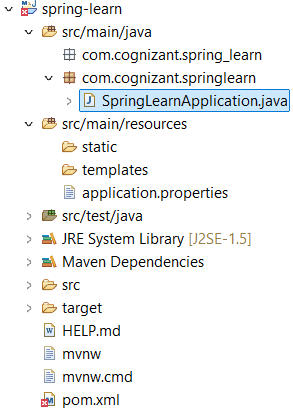
**Hands on 1**

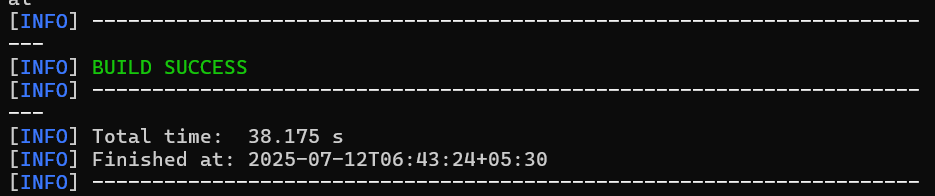
**Create a Spring Web Project using Maven**   
SME to walk through the following aspects related to the project created:

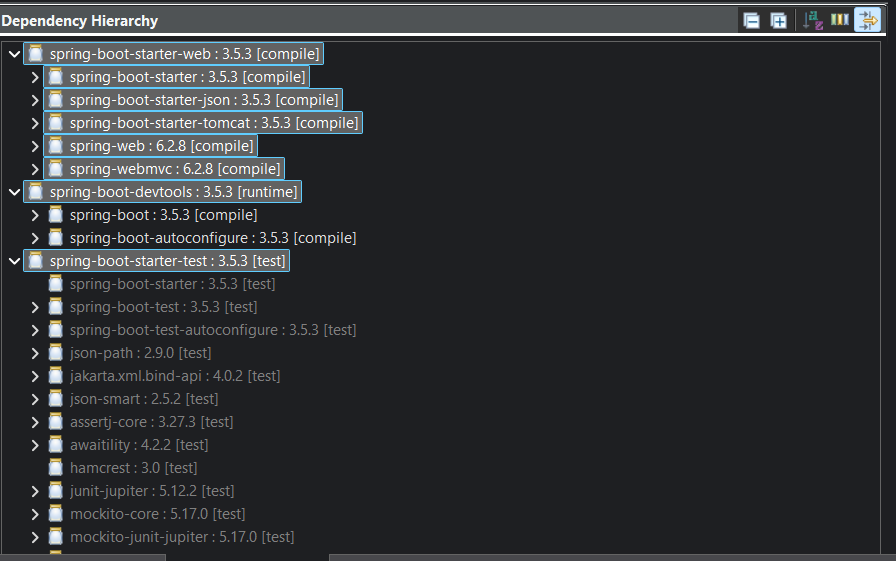
1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. SpringLearnApplication.java - Walkthrough the main() method.
5. Purpose of @SpringBootApplication annotation
6. pom.xml
   1. Walkthrough all the configuration defined in XML file
   2. Open 'Dependency Hierarchy' and show the dependency tree.

**Output :**

**The Spring Boot project was created using Spring Initializr and built successfully using Maven. Dependency Hierarchy was viewed in Eclipse to inspect project dependencies.**



****



**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration XML** 

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>

CODE:

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@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("START");

*displayDate*();

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

}

public static void displayDate() {

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

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

try {

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

***LOGGER***.debug("Parsed Date: {}", date);

} catch (Exception e) {

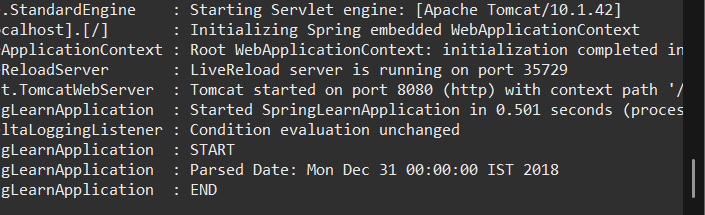
***LOGGER***.error("Date parsing failed", e);

}

}

}

OUTPUT:



**Exercise 3: 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");

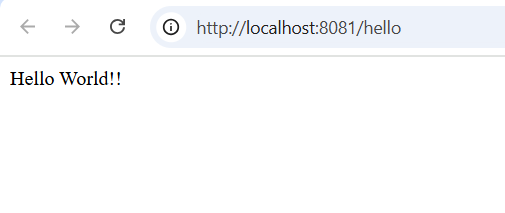
LOGGER.info("END");

return "Hello World!!";

