Backbase

Next Generation Portal Software

Preparing a typical development environment for CXP Backend projects using IntelliJ

Objectives

This guide explains how to configure a typical development environment for CXP Backend projects using *IntelliJ*. You will be able to execute the following activities:

- Prepare your IDE to be used for the first time
- Configure a CXP project
- Start and stop CXP servers using your IDE
- Configure the logging level of your application

Prerequisites

- Existing Launchpad-based Portal project: If you do not have configured this project yet, please follow this guide
- *IntelliJ IDEA* IDE: If you do not have this software yet, you can download it here.
  Two editions are offered: *Community* and *Ultimate*. The functionalities present in the *Community* edition are enough for learning purposes, and you can use it for this task. The *Ultimate* version is paid and has additional enterprise-wide features like *Spring* Integration

Timeframe

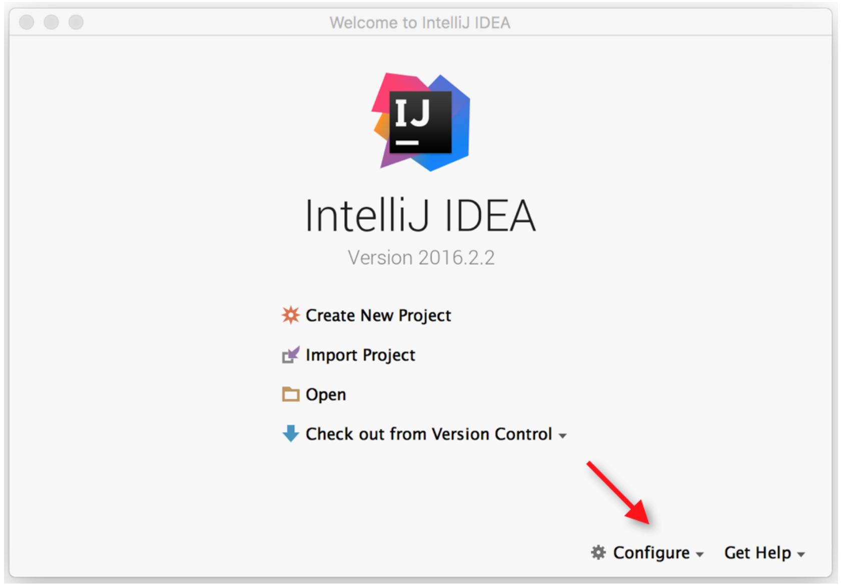
This task takes around 30 minutes.

Prepare your IDE to be used for the first time

During this step, you will set up your SDK (Standard Development Kit) and *Maven* in *IntelliJ.*

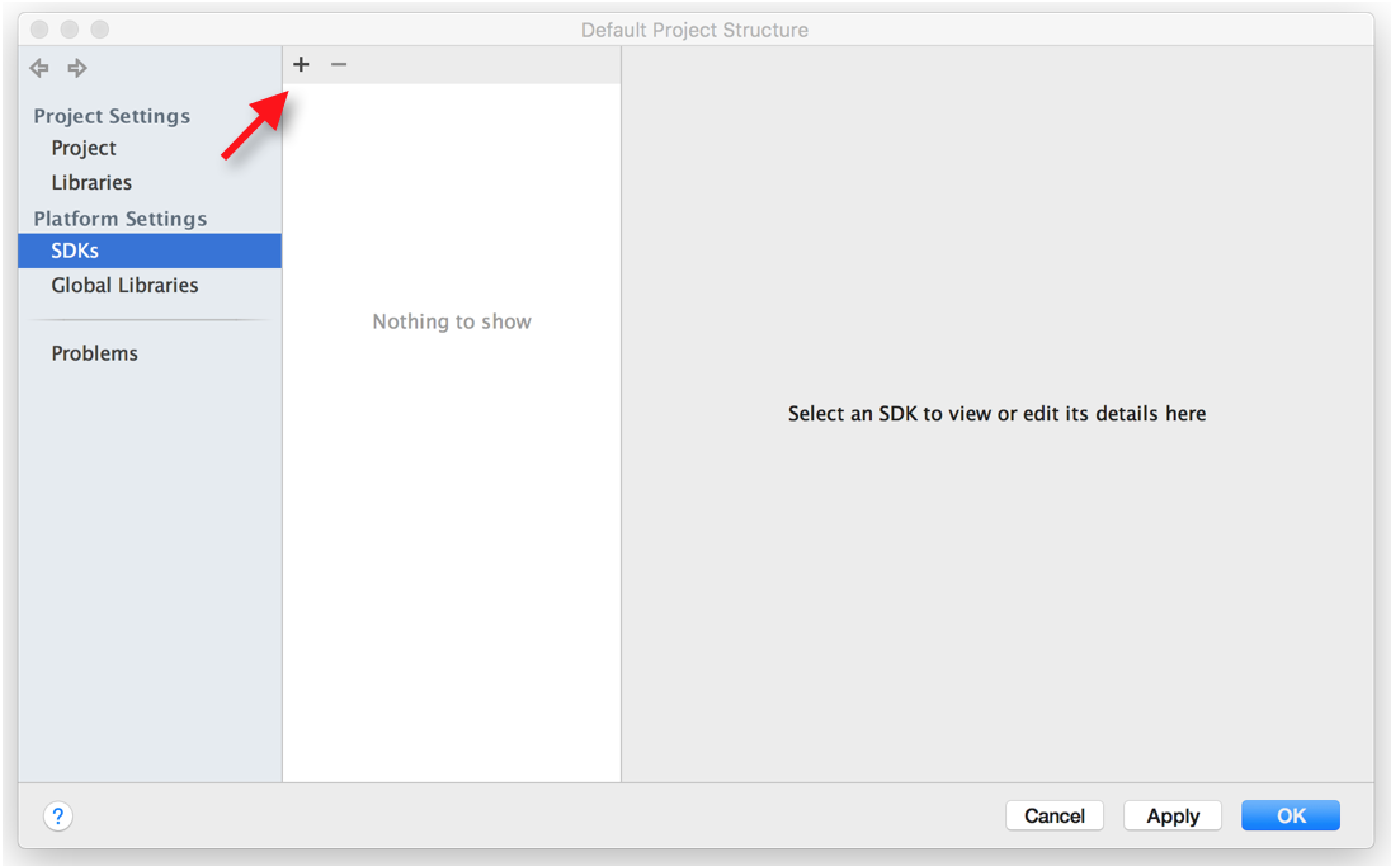
Configure SDK

1. Start *IntelliJ*. The initial screen will appear. Click the *Configure* present at the bottom of the welcome screen:



Please note that the layout of *IntelliJ* may vary depending on the operating system.

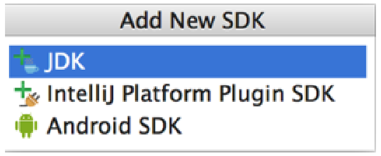
2. Select *Project Defaults* > *Project Structure*. The project configuration window will appear.
3. Select the *SDKs* option under *Platform Settings*, and click the **+** button highlighted in the image shown below:



