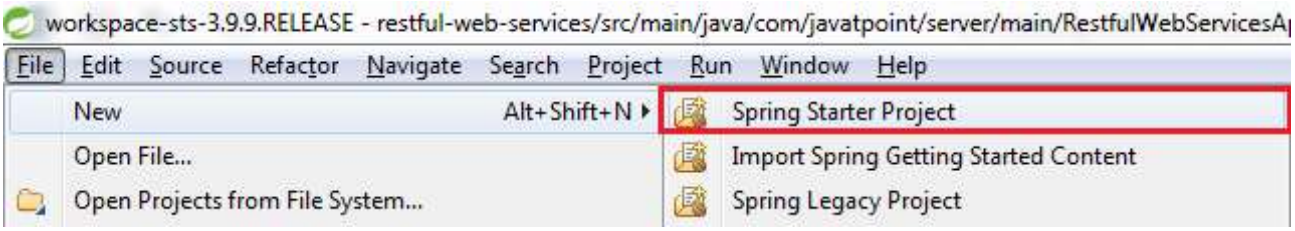
Initializing a RESTful Web Services Project with Spring Boot

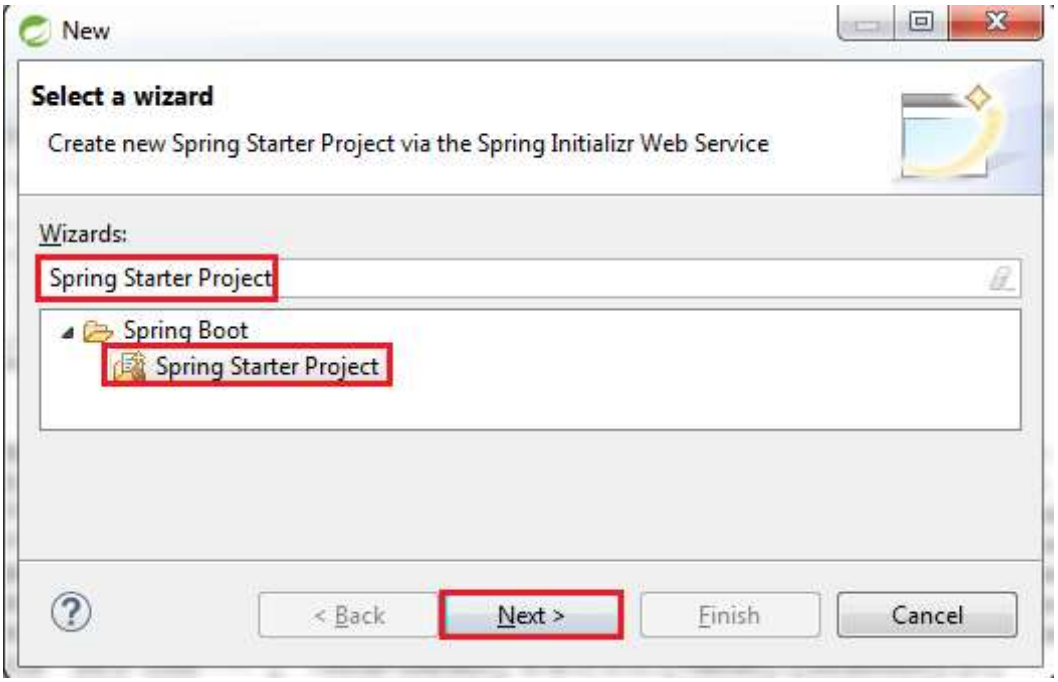
Step 1: Download the **Spring Tool Suite (STS)** from https://spring.io/tools3/sts/all and extract it.

Step 2: Launch the **STS**.

Step 3: Click on **File menu -> New -> Spring Starter Project ->**



If the **Spring Starter Project** is not enlisted, then click on **Other** at the bottom of the menu. A dialog box appears on the screen. Type **Spring Starter Project** in the **Wizards** text box and click on the **Next** button.



