Guía Anotaciones Spring + Spring Boot

PABLO DEL ÁLAMO

1 Configuración de Aplicaciones

- **@SpringBootApplication:** Configura automáticamente tu aplicación en Spring Boot. Es una combinación de otras tres anotaciones (@EnableAutoConfiguration, @ComponentScan y @Configuration).

- **@EnableAutoConfiguration:** Hace que Spring Boot intente configurar automáticamente la aplicación según las dependencias que tengas.

PABLO DEL ÁLAMO

- **@Configuration:** Declara una clase como fuente de configuración para tu aplicación. Generalmente contiene métodos que crean y configuran "beans".

- **@ConfigurationProperties:** Permite cargar valores de configuración personalizados desde un archivo de propiedades (.properties o .yml) a un objeto específico.

PABLO DEL ÁLAMO

2 Componentes y Organización

- **@Component:** Marca una clase como "bean" (componente) para que Spring pueda gestionarla. Se usa para cualquier clase que quieras que Spring maneje automáticamente.

- **@Service:** Similar a @Component, pero se usa para las clases de lógica de negocio, lo que hace el código más fácil de entender.

PABLO DEL ÁLAMO

- **@Repository:** Especial para clases de acceso a la base de datos. También es una subclase de @Component.

- **@Controller:** Marca una clase como un controlador, lo que significa que manejará solicitudes HTTP. Ideal para aplicaciones web.

- **@RestController:** Es una combinación de @Controller y @ResponseBody, que facilita devolver datos (JSON, por ejemplo) directamente como respuesta en APIs REST.

PABLO DEL ÁLAMO

- **@Scope:** Define el alcance de un bean (singleton, prototype, etc.).

- **@Lazy:** Indica que el bean debe ser inicializado de forma perezosa.

PABLO DEL ÁLAMO

Manejo de Datos

- **@Entity:** Marca una clase como una entidad JPA.
- **@Table:** Especifica el nombre de la tabla de la base de datos para la entidad.

PABLO DEL ÁLAMO

- **@Column:** Define las propiedades de una columna en la base de datos.

- **@Id:** Indica la clave primaria de una entidad.

- **@GeneratedValue:** Especifica cómo se generará el valor de la clave primaria.

- **@OneToOne:** Define una relación uno a uno entre entidades.

- **@OneToMany:** Define una relación uno a muchos entre entidades.

PABLO DEL ÁLAMO

- **@ManyToOne:** Define una relación muchos a uno entre entidades.

- **@ManyToMany:** Define una relación muchos a muchos entre entidades.

- **@Enumerated:** Especifica cómo se deben persistir los tipos enumerados.

- **@Transient:** Indica que un campo no debe ser persistido en la base de datos.

PABLO DEL ÁLAMO

Configuración de Propiedades

- **@Value:** Inyecta valores de propiedades en los campos.
- **@ConfigurationProperties:** Mapea propiedades externas a un objeto Java.

PABLO DEL ÁLAMO

5 Seguridad

- **@EnableWebSecurity:** Habilita la seguridad web.

- **@Secured:** Restringe el acceso a métodos según los roles especificados.

- **@PreAuthorize:** Permite expresiones de autorización más complejas.

PABLO DEL ÁLAMO

6

Manejo de Excepciones

- **@ControllerAdvice:** Permite manejar excepciones en controladores.
- **@ExceptionHandler:** Maneja excepciones lanzadas por controladores.
- **@ResponseStatus:** Asigna un código de estado HTTP a una excepción.

PABLO DEL ÁLAMO

7 Pruebas

- **@SpringBootTest:** Indica que la clase es una prueba que debería cargar el contexto de la aplicación.

- **@MockBean:** Crea un mock de un bean en el contexto de prueba.

- **@DataJpaTest:** Carga una configuración de prueba para acceso a datos con JPA.

- **@WebMvcTest:** Carga solo los componentes de MVC y permite probar controladores.

PABLO DEL ÁLAMO

8 Programación Reactiva

- **@EnableReactiveWeb:** Habilita la programación reactiva en aplicaciones web.

- **@RestController:** Controlador que utiliza programación reactiva.

