



# **Behavioral Health Information Technology and Standards (BHITS) Project Consent2Share Version 3.4.0 Development Guide**

**Sep 2017**

**Prepared by FEi Systems**

## Contents

INTRODUCTION.....	4
ROADMAP.....	5
<b>1. JAVA DEVELOPMENT ENVIRONMENT SETUP .....</b>	<b>6</b>
1.1 Java JDK .....	6
1.1.1 Install Java JDK .....	6
1.1.2 Set JAVA_HOME System Environment Variable and System Path.....	6
Set Environment Variables .....	6
1.1.3 Install Java Cryptography Extension (JCE) .....	8
1.2 Maven .....	8
1.2.1 Install Maven.....	8
1.2.2 Set M2_HOME System Environment Variable and System Path .....	8
1.3 Tomcat 8.....	9
1.3.1 Install Tomcat.....	9
1.3.2 Configure JVM and other options for Tomcat.....	9
1.3.3 Configure User to operate the "/manager/html" web application.....	9
1.4 MySql Server and Workbench .....	9
1.4.1 Install.....	9
1.4.2 Set up system environment variable and system path .....	9
1.5 Install Git for Windows .....	9
1.5.1 Create a Github Account Linking to Your FEI Email Address .....	9
1.6 Gradle.....	10
1.6.1 Install.....	10
1.6.2 Set GRADLE_HOME system environment variable and system path.....	10
1.6.3 The Gradle Wrapper.....	10
1.7 Install Flyway (Optional) .....	10
1.7.1 Prerequisites .....	10
1.7.2 Windows Installation: .....	10
<b>2 FRONTEND DEVELOPMENT ENVIRONMENT SETUP .....</b>	<b>12</b>
2.1 Install Platform - Node.js .....	12
2.2 Install Angular CLI .....	12
<b>3 INSTALL AND CONFIGURE INTELLIJ IDEA FOR CONSENT2SHARE V3 DEVELOPMENT.....</b>	<b>13</b>
3.1 Install .....	13
3.1.1 Configure/Update JDK.....	13
3.1.2 Configure Maven.....	15
3.1.3 Set up Tomcat .....	15
3.1.4 Configure Github .....	16
3.1.5 Configure for Frontend Development.....	16
3.1.6 Configure Default Settings for File Template .....	17

3.1.7	Multirun Plugin .....	17
3.1.8	Configure Settings for Importing Java Classes .....	17
3.1.9	Install Lombok Plugin .....	18
<b>4</b>	<b>LOAD AND DEVELOP CONSENT2SHARE V3 PROJECTS IN INTELLIJ IDEA .....</b>	<b>19</b>
4.1	Open Multiple Projects in the Same Window in IntelliJ IDEA .....	19
4.1.1	Create an Empty Project .....	19
4.1.2	Clone Project Git Repositories .....	20
4.1.3	Import C2S Projects as Modules in IntelliJ IDEA.....	21
4.2	Create Schemas in MySQL Workbench .....	25
4.3	Run/Debug Source Code .....	25
4.3.1	Run/Debug Configuration with Tomcat for UAA.....	25
4.3.2	Run/Debug Configuration with NPM for UI Projects .....	30
4.3.3	Use Multirun Plugin to Run Consent2Share.....	31
4.3.4	Add Environment Variables .....	33
4.3.5	Update Configuration for EdgeServerApplication.....	34
4.3.6	common-libraries Artifacts .....	34
4.3.7	Run Consent2Share Applications .....	34
4.3.8	Run SQL Scripts to Insert Data .....	35
<b>5</b>	<b>REFERENCES .....</b>	<b>36</b>



## Introduction

The main goal of this development guide is to provide clear direction to set up the Consent2Share application on Consent2Share developer's machine so that we can focus on coding quickly and not focus on setting up the project. This is a revised version of *Consent2Share Version 2 Development Guide*.

And if there is no explicit mention for the development environment operating system (OS), it is Windows 10 by default.



## Roadmap

To describe the development environment setup, this development guide is organized through the following chapters:

Chapter 1 describes steps to set up the Java (backend) development environment.

Chapter 2 describes steps to set up the frontend development environment.

Chapter 3 describes how to install and configure IntelliJ IDEA for Consent2Share V3 development.

Chapter 4 focuses on loading and developing Consent2Share V3 in IntelliJ IDEA.

(Note: If you are familiar with the *Consent2Share Version 2 Development Guide*, you can ignore Chapter 1 and Chapter 3.)

## 1. Java Development Environment Setup

The following development environment setups should be carried out in the following order.

### 1.1 Java JDK

Java 8 is used for Consent2Share development.

#### 1.1.1 Install Java JDK

Go to the [Oracle JDK download site](#) to download the appropriate JDK for your operating systems.

#### 1.1.2 Set JAVA\_HOME System Environment Variable and System Path

After you have installed Java 8, java.exe could come from three places,

```
C:\Users\tao.lin>where java
C:\Program Files\Java\jdk1.8.0_51\bin\java.exe
C:\ProgramData\Oracle\Java\javapath\java.exe
C:\Windows\System32\java.exe
```

In order to control where java.exe should be from, we need to do the following:

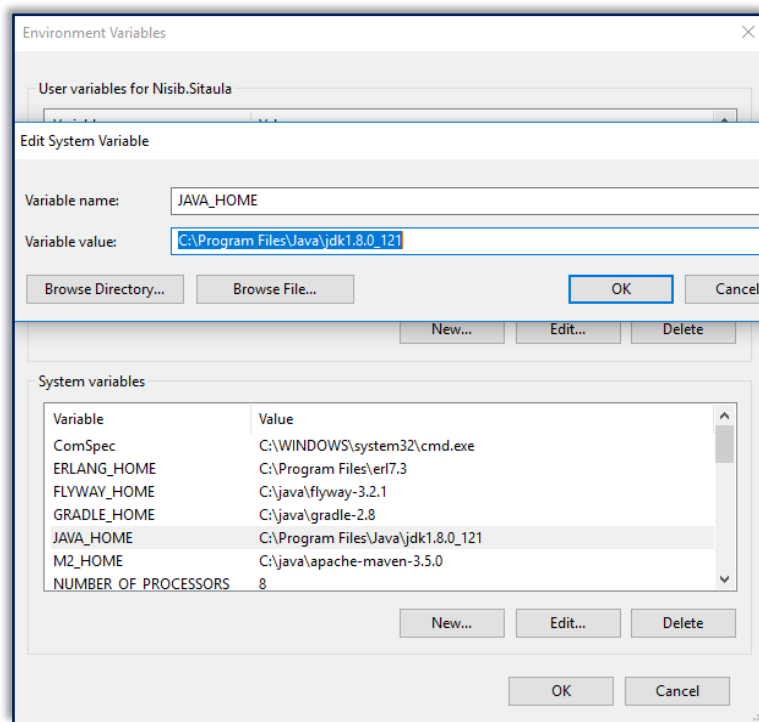
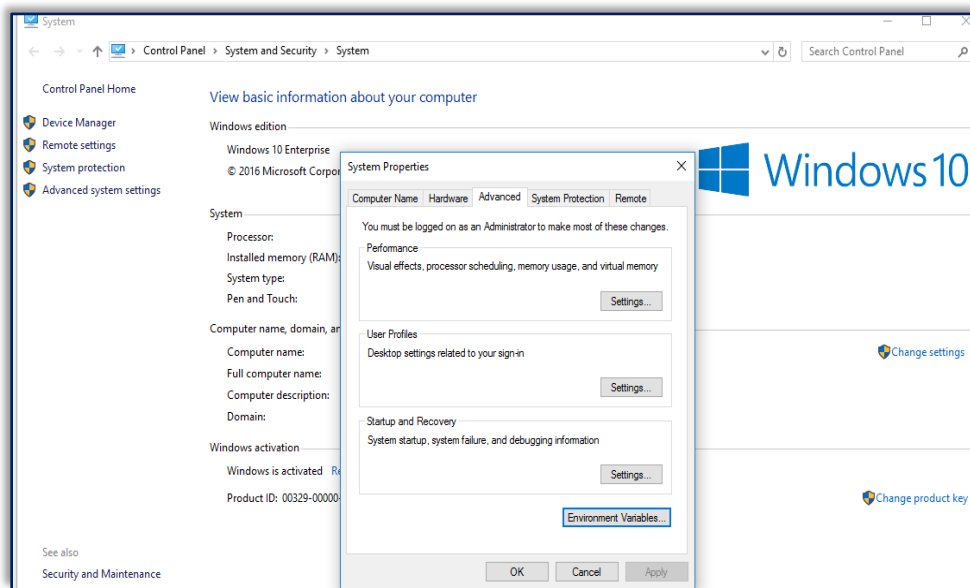
#### Set Environment Variables

Right click “This PC,” then select “Properties,” then “Advanced System settings,” and then “Environment variables”.

Put Environment Variable: JAVA\_HOME

Value: C:\Program Files\Java\jdk1.8.0\_xx (here xx is the update version)

Put %JAVA\_HOME%\bin in the System Path (at the very beginning of the Path variable value).



At the Command Prompt:

Run “java -version” to check Java version.

Run “where java” to verify the java path and make sure that java is from the path you specified in JAVA\_HOME (Java from JAVA\_HOME should appeared as the first item.)

### 1.1.3 Install Java Cryptography Extension (JCE)

Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8 Download is at <http://www.oracle.com/technetwork/java/javase/downloads/ice8-download-2133166.html>.

You can download from this link.

- Uncompress and extract the downloaded file.

This will create a subdirectory called UnlimitedJCEPolicyJDK8. This directory contains the following files:

- ✓ README.txt
- ✓ local\_policy.jar — Unlimited strength local policy file
- ✓ US\_export\_policy.jar — Unlimited strength US export policy file
- Install the unlimited strength policy JAR files.

In case you later decide to revert to the original "strong" but limited policy versions, first make a copy of the original JCE policy files (US\_export\_policy.jar and local\_policy.jar). Then replace the strong policy files with the unlimited strength versions extracted in the previous step.

The standard place for JCE jurisdiction policy JAR files is:

➤ %JAVA\_HOME%\jre\lib\security

## 1.2 Maven

### 1.2.1 Install Maven

- Maven 3 (Version 3.3.3) is used in Consent2Share development if not otherwise specified in individual project by Maven Wrapper.
- You can download it from <https://archive.apache.org/dist/maven/maven-3/> (the latest version from <https://maven.apache.org/download.cgi>).
- Extract maven from apache-maven-3.3.3-bin.zip to the c:\java folder.

### 1.2.2 Set M2\_HOME System Environment Variable and System Path

- Set M2\_HOME system environment variable:
  - ✓ Variable: M2\_HOME
  - ✓ Value: C:\java\apache-maven-3.3.3
- Put %M2\_HOME%\bin in the System Path.
- Run “mvn -version” to verify.



## 1.3 Tomcat 8

### 1.3.1 Install Tomcat

- You can download it from <https://tomcat.apache.org/download-80.cgi>
- Unzip to C:\java folder.

### 1.3.2 Configure JVM and other options for Tomcat

- You can configure JVM and other options for Tomcat.
- Go to the bin folder of the Tomcat installation, open catalina.bat/catalina.sh based on your operation system to see how you can configure.
- By default, Tomcat runs on JVM pointed by JAVA\_HOME environment variable.

### 1.3.3 Configure User to operate the "/manager/html" web application

Go to the conf folder of the Tomcat installation path, open tomcat-users.xml file, add the following line inside of tomcat-users element:

➤ `<user username="admin" password="admin" roles="manager-gui"/>`

## 1.4 MySQL Server and Workbench

### 1.4.1 Install

You can download it from <https://downloads.mysql.com/archives/installer/> (The latest version is located at <https://dev.mysql.com/downloads/windows/installer/>).