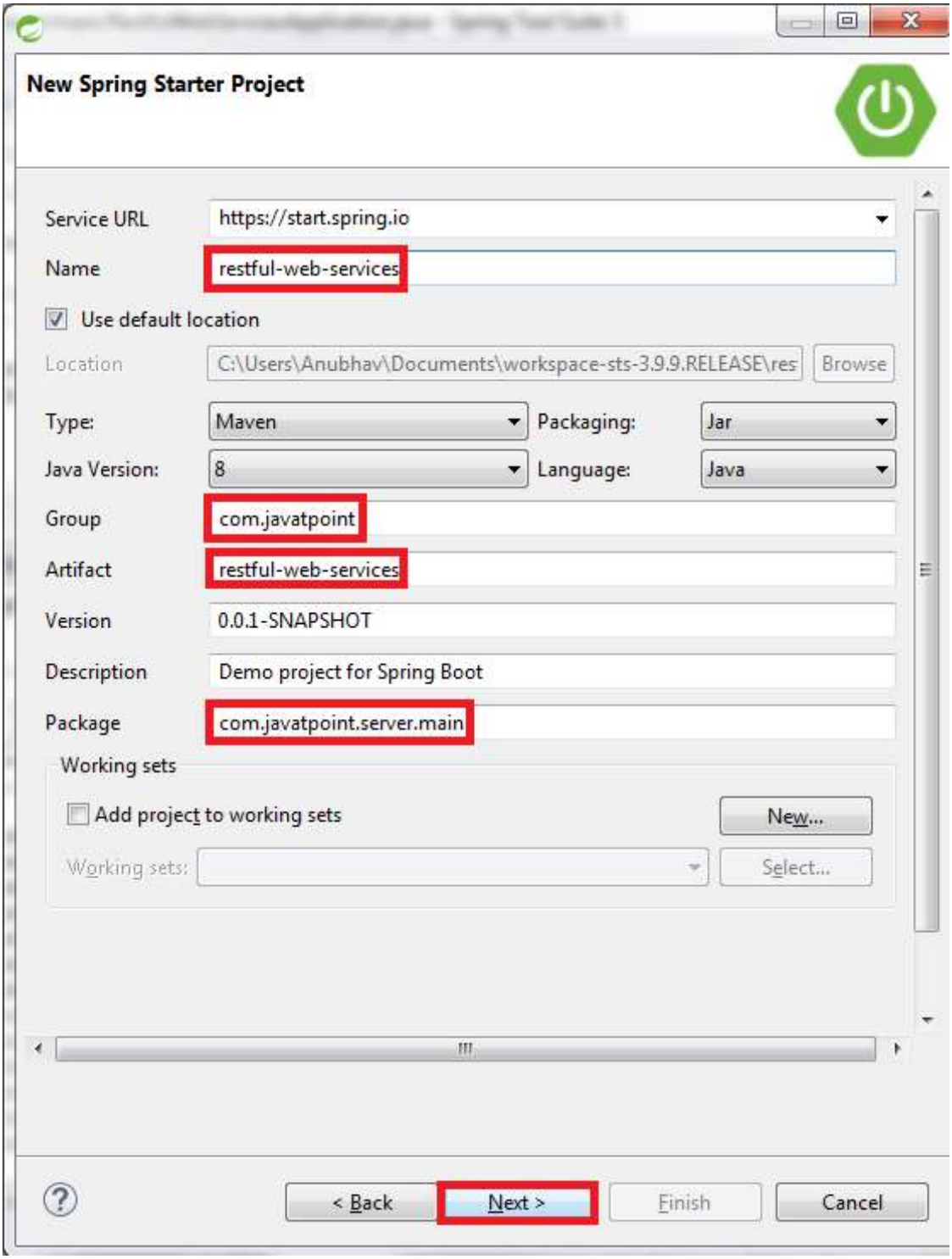
Step 4: provide the name, group, and package of the project. We have provided:

