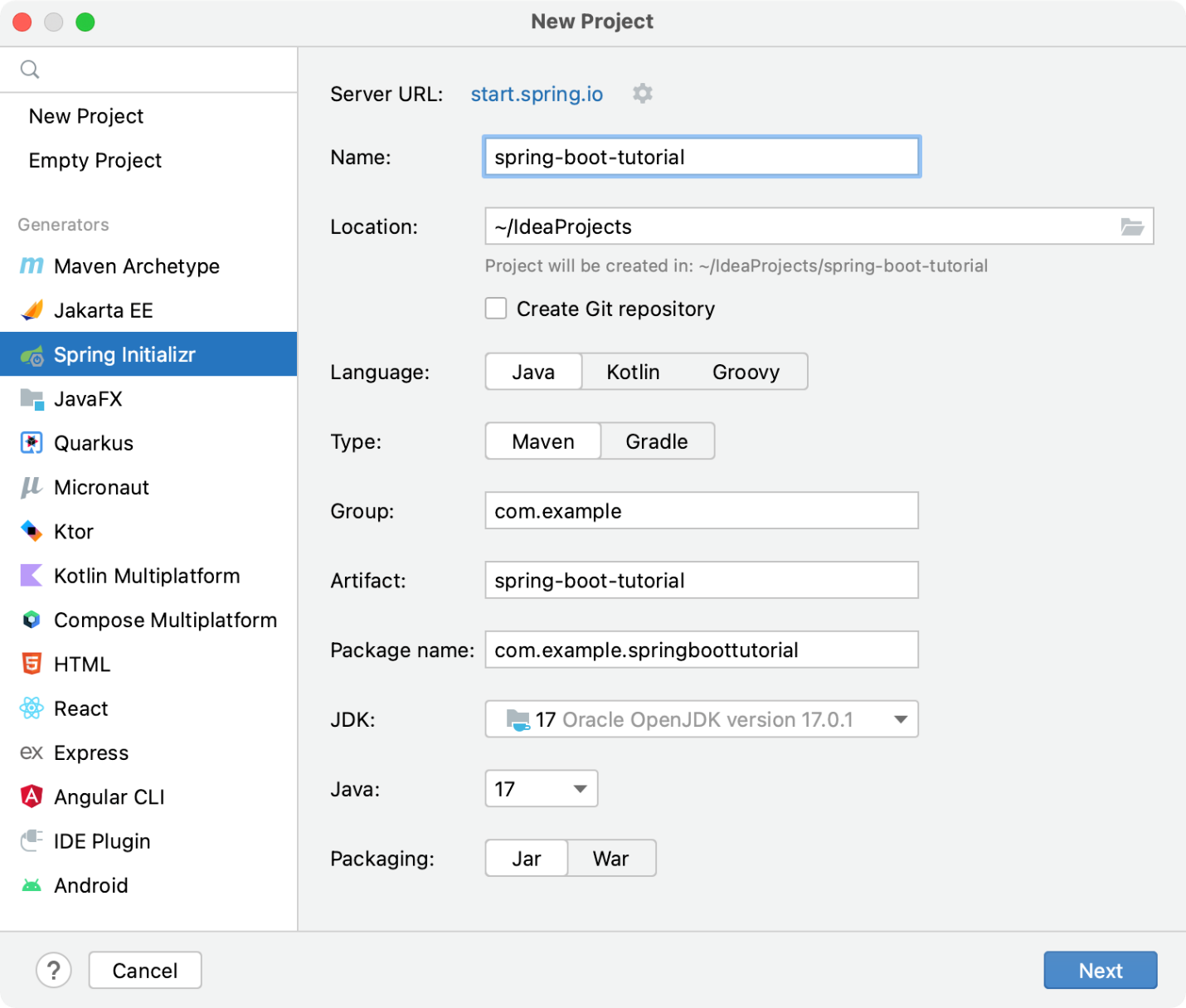
**Create a new Spring Boot project﻿**

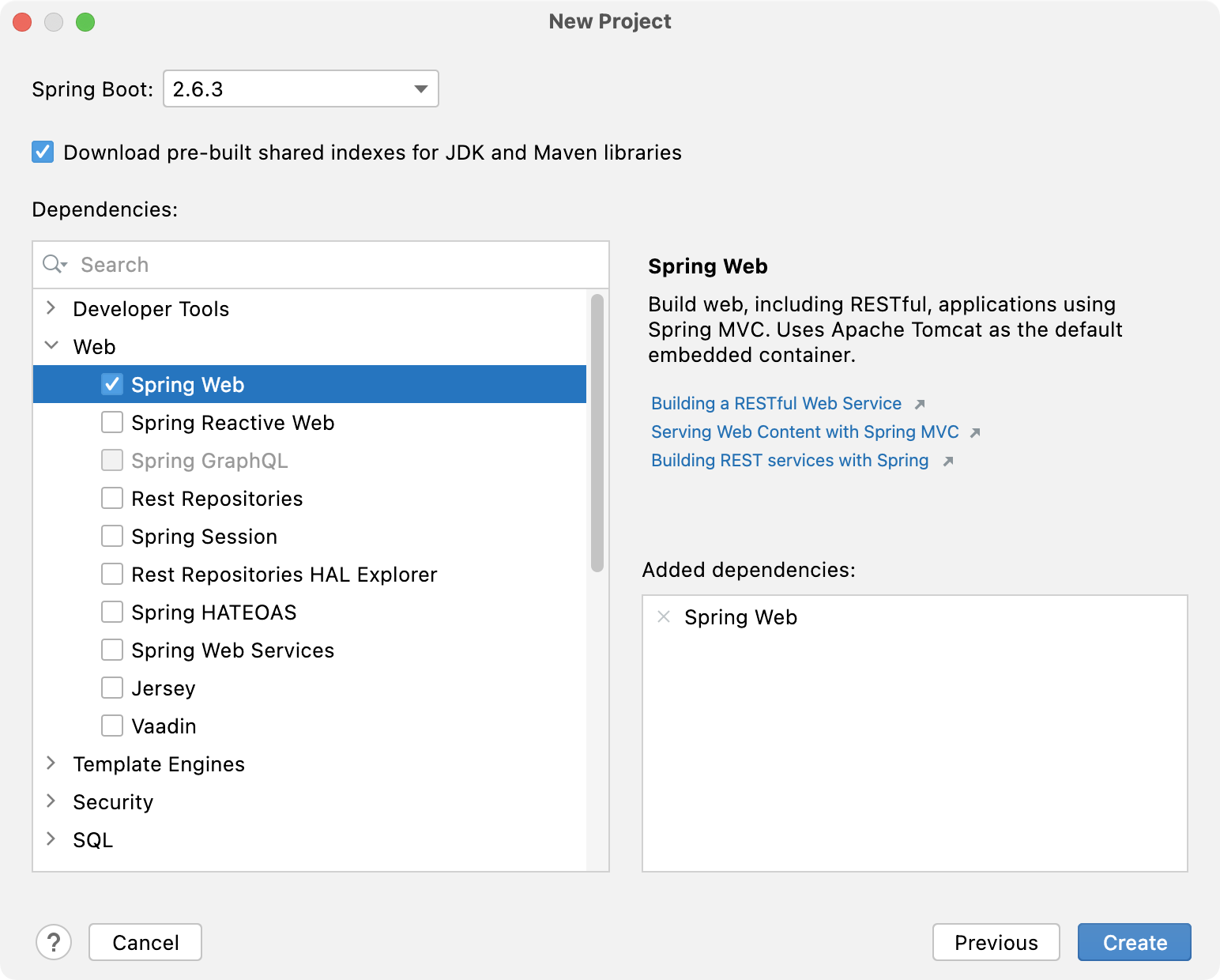
1. From the main menu, select File | New | Project.
2. In the left pane of the New Project wizard, select Spring Initializr.
3. Specify a name for the project: spring-boot-tutorial.

