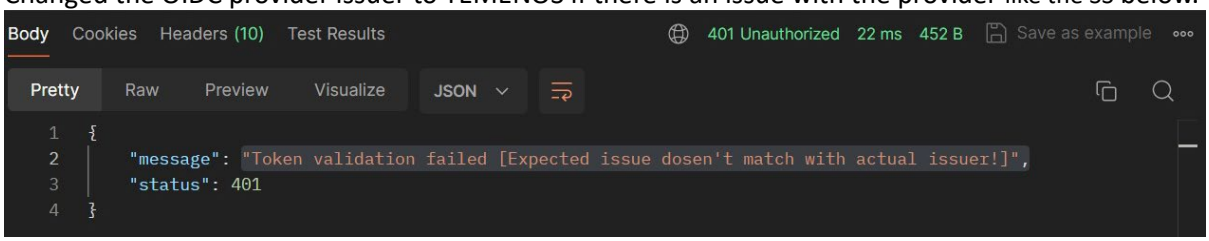
To Enable JWT:

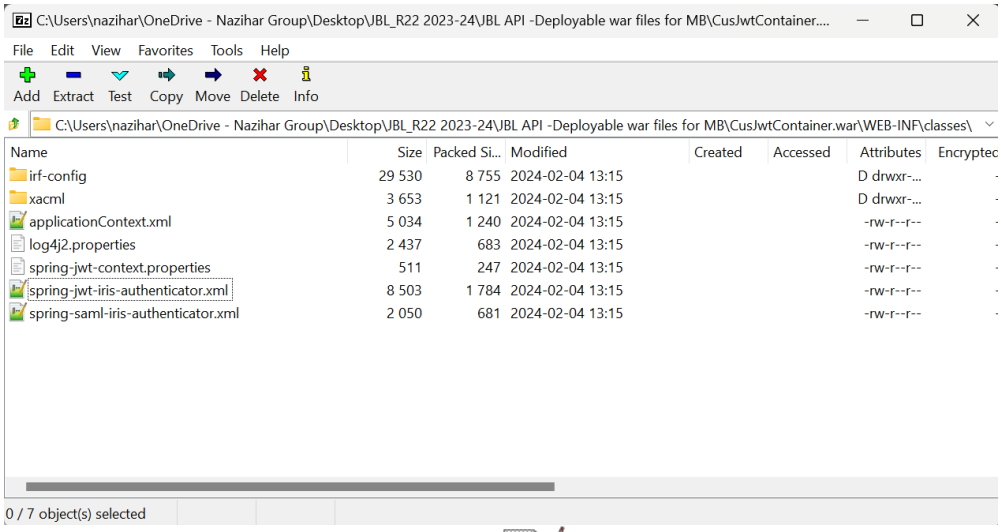
UnComment below set of lines to enable JWT.



Change spring-jwt-iris-authenticator.xml:

Changed the OIDC provider issuer to TEMENOS if there is an issue with the provider like the SS below.



