

## WEEK 4:

### HandsON 1: Create a Spring Web Project using Maven

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "spring-learn". It includes packages like "com.cognizant.spring.learn" containing "SpringLearnApplication.java", and resources like "src/main/java", "src/test/java", and "pom.xml".
- Code Editor:** Displays the code for "SpringLearnApplication.java".

```
1 package com.cognizant.spring_learn;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.boot.SpringApplication;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7
8 @SpringBootApplication
9 public class SpringLearnApplication {
10
11     // Logger instance
12     private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);
13
14     public static void main(String[] args) {
15         LOGGER.info("STARTING SPRINGLEARN APPLICATION");
16         SpringApplication.run(SpringLearnApplication.class, args);
17         LOGGER.info("APPLICATION STARTED");
18     }
19 }
```
- Console:** Shows the output of running the application.

```
Starting SpringLearnApplication using Java 21.0.5 with PID 21944 (:C:\Users\DELL\eclipse-workspace\spring-learn\target\classes)
No active profile set, falling back to 1 default profile: "default"
Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
Tomcat initialized with port 8080 (http)
Starting service [Tomcat]
Starting Service engine: [Apache Tomcat/10.1.42]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 1522 ms
LiveReload server is running on port 35729
Tomcat started on port 8080 (http) with context path '/'
Started SpringLearnApplication in 2.913 seconds (process running for 3.537)
APPLICATION STARTED
```
- Outline:** Shows the class structure with "SpringLearnApplication" and its methods.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "spring-learn". It includes packages like "com.cognizant.spring.learn" containing "SpringLearnApplication.java", and resources like "src/main/java", "src/test/java", and "pom.xml".
- Code Editor:** Displays the code for "SpringLearnApplication.java".

```
1 package com.cognizant.spring_learn;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.boot.SpringApplication;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7
8 @SpringBootApplication
9 public class SpringLearnApplication {
10
11     // Logger instance
12     private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);
13
14     public static void main(String[] args) {
15         LOGGER.info("STARTING SPRINGLEARN APPLICATION");
16         SpringApplication.run(SpringLearnApplication.class, args);
17         LOGGER.info("APPLICATION STARTED");
18     }
19 }
```
- Console:** Shows the output of running the application, including the Spring Boot logo.

```
14:54:32.812 [main] INFO com.cognizant.spring_learn.SpringLearnApplication -- STARTING SPRINGLEARN APPLICATION
14:54:33.070 [restartedMain] INFO com.cognizant.spring_learn.SpringLearnApplication -- STARTING SPRINGLEARN APPLICATION
:: Spring Boot ::      (v3.5.3)

2025-07-10T14:54:33.789+05:30  INFO 21944 --- [spring-learn] [  restartedMain] c.c.cognizant.spring_learn.SpringLearnApplication : Started SpringLearnApplication in 0.047 seconds (process running for 0:00:01)
2025-07-10T14:54:33.791+05:30  INFO 21944 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-10T14:54:33.801+05:30  INFO 21944 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-10T14:54:33.805+05:30  INFO 21944 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet Engine: [Tomcat]
2025-07-10T14:54:33.824+05:30  INFO 21944 --- [spring-learn] [  restartedMain] o.a.e.c.c.[Tomcat].INITIALIZATION : Initiation of initialization process for Tomcat
```
- Outline:** Shows the class structure with "SpringLearnApplication" and its methods.

