How To Setup a Grouper Development Environment for Grouper v2.4

This how-to describes how to set up a Grouper development environment so that you can code, test, and debug Grouper. This how-to is specific to Grouper 2.4. Instructions for Grouper 2.5 are over at How to Setup a Grouper Development Environment for Grouper v2.5.

The example commands and screenshots are from MacOS and Eclipse, and may vary slightly for different environments. However, the overall process should be similar on any modern operating system and development tool chain. Developers can use whatever tools that let them work most efficiently.

Prerequisites

Git for source code version control

1. Install Git
    a. Command line installs
        i. $ brew install git
        ii. https://github.com/fabriziocucci/git-bash-for-mac
        iii. Or Install from package https://git-scm.com/downloads
    b. Eclipse IDE plugin
        i. https://www.eclipse.org/egit/
    c. Github Desktop is also handy

Java - Grouper runs on Java

1. Install OpenJDK 8 or above. Grouper runs on Java.
    a. https://aws.amazon.com/corretto/
2. Note: do not use Java language features above Java 8 for most of Grouper - grouper, grouper-ui, grouper-ws, etc. The grouperClient code must be compliant with Java 6.

Apache Tomcat - Grouper runs in Tomcat

1. Download and unpack Tomcat 8.5.x
    a. https://tomcat.apache.org/download-80.cgi

Docker Desktop - to run our development database

1. Install Docker Desktop. We'll use this to run our development database.
    a. https://www.docker.com/products/docker-desktop

Eclipse - Grouper development happens in Eclipse (or your favorite IDE)

1. Install Eclipse IDE for Enterprise Java Developers or similar IDE
    a. https://www.eclipse.org/downloads/packages/

Clone the Grouper Source Repository

The Grouper source code repository is managed in GitHub at https://github.com/Internet2/grouper.

1. git clone --branch GROUPER_2_4_BRANCH https://github.com/Internet2/grouper.git grouper-2.4

```
thompsow@thompsow-mbpro ~/src
$ git clone --branch GROUPER_2_4_BRANCH https://github.com/Internet2/grouper.git grouper-2.4
Cloning into 'grouper-2.4'...
remote: Enumerating objects: 553, done.
remote: Counting objects: 100% (553/553), done.
remote: Compressing objects: 100% (378/378), done.
remote: Total 4529245 (delta 237), reused 335 (delta 57), pack-reused 4528692
Receiving objects: 100% (4529245/4529245), 2.42 GiB | 4.96 MiB/s, done.
Resolving deltas: 100% (4125267/4125267), done.
Updating files: 100% (18836/18836), done.
thompsow@thompsow-mbpro ~/src
$ cd grouper-2.4/
thompsow@thompsow-mbpro ~/src/grouper-2.4 (GROUPER_2_4_BRANCH)
$ ls
CHANGELOG.md            build.xml              grouper-qs-builder     travis
LICENSE.txt             grouper                grouper-ui
README.md               grouper-misc           grouper-ws
apacheds-ldappc-schema  grouper-parent         subject
thompsow@thompsow-mbpro ~/src/grouper-2.4 (GROUPER_2_4_BRANCH)
$
```

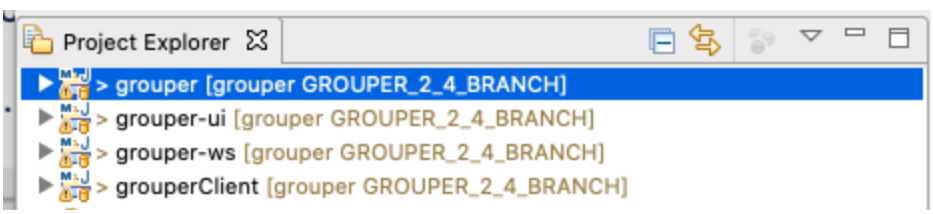
Note: The build strategy changed for the 2.5 release. Switching between 2.4 and 2.5 branches in the same directory and workspace when developing is not recommended. Instead, keep 2.4 and 2.5 work in separate local directories each with their git repository and their own eclipse workspace. For 2.5 development see How to Setup a Grouper Development Environment for Grouper v2.5.

