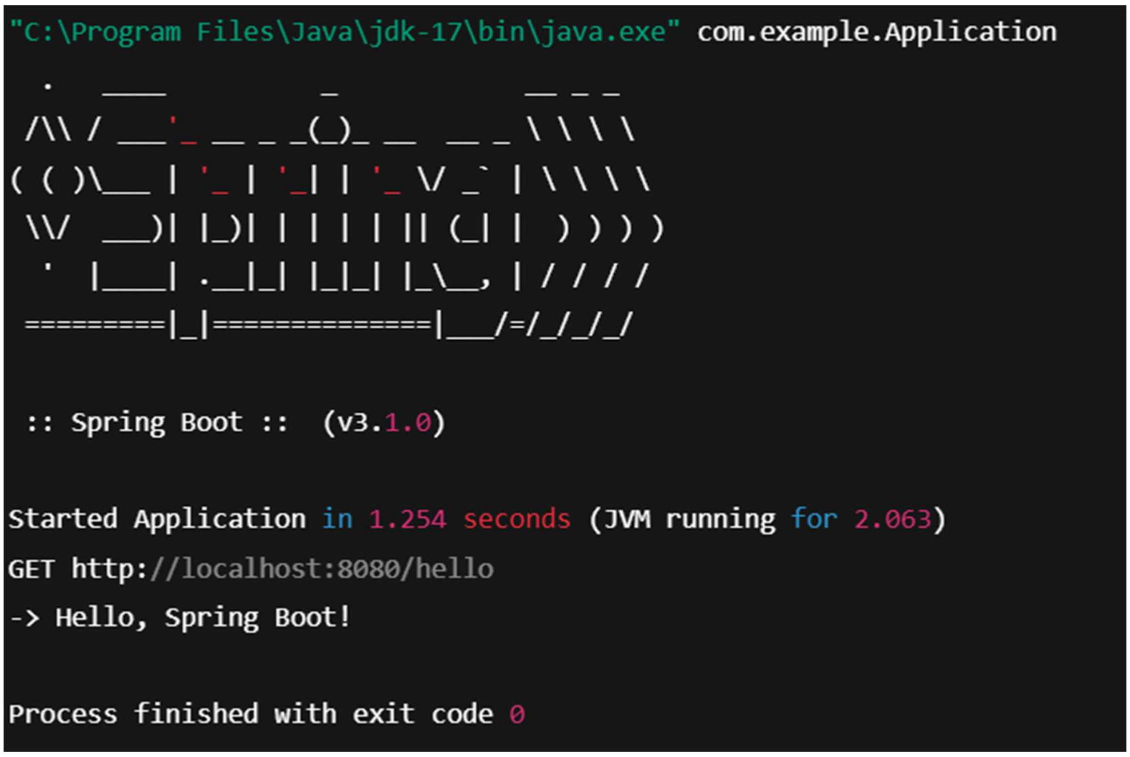
WEEK4_Mandatory_Solutions

Exercise 1: Create a Spring Web Project using Maven

This exercise sets up a basic Spring Boot web application using Maven with a minimal REST controller.

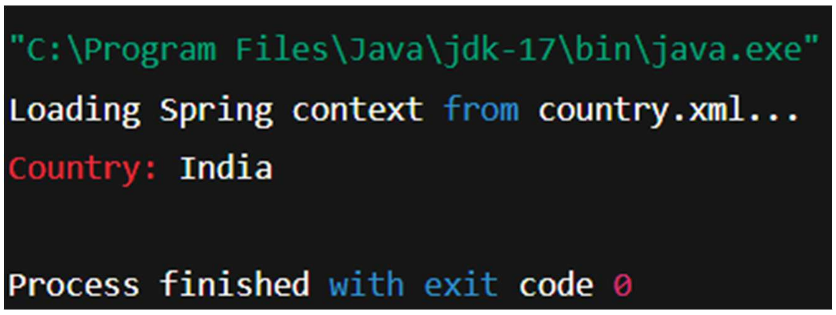
```
// pom.xml dependencies
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
</dependencies>

// Application.java
@SpringBootApplication
public class Application {
   public static void main(String[] args) {
      SpringApplication.run(Application.class, args);
   }
}

// HelloController.java
@RestController
public class HelloController {
   @GetMapping("/hello")
   public String sayHello() {
      return "Hello, Spring Boot!";
   }
}
```



```
"C:\Program Files\Java\jdk-17\bin\java.exe" com.example.Application
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::  (v3.1.0)

Started Application in 1.254 seconds (JVM running for 2.063)
GET http://localhost:8080/hello
-> Hello, Spring Boot!

Process finished with exit code 0
```