From the JDK list, select Download JDK and download the latest version of Oracle OpenJDK (as of writing this tutorial, it was version 16.0.2).

Select the latest Java version.



1. Click Next to continue.
2. Select the Spring Web dependency under Web. This dependency is required for any web application that uses [Spring MVC](https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html).



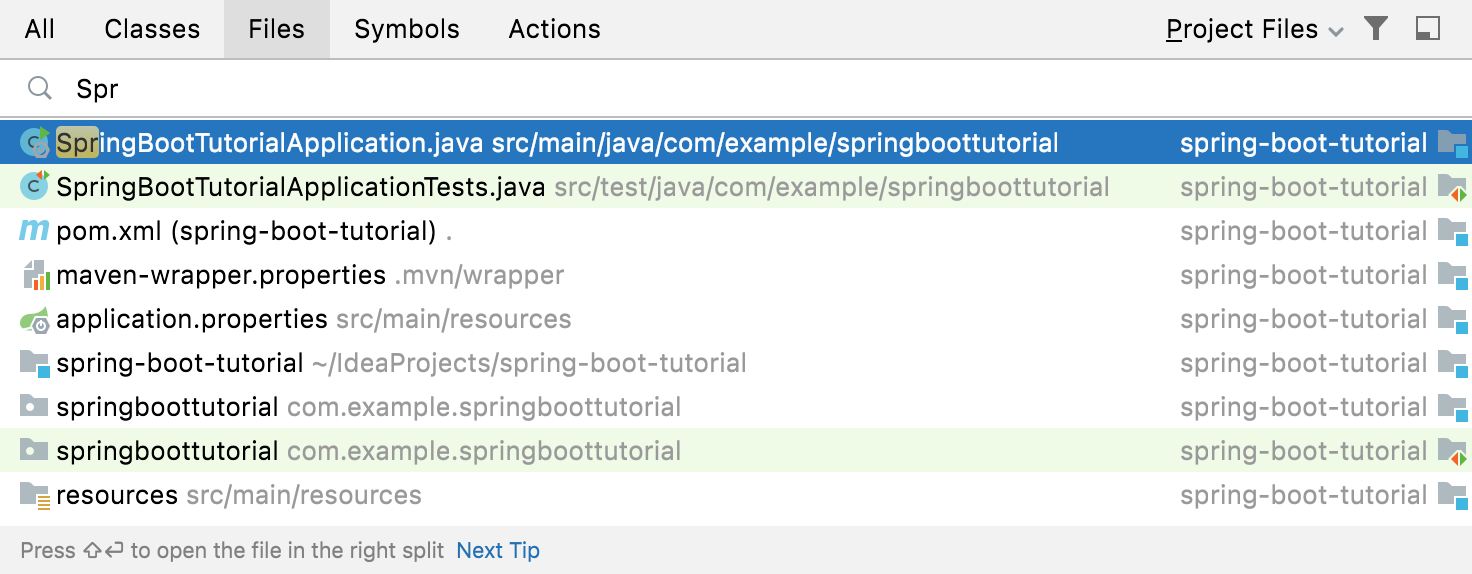
Click Create to generate and set up the project.

**Add a method that sends a greeting﻿**

Spring Initializr creates a class with the main() method to bootstrap your Spring application. In this tutorial, we'll add the sayHello() method directly to this class.

1. Open the **SpringBootTutorialApplication.java** file under **src/main/java/com/example/springboottutorial**.

IntelliJ IDEA provides the Go to File action to quickly find and open files. From the main menu, select Navigate | File or press Ctrl+Shift+N, start typing the name of the file and select it from the list.



1. Add the sayHello() method with all of the necessary annotations and imports so that the file looks like this:
2. package com.example.springboottutorial;
3. import org.springframework.boot.SpringApplication;
4. import org.springframework.boot.autoconfigure.SpringBootApplication;
5. import org.springframework.web.bind.annotation.GetMapping;
6. import org.springframework.web.bind.annotation.RequestParam;
7. import org.springframework.web.bind.annotation.RestController;
8. @SpringBootApplication
9. @RestController
10. public class SpringBootTutorialApplication {
11. public static void main(String[] args) {
12. SpringApplication.run(SpringBootTutorialApplication.class, args);
13. }
14. @GetMapping("/hello")
15. public String sayHello(@RequestParam(value = "myName", defaultValue = "World") String name) {
16. return String.format("Hello %s!", name);
17. }

}

The sayHello() method takes the name parameter and returns the word Hello combined with the parameter value. Everything else is handled by adding Spring annotations:

* + The @RestController annotation marks the SpringBootTutorialApplication class as a request handler (a REST controller).
  + The @GetMapping("/hello") annotation maps the sayHello() method to GET requests for /hello.
  + The @RequestParam annotation maps the name method parameter to the myName web request parameter. If you don't provide the myName parameter in your web request, it will default to World.