Import Projects into Eclipse

Start a new eclipse workspace and import grouper modules as individual projects. Project will import as Maven projects and automatically download the required Maven dependencies. The example commands below assume the git repository was cloned to the local directory '/Users/thompsow/src/grouper-2.4/'.

1. Make a variable for the root of the git repo called GIT_ROOT: e.g. for /Users/thompsow/src/grouper-2.4
2. Import grouperClient
    a. Eclipse -> File -> Open Projects for File System or Archive
    b. Import source: ${GIT_ROOT}/grouper-misc/grouperClient
3. Import the grouper
    a. Eclipse -> File -> Open Projects for File System or Archive
    b. Import source: /Users/thompsow/src/grouper-2.4/grouper
4. Import grouper-ui
    a. Eclipse -> File -> Open Projects for File System or Archive
    b. Import source: /Users/thompsow/src/grouper-2.4/grouper-ui
5. Import grouper-ws
    a. Eclipse -> File -> Open Projects for File System or Archive
    b. Import source: /Users/thompsow/src/grouper-2.4/grouper-ws/grouper-ws

All the project should now be open and compiled.



Link source and conf to grouper project

We want to be able to run and debug the Grouper so that it picks up client source changes as you develop.

1. Add dependent source and configuration folders to the grouper-ui Java Build Path
    a. Add grouperClient src
        i. Click 'Link Source…'
        ii. Click 'Variables…'
        iii. Select GROUPER_REPO
        iv. Click 'Extend…'
        v. Select 'grouper-misc/grouperClient/src/java
        vi. Folder name: grouperClient-src
    b. Add grouperClient conf
        i. Click 'Link Source…'
        ii. Click 'Variables…'
        iii. Select GROUPER_REPO
        iv. Click 'Extend…'
        v. Select 'grouper-misc/grouperClient/conf'

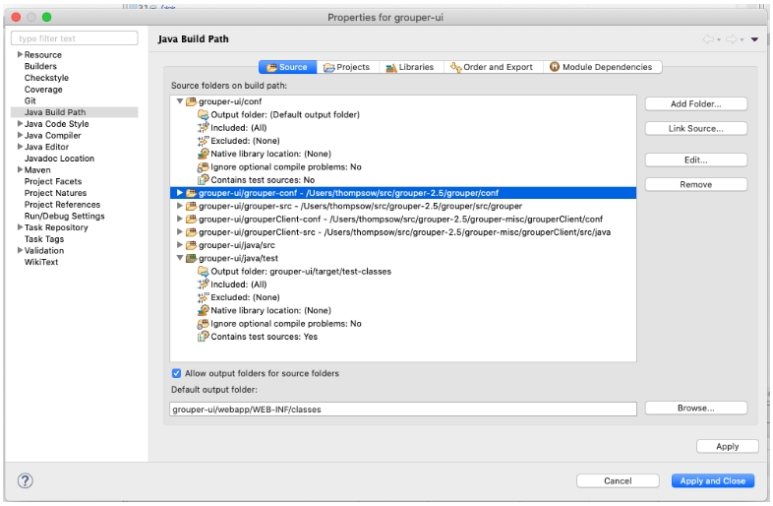
vi. Folder name: grouperClient-conf

Link source and conf to grouper-ui project

We want to be able to run and debug the Grouper UI from the grouper-ui/webapp folder, so that we can work on webapp artifacts (JSPs, etc), and at the same time update Java code in the grouper project and other code locations. To do this we will update the Java Build Path output folder so that compiled classes and other artifacts go to the right directories under grouper-ui/webapp. We will also add some dependent source and library folders to the grouper-ui Java Build Path.

