

Assignment No 10

and RESTful Web Services

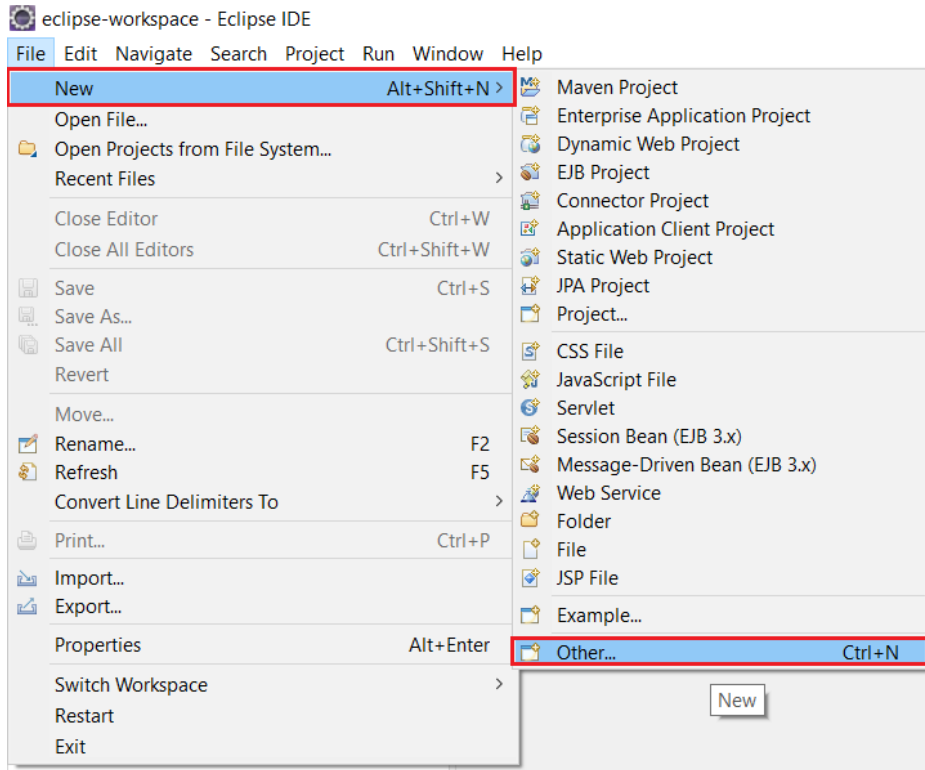
1. Write a program to create a simple Spring Boot application that prints a message.
2. Write a program to demonstrate RESTful Web Services with spring boot

Problem Statement 10.1. Write a program to create a simple Spring Boot application that prints a message.

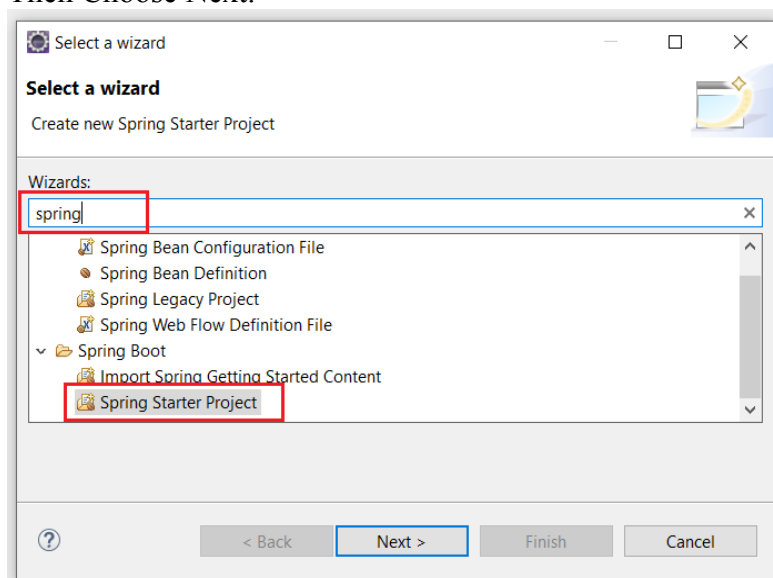
Note : Make sure you have installed the Spring Plugin in Eclipse Itself.

