**Giới thiệu**

Trong một bài viết gần nhất về [Criteria API](https://loda.me/huong-dan-su-dung-criteria-api-trong-hibernate-loda1552815848300) tôi đã hướng dẫn các bạn cách xây dựng các câu query xuống Database bằng các API do Hibernate cung cấp.

Và trong một bài viết khác về JPA Repository thì chúng ta cũng đã biết cách custom các query bằng cách đặt tên method:

1. [「Spring Boot #11」 Hướng dẫn Spring Boot JPA + MySQL](https://loda.me/spring-boot-11-huong-dan-spring-boot-jpa-my-sql-loda1558687596060)
2. [「Spring Boot #12」 Spring JPA Method + @Query](https://loda.me/spring-boot-12-spring-jpa-method-query-loda1558746200832)

Tuy nhiên, trong các phương pháp trên, vẫn sẽ còn một số các điểm bất cập, ví dụ như JpaRepository thì bạn sẽ phải viết quá nhiều method và mỗi cái sẽ phục vụ cho một mục đích cố định (không thể tái sử dụng, reuseable).

public interface UserRepository extends JpaRepository<User, Long> {

User findByEmailAddress(String emailAddress);

List<User> findByLastname(String lastname, Sort sort);

Page<User> findByFirstname(String firstname, Pageable pageable);

}

Để có thể tối ưu việc viết query một cách linh động hơn và có thể tái sử dụng lại, Spring mang tới cho chúng ta interface Specification.

Lúc này, cách tiếp cận của việc xây dựng query sẽ đại loại như sau:

userRepository.findAll(Specification.where(hasIdIn(Arrays.asList(1L, 2L, 3L, 4L, 5L)))

.and(hasType(UserType.NORMAL))

.or(hasId(10L)));

Với cách định nghĩa này, chúng ta có thể tái sử dụng query và tuỳ biến nó mọi lúc để phù hợp với yêu cầu.

Khái niệm Specification được xây dựng tương đương với Predicate trong Hibernate. Bạn hãy đọc bài dưới trước khi đi tiếp vào bài này:

1. [Hướng dẫn sử dụng Criteria API trong Hibernate (Phần 2)](https://loda.me/huong-dan-su-dung-criteria-api-trong-hibernate-phan-2-loda1575949746794)

Trong bài có sử dụng:

* [Lombok](https://loda.me/general-huong-dan-su-dung-lombok-giup-code-java-nhanh-hon-69-loda1552789752787)

**Cài đặt**

Nhớ thêm spring-boot-starter-data-jpa vào dependencies của bạn.

Trong phần này, tôi xài *H2 database* để demo. (H2 là dạng memory database, nó sẽ lưu trong ram và khi tắt chương trình nó sẽ mất sạch)

<?xml version="1.0" encoding="UTF-8"?>

<projectxmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:schemaLocation="http://maven.apache.org/POM/4.0.0https://maven.apache.org/xsd/maven-4.0.0.xsd"><modelVersion>4.0.0</modelVersion><parent><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-parent</artifactId><version>2.0.5.RELEASE</version><relativePath/> <!-- lookup parent from repository -->

</parent><groupId>me.loda.spring</groupId><artifactId>example-independent-maven-spring-project</artifactId><version>0.0.1-SNAPSHOT</version><name>example-independent-maven-spring-project</name><description>Demo project for Spring Boot</description><properties><java.version>1.8</java.version></properties><dependencies><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-web</artifactId></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-devtools</artifactId><scope>runtime</scope><optional>true</optional></dependency><dependency><groupId>org.projectlombok</groupId><artifactId>lombok</artifactId><optional>true</optional></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-test</artifactId><scope>test</scope><!-- <exclusions>-->

<!-- <exclusion>-->

<!-- <groupId>org.junit.vintage</groupId>-->

<!-- <artifactId>junit-vintage-engine</artifactId>-->

<!-- </exclusion>-->

<!-- </exclusions>-->

</dependency><!--spring jpa-->

<dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-data-jpa</artifactId></dependency><!--in memory database-->

<dependency><groupId>com.h2database</groupId><artifactId>h2</artifactId><scope>runtime</scope></dependency><!--https://mvnrepository.com/artifact/org.hibernate/hibernate-jpamodelgen -->

<dependency><groupId>org.hibernate</groupId><artifactId>hibernate-jpamodelgen</artifactId><version>5.4.9.Final</version><scope>provided</scope></dependency></dependencies><build><plugins><plugin><groupId>org.springframework.boot</groupId><artifactId>spring-boot-maven-plugin</artifactId></plugin></plugins></build></project>

Chú ý, xem kĩ hơn hibernate-jpamodelgen trong bài viết [này](https://loda.me/huong-dan-su-dung-criteria-api-trong-hibernate-phan-2-loda1575949746794)

**Specification**

Specification là một cách để định nghĩa các Predicate có thể tái sử dụng được.