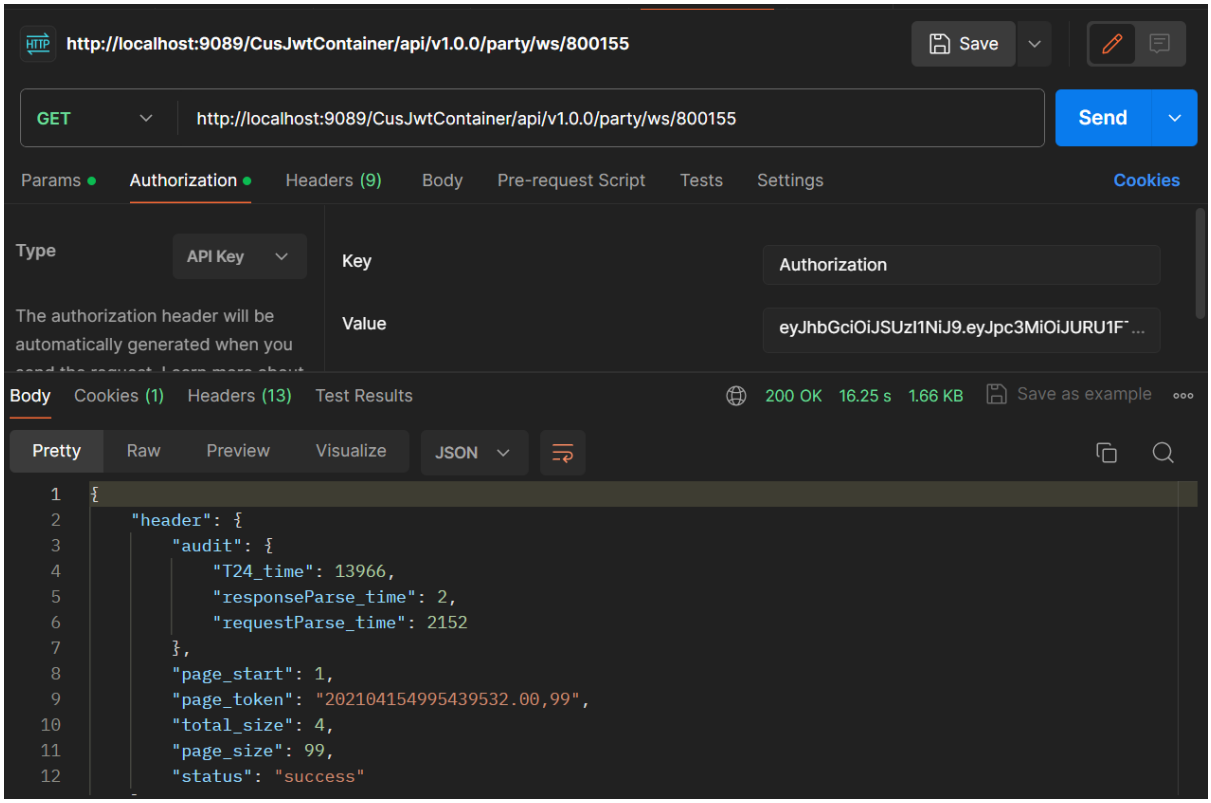


spring-jwt-iris-authenticator.xml

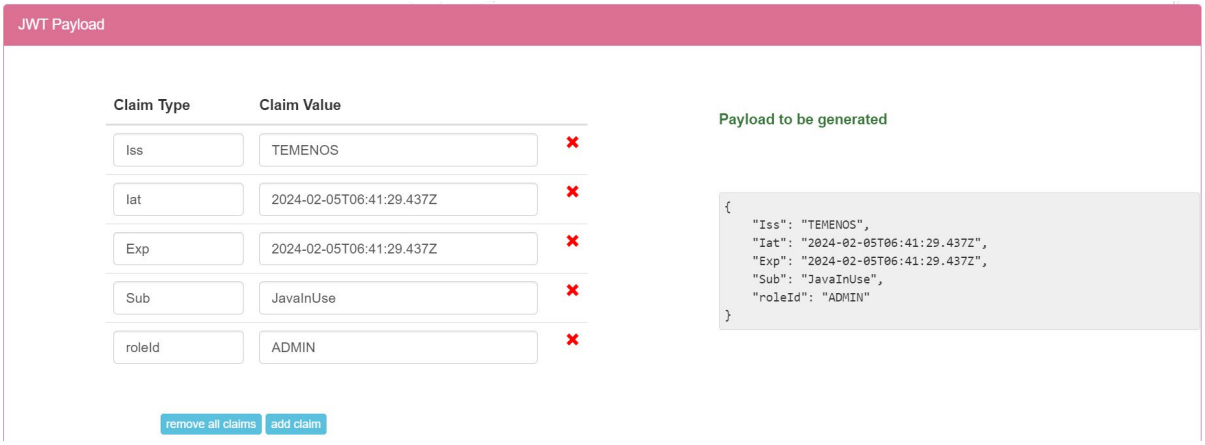
After changing the xml file it will look like this:

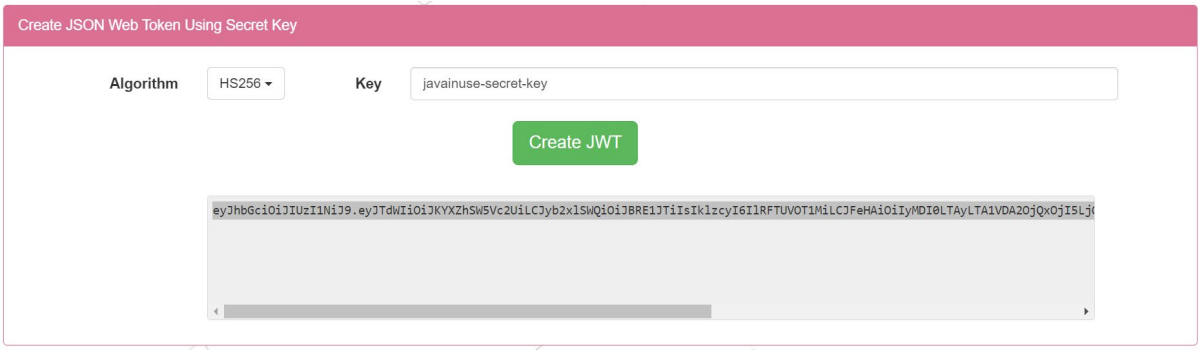
```
<!-- OIDC provider Server -->
<beans:bean id="oidcProviderServer" class="com.temenos.security.oidc.common.OidcProviderServer">
    <beans:property name="issuer" value="TEMENOS" />
    <beans:property name="pkEncoded" value="" />
    <beans:property name="pkCertEncoded" value="" />
    <beans:property name="pkCertFilePath" value="" />
    <beans:property name="pkJwksUri" value="" />
    <beans:property name="decryptingJwkEncoded" value="" />
    <beans:property name="principalClaim" value="sub" />
</beans:bean>
<!-- End OIDC provider Server -->
```

Now using **API Key,** the response is **OK** now.



Create JWT using Web tool(https://www.javainuse.com/jwtgenerator)





To be Continued...