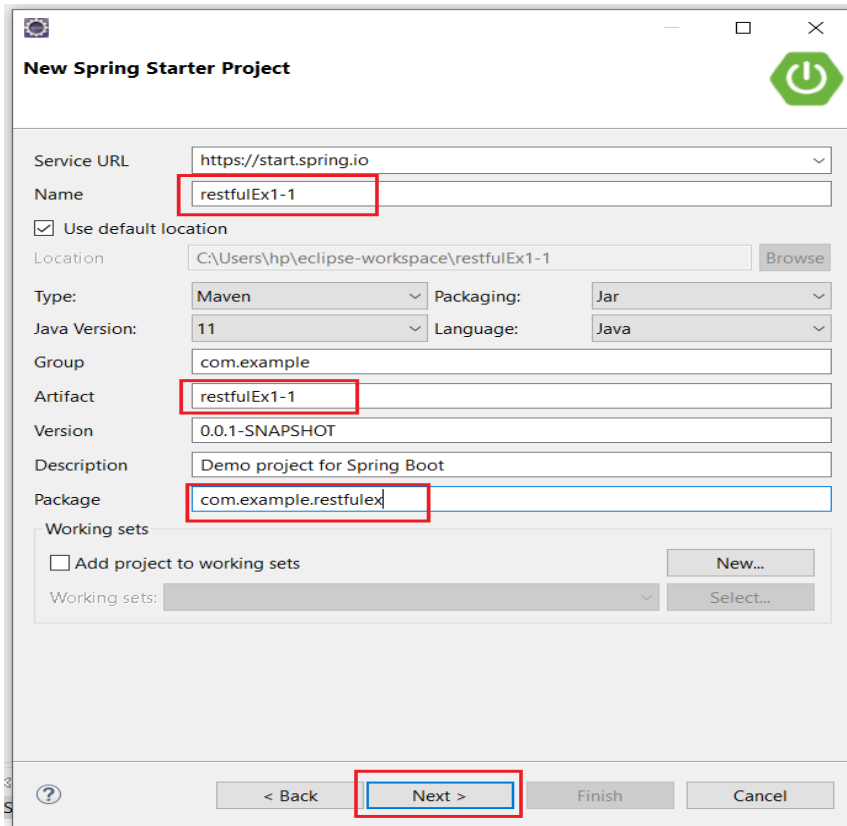
Step 1 :

1.1 : Open Eclipse. Go To File > New > Other.



1.2 : Search for 'Spring Boot' and Select 'Spring Starter Project'. Then Click on Next. On Next Wizard, Choose your Project Name, and other parameters such as Group ID, Artifact ID. Then Choose Next.





New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

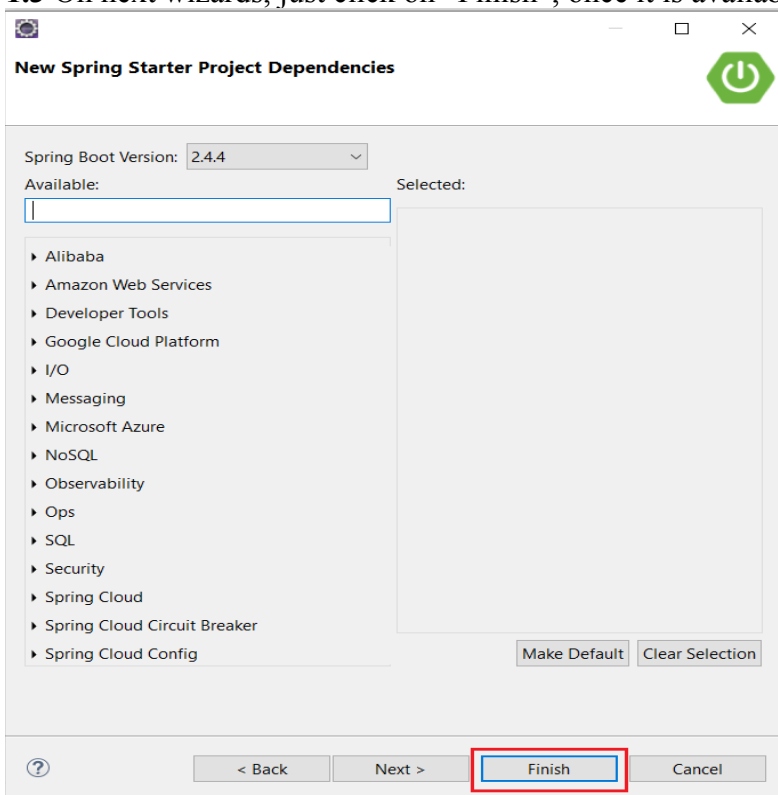
Package:

Working sets

☐ Add project to working sets

Working sets:

1.3 On next wizards, just click on “Finish”, once it is available.



New Spring Starter Project Dependencies

Spring Boot Version:

Available:

Selected:

- ▶ Alibaba
- ▶ Amazon Web Services
- ▶ Developer Tools
- ▶ Google Cloud Platform
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL
- ▶ Observability
- ▶ Ops
- ▶ SQL
- ▶ Security
- ▶ Spring Cloud
- ▶ Spring Cloud Circuit Breaker
- ▶ Spring Cloud Config

Step 2 : Go to <https://start.spring.io/>

Select All the Options specific to your Machine and Java Version.

After click on Add Dependencies -> search Spring Web -> Add Spring Web -> Click on Generate.

The screenshot shows the Spring Initializr web application interface. The top navigation bar includes the Spring logo and the text "Spring Initializr". The main content area is divided into several sections:

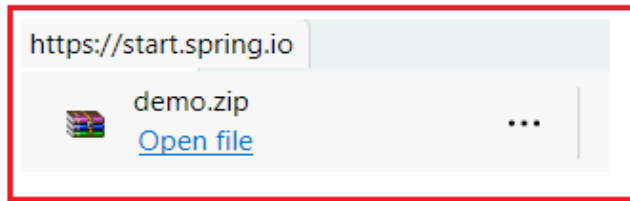
- Project:** Includes a sidebar menu with "Project" and "Language" sections. The "Project" section has radio buttons for "Maven Project" (selected) and "Gradle Project".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for "2.5.0 (SNAPSHOT)", "2.5.0 (M3)", "2.4.5 (SNAPSHOT)", "2.4.4" (selected), "2.3.10 (SNAPSHOT)", and "2.3.9".
- Project Metadata:** A form with fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo).
- Packaging:** Includes radio buttons for "Jar" (selected) and "War".
- Java:** Includes radio buttons for "16", "11", and "8" (selected).

The "Dependencies" section on the right shows "No dependency selected" and a button "ADD DEPENDENCIES... CTRL + B".

Below the main configuration area, a search bar contains the text "spring". A dropdown menu is open, showing a green box with the text "Spring Web WEB" and a description: "Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container." A button "ADD DEPENDENCIES... CTRL + B" is visible next to the search bar.

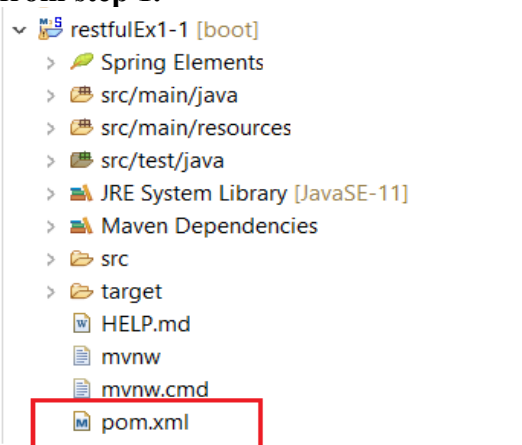
At the bottom, there are three buttons: "GENERATE CTRL + G" (highlighted with a red box), "EXPLORE CTRL + SPACE", and "SHARE...".

Click On Generate, a zip file will be downloaded.



Unzip the downloaded zip file and open the pom.xml file inside the demo folder.

