EBook Gratis

APRENDIZAJE spring-data-jpa

Free unaffiliated eBook created from
Stack Overflow contributors.

#spring-data-jpa

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: spring-data-jpa

It is an unofficial and free spring-data-jpa ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official spring-data-jpa.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con spring-data-jpa

Observaciones

Esta sección proporciona una descripción general de qué es spring-data-jpa y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de spring-data-jpa, y vincular a los temas relacionados. Dado que la Documentación para spring-data-jpa es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Versiones

Versión	Fecha de lanzamiento
1.9.0.RELEASE	2015-09-01
1.8.0.RELEASE	2015-03-23
1.7.0.RELEASE	2014-09-05
1.6.0.RELEASE	2014-05-20
1.5.0.RELEASE	2014-02-24
1.4.0.RELEASE	2013-09-09
1.3.0.RELEASE	2012-02-07
1.2.0.RELEASE	2012-10-10
1.1.0.RELEASE	2012-05-16
1.0.0.RELEASE	2011-07-21

Examples

Instalación o configuración

Para comenzar a usar la JPA de datos de Spring, debe incluir la dependencia en su proyecto con la de Spring core, todos juntos. Si está utilizando Maven como sistema de administración de dependencias (reemplace *el número* de versión de la versión que desea usar):

```
<dependencies>
    <dependency>
        <groupId>org.springframework.data</groupId>
```

```
        <artifactId>spring-data-jpa</artifactId>
        <version>version-number</version>
    </dependency>
</dependencies>
```

Y si estás usando Gradle:

```
dependencies {
    compile 'org.springframework.data:spring-data-jpa:version-number'
}
```

También puede configurarlo cuando use Spring Boot, solo incluya la dependencia de arranque y deshágase del número de versión:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
</dependencies>
```

buscar por propiedad de la entidad y buscar por propiedad de la entidad en

```
WarehosueEntity findWarehouseById(@Param("id") Long id);

List<WarehouseEntity> findWarehouseByIdIn(@Param("idList") List<Long> warehouseIdList);
```

Lea Empezando con spring-data-jpa en línea: https://riptutorial.com/es/spring-data-jpa/topic/4318/empezando-con-spring-data-jpa

Capítulo 2: Repositorios

Observaciones

El proyecto Spring Data permite a los programadores de aplicaciones trabajar con almacenes de datos utilizando una interfaz consistente que hace uso de una abstracción llamada `Repository` . Un `Repository` datos de Spring se modela siguiendo el patrón de repositorio popularizado por el diseño controlado por dominio . Spring Data proporciona una interfaz central de Java llamada `Repository` que los subproyectos pueden ampliar para proporcionar características específicas de los almacenes de datos.

Además de la interfaz del `Repository` , Spring Data también proporciona dos interfaces más: `CrudRepository` que define el contrato para la funcionalidad básica de *CRUD* (*crear* , *leer* , *actualizar* y *eliminar*); y `PagingAndSortingRepository` que extiende `CrudRepository` mediante la definición de un contrato para la paginación y la clasificación.

Estas tres interfaces principales (`Repository` , `CrudRepository` y `PagingAndSortingRepository`) aseguran que:

1. Los programadores de aplicaciones pueden acceder a los almacenes de datos (como bases de datos relacionales, bases de datos NoSQL basadas en documentos, bases de datos de gráficos, etc.) de manera consistente.
2. Es fácil cambiar el almacenamiento subyacente para una *entidad de dominio* (ver diseño controlado por dominio) sin tener que cambiar también la forma en que la aplicación interactúa con el almacén de datos.
3. Las implementaciones específicas pueden proporcionar características específicas para los almacenes de datos.

Examples

Creación de un repositorio para una entidad gestionada por JPA

Clase de **entidad**

```
@Entity
@Table(name = "USER")
public class User {
    @Id
    @Column(name = "ID")
    private Long id;

    @Column(name = "USERNAME")
    private String username;

