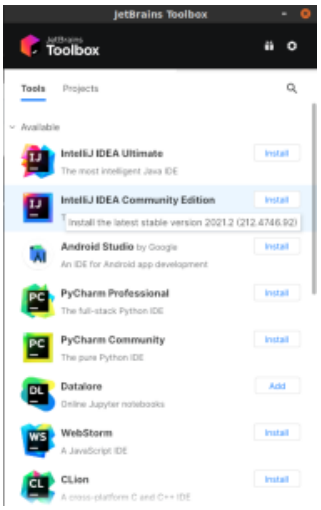
How to set up a Grouper development environment with IntelliJ

- Installation of IntelliJ (IDEA Community Edition)
- Simple Groovy project without Grouper source code
- Full development including Grouper project source code
    - Importing the Grouper project
    - Setting up Debugging
        - How to debug
            - Tomcat
            - GSH
            - Container process
    - (optional) Build the jar file artifacts
    - (optional) Set up a scripts module for personal GSH scripts

Installation of IntelliJ (IDEA Community Edition)

You can download IntelliJ IDEA directly from the JetBrains download page, or install JetBrains Toolbox, which makes it easy to upgrade.



Simple Groovy project without Grouper source code

In this simple project setup, Grouper is just added as a dependency, and the source code for it won't be available for debugging. Use this type of project if you just want to check the syntax of your scripts.

1) Create project

File New Project...