Copy the Contents of the pom.xml file & paste it in the pom.xml file of our created project from step 1.



Save the file, an automatic download process will start, wait till its completed.

Now you are good to go and develop Spring Boot Applications.

Problem Statement 1 : Write a program to create a simple Spring Boot application that prints a message.

Right click on src -> new -> class -> class name-

Problem Statement 1 : Write a program to create a simple Spring Boot application that prints a message.

Solution :

HelloWorldController.java

```
package com.example.demo;
```

```
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController  
public class HelloWorldController {  
    @RequestMapping("/")  
    public String hello()  
    {  
        return " Hello World...!";  
    }  
}
```

Main file will create automatically

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Restful11Application {

 public static void main(String[] args) {

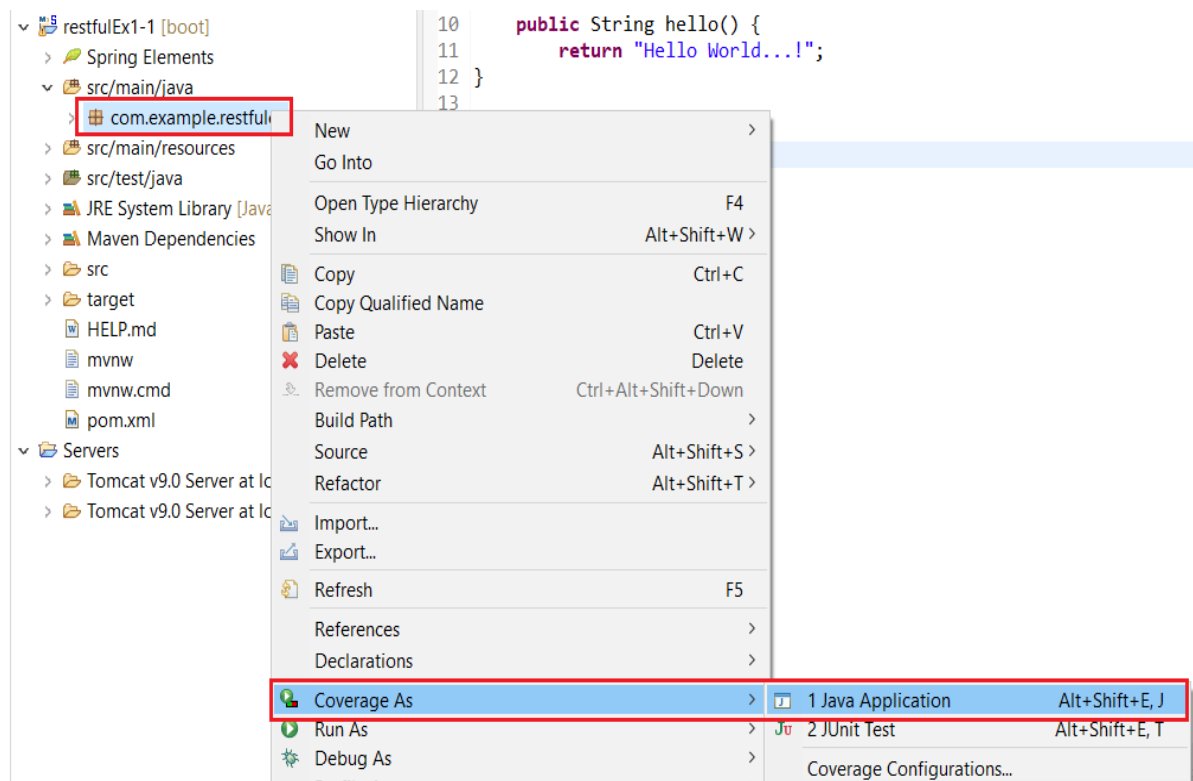
 SpringApplication.run(Restful11Application.class, args);