Set up username (root) and password (admin) as admin to access local MySQL Server.

### 1.4.2 Set up system environment variable and system path

- Environment Variable: MYSQL\_HOME
- Value: C:\Program Files\MySQL\MySQL Server 5.6
- Append path system variable with %MYSQL\_HOME%\bin

## 1.5 Install Git for Windows

Download Git-1.9.5-preview20141217.exe from <https://github.com/msysgit/msysgit/releases/>.

To handle line ending issue, open the gitbash and issue the following command:

➤ `git config --global core.autocrlf true`

### 1.5.1 Create a Github Account Linking to Your FEi Email Address

Once you create a GitHub account, ask the Team Lead to add you to our GitHub Team, so that you can access Consent2Share private Github repositories.

- Remember to set up a Git global user name and email address. When you commit changes in Git, this information will be used for the author and committer information in the commit, which is saved in the Git repository.
- To set up a global username:
  - `git config --global user.name "YourFirstName Your LastName"` (make sure you enter double '-' before global).
- To set up a global email address:
  - `git config --global user.email "your email address"`

- To verify your username and email set up:
  - `git config --global user.name`
  - `git config --global user.email`

## 1.6 Gradle

### 1.6.1 Install

- You can download it from <https://gradle.org/releases/>. e.g. gradle-2.8-all.zip is downloaded.
- Extract gradle from gradle-2.8-all.zip to the c:\java folder.

### 1.6.2 Set GRADLE\_HOME system environment variable and system path

- Set GRADLE\_HOME system environment variable:
  - Variable: GRADLE\_HOME
  - Value: C:\java\gradle-2.8 (Based on your directory)
  - Put %GRADLE\_HOME%\bin in System Path.
- Run “gradle -version” to verify

### 1.6.3 The Gradle Wrapper

The Gradle Wrapper is the preferred way of starting a Gradle build. The wrapper is a batch script on Windows, and a shell script for other operating systems. When you start a Gradle build via the wrapper, Gradle will be automatically downloaded and used to run the build.

You should check the wrapper into version control. By distributing the wrapper with your project, anyone can work with it without needing to install Gradle beforehand. Even better, users of the build are guaranteed to use the version of Gradle that the build was designed to work with. Of course, this is also great for continuous integration servers (i.e. servers that regularly build your project) as it requires no Gradle installation and configuration on the CI server.

## 1.7 Install Flyway (Optional)


Refer to “Flyway Installation Instructions” at [this link](#) on SharePoint.

### 1.7.1 Prerequisites

- Flyway 3.2.1 and mysql-connector-java-5.1.26-bin.jar.
- Download from internet:
  - <http://flywaydb.org/getstarted/download.html>
  - <http://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.26>

### 1.7.2 Windows Installation:

- Copy and unzip ‘flyway-commandline-3.2.1-windows-x64.zip’ to your machine
- Set FLYWAY\_HOME system environment variable and system path (Optional)
  - ✓ Variable: FLYWAY\_HOME
  - ✓ Value: C:\java\flyway-3.2.1 (Based on your directory)
  - ✓ Put %FLYWAY\_HOME% in System Path.

- 
- ✓ Run flyway--version to check Flyway version.
  - Copy mysql-connector-java-5.1.26-bin.jar to flyway-3.2.1\jars
  - To check flyway versioned database, you can go to flyway-3.2.1\conf directory and set up database configuration in flyway.conf file:
    - ✓ flyway.url=
    - ✓ flyway.user=
    - ✓ flyway.password=
  - Then go to command prompt by typing “flyway info” command to verify the change history of the database

For more, please read: <http://flywaydb.org/documentation/commandline/>

## 2 Frontend Development Environment Setup

### 2.1 Install Platform - Node.js

Go to <https://nodejs.org/>, download node.js for your operating system.

Or you can go to I:\Common\Software\Frontend to find downloaded node.js.

In addition to Node.js, npm is installed as well.

Verify the installation by issuing the following commands in command prompt:

- `node --version` (recommended:Node 6.9.0 or higher)
- `npm --version` (recommended:NPM 3 or higher)

### 2.2 Install Angular CLI

Angular CLI is a Command Line Interface (CLI) to automate your development workflow.

Install:

- `npm install -g @angular/cli@latest`

Then to verify the installation by issuing the following command:

- `ng --version`

### 3 Install and Configure IntelliJ IDEA for Consent2Share V3 Development

Consent2Share source code can be loaded and developed in any IDE. e.g. [IntelliJ IDEA](#), [Eclipse](#), [Spring Tool Suite](#), [NetBeans](#). Since Consent2Share source code heavily uses Spring Projects [IntelliJ IDEA](#) and [Spring Tool Suite](#) are the recommend IDEs.

The following set up is based on IntelliJ IDEA Ultimate version.

#### 3.1 Install

Download the latest IntelliJ IDEA Ultimate from <https://www.jetbrains.com/idea/download>. Then, start the installation. After installing, register your installation if you have the license.

##### 3.1.1 Configure/Update JDK

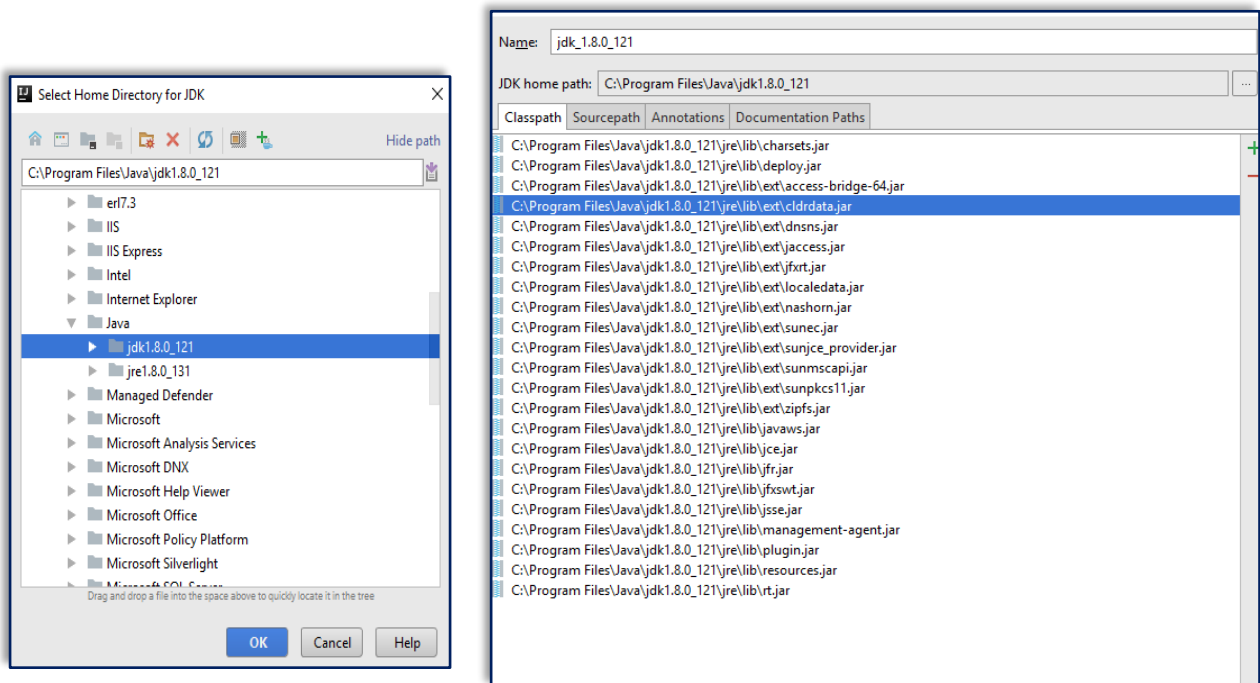
###### 3.1.1.1 Configure SDKs at the Global (IDE) Level

- Open the Default Project Structure dialog (from Welcome Screen → Configure → Project Defaults → Project Structure, or from File → Other Settings → Default Project Structure).
- In the left-hand pane, under Platform Settings, click SDKs.
- To add a new SDK, click add and select the desired SDK type.
- In the dialog that opens, select the SDK home directory and click OK.
- As a result, a new SDK is added to IntelliJ IDEA, and its settings are shown on the SDK page in the right-hand part of the dialog.
- Optionally, edit the SDK name and contents.
- If necessary, add more SDKs as described above.
- Click OK in the Project Structure dialog.

###### 3.1.1.2 Configure a Default Project SDK

- Open the Default Project Structure dialog (from Welcome Screen → Configure → Project Defaults → Project Structure, or from File → Other Settings → Default Project Structure).
- In the left-hand pane, under Project Settings, click Project.

- On the page that opens in the right-hand part of the dialog, select the necessary SDK from the Project SDK list.
- If the desired SDK is not present in the list, click New and select the necessary SDK type.



- In the dialog that opens, select the SDK home directory and click OK. As a result, a new SDK is added to IntelliJ IDEA as Global Platform Settings and selected as the Default project SDK.
- To view or edit the SDK name and contents, click Edit. (The SDK page will open.)
- Click OK in the Project Structure dialog.

### 3.1.1.3 Configure a Project SDK

- Open the Project Structure dialog (e.g. Ctrl+Shift+Alt+S, or from File → Project Structure).
- In the left-hand pane, under Project Settings, click Project.
- On the page that opens in the right-hand part of the dialog, select the necessary SDK from the Project SDK list.
- If the desired SDK is not present in the list, click New and select the necessary SDK type. In the dialog that opens, select the SDK home directory and click OK. As a result, a new SDK is added to IntelliJ IDEA as Global Platform Settings and selected as the project SDK.
- To view or edit the SDK name and contents, click Edit. (The SDK page will open.)
- Click OK in the Project Structure dialog.