Name: **restful-web-services**

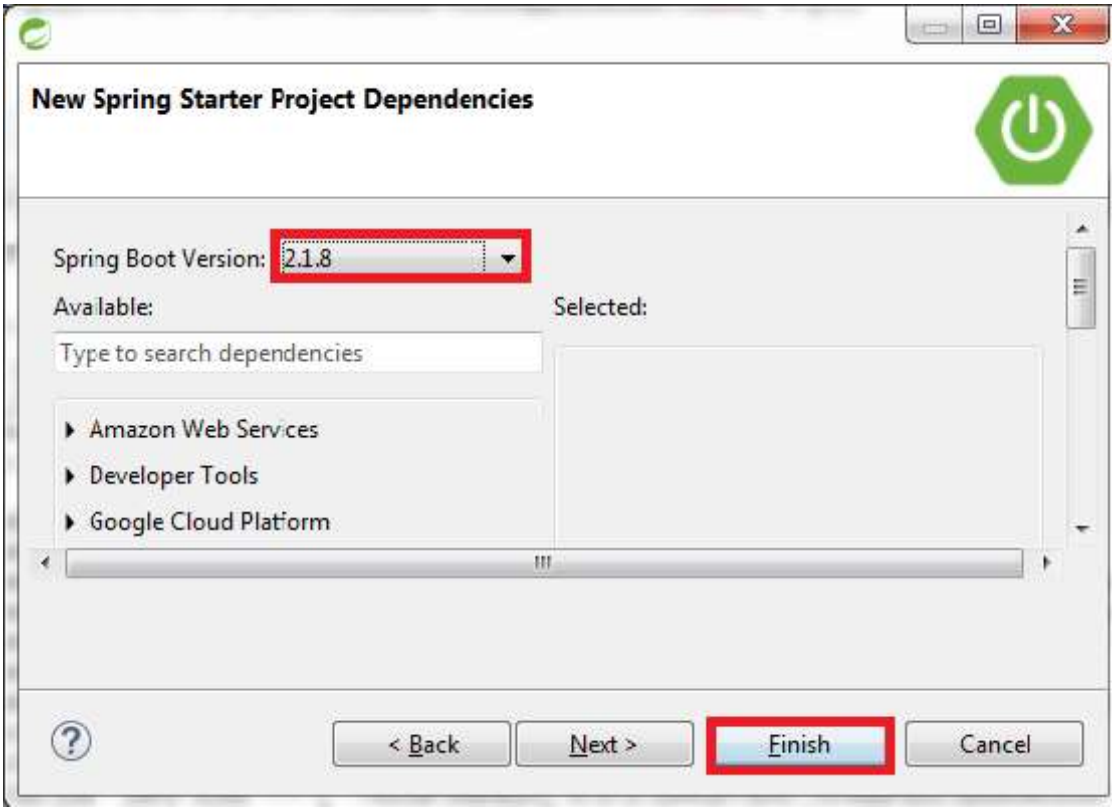
Group: **com.javatpoint**

Package: **com.javatpoint.server.main**

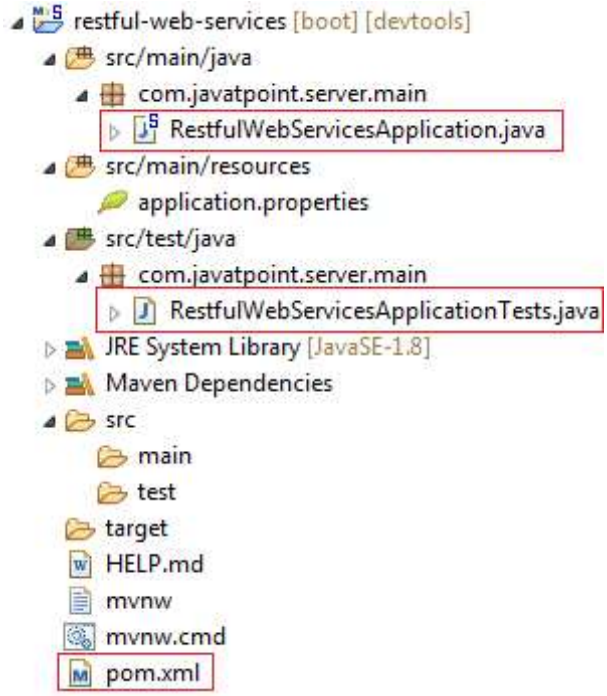
Click on the **Next** button.



Step 5: Choose the Spring Boot Version **2.1.8**.



Step 6: We can see the project structure in the project explorer window.