 }

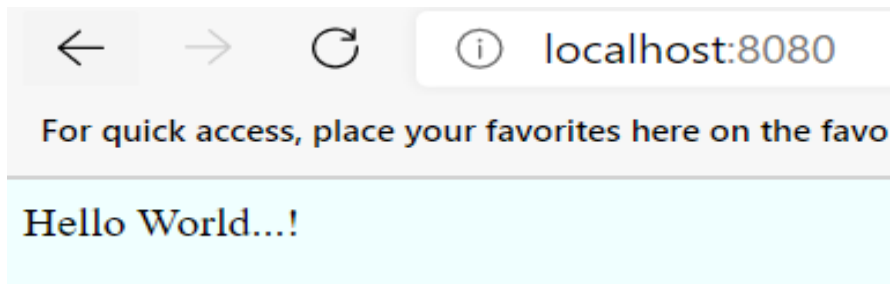
}

Output :

Go to Project-> src/main/java -> com.example.restfulex -> Right click on it then Run as -> Java Application



OUTPUT-

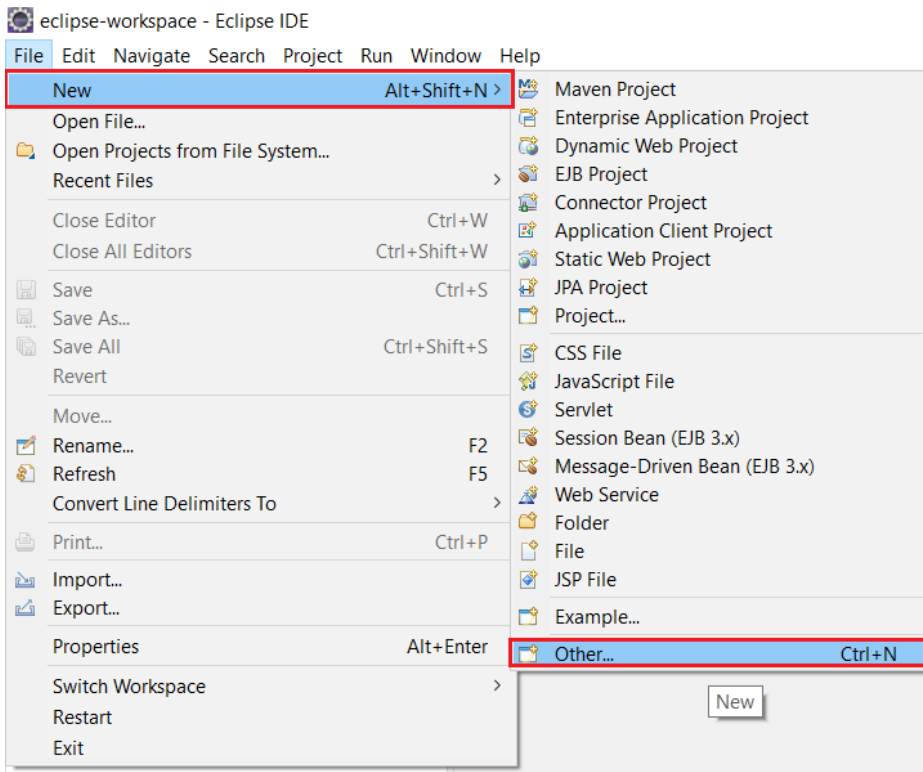


Problem Statement 10.2 : Write a program to demonstrate RESTful Web Services with spring boot

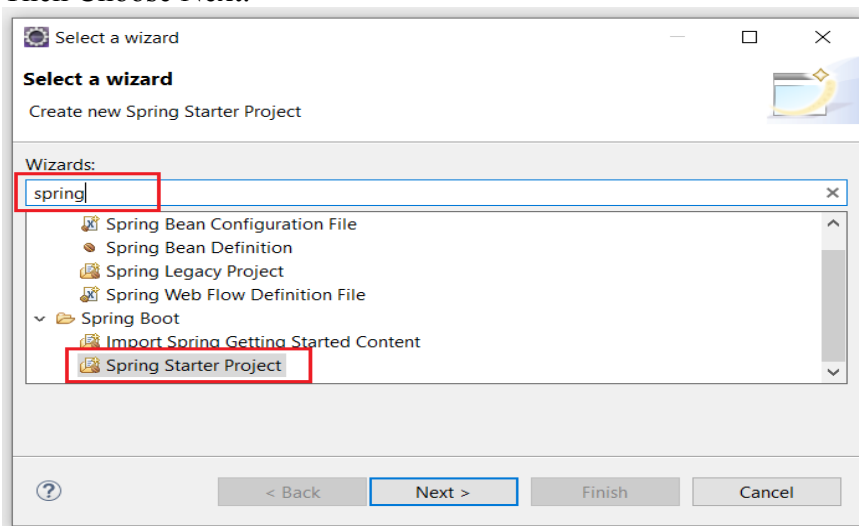
Solution :

