Spring Framework Annotations

Last Updated : 23 Jul, 2025

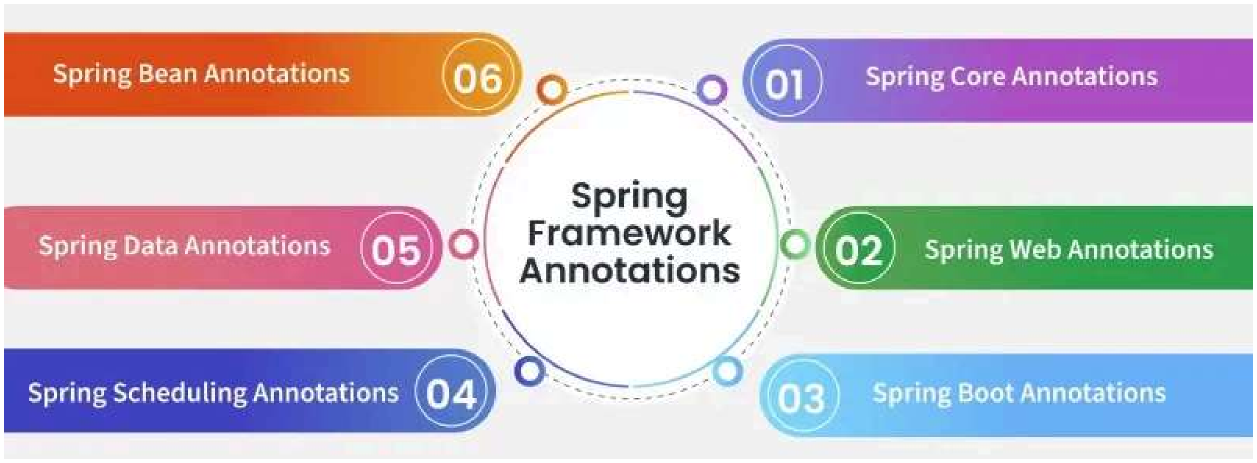
Spring framework is one of the most popular Java EE frameworks. It is an open-source lightweight framework that allows Java EE 7 developers to build simple, reliable, and scalable enterprise applications. Spring framework mainly focuses on providing various ways to help you manage your business objects.

Spring Annotations are a form of metadata that provides data about a program**.** Annotations are used to provide supplemental information about a program. They do not have a direct effect on the operation of the code they annotate, nor do they change the action of the compiled program. In this article, we will discuss the main types of annotations available in the Spring Framework with examples.

Types of Spring Framework Annotations

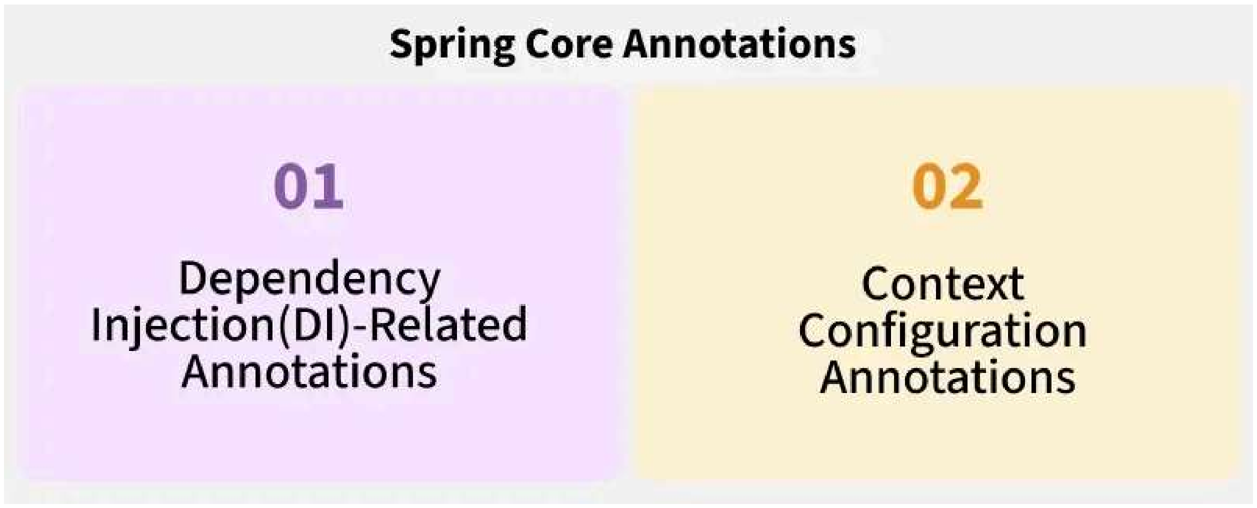
- Spring Core Annotations
- Spring Web Annotations
- Spring Boot Annotations
- Spring Scheduling Annotations
- Spring Data Annotations
- Spring Bean Annotations