### 3.1.2 Configure Maven

#### 3.1.2.1 Configure Maven at the Global (IDE) Level

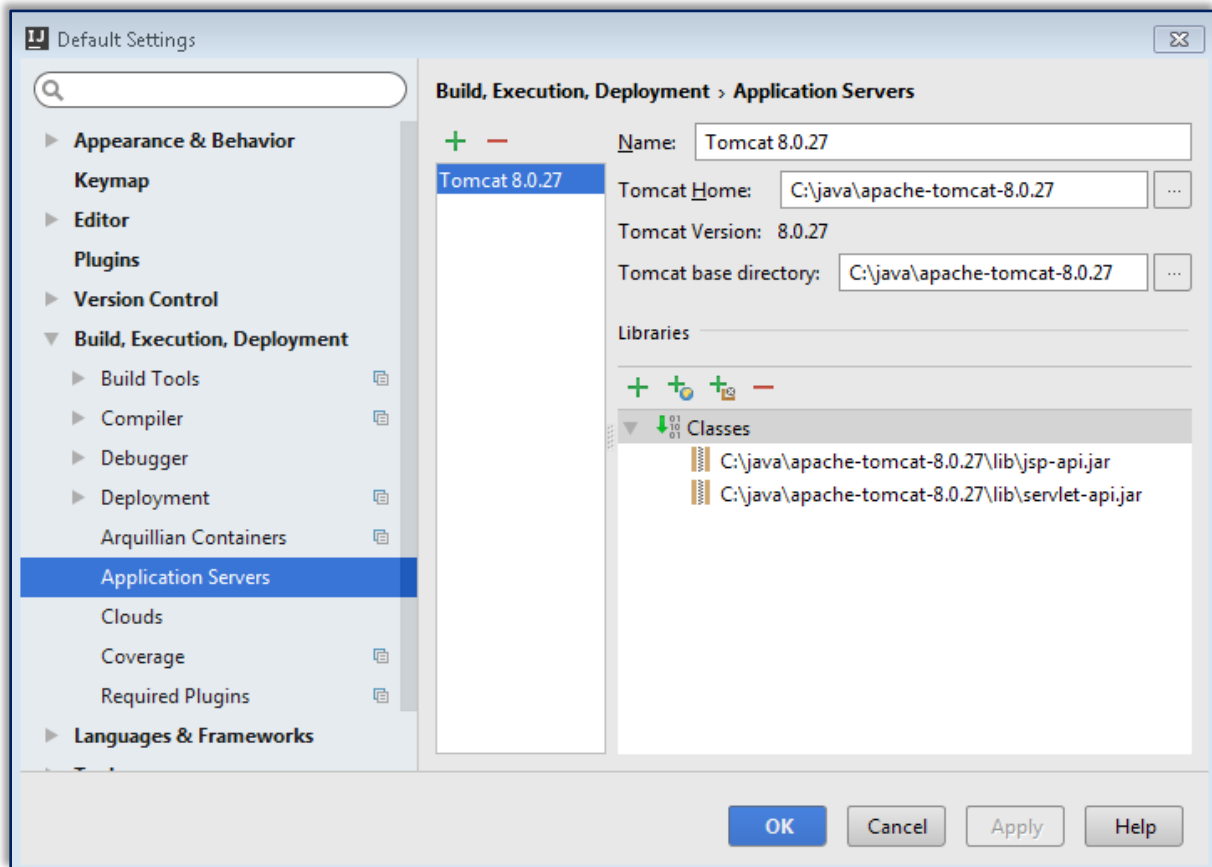
- Open the Default Settings dialog (from Welcome Screen → Configure → Project Defaults → Settings, or from File → Other Settings → Default Settings).
- In the left-hand pane, under Build, Execution, Deployment → Build Tools, click Maven.
- In the right-hand pane, set Maven home directory to the Maven directory which is the parent folder of bin folder.
- Click OK in the Default Settings dialog.

#### 3.1.2.2 Configure Maven for a Project

- Open the Settings dialog (from File → Settings).
- In the left-hand pane, under Build, Execution, Deployment → Build Tools, click Maven.
- In the right-hand pane, set Maven home directory to the Maven directory, which is the parent folder of the bin folder.
- Click OK in the Settings dialog.

### 3.1.3 Set up Tomcat

- From the Welcome Screen, click Configure → Settings, open Settings dialog. Alternatively, after you open your project, open the Settings dialog from File → Settings. Both ways achieve some results here, but the first way is better.
- In the left-hand pane, under Build, Execution, Deployment, click Application Servers.
- Click the green “+” in the middle pane to set up Tomcat. Provide a meaningful name for the application server e.g. Tomcat 8.0.27.



### 3.1.4 Configure Github

From the Welcome screen, open the Default Settings dialog. Version Control → Github. Select and enter the followings in the right-hand pane.

- Host: github.com
- Auth Type: Password
- Login: your username
- Password: your-password

### 3.1.5 Configure for Frontend Development

#### 3.1.5.1 Install the NodeJS Plugin

- In the Welcome Screen, click Configure → Settings → Plugins → Browse repositories, → type node.js to find the NodeJS plugin → Install plugin.
- Then you will be able to use Run → Edit Configuration, to open Run/Debug Configurations, and configure to run Node.JS.

#### 3.1.5.2 Install the Karma Plugin

In the Welcome Screen, click Configure → Settings → Plugins → Install JetBrains plugin ... → type karma to find the Karma plugin → click the green Install Plugin button to install.



### 3.1.6 Configure Default Settings for File Template

We do not want the “Created by” header in our source code files when creating new files. Use the following steps to configure:

- Select File → Other Settings → Default Settings.
- Expand the Editor option in the sidebar, then select File and Code Templates.
- Select the Includes tab, and then select File Header from the list. You will then see three lines of code in the box to the right of the list.
- Select all three lines of code in the box and delete them. Make sure the box is empty.
- Click the OK button.

### 3.1.7 Multirun Plugin

The Multirun Plugin enables multiple run configurations at once, group multiple run configurations, and the ability to start them in a single click. Not only can application and test run configurations be grouped, but other Multirun configurations can be organized into a single run configuration.

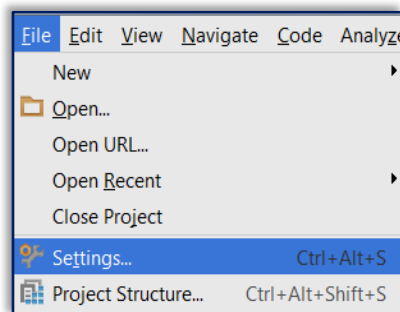
Use the following method to install Multirun Plugin:

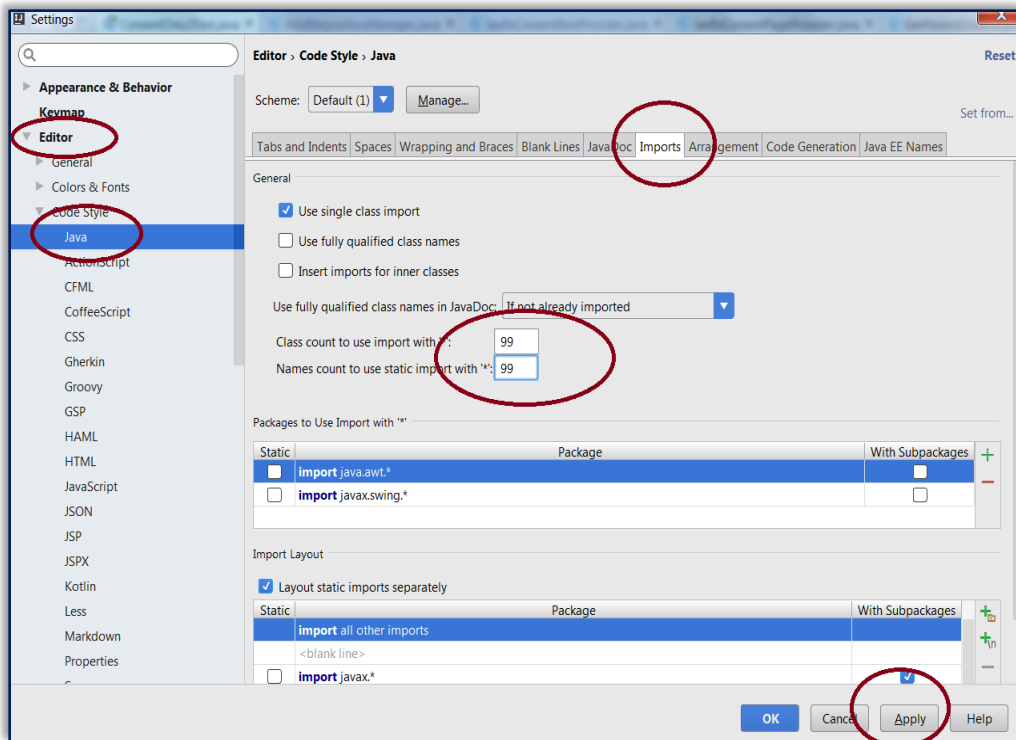
From Welcome Screen → Configure → Settings to Open Default Settings, Plugins then search for Multirun and install.

### 3.1.8 Configure Settings for Importing Java Classes

To force IntelliJ include each and every import individually instead of using (\*), do the following:

- Go to File → Settings → Editor → Code Style → Java → Import tab
- Set Class count to use import with ‘\*’ to 99 (a number which is bigger enough)
- Set Names count to use static import with ‘\*’ to 99 (a number which is large enough)





### 3.1.9 Install Lombok Plugin

Lombok is a framework that generates boilerplate code in an annotation-driven fashion and it has been utilized in several Consent2Share services. The projects will successfully build if the Lombok is on classpath (as a maven dependency), however one needs to also install a plugin to IDE to prevent errors. For IntelliJ Plugin installation:

- Go to: File → Settings → Plugins
- Search for Lombok to see whether it is installed.
- If it is already installed, no additional steps are required. If it is not installed, click the Search in Repositories link.
- Select Lombok Plugin and click the Install button. Potentially, you might need to restart the IDE after installation.
- See <https://projectlombok.org/> for details including documentation and support for other IDEs.

## 4 Load and Develop Consent2Share V3 Projects in IntelliJ IDEA

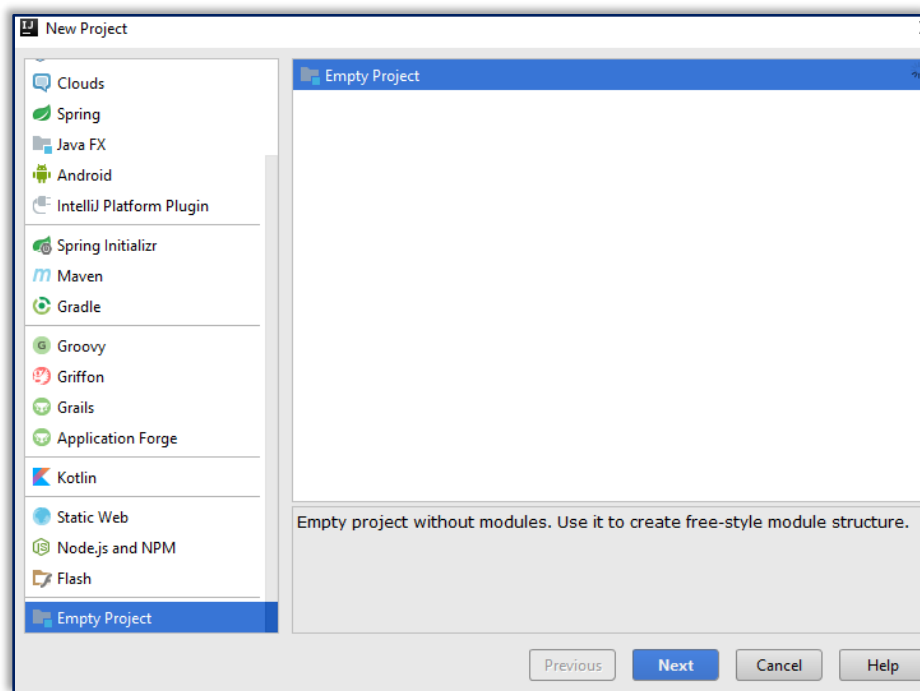
The first three chapters involved installing tools required to develop the Consent2Share application. This chapter focuses on how to set up Consent2Share V3 projects in IntelliJ IDEA for development.

