Spring Boot Coding Assignment Week 15

1. **Spring vs. Spring Boot:**

Spring is a robust and extensively adopted Java framework, furnishing extensive infrastructure to bolster the creation of Java applications. Its prowess lies in streamlining the intricate process of building enterprise-level applications, achieved through an array of specialized modules catering to diverse needs, including data retrieval, security fortification, web application development, and beyond. At its core, Spring encompasses a rich assortment of capabilities encompassing dependency injection, aspect-oriented programming, data connectivity, and the orchestration of transactions.

Spring Boot, on the other hand, is an extension of the Spring framework that focuses on simplifying the process of building production-ready applications. Spring Boot provides a set of conventions and defaults, along with various pre-configured templates, to enable developers to quickly set up and deploy Spring-based applications without the need for extensive configuration. It eliminates much of the boilerplate code and configuration that developers typically have to write when setting up a Spring application. Spring Boot embraces the concept of "convention over configuration" and includes embedded web servers, auto-configuration, and various tools for packaging and deployment.

Think of Spring Boot as the cooler cousin of the Spring framework. They're related, with Spring Boot building upon the solid foundation of Spring. Spring Boot takes all the good stuff from Spring but adds its own style to make setting up, configuring, and developing applications even easier. It's like Spring's upgraded version, designed to make life simpler for developers and let them create powerful Spring applications without the usual hassle.

Source:
[Spring Framework Documentation](https://spring.io/projects/spring-framework), [Spring Boot Documentation](https://spring.io/projects/spring-boot)

2. Annotations for CRUD operations in Spring Boot:

In Spring Boot applications, you can use various annotations to implement CRUD (Create, Read, Update, Delete) operations on data. Some commonly used annotations for CRUD operations include:

- `@Entity`: This annotation is used to mark a Java class as a JPA entity. It's typically used to represent a table in a relational database. Entities are used for the 'C' and 'R' operations in CRUD.

- `@Repository`: This annotation is used to indicate that a class provides the mechanism for storage, retrieval, and search operations on objects. It's commonly used for 'R' operations (reading data).

- `@Service`: This annotation marks a class as a Spring service bean. It's used to define business logic and is commonly used for 'C', 'U', and 'D' operations.

- `@RestController`: This annotation combines `@Controller` and `@ResponseBody`. It's used to define a controller class that handles HTTP requests and returns data directly as the response, often used for exposing CRUD operations via RESTful APIs.

- `@RequestMapping`: This annotation maps HTTP requests to handler methods in a controller. It's used to define the endpoints for CRUD operations.

These annotations, among others, play a crucial role in defining the structure and behavior of your Spring Boot application when it comes to CRUD operations.

Source: [Spring Framework Documentation](https://spring.io/projects/spring-framework), [Spring Boot Documentation](https://spring.io/projects/spring-boot)