Step 7: Go to the Maven Repository https://mvnrepository.com/ and add **Spring Web MVC, Spring Boot DevTools, JPA,** and **H2** dependencies in the pom.xml. After adding the dependencies, the pom.xml file looks like the following:

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
```

```
                http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.1.8.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.javatpoint</groupId>
<artifactId>restful-web-services</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>restful-web-services</name>
<description>Demo project for Spring Boot</description>
<properties>
<java.version>1.8</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
```

```
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-devtools -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<scope>runtime</scope>
</dependency>
<!--  https://mvnrepository.com/artifact/org.hibernate.javax.persistence/hibernate-jpa-2.1-api -->
<dependency>
<groupId>org.hibernate.javax.persistence</groupId>
<artifactId>hibernate-jpa-2.1-api</artifactId>
<version>1.0.0.Final</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
<dependency>
<groupId>com.h2database</groupId>
<artifactId>h2</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.apache.maven</groupId>
<artifactId>maven-archiver</artifactId>
<version>2.5</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
```

```
</plugin>
</plugins>
</build>
</project>
```

Step 8: Now open the **RestfulWebServicesApplication.java** file and Run the file as Java Application.

```
package com.javatpoint.server.main;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class RestfulWebServicesApplication
{
public static void main(String[] args)
{
SpringApplication.run(RestfulWebServicesApplication.class, args);
}
}
```

It does not perform any service but ensures that the application is running properly.

Output

```
o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
c.j.s.m.RestfulWebServicesApplication    : Started RestfulWebServicesApplication in 2.88 seconds (JVM running for 4875.671)
.ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
```

Creating a Hello World Service

Step 1: Create a new class with the name **HelloWorldController** in the package **com.javatpoint.server.main**.

Step 2: Whenever we create a web service, we need to define two things **Get** method and the **URI**. Now create the **helloWorld()** method which returns the string "Hello World." If we want to tell the spring MVC that it is going to handle the REST request, we have to add **@RestController** annotation. Now it becomes a rest controller which can handle the Rest request.

The next thing we have to do is create a mapping for the method. Add **@RequestMapping** annotation just above the helloWorld() method. The HelloWorldController looks like the following:

```
package com.javatpoint.server.main;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
//Controller
@RestController
public class HelloWorldController
{
//using get method and hello-world as URI
@RequestMapping(method=RequestMethod.GET, path="/hello-world")
public String helloWorld()
{
return "Hello World";
}
}
```

We can also improve the above code by using the **@GetMapping** annotation instead of @RequestMapping. Here the method specification is not required.

```
package com.javatpoint.server.main;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
//Controller
@RestController
public class HelloWorldController
{
//using get method and hello-world as URI
@GetMapping(path="/hello-world")
public String helloWorld()
{
return "Hello World";
}
}
```

Step 3: Run the **RestfulWebServiceApplication**. It displays the string **Hello World** on the browser.

Enhancing the Hello World Service to Return a Bean

In this section, we are going to generate a bean for the method helloWorld().

Step 1: Create a **helloWorldBean()** method in **HelloWordController.java** file. Map the URI to "**/hello-world-bean**" and return **HelloWorldBean**.

HelloWorldController.java

```
package com.javatpoint.server.main;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
//Controller
@RestController
public class HelloWorldController
{
//using get method and hello-world URI
@GetMapping(path="/hello-world")
public String helloWorld()
{
return "Hello World";
}
@GetMapping(path="/hello-world-bean")
public HelloWorldBean helloWorldBean()
{
return new HelloWorldBean("Hello World"); //constructor of HelloWorldBean
}
}
```

Step 2: Create a class **HelloWorldBean**.

Step 3: Generate Getters and **Setters**.

Right-click -> Source -> Generate Getters and Setters -> check the box -> Ok

Step 4: Generate **toString()**..

Right-click -> Source -> Generate toString().. -> Ok

HelloWorldBean.java

```
package com.javatpoint.server.main;
public class HelloWorldBean
{
public String message;
```

```
//constructor of HelloWorldBean
public HelloWorldBean(String message)
{
this.message=message;
}
//generating getters and setters
public String getMessage()
{
return message;
}
public void setMessage(String message)
{
this.message = message;
}
@Override
//generate toString
public String toString()
{
return String.format ("HelloWorldBean [message=%s]", message);
}
}
```

Step 5: Launch the **HelloWorldController**. The URL of the browser changes to **localhost:8080/hello-world-bean**.

It returns the message "**Hello World**" in JSON format.

```
{
message: "Hello World"
}
```