Step 1 :

1.1 : Open Eclipse. Go To File > New > Other.



1.2 : Search for 'Spring' and Select 'Spring Starter Project'. Then Click on Next.
On Next Wizard, Choose your Project Name, and other parameters such as Group ID, Artifact ID.
Then Choose Next.



New Spring Starter Project



Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:

1.3 On next wizards, just click on “Finish”, once it is available.



New Spring Starter Project Dependencies



Spring Boot Version:

Available:

Selected:

- Alibaba
- Amazon Web Services
- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud
- Spring Cloud Circuit Breaker
- Spring Cloud Config

Step 2 : Go to <https://start.spring.io/>

Select All the Options specific to your Machine and Java Version.

After click on Add Dependencies -> search

https://start.spring.io

For quick access, place your favorites here on the favorites bar. [Manage favorites now](#)

Project

☒ Maven Project ☐ Gradle Project

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 2.5.0 (SNAPSHOT) ☐ 2.5.0 (M3) ☐ 2.4.5 (SNAPSHOT) ☒ 2.4.4 ☐ 2.3.10 (SNAPSHOT) ☐ 2.3.9

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging

☒ Jar ☐ War

Java ☐ 16 ☒ 11 ☐ 8

Dependencies

[ADD DEPENDENCIES... CTRL + B](#)

No dependency selected

Spring Data JPA and h2 Database -> Add -> Click on Generate.

Spring Data JPA **SQL**

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database **SQL**

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Dependencies

[ADD DEPENDENCIES... CTRL + B](#)

Spring Data JPA **SQL**

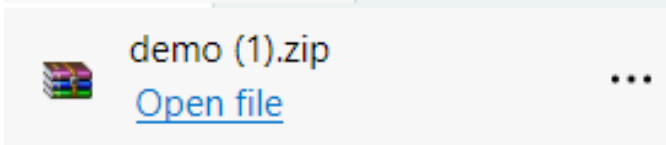
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database **SQL**

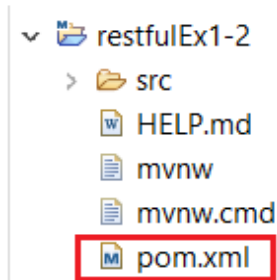
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.



Click On Generate, a zip file will be downloaded.



Unzip the downloaded zip file and open the pom.xml file inside the demo folder.
Copy the Contents of the pom.xml file & paste it in the pom.xml file of our created project from step 1.

