Backend de aplicaciones

Apunte 13
Spring Data

Introducción a Spring Data

- **Definición:** Spring Data es una extensión del Spring Framework, diseñada para simplificar la integración con múltiples fuentes de almacenamiento.

- **Submódulos:** Cuenta con diversos submódulos para adaptarse a diferentes tecnologías de bases de datos.

- **Automatización:** Permite la creación automática de repositorios a través de simples interfaces.

Configurando Spring Data

- **Dependencias:** Añadir al **pom.xml** o usar **initializr**.
- **Datos de Conexión:** Se configuran en **application.properties** (URL, Usuario, Contraseña).
- **Integración:** Spring Data se integra con facilidad con el núcleo del Spring Framework.

Módulos de Spring Data

- **Spring Data JPA:** Para Java Persistence API.
- **Spring Data JDBC:** Conexiones con bases de datos relacionales.
- **Spring Data MongoDB:** Optimizado para MongoDB.
- **Spring Data Rest:** Convierte repositorios en servicios REST.
- **Otros:** LDAP, Apache Cassandra, Apache Solar, etc.

Principales Interfaces de Repositorios

- **CrudRepository:** Operaciones CRUD básicas.
- **PagingAndSortingRepository:** Ofrece paginación y ordenación.
- **JpaRepository:** Más funcionalidades como gestión de caché y eliminación por lotes.
- **MongoRepository:** Diseñado para MongoDB.
- **ElasticsearchRepository:** Interacciones con Elasticsearch.
- **CassandraRepository:** Operaciones con Apache Cassandra.
- **Neo4jRepository:** Para bases de datos de grafo Neo4j.

Zoom en CrudRepository

- **Métodos Cruciales:** save(), findById(), delete(), findAll(), y más.
- **Versatilidad:** Adaptado para cualquier entidad.
- **Automatización:** Genera consultas y operaciones automáticamente.

JpaRepository: Ampliando las Posibilidades

- **Herencia:** Extiende CrudRepository y PagingAndSortingRepository.
- **Funcionalidad Especial:** Método saveAndFlush() para escritura inmediata en DB.
- **Optimizado:** Ideal para aplicaciones con JPA.

Consultas Dinámicas

- **Introducción:** Spring Data JPA permite la creación de consultas dinámicas simplemente por el nombre del método en el repositorio.
- **Beneficio:** Reduce la necesidad de escribir consultas SQL manuales.
- Ejemplos:

```
List<Persona> findByNombre(String nombre);
List<Persona> findByNombreAndApellido(String nombre, String apellido);
```

- Nota: El uso de "And" y "Or" en el nombre del método define cómo se conectan los criterios.

@Query - Consultas Personalizadas

- **Introducción:** @Query permite definir consultas específicas en JPQL o SQL nativo.
- **Beneficio:** Mayor flexibilidad y control sobre las consultas, especialmente útil para consultas complejas.
- **Ejemplo**

```
@Query("SELECT p FROM Persona p WHERE p.edad BETWEEN :minEdad AND :maxEdad AND LOWER(p.nombre) LIKE %:nombre%")
List<Persona> buscarPorEdadYNombre(int minEdad, int maxEdad, String nombre);
```

- **Nota:** Los parámetros en :nombreFormato en la consulta se enlazan con los parámetros del método.

¡GRACIAS!

¿Preguntas?