|  |
| --- |
|  |
| Open Reports System Developer's Guide |
|  |

Table of Contents

[Table of Contents 3](#_Toc524523778)

[Development Environment Setup 4](#_Toc524523779)

[Pre-requisites 4](#_Toc524523780)

[Software Requirements 4](#_Toc524523781)

[Setup 4](#_Toc524523782)

[Cloning the Jasper Repository 4](#_Toc524523783)

[Eclipse Setup 7](#_Toc524523784)

[Module Compilation 9](#_Toc524523785)

[Project Profiling 10](#_Toc524523786)

[Workflow 11](#_Toc524523787)

[Diagram 11](#_Toc524523788)

[New Features Workflow 11](#_Toc524523789)

[Updating/Revise Features Workflow 12](#_Toc524523790)

[Worflow Steps 14](#_Toc524523791)

# Development Environment Setup

## Pre-requisites

It is assumed that the reader already knows the following:

* Java
* Maven
* Git

## Software Requirements

* Java 8 ([installation for UBUNTU](https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-get-on-ubuntu-16-04))
* Maven ([installation for UBUNTU](https://www.mkyong.com/maven/how-to-install-maven-in-ubuntu/))
* Git ([installation for UBUNTU](https://www.liquidweb.com/kb/install-git-ubuntu-16-04-lts/))

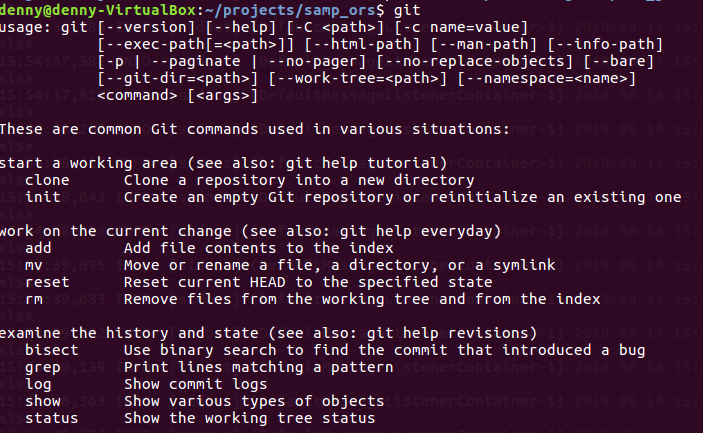
## Setup

*Notes:*

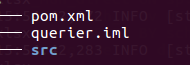
* *Installation may vary depending on the type of OS for the current dev setup it is for Ubuntu.*

### Cloning the Jasper Repository

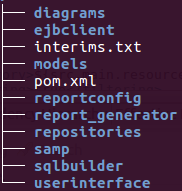
1. Execute first in console if git is successfully installed by executing git command. Console should show something like



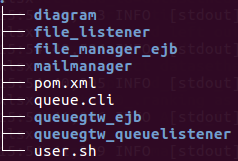
1. There will be 3 main directory for ORS
   1. Querier
      1. Create a directory querier
      2. In the directory, execute *git init*
      3. Then execute *git remote add origin* [*https://gitlab.deped.gov.ph/ors/querier*](https://gitlab.deped.gov.ph/ors/querier)
      4. Execute *git fetch* then place credentials
      5. After fetch, execute *git checkout master* then directory should contain



* + 1. Compile using maven. Execute *mvn clean package install* and a build success should appear.
    2. Execute *mvn eclipse:clean && mvn eclipse:eclipse*
  1. ORS
     1. Create a directory ors
     2. In the directory, execute *git init*
     3. Then execute *git remote add origin* [*https://gitlab.deped.gov.ph/ors/ors*](https://gitlab.deped.gov.ph/ors/ors)
     4. Execute *git fetch* then place credentials
     5. After fetch, execute *git checkout staging* then directory should contain



* + 1. Execute *mvn eclipse:clean && mvn eclipse:eclipse*
  1. Queue Gateway
     1. Create a directory queuegtw
     2. In the directory, execute *git init*
     3. Then execute *git remote add origin* [*https://gitlab.deped.gov.ph/ors/queuegateway*](https://gitlab.deped.gov.ph/ors/queuegateway)
     4. Execute git fetch then place credentials
     5. After fetch, execute *git checkout master* then directory should contain