1. grouper-ui -> File -> Properties -> Java Build Path -> Source tab
    a. Set 'Default output folder:' to: grouper-ui/webapp/WEB-INF/classes (you may have to create this directory)
2. Add dependent source and configuration folders to the grouper-ui Java Build Path
    a. Add grouper conf
        i. Click 'Link Source…'
        ii. Click 'Variables…'
        iii. Click 'New…'
            1. Name: GROUPER_REPO
            2. Click 'Folder…' and select the root repo folder
        iv. Select GROUPER_REPO
        v. Click 'Extend…'
        vi. Select 'grouper/conf'
        vii. Folder name: grouper-conf
    b. Add grouper src
        i. Click 'Link Source…'
        ii. Click 'Variables…'
        iii. Select GROUPER_REPO
        iv. Click 'Extend…'
        v. Select 'grouper/src/grouper'
        vi. Folder name: grouper-src
    c. Add grouperClient src
        i. Click 'Link Source…'
        ii. Click 'Variables…'
        iii. Select GROUPER_REPO
        iv. Click 'Extend…'
        v. Select 'grouper-misc/grouperClient/src/java
        vi. Folder name: grouperClient-src
    d. Add grouperClient conf
        i. Click 'Link Source…'
        ii. Click 'Variables…'
        iii. Select GROUPER_REPO
        iv. Click 'Extend…'
        v. Select 'grouper-misc/grouperClient/conf'
        vi. Folder name: grouperClient-conf
3. Make sure all folders except 'grouper-ui/java/test' are using the default output folder
4. Make sure grouper-ui/conf has 'Excluded:' set to (None)
5. Remove srcPoc and misc source folders (these are not needed)

The grouper-ui Java Build Path should now look something like this:



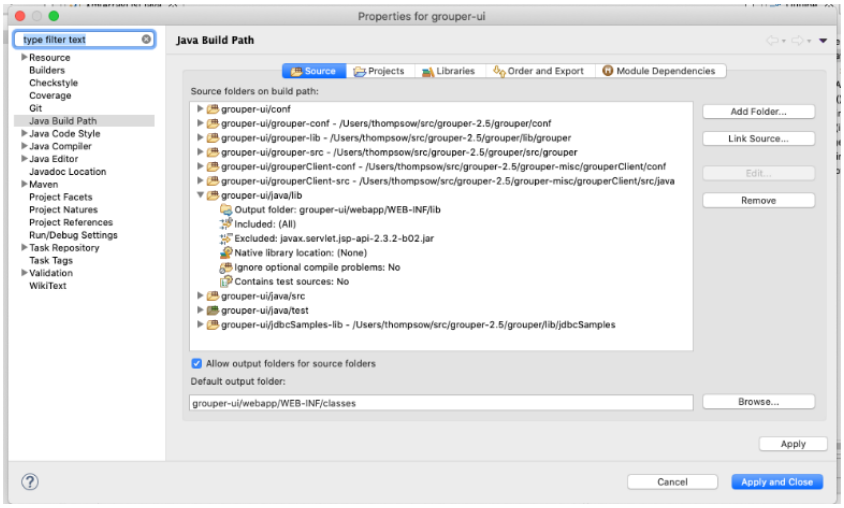
Add runtime jars to grouper-ui Java Build Path

We need to pull dependent jar files into grouper-ui/webapp/WEB-INF/lib directory. We'll do this by adding three more "source" folders to the grouper-ui Java Build Path.