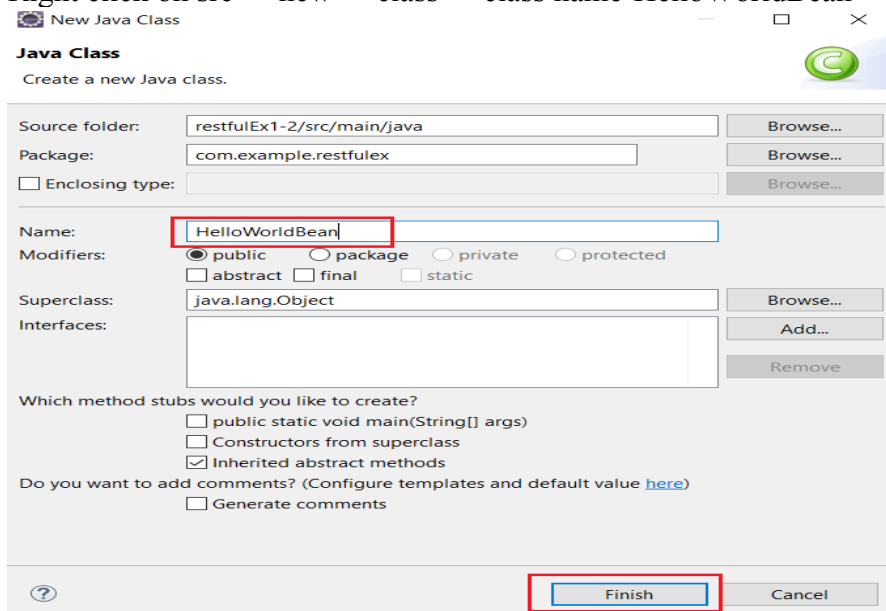


Save the file, an automatic download process will start, wait till its completed.

Now you are good to go and develop Spring Boot Applications.

Problem Statement 1 : Write a program to create a simple Spring Boot application that prints a message.

Right click on src -> new -> class -> class name-HelloWorldBean



Solution :

HelloWorldBean.java

```
package com.example.restfulex;
```

```
public class HelloWorldBean {
    public String message;

    public HelloWorldBean(String message)
    {
        this.message=message;
    }
    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);
    }
}
```

Right click on src -> new -> class -> class name-HelloWorldController HelloWorldController.java

```
package com.example.restfulex;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
```

```
public class HelloWorldController {
    @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
    }
}
```

RestfulEx12Application.java

```
package com.example.restfulex;
```

```
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
```

```
public class RestfulEx12Application {
```

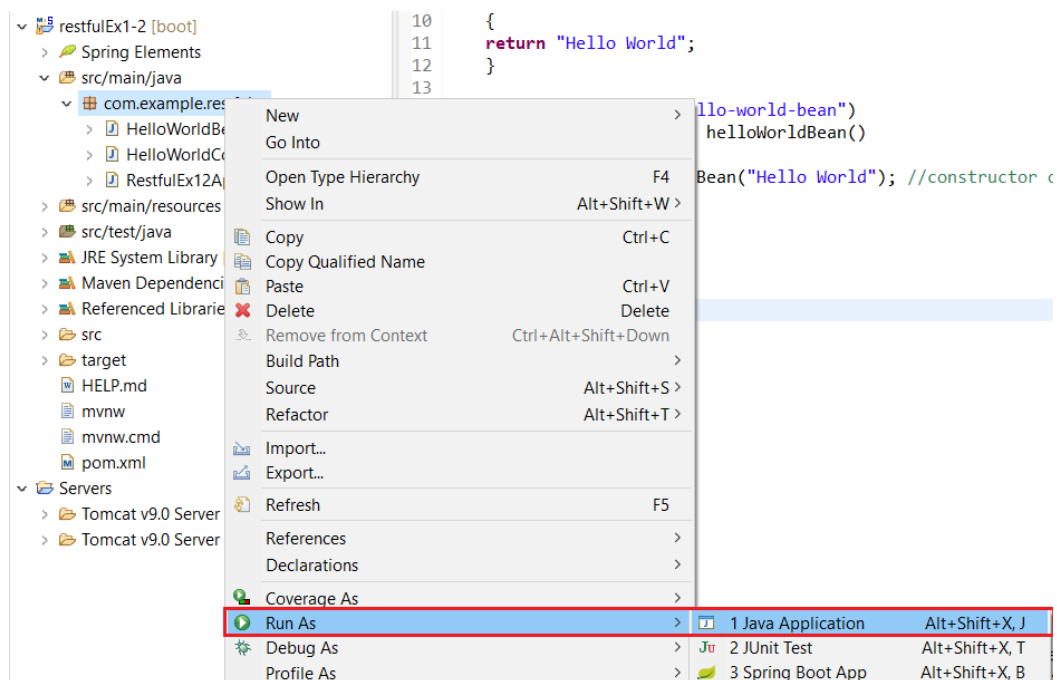
```
    public static void main(String[] args) {
```

```
        SpringApplication.run(RestfulEx12Application.class, args);
```