The below diagram demonstrates the types of Spring Framework Annotations


Spring Framework Annotations
01 Spring Core Annotations
02 Spring Web Annotations
03 Spring Boot Annotations
04 Spring Scheduling Annotations
05 Spring Data Annotations
06 Spring Bean Annotations

1. Spring Core Annotations

Spring annotations are present in the **org.springframework.beans.factory.annotation** and **org.springframework.context.annotation** packages. These annotations can be divided into two categories


Spring Core Annotations
01 Dependency Injection(DI)-Related Annotations
02 Context Configuration Annotations

Dependency Injection(DI)-Related Annotations

@Autowired: This annotation is applied to **fields, setter methods, and constructors**. It injects object dependency implicitly. The Spring container injects the dependency when it finds this annotation.

Field Injection Example:

```
class Student {
   @Autowired
```

```
    Address address;
}
```

Constructor Injection Example:

```
class Student {
    Address address;

    @Autowired
    Student(Address address) {
        this.address = address;
    }
}
```

Setter Injection Example:

```
class Student {
    Address address;

    @Autowired
    void setAddress(Address address) {
        this.address = address;
    }
}
```

- **@Qualifier:** This annotation specifies which bean to inject when multiple beans of the same type are available.
- **@Primary:** This annotation marks a bean as the default one to inject when multiple candidates are present.
- **@Bean:** This annotation defines a Spring bean explicitly in a configuration class.
- **@Scope:** This annotation specifies the scope of a Spring bean (e.g., singleton, prototype).

- **@Value:** This annotation injects values into fields from property files.
- **@Lazy:** This annotation indicates that the bean should be lazily initialized.
- **@Required:** This annotation marks a property as required for injection.
- **@Lookup:** This annotation is used for method injection with prototype-scoped beans.

Context Configuration Annotations

@Profile: This annotation is used to indicate that a **@Component** class or a **@Bean** method should only be used when a specific profile is active.

Example:

> *@Component*
> *@Profile("developer")*
> *public class Employee {}*

- **@Import:** This annotation imports other configuration classes.
- **@ImportResources:** This annotation loads Spring XML configuration files.
- **@PropertySource:** This annotation specifies the location of property files for the application context.

2. Spring Web Annotations

Spring Web annotations are present in the **org.springframework.web.bind.annotation** package. Some of the commonly used annotations in this category are:

@Controller: This annotation marks a class as a Spring MVC controller. It is used with **@RequestMapping** to handle web requests.

Example:

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
```

```
@Controller
public class DemoController {
    @RequestMapping("/hello")
    @ResponseBody
    public String helloGFG() {
        return "Hello GeeksForGeeks";
    }
}
```

- **@RequestMapping:** This annotation maps HTTP requests to handler methods in the controller.
- **@RequestBody:** This annotation binds the body of the HTTP request to a Java object.
- **@PathVariable:** This annotation extracts values from the URI template.
- **@RequestParam:** This annotation extracts query parameters from the URL.
- **@ResponseBody:** This annotation indicates that the return value of the method should be used as the response body.
- **@ExceptionHandler:** This annotation is used to handle exceptions thrown by request-handling methods.
- **@ResponseStatus:** This annotation specifies the HTTP status code for the response.
- **@RestController:** This annotation is a combination of @Controller and @ResponseBody, used for RESTful web services.
- **@ModelAttribute:** This annotation binds a method parameter or method return value to a named model attribute.
- **@CrossOrigin:** This annotation enables cross-origin resource sharing (CORS) for specific handler methods or controller classes.

3. Spring Boot Annotations

Spring Boot annotations are present in the **org.springframework.boot.autoconfigure** and **org.springframework.boot.autoconfigure.condition** packages. Some of the commonly used annotations in this category are:

@SpringBootApplication: This annotation is used to mark the main class of a Spring Boot application. It encapsulates @Configuration,

@EnableAutoConfiguration, and @ComponentScan annotations with their default attributes.

Example:

```
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

- **@EnableAutoConfiguration:** This annotation enables Spring Boot's auto-configuration mechanism.
- **@ConditionalOnClass:** This annotation configures a bean only if a specified class is present on the classpath.
- **@ConditionalOnMissingBean:** This annotation configures a bean only if a specified bean is not already present in the application context.

