SPRING REST WEEK 4

Hands on 1

1. Create a Spring Web Project using Maven
2. src/main/java - Folder with application code:

SpringLearnApplication.java – Main class to start the app

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

HelloController.java – Sample REST controller to test endpoints:

package com.cognizant.spring\_learn;

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

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

*@RestController*

public class HelloController {

*@GetMapping*("/")

public String hello() {

return "Spring Boot is running!";

}

}

1. src/main/resources – Configuration Files:

application.properties

spring.application.name=spring-learn

server.port=9090

1. src/test/java – Testing Folder

package com.cognizant.spring\_learn;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class SpringLearnApplicationTests {

@Test

void contextLoads() {

}

}

6. Key Configurations in pom.xml

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- DevTools (optional, for auto-restart) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Test dependencies (JUnit, Mockito, etc.) -->

...

</dependencies>

Dependency hierarchy

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Hands on 4**

2. Spring Core – Load Country from Spring Configuration XML

<bean> Tag:

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<**bean** id=*"country"* class=*"com.cognizant.springlearn.Country"*>

<**property** name=*"code"* value=*"IN"* />

<**property** name=*"name"* value=*"India"* />

</**bean**>

</**beans**>

2. SimpleLearnApplication.java

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(SpringLearnApplication.class);

public static void main(String[] args) {

***LOGGER***.info("START");

*displayCountry*();

***LOGGER***.info("END");

}

public static void displayCountry() {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = context.getBean("country", Country.class);

***LOGGER***.debug("Country : {}", country.toString());

}

}

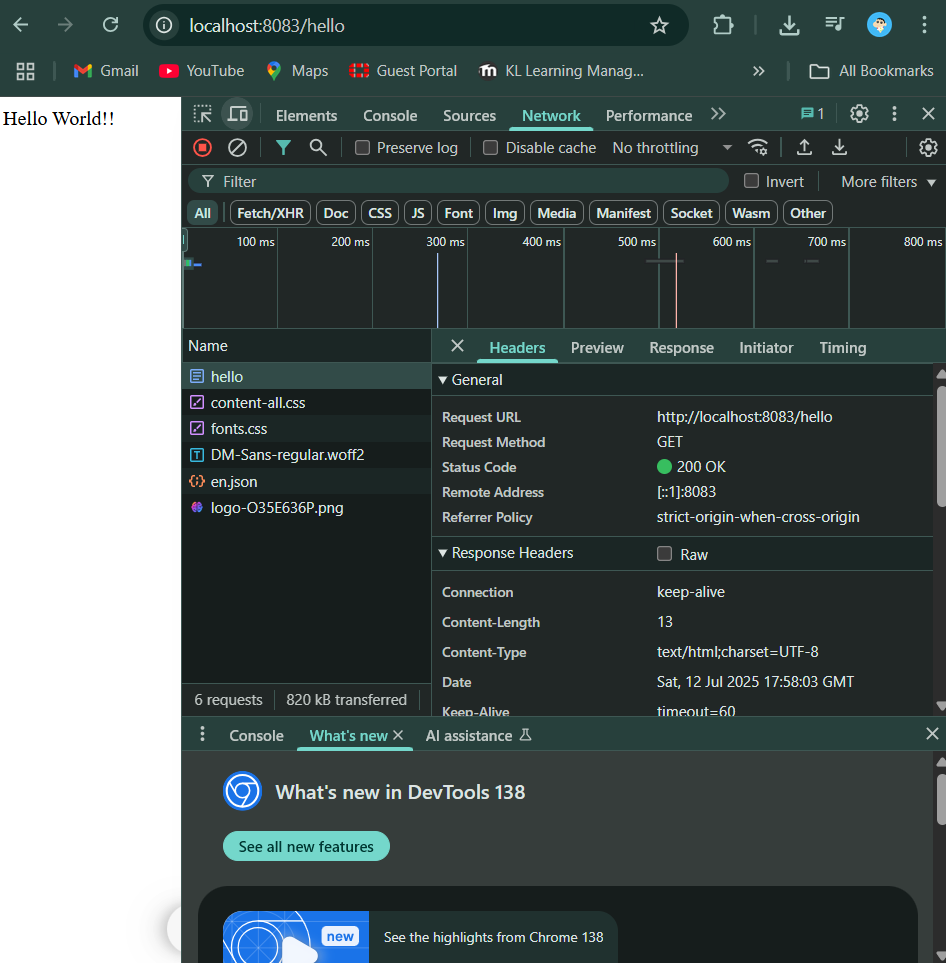
A screenshot of a computer program

AI-generated content may be incorrect.

1. Hello World RESTful Web Service

* HelloController.java
* package com.cognizant.springlearn.controller;
* import org.slf4j.Logger;
* import org.slf4j.LoggerFactory;
* import org.springframework.web.bind.annotation.GetMapping;
* import org.springframework.web.bind.annotation.RestController;
* *@RestController*
* public class HelloController {
* private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(HelloController.class);
* *@GetMapping*("/hello")
* public String sayHello() {
* ***LOGGER***.info("START sayHello()");
* String message = "Hello World!!";
* ***LOGGER***.info("END sayHello()");
* return message;
* }
* A screenshot of a computer

  AI-generated content may be incorrect.}
* In network tab of developer tools show the HTTP header details received



In postman click on "Headers" tab to view the HTTP header details received:

A screenshot of a computer

AI-generated content may be incorrect.

1. REST - Country Web Service

CountryController.java:

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

*@RestController*

public class CountryController {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(CountryController.class);

*@RequestMapping*("/country")

public Country getCountryIndia() {

***LOGGER***.info("START getCountryIndia()");

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = (Country) context.getBean("country");

***LOGGER***.info("END getCountryIndia()");

return country;

}

}

A screenshot of a computer

AI-generated content may be incorrect.

**1. What happens in the controller method?**

* When /country URL is hit:
  1. Spring maps it to getCountryIndia() method.
  2. This method loads the country bean from country.xml.
  3. Spring automatically converts the returned Country object into JSON using **Jackson** (added via spring-boot-starter-web).

**How is the Java Bean converted to JSON?**

* Spring Boot includes **Jackson** by default (via spring-boot-starter-web).
* When a method returns an object:
  + Spring’s @RestController uses HttpMessageConverter
  + Jackson serializes the object into JSON
* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. REST - Get country based on country code

* @GetMapping("/country/{code}"):  
  Maps URL to getCountry() method
* @PathVariable:  
  Extracts {code} from URL
* Calls countryService.getCountry(code)

Spring Boot uses **Jackson (ObjectMapper)** to automatically serialize Java objects into JSON.

* The Country object returned by the controller is converted to JSON because of @RestController.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

postman :

A screenshot of a computer

AI-generated content may be incorrect.

**Create authentication service that returns JWT** 

SECURITYCONFIG.JAVA:

package com.cognizant.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@*EnableWebSecurity

public class SecurityConfig {

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

}

JWTAAUTHAPPLICATION.JAVA:

package com.cognizant.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@*EnableWebSecurity

public class SecurityConfig {

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

}