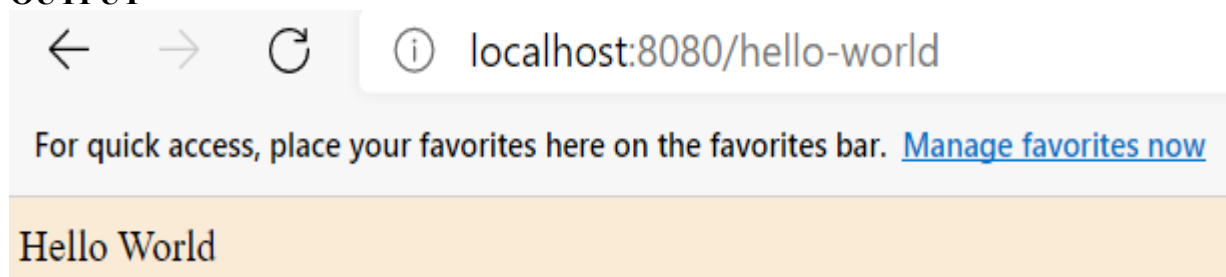
```
    }
```

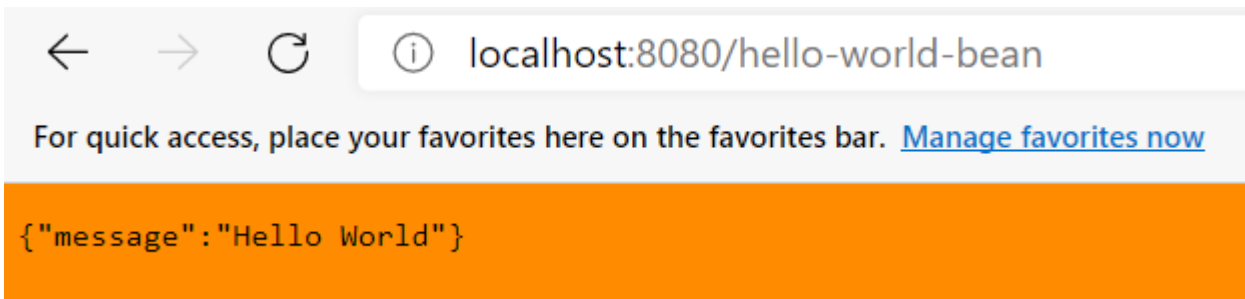
```
}
```

Go to Project-> src/main/java -> com.example.restfulex -> Right click on it then Run as -> Java Application



OUTPUT-

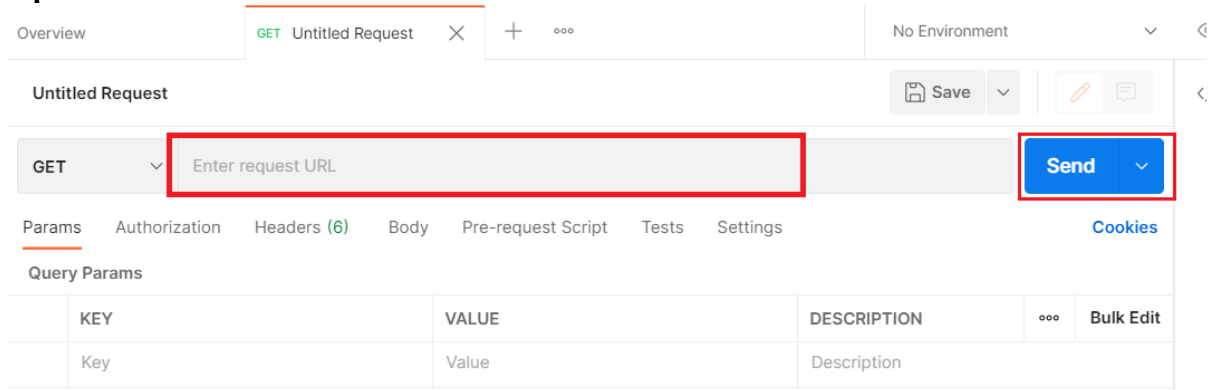




Testing API with PostMan.

EndPoint : <http://localhost:8080/hello-world-bean>

Open Poastman -> Enter url



After pest url-> click on send button.