}

}

**Output:**

****

**Exercise 2: 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.web.bind.annotation.RequestMapping;

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@RestController

public class CountryController {

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

@RequestMapping("/country")

public Country getCountryIndia() {

LOGGER.info("START");

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

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

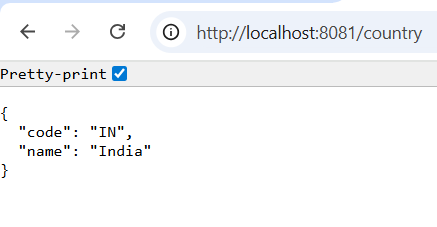
LOGGER.debug("Country: {}", country);

LOGGER.info("END");

return country;

}

}  
  
**Output:**



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

**//** **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.GetMapping;

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

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

import java.util.List;

@RestController

public class CountryController {

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

@GetMapping("/country")

public Country getCountryIndia() {

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

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

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

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

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

return country;

}

@GetMapping("/countries")

public List<Country> getAllCountries() {

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

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

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

LOGGER.debug("Country List: {}", countries);

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

return countries;

}

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

public Country getCountry(@PathVariable String code) {

LOGGER.info("START getCountry() with code: {}", code);

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

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

for (Country c : countries) {

if (c.getCode().equalsIgnoreCase(code)) {

LOGGER.info("END getCountry() - country found");

return c;

}

}

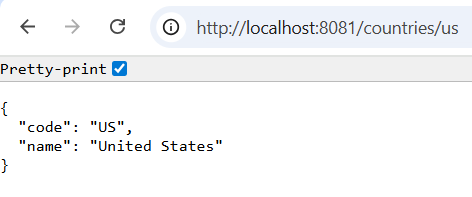
LOGGER.info("END getCountry() - country not found");

return null; // or throw exception

}

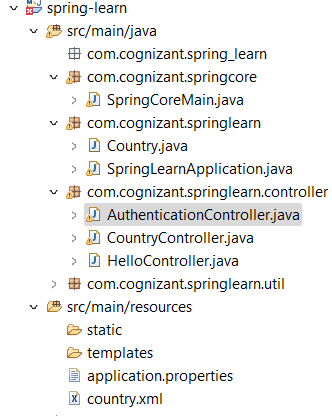
}

**//Output:**

****

****

**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*

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

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.2.5</version>

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

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Demo Spring Web Project with Logging</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot DevTools -->

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<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>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<mainClass>com.cognizant.springlearn.SpringLearnApplication</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**//** **AuthenticationController**

**package** com.cognizant.springlearn.controller;

**import** com.cognizant.springlearn.util.JwtUtil;

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

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.util.Base64Utils;

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

**import** java.nio.charset.StandardCharsets;

@RestController

**public** **class** AuthenticationController {

@Autowired

**private** JwtUtil jwtUtil;

@GetMapping("/authenticate")

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

String[] credentials = extractCredentials(authHeader);

String username = credentials[0];

String password = credentials[1];

// Normally verify with DB or in-memory users; for now hardcoded

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

String token = jwtUtil.generateToken(username);

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

} **else** {

**return** ResponseEntity.*status*(401).body("Unauthorized");

}

}

**private** String[] extractCredentials(String authHeader) {

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

**byte**[] credDecoded = Base64Utils.decodeFromString(base64Credentials);

String credentials = **new** String(credDecoded, StandardCharsets.***UTF\_8***);

**return** credentials.split(":", 2);

}

}

**// JwtUtil.java**

package com.cognizant.springlearn.util;

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 {

private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256); // Generates secure random key

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

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

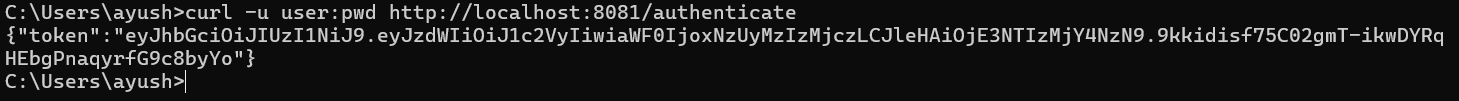
.signWith(key)

.compact();

}

}

**Output:**

****