**Spring rest**

**Exercise 1: Create a Spring Web Project using Maven**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

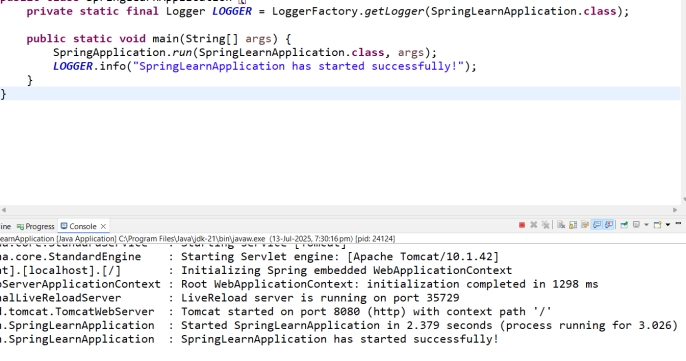
SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("SpringLearnApplication has started successfully!");

}

}

**Output:**

****

**Exercise 2: Spring Core – Load Country from Spring Configuration XML**

**Display Date Method**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

importorg.springframework.

boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.

context.support.ClassPathXmlApplicationContext;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.text.SimpleDateFormat;

import java.util.Date;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.class, args);

*LOGGER*.info("SpringLearnApplication has started successfully!");

*displayDate*();

}

public static void displayDate() {

// Load XML configuration

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

// Get the SimpleDateFormat bean

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

try {

Date date = format.parse("31/12/2018");

System.*out*.println("Parsed Date: " + date);

} catch (Exception e) {

e.printStackTrace();

}

}

}

**date-format.xml**

<?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

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

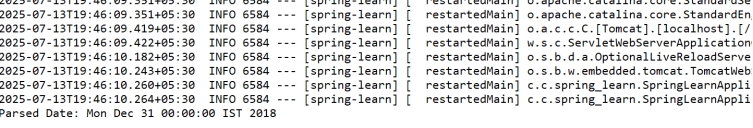
<bean id="dateFormat" class="java.text.SimpleDateFormat">

<constructor-arg value="dd/MM/yyyy" />

</bean>

</beans>

**Output:**



**Exercise 3:** **Hello World RESTful Web Service**

**SpringLearnApplication.java**

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**

package com.cognizant.spring\_learn.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 response = "Hello World!!";

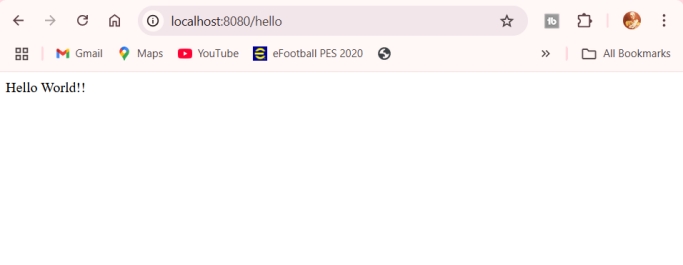
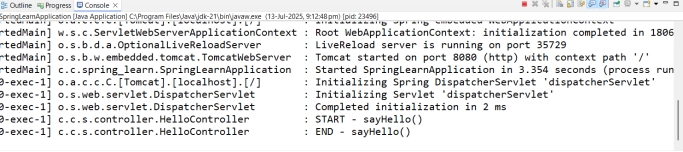
*LOGGER*.info("END - sayHello()");

return response;

}

}

**Output:**

****

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

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import java.util.List

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

List<Country> countries = countryService.getAllCountries();

LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

}

CountryController.java

package com.cognizant.orm\_learn.CountryController;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

@RequestMapping("/country")

public Country getCountryIndia() {

return countryService.getCountry("IN");

}

}

Country.java

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

private String name;

// Getters and Setters

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

@Override

public String toString() {

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

}

}

CountryRepository.java

package com.cognizant.orm\_learn.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.cognizant.orm\_learn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

CountryService.java

package com.cognizant.orm\_learn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

@Transactional

public Country getCountry(String code) {

return countryRepository.findById(code.toUpperCase()).orElse(null);

}

}

Pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orm-learn</name>

<description>Demo project for Spring Boot and JPA</description>

<parent>

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

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

<version>3.2.5</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Data JPA -->

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- MySQL Driver -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot DevTools (Optional) -->

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<!-- JUnit 5 and Spring Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<release>${java.version}</release>

</configuration>

</plugin>

</plugins>

</build>

</project>

application.properties

spring.application.name=orm-learn

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# DB Connection

# ===============================

# = DATABASE CONFIGURATION =

# ===============================

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# ===============================