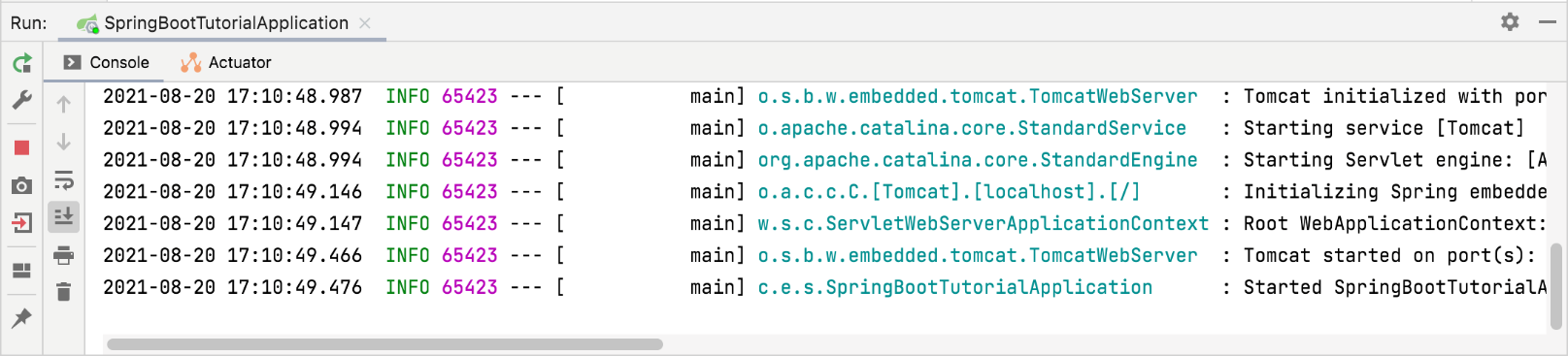
**Run your Spring application﻿**

IntelliJ IDEA creates a Spring Boot run configuration that you can use to run your new Spring application.

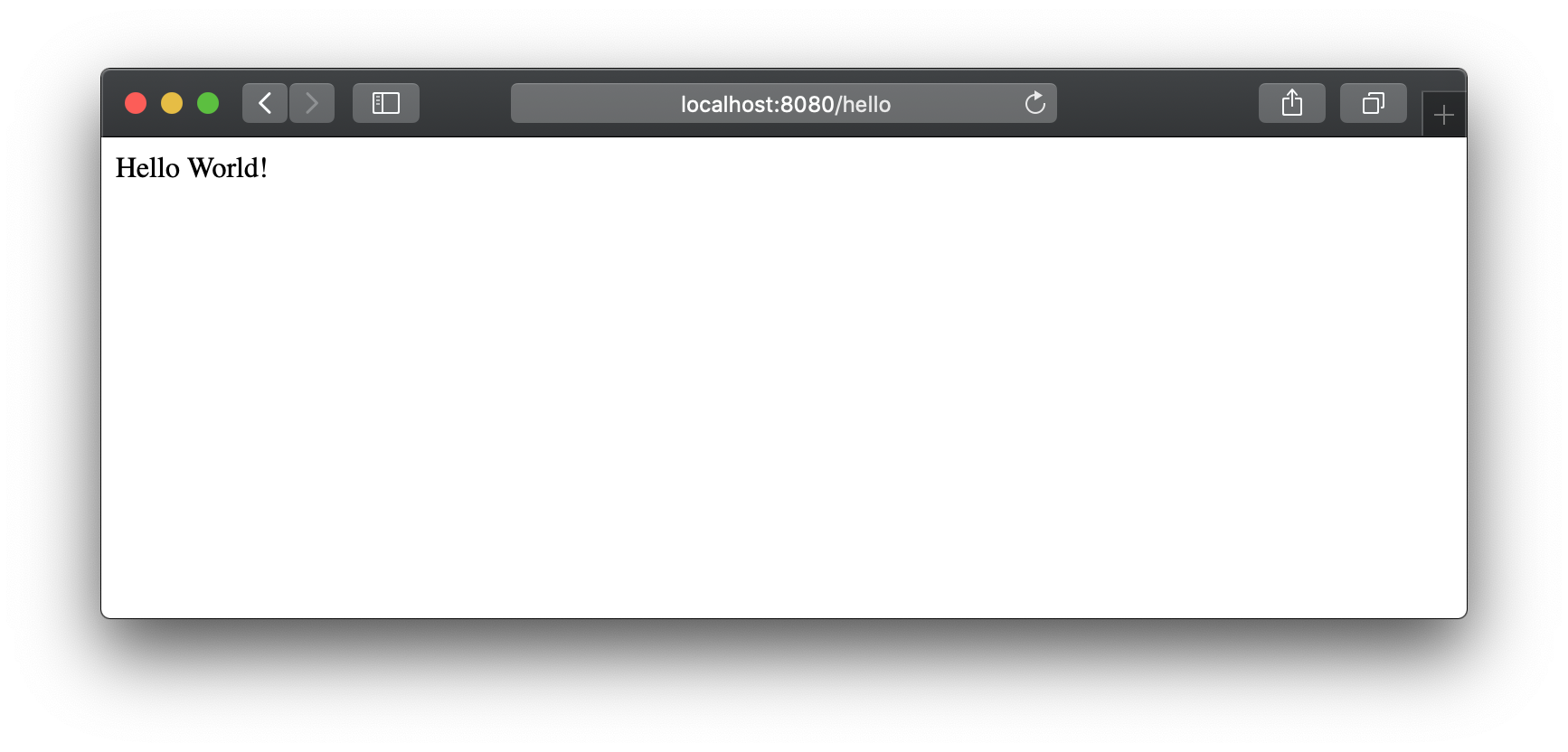
* If the run configuration is selected, press Shift+F10.

You can also use the  icon in the gutter of the **SpringBootTutorialApplication.java** file next to the class declaration or the main() method declaration.

By default, IntelliJ IDEA shows your running Spring Boot application in the [Run](https://www.jetbrains.com/help/idea/run-tool-window.html) tool window.



