Spring REST, JWT, and Security Hands-on Submission

# Hands-on 1: Create a Spring Web Project using Maven

Project: spring-learn  
Group: com.cognizant  
Dependencies: Spring Boot DevTools, Spring Web  
  
Steps:  
1. Go to https://start.spring.io/  
2. Set Group as "com.cognizant"  
3. Artifact Id as "spring-learn"  
4. Add Spring Boot DevTools, Spring Web  
5. Download and extract to Eclipse workspace  
6. Build:  
 mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456  
7. Import using Maven in Eclipse  
8. Verify main() execution using logs  
9. Run SpringLearnApplication  
  
SME Walkthrough:  
- src/main/java: Application code  
- src/main/resources: Configurations  
- src/test/java: Tests  
- SpringLearnApplication.java: Entry point  
- @SpringBootApplication: Config, AutoConfig, ComponentScan  
- pom.xml: Dependency management  
- Dependency Hierarchy: Check tree for transitive dependencies

# Hands-on 2: REST - Get Country Based on Country Code

Controller: com.cognizant.spring-learn.controller.CountryController  
  
@GetMapping("/countries/{code}")  
public Country getCountry(@PathVariable String code) {  
 return countryService.getCountry(code);  
}  
  
Service: com.cognizant.spring-learn.service.CountryService  
  
public Country getCountry(String code) {  
 List<Country> countryList = getCountriesFromXml();  
 return countryList.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElse(null);  
}  
  
Sample Request:  
http://localhost:8083/countries/in  
  
Sample Response:  
{  
 "code": "IN",  
 "name": "India"  
}

# Hands-on 3: REST - Country Web Service

URL: /country  
  
Controller: com.cognizant.spring-learn.controller.CountryController  
  
@RequestMapping("/country")  
public Country getCountryIndia() {  
 return context.getBean("in", Country.class);  
}  
  
Sample Request:  
http://localhost:8083/country  
  
Sample Response:  
{  
 "code": "IN",  
 "name": "India"  
}  
  
SME Points:  
- Controller maps GET request, returns bean as JSON.  
- Uses Jackson for JSON conversion.  
- Check headers in Network tab or Postman.

# Hands-on 4: Create Authentication Service that Returns JWT

Request:  
curl -s -u user:pwd http://localhost:8090/authenticate  
  
Response:  
{"token":"<jwt-token>"}  
  
Steps:  
1. Create AuthenticationController.  
2. Configure /authenticate in SecurityConfig.  
3. Read Authorization header, decode user:pwd.  
4. Validate user, generate JWT using utility.  
5. Return token in JSON response.

# Hands-on 5: Securing RESTful Web Services with Spring Security

Add to pom.xml:  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
</dependency>  
  
Steps:  
- Rebuild with mvn clean package with proxy.  
- Update Maven project in Eclipse.  
- Create package com.cognizant.spring-learn.security.  
- Create SecurityConfig extending WebSecurityConfigurerAdapter with @Configuration, @EnableWebSecurity.  
- Start app and test:  
 curl -s http://localhost:8090/countries  
  
Expected:  
{"status":401,"error":"Unauthorized"}  
  
Retrieve password from logs and execute:  
curl -s -u user:<password> http://localhost:8090/countries  
  
Expected:  
[{"code":"US","name":"United States"}, ... ]  
  
Points:  
- @EnableWebSecurity restricts REST APIs.  
- Uses Basic Auth, Authorization header, Base64 encoding of user:password.  
- Can be tested in Postman for headers and authorization.