Nothing to show

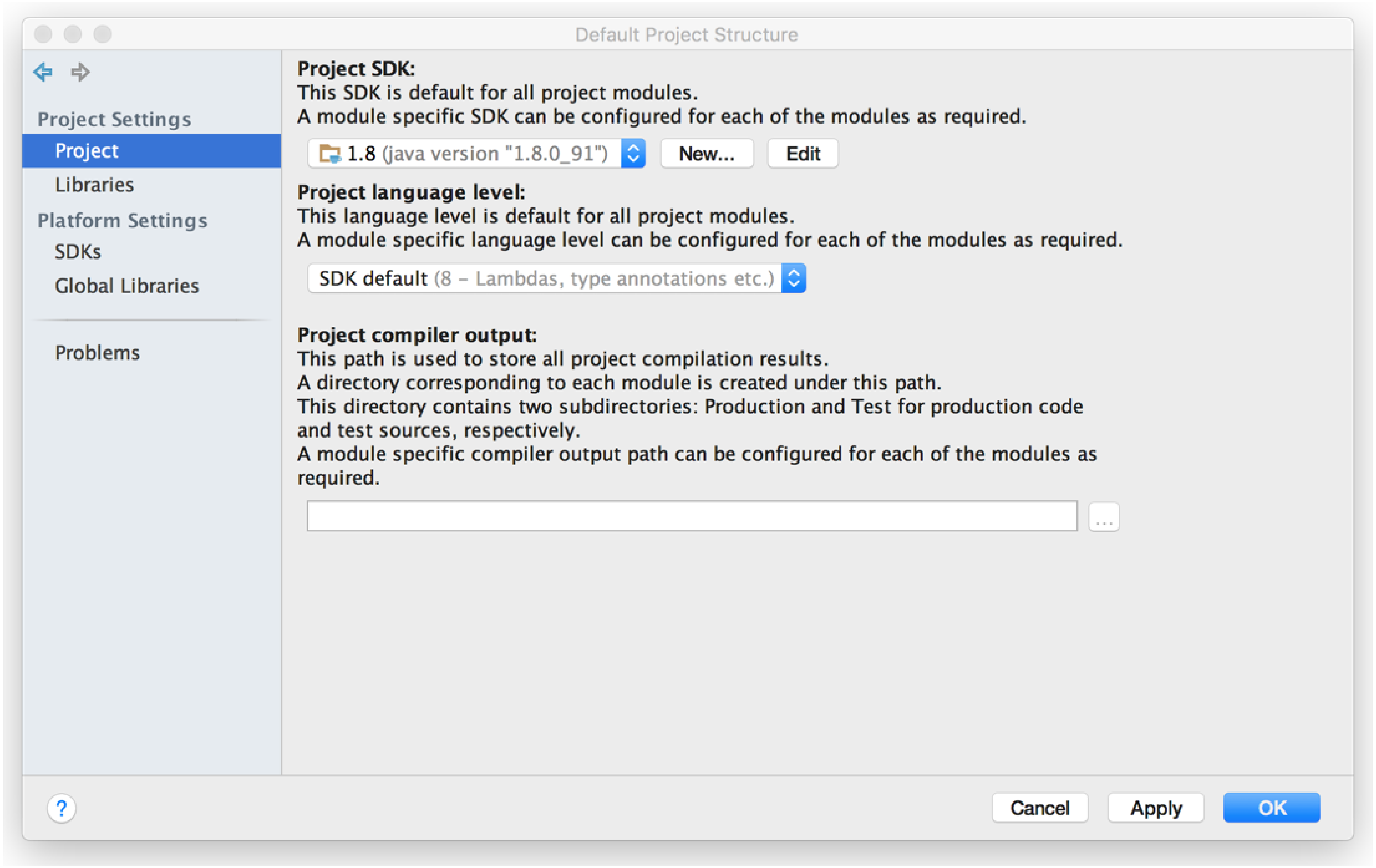
Select an SDK to view or edit its details here

4. Select the JDK (Java Development Kit) option among a set of predefined SDKs.



5. A dialog window will appear. Select your JDK home folder.
   If you forgot your installation folder, take a look at the content of your *JAVA_HOME* environment variable. Here the list of typical folders of JDK installations:

   - Windows machines: `c:\Program Files\Java\jdk`
   - Mac OSX machines: `/Library/Java/JavaVirtualMachines/jdk<version>.jdk/Contents/Home`
6. Select the *Project* option under *Project Settings*, and specify your JDK to be the default option for all projects. Make sure that the *Project language level* is greater than 7 (we are using 8 in this guide).



Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.

1.8 (java version "1.8.0_91")

Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.

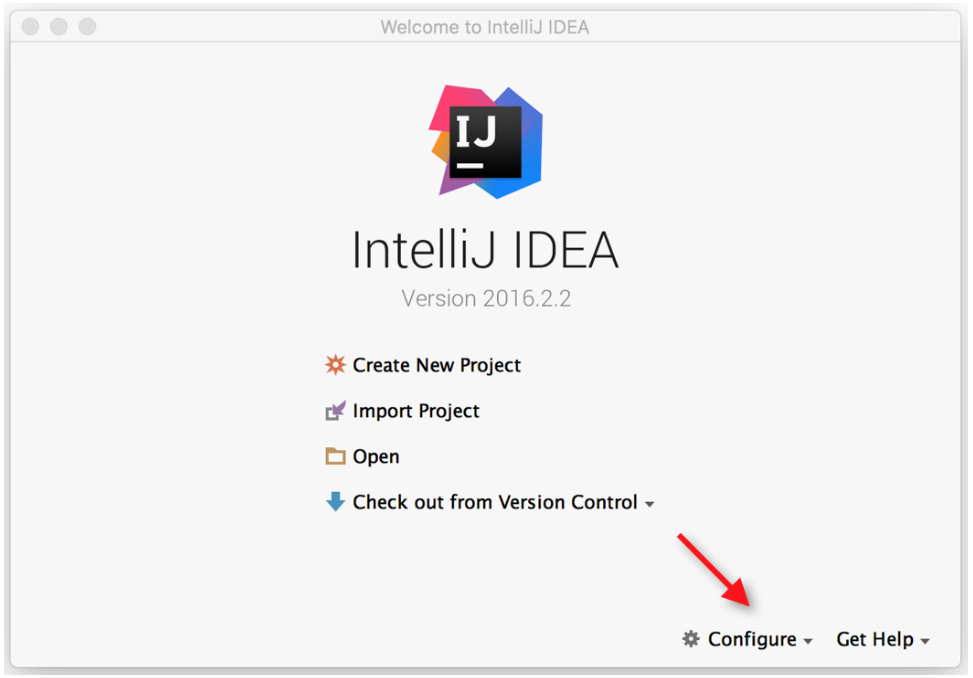
SDK default (8 – Lambdas, type annotations etc.)

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.