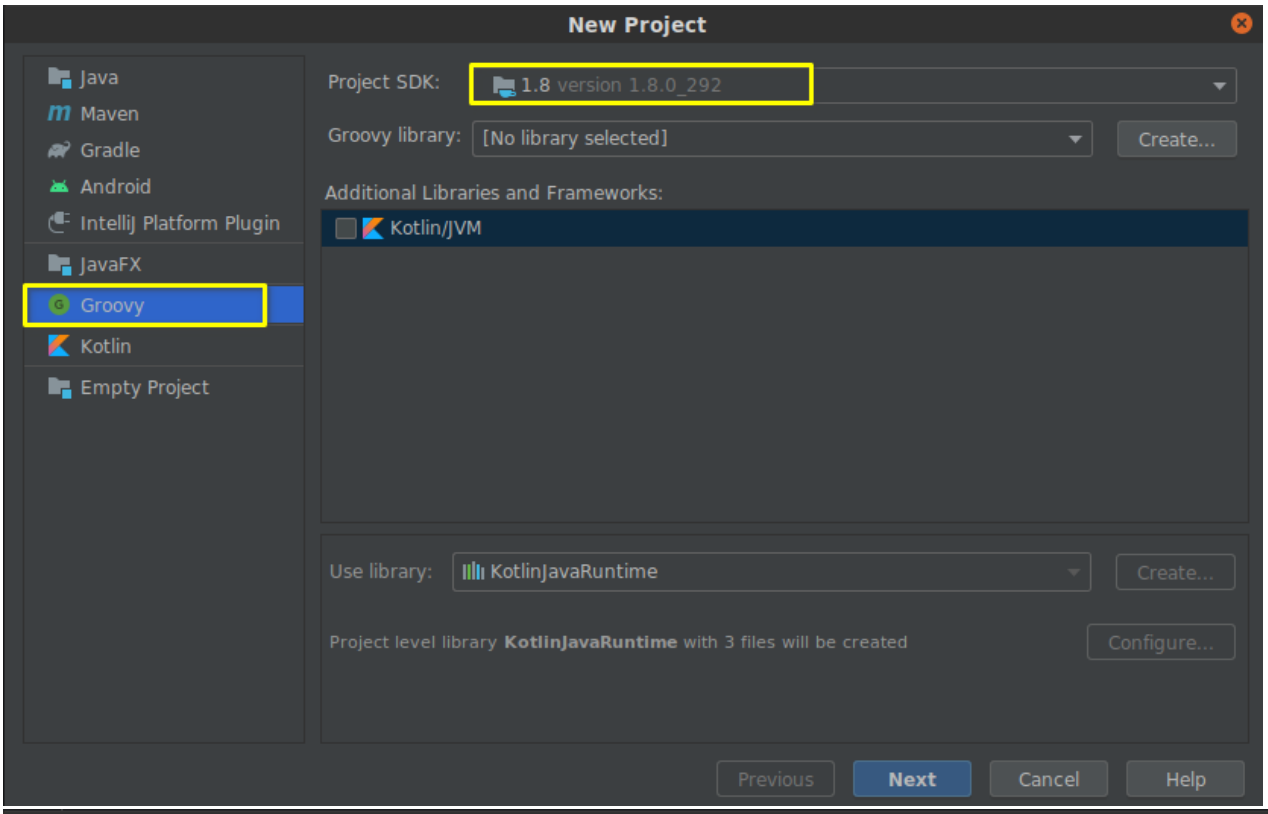
Type: Groovy

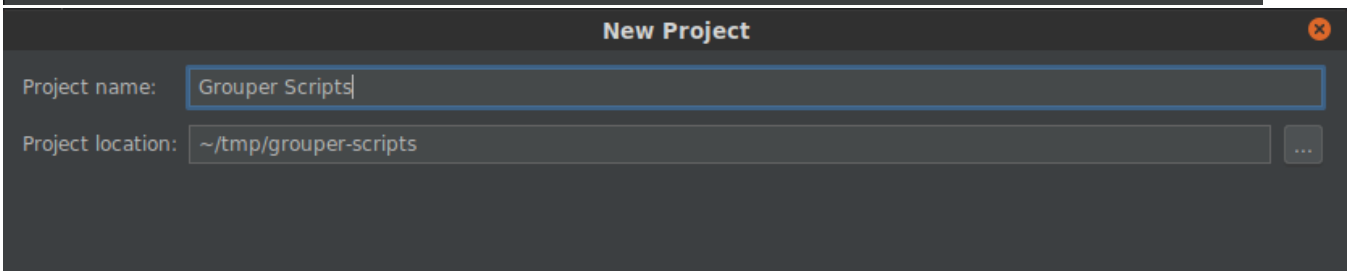
Project SDK: 1.8...

You shouldn't need to set the Groovy library, because there is one associated with the Grouper library (see step 2). But if you did want to manually add one, you can download from https://groovy.apache.org/, and unzip one of the SDK packages.

Next

Select project name and base directory anywhere you want it



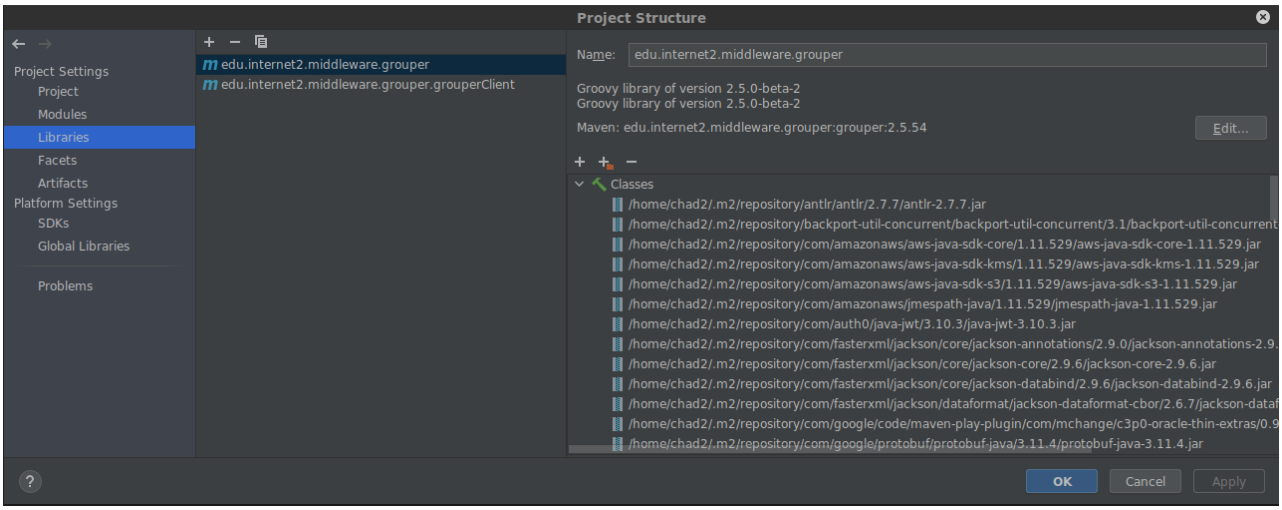


2) Add Grouper depedencies

File Project Structure Libraries Add From Maven...