* + 1. Execute *mvn eclipse:clean && mvn eclipse:eclipse*

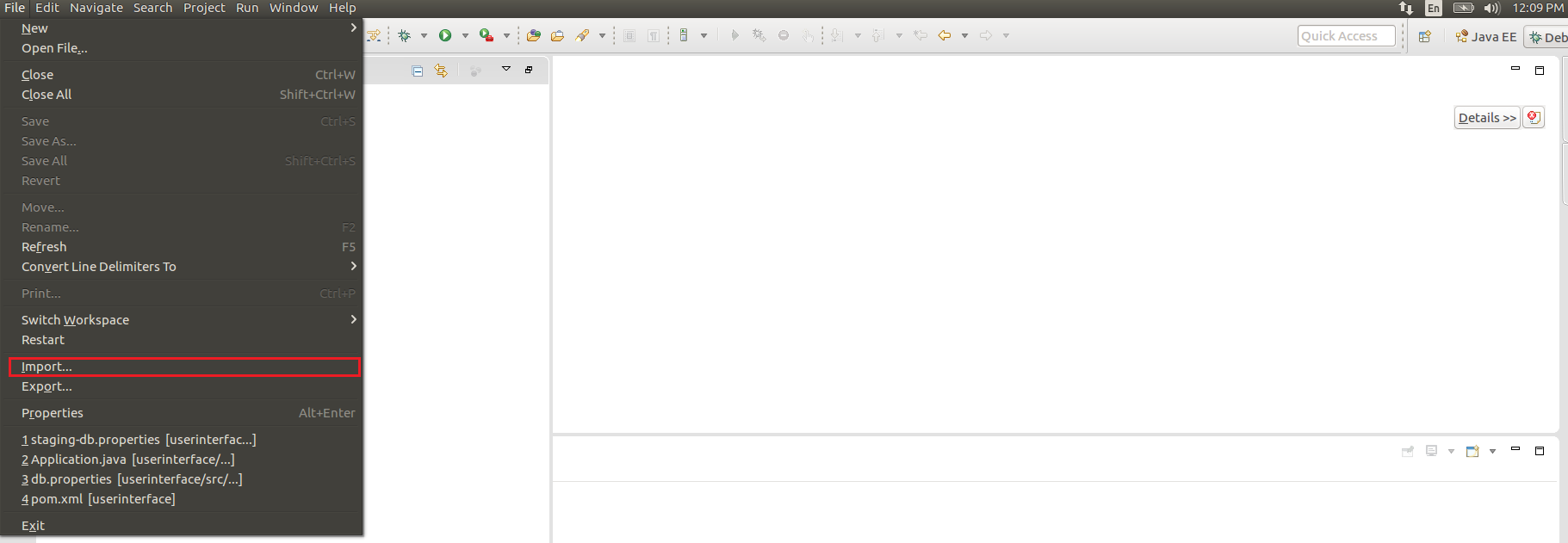
**Project Directory Hierarchy and Definition**

There are 3 main directories for ORS and each of them including their subdirectories have pom.xml files. Pom.xml is a metadata for maven to compile and organize projects.

