http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte2:psl:data:jpa:querymethod

Description

User Guide

Query Method

The JPA Module defines queries by way of strings or uses method-derived queries.

Generating Queries

How to define queries by way of strings:

```
public interface UserRepository extends Repository<User, Long> {
List<User> findByEmailAddressAndLastname(String emailAddress, String lastname);
```

JPA-standard API alters the pre-defined queries into:

```
select u from User u where u.emailAddress = ?1 and u.lastname = ?2
```

Method Available

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull	findByAgeIsNull	… where x.age is null
IsNotNull,NotNull	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)

OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> age)	… where x.age not in ?1
True	findByActiveTrue()	… where x.active = true
False	findByActiveFalse()	… where x.active = false

Using @Query

Using named queries to declare queries for entities is a valid approach and works fine for a small number of queries. As the queries themselves are tied to the Java method that executes them you actually can bind them directly using the Spring Data JPA @Query annotation rather than annotating them to the domain class. This will free the domain class from persistence specific information and co-locate the query to the repository interface.

Keep in mind that the pre-defined queries in the query methods take precedence over the duly declared @NamedQuery and named queries in XML.

How to declare queries via @Query:

```
public interface UserRepository extends JpaRepository<User, Long> {
   @Query("select u from User u where u.emailAddress = ?1")
   User findByEmailAddress(String emailAddress);
}
```

@Query using parameters

By default Spring Data JPA will use position based parameter binding as described in all the samples above. This makes query methods a little error prone to refactoring regarding the parameter position. To solve this issue you can use @Param annotation to give a method parameter a concrete name and bind the name in the query: The Spring Data JPA uses the parameters based upon where they are bound. See the following example for how the queries are declared using parameters:

```
public interface UserRepository extends JpaRepository<User, Long> {

   @Query("select u from User u where u.firstname = :firstname or u.lastname = :lastname")
   User findByLastnameOrFirstname(@Param("lastname") String lastname,
                                  @Param("firstname") String firstname);
}
```

References

http://static.springsource.org/spring-data/data-jpa/docs/current/reference/html/jpa.repositories.html