Software Engineering Tips

Complete Spring & Spring Boot Annotations

Tips

Dependency Injection & Bean Management

@Autowired

- **Purpose:** Automatically injects a bean by type.
- **Use case:** When you want Spring to resolve and inject a bean into a class without explicitly specifying it.

@Qualifier

- **Purpose:** Specifies which bean to inject when there are multiple beans of the same type.
- **Use case:** Use it along with @Autowired to resolve conflicts between similar beans.

@Value

- **Purpose:** Injects a value from application properties or environment variables.
- **Use case:** Useful for injecting constants, configuration values, or strings.

@Primary

- **Purpose**: Marks one bean as the default when multiple candidates are available for injection.
- **Use case:** Avoids the need for @Qualifier by specifying a preferred bean.

Tips

Data Access & JPA

@Entity

- **Purpose:** Marks a class as a JPA entity.
- **Use case:** Tells the JPA provider (e.g., Hibernate) that this class should be mapped to a database table.

@Table

- **Purpose:** Configures the table name and schema.
- **Use case:** Used with @Entity to specify the table details.

@Column

- **Purpose:** Maps a field to a specific column in the table.
- **Use case: A**llows setting column name, length, nullable, unique, etc.

@Id

- **Purpose:** Marks the field as the primary key.
- **Use case:** Required for all JPA entities to uniquely identify records.

@GeneratedValue

- **Purpose:** Specifies the strategy for primary key generation.
- **Use case:** Common strategies include AUTO, IDENTITY, SEQUENCE, TABLE.

@Repository

- **Purpose:** Marks a class as a DAO (Data Access Object).
- **Use case:** Enables Spring's automatic exception translation for persistence-related exceptions.

@Transactional

- **Purpose:** Manages transaction boundaries.
- **Use case:** Ensures operations are wrapped in a database transaction, with rollback support on failure.

Tips

Spring Security Annotations

@EnableWebSecurity

- **Purpose:** Enables Spring Security's web security features.
- **Use case:** Added to a @Configuration class to activate Spring Security's filter chain.

Method-Level Authorization

@PreAuthorize

- **Purpose:** Authorizes access before a method is invoked, using SpEL (Spring Expression Language).
- **Use case:** @PreAuthorize("hasRole('ADMIN')") ensures only users with the ADMIN role can execute the method.

@Secured

- **Purpose:** Restricts method access based on roles (similar to @PreAuthorize, but simpler).
- **Use case:** @Secured("ROLE_USER") restricts access to users with the USER role.

@WithMockUser

- **Purpose:** Used in unit or integration tests to simulate an authenticated user.
- **Use case:** @WithMockUser(username="admin", roles={"ADMIN"}) allows testing secured methods without real authentication.

Tips

Spring Web & REST Annotations

@RequestMapping

- **Purpose:** Maps HTTP requests to handler methods (for any HTTP method).
- **Use case:** General-purpose mapping for GET, POST, etc. You can specify method, path, headers, etc.

@GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping

- **Purpose:** Shorthand annotations for @RequestMapping(method = ...).
- **Use case:** Used to handle specific HTTP methods:
    - @GetMapping: for GET requests
    - @PostMapping: for POST requests
    - @PutMapping: for PUT requests
    - @DeleteMapping: for DELETE requests
    - @PatchMapping: for PATCH requests

Tips

Request/Response Data Binding

@RequestBody

- **Purpose:** Binds the HTTP request body (usually JSON) to a Java object.
- **Use case:** Used in POST/PUT methods to deserialize incoming data.

@ResponseBody

- **Purpose:** Binds the return value of a method to the HTTP response body.
- **Use case:** Converts the return value (usually an object) to JSON or XML.

