**Hands on 1: Create a Spring Web Project using Maven**

**Step 1 – Create Maven Project Structure**  
mvn archetype:generate -DgroupId=com.cognizant -DartifactId=spring-learn -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

**Step 2 – Add Spring Dependencies in pom.xml**

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Optional: DevTools -->

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<!-- Optional: Spring Boot Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<parent>

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

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

<version>3.2.2</version>

</parent>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

**Step 3 – Create Main Application Class**

package com.cognizant.springlearn;

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);

}

}

**Step 4 – Create a Simple Controller**

package com.cognizant.springlearn.controller;

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

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

@RestController

public class HelloController {

@GetMapping("/hello")

public String hello() {

return "Hello, Spring Web with Maven!";

}

}

**Step 5 – Build and Run the Application**

mvn clean install

mvn spring-boot:run

**Hands-on 2: Spring Core – Load Country from Spring Configuration XML**

**Step 1 – Create country.xml in src/main/resources**

<?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>

**Step 2 – Create Country.java class**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

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

private String code;

private String name;

public Country() {

LOGGER.debug("Inside Country Constructor.");

}

public String getCode() {

LOGGER.debug("Getting code: {}", code);

return code;

}

public void setCode(String code) {

LOGGER.debug("Setting code: {}", code);

this.code = code;

}

public String getName() {

LOGGER.debug("Getting name: {}", name);

return name;

}

public void setName(String name) {

LOGGER.debug("Setting name: {}", name);

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**Step 3 – Add displayCountry() in SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

}

public static void displayCountry() {

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

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

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

}

}

Inside Country Constructor.

Setting code: IN

Setting name: India

Getting code: IN

Getting name: India

Country: Country [code=IN, name=India]

**SME Talking Points for Review**

* <bean>: Defines a bean (an object to be managed by Spring).
* id: Unique name to refer to the bean.
* class: Fully qualified class name of the bean.
* <property>: Used to inject values into the bean fields.
* name: Name of the setter (e.g., setCode, setName).
* value: Literal value to inject.
* ApplicationContext: Central interface to access Spring beans.
* ClassPathXmlApplicationContext: Loads Spring context from XML on classpath.
* context.getBean("beanId", Class.class): Retrieves and returns the bean instance from Spring's IOC container.

**Hands-on 3: Hello World RESTful Web Service**

**Step 1 – Create 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()");

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

return "Hello World!!";

}

}

**Step 2 – Run the Application**

Run the SpringLearnApplication.java file.

Access in browser or Postman:

GET <http://localhost:8083/hello>

**Sample Output**

**Response:**

Hello World!!

**Log Output**

INFO HelloController - START sayHello()

INFO HelloController - END sayHello()

**SME Talking Points**

**In Chrome Developer Tools → Network Tab:**

* Open browser and visit http://localhost:8083/hello
* Right-click → Inspect → Network tab → Click on /hello request
* View HTTP headers like:
  + Request Method: GET
  + Status Code: 200 OK
  + Content-Type: text/plain;charset=UTF-8
  + Server: Apache Tomcat

**In Postman → Headers Tab:**

* Open Postman → Send GET request to /hello
* Click on **Headers** tab to see response headers like:
  + Content-Type
  + Date
  + Content-Length
  + Server

**Hands-on 4: REST – Country Web Service**

**Step 1 – Create 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 ClassPathXmlApplicati

onContext("country.xml");

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

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

return country;

}

}

**Step 2 – Sample Request**

Request URL: <http://localhost:8083/country>

Response: {

"code": "IN",

"name": "India"

}

**SME Talking Points**

**What happens in the controller method?**

* ApplicationContext loads country.xml
* The Country bean is fetched using context.getBean()
* Spring returns this Java object as the response

**How is the bean converted into JSON?**

* Spring Boot uses **Jackson** (auto-included in spring-boot-starter-web) to convert Java objects (POJOs) to JSON automatically
* @RestController + return type as Country enables auto JSON conversion

**Inspect Headers**

**In Chrome (DevTools → Network Tab → /country request):**

* **Status Code**: 200 OK
* **Content-Type**: application/json
* **Request Method**: GET

**In Postman (Headers Tab):**

* Key headers:
  + Content-Type: application/json
  + Content-Length
  + Date
  + Server: Apache-Coyote or Tomcat

**Hands-on 5: REST – Get Country Based on Country Code**

**Step 1 – Update country.xml to Include List of Countries**

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<bean class="com.cognizant.springlearn.Country">

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

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

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="US" />

<property name="name" value="United States" />

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="JP" />

<property name="name" value="Japan" />

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="DE" />

<property name="name" value="Germany" />

</bean>

</list>

</constructor-arg>

</bean>

**Step 2 – Create CountryService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) {

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

List<Country> countryList = context.getBean("countryList", List.class);

return countryList.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElse(null);

}

}

**Step 3 – Update CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;

import com.cognizant.springlearn.service.CountryService;

import org.springframework.beans.factory.annotation.Autowired;

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/countries/{code}")

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

**Step 4 – Sample Request**

GET <http://localhost:8083/countries/in>

Sample Response

{

"code": "IN",

"name": "India"

}

**Hands-on 6: JWT Authentication Service – Generate JWT Token**

**Step 1 – Add Dependencies in pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

**Step 2 – Create JwtUtil.java Utility Class**

package com.cognizant.springlearn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String SECRET\_KEY = "secret";

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 min expiry

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**Step 3 – Create AuthenticationController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {

// Basic dXNlcjpwd2Q=

String base64Credentials = authHeader.substring("Basic ".length());

byte[] decoded = Base64.getDecoder().decode(base64Credentials);

String credentials = new String(decoded);

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

if ("user".equals(username) && "pwd".equals(password)) {

String token = JwtUtil.generateToken(username);

return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");

} else {

return ResponseEntity.status(401).body("Invalid Credentials");

}

}

}

**Step 4 – Configure Port in application.properties**

server.port=8090

**Step 5 – Run and Test**

Use this curl command: curl -s -u user:pwd <http://localhost:8090/authenticate>

Sample Response: {"token":"<jwt-token-value>"}