Harshitha M

Superset id : 6390499

WEEK 4

Spring Boot Hands-On

# Hands-On 1: Create Spring Web Project Using Maven

## SpringLearnApplication.java:

@SpringBootApplication  
public class SpringLearnApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(SpringLearnApplication.class, args);  
 System.out.println("SpringLearnApplication has started successfully.");  
 }  
}

## Project Structure Explanation:

* src/main/java - Contains application code.
* src/main/resources - Configuration files like application.properties.
* src/test/java - Test classes.
* SpringLearnApplication.java - Entry point of the application with main() method.
* @SpringBootApplication - A convenience annotation that combines @Configuration, @EnableAutoConfiguration, and @ComponentScan.
* pom.xml - Manages dependencies and configurations.
* **Output:**

A screenshot of a computer code

AI-generated content may be incorrect.

# Hands-On 2: Load Country from Spring Configuration XML

## Steps:

* 1. Create a file named date-format.xml in src/main/resources.
* 2. Add the following XML configuration:

<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  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="dateFormat" class="java.text.SimpleDateFormat">  
 <constructor-arg value="dd/MM/yyyy" />  
 </bean>  
</beans>

* 3. Add a method displayDate() in SpringLearnApplication.java that loads the context and parses the date.

## Updated SpringLearnApplication.java:

@SpringBootApplication  
public class SpringLearnApplication {  
  
 public static void main(String[] args) throws Exception {  
 SpringApplication.run(SpringLearnApplication.class, args);  
 System.out.println("SpringLearnApplication has started successfully.");  
 displayDate();  
 }  
  
 public static void displayDate() throws Exception {  
 ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");  
 SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);  
 String dateStr = "31/12/2018";  
 Date date = format.parse(dateStr);  
 System.out.println("Parsed Date: " + date);  
 }  
}  
A screenshot of a computer

AI-generated content may be incorrect.

**Handson-3: Hello World RESTful Web Service**

**HelloController.java:**

Steps:

1. Create a new class named HelloController in package com.cognizant.spring-learn.controller.

2. Annotate the class with @RestController.

3. Create a method sayHello() and annotate with @GetMapping("/hello").

4. Log 'Start' and 'End' of the method using System.out.println().

5. Return the string "Hello World!!".

6. Run the application and open http://localhost:8083/hello in browser or Postman.

7. Verify the response and headers in developer tools or Postman.

**HelloController.java:**

@RestController  
public class HelloController {  
  
 @GetMapping("/hello")  
 public String sayHello() {  
 System.out.println("START - sayHello()");  
 String message = "Hello World!!";  
 System.out.println("END - sayHello()");  
 return message;  
 }  
}

A screen shot of a computer program

AI-generated content may be incorrect.

**Handson-4: REST - Country Web Service**

Steps:

**Steps:**

1. Create a new class named CountryController in com.cognizant.spring-learn.controller.

2. Define a method getCountryIndia() annotated with @RequestMapping("/country").

3. Load 'country.xml' in ApplicationContext.

4. Retrieve the 'in' bean using context.getBean("in", Country.class).

5. Return the Country object - Spring Boot will auto-convert it to JSON.

6. Run the app and access http://localhost:8083/country.

7. Check JSON response and HTTP headers in browser and Postman.

**CountryController.java:**

@RequestMapping("/country")  
public Country getCountryIndia() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = (Country) context.getBean("in");  
 System.out.println("Returning country: " + country);  
 return country;  
}

A screenshot of a computer

AI-generated content may be incorrect.

**Handson-5: REST - Get Country Based on Country Code**

Steps:

1. In CountryController, add method getCountry(String code) annotated with @GetMapping("/countries/{code}").

2. Inject a CountryService using @Autowired.

3. In service, load country list from XML and match using case-insensitive code.

4. Return the matched Country object.

5. Run the app and test: http://localhost:8083/countries/in.

6. Check the response in browser and Postman.

CountryController.java:  
@GetMapping("/countries/{code}")  
public Country getCountry(@PathVariable String code) throws CountryNotFoundException {  
 System.out.println("START - getCountry(): code=" + code);  
 Country country = countryService.getCountry(code);  
 System.out.println("END - getCountry(): found=" + country);  
 return country;  
}

CountryService.java:  
public Country getCountry(String code) throws CountryNotFoundException {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> countries = (List<Country>) context.getBean("countryList");  
  
 return countries.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElseThrow(() -> new CountryNotFoundException("Country not found"));  
}

