**Building the Shared Library from Eclipse**

Odyssey APIs Generator is a builder for creating a shared library based on Tyler Technologies' XSD files.

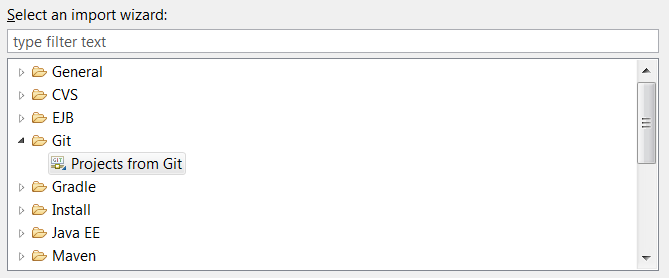
Java 10, Apache Maven 3.5.2, and Eclipse Neon.3 Release (4.6.3) or above must be installed before you execute the following procedure.

1. **Eclipse Setup**

Project files are available in the following GitHub repository: <https://github.com/rperalta1/OdysseyAPIGenerator>

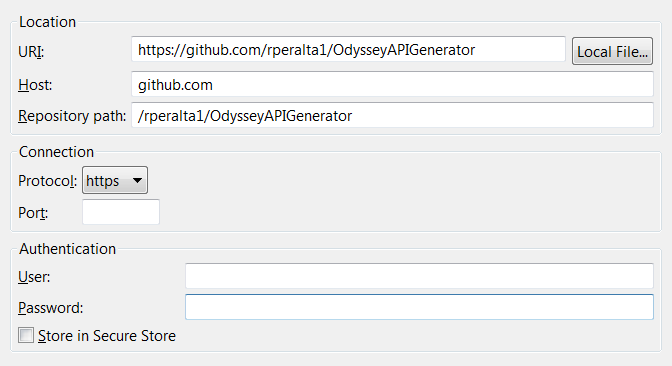
*If you are familiar with GitHub download the project into an Eclipse workspace and go to step number 2.*

*Create a new workspace* (or directory), open it with Eclipse, from the Eclipse’s ***Project Explorer*** tab, and right click select ***Import*** -> ***Import…*** -> ***Git*** -> ***Projects from Git*** (Figure 1).



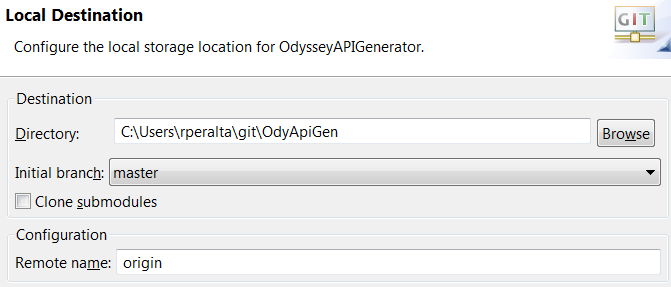
**Figure 1** Selecting an import Wizard

Click on next and select ***Clone URI***, fill the next window with the information provided in Figure 2; *do not forget to use your own GitHub credentials* (***User*** and ***Password***):

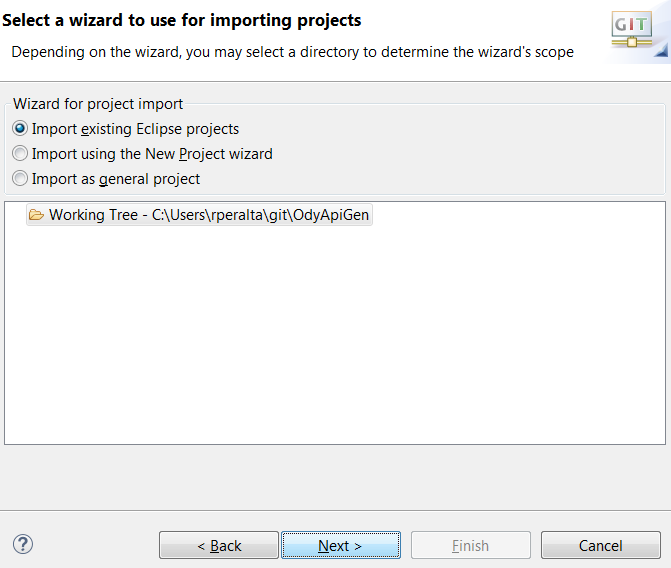


**Figure 2** Source Git Repository selection

The next window is for ***Branch Selection,*** choose ***master*** and click on Next. Now select the local destination as presented in Figure 3.