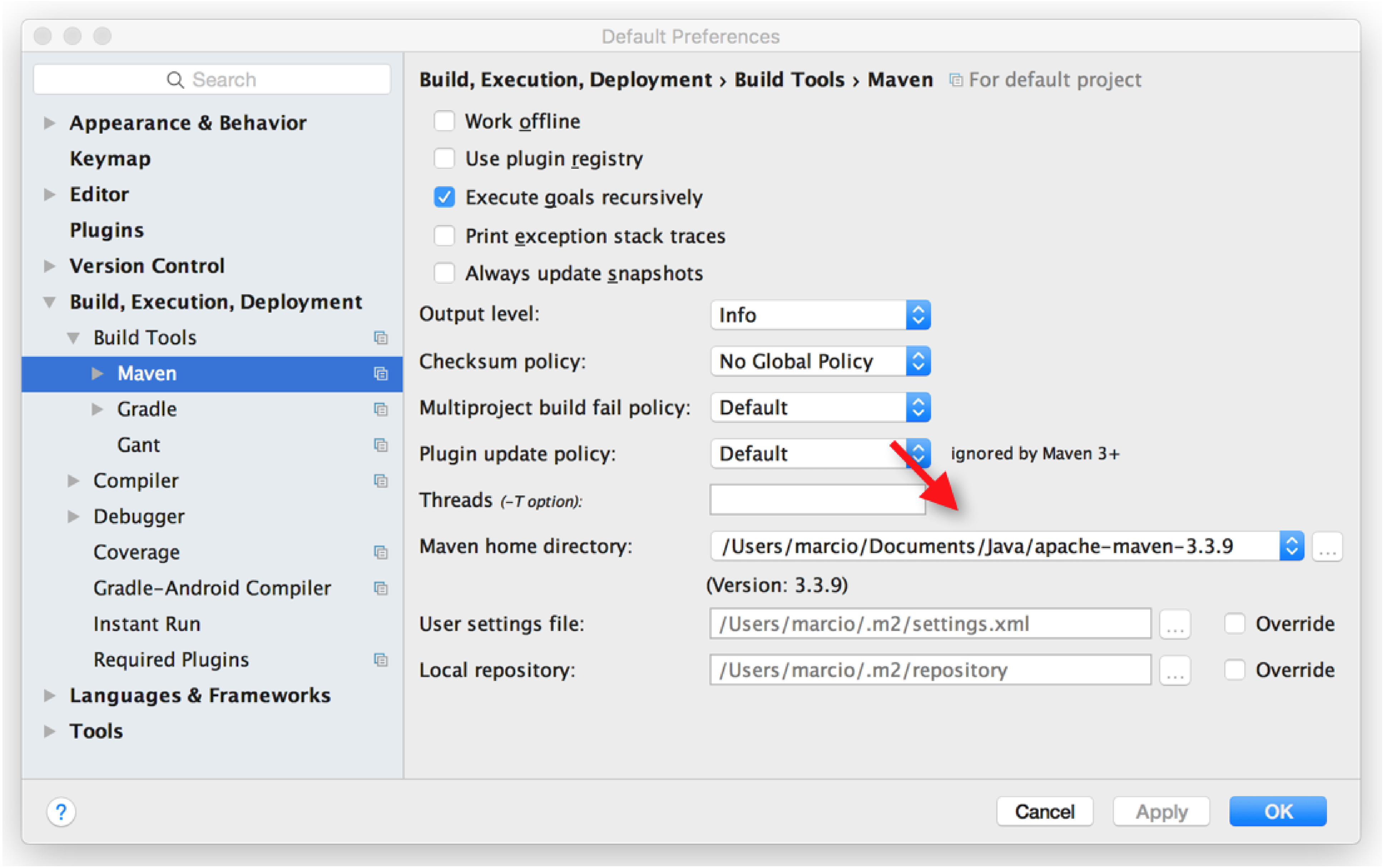
7. Click *Apply* and *OK*

Configure *Maven*

1. Use the same welcome window of *IntelliJ* and click the *Configure* link present at the bottom of the screen:



2. Select *Preferences (Settings on Windows)*. The preferences window will appear.
3. Choose the option *Build, Execution, Deployment* > *Build Tools* > *Maven* from the tree.
4. Change the *Maven* home directory to point to your installation, instead of the bundled version. If you forgot your folder, take a look at the content of your *M2_HOME* environment variable. *IntelliJ* also detects existing *Maven* installations, and you can found the list by clicking on the combo box.



Build, Execution, Deployment › Build Tools › Maven For default project

- [ ] Work offline
- [ ] Use plugin registry
- [x] Execute goals recursively
- [ ] Print exception stack traces
- [ ] Always update snapshots

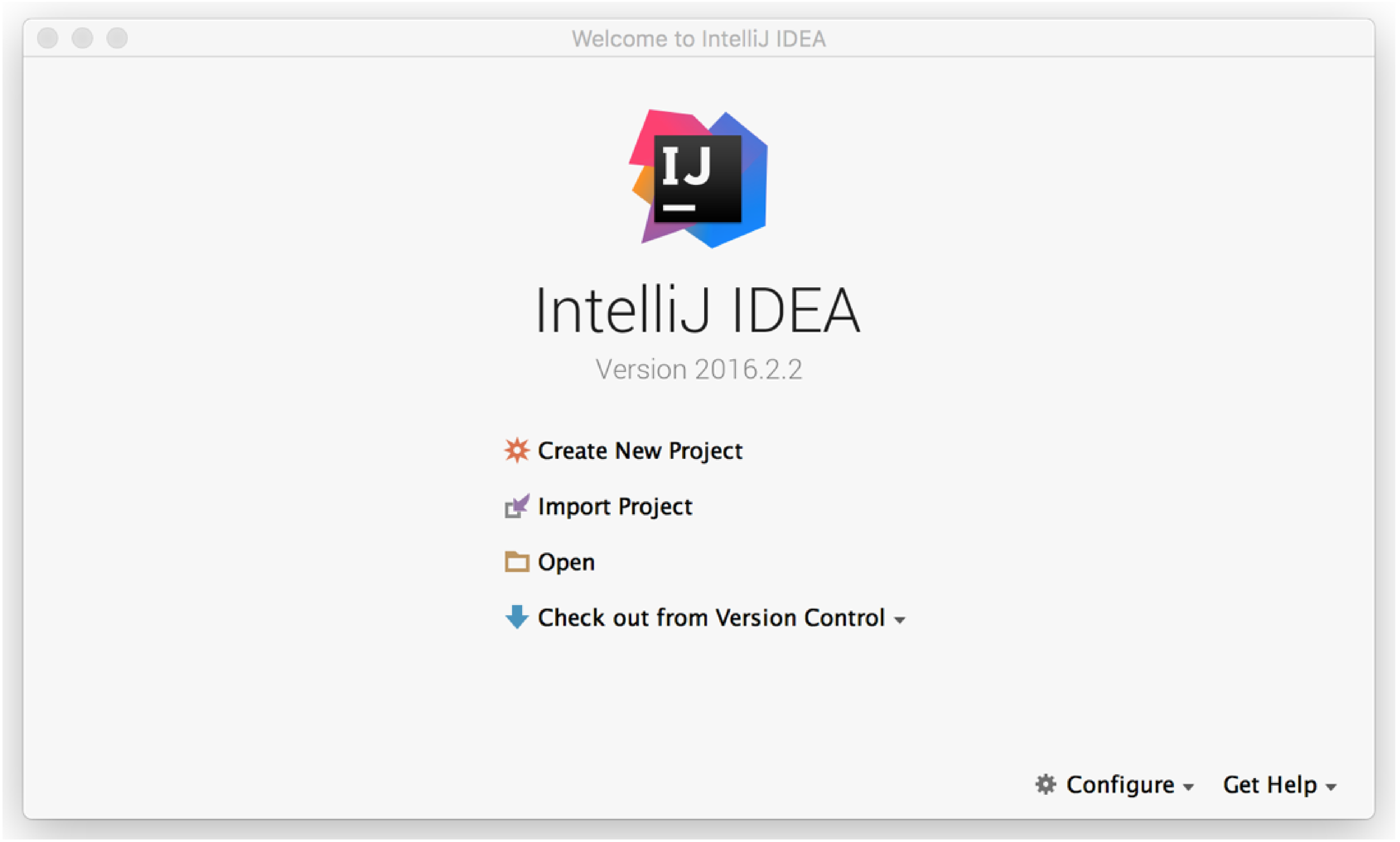
Output level:	Info	
Checksum policy:	No Global Policy	
Multiproject build fail policy:	Default	
Plugin update policy:	Default	ignored by Maven 3+
Threads *(-T option)*:		
Maven home directory:	/Users/marcio/Documents/Java/apache-maven-3.3.9 (Version: 3.3.9)	
User settings file:	/Users/marcio/.m2/settings.xml	[ ] Override
Local repository:	/Users/marcio/.m2/repository	[ ] Override

Note that *IntelliJ* updated the default location for the user settings file, as well as your local repository location.

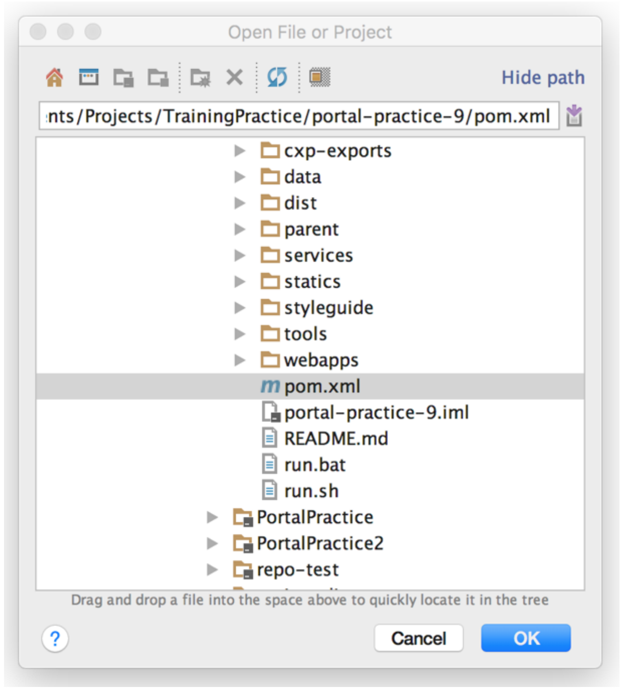
Configuring a CXP project

Now that you have the JDK and *Maven* configured in your IDE, it is time to open your CXP project.

