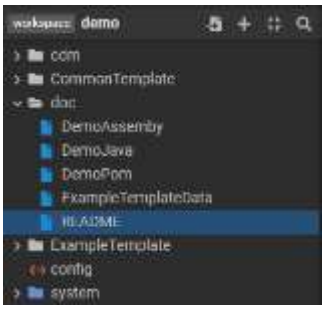
Onboarding FINOS and JAR/MAVEN into systems for FINOS

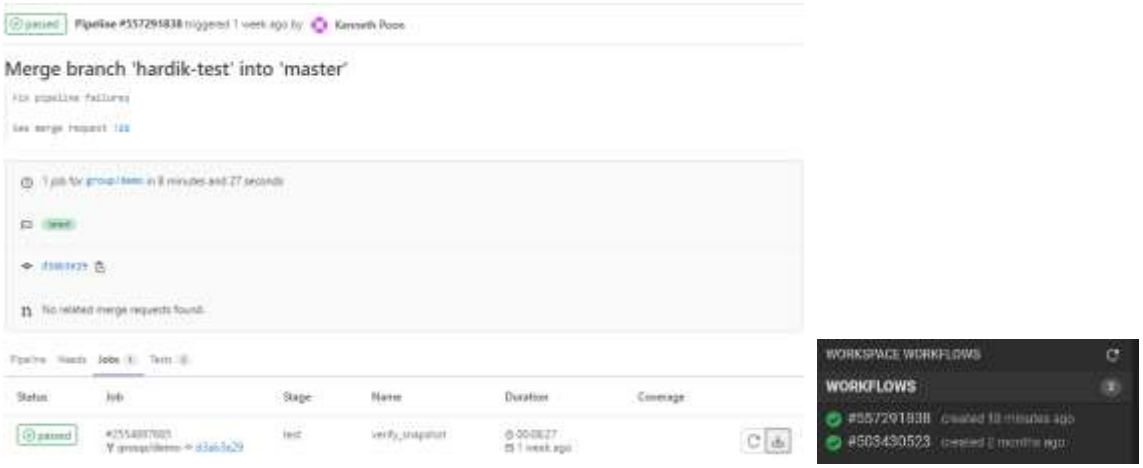
***Please note*: to run below testing you need JAVA 11 and MAVEN 3 setup in your local PC. Please contact your engineering team/ system administrator to help set these up as per your organization guidelines.**

1. Open Legend Studio and use group workspace – **Demo** in **Product Control Common Template project**. Click on next- https://legend.finos.org/studio/

2. To start testing, Open the **README** file to follow the testing steps below.



3. **From Workflow Manager** Download the latest artifact: Open the pipeline from **Workflow manager** and click on the latest accepted version to see the workflow detail, it will bring you to GItlab.com. Download the following files from the artifacts of the **passed** pipeline in GitLab.com with a green tick > **CI/CD > Jobs, download artifacts files** to your **developer drive** as a zip file. You need to extract the files in developer drive (assuming **H:/** drive is the developer drive).[1]



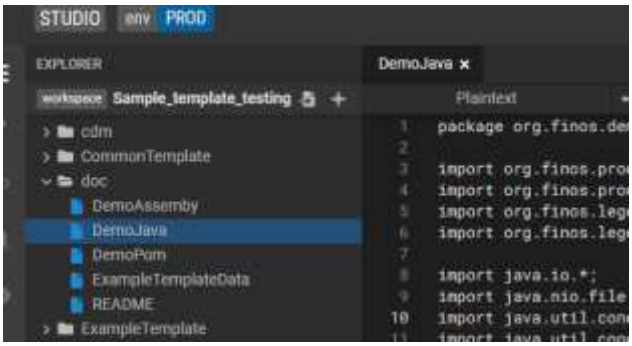
[1] Whenever we author changes in Studio and save those changes, in background we merge those changes to GitLab. This triggers a workflow which process the Studio code and generate jars which helps integrating Studio code with java code using Maven.
Service execution jar contains code to execute the services you authored in Studio.