1. grouper-ui -> File -> Properties -> Java Build Path

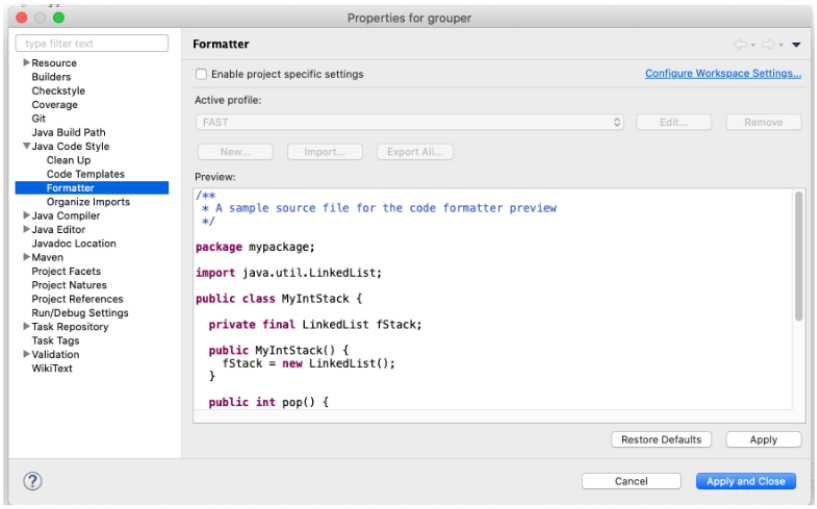
a. Add grouper jars
    i. Link Source…
        1. Linked folder location: GROUPER_REPO/grouper/lib/grouper
        2. Folder name: grouper-lib
b. Configure grouper-lib source to output to path relative to ‘grouper-ui’
    i. Check “Allow output folders for source folders
    ii. Click “Output folder” on grouper-lib
    iii. Select ‘Specific output folder (path relative to ‘grouper-ui’)
        1. create/select ‘webapp/WEB-INF/lib’
c. Add jdbc sample jars
    i. Link Source…
        1. Linked folder location: GROUPER_REPO/grouper/lib/jdbcSamples
        2. Folder name: jdbcSamples-lib
        3. Configure jdbcSamples-lib source to output to path relative to ‘grouper-ui’
            a. webapp/WEB-INF/lib
a. Add grouper-ui jars
    i. Add Folder…
        1. Select java/lib
        2. Check “Allow output folder for source folders
        3. Configure grouper-lib source to output to path relative to ‘grouper-ui’
            a. webapp/WEB-INF/lib
        4. Add javax.servlet.jsp-api.{version}.jar to Exclusion patterns

The grouper-ui Java Build Path should now look something like this:



Configure Eclipse Code Formatter

1. Eclipse -> Preferences -> Java Code Style -> Formatter
2. Import… (navigate and select ../grouper-2.4/grouper/misc/eclipse/fastFormat.xml)
3. Apply and close



Development Database

Multiple databases are supported including Oracle, mySQL, and PostgreSQL. We'll use PostgreSQL for this how-to. The steps for other databases would be similar.

Start the development database

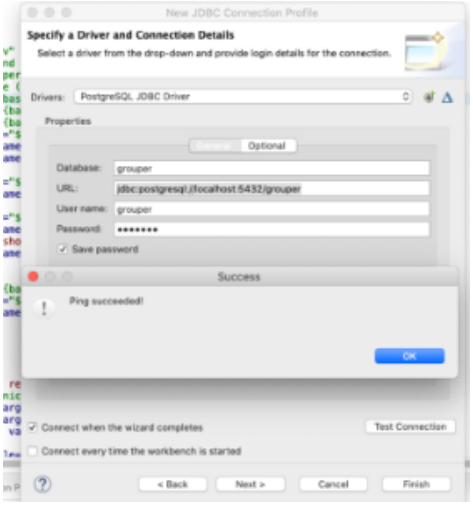
> **Running Postgres in Docker on a Windows Host**
>
> Postgres' docker image runs the database as an unprivileged user *postgres.* The container's startup script attempts to chown /var/lib/postgresql /data folder to this user. In Docker Desktop for Windows, this causes the script to throw an error and exit. Bind volumes cannot easily have their ownership changed from within the container running on a Windows host. The workaround for this is creating a named volume instead.

We will run postgres with a mounted external volume to preserve data between docker container restarts.