Add these two dependencies (set version as desired). Don't worry if a search doesn't find it. Just hit enter after pasting in the strings

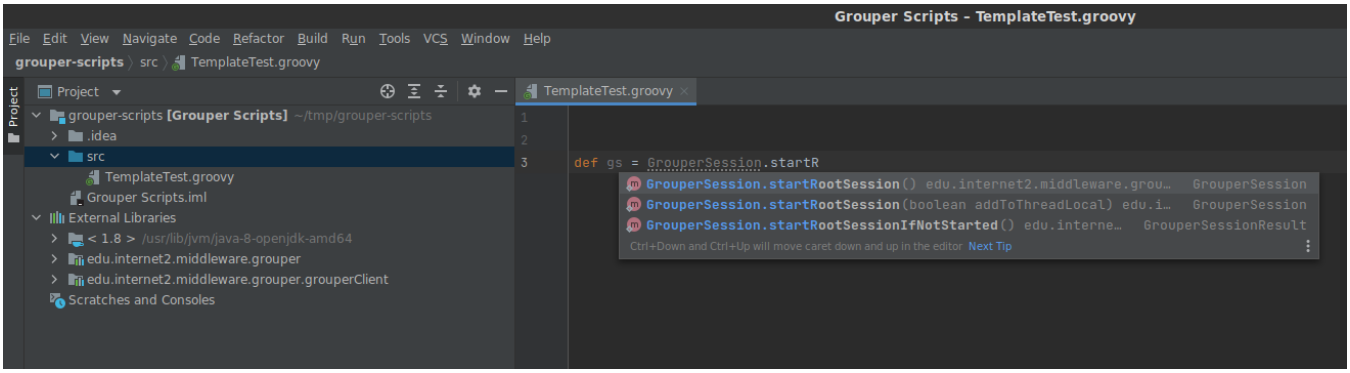
- edu.internet2.middleware.grouper:grouperClient:2.5.54
- edu.internet2.middleware.grouper:grouper:2.5.54



Note that adding the Grouper dependency also sets a Groovy library version 2.5.0-beta-2, or whatever version is currently a dependency of Grouper.

3) Create Groovy scripts

The module will automatically create a src/ folder. Right click on it, select New Groovy Script (if it fails in your version due to a bug in 2021.2 use Groovy Class instead). You can delete the boilerplate class definition, and just write code outside of a class.



Full development including Grouper project source code

Importing the Grouper project

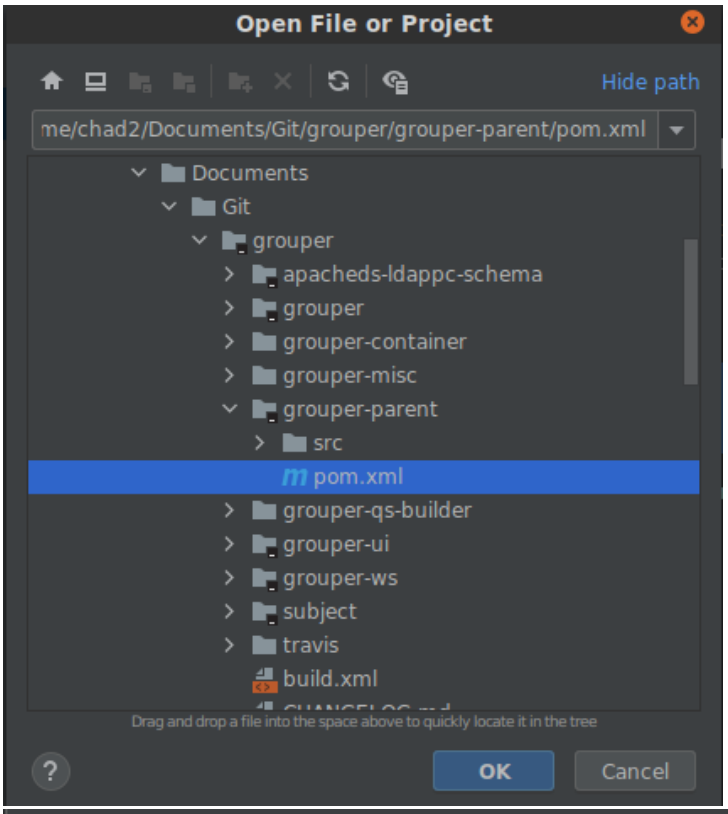
1) Download the Git repository from GitHub.