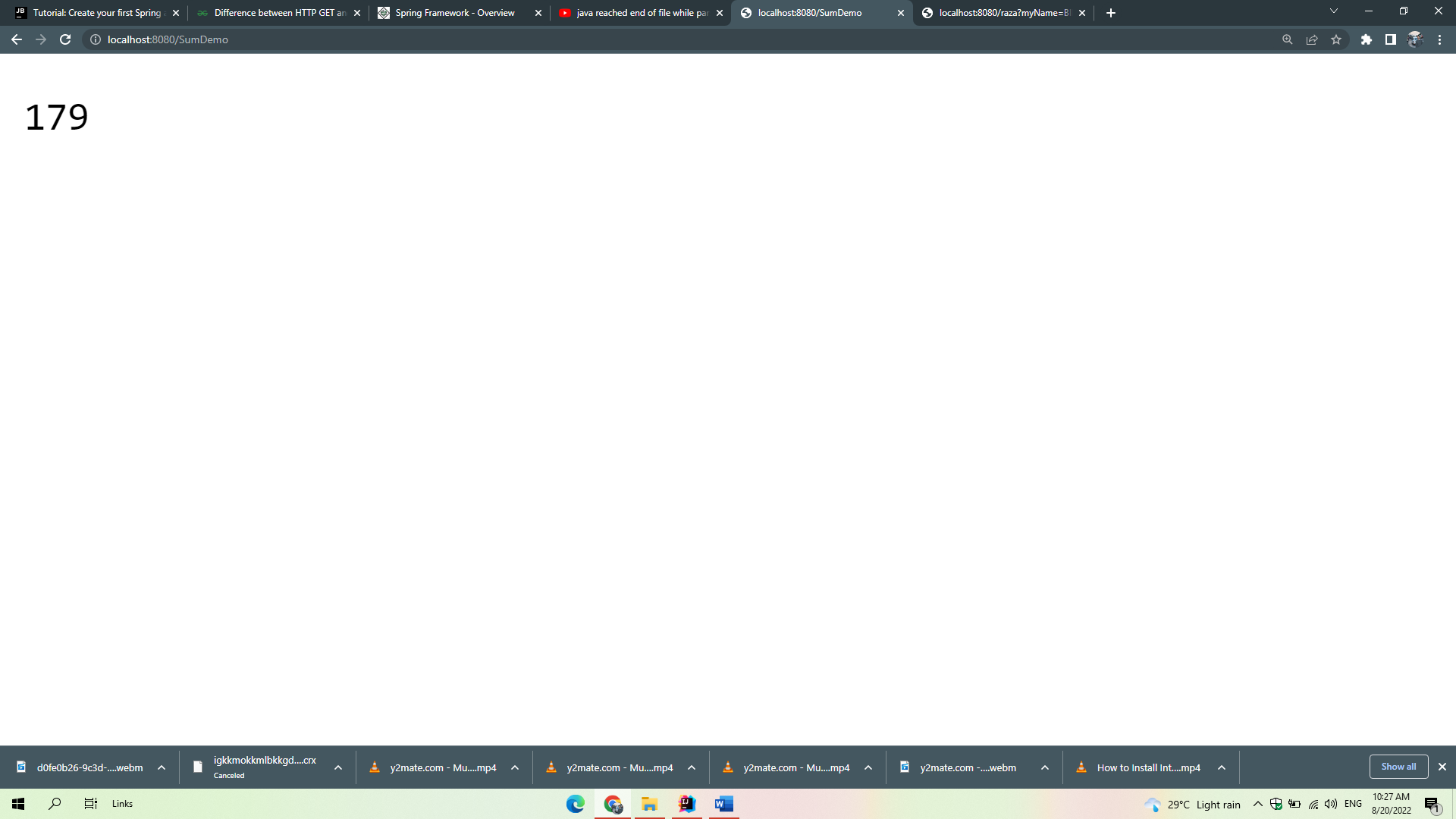
The Console tab shows the output of Spring log messages. By default, the built-in Apache Tomcat server is listening on port 8080. Open your web browser and go to <http://localhost:8080/hello>. If you did everything right, you should see your application respond with Hello World!.



This is the default generic response. You can provide a parameter in your web request to let the application know how to greet you properly. For example, try <http://localhost:8080/hello?myName=Human>.

Add 2nd Method to the File

|  |
| --- |
| @GetMapping("/SumDemo")  public int sumApp() {  int no1, no2;  no1 = 89;  no2 =90;  int sum= no1 + no2;  return sum;  } |

Recompile and Run the Application 

**Add a home page﻿**

The created Spring Boot application has one endpoint available at [/hello](http://localhost:8080/hello). However, if you open the root context of your application at <http://localhost:8080/>, you will get an error because there is no root resource defined. Let's add a static HTML home page with links to your endpoint.

1. Create the **index.html** file under **/src/main/resources/static/**.

In the Project tool window, right-click the **/src/main/resources/static/** directory, select New | HTML File, specify the name **index.html**, and press Enter.

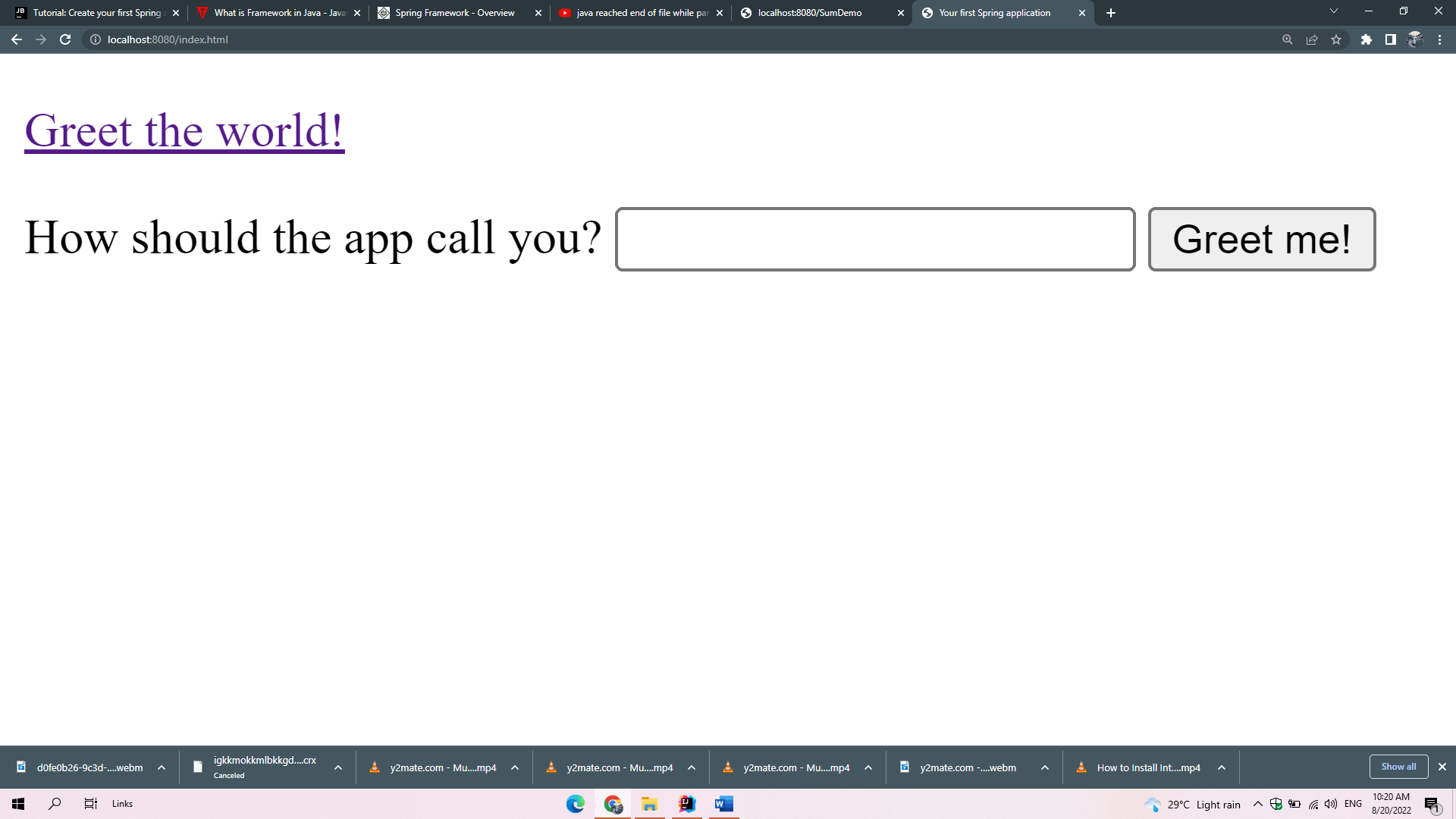
1. Modify the default template or replace it with the following HTML code:
2. <!DOCTYPE HTML>
3. <html>
4. <head>
5. <title>Your first Spring application</title>
6. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
7. </head>
8. <body>
9. <p><a href="/hello">Greet the world!</a></p>
10. <form action="/hello" method="GET" id="nameForm">
11. <div>
12. <label for="nameField">How should the app call you?</label>
13. <input name="myName" id="nameField">
14. <button>Greet me!</button>
15. </div>
16. </form>
17. </body>