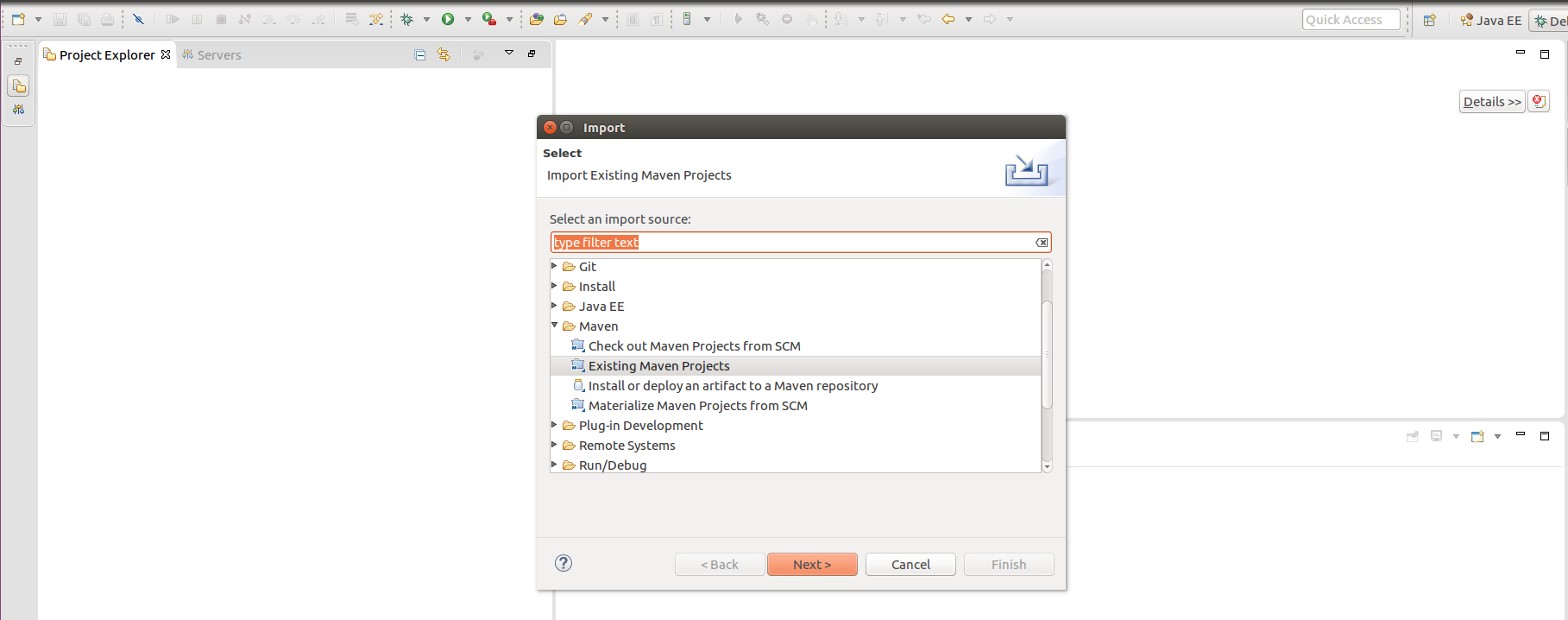
* Querier
  + src - contains source code api for sql builder library
* ORS
  + models - This is where entities or java objects are stored
  + reportconfig - general configuration files
  + report\_generator - api module for generation of reports
  + repositories - api for spring data hibernate
  + sqlbuilder - api module for sql building (dependency on querier module)
  + userinterface - module for UI of the system (jsp, html etc)
* Queue Gateway
  + file\_listener - manages file deletion and uses redis api to keep track of time to live
  + mail\_manager - api for sending email
  + queuegtw\_ejb - an interface for userinterface to connect for report request.
  + queuegtw\_queuelistener - a module that listens to a queue for report processing

