Spring Annotations (Cheat Sheet)

1️⃣ Core Spring Annotations (Dependency Injection & Beans)

These annotations help in **defining beans** and handling **dependency injection** in a Spring application.

Annotation	Description	Example Usage
`@Component`	Marks a Java class as a **Spring-managed component**	`@Component class MyService {}`
`@Service`	Specialized `@Component` for **service layer**	`@Service class UserService {}`
`@Repository`	Specialized `@Component` for **DAO/repository layer**	`@Repository class UserRepository {}`
`@Controller`	Specialized `@Component` for **MVC controllers**	`@Controller class UserController {}`
`@RestController`	Combination of `@Controller` + `@ResponseBody` for REST APIs	`@RestController class ApiController {}`
`@Autowired`	Injects dependencies automatically	`@Autowired private UserService service;`
`@Qualifier`	Used with `@Autowired` to resolve conflicts when multiple beans of the same type exist	`@Autowired @Qualifier("beanName") private Service service;`
`@Bean`	Declares a **bean manually** inside `@Configuration` class	`@Bean public DataSource dataSource() { return new DataSource(); }`
`@Value`	Injects values from properties files	`@Value("${server.port}") private int port;`
`@Primary`	Marks a bean as **primary** when multiple beans exist	`@Primary @Bean public MyBean primaryBean() {}`

2️⃣ Spring Boot Configuration Annotations

These annotations help in configuring a Spring Boot application efficiently.

Annotation	Description	Example Usage
`@SpringBootApplication`	Combination of `@Configuration` + `@EnableAutoConfiguration` + `@ComponentScan`	`@SpringBootApplication class App {}`
`@Configuration`	Marks a class as a **configuration class**	`@Configuration class AppConfig {}`
`@ComponentScan`	Specifies package(s) to scan for components	`@ComponentScan("com.ednue")`
`@PropertySource`	Loads external properties files	`@PropertySource("classpath:app.properties")`
`@EnableAutoConfiguration`	Enables Spring Boot's **auto-configurations**	`@EnableAutoConfiguration`
`@ConditionalOnProperty`	Enables a bean **based on property value**	`@ConditionalOnProperty(name="feature.enabled", havingValue="true")`

3️⃣ Spring MVC & REST API Annotations

These annotations are used to create **RESTful APIs** and handle **HTTP requests**.

Annotation	Description	Example Usage
`@GetMapping`	Maps an HTTP GET request	`@GetMapping("/users")`
`@PostMapping`	Maps an HTTP POST request	`@PostMapping("/users")`
`@PutMapping`	Maps an HTTP PUT request	`@PutMapping("/users/{id}")`
`@DeleteMapping`	Maps an HTTP DELETE request	`@DeleteMapping("/users/{id}")`
`@RequestParam`	Extracts query parameters	`@RequestParam("name") String name`
`@PathVariable`	Extracts values from URL path	`@PathVariable("id") Long id`
`@RequestBody`	Maps request body to a Java object	`@RequestBody User user`
`@ResponseBody`	Sends Java object as **HTTP response (JSON/XML)**	`@ResponseBody User getUser()`
`@ExceptionHandler`	Handles specific exceptions globally	`@ExceptionHandler(RuntimeException.class)`
`@CrossOrigin`	Enables CORS for a REST API	`@CrossOrigin(origins="*")`

4️⃣ Spring AOP (Aspect-Oriented Programming) Annotations

These annotations help in handling **cross-cutting concerns** like logging, security, and transactions.

Annotation	Description	Example Usage
`@Aspect`	Defines a class as an **Aspect**	`@Aspect class LoggingAspect {}`
`@Before`	Executes **before** a method execution	`@Before("execution(* com.ednue.service.*.*(..))")`
`@After`	Executes **after** method execution	`@After("execution(* com.ednue.service.*.*(..))")`
`@Around`	Executes **before and after** a method	`@Around("execution(* com.ednue.service.*.*(..))")`
`@Pointcut`	Defines reusable pointcuts for advice	`@Pointcut("execution(* com.ednue.service.*.*(..))")`

5️⃣ Spring Security Annotations

These annotations help in securing Spring Boot applications.

Annotation	Description	Example Usage
`@EnableWebSecurity`	Enables Spring Security in the app	`@EnableWebSecurity class SecurityConfig {}`
`@PreAuthorize`	Restricts method access based on roles	`@PreAuthorize("hasRole('ADMIN')")`
`@Secured`	Restricts access to specific roles	`@Secured("ROLE_USER")`
`@EnableGlobalMethodSecurity`	Enables method-level security	`@EnableGlobalMethodSecurity(prePostEnabled=true)`
`@AuthenticationPrincipal`	Injects the **current logged-in user**	`@AuthenticationPrincipal User user`

6️⃣ Spring Transactions Annotations

These annotations handle database transactions efficiently.

Annotation	Description	Example Usage
`@Transactional`	Manages transactions automatically	`@Transactional class BankService {}`
`@EnableTransactionManagement`	Enables annotation-driven transaction management	`@EnableTransactionManagement`

7️⃣ Spring Scheduling Annotations

These annotations enable **scheduled task execution**.

Annotation	Description	Example Usage
`@EnableScheduling`	Enables scheduling support	`@EnableScheduling class SchedulerConfig {}`
`@Scheduled`	Runs a method at a fixed rate	`@Scheduled(fixedRate = 5000) public void task() {}`

8️⃣ Spring Caching Annotations

These annotations enable **caching** in Spring Boot applications.

Annotation	Description	Example Usage
`@EnableCaching`	Enables caching in the app	`@EnableCaching class CacheConfig {}`
`@Cacheable`	Caches method result	`@Cacheable("users") public User getUser(Long id)`
`@CacheEvict`	Removes an entry from cache	`@CacheEvict(value="users", key="#id")`

🔥 Quick Summary Table

Category	Key Annotations
Core Beans	`@Component`, `@Service`, `@Repository`, `@Controller`, `@RestController`, `@Bean`, `@Autowired`, `@Qualifier`
Spring Boot	`@SpringBootApplication`, `@Configuration`, `@ComponentScan`, `@PropertySource`
MVC & REST	`@GetMapping`, `@PostMapping`, `@RequestParam`, `@PathVariable`, `@RequestBody`, `@ResponseBody`
AOP	`@Aspect`, `@Before`, `@After`, `@Around`, `@Pointcut`

Security	`@EnableWebSecurity`, `@PreAuthorize`, `@Secured`
Transactions	`@Transactional`, `@EnableTransactionManagement`
Scheduling	`@EnableScheduling`, `@Scheduled`
Caching	`@EnableCaching`, `@Cacheable`, `@CacheEvict`

Key Notes:

- `@SpringBootApplication` includes `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`, so you **don't need to declare them separately**.
- `@RestController` is equivalent to `@Controller` + `@ResponseBody`.
- `@Service`, `@Repository`, and `@Controller` **inherit from** `@Component`, so no need to use `@Component` explicitly.

Source : https://github.com/gurukannan22/Java-Learning