# Lab MVC: Spring Rest

## Objectives

In this lab, you will use Spring MVC to implement a greeting Restful web service that will accept HTTP GET requests at: http://localhost:8080/greeting.

The service returns a web page that contains the greeting “Hello, World!” The request can include an optional name parameter in the query string like this <http://localhost:8080/greeting?name=User>. The name parameter value overrides the default value of World,

## Instructions

### Step 1 Create the Spring Boot project

1. Go to https://start.spring.io.
2. We will be using the Spring Web starter. All we need is the Spring Web dependency

Graphical user interface, text, application

Description automatically generated

### Step 2: Ensure Maven is installed

1. Text

   Description automatically generatedCheck to see Maven is installed by opening a command window and executing “mvn -version. You should see the following

### Step 3: Create the project

1. Create a new eclipse workspace
2. Graphical user interface, text, application

   Description automatically generatedUse the file import option to import the unzipped director as a Maven project.
3. Graphical user interface, text, application

   Description automatically generatedOne the project has been imported; Maven will build all the dependencies. This may take a while but eventually your project should look like this.
4. Text

   Description automatically generatedRun the DataApplication.java main() method as a Java Application to ensure that everything is working.

1. Graphical user interface, text, application, email

   Description automatically generatedThe web service is up and running, but we haven’t added any code so it does nothing. You can see this by going to the URL for the service.
2. Once you have verified that the service runs, shut it down with the red square on the console

Graphical user interface, text

Description automatically generated

### Step 3: Create the model class

1. The resources that are being referenced by the web service are Greeting entities.
2. These are Pojos that need to know nothing about the controller or the views that are being used.
3. Text

   Description automatically generatedNotice that each resource created will have an id, a convention that is consistent with the REST approach

### Step 4: Create the Controller

1. In MVC, the controller handles incoming requests and routes them to the correct model objects to me managed.
2. Spring Web provides a number of annotations to define how a controller class is to be managed.

Text

Description automatically generated with medium confidence

1. The @RestController annotation tells Spring that this class is the controller.
2. The view object is normally what is presented to the client. In this case the view is just a string so the template static variable is our model. In a more sophisticate application, we would use something like a templating engine like Thymeleaf to serve up HTML pages.
3. The @GetMapping(URL) maps specific URLs to processing objects. In this case the effect is to create a new model Greeting object and return it.
4. The @RequestParam() annotation looks for an argument in the URL and if there is one, assigns it to the string that appears as the value of the name parameter in the URL. If there is no parameter, it defaults to “World”
5. Since we are not using an HTML page, what we will see is just a JSon object representing the Greeting resource.

Graphical user interface, text, application

Description automatically generated

1. Reloading the page shows that a new request in generated and a new Greeting resource created in response.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Graphical user interface, text, application, chat or text message

   Description automatically generated Calling with a parameter produces the following

### Step:5 Change the URL

1. Shut down the server using the red square button
2. Text

   Description automatically generated with medium confidenceReplace the mapping to /greeting to anything else, like below where “/howdy” is being used
3. Restart the application and test it out

Graphical user interface, text, application, chat or text message

Description automatically generated