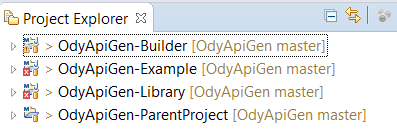


**Figure 3** Local Destination selection

Click on ***Next***. Finally import the project into the workspace as shown in Figure 4.

**Figure 4** Importing Projects into Eclipse’s Workspace

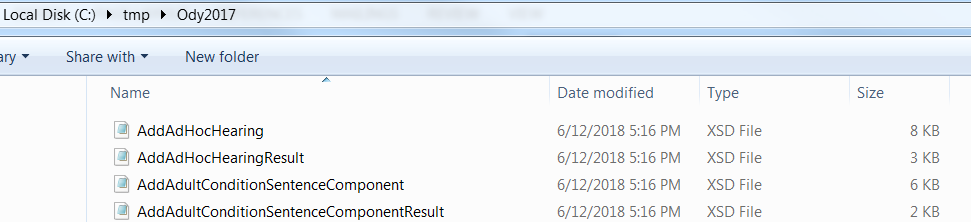
If everything was done correctly you will find the workspace as depicted on Figure 5.



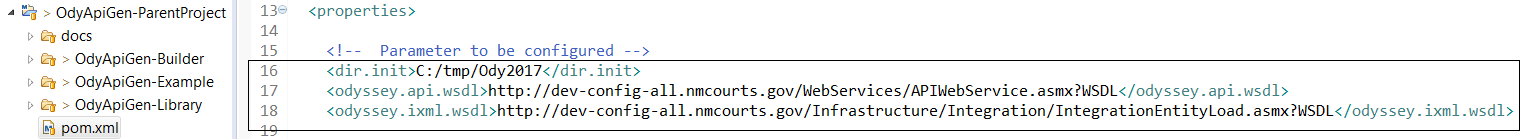
**Figure 5** Eclipse workspace after importing projects

1. **Parameters Setup**

Before the library is generated some inputs and parameters must be adjusted. First, create a directory, for instance c:\tmp\2017, and in that folder put all the XSD files provided by Tyler as presented in Figure 6 (let’s assume we are working with 2017 XSD files). Please note that we have successfully tested this procedure with Odyssey 2013 and Odyssey 2014 XSDs, which work without modification. However, for Odyssey 2017, the XSD files provided by Tyler have a few minor errors which must be corrected prior to this procedure being performed. Details on the issues with Odyssey 2017 XSDs, as well as work-arounds and fixes, are described in the document entitled *Odyssey API Library Overview*.docx found in this project’s /docs folder.



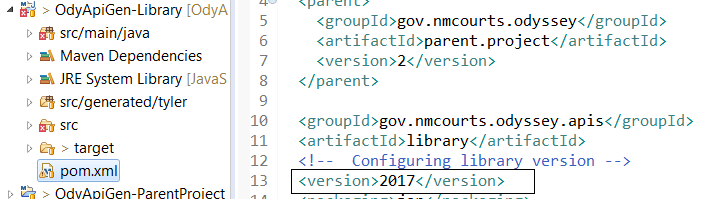
**Figure 6** View of the directory where XSD files are located

Now, it is time to adjust parameters inside the pom.xml files. Go to the ***OdyApiGen-ParentProject*** and open the pom.xml file in Eclipse (Figure 7).

