**Week 4 Solutions**

**Mandatory Hands-On**

**Spring-rest Hands-On-1**

**Exercise 1**

**Code**

**SpringLearnApplication**

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) {  
 System.*out*.println("Application Started...");  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 }  
  
}

**application.properties**

spring.application.name=spring-learn  
server.port=8081

**Output**

**A black rectangular object with a black border

AI-generated content may be incorrect.**

Port 8080 was occupied, so I used port 8081

**Exercise 4**

**SpringLearnApplication.java**

package com.cognizant.springlearn;  
  
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 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());  
 }  
}

**Country.java**

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 country code: {}", code);  
 return code;  
 }  
  
 public void setCode(String code) {  
 *LOGGER*.debug("Setting country code: {}", code);  
 this.code = code;  
 }  
  
 public String getName() {  
 *LOGGER*.debug("Getting country name: {}", name);  
 return name;  
 }  
  
 public void setName(String name) {  
 *LOGGER*.debug("Setting country name: {}", name);  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country{" +  
 "code='" + code + '\'' +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

**application.properties**

logging.level.com.cognizant.springlearn=DEBUG

**OUTPUT**

**A computer screen with many small squares

AI-generated content may be incorrect.**

**Spring-rest Hands-On-2**

**Hello World RESTful Web Service**

**HelloController.java**

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 sayHello() {  
 return "Hello World!!";  
 }  
}

**Output**

**A screen shot of a computer

AI-generated content may be incorrect.** **A screenshot of a computer

AI-generated content may be incorrect.**

**REST - Country Web Service**

**CountryController**

package com.cognizant.springlearn.controller;  
  
import com.cognizant.springlearn.Country;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class CountryController {  
  
 @Autowired  
 private Country country; // Spring injects the "in" bean  
  
 @RequestMapping("/country")  
 public Country getCountryIndia() {  
 return country;  
 }  
}

**Country**

package com.cognizant.springlearn;  
  
public class Country {  
 private String code;  
 private String 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;  
 }  
}

**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.Country">  
 <property name="code" value="IN" />  
 <property name="name" value="India" />  
 </bean>  
  
</beans>

**Output**

**A black and green rectangle with a grey stripe

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**REST - Get country based on country code**

**SpringLearn**

package com.cognizant.springlearn;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.ImportResource;  
  
@SpringBootApplication  
@ImportResource("classpath:country.xml") // Load the XML bean configuration  
public class SpringLearnApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 }  
}

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**JWT-Hands-on**

**Create authentication service that returns JWT**

**AuthenticationController.java**

package com.cognizant.springlearn.controller;  
  
import com.cognizant.springlearn.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
public class AuthenticationController {  
  
 @Autowired  
 private AuthenticationManager authenticationManager;  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @Autowired  
 private UserDetailsService userDetailsService;  
  
 @PostMapping("/authenticate")  
 public String createToken(@RequestBody AuthRequest authRequest) throws Exception {  
 try {  
 Authentication authentication = authenticationManager.authenticate(  
 new UsernamePasswordAuthenticationToken(authRequest.getUsername(), authRequest.getPassword())  
 );  
 } catch (AuthenticationException e) {  
 throw new Exception("Invalid username or password");  
 }  
  
 final UserDetails userDetails = userDetailsService.loadUserByUsername(authRequest.getUsername());  
 return jwtUtil.generateToken(userDetails.getUsername());  
 }  
  
 // DTO for authentication request  
 public static class AuthRequest {  
 private String username;  
 private String password;  
  
 public String getUsername() { return username; }  
 public void setUsername(String username) { this.username = username; }  
  
 public String getPassword() { return password; }  
 public void setPassword(String password) { this.password = password; }  
 }  
}

**SecurityConfig.java**

package com.cognizant.springlearn.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 UserDetails user = User.*withUsername*("user")  
 .password("{noop}password") // {noop} means no password encoder, just plain text  
 .roles("USER")  
 .build();  
 return new InMemoryUserDetailsManager(user);  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig) throws Exception {  
 return authConfig.getAuthenticationManager();  
 }  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http.csrf().disable()  
 .authorizeRequests()  
 .antMatchers("/authenticate", "/public").permitAll()  
 .anyRequest().authenticated()  
 .and()  
 .httpBasic();  
 return http.build();  
 }  
}

**Output**A black screen with many small squares

AI-generated content may be incorrect.