</html>

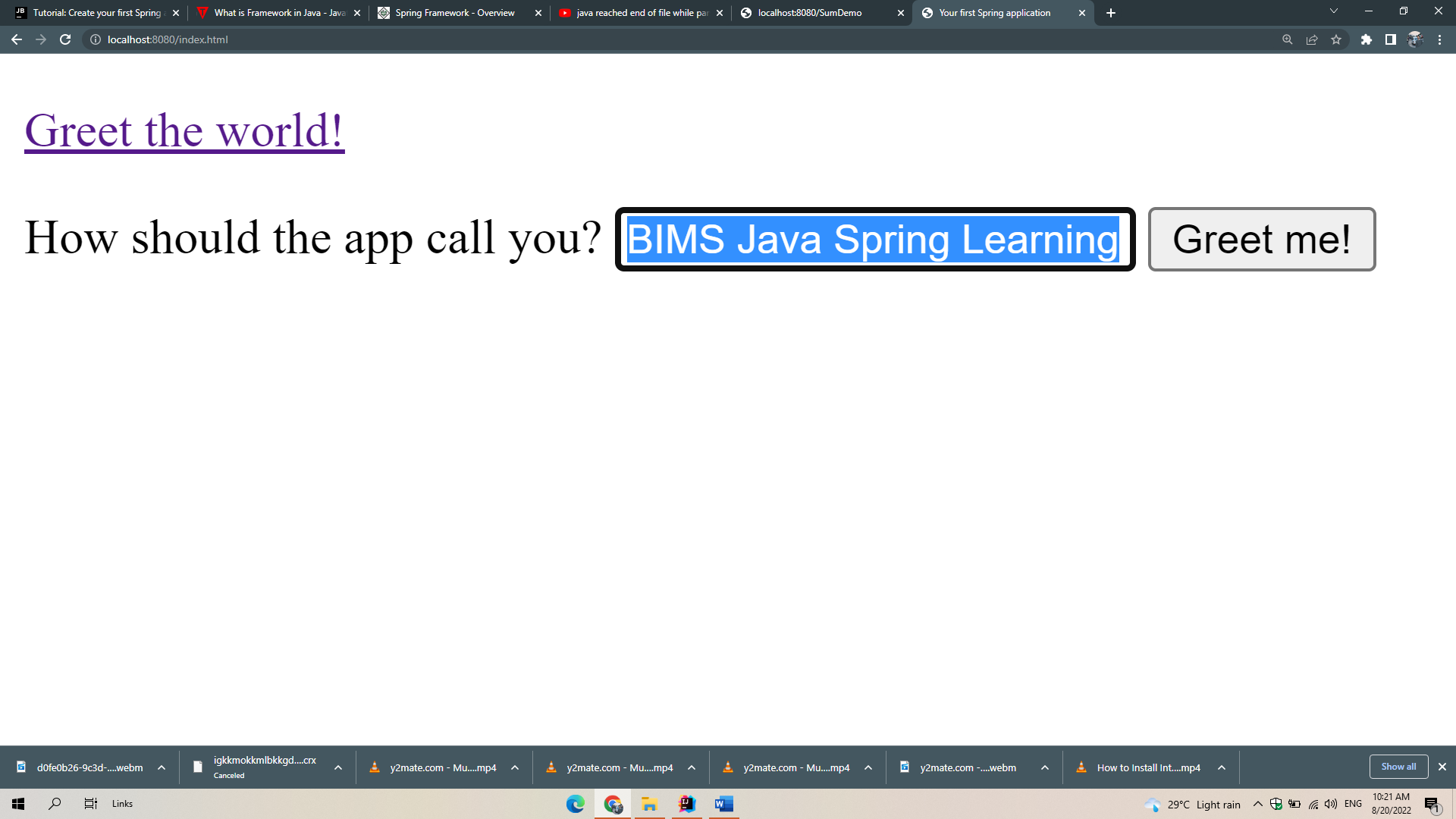
1. In the Run tool window, or press Shift+F10 to restart your Spring application.

Now your application will serve **index.html** as the root resource at <http://localhost:8080/> (<http://localhost:8080/>index.html)