1. Create a named volume 'docker create volume grouper-postgres'
2. Run 'docker run --name grouperdb -e POSTGRES_PASSWORD=grouper -e POSTGRES_USER=grouper -d -p 5432:5432 -v grouper-postgres: /var/lib/postgresql/data postgres'

Connect to development database in Eclipse

1. Window -> Perspective -> Open Perspective -> Other -> Database Development -> Open
2. Right click 'Database Connections' and create PostgreSQL connection profile
3. Add the postgess JDBC driver found in under ../grouper/grouper-ui/webapp/WEB-INF/lib/
4. Enter the connection details and then click Test Connection



Configure minimum properties files for development

Create and configure the grouper-hibernate.properties for postgres

1. cd ../grouper-2.4/grouper/conf
2. cp ../misc/grouper.hibernate.example.properties grouper.hibernate.properties
3. Edit ../grouper/grouper/conf/grouper.hibernate.properties

```
28 ##########################################
29 ## DB settings
30 ##########################################
31
32 # e.g. mysql:           jdbc:mysql://localhost:3306/grouper
33 # e.g. p6spy (log sql): [use the URL that your DB requires]
34 # e.g. oracle:          jdbc:oracle:thin:@server.school.edu:1521:sid
35 # e.g. hsqldb (a):      jdbc:hsqldb:dist/run/grouper;create=true
36 # e.g. hsqldb (b):      jdbc:hsqldb:hsql://localhost:9001/grouper
37 # e.g. postgres:        jdbc:postgresql://localhost:5432/database
38 # e.g. mssql:           jdbc:sqlserver://localhost:3280;databaseName=grouper
39 hibernate.connection.url = jdbc:postgresql://localhost:5432/grouper
40
41 hibernate.connection.username         = grouper
42 # If you are using an empty password, depending upon your version of
43 # Java and Ant you may need to specify a password of "".
44 # Note: you can keep passwords external and encrypted: https://bugs.internet2.edu/jira/browse/GRP-122
45 hibernate.connection.password         = grouper
46
```

Create and configure morphString.properties

1. cd .../grouper-2.4/grouper/conf
2. cp .../misc/morphString.example.properties morphString.properties
3. Edit .../grouper/grouper/conf/morphString.properties
4. encrypt.key = not_a_random_key

Create remaining required properties files

All of the *.base.properties in grouper/conf need to be copied to *.properties files in the same directory.

Also need to copy .../grouper/conf/grouperText/grouper.textNg.en.us.base.properties to grouperText/grouper.text.en.us.properties.

```
conf — -bash — 104×23
thompsow@thompsow-mbpro ~/src/grouper-2.4/grouper/conf (GROUPER_2_4_BRANCH)
$ ls
GSHFileLoad.properties          grouper.base.properties          grouperUtf8.txt
README.txt                      grouper.cache.base.properties    log4j.example.properties
ddl                             grouper.cache.properties         log4j.properties
groovysh.profile                grouper.client.base.properties   morphString.base.properties
grouper-loader.base.properties  grouper.client.properties        morphString.properties
grouper-loader.properties       grouper.hibernate.base.properties server.example.properties
grouper-ui-ng.base.properties   grouper.hibernate.properties      spy.example.properties
grouper-ui.properties           grouper.properties               sqltool.rc
grouper-ws-ng.base.properties   grouperRulesEmailTemplates       subject.base.properties
grouper-ws.properties           grouperText                      subject.properties
```

Bootstrap the Grouper Database

For development purposes, we'll bootstrap the Grouper database, add sample subjects, and reset the database using a few Java classes.

