ItsRunTym

Spring & Spring Boot

Annotations

Core Spring Annotations

1. **@Component**: Marks a Java class as a Spring component.
2. **@Controller**: Marks a Java class as a Spring MVC controller.
3. **@Service**: Marks a Java class as a service layer component.
4. **@Repository**: Marks a Java class as a data access object (DAO).
5. **@Configuration**: Indicates that a class declares one or more `@Bean` methods.
6. **@Bean**: Indicates that a method produces a bean to be managed by the Spring container.
7. **@Autowired**: Used for automatic dependency injection.
8. **@Qualifier**: Specifies which bean should be injected when multiple candidates are present.
9. **@Value**: Injects values into fields, method parameters, or constructor arguments.
10. **@Scope**: Configures the scope of a bean (e.g., singleton, prototype).
11. **@Lazy**: Marks a bean to be lazily initialized.
12. **@Primary**: Indicates that a bean should be given preference when multiple candidates are qualified.

Spring MVC Annotations

1. **@RequestMapping**: Maps HTTP requests to handler methods of MVC and REST controllers.
2. **@GetMapping**: Shortcut for `@RequestMapping(method = RequestMethod.GET)`.
3. **@PostMapping**: Shortcut for `@RequestMapping(method = RequestMethod.POST)`.
4. **@PutMapping**: Shortcut for `@RequestMapping(method = RequestMethod.PUT)`.
5. **@DeleteMapping**: Shortcut for `@RequestMapping(method = RequestMethod.DELETE)`.
6. **@PatchMapping**: Shortcut for `@RequestMapping(method = RequestMethod.PATCH)`.

7. **@RequestParam**: Binds a web request parameter to a method parameter.
8. **@PathVariable**: Binds a URI template variable to a method parameter.
9. **@RequestBody**: Binds the body of the web request to a method parameter.
10. **@ResponseBody**: Indicates that the return value of a method should be used as the response body.
11. **@RequestHeader**: Binds a method parameter to a request header.
12. **@CookieValue**: Binds a method parameter to a cookie value.
13. **@SessionAttribute**: Binds a method parameter to a session attribute.
14. **@ModelAttribute**: Binds a method parameter or return value to a named model attribute and exposes it to a web view.
15. **@InitBinder**: Identifies methods which initialize the WebDataBinder, which are used for customizing request parameter binding.
16. **@ExceptionHandler**: Defines the method that handles exceptions thrown by request-handling methods.

Spring Boot Annotations

1. **@SpringBootApplication**: Combines `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan` with their default attributes.
2. **@EnableAutoConfiguration**: Enables Spring Boot's auto-configuration mechanism.
3. **@ComponentScan**: Configures component scanning directives for use with `@Configuration` classes.
4. **@ConfigurationProperties**: Binds the properties defined in the external configuration files to the fields in a Java class.
5. **@SpringBootTest**: Used to bootstrap the entire container for integration tests.
6. **@TestConfiguration**: Indicates that a class declares one or more `@Bean` methods to be used in tests.
7. **@RestController**: Combines `@Controller` and `@ResponseBody`, simplifying the creation of RESTful web services.
8. **@RequestScope**: A specialized version of `@Scope` for a single HTTP request lifecycle.
9. **@SessionScope**: A specialized version of `@Scope` for a HTTP session lifecycle.
10. **@ApplicationScope**: A specialized version of `@Scope` for a web application lifecycle.

Transaction Management Annotations

1. **@Transactional**: Indicates that a method or class should be executed within a transactional context.
2. **@EnableTransactionManagement**: Enables Spring's annotation-driven transaction management capability.

Aspect-Oriented Programming (AOP) Annotations

1. **@Aspect**: Marks a class as an aspect.
2. **@Before**: Declares a before advice.
3. **@After**: Declares an after advice.
4. **@AfterReturning**: Declares a returning advice.
5. **@AfterThrowing**: Declares a throwing advice.
6. **@Around**: Declares an around advice.
7. **@Pointcut**: Declares a reusable pointcut expression.

Scheduling Annotations

1. **@EnableScheduling**: Enables Spring's scheduled task execution capability.
2. **@Scheduled**: Used for configuring scheduled tasks.
3. **@Async**: Marks a method or type as asynchronous.
4. **@EnableAsync**: Enables Spring's asynchronous method execution capability.

