Prerequisite: Geoserver

Geoserver is recommended for the project. It is used to locally test the generated WPS
http://docs.geoserver.org/latest/en/user/installation/index.html#installation
http://docs.geoserver.org/latest/en/user/installation/win_installer.html

- *GeoServer* installation manual: http://geoserver.org/download/
- during install, you have to provide
  - an Admin id and a Password, by default *admin* and *geoserver*
  - a port for request invocation, by default 8080
- WPS extension installation : http://docs.geoserver.org/stable/en/user/services/wps/install.html
- download the WPS extension : http://geoserver.org/release/stable/
- unzip the extension in the *GeoServerinstall*/`webapps/geoserver/WEB-INF/lib` repository:
  - unzip the WPS extension
  - add `gt-wps-17.2.jar` in the same repository, the version number (here 17.2) has to be the same as the version number of installed geotools jar in GeoServer
- Start the server, using the generated *Start GeoServer* in the *GeoServer Menu*
- Check that *WPS* is in the *Services* menu of *GeoServer*
- To test a deployed WPS, use your browser with the local address http://localhost:8080/geoserver
- Connection to the server using *admin / geoserver*
- To stop the server, use the generated *Stop GeoServer* in the GeoServer menu

Application and WPS design environment

- *https://github.com/jpbabau/Noumea*
- Download and unzip *EclipseWPS-dev*
- *EclipseWPS-dev* includes a *Win32 Neon* version of *Eclipse* (in the *EclipseNeon* folder), a workspace *wpsWorkspace* containing the source projects files, and a workspace *runtime-EclipseApplication* containing the WPS projects
- To start *EclipseWPS-dev*, double-click on *Eclipse*
  - Select the *wpsWorkspace* workspace
- You may modify the source code of the different projects
  - *wfwps*: EMF project containing the ecore metamodel (*wfwps.ecore*)
  - *wfwps.design*: Sirius project containing the graphical interface description (*wfwps.odesign*)
  - *wfwps.acceleo*: Acceleo project containing the code generation templates (*generate.mtl, generateLoacalWPS.mtl, generateWF.mtl*)
  - *NoumeaUI* : JavFX project containing the user interface

Application Running

- Select the *NoumeaUI* project
- Right-click *Run As / Eclipse Application*
  - Launch a new Eclipse instance

Create a WPS library

- In the *wfwpsLibrary* project create a wps library
  - *File/New/Otheré /Example EMF Model Creation Wizards/Wfwps Model*
  - *Next*
  - Select the *wfwpsLibrary* folder

  - Give a name to your lib *myLib.wfwps* (the extension wfwps is mandatory)
  - *Next*
  - *Model Object :* select *Workflow Wps*
  - *Finish*
- Select the *wfwpsLibrary* folder
  - Right click and *Viewpoints selection*
  - If not selected, click on *wfwps view point*

Create a WPS Java project by reusing the TemplateProject

- Copy/paste the *TemplateProject* modifying the new *projectName*
  - The project uses the jdk 1.8 (*Build Path/Configure Build Path*)
- Modify the following information in the *pom.xml* file

```
<groupId>yourGroupID</groupId>
<artifactId>NewProjectName</artifactId>
<version>VersionNumber</version>
<name>WPSjarName</name>
```

- Modify if necessary the version numbers in the *pom.xml* file

```
<geotools.version>17.2</geotools.version>
<geoserver.version>2.11.2</geoserver.version>
```

  - the version numbers have to be the same as the geoserver et geotools version numbers installed in your *Geoserver* installation
- add the domain-specific java code in a specific package in the *src/main/java* folder
  - you need to implement a non static *public* function *myFunction* member in a class *myClass.java*
  - the class has a public by-default constructor, with no parameter
  - the inputs and the output of the function must have the following types : *boolean*, *int*, *double*, *String*, *Geometry*, *FeatureCollection<SimpleFeatureType, SimpleFeature>*
  - the execution of the function is self-consistent, no extra code has to be executed before it

Edit models of WPS

- launch the Noumea User Interface
  - *My FX View* tab: click on the *Noumea Button*
  - *Configuration* tab
    - select your *Library* file *myLib.wfwps*
    - select your *Java Project*
  - *Modelling* tab
    - select your *class* file *myClass.java*
    - select your *function myFunction*
    - Click on *Modelling*
      - Gives a name for your WPS
- WPS java code generation
  - *Generation* tab
    - select your function in the *Local WPS List*
    - click on Generate WPS
      - Java code has been generated in src/main/java/

Edit models of workflow

- in *myLib.wfwps*, select the *Workflow Wps, right click New Representation / new Library Diagram*

- the model of java WPS are yet represented
- use the *Properties* tab to have access to the properties of modeled elements
- add workflow (*new Workflow* in the *Worflow Palette*)
  - add a *name* and an *abstract* to the workflow (*Properties* tab)
- add inputs and output to the workflow (*new Workflow Input* in the *Workflow Palette*)
  - add a *name* and an *abstract* for each (*Properties* tab)
  - select the correct type for each (*Properties* tab)
- double click on the workflow to open the corresponding graphical workflow editor
  - the inputs and outputs of the workflow are represented and cannot be deleted from this view
- import local WPS (*Local WPS Call* in the *Workflow Palette*)
  - select the corresponding local WPS by clicking on é (*Properties* tab)
  - double-click on the added *WPS Call* to add inputs and output
- import remote WPS (*Remote WPS Call* in the *Workflow Palette*)
  - select the corresponding remote WPS by clicking on é (*Properties* tab)
  - double-click on the added *WPS Call* to add inputs and output
- use the palette to add *WMS Call*, *WFS Call* and constant data (*Boolean Value*, *Integer Value*, *Double Value*, *String Value*)
  - add *name*, *abstract*, and properties through the *Properties* tab
- add links between elements (*new Link* in the *Workflow Palette*)
- To delete an element (WPS, link, constant, é)
  - *Right click on the element - Edit / Delete from Model*
- Validate the diagram before generation
  - *Right click on the diagram - Validate Diagram*
- WPS java code generation
  - *Configuration* tab
    - Re-select the *Library* file *myLib.wfwps*
  - *Generation* tab
    - select the workflow in the *Workflow List*
    - click on Generate WF
      - Java code has been generated in src/main/java/

WPS deployment

- Select the Java Project
  - *Right click -> Run As / Maven Build*
    - *Goals: package* (required for the first *Maven Build*)
  - *Run*
    - generation of the `WPSjarName-VersionNumber`.jar in the *projectName/target* folder
- Noumea User Interface
  - checks that the *Geoserver* project repository is set
  - *Configuration* tab
    - Select the *GeoServer Path*
      - The folder containing the Geoserver *bin* folder
  - *Deployment* tab
    - Just click on *Deploy*
      - stop *GeoServer*, copy the generated jar in the corresponding GeoServer folder *GeoServerinstall*/`webapps/geoserver/WEB-INF/lib` re-start *GeoServer*

WPS test

- you can test the deployed WPS with *GeoServer* using the local address http://localhost:8080/geoserver