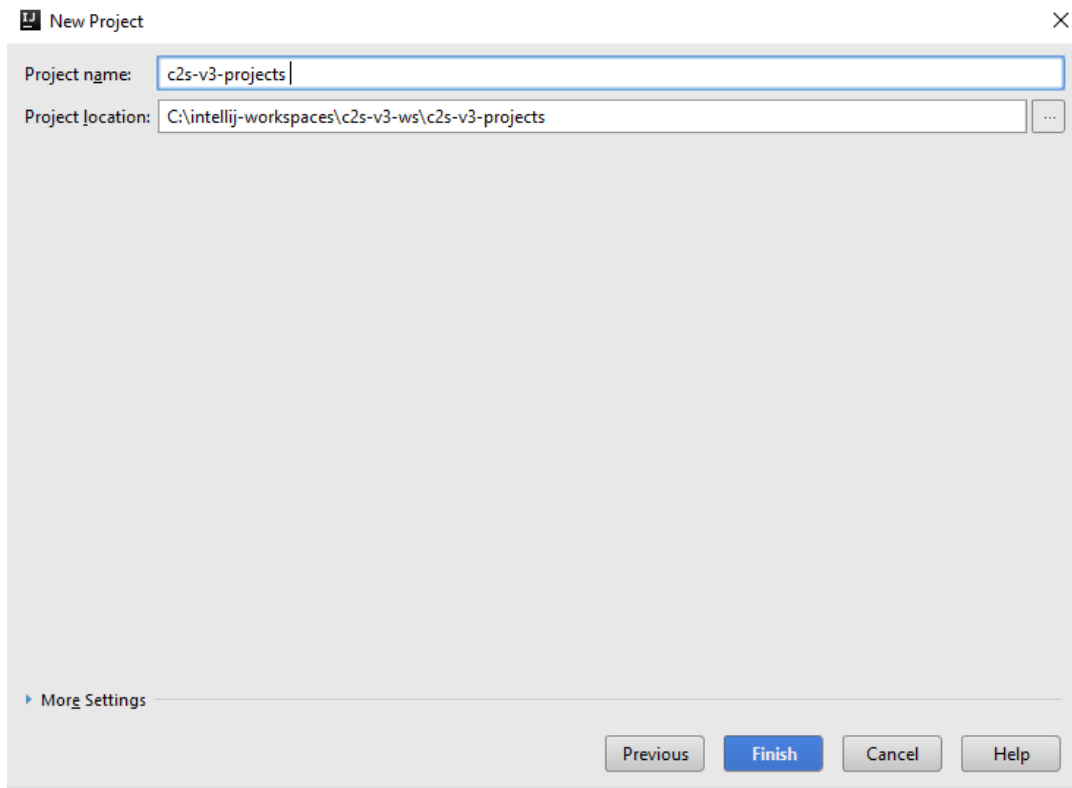
### 4.1 Open Multiple Projects in the Same Window in IntelliJ IDEA

If you are working with multiple projects, it is very convenient to open these projects in the same window for easy development. We can do this by creating an empty project and housing other projects as modules:

#### 4.1.1 Create an Empty Project

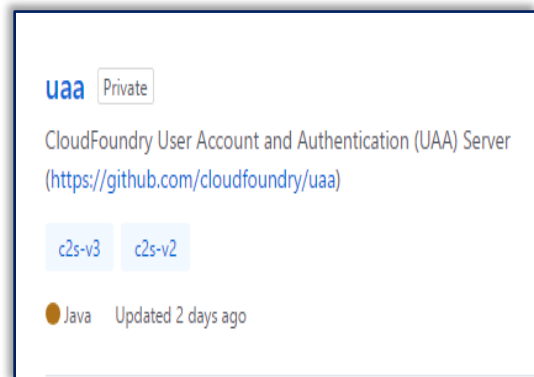
- In C:\ drive, create a new folder: intellij-workspaces
- Inside C:\intellij-workspaces, create another empty folder: c2s-v3-ws.
- Open IntelliJ
- From the Welcome Screen, click Create New Project or File → New → Project → Empty Project
- Give the name of the project (eg: c2s-v3-projects) and set the project location in c2s-v3-ws\c2s-v3-projects then click finish button.





#### 4.1.2 Clone Project Git Repositories

In your `intellij-workspaces\c2s-v3-ws` folder, we can now clone all the Git repositories for Consent2Share labeled by `c2s-v3` topic. Examples of such Git repositories are given below:



Clone all of the Git repositories labeled with c2s-v3 topic.

#### 4.1.2.1 Place c2s-config-data Repository

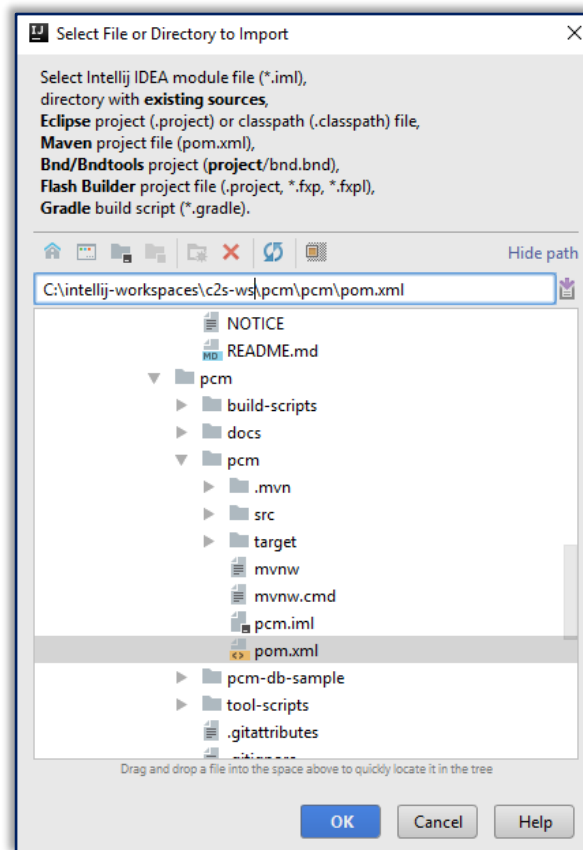
Based on the configuration (config-server/src/main/resources/application.yml) in config-server project, the configuration data Git repository (c2s-config-data) need be placed in a specific location which is C:\java.

Move the cloned c2s-config-data Git repository to C:\java folder or directly go to C:\Java and clone c2s-config-data Git repository.

#### 4.1.3 Import C2S Projects as Modules in IntelliJ IDEA

##### 4.1.3.1 Import all the Maven Projects

In the opened empty project that we created in section 4.1.1, go to File → New → Module from Existing Sources. Go to the appropriate cloned C2S project git repository work directory and select its pom.xml (or you can just select the folder which contains the pom.xml file), and click OK button to open Import Module modal dialog.



In the opened Import Module modal dialog, select Maven for Import module from external model, then click Next button to go to next step in Import Module modal dialog.

Select the *Import Maven projects automatically* option.

☒ Search for projects recursively

☐ Keep project files in:

☐ Import Maven projects automatically

☒ Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)

☐ Create module groups for multi-module Maven projects

☒ Keep source and test folders on reimport

☒ Exclude build directory (%PROJECT\_ROOT%/target)

☒ Use Maven output directories

Generated sources folders:

Phase to be used for folders update:

IDEA needs to execute one of the listed phases in order to discover all source folders that are contained in the project.  
Note that all test-\* phases firstly generate and compile production sources.

Automatically download: ☐ Sources ☐ Documentation

Dependency types:

Comma separated list of dependency types that should be imported

Click Next button, then click Finish button to close Import Module modal dialog.

In the project Tool Window, the project is listed as a module.

Follow the above steps to import all the C2S Maven projects as modules for the project named c2s-v3-projects.

Don't forget to import the server Maven projects in the UI projects (e.g. c2s-ui, master-ui etc.)

Alternatively, you can import all Maven projects in Project Structure modal dialog: go to File → Project Structure... to open Project Structure modal dialog, click Modules under Project Settings to open two panes on the right, in the middle pane, click + button and then select Import Module to import existing Maven projects.

#### 4.1.3.2 Import Client Projects

For Consent2Share UI projects, such as c2s-ui, the Maven pom.xml is located in the server folder. Using the pom.xml file only loads the server side project. We have to import the client site projects for all UI projects in IntelliJ IDEA.

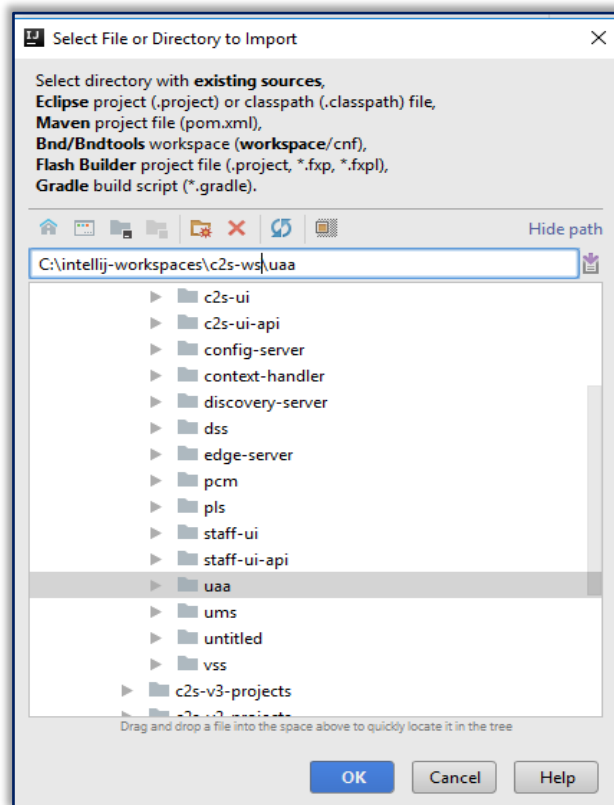
To import c2s-ui client project, go to File > New > Module from Existing Sources. In the opened modal dialog, choose c2s-ui\client directory and click OK button to open Import Module modal dialog. Choose 'Create module from existing source' in the modal dialog. Continue to follow along to finish the dialog.

Once the client project as a module is imported, it is named as client in the Project Tool Window. We need rename it to distinguish it from other client projects. Right click the client module and select 'Open Module Settings' and provide a name. For example: c2s-ui-client.

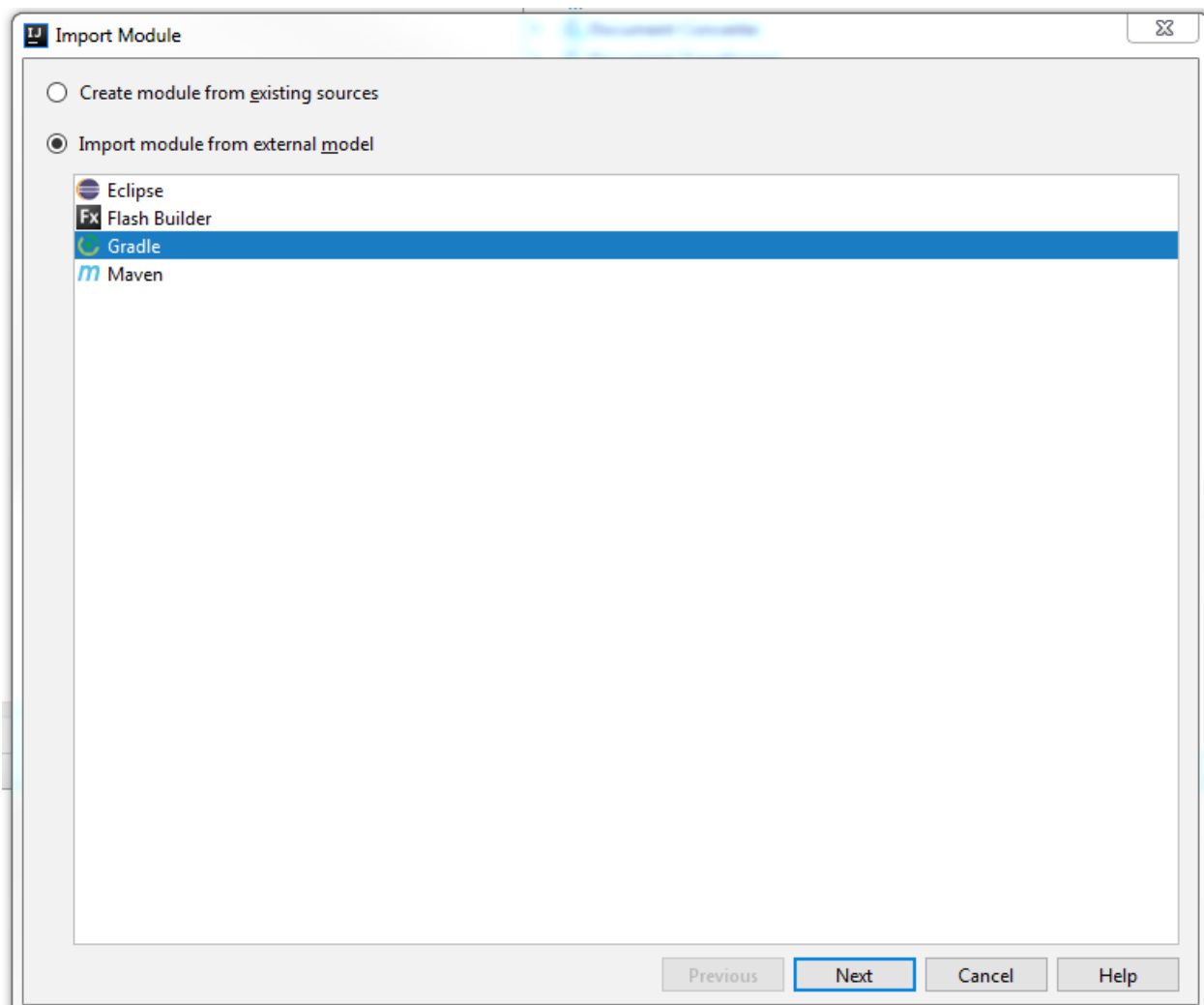
Follow the same steps to import other client projects.

