**SUPERSET ID - 6412891**

# Week 4

# Spring REST Hands-on Exercises (1–5)

## 1. Hello World RESTful Web Service

@RestController  
public class HelloController {  
  
 private static final Logger LOGGER = LoggerFactory.getLogger(HelloController.class);  
  
 @GetMapping("/hello")  
 public String sayHello() {  
 LOGGER.info("START: sayHello()");  
 String message = "Hello World!!";  
 LOGGER.info("END: sayHello()");  
 return message;  
 }  
}

## 2. REST - Country Web Service

@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 = context.getBean("country", Country.class);  
 LOGGER.info("END: getCountryIndia()");  
 return country;  
 }  
}

## 3. REST - Get Country Based on Country Code

@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);  
 }  
}  
  
@RestController  
public class CountryController {  
  
 @Autowired  
 private CountryService countryService;  
  
 @GetMapping("/countries/{code}")  
 public Country getCountry(@PathVariable String code) {  
 return countryService.getCountry(code);  
 }  
}

## 4. Create Authentication Service that Returns JWT (Controller)

@RestController  
public class AuthenticationController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(HttpServletRequest request) {  
 String authHeader = request.getHeader("Authorization");  
 if (authHeader != null && authHeader.startsWith("Basic ")) {  
 String base64Credentials = authHeader.substring("Basic ".length());  
 byte[] credDecoded = Base64.getDecoder().decode(base64Credentials);  
 String credentials = new String(credDecoded, StandardCharsets.UTF\_8);  
 String[] values = credentials.split(":", 2);  
 String username = values[0];  
 String token = jwtUtil.generateToken(username);  
 Map<String, String> response = new HashMap<>();  
 response.put("token", token);  
 return response;  
 }  
 throw new RuntimeException("Invalid Authorization Header");  
 }  
}

## 5. SecurityConfig Configuration

@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 UserDetails user = User.withUsername("user")  
 .password("{noop}pwd")  
 .roles("USER")  
 .build();  
 return new InMemoryUserDetailsManager(user);  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(authz -> authz  
 .requestMatchers("/authenticate").authenticated()  
 .anyRequest().permitAll())  
 .httpBasic(Customizer.withDefaults());  
 return http.build();  
 }  
}