PABLO DEL ÁLAMO

Programación Asincrónica

- **@Async:** Marca un método como asincrónico.
- **@EnableAsync:** Habilita la ejecución de métodos asincrónicos.

PABLO DEL ÁLAMO

10 Configuraciones Web

- **@RequestMapping:** Mapea solicitudes HTTP a métodos de controlador.

- **@GetMapping:** Atajo para @RequestMapping(method = RequestMethod.GET).

- **@PostMapping:** Atajo para @RequestMapping(method = RequestMethod.POST).

PABLO DEL ÁLAMO

- **@PutMapping:** Atajo para @RequestMapping(method = RequestMethod.PUT).

- **@DeleteMapping:** Atajo para @RequestMapping(method = RequestMethod.DELETE).

- **@PatchMapping:** Atajo para @RequestMapping(method = RequestMethod.PATCH).

- **@PathVariable:** Vincula un valor de URI a un parámetro de método.

PABLO DEL ÁLAMO

- **@RequestParam:** Vincula un parámetro de consulta a un parámetro de método.

- **@RequestBody:** Vincula el cuerpo de la solicitud a un objeto de método.

- **@ResponseBody:** Indica que el valor devuelto de un método se serializa en el cuerpo de la respuesta.

- **@CookieValue:** Vincula un valor de cookie a un parámetro de método.

PABLO DEL ÁLAMO

11 Otros

- **@Profile:** Indica que un bean debe estar disponible solo en un perfil específico.

- **@Transactional:** Define el alcance de una transacción.

- **@EventListener:** Escucha eventos de la aplicación.

PABLO DEL ÁLAMO

- **@Scheduled:** Marca un método que debe ejecutarse en un intervalo programado.

- **@Conditional:** Indica que un bean debe ser creado solo si se cumple una condición.

- **@Autowired:** Permite la inyección automática de dependencias.

- **@Qualifier:** Especifica qué bean se debe inyectar cuando hay múltiples candidatos.

- **@Resource:** Inyecta un recurso de JNDI o un bean de Spring.

PABLO DEL ÁLAMO

- **@PreDestroy:** Indica que un método debe ejecutarse antes de que el bean sea destruido.

- **@Scheduled:** Marca un método para ejecución programada.

- **@RequestScope:** Define un alcance de solicitud para un bean.

- **@SessionScope:** Define un alcance de sesión para un bean.

- **@ApplicationScope:** Define un alcance de aplicación para un bean.

- **@PostConstruct:** Indica que un método debe ejecutarse después de que se haya completado la inyección de dependencias.

PABLO DEL ÁLAMO

12 Anotaciones para Aspectos

- **@Aspect:** Define una clase como un aspecto.
- **@Pointcut:** Define una expresión que apunta a un conjunto de uniones de métodos.
- **@Before:** Define un consejo que se ejecuta antes de una unión de métodos.
- **@After:** Define un consejo que se ejecuta después de una unión de métodos.
- **@Around:** Define un consejo que se ejecuta alrededor de una unión de métodos.

PABLO DEL ÁLAMO

13 Anotaciones de Batch

- **@EnableBatchProcessing:** Habilita el procesamiento por lotes en la aplicación.

- **@Job:** Define un trabajo en el contexto de procesamiento por lotes.

- **@Step:** Define un paso dentro de un trabajo de procesamiento por lotes.

PABLO DEL ÁLAMO

14 Anotaciones de Mensajería

- **@EnableJms:** Habilita el soporte para Java Messaging Service.
- **@JmsListener:** Marca un método como un oyente JMS que recibe mensajes.

PABLO DEL ÁLAMO

15 Anotaciones de Configuración de Microservicios

- **@EnableDiscoveryClient:** Habilita la funcionalidad de descubrimiento de servicios en microservicios.

- **@EnableCircuitBreaker:** Habilita el patrón Circuit Breaker en la aplicación.

- **@LoadBalanced:** Indica que un RestTemplate debe ser balanceado en carga.

PABLO DEL ÁLAMO

¿Te ha resultado útil?

- Comparte esta guía con tu equipo o amigos desarrolladores.
- Guárdala para tenerla siempre a mano.
- ¡Dale un like o comenta si tienes preguntas!

PABLO DEL ÁLAMO