### Eclipse Setup

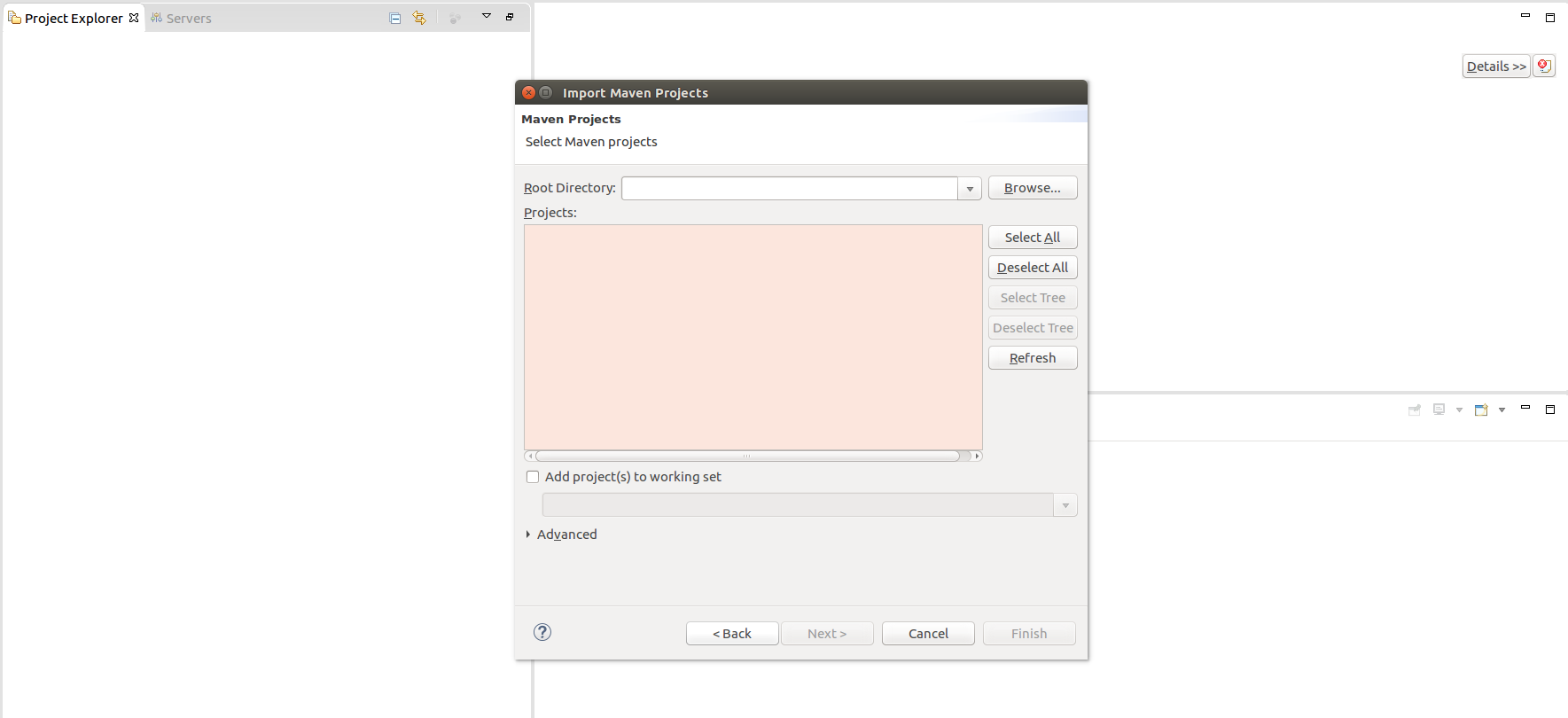
1. Download Eclipse installer [here](https://www.eclipse.org/downloads/packages/installer)
2. Follow instructions on how to install then run eclipse
3. On the eclipse main page you should see this screen



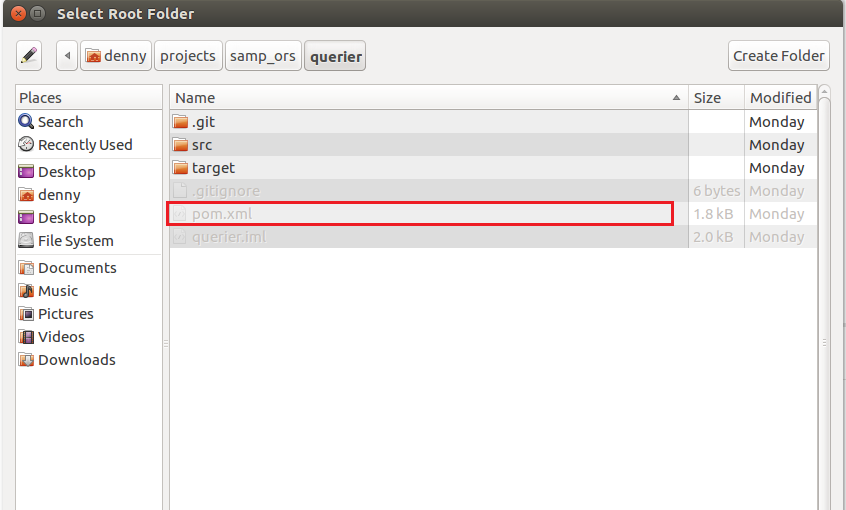
1. In the menu bar, click File -> import



1. In the import page, select Existing Maven Projects then click Next.



1. Select the Browse button in Root Directory. A file dialogue box will appear. Locate your project directory and look for the pom.xml



