


Spring Boot Annotations

@SpringBootApplication:

- combines **@Configuration**, **@EnableAutoConfiguration**, and **@ComponentScan**
- marks the main class of a Spring Boot Application

@Controller: defines a class as a Spring MVC controller for handling HTTP web requests

@RestController: combines **@Controller** and **@ResponseBody** to simplify RESTful API development

@RequestMapping: maps HTTP requests to controller methods

@GetMapping: method handles HTTP GET requests and maps them to a specific controller method

@PostMapping: method handles HTTP POST requests and maps them to a controller method

@PutMapping: method handles HTTP PUT requests, typically used for updating resources

@DeleteMapping: method handles HTTP DELETE requests to remove resources or data

@Autowired: injects dependencies into a Spring bean

@Service: marks a class as a service, typically used for business logic

@Repository: marks a class as a repository, typically for database access

@Entity: marks a class as a JPA entity, corresponding to a database table

@Table: specifies the details of the database table associated with the entity

@Column: customizes the mapping of a field or property to a database column in an entity

@Id: marks a field or property as the primary key of the entity

@GeneratedValue: specifies the strategy for generating primary key values for an entity

@Component: marks a class as a Spring component, signifying that it is eligible for automatic detection and instantiation as a Spring bean

@Configuration: marks a class as a source of bean configuration in an application

@PropertySource: loads external configuration properties for an application

@Value: injects values from properties or configuration into Spring beans in an application

@Qualifier: helps specify which bean to inject when there are multiple beans of the same type

@Profile: controls when a bean or configuration is active in an application based on specified profiles

@SpringBootTest: marks a test class as a Spring Boot integration test

@EnableWebSecurity: used to enable and configure Spring Security for your web application

@EnableWebMvc: used to enable Spring Web MVC, which is the framework for building web applications using the Model-View-Controller pattern

* A Spring bean is a component of the Spring application. Think of a Spring bean as a building block of the whole Spring application.

** Configuration means specifying settings and options to determine how the application behaves.

*** By “application” I am referring to a Spring Boot application