**Figure 7** Configuring parameters parent-project’s pom.xml file

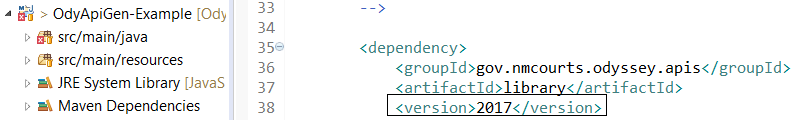
Line 16 to 18 define variables needed for the building process; variable “***dir.init***” must hold the absolute path to the directory where the Tyler XSD files were stored. For this example the values must be “***c:\tmp\2017”***. The next two variables “***odyssey.api.wsdl***” and “***odyssey.ixml.wsdl***” stored the URL were your Odyssey WDSL are defined. In the current example we are using WSDLs from a test environment – “***dev-config***”. After configurations changes are done, do not forget to save the pom.xml file.

The next parameter configuration must be done on the ***OdyApiGen-Library*** project, pom.xml file (Figure 8). Go to line 13 and write the version of the library, for this example it is 2017 (remember we are assuming we are working with 2017 XSD files). When configuration is done save the file.



**Figure 8** Configuring parameters library’s pom.xml file

In a similar fashion, go to ***OdyApiGen-Example*** project, open the pom.xml file, go to line 38, and set up the version as 2017 (Figure 9). Save the file when changes are done. When the shared library is generated it will be installed in the local Maven repository of the computer where you are working, and it will be identified like any other JAR file by the groupID, artifact, and version. The ***OdyApiGen-Library*** project defines the version as 2017 and the ***OdyApiGen-Example*** project must use the same value in order to include the generated share library as a Maven dependency on the ***OdyApiGen-Example*** project.