Caching Annotations

1. **@EnableCaching**: Enables Spring's annotation-driven cache management capability.
2. **@Cacheable**: Indicates that the result of invoking a method (or all methods in a class) can be cached.
3. **@CachePut**: Updates the cache with the result of a method.
4. **@CacheEvict**: Indicates that a cache entry should be removed.
5. **@Caching**: Allows multiple cache annotations to be used on a single method.

Spring Security Annotations

1. **@EnableWebSecurity**: Enables Spring Security's web security support.
2. **@Secured**: Specifies a list of security configuration attributes for method authorization.
3. **@PreAuthorize**: Allows method calls based on the evaluation of an expression.
4. **@PostAuthorize**: Allows method calls based on the evaluation of an expression after the method has been invoked.

5. **@RolesAllowed**: Specifies a list of roles allowed to access a method.

These annotations cover a wide range of functionality in Spring and Spring Boot applications, allowing developers to efficiently manage configuration, dependency injection, web requests, transactions, aspects, scheduling, caching, and security.

Core Spring Annotations

1. **@Component**

```
java

@Component
public class MyComponent {
    public void doSomething() {
        System.out.println("Doing something");
    }
}
```

2. **@Controller**

```
java

@Controller
public class MyController {
    @RequestMapping("/hello")
    public String hello() {
        return "hello";
    }
}
```

3. **@Service**

```
java

@Service
public class MyService {
    public void performService() {
        System.out.println("Service performed");
```

```
    }
}
```

4. **@Repository**

```
java

@Repository
public class MyRepository {
    public void saveData() {
        System.out.println("Data saved");
    }
}
```

5. **@Configuration**

```
java

@Configuration
public class AppConfig {
    @Bean
    public MyBean myBean() {
        return new MyBean();
    }
}
```

6. **@Bean**

```
java

@Configuration
public class AppConfig {
    @Bean
    public MyBean myBean() {
        return new MyBean();
    }
}
```

7. **@Autowired**

```
java

@Component
public class MyComponent {
    @Autowired
    private MyService myService;

    public void useService() {
        myService.performService();
    }
}
```

8. **@Qualifier**

```
java

@Component
public class MyComponent {
    @Autowired
    @Qualifier("specificService")
    private MyService myService;

    public void useService() {
        myService.performService();
    }
}
```

9. **@Value**

```
java

@Component
public class MyComponent {
    @Value("${my.property}")
    private String myProperty;

    public void printProperty() {
        System.out.println(myProperty);
    }
}
```

10. **@Scope**

```
java

@Component
@Scope("prototype")
public class MyPrototypeBean {
    public void showMessage() {
        System.out.println("Prototype Bean");
    }
}
```

11. **@Lazy**

```
java

@Component
@Lazy
public class LazyComponent {
    public LazyComponent() {
        System.out.println("LazyComponent initialized");
    }
}
```

12. **@Primary**

```
java

@Service
@Primary
public class PrimaryService implements MyService {
    public void performService() {
        System.out.println("Primary service performed");
    }
}
```

Spring MVC Annotations

1. **@RequestMapping**

```
java

@Controller
@RequestMapping("/api")
public class ApiController {
    @RequestMapping("/greet")
    public String greet() {
        return "Greetings from API";
    }
}
```

2. **@GetMapping**

```
java

@RestController
public class MyRestController {
    @GetMapping("/resource")
    public String getResource() {
        return "Resource data";
    }
}
```

3. **@PostMapping**

```
java

@RestController
public class MyRestController {
    @PostMapping("/resource")
    public String createResource(@RequestBody Resource
resource) {
        return "Resource created";
    }
}
```

4. **@PutMapping**

```
java
```

```
@RestController
public class MyRestController {
    @PutMapping("/resource")
    public String updateResource(@RequestBody Resource
resource) {
        return "Resource updated";
    }
}
```

5. **@DeleteMapping**

```
java

@RestController
public class MyRestController {
    @DeleteMapping("/resource/{id}")
    public String deleteResource(@PathVariable String id) {
        return "Resource deleted: " + id;
    }
}
```

