

FINTECH LAB1 REPORT

STEPS:

Step 1: Go to Spring Initializr

- Visit <https://start.spring.io/>.

Step 2: Configure the Project

- Project: Select either Maven or Gradle project.
- Language: Choose Java.
- Spring Boot Version: Select the latest stable version of Spring Boot.
- Group: Your organization name or domain (e.g., com.example).
- Artifact: The name of your project (e.g., myproject).
- Name: The name of your project (e.g., myproject).
- Description: Write a short description (optional).
- Package Name: The root package for your project (e.g., com.example.myproject).
- Packaging: Select Jar (default) or War (if deploying to an application server like Tomcat).
- Java Version: Select the appropriate Java version (11 or 17 are recommended).

Step 3: Add Dependencies

Click on the Add Dependencies button and select the dependencies you need for your project. Some common ones are:

- Spring Web: To create RESTful APIs and web applications.

- Spring Data JPA: For database interaction with JPA/Hibernate.
- H2: For an in-memory database (optional).
- MySQL Driver: If you're using a MySQL database.
- Spring Boot DevTools: For easier development with auto-reload.

3

- Spring Security: For adding security (optional).

Step 4: Generate the Project

- Click on Generate to download the project as a ZIP file.
- Extract the downloaded ZIP to a folder.

Step 5: Import the Project into Your IDE

- Open your preferred IDE (e.g., IntelliJ IDEA, Eclipse, or VS Code).
- In IntelliJ IDEA:
 - o Go to File > Open and select the folder where you extracted the Spring Boot project.
- In Eclipse:
 - o Go to File > Import > Existing Maven Projects.
 - o Browse to the folder and import it.

Step 6: Build and Run the Project

- In IntelliJ IDEA or Eclipse, find the Application.java file under the src/main/java directory. It is usually named after your project name (e.g., MyprojectApplication.java).

@SpringBootApplication

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

```
    SpringApplication.run(MyprojectApplication.class, args);
}
}
```

- Run the main method in the Application.java class to start the Spring Boot application.

Step 7: Verify the Application

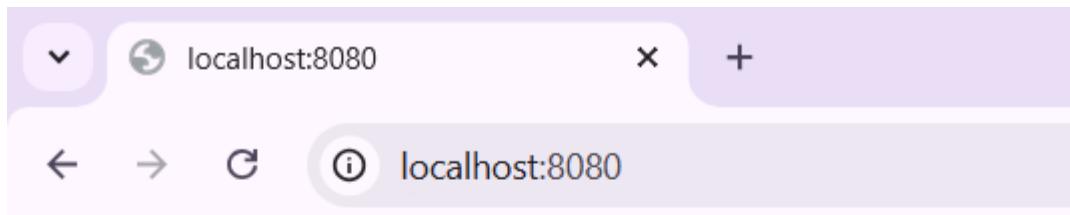
- Open a browser and go to <http://localhost:8080/> (default port).
- If you see the error Whitelabel Error Page, the application has started successfully (since no endpoints have been configured yet).

Lab Exercises:

- 1 Setup the spring boot application and print the hello world message

MyprojectApplication.java

```
package com.example.myproject;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@SpringBootApplication
@RestController
public class MyprojectApplication {
    public static void main(String[] args) {
        SpringApplication.run(MyprojectApplication.class, args);
    }
    @GetMapping("/")
    public String hello() {
        return "Hello World";
    }
}
```



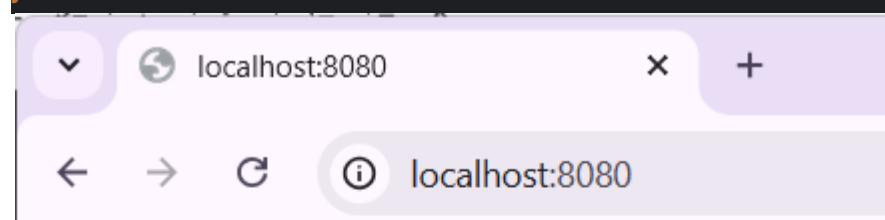
2 Setup the spring boot application with hello world message in separate package named controller

MyprojectApplication.java

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

HelloController.java

```
package com.example.myproject.controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class HelloController {
    @GetMapping("/")
    public String hello() {
        return "Hello World";
    }
}
```



Hello World from controller

3 Test the above application in POSTMAN

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (KrishangJain's Workspace), 'Environments', 'History', and 'Flows'. The main area shows a collection named 'My Collection' with two items: 'Get data' (selected) and 'Post data'. The 'Get data' item is a GET request to 'http://localhost:8080'. The 'Body' tab of the request details shows a single key-value pair: 'Key' and 'Value'. In the response section, the status is '200 OK', time is '9 ms', size is '191 B', and the response body is 'Hello World from controller'. At the bottom, there are tabs for 'Body', 'Cookies', 'Headers (5)', 'Test Results (1/1)', and 'Raw' (which is selected). There are also buttons for 'Save Response', 'Runner', 'Start Proxy', 'Cookies', 'Vault', and 'Trash'.