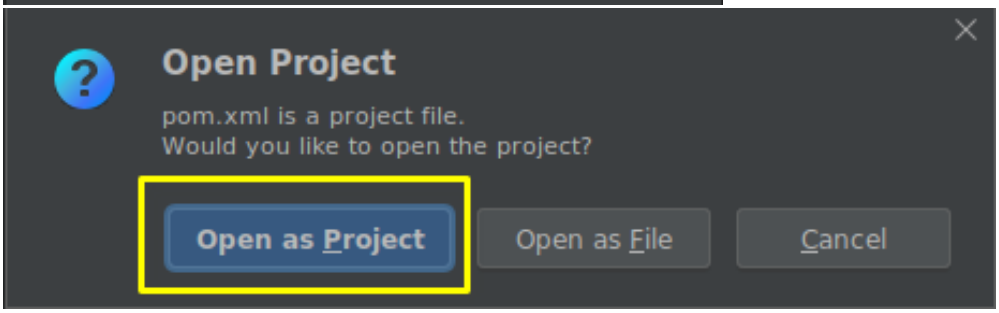
If you just need the current source and not the full history, you can add --depth 1 -b GROUPER_2_5_BRANCH which will just get the last commit on the 2.5 branch, and be a smaller download

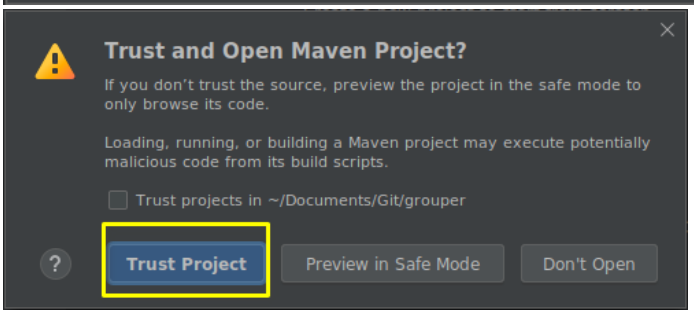
```
# $ git clone https://github.com/Internet2/grouper.git
#   OR
$ git clone --depth 1 -b GROUPER_2_5_BRANCH https://github.com/Internet2/grouper.git

Cloning into 'grouper'...
remote: Enumerating objects: 20992, done.
remote: Counting objects: 100% (20992/20992), done.
remote: Compressing objects: 100% (15795/15795), done.
remote: Total 20992 (delta 6768), reused 16398 (delta 4588), pack-reused 0
Receiving objects: 100% (20992/20992), 83.26 MiB | 2.83 MiB/s, done.
Resolving deltas: 100% (6768/6768), done.
Updating files: 100% (19449/19449), done.
```

2) Import as a Maven project in IDEA

(Projects Open Open File or Project browse for .../grouper/grouper-parent/pom.xml Open as Project Trust and Open Maven Project Trust Project

