**Notaciones de Spring Framework**

**Configuración**

1. **@SpringBootApplication**
   * Marca la clase principal de una aplicación Spring Boot.
   * Combina:
     + @Configuration: Indica que la clase tiene definiciones de beans.
     + @EnableAutoConfiguration: Habilita la configuración automática.
     + @ComponentScan: Escanea paquetes para detectar componentes y configuraciones.
2. **@Configuration**
   * Declara que una clase tiene métodos que producen beans gestionados por Spring.
3. **@Bean**
   * Declara un método que retorna un bean administrado por el contenedor de Spring.
4. **@PropertySource**
   * Especifica un archivo de propiedades que debe ser cargado en el entorno.

**Inyección de Dependencias**

1. **@Autowired**
   * Inyecta automáticamente una dependencia en un componente o bean.
2. **@Qualifier**
   * Especifica qué bean se debe inyectar cuando hay varias implementaciones disponibles.
3. **@Primary**
   * Marca un bean como el principal cuando hay múltiples candidatos para inyección.

**Definición de Componentes**

1. **@Component**
   * Marca una clase como un componente gestionado por Spring.
2. **@Service**
   * Especialización de @Component para clases de lógica de negocio.
3. **@Repository**

* Especialización de @Component para clases que acceden a la base de datos.

1. **@Controller**

* Marca una clase como un controlador que maneja solicitudes web.

1. **@RestController**

* Combina @Controller y @ResponseBody para simplificar controladores que devuelven datos en formato JSON.

**Manejo de Solicitudes HTTP**

1. **@RequestMapping**

* Mapea solicitudes HTTP a un controlador o método específico.
  + Puede usarse con subnotaciones como:
    - @GetMapping
    - @PostMapping
    - @PutMapping
    - @DeleteMapping
    - @PatchMapping

1. **@PathVariable**

* Extrae valores de la URL (por ejemplo, /api/users/{id}).

1. **@RequestParam**

* Recupera parámetros de la consulta (query params) de la URL.

1. **@RequestBody**

* Vincula el cuerpo de una solicitud HTTP a un objeto Java.

1. **@ResponseBody**

* Indica que el valor devuelto por un método debe ser el cuerpo de la respuesta.

1. **@CrossOrigin**

* Habilita solicitudes CORS desde otros dominios.

**Seguridad**

1. **@PreAuthorize**

* Especifica expresiones de seguridad antes de ejecutar un método.

1. **@Secured**

* Restringe el acceso a métodos según roles específicos.

1. **@RolesAllowed**

* Similar a @Secured, especifica roles permitidos para un método.

**Manejo de Errores**

1. **@ExceptionHandler**

* Maneja excepciones específicas en un controlador.

1. **@ControllerAdvice**

* Define manejo global de excepciones para controladores.

1. **@ResponseStatus**

* Define el código de estado HTTP a devolver para una excepción.

**Persistencia con Spring Data**

1. **@Entity**

* Marca una clase como una entidad JPA.

1. **@Id**

* Define el campo de clave primaria en una entidad.

1. **@GeneratedValue**

* Indica que el valor de la clave primaria será generado automáticamente.

1. **@Table**

* Especifica el nombre de la tabla asociada a una entidad.

1. **@Column**

* Configura detalles de una columna específica en una tabla.

1. **@ManyToOne**, **@OneToMany**, **@OneToOne**, **@ManyToMany**

* Configuran relaciones entre entidades.

1. **@Transactional**

* Indica que un método o clase debe ejecutarse dentro de una transacción.

**Validación**

1. **@Valid**

* Activa la validación de un objeto basado en anotaciones de Bean Validation.

1. **@NotNull**, **@Size**, **@Email**, etc.

* Validaciones específicas para campos en un modelo.

**Otros**

1. **@Scheduled**

* Programa tareas para ejecutarse en intervalos específicos.

1. **@Async**

* Permite la ejecución asíncrona de métodos.

1. **@EnableCaching**

* Activa la funcionalidad de caché.

1. **@Value**

* Inyecta valores desde el archivo de configuración (application.properties o application.yml).

**@GeneratedValue**

| **Estrategia** | **Base de Datos Ideal** | **Características** |
| --- | --- | --- |
| GenerationType.AUTO | Cualquiera | Delega al proveedor de JPA la elección de la estrategia más adecuada para la base de datos utilizada. |
| GenerationType.IDENTITY | MySQL, PostgreSQL | Utiliza auto-incremento; sencillo de usar pero menos eficiente en transacciones masivas. |
| GenerationType.SEQUENCE | Oracle, PostgreSQL | Altamente eficiente, recomendado para bases de datos que soportan secuencias. |
| GenerationType.TABLE | Cualquier base de datos | Requiere una tabla extra; más lento y rara vez se utiliza. |