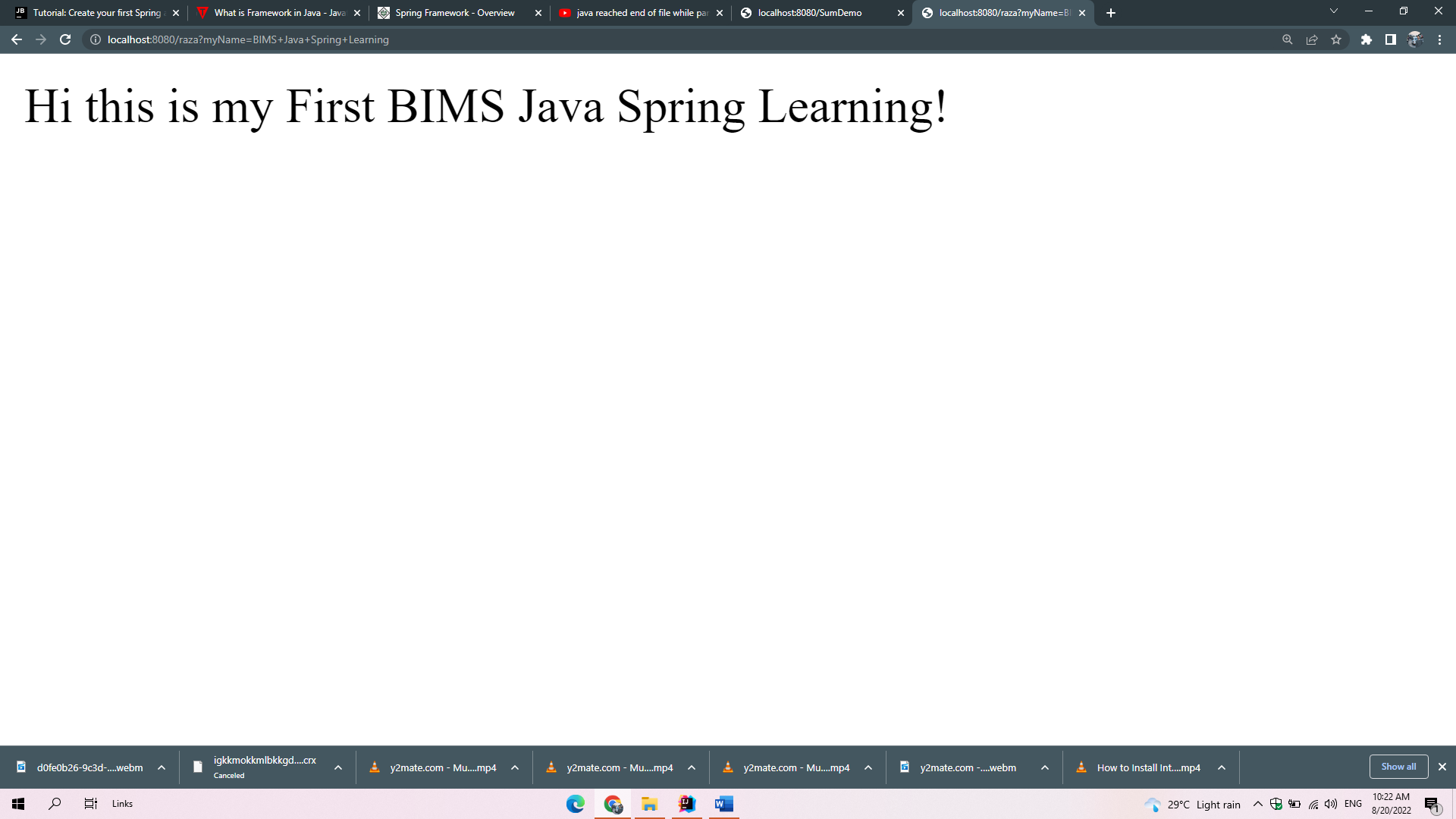
When you Run the Application, you will see the following page



Enter the Message here



Press the Greet me button, you will see the following output



Note, in the

<form action="/hello" method="GET" id="nameForm">

**HTTP GET:** The Hypertext Transfer Protocol(HTTP) Get method is mainly used at the client (Browser) side to send a request to a specified server to get certain data or resources. Using this method the server should only let us receive the data and not change its state. Hence it is only used to view something and not to change it. Get method is one of the most used HTTP method. The request parameter of the get method is appended to the URL. Get request is better for the data which does not need to be secure (It means the data which does not contain images or word documents).

**HTTP POST:** The Hypertext Transfer Protocol(HTTP) Post method is mainly used at the client (Browser) side to send data to a Specified server in order to create or rewrite a particular resource/data. This data sent to the server is stored in the request body of the HTTP request. Post method eventually leads to the creation of a new resource or updating an existing one. Due to this dynamic use, it is one of the most used HTTP methods. It is not one of the most secure methods because the data that is been sent is included in the body of the request and not in the URL. Post request is better for the data which needs to be secure (It means the data which contains images or word documents).

Here is the difference of GET and POST Methods

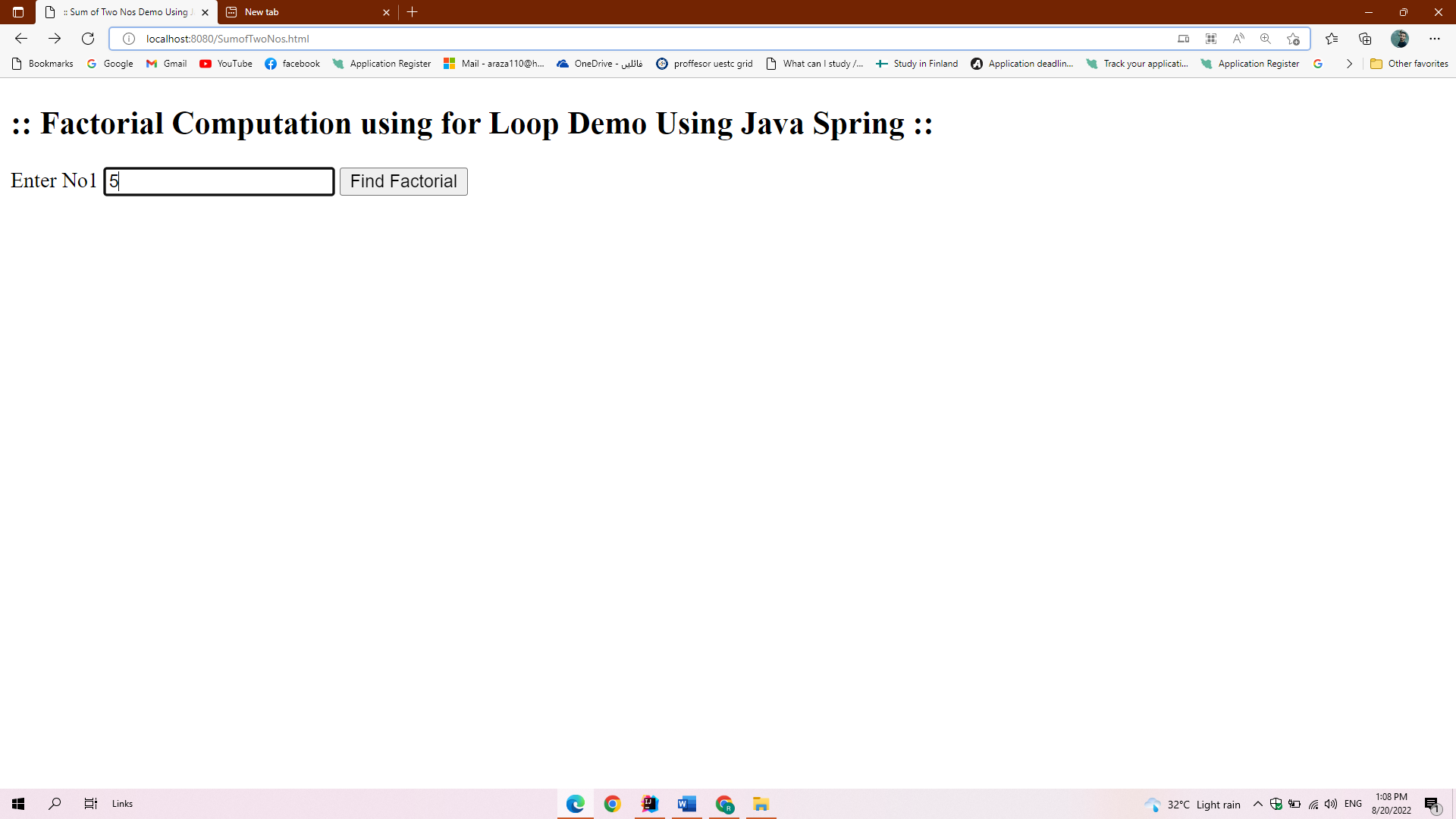
|  |  |
| --- | --- |
| **HTTP GET** | **HTTP POST** |
| In GET method we can not send large amount of data rather limited data is sent because the request parameter is appended into the URL. | In POST method large amount of data can be sent because the request parameter is appended into the body. |
| GET request is comparatively better than Post so it is used more than the  Post request. | POST request is comparatively less better than Get so it is used less than the Get request. |
| GET request is comparatively less secure because the data is exposed in the URL bar. | POST request is comparatively more secure because the data is not exposed in the URL bar. |
| Request made through GET method are stored in Browser history. | Request made through POST method is not stored in Browser history. |
| GET method request can be saved as bookmark in browser. | POST method request can not be saved as bookmark in browser. |
| Request made through GET method are stored in cache memory of Browser. | Request made through POST method are not stored in cache memory of Browser. |
| Data passed through GET method can be easily stolen by attackers. | Data passed through POST method can not be easily stolen by attackers. |
| In GET method only ASCII characters are allowed. | In POST method all types of data is allowed. |

