Practice Work 1

Spring Boot Project Setup and Hello World

Web Component Development (Java EE)
Week 1

Textbook: Pro Spring Boot 3 with Kotlin, 3rd Edition, Chapter 1

Institution: IITU

Department: Information Systems

Total Points: 100

Developed by Syrym Zhakypbekov IITU

2026

Contents

Objective

Objective

Get familiar with Spring Boot by creating your first application, understanding project structure, and running a simple "Hello World" web application.

Background

Spring Boot simplifies Java application development by providing auto-configuration and convention-over-configuration approach. This practice work introduces you to Spring Initializr, project structure, and basic Spring Boot concepts.

Vocabulary

Vocabulary

- **Spring Boot:** Framework for building production-ready applications
- **Spring Initializr:** Web-based tool for generating Spring Boot projects
- **@SpringBootApplication:** Main annotation that enables auto-configuration
- **Embedded Server:** Web server (Tomcat) included in the application
- **Auto-configuration:** Automatic setup of Spring components based on dependencies
- **Starter:** Pre-configured dependency sets for common use cases

Tasks

Task 1: Create Project Using Spring Initializr (30 Points)

Task

Steps:

1. Go to https://start.spring.io
2. Configure project:
   - Project: Gradle - Groovy (or Maven if you prefer)
   - Language: Java
   - Spring Boot: 3.2.x or latest stable version
   - Project Metadata:
     - Group: `com.iitu.student`
     - Artifact: `hello-spring-boot`
     - Name: `hello-spring-boot`
     - Package name: `com.iitu.student.hellospringboot`
     - Packaging: JAR
     - Java: 17 or 21
3. Add Dependencies:
   - Click "Add Dependencies"
   - Search and add: "Spring Web"
   - This adds `spring-boot-starter-web` dependency
4. Generate and Download:
   - Click "Generate" button
   - Download the ZIP file
   - Extract to your workspace
5. Import to IDE:
   - Open IntelliJ IDEA (or your preferred IDE)
   - File $\rightarrow$ Open $\rightarrow$ Select extracted folder
   - Wait for Gradle/Maven sync to complete

Deliverable

- Screenshot of Spring Initializr with your configuration
- Brief description of what Spring Initializr generated

Task 2: Explore Project Structure (20 Points)

Task

Steps:

1. Examine the generated project structure:

```
hello-spring-boot/
  src/
    main/
      java/
        com/iitu/student/hellospringboot/
          HelloSpringBootApplication.java
      resources/
        application.properties
    test/
      java/
        com/iitu/student/hellospringboot/
          HelloSpringBootApplicationTests.java
  build.gradle (or pom.xml for Maven)
  README.md
```

2. Open and examine files:
   - `HelloSpringBootApplication.java` - Main application class
   - `application.properties` - Configuration file
   - `build.gradle` (or `pom.xml`) - Build configuration
3. Answer these questions:
   a) What is the purpose of `@SpringBootApplication` annotation?
   b) What does the `main()` method do?
   c) What is `application.properties` used for?
   d) What dependencies are included in `build.gradle/pom.xml`?

Deliverable

- Answers to the four questions (2–3 sentences each)
- Screenshot of your project structure in IDE

Task 3: Create Hello World Controller (30 Points)

Task

Steps:

1. Create a new Java class: `HelloController.java`
   Location: `src/main/java/com/iitu/student/hellospringboot/`
2. Add the following code:

```java
package com.iitu.student.hellospringboot;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/")
    public String hello() {
        return "Hello, Spring Boot!";
    }

    @GetMapping("/greeting")
    public String greeting() {
        return "Welcome to Web Component Development!";
    }
}
```

3. Understand the annotations:
   - `@RestController`: Marks class as a REST controller
   - `@GetMapping`: Maps HTTP GET requests to methods
   - `"/"` and `"/greeting"`: URL paths

Deliverable

- `HelloController.java` source code
- Brief explanation of what `@RestController` and `@GetMapping` do

Task 4: Run and Test Application (20 Points)

Task

Steps:

1. Run the application:
   - In IntelliJ: Right-click `HelloSpringBootApplication.java` → Run
   - Or use command line: `./gradlew bootRun` (Gradle) or `mvn spring-boot:run` (Maven)
2. Observe the console output:
   - Look for "Started HelloSpringBootApplication"
   - Note the port number (usually 8080)
   - Look for embedded Tomcat server startup messages
3. Test the endpoints:
   - Open browser: `http://localhost:8080/`
   - Should see: "Hello, Spring Boot!"
   - Open: `http://localhost:8080/greeting`
   - Should see: "Welcome to Web Component Development!"
4. Alternative: Use curl command:

```
curl http://localhost:8080/
curl http://localhost:8080/greeting
```

Deliverable

- Screenshot of application running (console output)
- Screenshot of browser showing "Hello, Spring Boot!"
- Screenshot of browser showing greeting message

Submission Instructions

1. Create folder: `Practice_01/`
   - Include all source files
   - Include screenshots
   - Include answers to questions
2. Initialize Git repository:

```
git init
git add .
git commit -m "feat: add Practice 1 - Spring Boot Hello World"
```

3. Create a text file with GitHub repository link (if using GitHub)
4. Upload to learning management system

Grading Rubric

Task	Points
Task 1: Project creation	30
Task 2: Project structure exploration	20
Task 3: Hello World controller	30
Task 4: Running and testing	20
Total	**100**

Additional Notes

- Make sure Java 17 or 21 is installed and configured
- If port 8080 is busy, change it in `application.properties`:

```
server.port=8081
```

- Read error messages carefully - they usually tell you what's wrong
- Explore the auto-generated test file to understand testing structure