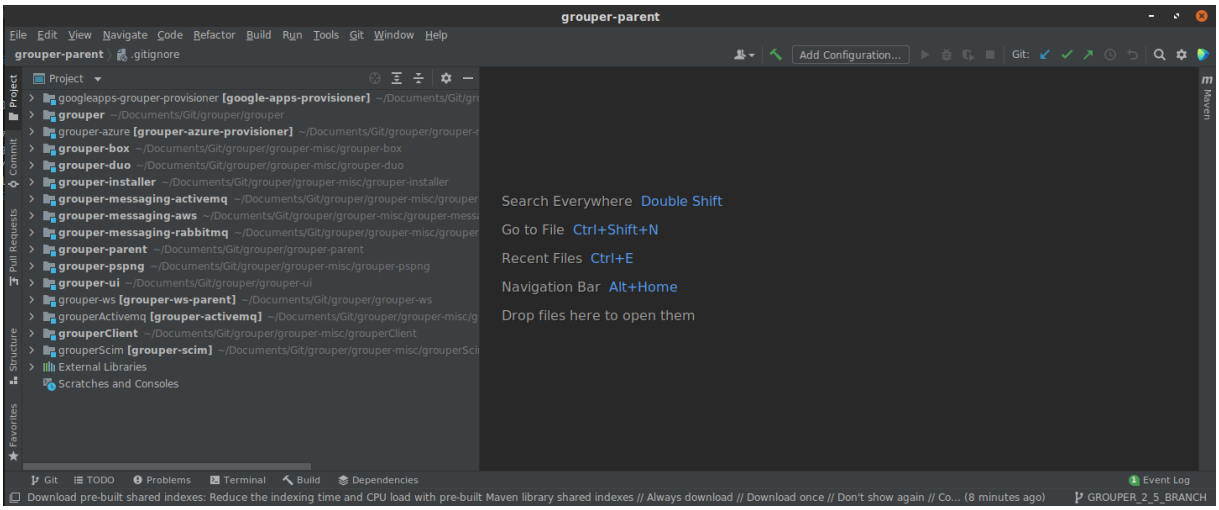






3) Wait a few minutes for it to download dependencies and index the project

This could take 10 minutes or more. While it's working, you won't see any project structure in the navigation pane. After it's done, you will see all the main Grouper projects.



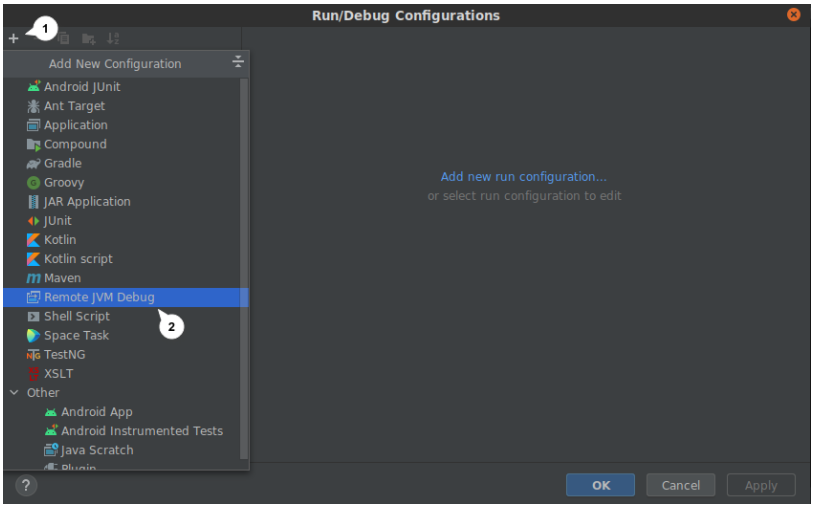
4) Set up language compatibility for Java 8