4. Spring Scheduling Annotations

Spring Scheduling annotations are present in the **org.springframework.scheduling.annotation** package. Some of the commonly used annotations in this category are:

- **@EnableAsync:** This annotation enables asynchronous processing in Spring.
- **@EnableScheduling:** This annotation enables scheduling in Spring.
- **@Async:** This annotation marks a method to be executed asynchronously.
- **@Scheduled:** This annotation marks a method to be executed on a fixed schedule.

5. Spring Data Annotations

Spring Data annotations provide an abstraction over data storage technologies. Some of the commonly used annotations in this category are:

@Transactional: This annotation marks a method or class as transactional.

Example:

```
@Transactional
void performPayment() {}
```

@Id: This annotation marks a field in a model class as the primary key.

Example:

```
class Student {
    @Id
    Long id;
}
```

- **@Query:** This annotation defines custom queries for repositories.
- **@Procedure:** This annotation marks a method for calling a stored procedure.
- **@Lock:** This annotation specifies locking behavior for methods.
- **@Modifying:** This annotation marks a query as modifying (e.g., for update or delete operations).
- **@EnableJpaRepositories:** This annotation enables JPA repositories in a Spring application.
- **@Document (MongoDB):** This annotation marks a class as a MongoDB document.
- **@Field (MongoDB):** This annotation specifies a field in a MongoDB document.

6. Spring Bean Annotations

Spring Bean annotations are used to configure beans in a Spring container. Some of the commonly used annotations in this category are:

- **@ComponentScan:** This annotation is used to specify the base packages for scanning components.
- **@Configuration:** This annotation indicates that the class can be used by the Spring IoC container as a source of bean definitions.

- **@Stereotype:** This annotation is used to automatically register beand in the spring context based on their roles.

Example:

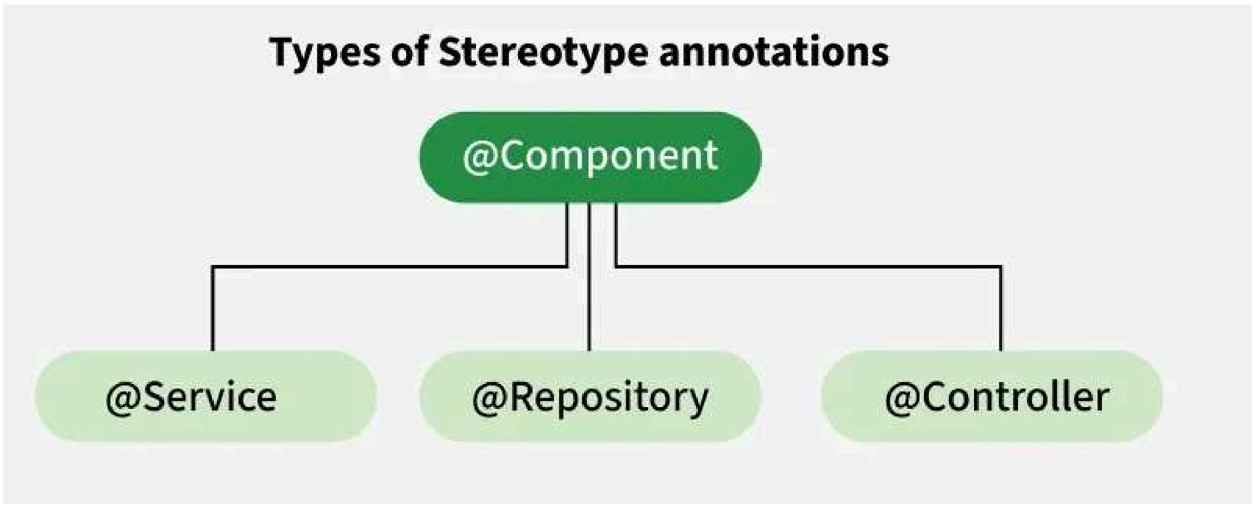
```
@Service
public class UserService {
    // business logic
}

@Repository
public class UserRepository {
    // CRUD operations
}

@Controller
public class UserController {
    // handle user requests
}
```

Types of @Stereotype Annotation

The below diagram demonstrates the types of Stereotype annotation



- **@Component:** This annotation is a generic stereotype for any Spring-managed component.
- **@Service:** This annotation is used to indicate that a class holds business logic.
- **@Repository:** This annotation is used to indicate that a class deals with CRUD operations, typically used with DAO or Repository

implementations.

- **@Controller:** This annotation is used to indicate that a class is a front controller, responsible for handling user requests.