WEEK 7

React Hands-On Labs

1: Hello World REST API

🔹 Overview:

A simple REST controller that returns "Hello, World!" when accessed via browser or Postman.

🔹 Steps:

1. Create a Spring Boot project.

2. Add dependency: spring-boot-starter-web.

3. Create a Java class with @RestController.

4. Map /hello to return "Hello, World!".

🔹 Java Code:

package com.example.helloworld;

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

public class HelloWorldApplication {

public static void main(String[] args) {

SpringApplication.run(HelloWorldApplication.class, args);

}

}

@RestController

class HelloController {

@GetMapping("/hello")

public String hello() {

return "Hello, World!";

}

}

A screen shot of a computer

AI-generated content may be incorrect.

2: Country REST API

🔹 Overview:

Returns a country name from country.xml.

🔹 Steps:

1. Create a Spring Boot project.

2. Add dependency: spring-boot-starter-web, spring-boot-starter-xml.

3. Create a model class Country.

4. Create country.xml in src/main/resources.

5. Use @RestController to read XML and return Country object.

🔹 country.xml:

Save this under src/main/resources/country.xml.

<country>

<code>IN</code>

<name>India</name>

</country>

🔹 Java Code:

package com.example.countryapi;

import jakarta.xml.bind.annotation.XmlElement;

import jakarta.xml.bind.annotation.XmlRootElement;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.core.io.ClassPathResource;

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

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

import javax.xml.bind.JAXBContext;

import javax.xml.bind.Unmarshaller;

import java.io.File;

@SpringBootApplication

public class CountryApiApplication {

public static void main(String[] args) {

SpringApplication.run(CountryApiApplication.class, args);

}

}

@XmlRootElement(name = "country")

class Country {

private String code;

private String name;

@XmlElement

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

@XmlElement

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

@RestController

class CountryController {

@GetMapping("/country")

public Country getCountry() throws Exception {

File file = new ClassPathResource("country.xml").getFile();

JAXBContext context = JAXBContext.newInstance(Country.class);

Unmarshaller unmarshaller = context.createUnmarshaller();

return (Country) unmarshaller.unmarshal(file);

}

}

A screen shot of a computer code

AI-generated content may be incorrect.

3: Date Format XML Reader

🔹 Overview:

Reads date format from date-format.xml.

🔹 Steps:

1. Create a Spring Boot project.

2. Add spring-boot-starter-web, spring-boot-starter-xml.

3. Create model class DateFormat.

4. Add date-format.xml under src/main/resources.

🔹 date-format.xml:

<dateFormat>

<pattern>dd-MM-yyyy</pattern>

</dateFormat>

🔹 Java Code:

package com.example.dateformatapi;

import jakarta.xml.bind.annotation.XmlElement;

import jakarta.xml.bind.annotation.XmlRootElement;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.core.io.ClassPathResource;

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

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

import javax.xml.bind.JAXBContext;

import javax.xml.bind.Unmarshaller;

import java.io.File;

@SpringBootApplication

public class DateFormatApiApplication {

public static void main(String[] args) {

SpringApplication.run(DateFormatApiApplication.class, args);

}

}

@XmlRootElement(name = "dateFormat")

class DateFormat {

private String pattern;

@XmlElement

public String getPattern() {

return pattern;

}

public void setPattern(String pattern) {

this.pattern = pattern;

}

}

@RestController

class DateFormatController {

@GetMapping("/date-format")

public DateFormat getDateFormat() throws Exception {

File file = new ClassPathResource("date-format.xml").getFile();

JAXBContext context = JAXBContext.newInstance(DateFormat.class);

Unmarshaller unmarshaller = context.createUnmarshaller();

return (DateFormat) unmarshaller.unmarshal(file);

}

}

A screen shot of a computer

AI-generated content may be incorrect.

4: REST API with Path Variable

🔹 Overview:

Returns a greeting for the user via path variable.

🔹 Steps:

1. Create Spring Boot project.

2. Add dependency: spring-boot-starter-web.

3. Use @PathVariable in controller.

🔹 Java Code:

package com.example.greetapi;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.web.bind.annotation.\*;

@SpringBootApplication

public class GreetApiApplication {

public static void main(String[] args) {

SpringApplication.run(GreetApiApplication.class, args);

}

}

@RestController

class GreetController {

@GetMapping("/greet/{name}")

public String greetUser(@PathVariable String name) {

return "Hello, " + name + "!";

}

}

A screen shot of a computer

AI-generated content may be incorrect.

5: REST API Returning List of Countries

🔹 Overview:

Returns a list of countries using List<Country>.

🔹 Steps:

1. Create Spring Boot project.

2. Create Country class.

3. Create endpoint that returns list.

🔹 Java Code:

package com.example.countrieslist;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Arrays;

@SpringBootApplication

public class CountriesListApplication {

public static void main(String[] args) {

SpringApplication.run(CountriesListApplication.class, args);

}

}

class Country {

private String code;

private String name;

public Country(String code, String name) {

this.code = code;

this.name = name;

}

public String getCode() {

return code;

}

public String getName() {

return name;

}

}

@RestController

class CountriesController {

@GetMapping("/countries")

public List<Country> getCountries() {

return Arrays.asList(

new Country("IN", "India"),

new Country("US", "United States"),

new Country("JP", "Japan")

);

}

}

A screen shot of a computer code

AI-generated content may be incorrect.