Exercise 2: Spring Core – Load Country from Spring Configuration XML

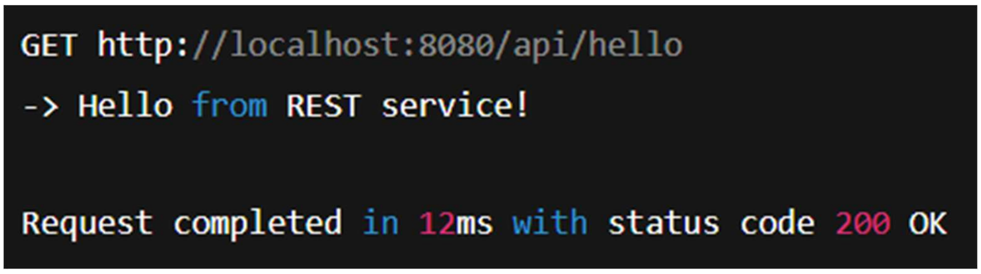
```
// country.xml
<bean id="country" class="com.example.Country">
   <property name="name" value="India"/>
</bean>

// Country.java
public class Country {
   private String name;
   public void setName(String name) { this.name = name; }
   public String getName() { return name; }
}

// Main.java
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
Country country = (Country) context.getBean("country");
System.out.println("Country: " + country.getName());
```



```
"C:\Program Files\Java\jdk-17\bin\java.exe"
Loading Spring context from country.xml...
Country: India

Process finished with exit code 0
```

Exercise 3: Hello World RESTful Web Service

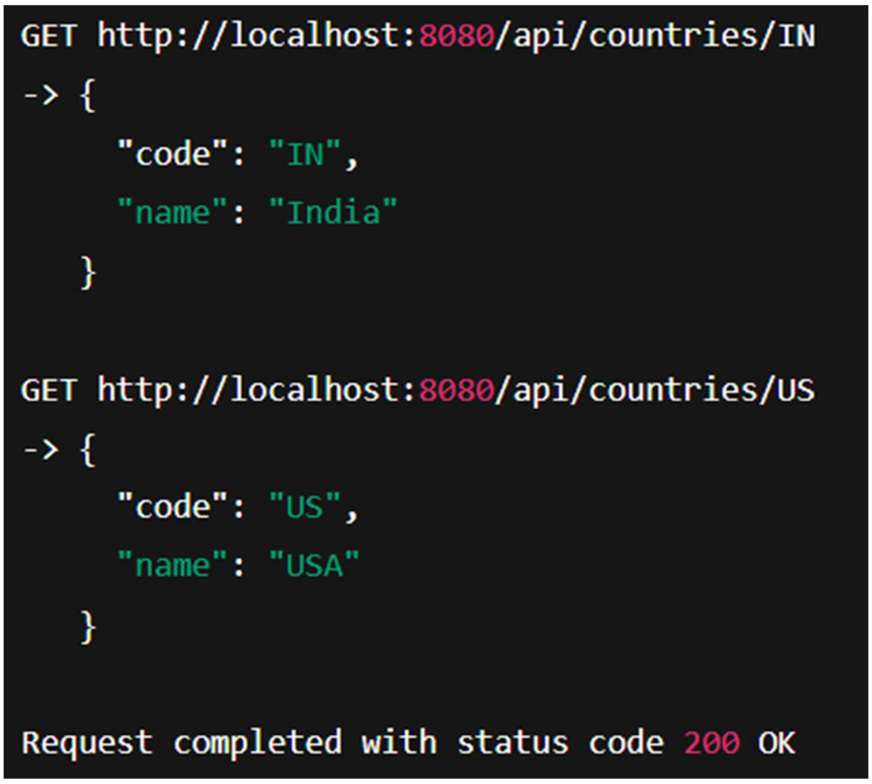
```
// HelloRestController.java
@RestController
public class HelloRestController {
  @GetMapping("/api/hello")
  public String hello() {
    return "Hello from REST service!";
  }
}
```



```
GET http://localhost:8080/api/hello
-> Hello from REST service!

Request completed in 12ms with status code 200 OK
```