Run GrouperShell from Eclipse to initialize the Grouper database:

1. Right click GrouperShell in the Grouper project explorer
2. Select 'Run as' and then 'Run Configurations…'
3. Name: GrouperShell -registry -runscript -noprompt
4. (x) = Arguments tab
    a. Program arguments: -registry -runscript -noprompt
5. Classpath tab
    a. Highlight User Entries and click 'Advanced…'
    a. Add the 'conf' folder to the classpath for the properties files
6. Click 'Apply'
7. Click 'Run'

Run GrouperShell from Eclipse to check the Grouper database:

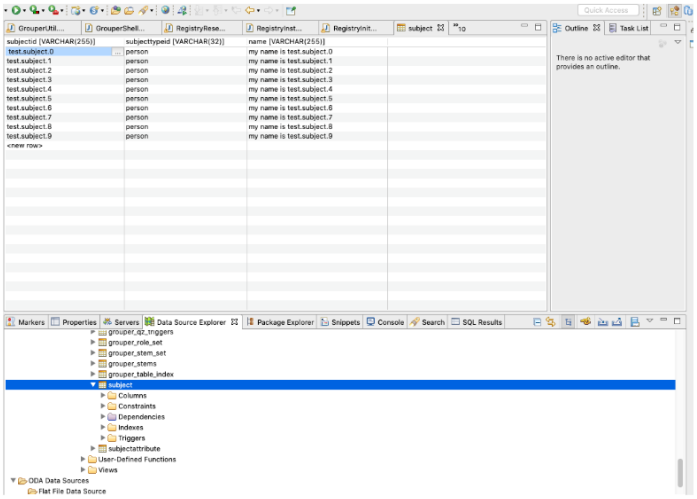
1. Right click GrouperShell in grouper project explorer
2. Select 'Run as' and then 'Run Configurations…'
    a. Name: GrouperShell -registry -check -noprompt

3. (x) = Arguments tab
    a. -registry -check -noprompt
    a. Program arguments:
4. Classpath tab
    a. Highlight User Entries and click ‘Advanced…’
    b. Add the ‘conf’ folder to the classpath
5. Click ‘Apply’
6. Click ‘Run’

Run RegistryReset with ‘addSubjects’ as an argument to add sample subjects:

1. Right click RegistryReset in grouper project explorer
2. Select ‘Run as’ and then ‘Run Configurations…’
    a. Name: RegistryReset addSubjects
3. (x) = Arguments tab
    a. Program arguments:
        i. addSubjects
4. Classpath tab
    a. Highlight User Entries and click ‘Advanced…’
    b. Add the ‘conf’ folder to the classpath
    c. Click ‘Apply’
5. Click ‘Run’

Query the subjects table from the Eclipse Data Source Explorer to see the added subjects:

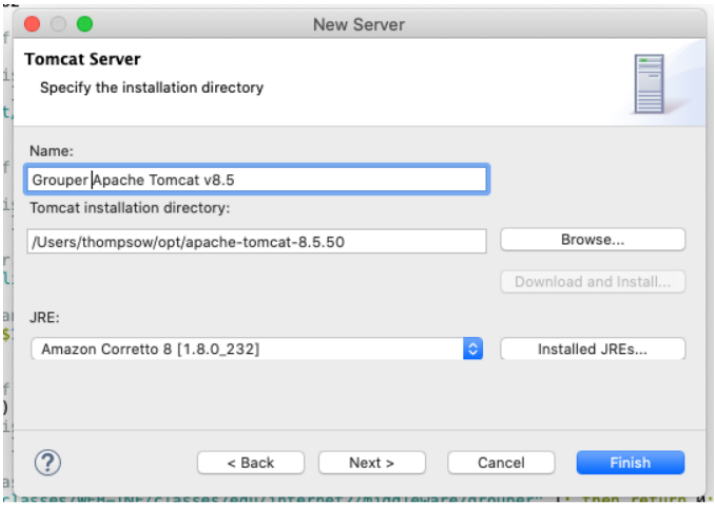


Run grouper-ui in Eclipse with Tomcat

Now that we have a Grouper database and some test subjects, the next step is to add the grouper-ui/webapp directory to the Eclipse Tomcat launcher so we can run and debug the grouper-ui.