6. **@RequestParam**

```
java

@RestController
public class MyRestController {
    @GetMapping("/resource")
    public String getResource(@RequestParam String id) {
        return "Resource ID: " + id;
    }
}
```

7. **@PathVariable**

```
java

@RestController
public class MyRestController {
    @GetMapping("/resource/{id}")
    public String getResource(@PathVariable String id) {
        return "Resource ID: " + id;
    }
}
```

8. **@RequestBody**

```
java

@RestController
public class MyRestController {
    @PostMapping("/resource")
```

```
    public String createResource(@RequestBody Resource
resource) {
        return "Resource created";
    }
}
```

9. **@ResponseBody**

```
java

@Controller
public class MyController {
    @RequestMapping("/data")
    @ResponseBody
    public String getData() {
        return "Raw data response";
    }
}
```

10. **@RequestHeader**

```
java

@RestController
public class MyRestController {
    @GetMapping("/header")
    public String getHeader(@RequestHeader("User-Agent")
String userAgent) {
        return "User-Agent: " + userAgent;
    }
}
```

11. **@CookieValue**

```
java

@RestController
public class MyRestController {
    @GetMapping("/cookie")
    public String getCookie(@CookieValue("sessionId") String
sessionId) {
        return "Session ID: " + sessionId;
    }
}
```

12. **@SessionAttribute**

```
java

@Controller
@SessionAttributes("user")
public class MyController {
    @GetMapping("/session")
```

```
    public String
getSessionAttribute(@SessionAttribute("user") User user) {
        return "User: " + user.getName();
    }
}
```

13.@ModelAttribute

```
java

@Controller
public class MyController {
    @ModelAttribute("user")
    public User addUser() {
        return new User("John");
    }

    @GetMapping("/user")
    public String getUser(@ModelAttribute("user") User user) {
        return "User: " + user.getName();
    }
}
```

14.@InitBinder

```
java

@Controller
public class MyController {
    @InitBinder
    public void initBinder(WebDataBinder binder) {
        binder.setDisallowedFields("id");
    }

    @PostMapping("/submit")
    public String submitForm(@ModelAttribute User user) {
        return "Submitted user: " + user.getName();
    }
}
```

15.@ExceptionHandler

```
java

@Controller
public class MyController {
    @ExceptionHandler(Exception.class)
    public String handleException(Exception ex) {
        return "Error: " + ex.getMessage();
    }
}
```

Spring Boot Annotations

1. **@SpringBootApplication**

```
java

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

2. **@EnableAutoConfiguration**

```
java

@Configuration
@EnableAutoConfiguration
public class MyConfig {
}
```

3. **@ComponentScan**

```
java

@Configuration
@ComponentScan(basePackages = "com.example")
public class MyConfig {
}
```

4. **@ConfigurationProperties**

```
java

@Component
@ConfigurationProperties(prefix = "my")
public class MyProperties {
    private String property;

    // getters and setters
}
```

5. **@SpringBootTest**

```
java

@SpringBootTest
public class MySpringBootTest {
    @Test
    public void contextLoads() {
    }
}
```

6. **@TestConfiguration**

```
java

@TestConfiguration
public class MyTestConfig {
    @Bean
    public MyBean myBean() {
        return new MyBean();
    }
}
```

7. **@RestController**

```
java

@RestController
public class MyRestController {
    @GetMapping("/hello")
    public String hello() {
        return "Hello, World!";
    }
}
```

8. **@RequestScope**

```
java

@Component
@RequestScope
public class RequestScopedBean {
    public void doSomething() {
        System.out.println("Request Scoped Bean");
    }
}
```

9. **@SessionScope**

```
java

@Component
@SessionScope
public class SessionScopedBean {
    public void doSomething() {
        System.out.println("Session Scoped Bean");
    }
}
```

10. **@ApplicationScope**

```
java

@Component
```

```
@ApplicationScope
public class ApplicationScopedBean {
    public void doSomething() {
        System.out.println("Application Scoped Bean");
    }
}
```

Transaction Management Annotations

1. @Transactional

```
java

import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class MyService {

    @Transactional
    public void performTransaction() {
        // transactional code here
    }
}
```

2. @EnableTransactionManagement