File Project Structure Project Project SDK = Java 8

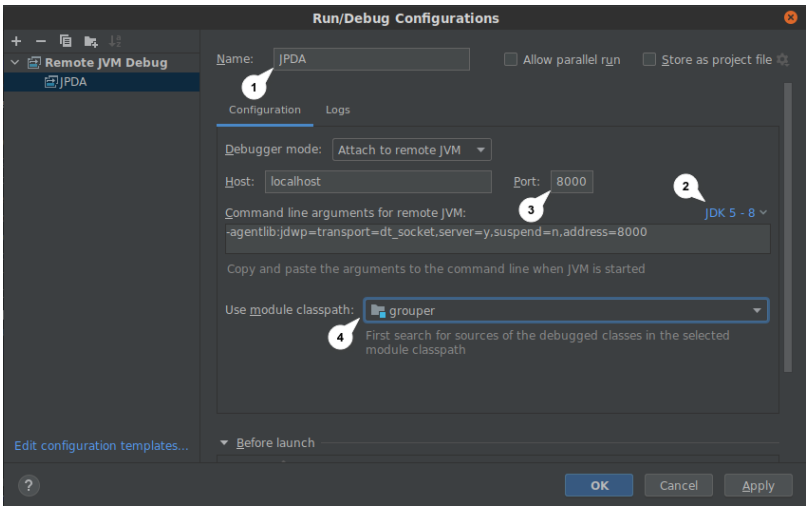
Setting up Debugging

1) Open wizard from menu Run Edit Configurations

2) Add a new "Remote JVM Debug"



3) Name it "JPDA" or anything else, choose JDK 5-8, set port to 8000 (this is the Tomcat debug default, so helps to match it), use module classpath grouper.



How to debug

Once a process is listening on port 8000, you can start debugging the "JPDA" configuration

Tomcat

Make sure you are using Java 8 in your environment, not a higher version (`java -version` to verify). Start debugging with `catalina jpda start`. There won't be any output except in the logs, but it will start listening, and then you can debug. Stop Tomcat with `catalina jpda stop`.

GSH

Make sure you are using Java 8 in your environment, not a higher version (`java -version` to verify). Get the extra Java arguments by editing the configuration and copying the `-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000` argument in the command line pane. Then start gsh with:

```
GSH_JVMARGS=-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000 ./gsh.sh
```

If you have something you want to breakpoint early in the startup, you can change suspend=n to suspend=y, which will wait for you to start debugging before starting GSH.

Container process

1) You need to enable port 8000 in your subimage for the port mapping to be available.

Dockerfile: `EXPOSE 8000`

2) Add these when running a container

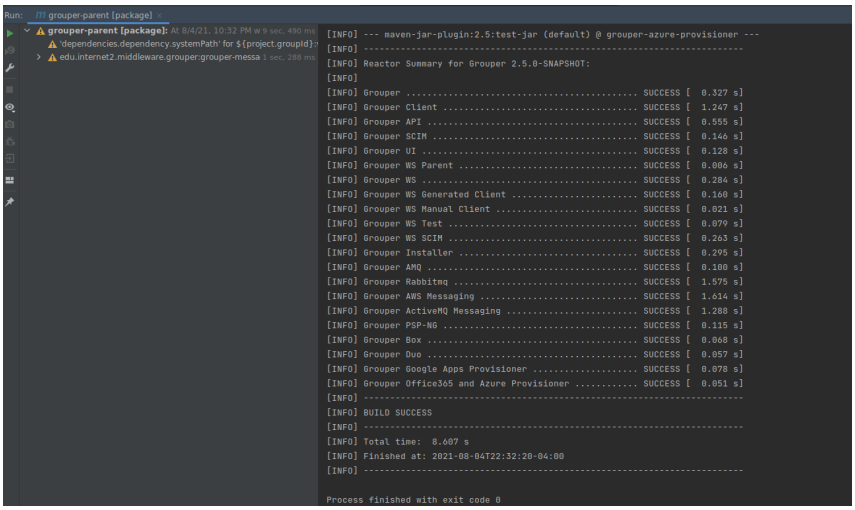
```
-e GROUPER_EXTRA_CATALINA_OPTS='-agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=8000'  -p 8000:8000
```

(optional) Build the jar file artifacts

1) On the right side of the app is a minimized Maven window. Click to expand it.

2) To builld all projects, select the *Grouper (root)* project, Licfecycle package , right click and select Run Maven Build

3) Progress will show in a terminal pane. When complete, all the jar files will be in their respective target/ subdirectories



(optional) Set up a scripts module for personal GSH scripts

You can create additional modules in Grouper that can live outside of the Grouper project. You can then create Groovy files that are aware of Grouper classes and can do type checking and context help.

