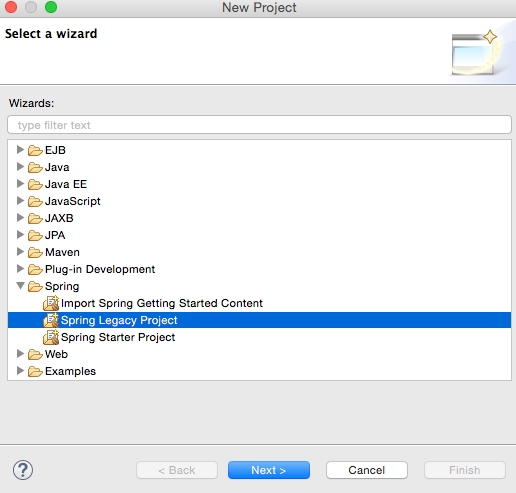
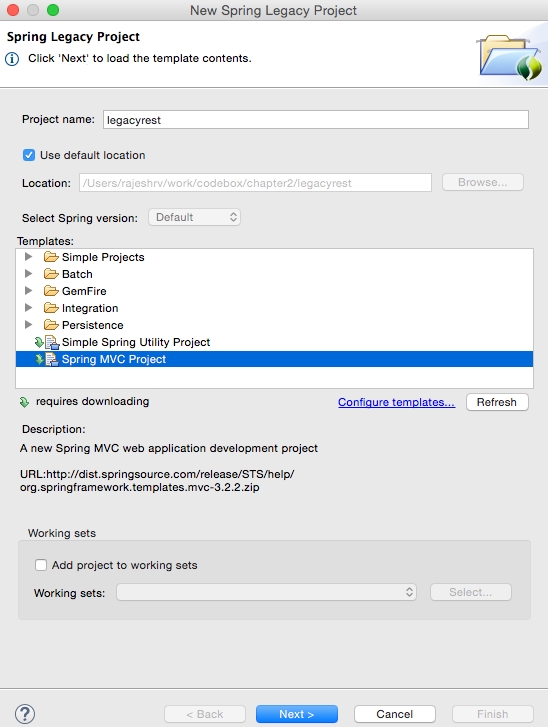
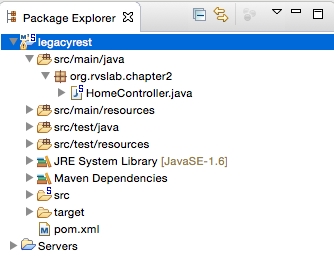
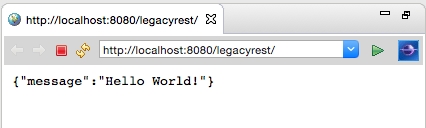
The following are the steps to develop the first RESTful service:

* Start STS and set a workspace of choice for this project.
* Navigate to **File** | **New** | **Project**.
* Select **Spring Legacy Project** as shown in the following screenshot and click on **Next**:
* 
* Select **Spring MVC Project** as shown in the following diagram and click on **Next**:
* 
* Select a top-level package name of choice. This example uses org.rvslab.chapter2.legacyrest as the top-level package.
* Then, click on **Finish**.
* This will create a project in the STS workspace with the name legacyrest.
* Before proceeding further, pom.xml needs editing.
* Change the Spring version to 4.2.6.RELEASE, as follows:
* Copy
* <org.springframework-version>4.2.6.RELEASE</org.springframework-version>
* Add **Jackson** dependencies in the pom.xml file for JSON-to-POJO and POJO-to-JSON conversions. Note that the 2.\*.\* version is used to ensure compatibility with Spring 4.
* Copy
* <dependency>  
   <groupId>com.fasterxml.jackson.core</groupId>  
   <artifactId>jackson-databind</artifactId>  
   <version>2.6.4</version>  
  </dependency>
* Some Java code needs to be added. In **Java Resources**, under **legacyrest**, expand the package and open the default **HomeController.java** file:
* 
* The default implementation is targeted more towards the MVC project. Rewriting HomeController.java to return a JSON value in response to the REST call will do the trick. The resulting HomeController.java file will look similar to the following:
* Copy
* @RestController  
  public class HomeController {  
   @RequestMapping("/")  
   public Greet sayHello(){  
   return new Greet("Hello World!");  
   }  
  }  
  class Greet {   
   private String message;  
   public Greet(String message) {  
   this.message = message;  
   }  
   //add getter and setter  
  }
* Examining the code, there are now two classes:
  + Greet: This is a simple Java class with getters and setters to represent a data object. There is only one attribute in the Greet class, which is message.
  + HomeController.java: This is nothing but a Spring controller REST endpoint to handle HTTP requests.
* Note that the annotation used in HomeController is @RestController, which automatically injects @Controller and @ResponseBody and has the same effect as the following code:
* Copy
* @Controller  
  @ResponseBody  
  public class HomeController { }
* The project can now be run by right-clicking on **legacyrest**, navigating to **Run As** | **Run On Server**, and then selecting the default server (**Pivotal tc Server Developer Edition v3.1**) that comes along with STS.
* This should automatically start the server and deploy the web application on the TC server.
* If the server started properly, the following message will appear in the console:
* Copy
* **INFO : org.springframework.web.servlet.DispatcherServlet - FrameworkServlet 'appServlet': initialization completed in 906 ms**  
  **May 08, 2016 8:22:48 PM org.apache.catalina.startup.Catalina start**  
  **INFO: Server startup in 2289 ms**
* If everything is fine, STS will open a browser window to http://localhost:8080/legacyrest/ and display the JSON object as shown in the browser. Right-click on and navigate to **legacyrest** | **Properties** | **Web Project Settings** and review **Context Root** to identify the context root of the web application:
* 

The alternate build option is to use Maven. Right-click on the project and navigate to **Run As** | **Maven install**. This will generate chapter2-1.0.0-BUILD-SNAPSHOT.war under the target folder. This war is deployable in any servlet container such as Tomcat, JBoss, and so on.