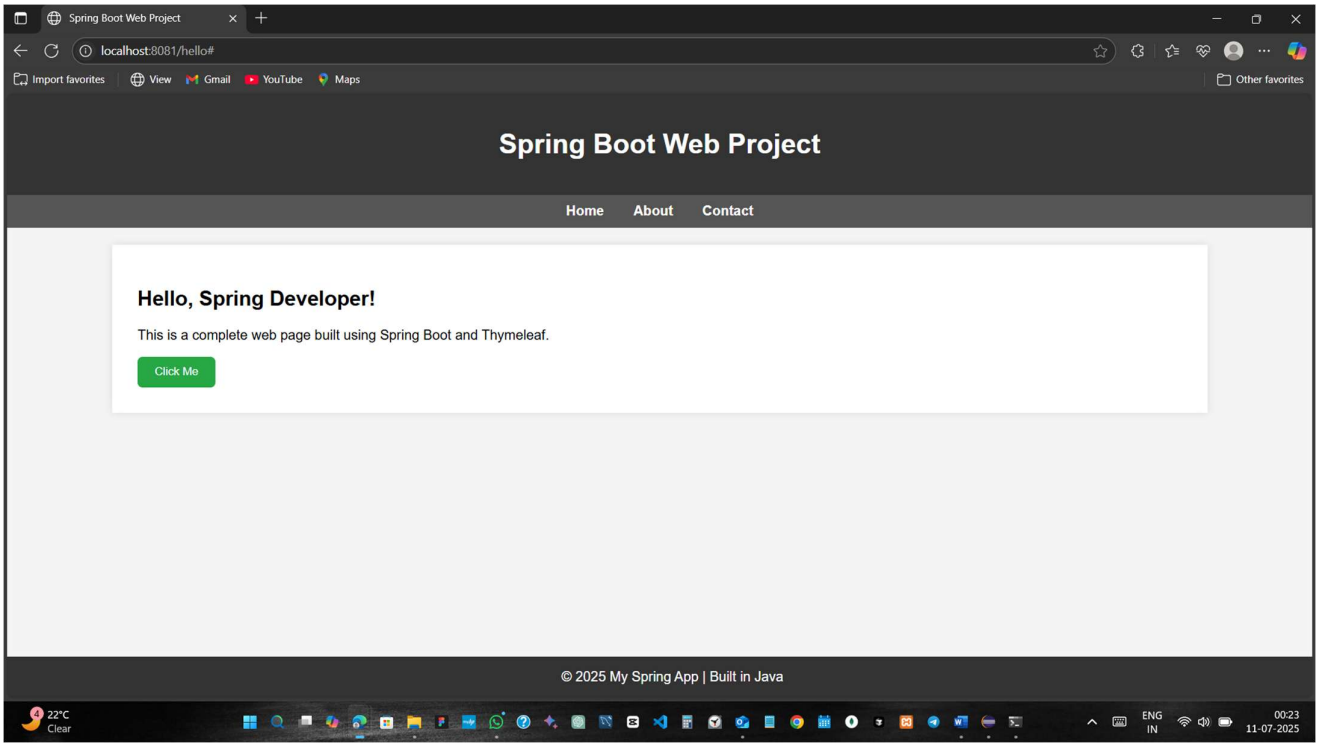
Hands on 1
Create a Spring Web Project using Maven

Result:



Note: Solution code is written below the Question [From Page No. 3]

Follow steps below to create a project:

1. Go to https://start.spring.io/
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456’ command in command line
8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
9. Include logs to verify if main() method of SpringLearnApplication.
10. Run the SpringLearnApplication class.

SME to walk through the following aspects related to the project created:

1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. SpringLearnApplication.java - Walkthrough the main() method.
5. Purpose of @SpringBootApplication annotation
6. pom.xml
    1. Walkthrough all the configuration defined in XML file
    2. Open 'Dependency Hierarchy' and show the dependency tree.

Step 1: Create Spring Boot Project

- Go to: https://start.spring.io
- Select:
    - Project: **Maven**
    - Language: **Java**
    - Group: com.cognizant
    - Artifact: spring-learn
    - Dependencies:
        - **Spring Web**
        - **Spring Boot DevTools**
- Click **Generate**, and download the ZIP.

Step 2: Import into Eclipse

- Extract ZIP into Eclipse workspace.
- Open Eclipse → File → Import → Maven → Existing Maven Projects → Browse to extracted folder → Finish.

Step 3: Understand Project Structure

Folder/File	Description
src/main/java	Java application source code
src/main/resources	Configuration files and HTML templates
src/test/java	Unit testing code
pom.xml	Maven config (dependencies, plugins)
SpringLearnApplication.java	Main class with main() method

Step 4: Run the Application

- Right-click SpringLearnApplication.java → Run As → Java Application
- Console will show logs.

Step 5: Add REST API Endpoint

HelloController.java

```
package com.cognizant.spring_learn;

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

@RestController
public class HelloController {

  @GetMapping("/hello")
  public String sayHello() {
    return "Hello from Spring Boot!";
  }
}
```

Open: http://localhost:8081/hello

Step 6: Understanding pom.xml

- Manages dependencies like:
  - spring-boot-starter-web
  - spring-boot-devtools
  - spring-boot-starter-thymeleaf
- Also includes project metadata (groupId, artifactId, etc.)

Step 7: View Dependency Hierarchy

- Eclipse: Right-click Project → Maven → Show Dependency Hierarchy
- Shows all dependencies and sub-dependencies

HelloController.java

```
package com.cognizant.spring_learn; //

import org.springframework.stereotype.Controller;

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


@Controller

public class HelloController

{

  @GetMapping("/hello")

  public String showHomePage() {

    return "index";

  }

}
```

Index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Spring Boot Web Project</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0; padding: 0;
      background-color: #f4f4f4;
    }
    header {
      background-color: #333;
      padding: 20px;
      color: white;
      text-align: center;
    }
    nav {
      background-color: #555;
      padding: 10px;
      text-align: center;
    }
    nav a {
      color: white;
      margin: 0 15px;
      text-decoration: none;
      font-weight: bold;
    }
    .container {
      padding: 30px;
```

```
        background-color: white;
        margin: 20px auto;
        width: 80%;
        box-shadow: 0 0 10px rgba(0,0,0,0.1);
      }
      footer {
        background-color: #333;
        color: white;
        text-align: center;
        padding: 15px;
        position: fixed;
        width: 100%;
        bottom: 0;
      }
      button {
        padding: 10px 20px;
        background-color: #28a745;
        color: white;
        border: none;
        border-radius: 6px;
        cursor: pointer;
      }
      button:hover {
        background-color: #218838;
      }
    </style>
</head>
<body>

<header>
    <h1>Spring Boot Web Project</h1>
</header>
```

```
<nav>
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Contact</a>
</nav>

<div class="container">
  <h2>Hello, Spring Developer!</h2>
  <p>This is a complete web page built using Spring Boot and Thymeleaf.</p>
  <button onclick="alert('You clicked the button!')">Click Me</button>
</div>

<footer>
  &copy; 2025 My Spring App | Built in Java
</footer>

</body>
</html>
```

Result: