WEEK 4 ASSIGNMENTS:

1.Create a Spring Web Project using Maven

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>
    <groupId>com.example</groupId>
    <artifactId>spring-rest-handson</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

2. Spring Core – Load Country from Spring Configuration XML

country.xml

```
<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
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="country" class="com.example.model.Country">
        <property name="code" value="IN"/>
        <property name="name" value="India"/>
    </bean>
</beans>
```

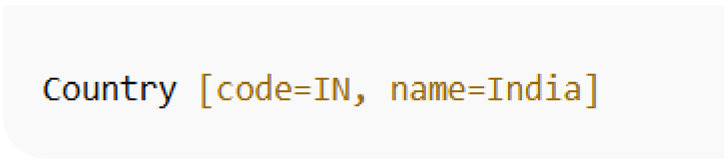
Country.java

```
package com.example.model;
public class Country {
    private String code;
    private String name;
 // Getters and Setters
    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 + "]";
    }
}
```

MainApp.java

```
package com.example;
import com.example.model.Country;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        Country country = (Country) context.getBean("country");
        System.out.println(country);
    }
}
```

OUTPUT:



```
Country [code=IN, name=India]
```

3.Hello World RESTful Web Service

HelloController.java

```
package com.example.controller;
import org.springframework.web.bind.annotation.*;
@RestController
public class HelloController {
  @GetMapping("/hello")
    public String sayHello() {
        return "Hello World";
    }
}
```

OUTPUT:

```
GET http://localhost:8080/hello
Response: Hello World
```

4. REST - Country Web Service

Country.java

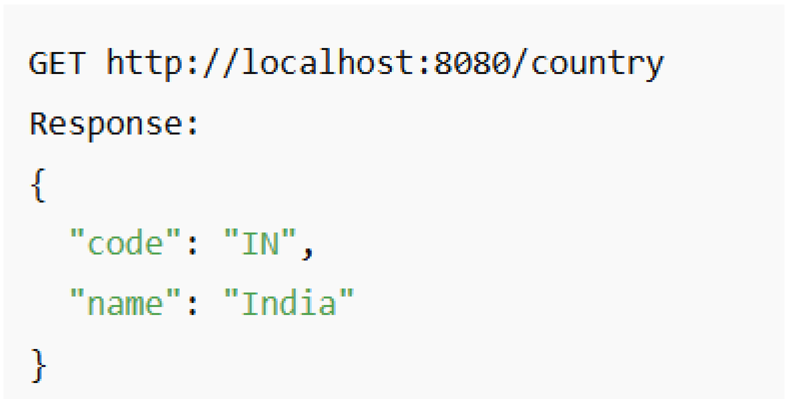
```
package com.example.model;
public class Country {
    private String code;
    private String name;

    // Constructor, Getters, Setters
    public Country(String code, String name) {
        this.code = code;
        this.name = name;
    }
    public String getCode() { return code; }
    public String getName() { return name; }
}
```

CountryController.java

```
package com.example.controller;
import com.example.model.Country;
import org.springframework.web.bind.annotation.*;
@RestController
public class CountryController {
@GetMapping("/country")
    public Country getCountry() {
        return new Country("IN", "India");
    }
```

}

OUTPUT:



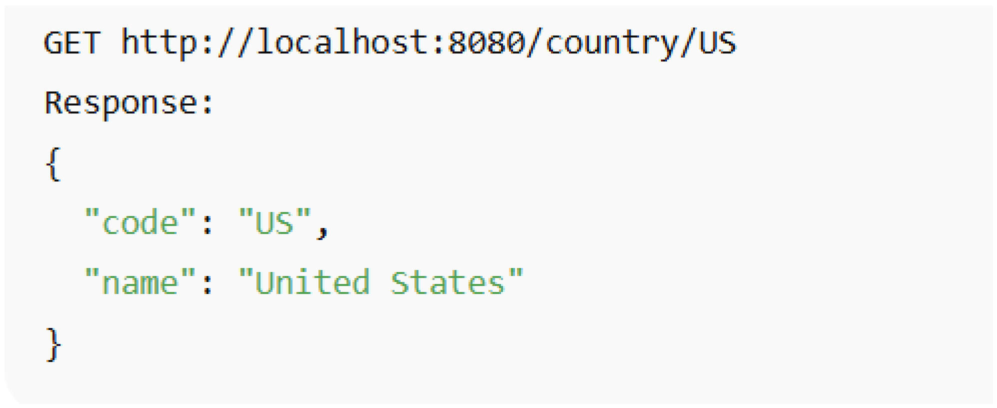
```
GET http://localhost:8080/country
Response:
{
  "code": "IN",
  "name": "India"
}
```

5.REST - Get Country Based on Country Code

CountryController.java

```
@GetMapping("/country/{code}")
public Country getCountryByCode(@PathVariable String code) {
   if (code.equalsIgnoreCase("IN")) {
      return new Country("IN", "India");
   } else if (code.equalsIgnoreCase("US")) {
      return new Country("US", "United States");
   } else {
      return new Country("NA", "Not Available");
   }
}
```

OUTPUT:



```
GET http://localhost:8080/country/US
Response:
{
  "code": "US",
  "name": "United States"
}
```

6. JWT Authentication – Already Provided Earlier

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>
    <groupId>com.example</groupId>
    <artifactId>jwt-handson</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

JwtHandsonApplication.java

```
package com.example;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class JwtHandsonApplication {
    public static void main(String[] args) {
        SpringApplication.run(JwtHandsonApplication.class, args);
    }
}
```

AuthRequest.java

```
package com.example.model;
public 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; }
}
```

AuthResponse.java

```
package com.example.model;
public class AuthResponse {
    private String jwtToken;

    public AuthResponse(String jwtToken) {
        this.jwtToken = jwtToken;
    }

    public String getJwtToken() {
        return jwtToken;
    }
}
```

JWT Utility

JwtUtil.java

```
package com.example.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 secret = "shuruthika-secret-key";
    public String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60)) // 1 hour
                .signWith(SignatureAlgorithm.HS256, secret)
                .compact();
    }
}
```

Controller

AuthController.java

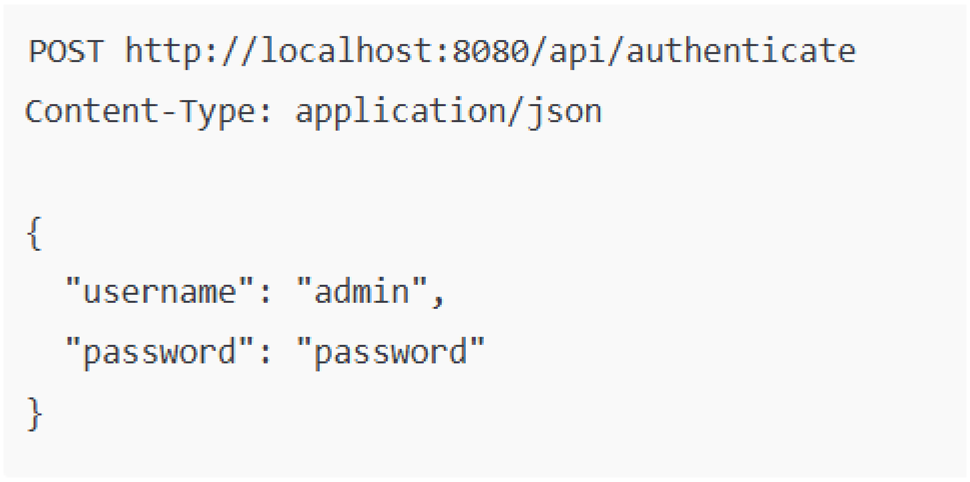
```
package com.example.controller;
import com.example.model.AuthRequest;
import com.example.model.AuthResponse;
import com.example.util.JwtUtil;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api")
public class AuthController {

    private final JwtUtil jwtUtil;

    public AuthController(JwtUtil jwtUtil) {
        this.jwtUtil = jwtUtil;
    }

    @PostMapping("/authenticate")
    public AuthResponse generateToken(@RequestBody AuthRequest request) {
        if ("admin".equals(request.getUsername()) &&
"password".equals(request.getPassword())) {
            String token = jwtUtil.generateToken(request.getUsername());
            return new AuthResponse(token);
        } else {
            throw new RuntimeException("Invalid Credentials");
        }
    }
}
```

Security Config

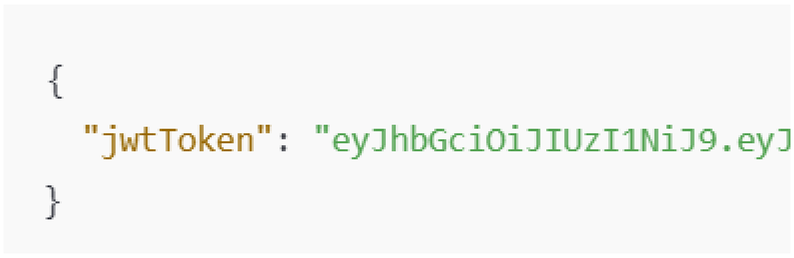
SecurityConfig.java

```
package com.example.config;
import org.springframework.context.annotation.Bean;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.web.SecurityFilterChain;

import org.springframework.context.annotation.Configuration;

@Configuration
public class SecurityConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth.anyRequest().permitAll());
        return http.build();
    }
}
```

OUTPUT:



```
POST http://localhost:8080/api/authenticate
Content-Type: application/json

{
  "username": "admin",
  "password": "password"
}
```



```
{
  "jwtToken": "eyJhbGciOiJIUzI1NiJ9.eyJ
}
```