1. Once pom.xml is located, select OK.

### Module Compilation

* **For first time compile (in order)**
  + Querier
    - Go to querier/ then execute *mvn clean package install -Dmaven.test.skip=true*
  + ORS
    - Go to ors/ then execute *mvn clean package install -Dmaven.test.skip=true*
  + Queue Gateway
    - Go to queuegtw/ then do the following in sequence for every directory
      * queuegtw\_ejb/
        + *mvn clean package && mvn install -Dmaven.test.skip=true*
      * file\_manager\_ejb/
        + *mvn clean package && mvn install -Dmaven.test.skip=true*
      * mailmanager/
        + *mvn clean package install -Dmaven.test.skip=true*
      * file\_listener/
        + *mvn clean package -Dmaven.test.skip=true*
      * queuegtw\_queuelistener/
        + *mvn clean package -Dmaven.test.skip=true*
* **For recurring compile**
  + For library modules, the command for compile is *mvn clean package install -Dmaven.test.skip=true*. Modules are:
    - ors/
      * Models
      * Repositories
      * Reportconfig
      * Report\_generator
      * Sqlbuilder
    - querier/
    - queuegtw/
      * special case for the following modules use *mvn clean package && mvn install*
        + File\_manager\_ejb
        + Queuegtw\_ejb
      * mailmanager
  + For deployable modules, use command *mvn clean package -Dmaven.test.skip=true.* Modules are:
    - ors/
      * Userinterface
    - queuegtw/
      * File\_listener
      * queuegtw\_queuelistener

### Project Profiling

*Notes*

* + *Currently has two active profiles: local and staging. This is done via maven which enable every profile. Each profile would have different configurations assigned. For instance in the local profile, the database would be pointing at the local database and for staging which would be pointing to a different database. This is not limited to database connections alone but to other connectivities as well across project hierarchy.*
  + *The objective of the local setup is to enable the module to run in the local machine while staging setup is to enable the module to be deployed in staging*
* ORS
  + Local Profile Setup
    - Go to ors directory
    - Execute *mvn eclipse:clean && mvn eclipse:eclipse && mvn clean package && cd userinterface/ && mvn eclipse:clean && mvn eclipse:eclipse && mvn clean package*
    - Execute mvn clean package
    - Refresh userinterface module in eclipse
    - In eclipse, Search for Application.java and run the app.
    - To test, access via browser with url *localhost:8181*
  + Staging Profile Setup
    - Go to ors directory
    - Execute *mvn eclipse:clean && mvn eclipse:eclipse -P staging && mvn clean package -P staging && cd userinterface/ && mvn eclipse:clean && mvn eclipse:eclipse -P staging && mvn clean package -P staging*
    - Go to userinterface/target
    - Execute *mv userinterface-0.0.1-SNAPSHOT.war ROOT.war* (Note: the compiled war is changed to ROOT.war to override the ROOT directory that resides in tomcat
    - Use scp to transfer the ROOT.war to staging server (For staging setup, see Staging Setup Tomcat Server section)
* Querier
  + N/A
* Queue Gateway
  + Local Profile Setup
    - Using command *mvn clean package && mvn install*, execute to every directory in sequence
      * /queuegtw/file\_manager\_ejb/
      * /queuegtw/queuegtw\_ejb/

# Workflow

## Diagram

### New Features Workflow



### Updating/Revise Features Workflow



## Worflow Steps

* Report an Issue – this is when the tester or user reports that there is a defect in the system
* CRF / New Feature Specs – if there is an issuance of new Change Request Form or agreed upon new feature
* Create an Issue in Git – The developer will log a ticket to Git as a form of documentation. Go to Gitlab then log the issue under jasper-template
* Pull Master Branch – ensuring that the latest copy of master is in local repository. This is done through executing *git checkout master && git pull origin master* in your local branch
* Create a branch – for every changes whether its a revision or a new feature, it is a best practice to branch out the revision and not to directly alter to master branch
* Make Revision / Create and Develop feature – revise accordingly based on the specs / issue
* Deploy in Staging – Deployment of changes to staging (see Deploying an Existing Report in Server section)
* Merge branch in master – after the approval of the tester from testing in staging, the changes should be merged to master
* Push to Git – Push everything from local repository to Git repository
* Deploy in Production – Deployment to production (see Deploying an Existing Report in Server section)