4. In H:/ drive create a folder Legend and within that create a folder "**demo**" and extract the artifacts file there (*H:\Legend\demo\artifacts*). This file contains all the job running artifacts for the M2M mappings.
   * the parent pom file
   * the service-execution jar file
   * the service-execution pom file

5. In your local developer **H:\ Drive**, [2]

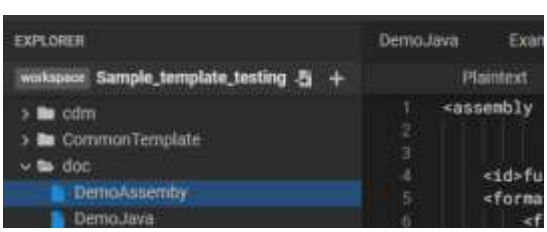
```
template-demo
|-src
| |-main
|   |-java
|   | |-org
|   |   |-finos
|   |     |-demo
|   |       |-Demo.java  (copy content from doc::DemoJava)
|   |-resources (in main folder)
|     |-Sample.csv      (copy content from doc::ExampleTemplateData)
|-full-assembly.xml     (copy content from doc::DemoAssembly)
|-pom.xml               (copy content from doc::DemoPom)
```

   a. Create a maven project workspace under a new folder "**template-demo**" in the following folder structure: ***H:\template-demo\src\main\java\org\finos\demo***.
   b. In the demo folder create a new text file and then copy the contents from **DemoJava** from FINOS Legend doc folder to the text file. Then save the file as **Demo.java** so that it saves as a java file (Save as type should be All Files).



   c. Now in ***template-demo>src>main*** folder, create a "**resources**" folder. Here creates a new csv file "**Sample.csv**" which will have the sample template input data. You can take the data from "ExampleTemplateData" under Doc from FINOS. **Create a text file and then save it as .csv file**. *In sample csv file make sure the date format is correct. Please ensure the sample data saved in a correct date format (YYYY-MM-DD) in the csv file.*
   d. Now in the ***template-demo*** folder, create a full-assembly xml file. First create a text file and copy the contents from **DemoAssembly** and then save it as a .xml file.
   e. Now in the ***template-demo*** folder, create a pom.xml file. First create a text file and copy the contents from **DemoPom** and then save it as a **pom.xml** file.

[2] In this step, we go through series of step to setup a vanilla Maven project. We have provided example files which you can use to get started quickly.

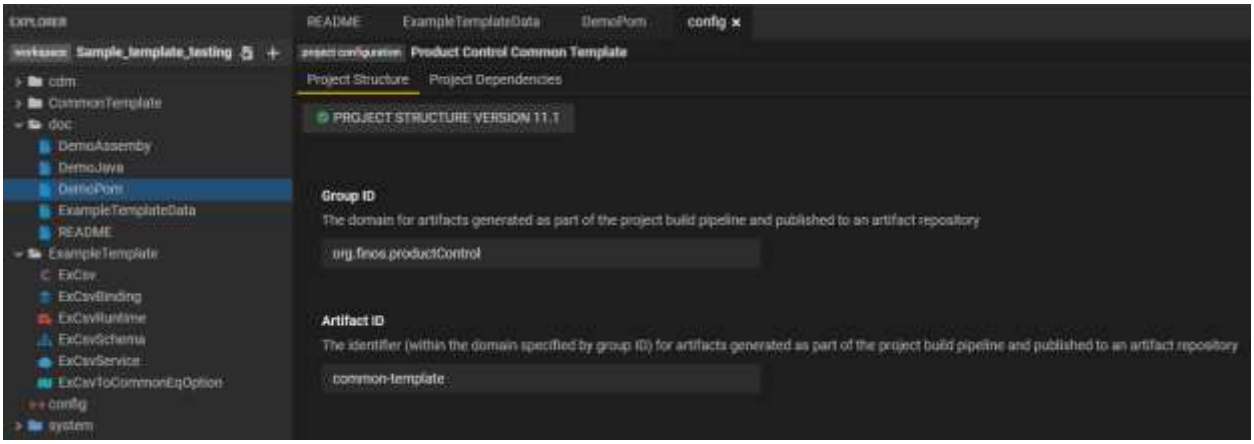


6. Now in the **pom.xml** file created above, make sure the properties in pom.xml of the template-demo project are as follows. Change only if different to below. Open the pom file as a text file to edit in the <properties> section.[3]

<template.groupId> = "org.finos.productcontrol" (groupId of the Legend project, should not need to change).
<template.artifactId> = "common-template" artifact Id of the Legend project (should not need to change).
<template.workspace> = the workspace you are using in legend (see instruction in pom.xml). replace update_me to "**group_demo**".

```
    <properties>
        <template.groupId>org.finos.productControl</template.groupId>
        <template.artifactId>common-template</template.artifactId>

        <!-- ********************************************************************************************************** -->
        <!-- update template.workspace                                                                                  -->
        <!-- ************************                                                                                   -->
        <!-- If a group workspace then use group_<workspace_name>                                                      -->
        <!-- If a personal workspace then use workspace_<owner>_<worksapce_name>' where <owner> is your GitLab account id. -->
        <!-- ********************************************************************************************************** -->
        <template.workspace>group_demo</template.workspace>
```