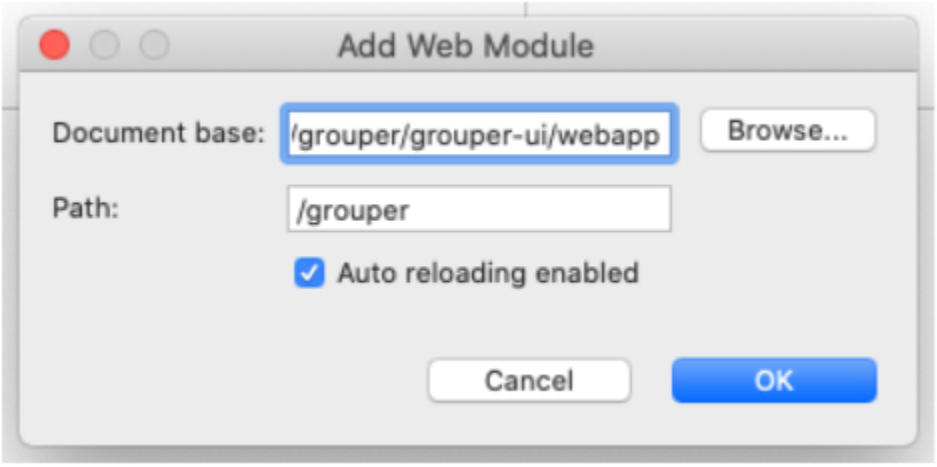
Add Tomcat server to Eclipse:

1. Eclipse -> J2EE Perspective -> Servers Tab
2. Click “No servers are available. Click this link to create a new server…”
    a. Select 'Tomcat v8.5'
    b. Select your tomcat installation directory
    c. Select your JRE



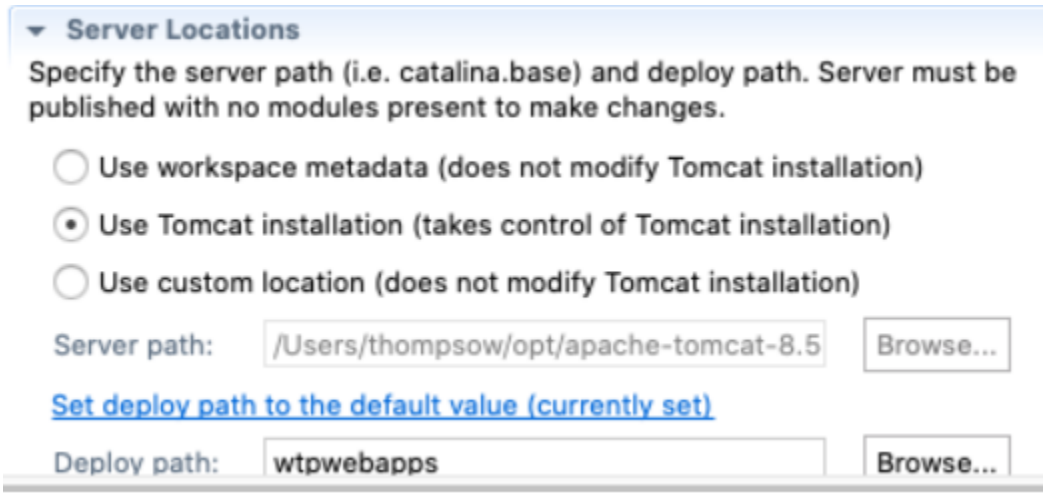
Add grouper-ui web module to Tomcat Server

1. Double click on Tomcat v8.5 at localhost [Stopped, Republish] to access configuration panel
2. Click on 'Modules' tab, click 'Add External Web Module…"