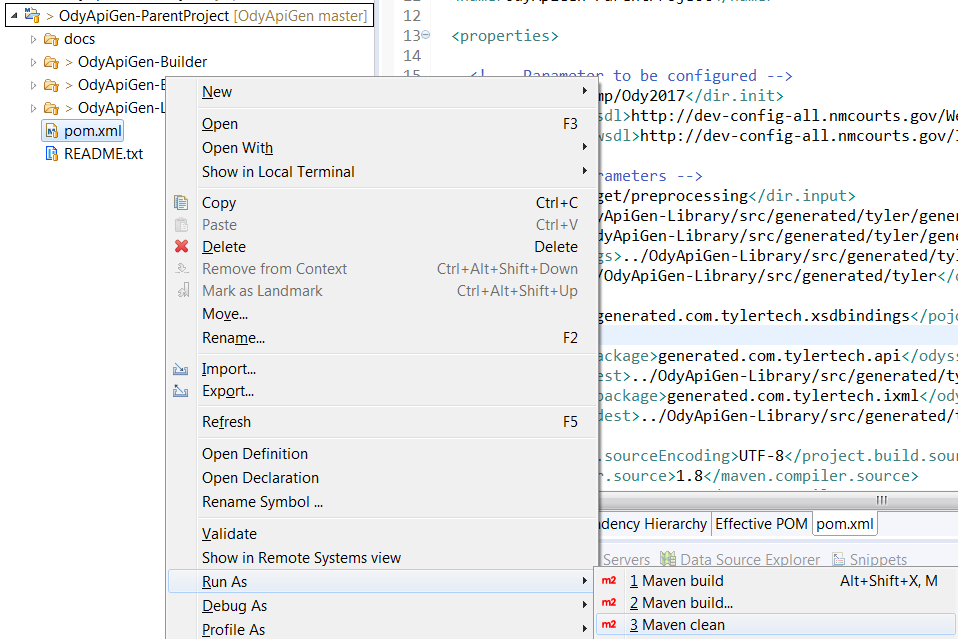


**Figure 9** Configuring parameters example’s pom.xml file

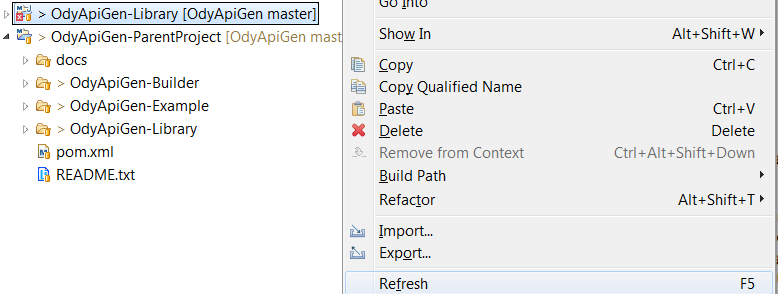
1. **Library Creation**

***OdyApiGen-ParentProject*** is a module maven project that will call the ***OdyApiGen-Builder*** module/project first and then the ***OdyApiGen-Library*** project. For running both modules, go to ***OdyApiGen-ParentProject*** project select the pom.xml file, right click select “***Run As***”, and click on “***Maven clean***” (details can be found on Figure 10). In this first phase, clean, previous generated files will be deleted if there is any.

When the clean phase has finished it is a good idea to refresh the project. Just select every module/project, call its context menu, and select the “***Refresh***”. Figure 11 shows how to perform the task on ***OdyApiGen-*** ***Library*** module/project.

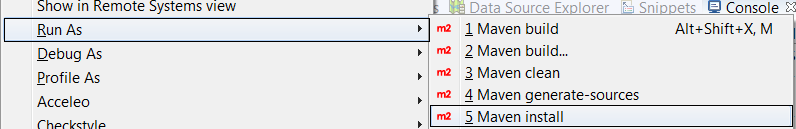


**Figure 10** Cleaning project



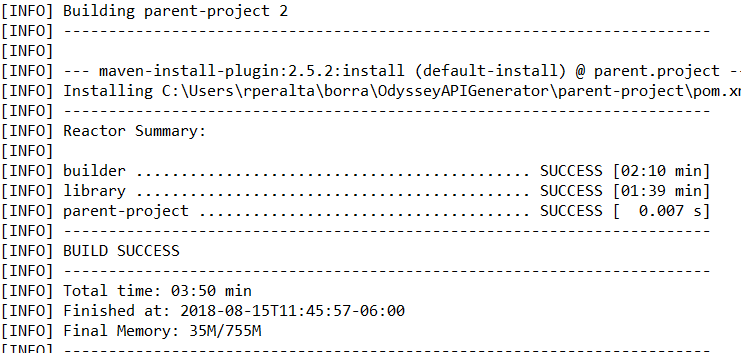
**Figure 11** Refreshing projects

To create the library, again select the pom.xml file under the ***OdyApiGen-ParentProject***, righ click and select “***Maven install***” (Figure 12).



**Figure 12** Generating the shared library

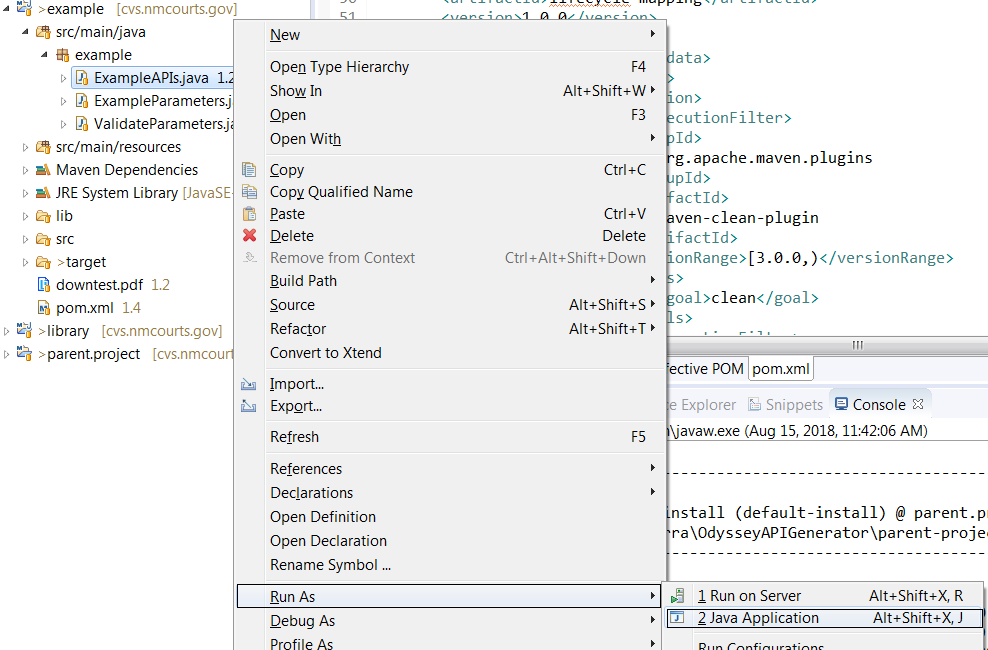
If all the process were done properly you should get message on Eclipse console that indicates the build was successful (Figure 13). If not, go to the Project Eclipse menu and select “**Clean**” and tried again. Notice also the share library will be available as a JAR file under ***OdyApiGen-ParentProject/OdyApiGen-Library/target*** folder. From there it can be copy to other projects and be used like any other JAR file (it must be included on the build path of the project).