1. Start your *IntelliJ* IDE (or use the one that is already started).
2. Click the *open* link present on the initial screen.



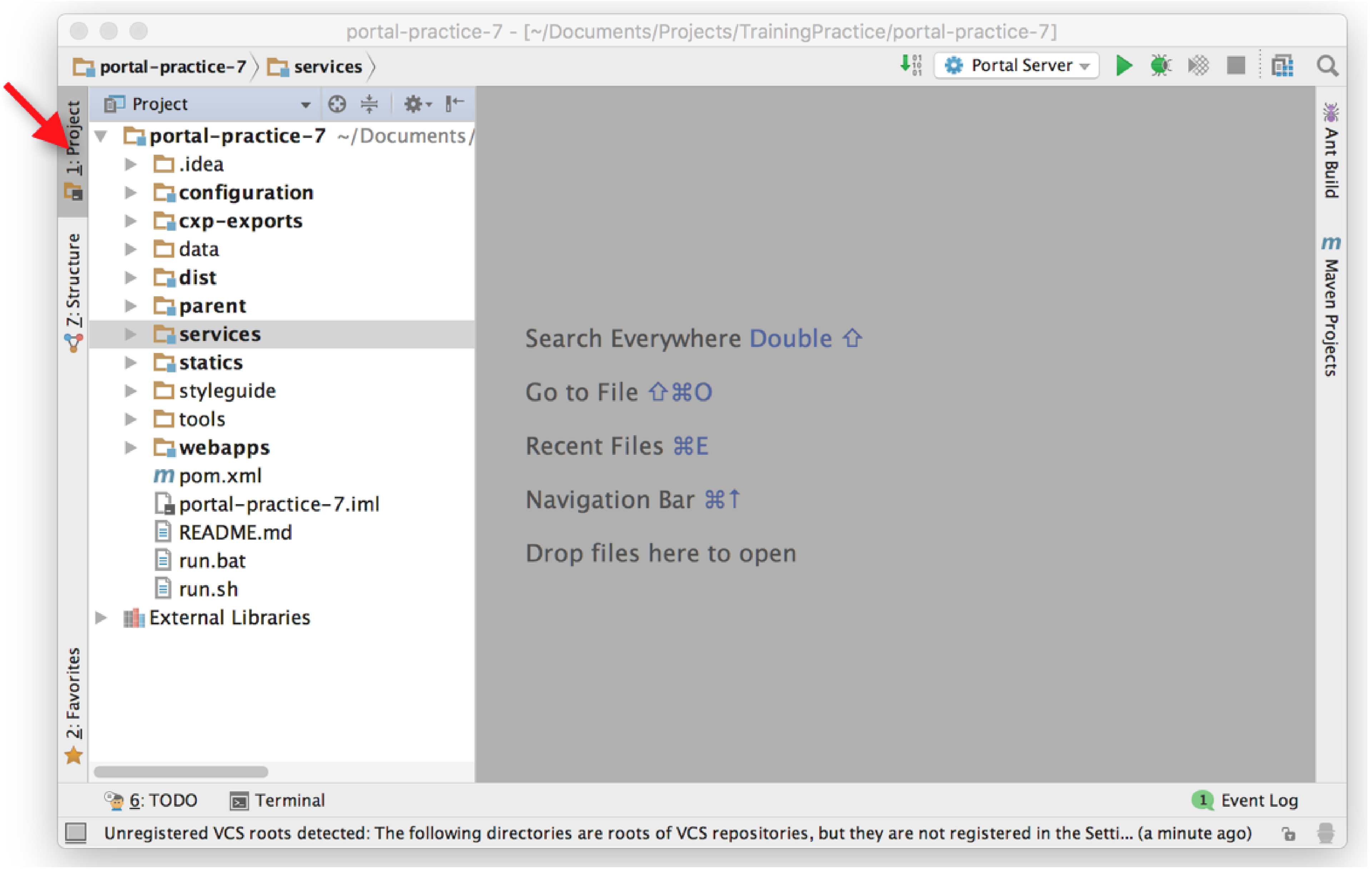
3. A file selection dialog will appear. Select the *pom.xml* file located at the root of your CXP project.


Open File or Project
Hide path
Drag and drop a file into the space above to quickly locate it in the tree
Cancel
OK

```
nts/Projects/TrainingPractice/portal-practice-9/pom.xml
```

- cxp-exports
- data
- dist
- parent
- services
- statics
- styleguide
- tools
- webapps
- pom.xml
- portal-practice-9.iml
- README.md
- run.bat
- run.sh
- PortalPractice
- PortalPractice2
- repo-test

4. Click *OK*. A set of background tasks to configure and index your *Maven* project will start.
5. It will take some time (approx. 2 mins) to load the project, index all libraries, and download plugins. Finally, the structure below will be available:

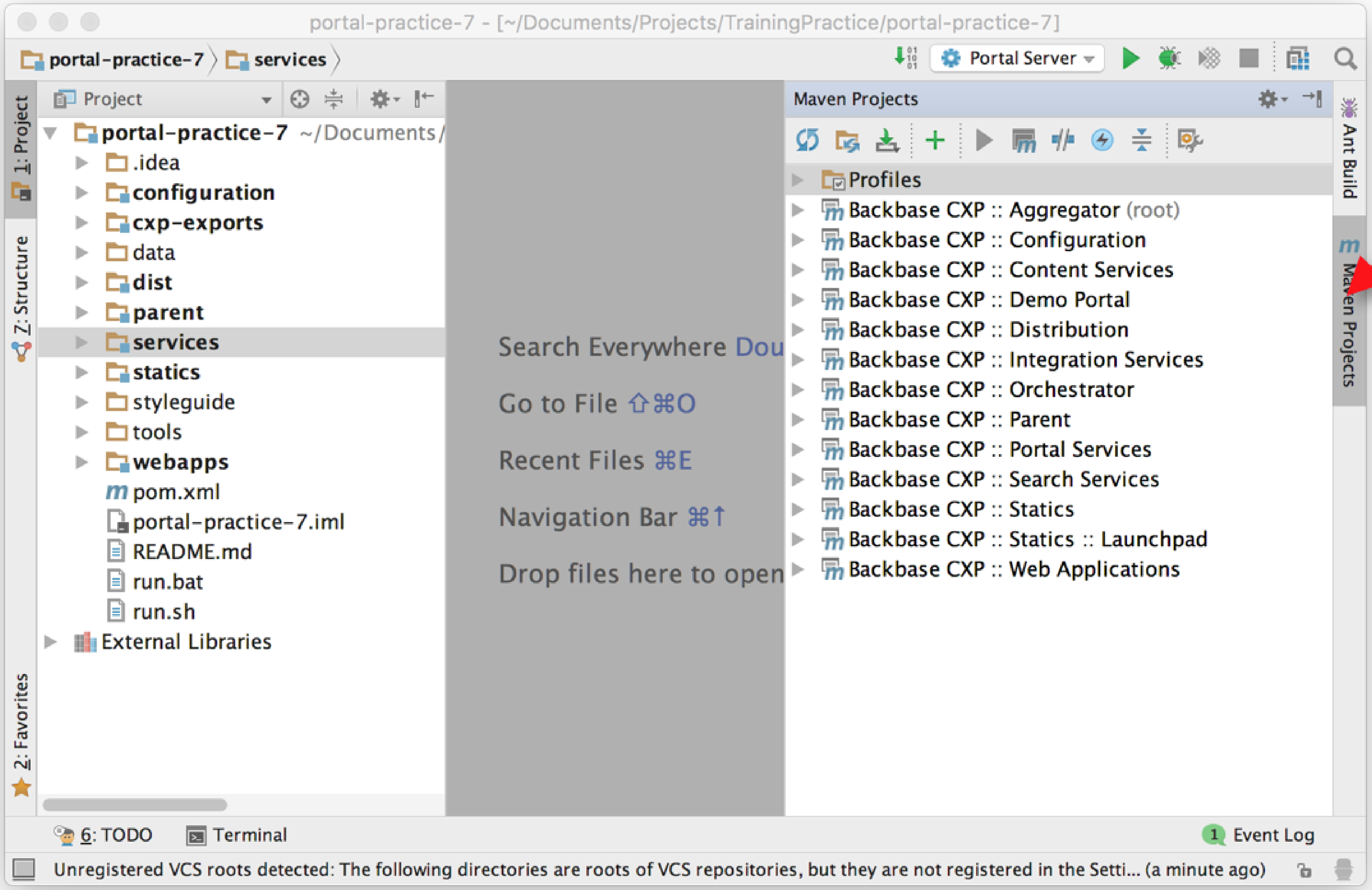


Unregistered VCS roots detected: The following directories are roots of VCS repositories, but they are not registered in the Setti... (a minute ago)

Now it is time to navigate through the project structure.

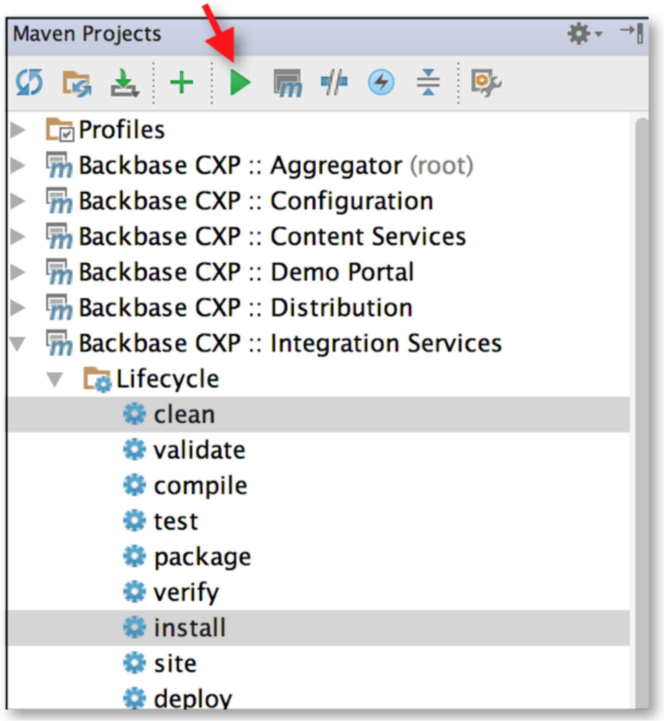
1. Click the located at the left bottom of your screen to toggle views. You will see several tabs (e.g. *Project*, *Maven*, *Terminal*, and so on).

2. Click the *Project* view (left-hand corner) and navigate to the project structure.
3. Expand your project and navigate through the project structure.
4. Click the *Maven projects* view (right-hand corner) and see all projects.


portal-practice-7 - [~/Documents/Projects/TrainingPractice/portal-practice-7]
portal-practice-7
services
Portal Server
Project
1: Project
Z: Structure
2: Favorites
portal-practice-7 ~/Documents/
.idea
configuration
cxp-exports
data
dist
parent
services
statics
styleguide
tools
webapps
pom.xml
portal-practice-7.iml
README.md
run.bat
run.sh
External Libraries
Search Everywhere Dou
Go to File ⇧⌘O
Recent Files ⌘E
Navigation Bar ⌘↑
Drop files here to open
Maven Projects
Profiles
Backbase CXP :: Aggregator (root)
Backbase CXP :: Configuration
Backbase CXP :: Content Services
Backbase CXP :: Demo Portal
Backbase CXP :: Distribution
Backbase CXP :: Integration Services
Backbase CXP :: Orchestrator
Backbase CXP :: Parent
Backbase CXP :: Portal Services
Backbase CXP :: Search Services
Backbase CXP :: Statics
Backbase CXP :: Statics :: Launchpad
Backbase CXP :: Web Applications
Ant Build
Maven Projects
6: TODO
Terminal
Event Log

Unregistered VCS roots detected: The following directories are roots of VCS repositories, but they are not registered in the Setti... (a minute ago)

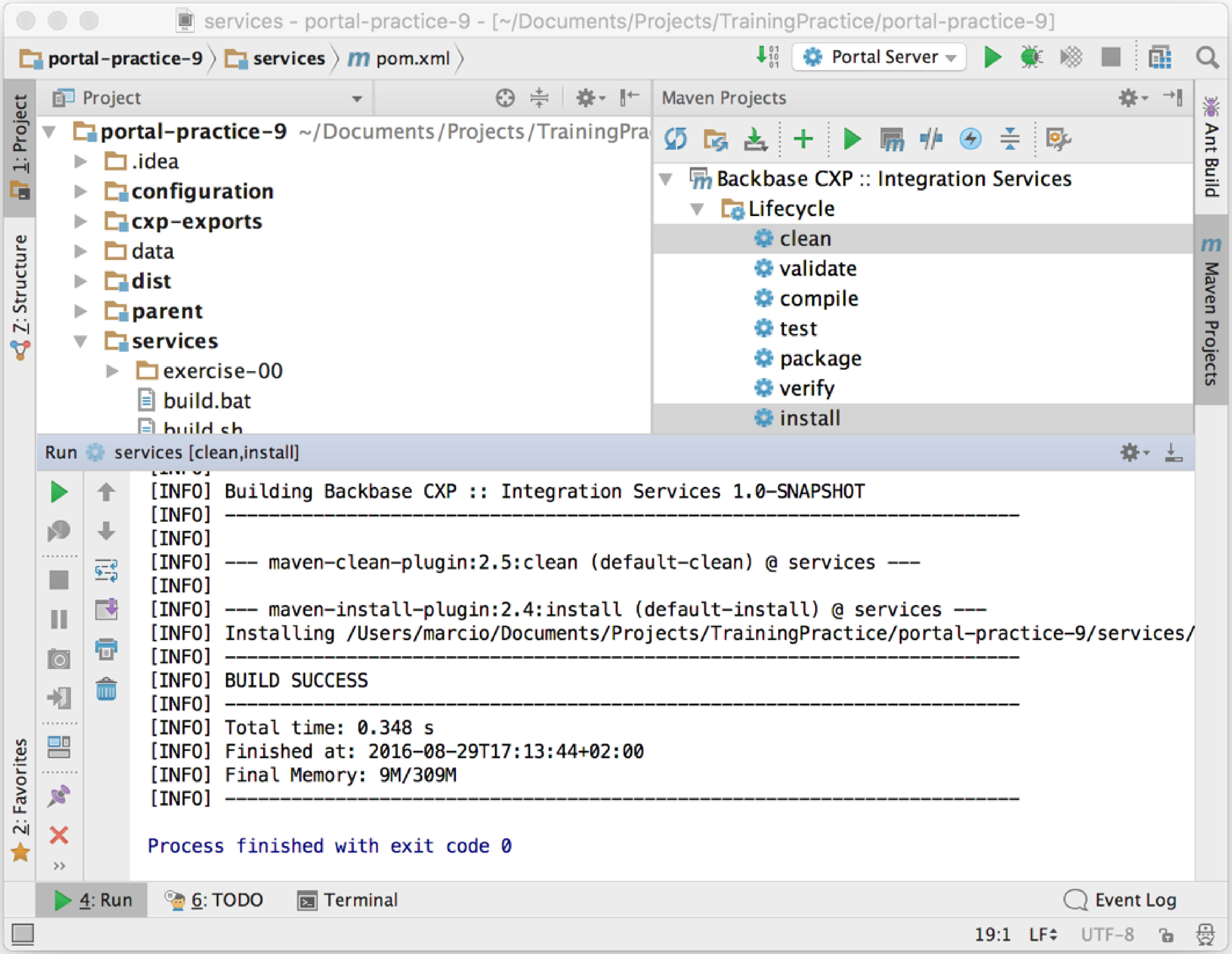
5. You should be able to run Maven goals from the IDE. For example:
    - Select *Backbase CXP :: Integration Services* project and expand the lifecycle element present in the tree.
    - Select the goals *clean* and *install* and click the play ▶ on the toolbar. For multiple goals, use the *CTRL* key on Windows or *cmd* ⌘ key on Mac.