    @ManyToOne
    @JoinColumn("ORGANIZATION_ID")
    private Organization organization;
}
```

Interfaz de **repositorio**

```
@Repository
public interface UserRepository extends CrudRepository<User, Long> {
    public User findByUsername(String username);
}
```

La declaración del método en la interfaz generará la siguiente consulta jpql:

```
select u from User u where u.username = :username
```

Alternativamente podemos definir una consulta personalizada:

```
@Query("select u from User u where u.username = :username")
public User findByUsername(@Param("username") String username)
```

Podemos agregar fácilmente la clasificación a la declaración del método:

```
public interface UserRepository extends PagingAndSortingRepository<User, Long> {
    public User findByUsernameOrderByUsernameAsc(String username);
}
```

También podemos usar el soporte de paginación incorporado:

```
public Page<User> findByOrganizationPaged(Organization organization, Pageable pageable);
```

la capa de servicio (o quienquiera que llame a este método) pasará una PageRequest al método:

```
public Page<User> getByOrganizationPagedOrderByUsername(Organization organization, int page,
int size, String direction){
    return userRepository.findByOrganizationPaged(organization, new PageRequest(page, size,
Direction.valueOf(direction),
                "username")
}
```

Encontrar todas las instancias de una clase de entidad

Todas las instancias (objetos) de una clase de entidad se pueden cargar desde la tabla de base de datos subyacente de la siguiente manera (similar a recuperar todas las filas de la tabla):

```
Iterable<Foo> foos = fooRepository.findAll();
```

El método `findAll` es proporcionado por la interfaz `CrudRepository` . Devuelve un tipo de `Iterable` lugar de un tipo más concreto, como `List` o `Set` ya que algunas implementaciones de la interfaz no pueden devolver un tipo de `Collection` y, por lo tanto, usar un tipo de `Collection` para el valor devuelto resultará en la pérdida de funcionalidad para ellos.

Al invocar los resultados del método `findAll` , la consulta JPA `select foo from Foo foo` se ejecuta en la base de datos subyacente.

Encontrar una instancia particular de una clase de entidad por el identificador

Una instancia particular de una clase de entidad se puede cargar de la siguiente manera:

```
Foo foo = fooRepository.findOne(id);
```

El método `findOne` es proporcionado por la interfaz `CrudRepository` . Espera un identificador que identifique de forma única una instancia de entidad (por ejemplo, una clave principal en una tabla de base de datos). El tipo de Java para el parámetro `id` debe coincidir con el tipo asignado al atributo de entidad anotado con la anotación JPA `@Id` .

Al invocar los resultados del método `findOne` , la consulta JPA `select foo from Foo foo where foo.[primary-key-column] = :id` se ejecuta en la base de datos subyacente.

Encontrar todas las instancias de una clase de entidad con un atributo que coincida con un valor especificado

Todas las instancias de una clase de entidad con uno de los atributos de clase que coinciden con un valor específico se pueden recuperar de la siguiente manera:

```
public interface FooRepository extends CrudRepository<Foo, Long> {
  List<Foo> findAllByName(String name);
}
```

Al invocar los resultados del método `findAllByName` en la consulta JPA, `select foo from Foo foo where foo.name = :name` se ejecutará en la base de datos subyacente.

Puntos a tener en cuenta

1. `name` debe ser un atributo en la clase de entidad `Foo` .
2. El nombre del método debe comenzar con `find` , `get` o `read` . Otras palabras clave como `select` no funcionarán.
3. No hay garantía sobre el orden en que se devolverán los resultados.

Lea Repositorios en línea: https://riptutorial.com/es/spring-data-jpa/topic/5688/repositorios

Creditos

S. No	Capítulos	Contributors
1	Empezando con spring-data-jpa	Community, Sheldon Papa, Xtreme Biker
2	Repositorios	amicoderozer, Gautam Jose, ido flax, manish, sanjaykumar81, SirKometa