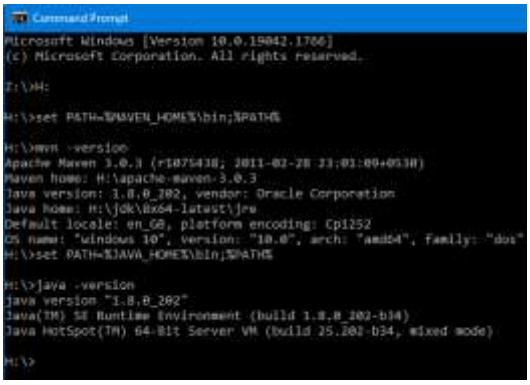


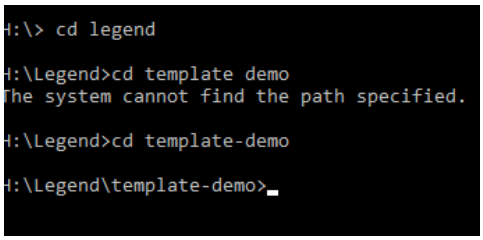

```
<properties>
    <template.groupId>org.finos.productControl</template.groupId>
    <template.artifactId>common-template</template.artifactId>

    <!-- ********************************************************************************************************** -->
    <!-- update template.workspace                                                                                  -->
    <!-- ************************                                                                                   -->
    <!-- If a group workspace then use group_<workspace_name>                                                      -->
    <!-- If a personal workspace then use workspace_<owner>_<worksapce_name>' where <owner> is your GitLab account id. -->
    <!-- ********************************************************************************************************** -->
    <template.workspace>group_demo</template.workspace>
```

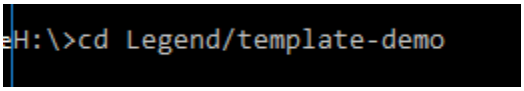
[3] Maven helps us use external dependencies with our codebase. In this step we provide the path of the jars we downloaded/extracted in step 3 from GitLab to our java project we set-up in step 4.

7. Open CMD. Switch to H drive (write H: and enter).
8. Please make sure the below highlighted version of Maven and Java are installed in your developer drive before beginning this testing. In CMD, write **java –version** and **mvn –version** to check the correct applications installed.
   Please contact your system administrator to get this installed.





9. Now use the file path downloaded from GitLab pipeline from step1 in **H:\Legend\demo\artifacts**. In CMD, write "cd" to change directory. In H:/ path write **cd Legend/template-demo** to find the path where we have saved the pom and xml file. [4]



   Now run the below 2 paths to change the links to correct demo file. Make sure no space in b/w. Depending on the artifacts you are downloading from your Step1, the below path an dfile number would change. Please update before you run it initially.

   a. **<path-to-parent-pom> is for link to the parent pom file in artifacts**:
      mvn install:install-file -Dfile=***H:\Legend\demo\artifacts\common-template-group_demo-20220707.134704-1.pom*** -DpomFile=***H:\Legend\demo\artifacts\common-template-group_demo-20220707.134704-1.pom***

   b. **<path-to-service-execution-jar> is for the service execution jar file**:
      mvn install:install-file -Dfile=***H:\Legend\demo\artifacts\common-template-service-execution-group_demo-20220707.135402-1.jar*** -DpomFile=***H:\Legend\demo\artifacts\common-template-service-execution-group_demo-20220707.135402-1.pom***

10. In CMD, in the template_demo directory in H:/ drive write **mvn package**.[5]

[4] In this step we install the jars we downloaded in step 3 to the path we specified in step 5.
[5] In this step we use maven to package together the code we authored in step 4 and jars we downloaded in step 3. We can do this because we provided the path of downloaded jars in step 5.
Produced jar from this step helps us execute service transformation using java in stand-alone fashion. We use this jar in step 8 & step 9.

11. *Run the java jar file in CMD using:* ***java -jar target/template-demo-full.jar csv-to-common-template H:\Legend\template-demo\src\main\resource\Sample.csv common-template.json***
    Output file name
    Input file name

12. Convert the result above to common template JSON and write it to common-template.json. This will take the output from above, convert it to CDM JSON and write it to **cdm.json**
> java -jar target/template-demo-full.jar common-template-to-cdm common-template.json cdm.json

13. The output is saved in the template-demo folder in your developer driver H: under the name of "common-template" as a JSON file. You can convert this into csv and send to the vendors.