Maven Projects

- Profiles
- Backbase CXP :: Aggregator (root)
- Backbase CXP :: Configuration
- Backbase CXP :: Content Services
- Backbase CXP :: Demo Portal
- Backbase CXP :: Distribution
- Backbase CXP :: Integration Services
    - Lifecycle
        - clean
        - validate
        - compile
        - test
        - package
        - verify
        - install
        - site
        - deploy

    - You will see the console with the execution results. The project must be compiled without errors.



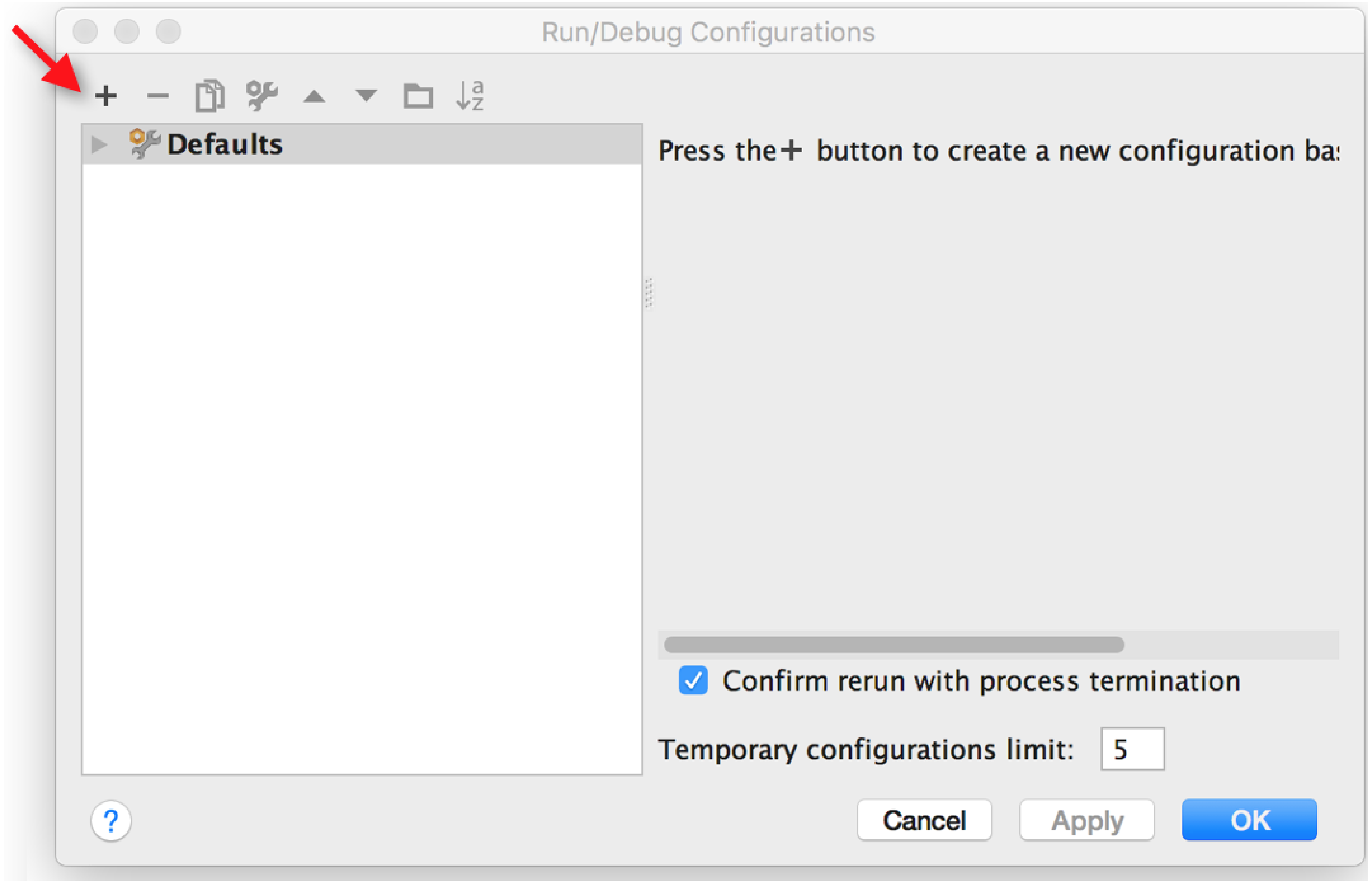
```
[INFO] Building Backbase CXP :: Integration Services 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ services ---
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ services ---
[INFO] Installing /Users/marcio/Documents/Projects/TrainingPractice/portal-practice-9/services/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.348 s
[INFO] Finished at: 2016-08-29T17:13:44+02:00
[INFO] Final Memory: 9M/309M
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

Start and stop CXP servers using your IDE

As you could see in the how-to guide used for the project preparation, the portal could be started by running a shell script on your local machine (`run.sh` in Unix-based machines, and `run.bat` for Windows). However, developers need more control over the log messages to check errors and information messages, as well as starting servers using the debug mode.

During this step, you will configure the shortcuts to CXP web apps projects.

1. Select *Run* > *Edit Configurations* from the *IntelliJ* menu. A dialog window will appear.

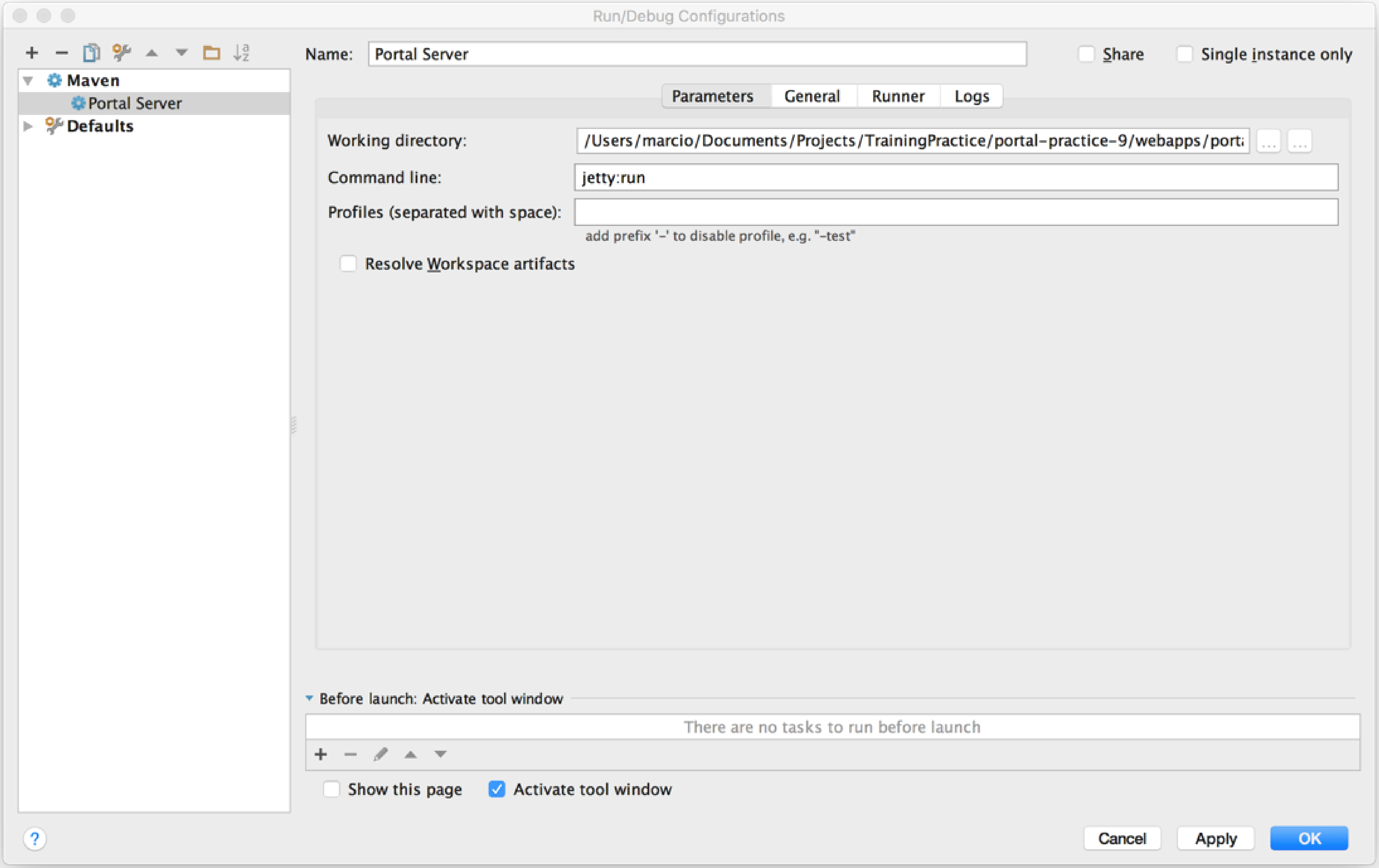


Press the + button to create a new configuration ba

☑ Confirm rerun with process termination

Temporary configurations limit: 5

2. Click the + to create a new configuration and select *Maven*.



Name: Portal Server
Share
Single instance only

Working directory: /Users/marcio/Documents/Projects/TrainingPractice/portal-practice-9/webapps/porta
Command line: jetty:run
Profiles (separated with space):
add prefix '-' to disable profile, e.g. "-test"
Resolve Workspace artifacts

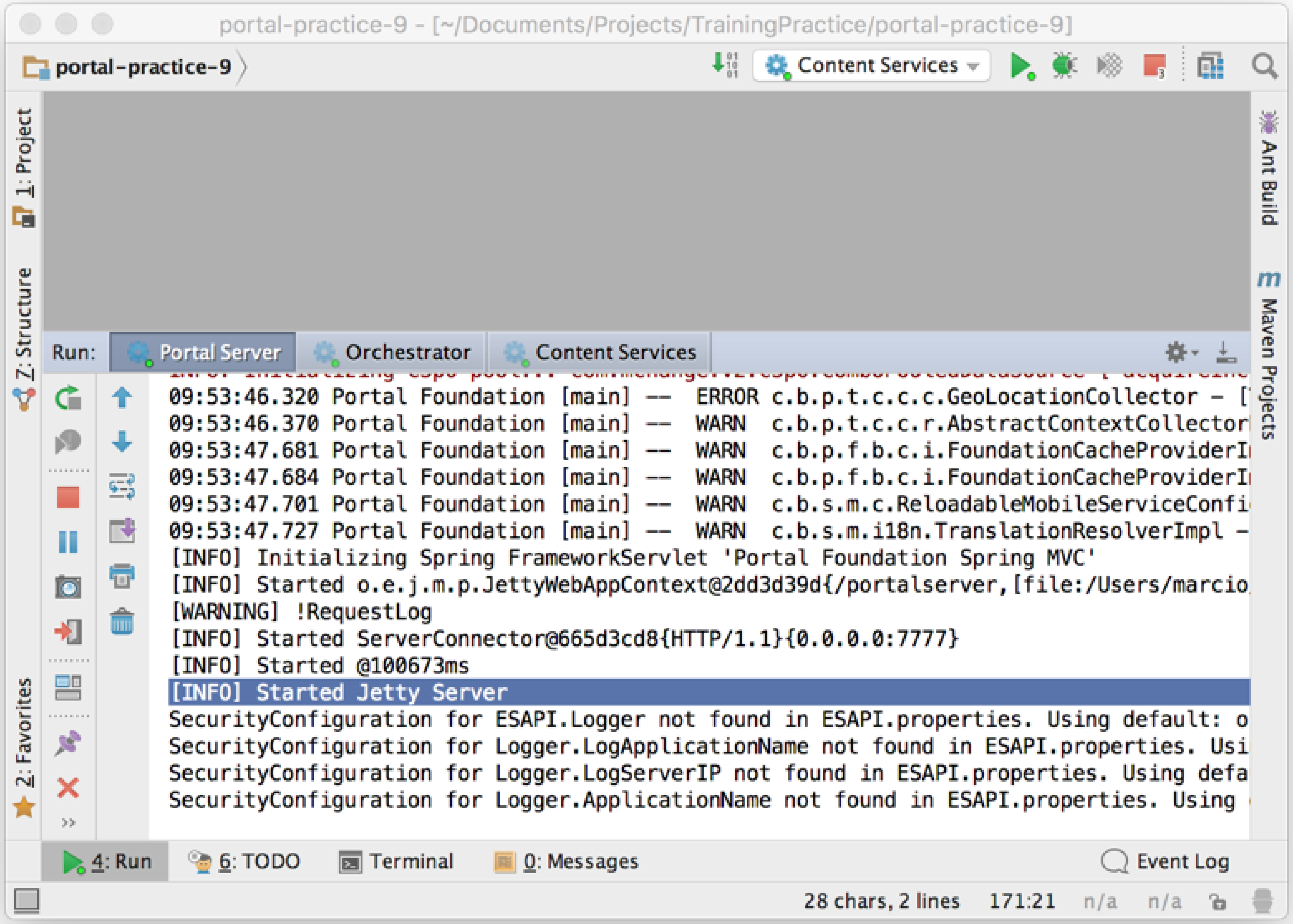
Before launch: Activate tool window
There are no tasks to run before launch
Show this page
Activate tool window

3. Give a name (e.g. *Portal Server*) to this shortcut. For the working directory, select the folder `webapps/portalserver`