#### 4.1.3.3 Import UAA

UAA is a Gradle project. To import UAA project, go to File → New → Module from existing sources and then select the uaa directory which contains build.gradle. Click OK button to open Import Module modal dialog.

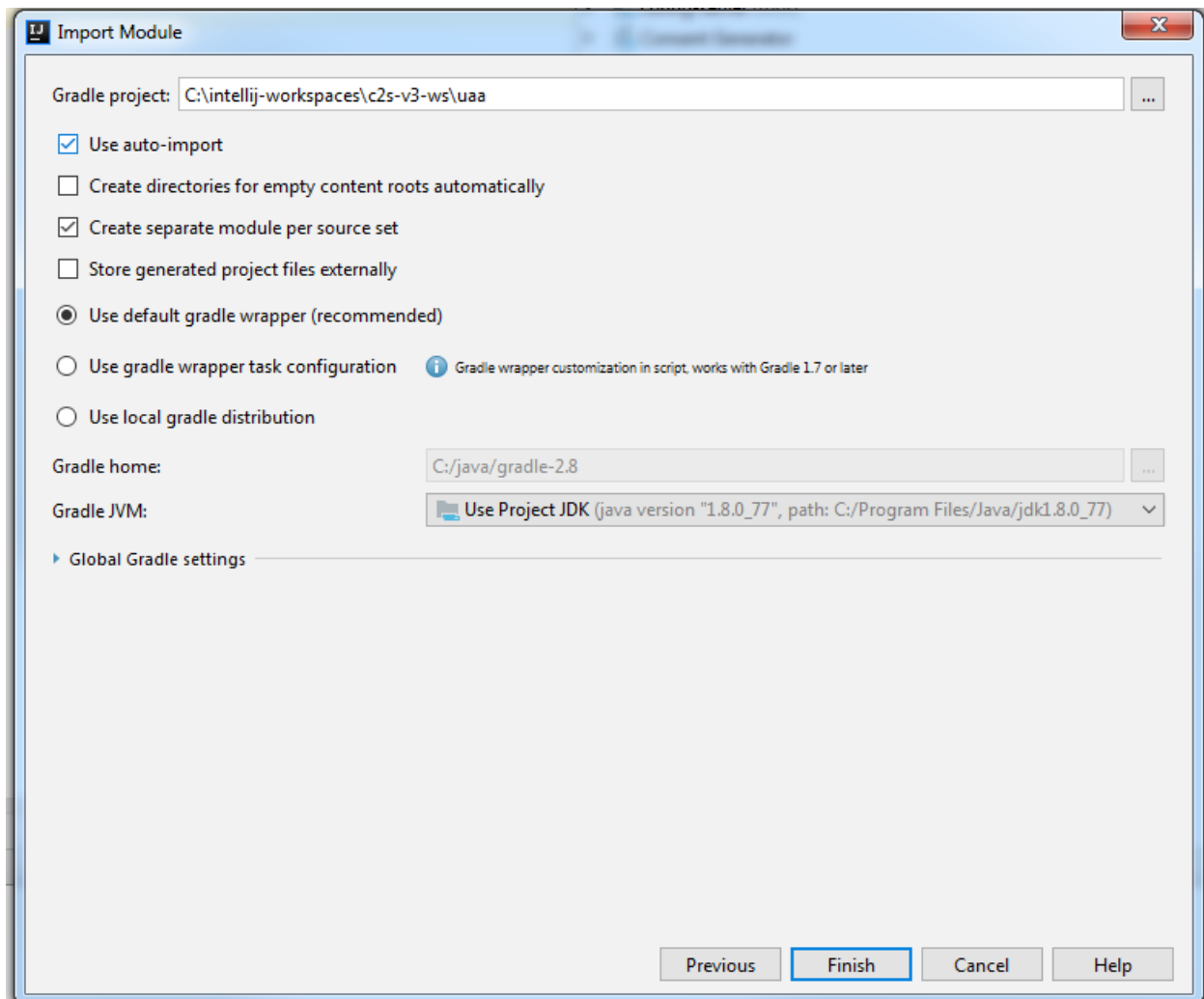


In the dialog, choose Import module from external model, and then choose Gradle. Click Next button.



Check Use auto-import, click Finish button.





## 4.2 Create Schemas in MySQL Workbench

Create empty schemas called: pcm, phr, uaa, ums, vss and pls.

Ex: CREATE SCHEMA 'uaa';

Tables and data will be created and inserted when corresponding applications are running for the first time.

## 4.3 Run/Debug Source Code

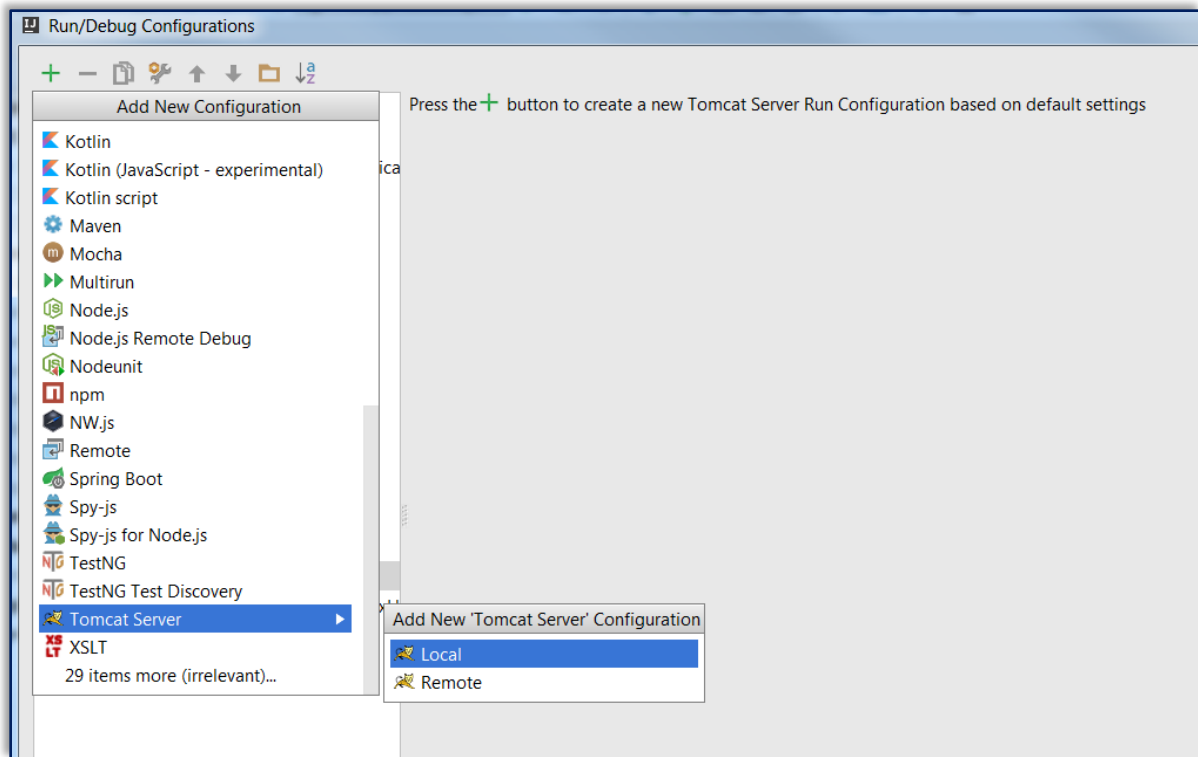
When you import a Spring Boot project, a Run/Debug configuration is automatically generated.

For UAA you need to configure Run/Debug configuration to use Tomcat server.

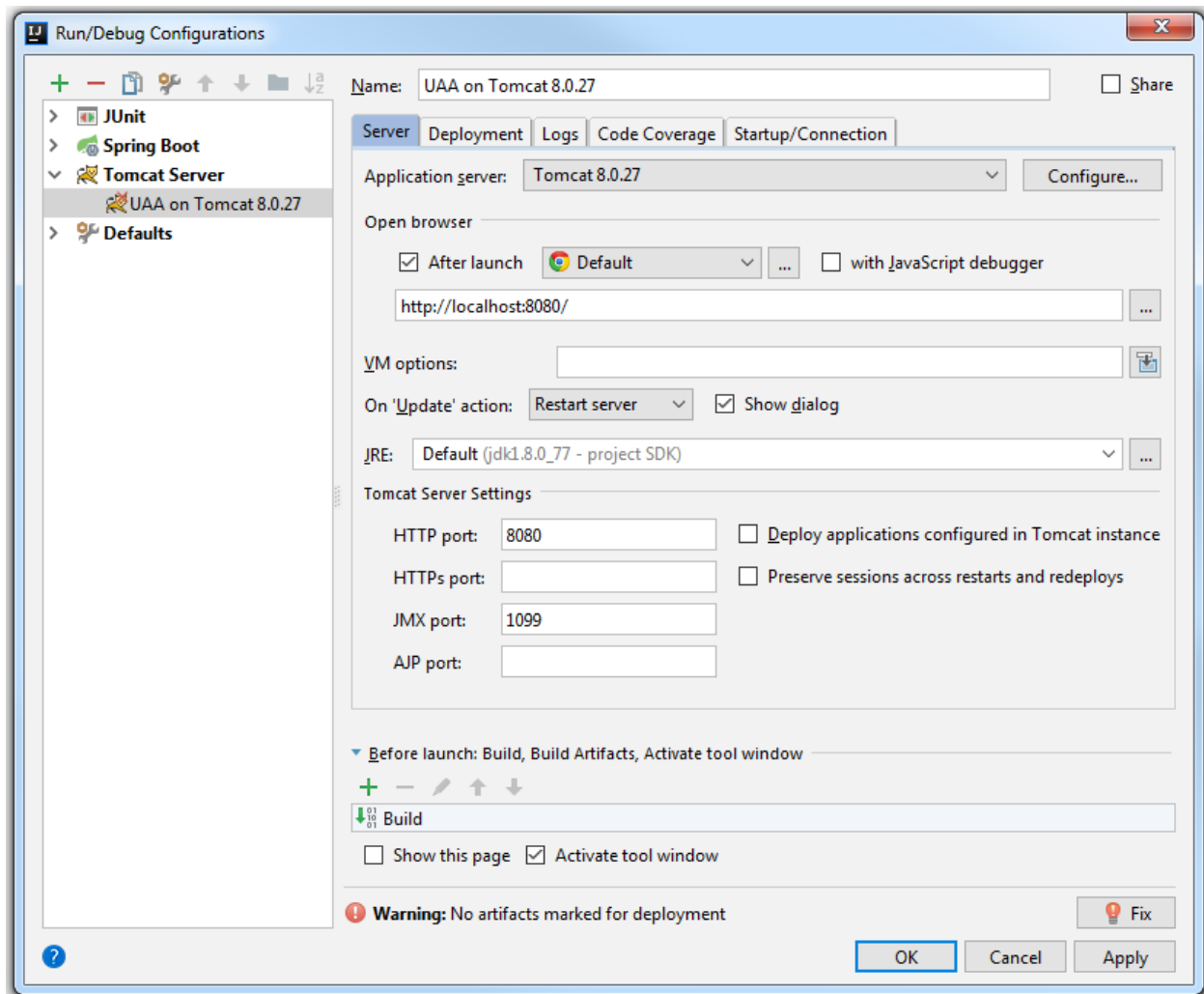
### 4.3.1 Run/Debug Configuration with Tomcat for UAA

Click Run → Edit Configurations to open “Run/Debug Configurations” dialog.