**Figure 13** Console view after the generation of the shared library ends

1. **Running the Example Project**

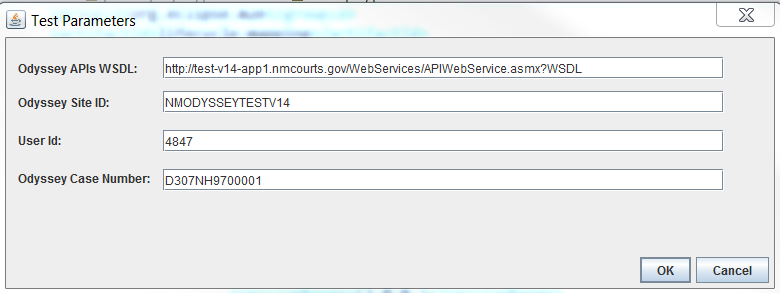
Go to the workspace and collapse the ***OdyApiGen-Example*** project until you find the example package. Then, select class ***ExampleAPIs.java***, and run it as a Java Application as presented on Figure 21.



**Figure 21** How to run the example project

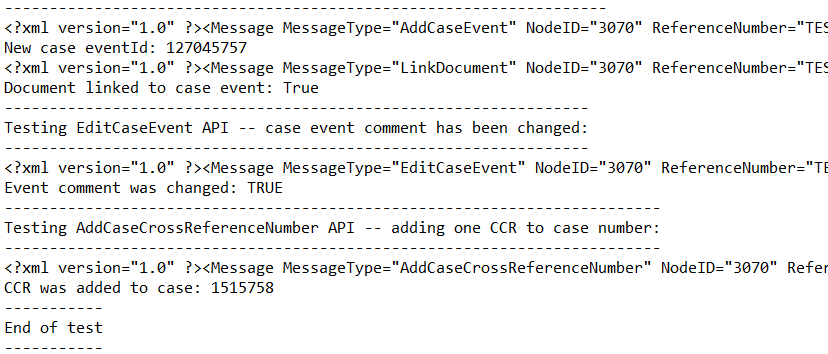
The example program will use the recently library installed on the local Maven repository, and will present to program users a windows for configuring parameters (Figure 22). Use the information you have from your Tyler Odyssey installation to provide the API WSDL, Odyssey Site ID, and User ID. The last parameter is the ID of a user account defined on Odyssey. To find it select a user account (i.e. “***appwtr***”) and then run a query on table Operations.dbo.AppUser of your Odyssey DB, and you will find the User ID (4847 corresponds to “***appwtr***” user account).

Also you have to provide a valid Odyssey case number with at least one case event that has a PDF document attached to it. If you do not provide an Odyssey case number with this data just part of the program will work. The code stored on ExampleAPIs.java will test FindCaseByCaseNumber, LoadCase, GetDocumentInfoByEntity, GetDocument, AddCaseEvent, LinkDocument, EditCaseEvent, and AddCaseCrossReferenceNumber APIs.



**Figure 22** Parameters to be configured during the execution of the example

If everything was done properly, you will see on the console IXML messages exchanges between the program and Odyssey as depicted on Figure 23.



**Figure 23** Console view after the example program finished