1) Create the module

File New Module Groovy

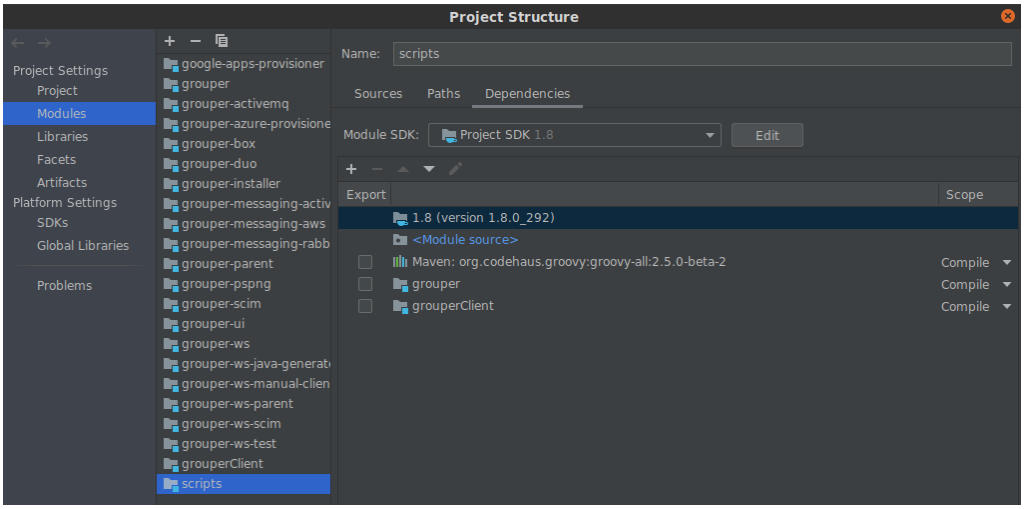
The default Module SDK should already be 1.8, and the Groovy library should be 2.5.0-beta-2

You can change the module name, and base location to anywhere, even outside of the Grouper project folder

2) Add dependencies

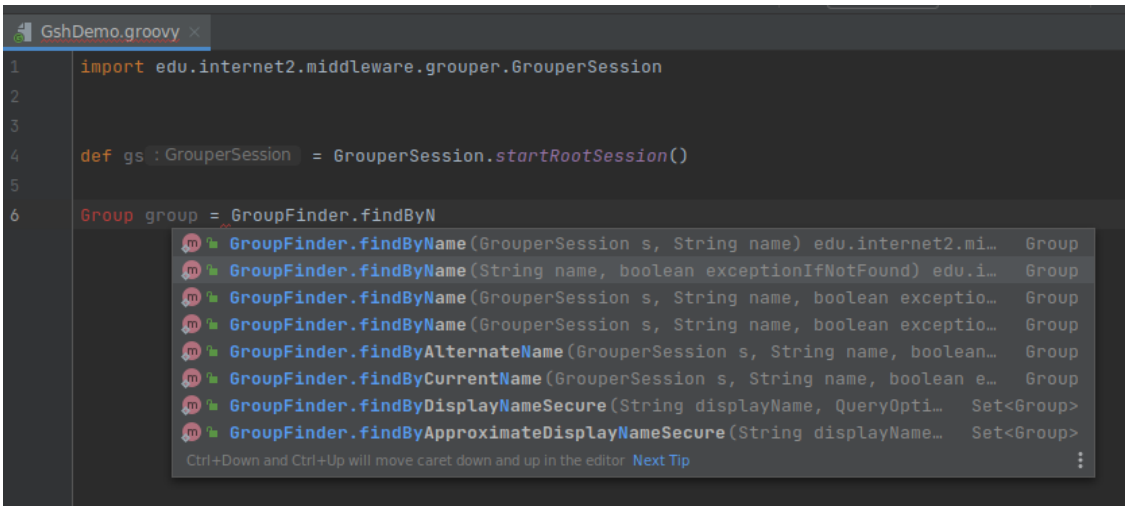
File Project Structure Modules (or your module name)

In the Dependencies tab, click on the Plus sign, add a Module dependency. Add two dependencies for grouper and grouperClient (and others if you wish)



3) Create a new script

The module will automatically create a src/ folder. Right click on it, select New Groovy Script (or Groovy Class and delete the boilerplate class definition, as a workaround if your version hasn't fixed this bug).



4) Run the script

There is probably a way to run this within the IDE, but I haven't tried it lately.