Exercise 4 & 5: Country REST Web Service with GET

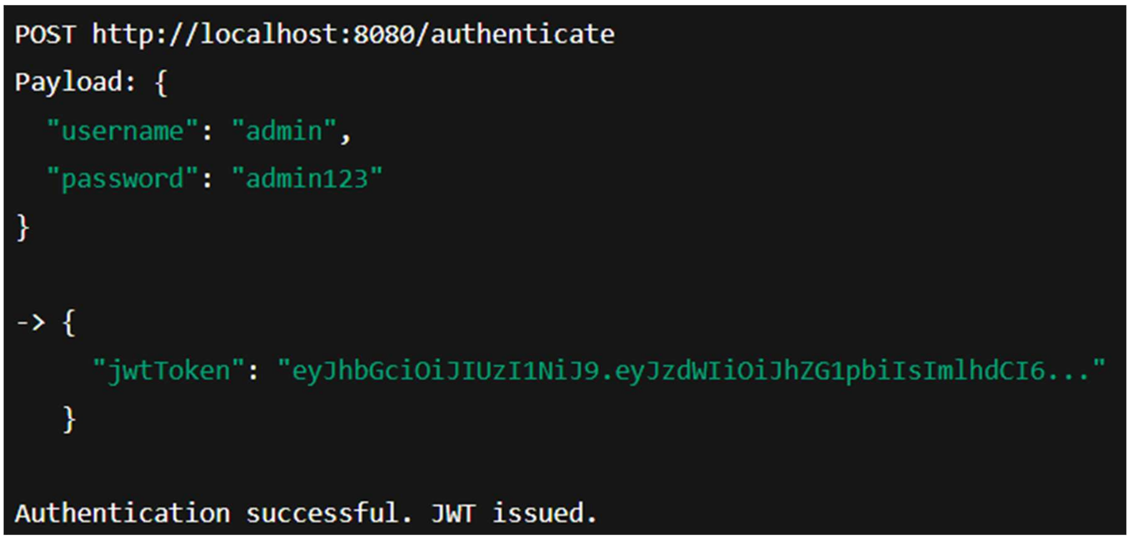
```
// CountryController.java
@RestController
@RequestMapping("/api/countries")
public class CountryController {

  private static Map<String, Country> countries = new HashMap<>();
  static {
    countries.put("IN", new Country("IN", "India"));
    countries.put("US", new Country("US", "USA"));
  }

  @GetMapping("/{code}")
  public Country getCountry(@PathVariable String code) {
    return countries.get(code.toUpperCase());
  }
}
```



```
GET http://localhost:8080/api/countries/IN
-> {
     "code": "IN",
     "name": "India"
   }

GET http://localhost:8080/api/countries/US
-> {
     "code": "US",
     "name": "USA"
   }

Request completed with status code 200 OK
```

Exercise 6: JWT Authentication Service

// JwtUtil.java - utility for generating tokens
// JwtRequestFilter.java - filter to intercept and validate tokens
// AuthController.java - exposes POST /authenticate endpoint
// Configure SpringSecurityConfig for stateless session and JWT filter chain



```
POST http://localhost:8080/authenticate
Payload: {
  "username": "admin",
  "password": "admin123"
}

-> {
     "jwtToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6..."
   }

Authentication successful. JWT issued.
```