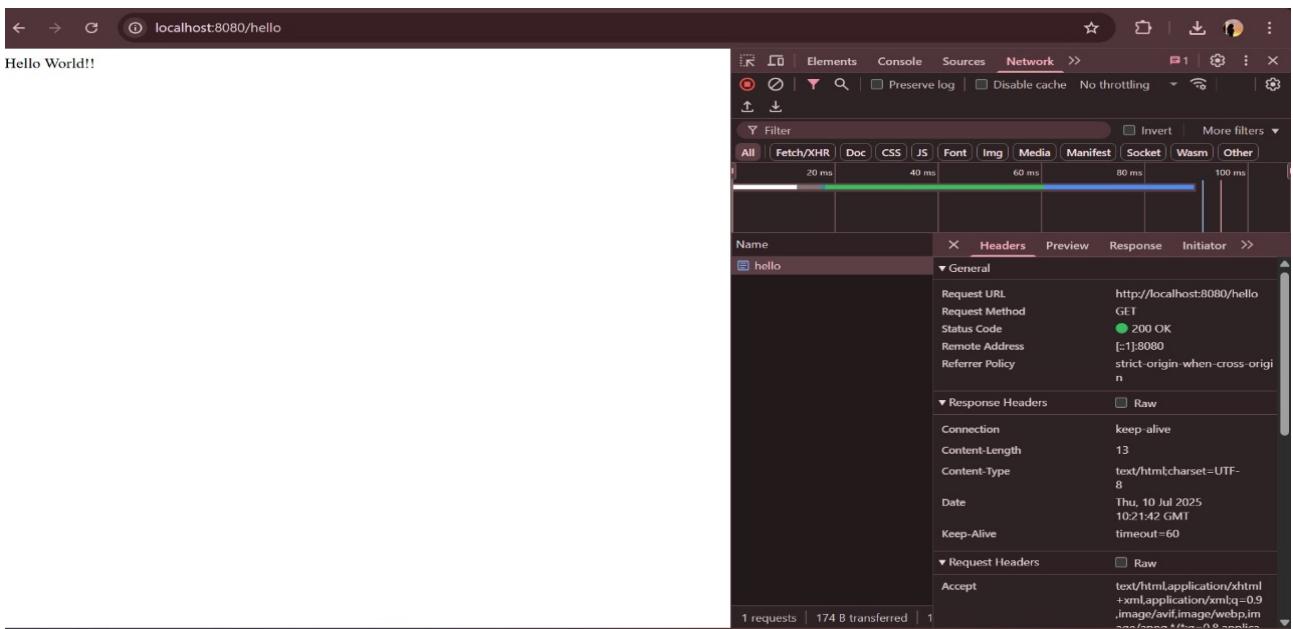
### HandsON 2: Spring Core – Load Country from Spring Configuration XML

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java projects like employeeapp, junitdemo, LibraryDIPProject, LibraryManagement, loggingdemo, mockito, Mockitoo, ormlearn, orm-learn, pgm1\_maven, pgm2\_maven, pgm4\_maven, prg2, prgm4-example-jar, pro2-example-jar, prog1, prog1-example-jar, and spring-learn.
- Code Editor:** Displays the `SpringLearnApplication.java` file with code related to Spring and logging.
- Terminal:** Shows the command-line output of running the application, indicating it loaded a bean named 'country' from a classpath XML context.

## HandsOn 3: Hello World RESTful Web Service

Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework.



## HandsOn 4: REST - Country Web Service

Pretty-print □

```
{"code":"IN","name":"India"}
```

Network □

Filter □ Invert More filters □

All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other

Name Headers Preview Response Initiator >

country General

Request URL http://localhost:8080/country  
Request Method GET  
Status Code 200 OK  
Remote Address [::1]:8080  
Referer Policy strict-origin-when-cross-origin

Response Headers □ Raw

Connection keep-alive  
Content-Type application/json  
Date Thu, 10 Jul 2025 10:19:26 GMT  
Keep-Alive timeout=60  
Transfer-Encoding chunked

Request Headers □ Raw

Accept text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Accept-Encoding gzip, deflate, br, zstd  
Accept-Language en-US,en;q=0.9  
Cache-Control max-age=0  
Connection keep-alive  
Cookie jenkins-timestamper-rr-1000000000000000000

