**WEEK-4 Mandatory and additional exercise**

**Hands on 1**

**Create a Spring Web Project using Maven**

A screenshot of a computer

AI-generated content may be incorrect.

SpringLearnApplication.java

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

System.out.println("SpringLearnApplication started...");

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**OUTPUT:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Hands on 2**

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

**SpringLearnApplication.java**

package com.cognizant;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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);

*displayDate*();

}

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");

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

} catch (ParseException e) {

System.***out***.println("Date parsing failed: " + e.getMessage());

}

((ClassPathXmlApplicationContext) context).close(); // Good practice to close the context

}

}

**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:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Hello World RESTful Web Service**   
  
**Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework.**

**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 response = "Hello World!!";

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

return response;

}

}

**OUTPUT:**

**A white background with a brown border

AI-generated content may be incorrect.**

**REST - Country Web Service   
  
Write a REST service that returns India country details in the earlier created spring learn application.**

**Country.java**

**package com.cognizant.springlearn.model;**

**public class Country {**

**private String code;**

**private String name;**

**public Country() {}**

**public Country(String code, String name) {**

**this.code = code;**

**this.name = name;**

**}**

**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 + "]";**

**}**

**}**

**Country.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=*"in"* class=*"com.cognizant.springlearn.model.Country"*>**

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

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

**</bean>**

**</beans>**

**CountryController.java:**

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

**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;**

**import com.cognizant.springlearn.model.Country;**

***@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("in");**

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

**((ClassPathXmlApplicationContext) context).close();**

**return country;**

**}**

**}**

**OUTPUT:**

**A black screen with a brown border

AI-generated content may be incorrect.**

**REST - Get country based on country code   
  
Write a REST service that returns a specific country based on country code. The country code should be case insensitive.**

**Country.java:  
package com.cognizant.springlearn.model;**

**public class Country {**

**private String code;**

**private String name;**

**public Country() {**

**}**

**public Country(String code, String name) {**

**this.code = code;**

**this.name = name;**

**}**

**public String getCode() {**

**return code;**

**}**

**public void setCode(String code) {**

**this.code = code.toUpperCase();**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**}**

**Country.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=*"countryList"* class=*"java.util.ArrayList"*>**

**<constructor-arg>**

**<list>**

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

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

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

**</bean>**

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

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

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

**</bean>**

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

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

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

**</bean>**

**</list>**

**</constructor-arg>**

**</bean>**

**</beans>**

**CountryController.java:**

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

**import com.cognizant.springlearn.model.Country;**

**import com.cognizant.springlearn.service.CountryService;**

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

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

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

***@RestController***

**public class CountryController {**

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

***@Autowired***

**private CountryService countryService;**

***@GetMapping*("/country/{code}")**

**public Country getCountry(*@PathVariable* String code) {**

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

**Country country = countryService.getCountry(code);**

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

**return country;**

**}**

**}**

**CountryService.java:**

**package com.cognizant.springlearn.service;**

**import com.cognizant.springlearn.model.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 {**

**private ApplicationContext context;**

**public Country getCountry(String code) {**

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

**List<Country> bean = (List<Country>) context.getBean("countryList");**

**List<Country> countryList = bean;**

**return countryList.stream()**

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

**.findFirst()**

**.orElse(null);**

**}**

**}**

**OUTPUT:**

**A black rectangular object with a brown border

AI-generated content may be incorrect.**

**Create authentication service that returns JWT   
  
As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.  
  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.**

**JwtUtil.java:  
package com.cognizant.springlearn.util;**

**import io.jsonwebtoken.Jwts;**

**import io.jsonwebtoken.SignatureAlgorithm;**

**import org.springframework.stereotype.Component;**

**import java.util.Date;**

***@Component***

**public class JwtUtil {**

**private final String secretKey = "yourSecretKey";**

**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(*SignatureAlgorithm*.*HS256*, secretKey)**

**.compact();**

**}**

**}**

**AuthenticationController.java:**

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

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

**import jakarta.servlet.http.HttpServletRequest;**

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

**import org.springframework.http.HttpStatus;**

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

**import org.springframework.web.server.ResponseStatusException;**

**import java.util.Base64;**

**import java.util.HashMap;**

**import java.util.Map;**

***@RestController***

**public class AuthenticationController {**

***@GetMapping*("/authenticate")**

**public Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {**

**// Decode the header manually (Basic Auth comes in encoded format)**

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

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

**String credentials = new String(credDecoded);**

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

**String username = values[0];**

**String password = values[1];**

**// Validate credentials (for demo: accept any fixed username/pwd)**

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

**throw new ResponseStatusException(*HttpStatus*.*UNAUTHORIZED*, "Invalid Credentials");**

**}**

**// Generate dummy JWT token (or use a JWT library)**

**String token = "dummy-jwt-token-for-" + username;**

**Map<String, String> response = new HashMap<>();**

**response.put("token", token);**

**return response;**

**}**

**}**

**SecurityConfig.java:**

**package com.cognizant.springlearn.config;**

**import org.springframework.context.annotation.Bean;**

**import org.springframework.context.annotation.Configuration;**

**import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;**

**import org.springframework.security.web.SecurityFilterChain;**

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

***@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()**

**)**

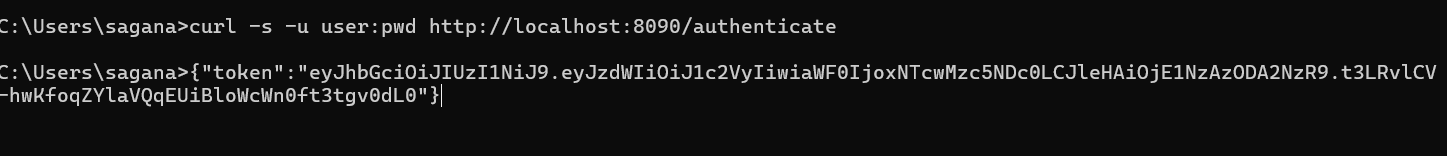
**.httpBasic();**

**return http.build();**

**}**

**}**

**OUTPUT:**

****