```
java

import org.springframework.context.annotation.Configuration;
import
org.springframework.transaction.annotation.EnableTransactionManageme
nt;

@Configuration
@EnableTransactionManagement
public class AppConfig {
}
```

Aspect-Oriented Programming (AOP) Annotations

1. @Aspect

```
java

import org.aspectj.lang.annotation.Aspect;

@Aspect
public class MyAspect {
}
```

2. @Before

```
java

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
public class MyAspect {

    @Before("execution(* com.example.MyService.*(..))")
    public void beforeAdvice() {
    }
}
```

3. @After

```
java

import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;

@Aspect
public class MyAspect {

    @After("execution(* com.example.MyService.*(..))")
    public void afterAdvice() {
    }
}
```

4. @AfterReturning

```
java

import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;

@Aspect
public class MyAspect {

    @AfterReturning(pointcut = "execution(*
com.example.MyService.*(..))", returning = "result")
    public void afterReturningAdvice(Object result) {
    }
}
```

5. @AfterThrowing

```
java

import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Aspect;

@Aspect
public class MyAspect {
```

```
    @AfterThrowing(pointcut = "execution(*
com.example.MyService.*(..))", throwing = "error")
    public void afterThrowingAdvice(Throwable error) {
    }
}
```

6. @Around

```
java

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;

@Aspect
public class MyAspect {

    @Around("execution(* com.example.MyService.*(..))")
    public Object aroundAdvice(ProceedingJoinPoint joinPoint) throws
Throwable {
        return joinPoint.proceed();
    }
}
```

7. @Pointcut

```
java

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class MyAspect {

    @Pointcut("execution(* com.example.MyService.*(..))")
    public void myPointcut() {
    }
}
```

Scheduling Annotations

1. @EnableScheduling

```
java

import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableScheduling;

@Configuration
@EnableScheduling
public class AppConfig {
}
```

2. @Scheduled

```
java

import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

@Component
public class MyScheduledTask {

    @Scheduled(fixedRate = 5000)
    public void performTask() {
    }
}
```

3. @Async

```
java

import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;

@Service
public class MyAsyncService {

    @Async
    public void performAsyncTask() {
    }
}
```

4. @EnableAsync

```
java

import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableAsync;

@Configuration
@EnableAsync
public class AppConfig {
}
```

Caching Annotations

1. @EnableCaching

```
java

import org.springframework.cache.annotation.EnableCaching;
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableCaching
```

```
public class AppConfig {
}
```

2. @Cacheable

```
java

import org.springframework.cache.annotation.Cacheable;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @Cacheable("items")
    public String getItem(int id) {
        return "item" + id;
    }
}
```

3. @CachePut

```
java

import org.springframework.cache.annotation.CachePut;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @CachePut("items")
    public String updateItem(int id) {
        return "updatedItem" + id;
    }
}
```

4. @CacheEvict

```
java

import org.springframework.cache.annotation.CacheEvict;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @CacheEvict("items")
    public void removeItem(int id) {
    }
}
```

5. @Caching

```
java
```

```
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.cache.annotation.CachePut;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.cache.annotation.Caching;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @Caching(
        cacheable = { @Cacheable("items") },
        put = { @CachePut("items") },
        evict = { @CacheEvict("items") }
    )
    public String manageItem(int id) {
        return "item" + id;
    }
}
```

Spring Security Annotations

1. @EnableWebSecurity

```
java

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.configuration.Ena
bleWebSecurity;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig {
}
```

2. @Secured

```
java

import org.springframework.security.access.annotation.Secured;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @Secured("ROLE_USER")
    public void securedMethod() {
    }
}
```

3. @PreAuthorize

```
java

import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @PreAuthorize("hasRole('ROLE_USER')")
    public void preAuthorizedMethod() {
    }
}
```

4. @PostAuthorize

```
java

import org.springframework.security.access.prepost.PostAuthorize;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @PostAuthorize("returnObject == authentication.name")
    public String postAuthorizedMethod() {
        return "username";
    }
}
```

5. @RolesAllowed

```
java

import javax.annotation.security.RolesAllowed;
import org.springframework.stereotype.Service;

@Service
public class MyService {

    @RolesAllowed("ROLE_USER")
    public void rolesAllowedMethod() {
    }
}
```