1 requests | 201 B transferred | 2

## HandsOn 5: REST - Get country based on country code

Pretty-print □

```
{"code":"IN","name":"India"}
```

Network □

Filter □ Invert More filters □

All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other

Name Headers Preview Response Initiator >

country General

Request URL http://localhost:8080/country  
Request Method GET  
Status Code 200 OK  
Remote Address [::1]:8080  
Referer Policy strict-origin-when-cross-origin

Response Headers □ Raw

Connection keep-alive  
Content-Type application/json  
Date Thu, 10 Jul 2025 10:19:26 GMT  
Keep-Alive timeout=60  
Transfer-Encoding chunked

Request Headers □ Raw

Accept text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Accept-Encoding gzip, deflate, br, zstd  
Accept-Language en-US,en;q=0.9  
Cache-Control max-age=0  
Connection keep-alive  
Cookie jenkins-timestamper-rr-1000000000000000000

1 requests | 201 B transferred | 2

The screenshot shows the Eclipse IDE interface with the following details:

- Project Structure:** The Package Explorer view shows a project named "spring-learn". It contains several Java packages: com.cognizant.springlearn, com.cognizant.springlearn.controller, com.cognizant.springlearn.model, com.cognizant.springlearn.service, and com.cognizant.springlearn.util. There are also resources like static, templates, application.properties, and country.xml.
- Logs:** The Console tab displays the output of the Tomcat server. It shows the initialization of Tomcat, the starting of the Spring DispatcherServlet, and the execution of controller methods like sayHello().
- Outline:** The Outline view is empty, indicating no active editor.

```

SpringLearnApplication [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Jul-2025, 3:46:18pm elapsed: 0:06:15) [pid: 12492]
.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
>apache.catalina.core.StandardService : Starting service [tomcat]
>apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1706 ms
.b.d.a.OptionalaliveReloadServer : LiveReload server is running on port 35729
.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
.springframeworklearn.SpringApplication : Started SpringApplication in 3.193 seconds (process running for 3.81)
.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
: Completed initialization in 0 ms
.s.controller.CountryController : START: getCountryIndia()
.s.controller.CountryController : END: getCountryIndia()
.s.controller.CountryController : START: getCountryIndia()
.s.controller.CountryController : END: getCountryIndia()
.s.controller.HelloController : START: sayHello()
.s.controller.HelloController : END: sayHello()
.s.controller.HelloController : START: sayHello()
.s.controller.HelloController : END: sayHello()

```

## HandsON 6: Create authentication service that returns JWT

The screenshot shows the Eclipse IDE interface with the following details:

- Project Structure:** The Package Explorer view shows the same "spring-learn" project structure as the previous screenshot.
- Code:** The SecurityConfig.java file is open in the Editor. It defines a filterChain that enables basic authentication and returns a JWT token.
- Logs:** The Console tab shows the command-line output of a curl request to the application, which returns a JSON response containing a JWT token.
- Outline:** The Outline view shows the SecurityConfig class and its filterChain method.

```

import org.springframework.security.web.SecurityFilterChain;
public class SecurityConfig {
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf().disable()
            .authorizeRequests()
                .antMatchers("/authenticate").permitAll() // Allow access without auth
                .anyRequest().authenticated()
            .and()
            .httpBasic(); // enable basic auth
        return http.build();
    }
}

HTTP/1.1 401
Basic authentication problem, ignoring.
< WWW-Authenticate: Basic realm="Realm"
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: DENY
< Set-Cookie: JSESSIONID=C316EC02739175D2D64AA667E4FCF5C5; Path=/; HttpOnly
Basic authentication problem, ignoring.
< WWW-Authenticate: Basic realm="Realm"
< Content-Length: 0
< Date: Thu, 10 Jul 2025 10:55:42 GMT
<
{"token": "eyJhbGciOiJIUzI1NiJ9.eyJdWIiOj1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRv1CV-hwKfoqZYlaVQqEUiBlowcWn0ft3tgv0dL0"}}

```