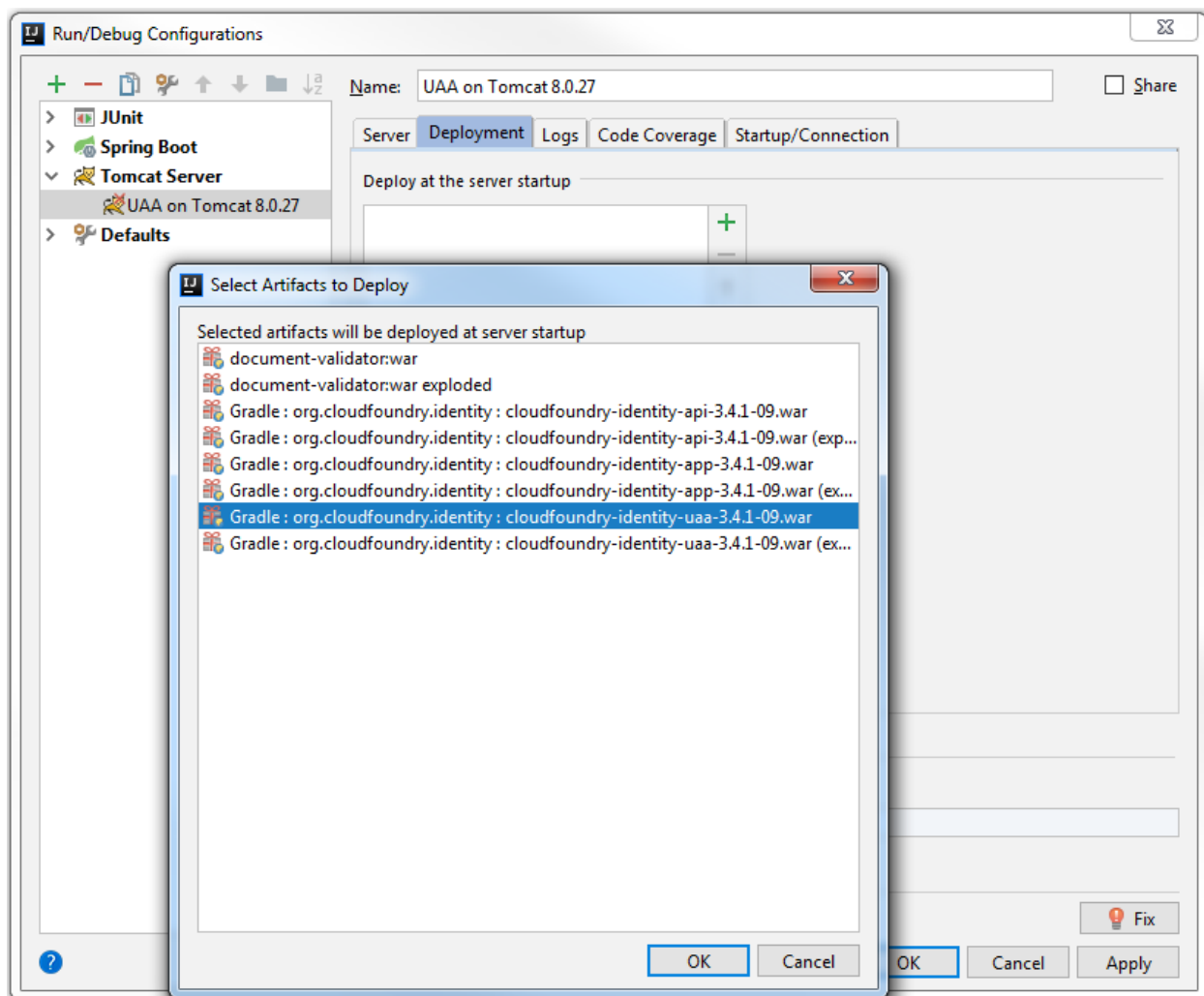
Click “+” in the upper left, and Select and press Tomcat Server → local to open new Tomcat Server Run Configuration.



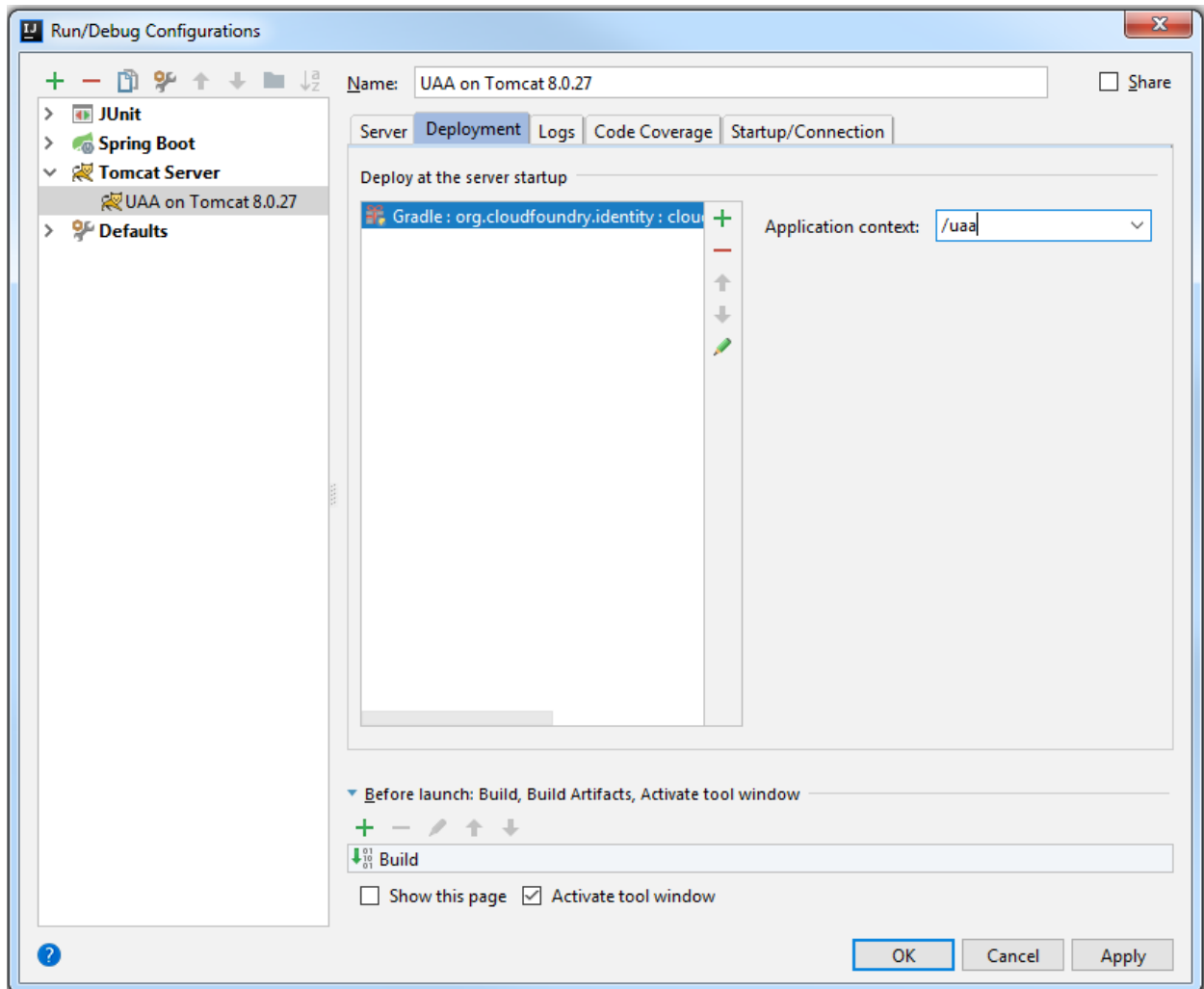
Then give the Configuration a relevant name like: “UAA on Tomcat 8.0.27”.



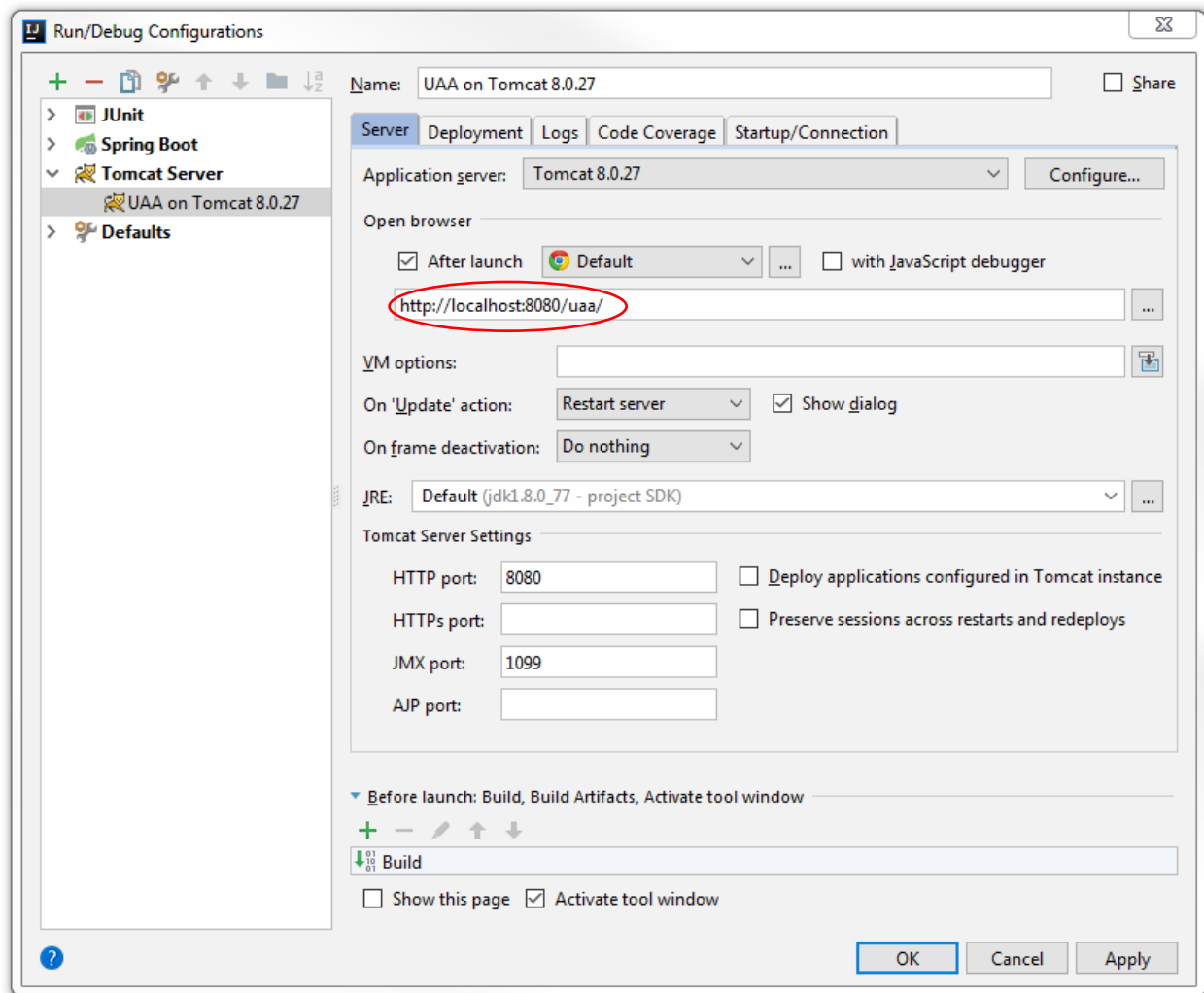
Then click the Deployment tab and click the “+” button under “Deploy at the server startup”. Then click “Artifact...” to open “Select Artifacts to Deploy” modal dialog. Choose uaa war.