Extracting Request Data

@PathVariable

- **Purpose:** Extracts values from the URI path.
- **Use case:** For example, /users/{id} → @PathVariable("id") Long id

@RequestParam

- **Purpose**: Extracts query parameter values from the URL.
- **Use case:** /search?query=java → @RequestParam("query") String query

@RequestHeader

- **Purpose:** Accesses HTTP header values.
- **Use case:** @RequestHeader("Authorization") String token

Tips

Core Spring Boot Annotations

@SpringBootApplication

- **Purpose:** Main entry point of a Spring Boot application.
- **Combines:**
  - @Configuration: Marks the class as a source of bean definitions.
  - @EnableAutoConfiguration: Enables auto-configuration based on the classpath.
  - @ComponentScan: Scans for components (e.g., @Component, @Service, @Controller, etc.) in the package and subpackages.
- **Use case:** Applied to the main class to bootstrap the application.

@EnableAutoConfiguration

- **Purpose:** Enables Spring Boot to automatically configure beans based on classpath dependencies.
- **Use case:** Spring sets up defaults (e.g., embedded Tomcat, DataSource, etc.) so you don't have to configure them manually.

@ComponentScan

- **Purpose:** Scans the specified package(s) for Spring-managed components.
- **Use case:** Automatically discovers beans like controllers and services.

Tips

@Configuration

- **Purpose:** Indicates that the class can be used by Spring IoC container as a source of bean definitions.
- **Use case:** Typically used in Java config classes to define beans.

@Bean

- **Purpose:** Declares a bean that is managed by the Spring container.
- **Use case:** Used inside @Configuration classes to define beans manually.

Component Stereotypes in Spring

@Component

- **Purpose:** A generic stereotype for any Spring-managed component.
- **Use case:** Used when a class doesn't fall into a more specific layer (like service, repository, or controller).

@Service

- **Purpose:** A specialization of @Component, used for business logic or service-layer classes.
- **Use case:** Marks classes that contain service operations, often called from controllers.

Tips

@Repository

- **Purpose:** Specialization of @Component, used for data access logic (DAOs).
- **Extra feature:** Enables automatic exception translation of persistence exceptions to Spring's DataAccessException.

@Controller

- **Purpose:** Marks a class as a Spring MVC controller for handling web requests.
- **Use case:** Typically used for rendering views in traditional web apps (e.g., Thymeleaf templates).

@RestController

- **Purpose:** A shortcut for @Controller + @ResponseBody.
- **Use case:** Used in RESTful APIs, where the return value is sent directly as JSON/XML in the HTTP response body.

Tips

Spring Testing Annotations

@SpringBootTest

- **Purpose:** Boots the full application context for integration testing.
- **Use case:** Used when testing multiple components together (e.g., services, repositories, configs).

@DataJpaTest

- **Purpose:** Configures an in-memory database and Spring Data JPA for testing repository logic.
- **Use case:** Fast, isolated testing of the data access layer. Only loads beans related to JPA (like @Repository).

@WebMvcTest

- **Purpose:** Loads only the web layer (controllers, filters, etc.) without starting the full context.
- **Use case:** Unit testing Spring MVC controllers in isolation.

@MockBean

- **Purpose:** Replaces a real Spring bean with a Mockito mock in the test context.
- **Use case:** Used in conjunction with @SpringBootTest, @WebMvcTest, etc., to isolate dependencies.

Tips

Spring Boot Configuration & DevTools Annotations

@EnableConfigurationProperties

- **Purpose:** Enables support for @ConfigurationProperties annotated beans.
- **Use case:** Used when you want to register external configuration POJOs manually, especially in @Configuration classes.

@ConfigurationProperties

- **Purpose:** Binds external configuration (like application.yml or application.properties) to a Java POJO.
- **Use case:** Cleanly map structured config (e.g., server.port, app.name) to a class with setters or constructor binding.

@Profile

- **Purpose:** Specifies that a bean should be active only for specific Spring profiles.
- **Use case:** Useful for environment-specific configurations (e.g., dev, prod, test).

Software Engineering Tips

Find this useful? like and share this post with your friends.

#LearnEasyWithLaithy

Save