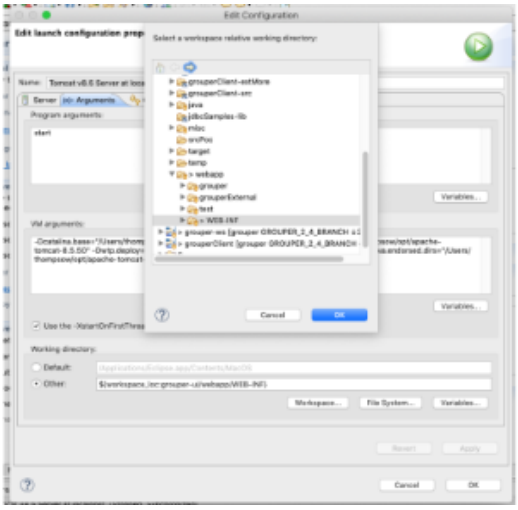
Configure Server Location

1. Select 'Use Tomcat installation (takes control Tomcat installation)



Configure Tomcat Server Working Directory to direct Grouper logs

1. Servers tab -> Overview -> click on “Open launch configuration”
2. (x)= Arguments tab
3. Working directory:
    a. Select grouper-ui/webapp/WEB-INF
    a. Choose Other
    b. Click Workspace…



Grouper logs will now show up under ../grouper-ui/webapp/WEB-INF/logs

Create Grouper Users in tomcat-user.xml

Add GrouperSystem and test.subject.0 through 9 to tomcat-user.xml

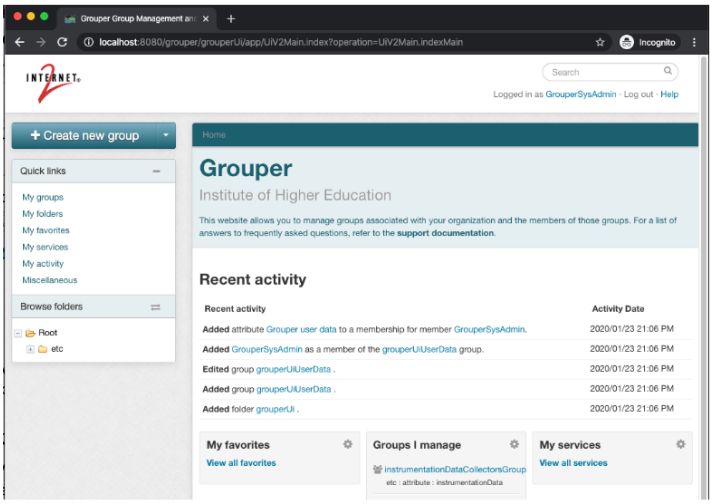
1. Add the following to ../apache-tomcat-8.5.50/conf/tomcat-users.xml

```
<role rolename="grouper_user"/>
<user username="GrouperSystem" password="" roles="grouper_user"/>
<user username="test.subject.0" password="" roles="grouper_user"/>
<user username="test.subject.1" password="" roles="grouper_user"/>
<user username="test.subject.2" password="" roles="grouper_user"/>
<user username="test.subject.3" password="" roles="grouper_user"/>
<user username="test.subject.4" password="" roles="grouper_user"/>
<user username="test.subject.5" password="" roles="grouper_user"/>
<user username="test.subject.6" password="" roles="grouper_user"/>
<user username="test.subject.7" password="" roles="grouper_user"/>
<user username="test.subject.8" password="" roles="grouper_user"/>
<user username="test.subject.9" password="" roles="grouper_user"/>
```

Run Grouper from Eclipse

Start Tomcat from Eclipse by selecting the server under the Servers tab and clicking the green ‘Run’ button.

Grouper UI should be available at http://localhost:8080/grouper. You should be able to log in with GrouperSystem or any of the test subjects and no password.



Debug Grouper from Eclipse

1. Set a breakpoint in UiV2Main.java line #114
2. Start the server in debug mode
3. Login and try to search in the Grouper UI
    a. This should hit the breakpoint in Eclipse and allow you step through the code.

Congrats! You now have a working Grouper development environment. Now go check out Grouper developers coding standards and then pick up some JIRAs!