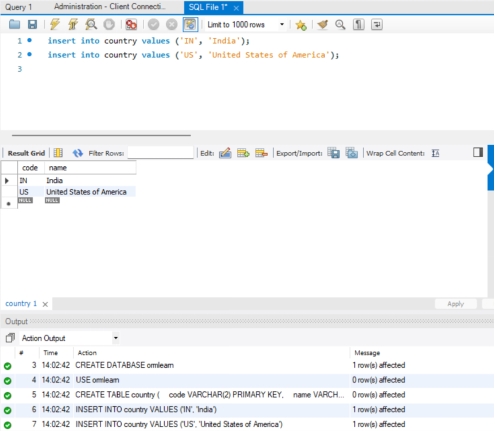
# = JPA & HIBERNATE =

# ===============================

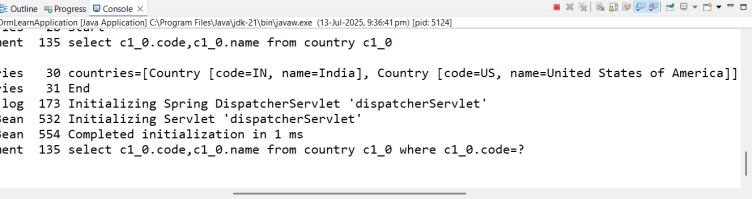
spring.jpa.hibernate.ddl-auto=none

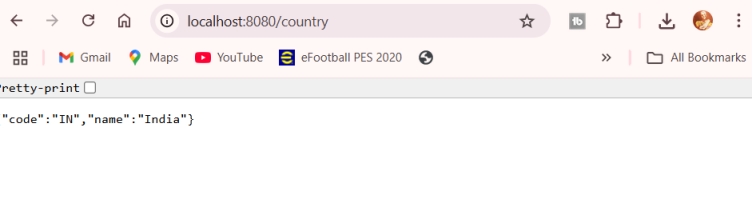
spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect



**Output:**

****

****

**Exercise 5: REST - Get country based on country code**

**CountryService.java**

package com.cognizant.orm\_learn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

@Transactional

public Country getCountry(String code) {

return countryRepository.findById(code.toUpperCase()).orElse(null);

}

@Transactional

public Country getCountry(String code) {

return countryRepository.findById(code.toUpperCase()).orElse(null);

}

}

**CountryController.java**

package com.cognizant.orm\_learn.CountryController;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

@RequestMapping("/country")

public Country getCountryIndia() {

return countryService.getCountry("IN");

}

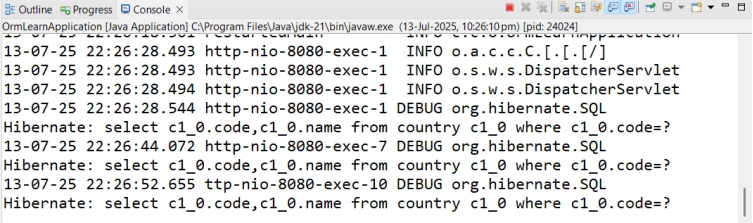
@GetMapping("/countries/{code}")

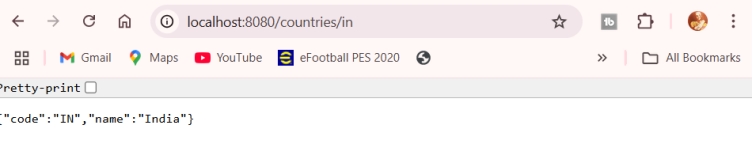
public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

**Output: **

****

**JWT-handson**

**Exercise 6: Create authentication service that returns JWT**

**Pom.xml :**

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>auth-service</artifactId>

<version>1.0-SNAPSHOT</version>

<parent>

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

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

<version>3.2.0</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JSON Web Token (JJWT) with Jackson support -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId> <!-- for JSON serialization -->

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<!-- Lombok (Optional, for cleaner code) -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<!-- Spring Boot Test (Optional for unit testing) -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**AuthServiceApplication.java :**

package com.example.auth\_service\_1;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AuthServiceApplication {

public static void main(String[] args) {

SpringApplication.*run*(AuthServiceApplication.class, args);

}

}

**AuthenticationController.java** :

package com.example.auth\_service\_1.AuthenticationController;

import com.example.auth\_service\_1.JwtUtil.\*;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

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

@RestController

public class AuthenticationController {

private final JwtUtil jwtUtil;

public AuthenticationController(JwtUtil jwtUtil) {

this.jwtUtil = jwtUtil;

}

@RequestMapping("/authenticate")

public ResponseEntity<?> authenticate(Authentication authentication) {

String username = authentication.getName();

String token = jwtUtil.generateToken(username);

return ResponseEntity.*ok*().body(new JwtResponse(token));

}

// Response DTO

static class JwtResponse {

private String token;

public JwtResponse(String token) { this.token = token; }

public String getToken() { return token; }

public void setToken(String token) { this.token = token; }

}

}

SecurityConfig.java :

package com.example.auth\_service\_1.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.core.userdetails.\*;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User

.*withUsername*("user")

.password("{noop}pwd") // {noop} = no password encoding

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").authenticated()

)

.httpBasic(); // Enable Basic Auth

return http.build();

}

}

**JwtUtil.java :**

package com.example.auth\_service\_1.JwtUtil;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

@Component

public class JwtUtil {

// The key should be at least 256 bits (32 characters for HS256)

private final String SECRET\_KEY = "A1b2C3d4E5f6G7h8I9j0K!LmN@OpQrSt";

private final Key key = Keys.*hmacShaKeyFor*(SECRET\_KEY.getBytes());

public String generateToken(String username) {

long now = System.*currentTimeMillis*();

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date(now))

.setExpiration(new Date(now + 1000 \* 60 \* 60)) // 1 hour

.signWith(key, SignatureAlgorithm.*HS256*)

.compact();

}

}

**Output :**