4. For the command line field, inform the goal `jetty:run`. This goal will start a jetty JEE container in your local machine for the selected project.

   If you want to understand better how each server is started and the different startup options, take a look at the shell script located in the root folder of your project (`run.sh` for Unix-based systems, `run.bat` for Windows).

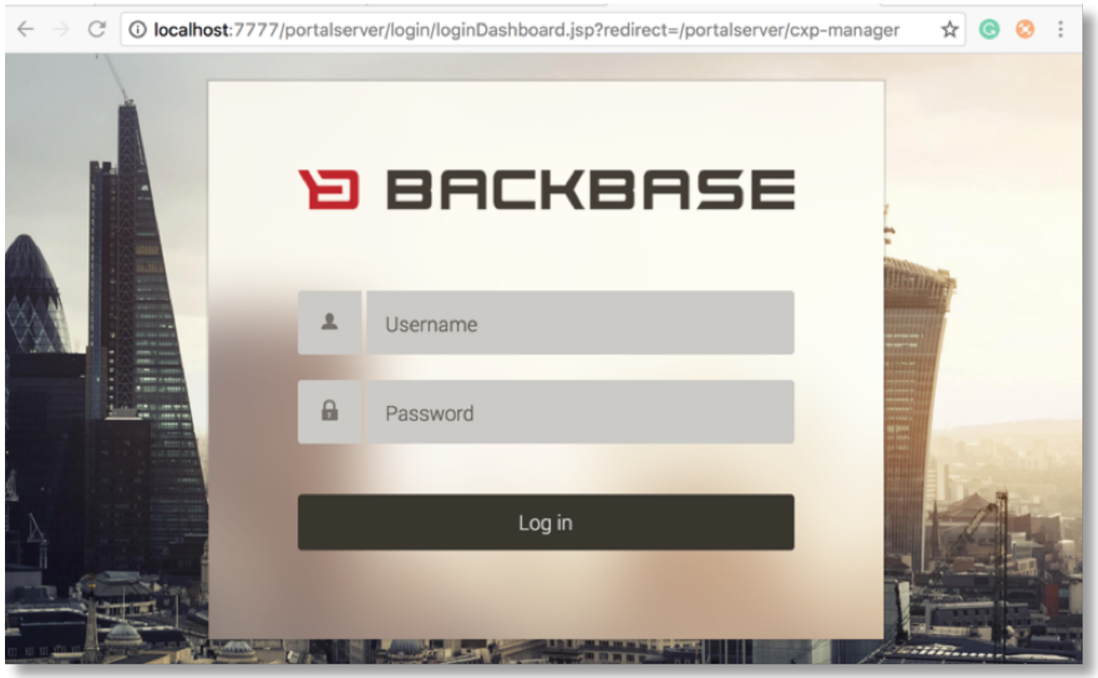
5. Click *Apply* and *OK.*
6. Execute steps from 2 to 5 to configure the shortcut for the remaining servers: content services (`webapps/contentservices`) and the orchestrator (`webapps/orchestrator`)
7. Now, you will see the set of shortcuts to start your servers in normal or debug mode in your toolbar

8. Start each server in normal or debug mode. Each server has its console so that you can follow all messages separately. From the same console window, you can restart and stop the server.
9. These actions will spawn three new console windows, running Portal Server, Content Services and Orchestrator on *Jetty*. Wait until you see the message "*Started Jetty Server*" in each of the three windows.



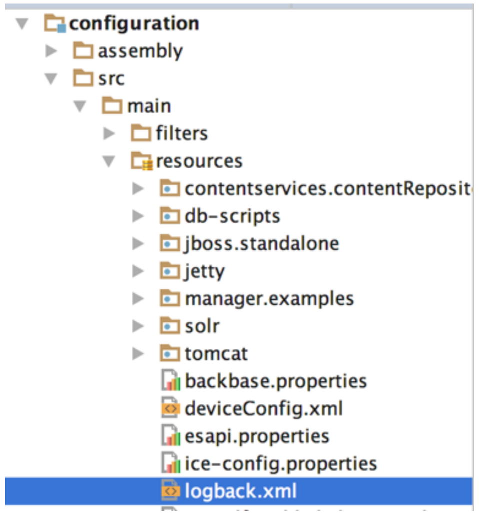
```
09:53:46.320 Portal Foundation [main] --  ERROR c.b.p.t.c.c.c.GeoLocationCollector - [
09:53:46.370 Portal Foundation [main] --  WARN  c.b.p.t.c.c.r.AbstractContextCollector
09:53:47.681 Portal Foundation [main] --  WARN  c.b.p.f.b.c.i.FoundationCacheProviderI
09:53:47.684 Portal Foundation [main] --  WARN  c.b.p.f.b.c.i.FoundationCacheProviderI
09:53:47.701 Portal Foundation [main] --  WARN  c.b.s.m.c.ReloadableMobileServiceConfi
09:53:47.727 Portal Foundation [main] --  WARN  c.b.s.m.i18n.TranslationResolverImpl -
[INFO] Initializing Spring FrameworkServlet 'Portal Foundation Spring MVC'
[INFO] Started o.e.j.m.p.JettyWebAppContext@2dd3d39d{/portalserver,[file:/Users/marcio
[WARNING] !RequestLog
[INFO] Started ServerConnector@665d3cd8{HTTP/1.1}{0.0.0.0:7777}
[INFO] Started @100673ms
[INFO] Started Jetty Server
SecurityConfiguration for ESAPI.Logger not found in ESAPI.properties. Using default: o
SecurityConfiguration for Logger.LogApplicationName not found in ESAPI.properties. Usi
SecurityConfiguration for Logger.LogServerIP not found in ESAPI.properties. Using defa
SecurityConfiguration for Logger.ApplicationName not found in ESAPI.properties. Using
```

10. Open the CXP manager using your preferred browser: http://localhost:7777/portalserver/cxp-manager
11. You should see the welcome login screen. Congrats! You have your CXP development environment set.