Passing Values and Parsing into Integer from HTML form

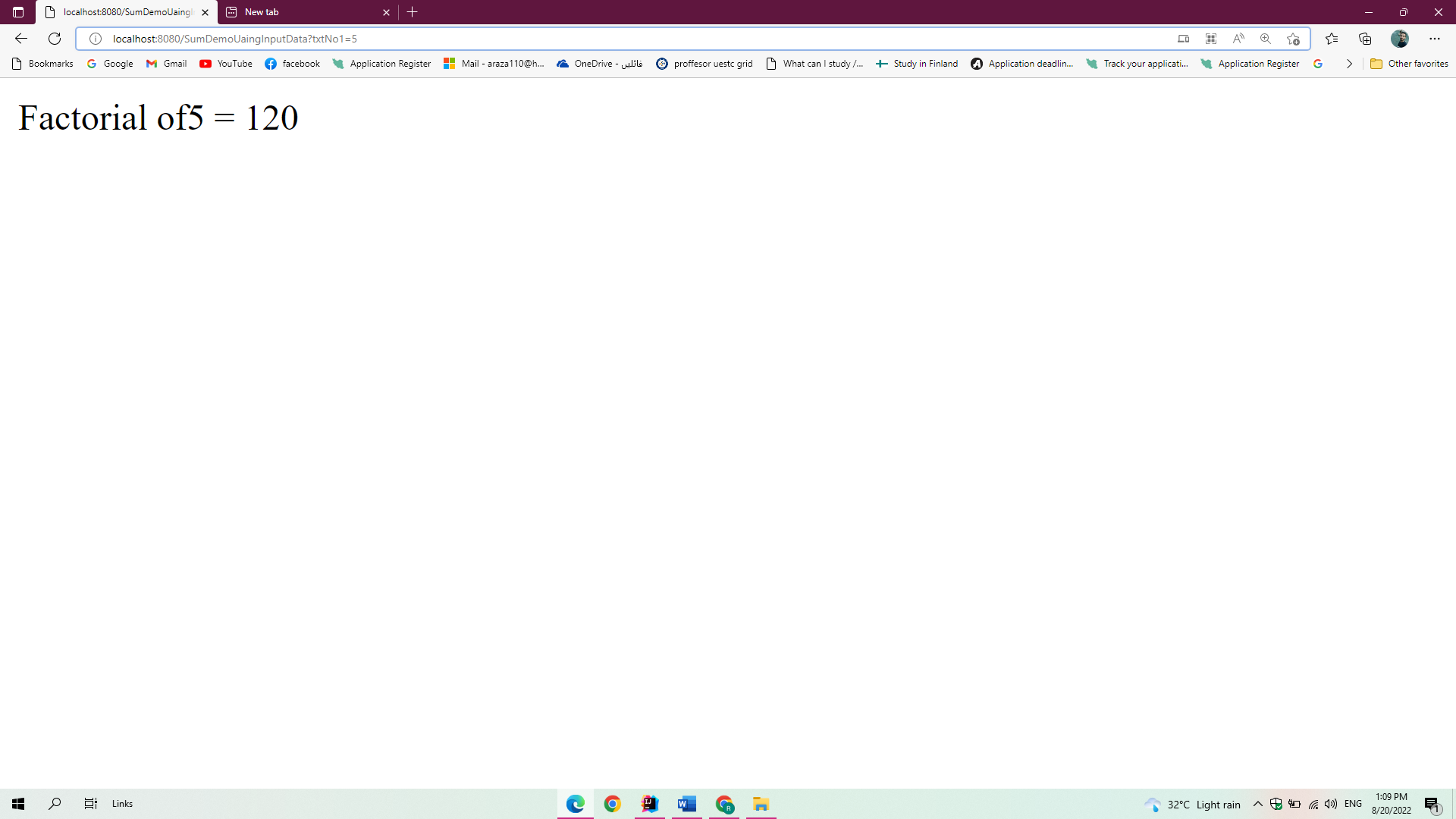
**Create an HTML Form as shown**

|  |
| --- |
| <!DOCTYPE **HTML**> <**html**> <**head**>  <**title**>:: Sum of Two Nos Demo Using Java Spring :: </**title**>  <**meta http-equiv="Content-Type" content="text/html; charset=UTF-8"** /> </**head**> <**body**>  <**h2**>:: Factorial Computation using for Loop Demo Using Java Spring :: </**h2**>  <**form action="/SumDemoUaingInputData" method="GET" id="nameForm"**>   <**div**>  <**label for="lblNo1"**>Enter No1 </**label**>  <**input name="txtNo1" id="nameField1"**>   <**button**>Find Factorial</**button**>  </**div**> </**form**> </**body**> </**html**> |