Click OK button to close “Select Artifacts to Deploy” dialog and return to “Run/Debug Configurations” dialog. And make sure to change “Application Context” to /uaa.



Now switch back to Server tab, you should notice that the url under the Open browser is changed to: <http://localhost:8080/uaa/>



Click OK button to close “Run/Debug Configurations” dialog.

Add UAA Configuration Path Environment Variable:

You can add UAA\_CONFIG\_PATH as OS environment variable or JVM environment variable or application server environment variable. Here we set up the environment variable at application server level.

Append the following configuration line to the file of “catalina.properties” under the Tomcat directory. (C:\java\apache-tomcat-8.0.27\conf):

UAA\_CONFIG\_PATH=C:\\intellij-workspaces\\c2s-v3-ws\\uaa\\config-template

You can run this configuration to get a running UAA in the url <http://localhost:8080/uaa/>. The first time running of UAA creates tables and inserts data in the empty uaa schema.

#### 4.3.2 Run/Debug Configuration with NPM for UI Projects

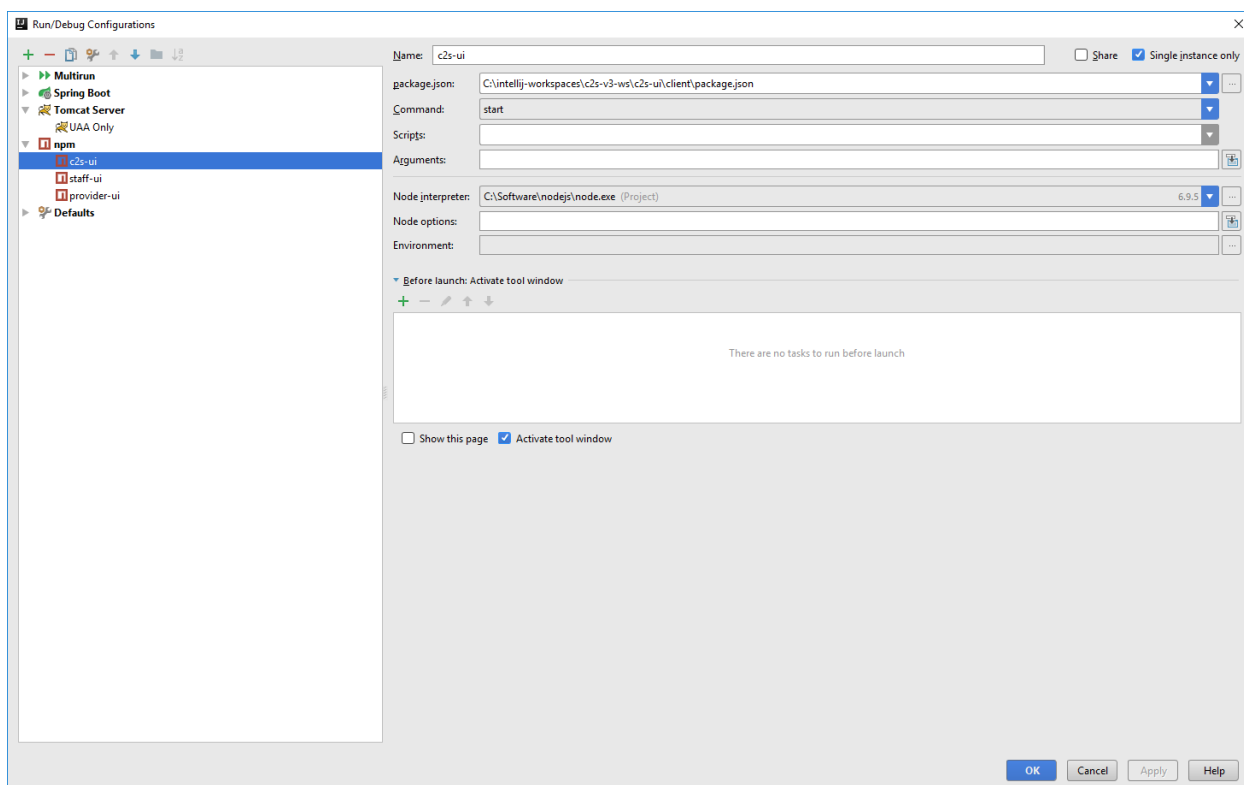
You can run Consent2Share ui projects by using the Spring Boot run configurations generated by IntelliJ.

The alternative way to run UI projects (Angular SPA projects) is to set up npm Run Configurations for these UI projects.

Click Run → Edit Configurations to open “Run/Debug Configurations” dialog.

Click “+” in the upper left, and select and press npm to open new npm Run Configuration.

Choose npm and ‘+’ icon. Create a c2s-ui run configuration as shown below with package.json pointing to c2s-ui project package.json in client folder:



Create the same kinds of run configurations for master-ui, staff-ui and provider-ui.

You couldn't really get these projects working by only using these npm run configurations without running the required backend services.

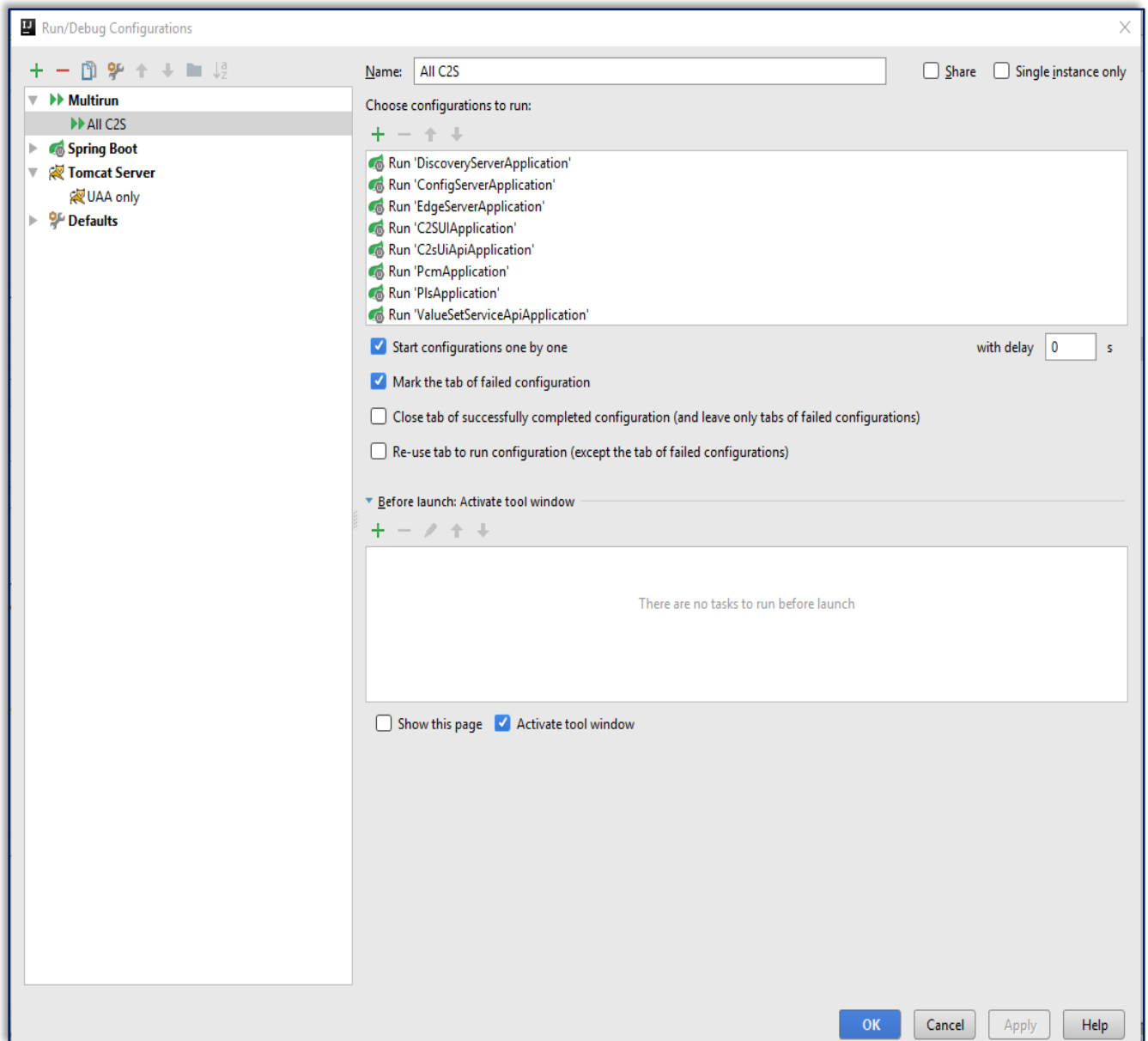
#### 4.3.3 Use Multirun Plugin to Run Consent2Share

Go to Run > Edit Configurations to open “Run/Debug Configurations” dialog. Click the “+” button at the left top side of the dialog to open “Add New Configuration” list. Click “Multirun” in the list to add a new Multirun configuration.

Give a meaningful name to the new Multirun configuration. Under “Choose configuration to run:”, click “+” to add Consent2Share Run configurations for this multirun configuration.

Always add “Run ‘DiscoveryServerApplication’” and “Run ‘ConfigServerApplicaition’” first. Add all other required Consent2Share Spring Boot Run configurations to the Run Multirun

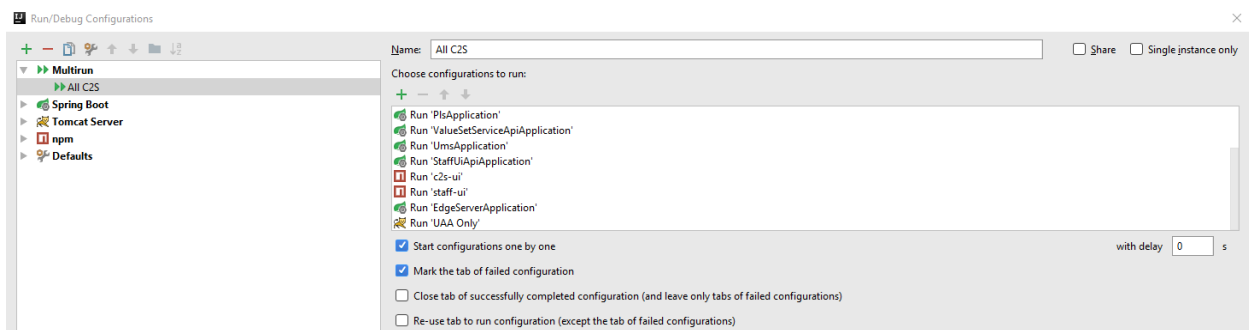
configuration.