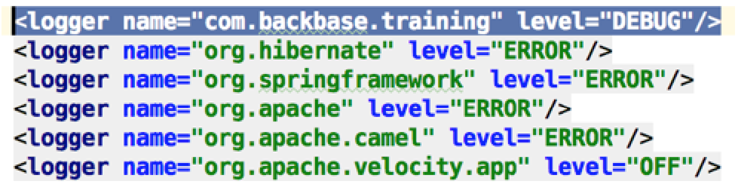

localhost:7777/portalserver/login/loginDashboard.jsp?redirect=/portalserver/cxp-manager
BACKBASE
Username
Password
Log in

Configuring the logging level of your application

Logging is a recurrent and critical feature for all applications. Backbase CXP ships with a default *Logback* configuration file called `logback.xml`. This file is located under the *configuration* subproject, in the folder src/main/resource, as can be seen in the figure below:

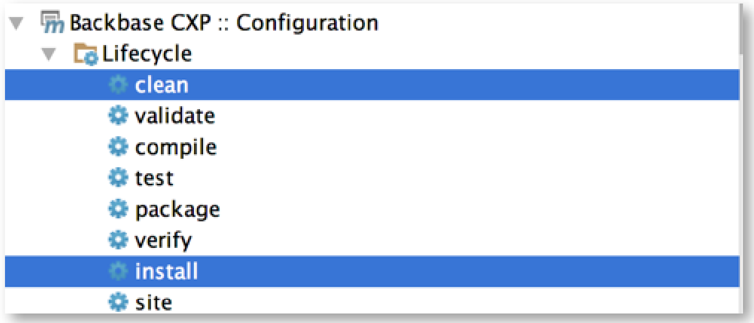


1. Open the `logback.xml` file and navigate through the XML structure. You will find a set of *appenders* and *loggers*. *Appenders* define how logging messages will be shown or stored (e.g. console, flat files, databases). *Loggers* describe the severity level of logging messages (e.g. debug, info, warning, error).
2. If you are using CXP for the first time (attending a training, doing experiments), we recommend adding a new *logger* for your packages (e.g. `com.backbase.training`), with the logging level set to *debug*. It states that all classes that belong to this package are eligible to debugging logging to the console. The figure below shows an example of modified `logback.xml`, with all classes and sub-packages belonging to the package `com.backbase.training` set to the *debug* level.



```
<logger name="com.backbase.training" level="DEBUG"/>
<logger name="org.hibernate" level="ERROR"/>
<logger name="org.springframework" level="ERROR"/>
<logger name="org.apache" level="ERROR"/>
<logger name="org.apache.camel" level="ERROR"/>
<logger name="org.apache.velocity.app" level="OFF"/>
```

3. Do not forget to compile your *configuration* project and restart the portal after each change. To do so, open your console window, go to the configuration folder and type `mvn clean install`.
   You can also compile the configuration project using your IDE (see the *Maven Projects* tab in your *IntelliJ* interface).



For more information on how to customize your logging files, take a look at the CXP documentation at My Backbase.

This entry was posted in Integration, Launchpad on December 7, 2016 [http://wordpress-awsprod.extranet.backbasecloud.com/?p=5475] by Marcio Fuckner.

About Marcio Fuckner

Marcio is a Trainer at the Backbase Academy team. His role is sharing best practices related to the software development lifecycle of financial institutions using Backbase products.

View all posts by Marcio Fuckner →