**Run the form, you will see the following output**

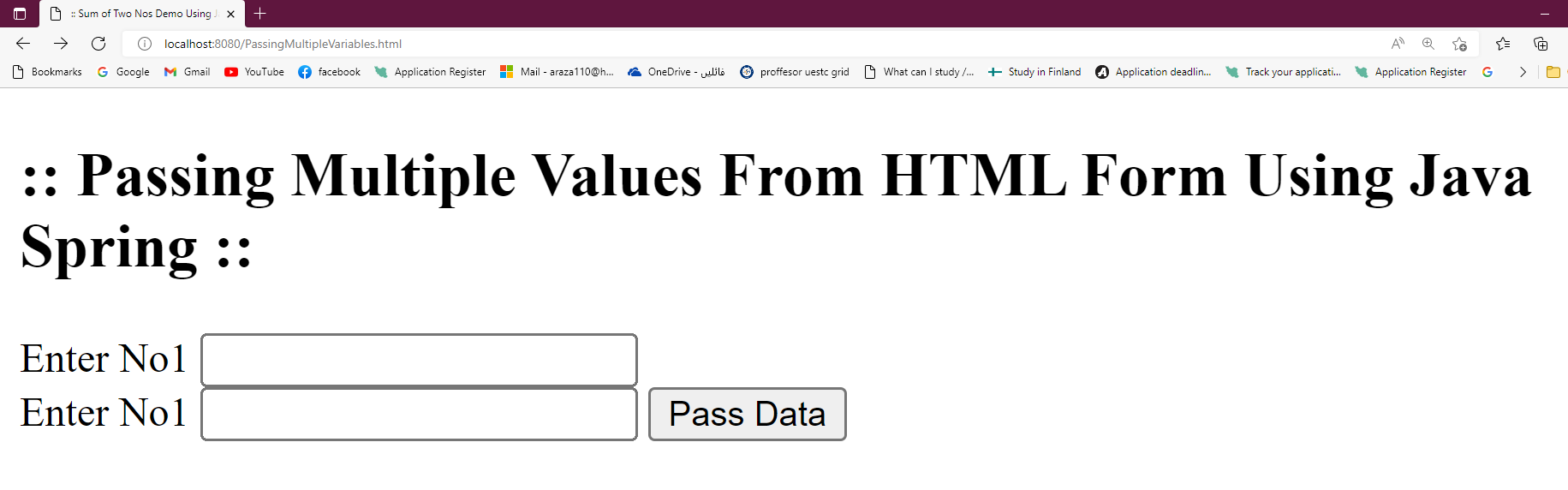


**Enter 5 and press the Find Factorial Button**



**Passing Multiple Values from HTML to Java Spring Code**

Design the following HTML form



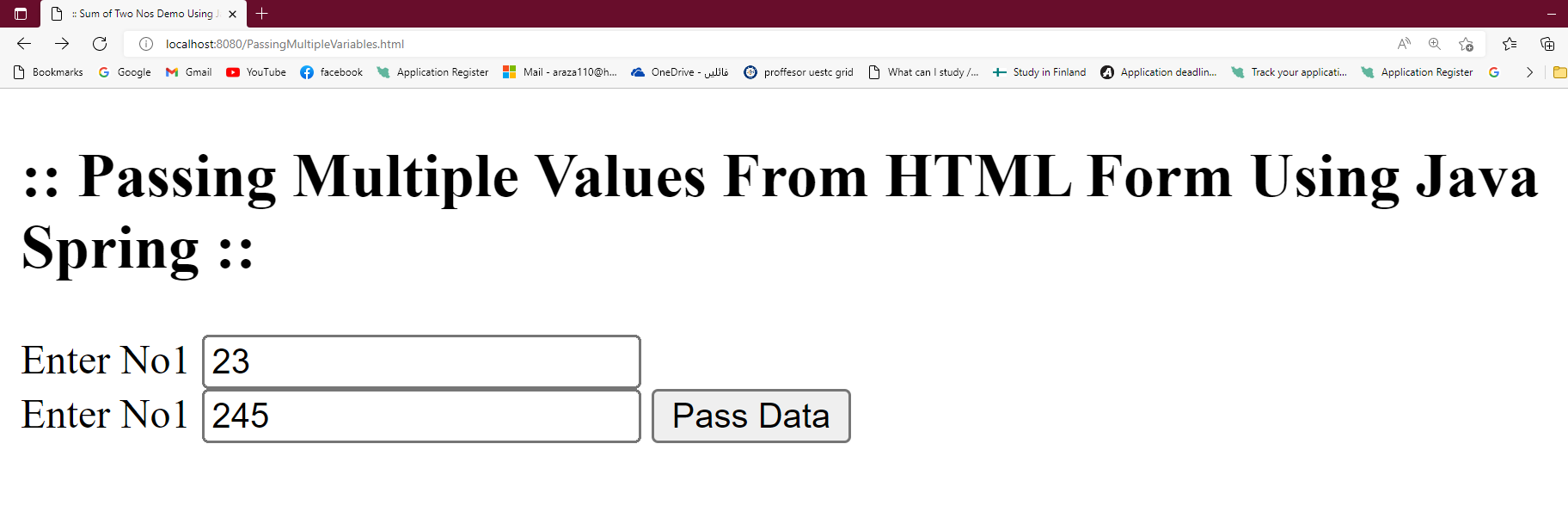
**HTML Code is**

|  |
| --- |
| <!DOCTYPE **HTML**> <**html**> <**head**>  <**title**>:: Sum of Two Nos Demo Using Java Spring :: </**title**>  <**meta http-equiv="Content-Type" content="text/html; charset=UTF-8"** /> </**head**> <**body**>  <**h2**>:: Passing Multiple Values From HTML Form Using Java Spring :: </**h2**>  <**form action="/SumTwoNosDemo" method="GET" id="nameForm"**>   <**div**>  <**label for="lblNo1"**>Enter No1 </**label**>  <**input name="txtNo1" id="nameField1"**>  <**br**/>  <**label for="lblNo2"**>Enter No1 </**label**>  <**input name="txtNo2" id="nameField2"**>  <**button**>Pass Data</**button**>  </**div**> </**form**> </**body**> </**html**> |

**Add the following Mapping Method to Spring File as shown**

|  |
| --- |
| @GetMapping(**"/SumTwoNosDemo"**) **public** String MultipleInputsDemo(@RequestParam(**"txtNo1"**) **int** v1, @RequestParam(**"txtNo2"**) **int** v2) {  **int** no1, no2;   no1 = v1;   no2 = v2;   **int** sum= no1 + no2;   **return "Sum of "** + v1 + **" And "** + v2 + **" = "** + sum;  } |

**Run the Application**



**Press the Pass Data Button, you will see the following output**

