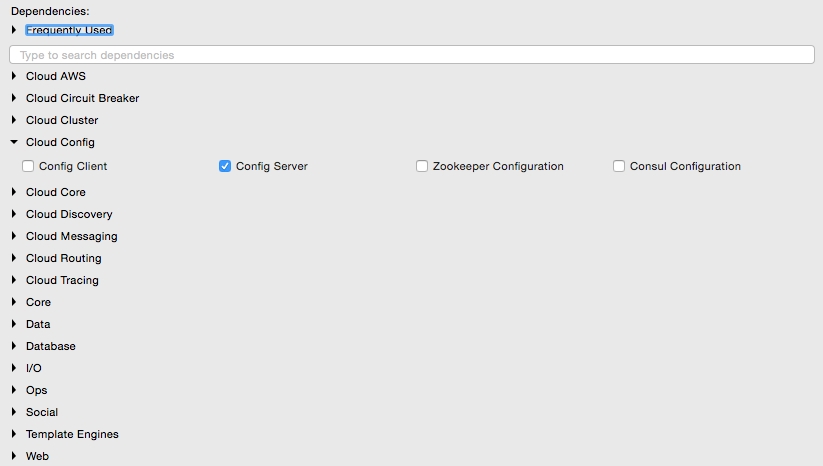
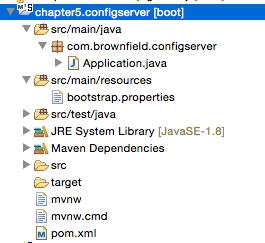
Lab 11 - Setting Up the Config Server

The following steps need to be followed to create a new Config server using STS:

* Create a new **Spring Starter Project**, and select **Config Server** and **Actuator** as shown in the following diagram:
* 
* Set up a Git repository. This can be done by pointing to a remote Git configuration repository like the one at <https://github.com/spring-cloud-samples/config-repo>. This URL is an indicative one, a Git repository used by the Spring Cloud examples. We will have to use our own Git repository instead.
* Alternately, a local filesystem-based Git repository can be used. In a real production scenario, an external Git is recommended. The Config server in this chapter will use a local filesystem-based Git repository for demonstration purposes.
* Enter the commands listed next to set up a local Git repository:
* **$ cd $HOME**  
  **$ mkdir config-repo**  
  **$ cd config-repo**  
  **$ git init .**  
  **$ echo message : helloworld > application.properties**  
  **$ git add -A .**  
  **$ git commit -m "Added sample application.properties"**
* This code snippet creates a new Git repository on the local filesystem. A property file named application.properties with a message property and value helloworld is also created.
* The file application.properties is created for demonstration purposes. We will change this in the subsequent sections.
* The next step is to change the configuration in the Config server to use the Git repository created in the previous step. In order to do this, rename the file application.properties to bootstrap.properties:
* 
* Edit the contents of the new bootstrap.properties file to match the following:
* server.port=8888  
  spring.cloud.config.server.git.uri: file://${user.home}/config-repo
* Port 8888 is the default port for the Config server. Even without configuring server.port, the Config server should bind to 8888. In the Windows environment, an extra / is required in the file URL.
* Optionally, rename the default package of the auto-generated Application.java from com.example to com.brownfield.configserver. Add @EnableConfigServer in Application.java:
* @EnableConfigServer  
  @SpringBootApplication  
  public class ConfigserverApplication {
* Run the Config server by right-clicking on the project, and running it as a Spring Boot app.
* Visit http://localhost:8888/env to see whether the server is running. If everything is fine, this will list all environment configurations. Note that /env is an actuator endpoint.
* Check http://localhost:8888/application/default/master to see the properties specific to application.properties, which were added in the earlier step. The browser will display the properties configured in application.properties. The browser should display contents similar to the following:
* {"name":"application","profiles":["default"],"label":"master","version":"6046fd2ff4fa09d3843767660d963866ffcc7