If you don't want to Spring Boot Run configurations for Consent2Share UI projects, you add npm Run Configurations related to UI projects to this Multirun Configuration. But don't add both kinds of the Run Configurations for the same UI project at the same time.

Also add "Run 'UAA on Tomcat 8.0.27'" configuration if you want to run UAA together with 'All C2S' configuration otherwise it needs to be run separately.





To speed up development, you don't need to run all Consent2Share projects in this Multirun Run Configuration. You have the flexibility to choose which projects to be included in this Multirun configuration. Or you may want to create several Multirun Run Configurations each of which has different projects included.

#### 4.3.4 Add Environment Variables

To run Consent2Share in your IDE, we need set up the following environment variables with corresponding values (variable name: variable value):

```
spring.mail.protocol: smtp
spring.mail.host: ${your-smtp-mail-host}
spring.mail.port: 25
spring.mail.username: ${ask-team-member}
spring.mail.password: ${your-smtp-mail-password}
spring.mail.properties.mail.smtp.auth: true
spring.mail.properties.mail.smtp.ssl.trust: ${your-smtp-mail-host}
spring.mail.properties.mail.smtp.starttls.enable: true
```

```
UAA_SMTP_HOST: %spring.mail.host%
UAA_SMTP_PORT: %spring.mail.port%
UAA_SMTP_USER: %spring.mail.username%
UAA_SMTP_PASSWORD: %spring.mail.password%
```

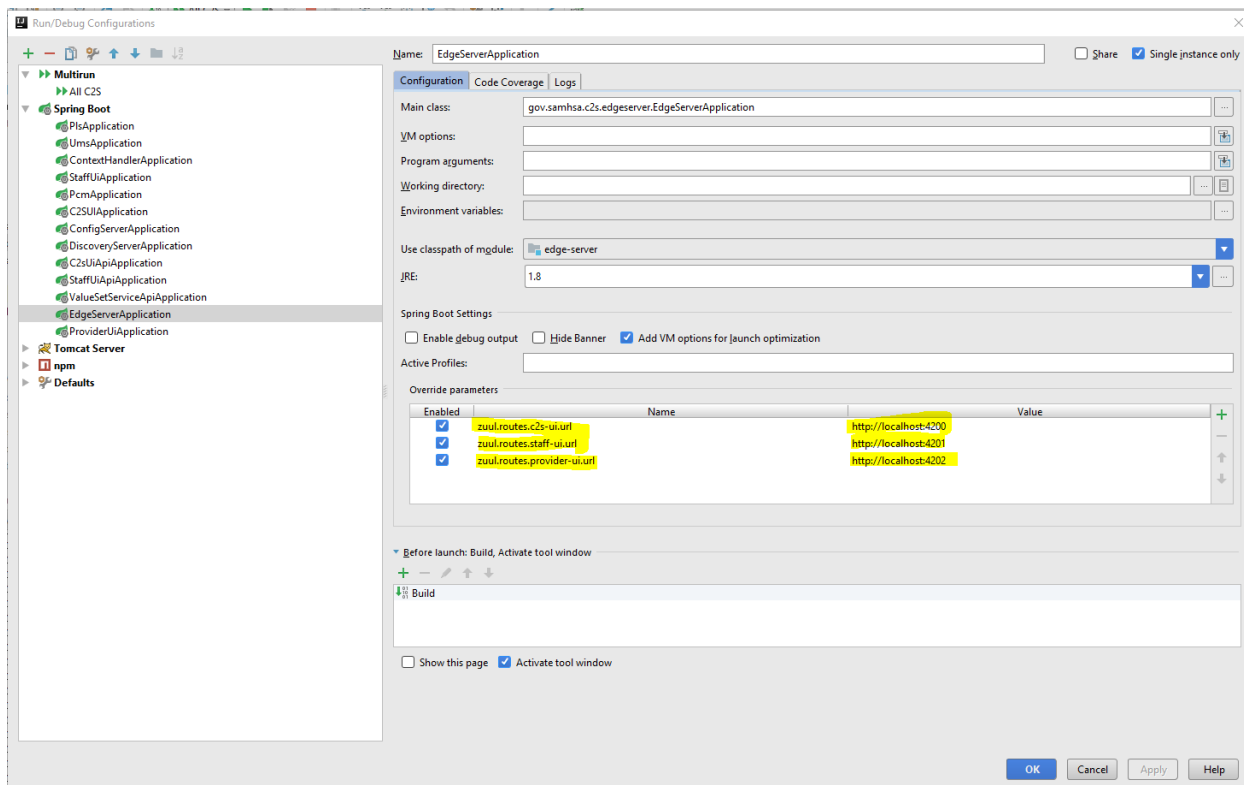
These environment variables can be set up as OS environment variables or JVM environment variable or application server environment variables. For Spring Boot applications, parameters can be set up at command line as well. For convenience here we set up them as OS environment variables.

The reason that we set up these environment variables is that we don't want to hard-code these sensitive information in configuration files which are source controlled and open-sourced. Though we could encrypt these values and put the encrypted values in configuration files for local development in IDE.

#### 4.3.5 Update Configuration for EdgeServerApplication

To quickly run UI projects in debug mode, we do the following set up for EdgeServer run configuration.

Go to Run → “Edit Configurations...” to open “Run/Debug Configurations” modal dialog. Click “Spring Boot” → EdgeServerApplication. Choose Configuration tab and add the values as shown below in the ‘Override parameters’ section:

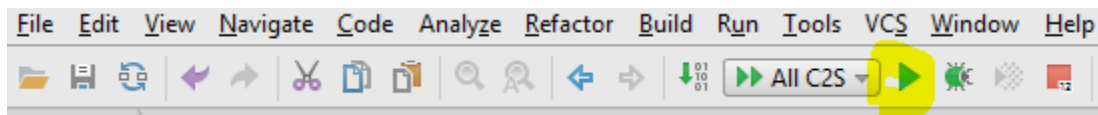


#### 4.3.6 common-libraries Artifacts

Consent2Share projects depend common-libraries Artifacts. If these common-libraries artifacts are not available in the artifacts repository (specified in Maven settings.xml) that you are using, you need build these artifacts first to put them in your artifacts repository or your local maven repository.

#### 4.3.7 Run Consent2Share Applications

Now, you can run the Consent2Share applications by clicking the button shown below.



The first time to run these applications, tables are generated for the empty schemas created in section 4.2.

You can use the following urls to check Consent2Share:

- <http://localhost:8761/>  
Eureka Dashboard
- <http://localhost:8080/uaa/>  
UAA login page
- localhost/c2s-ui
- localhost/staff-ui
- localhost/provider-ui

Check the Eureka server to check the instances currently registered.

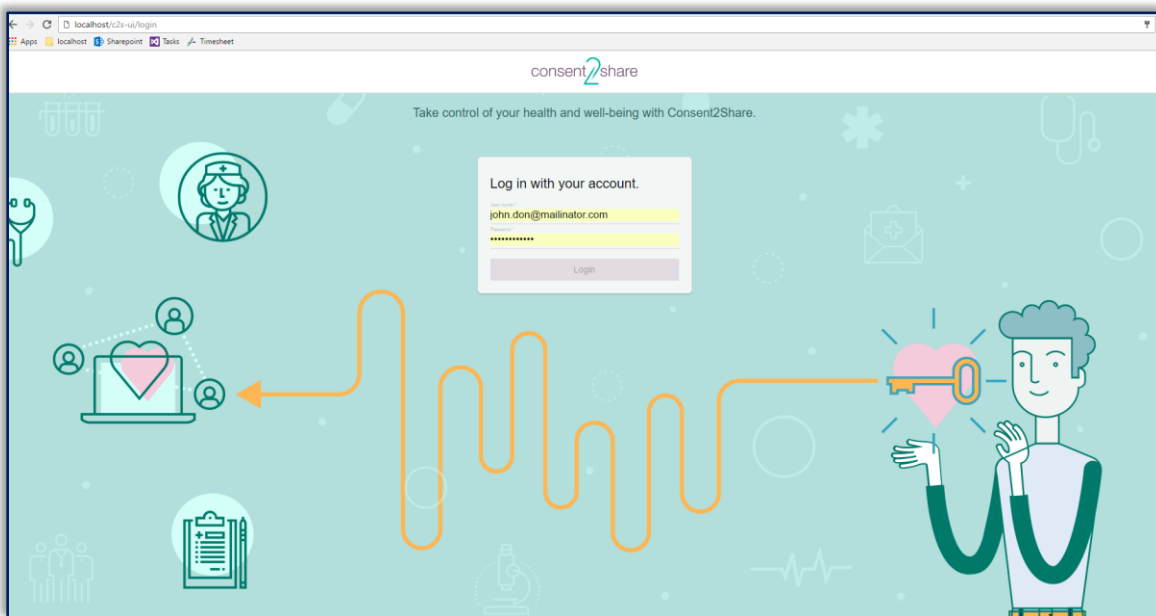
The screenshot shows the Spring Eureka Dashboard. The top navigation bar includes 'HOME' and 'LAST 1000 SINCE STARTUP'. The 'System Status' section displays the following information:

Environment	test	Current time	2017-06-15T13:55:59 -0400
Data center	default	Uptime	02:30
		Lease expiration enabled	true
		Renews threshold	0
		Renews (last min)	16

Below the status section, a red warning message states: "THE SELF PRESERVATION MODE IS TURNED OFF. THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS."


The 'DS Replicas' section shows 'Instances currently registered with Eureka' in a table:

Application	AMIs	Availability Zones	Status
C2S-UI-API	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:c2s-ui-api:0227ae7e0a11a895fa876e17edb9096a
CONFIG-SERVER	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:config-server:f7cc2f48274771b232c734f11bdf25bf
EDGE-SERVER	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:edge-server:f2f616dc38f323f06b016139525e894
PCM	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:pcm:71373b26cd61b740f046cea7b322ee5d
PLS	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:pls:32a13f8bb178485291d8d5fbc03b2e0
STAFF-UI-API	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:staff-ui-api:8b43adc849333757e037d93767db25b8c
UMS	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:ums:ca08156d7badb22b9b6d882f2e6f962
VSS	n/a (1)	(1)	UP (1) - URajkarnikarLT.fe1.local:vss:fb03f6915c0876e38f8e2f02b5826905



#### 4.3.8 Run SQL Scripts to Insert Data

Run the scripts available at the following locations to insert lookup data and sample data:



C:\intellij-workspaces\c2s-v3-ws\pcm\pcm-db-sample

C:\intellij-workspaces\c2s-v3-ws\ums\ums-db-sample

C:\intellij-workspaces\c2s-v3-ws\vss\vss-db-sample

C:\intellij-workspaces\c2s-v3-ws\phr\phr-db-sample

C:\intellij-workspaces\c2s-v3-ws\phr\pls-db-sample

## 5 References

### Github References

<https://git-scm.com/book/en/v2>

<https://app.pluralsight.com/library/courses/git-fundamentals/table-of-contents>

### Maven References

<http://maven.apache.org/guides/>

<https://app.pluralsight.com/library/courses/maven-fundamentals/table-of-contents>

### Gradle References

<https://gradle.org/docs>

<https://app.pluralsight.com/library/courses/gradle-fundamentals/table-of-contents>

### Git Credential Storage

<https://github.com/Microsoft/Git-Credential-Manager-for-Windows>