Bản chất Specification là một funtional interface với 1 hàm duy nhất như sau:

public interface Specification<T> {

Predicate toPredicate(Root<T> root, CriteriaQuery query, CriteriaBuilder cb);

}

Tham số đầu vào là 3 khái niệm tôi đã giới thiệu ở bài [Criteria API](https://loda.me/huong-dan-su-dung-criteria-api-trong-hibernate-loda1552815848300), bao gồm:

1. Root
2. CriteriaQuery
3. CriteriaBuilder

tôi sẽ demo trước một số implementation của Specification và đề cập cách sử dụng nó ở phía dưới:

*User.java*

@Entity

@Data

@Builder

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private UserType type;

private String name;

public enum UserType {

NORMAL, VIP;

}

}

*UserSpecification.java*

public final class UserSpecification {

/\*\*

\* Lấy ra user có UserType chỉ định

\* @param type

\* @return

\*/

public static Specification<User> hasType(UserType type) {

return (root, query, cb) -> cb.equal(root.get(User\_.TYPE), type);

}

/\*\*

\* Lấy ra user có id chỉ định

\* @param userId

\* @return

\*/

public static Specification<User> hasId(long userId) {

return (root, query, cb) -> cb.equal(root.get(User\_.ID), userId);

}

/\*\*

\* Lấy ra user nằm trong tập ID chỉ định

\* @param userIds

\* @return

\*/

public static Specification<User> hasIdIn(Collection<Long> userIds) {

return (root, query, cb) -> root.get(User\_.ID).in(userIds);

}

}

Tôi định nghĩa ra các static method trả ra ngoài là các implement của Specification. Nó không hẳn là đoạn code đẹp nhất ==! nhưng nó là một cách làm hay để có thể định nghĩa một tập các điều kiện có thể sử dụng lại được cho User.

**JpaSpecificationExecutor**

Để có thể sử dụng được Specification, bạn cần kế thừa JpaSpecificationExecutor từ Spring JPA

public interface UserRepository extends JpaRepository<User, Long>,

JpaSpecificationExecutor<User> {

}

Lúc này, ngoài các method truyền thống như findAll(), findOne(), findBy() thì bạn sẽ thấy xuất hiện các method mới có tham số đầu vào là Specification<T>:

Optional<T> findOne(@Nullable Specification<T> var1);

List<T> findAll(@Nullable Specification<T> var1);

Page<T> findAll(@Nullable Specification<T> var1, Pageable var2);

List<T> findAll(@Nullable Specification<T> var1, Sort var2);

long count(@Nullable Specification<T> var1);

**Usage**

Lúc này, để sử dụng, bạn gọi Specification.where() để xây dựng cho mình tập các điều kiện để query

// Lấy ra user nằm trong tập ID đã cho và có type là NORMAL

// hoặc lấy ra user có ID = 10

Specification conditions = Specification.where(UserSpecification.hasIdIn(Arrays.asList(1L, 2L, 3L, 4L, 5L)))

.and(UserSpecification.hasType(UserType.NORMAL))

.or(UserSpecification.hasId(10L));

// Truyền Specification vào hàm findAll()

userRepository.findAll(conditions).forEach(System.out::println);

OUTPUT:

User(id=1, type=NORMAL, name=name-0)

User(id=2, type=NORMAL, name=name-1)

User(id=5, type=NORMAL, name=name-4)

User(id=10, type=VIP, name=name-9)

Ngoài ra, bạn có thể import trực tiếp các hàm static vào để code gọn hơn:

import static me.loda.spring.specification.User.UserType.NORMAL;

import static me.loda.spring.specification.UserSpecification.\*;

...

// Lấy ra user nằm trong tập ID đã cho và có type là NORMAL

// hoặc lấy ra user có ID = 10

Specification conditions = Specification.where(hasIdIn(Arrays.asList(1L, 2L, 3L, 4L, 5L)))

.and(hasType(NORMAL))

.or(hasId(10L));

// Truyền Specification vào hàm findAll()

userRepository.findAll(conditions).forEach(System.out::println);

Gọn hơn thì không nên tạo ra 1 biến tham chiếu thừa thãi:

userRepository.findAll(Specification.where(hasIdIn(Arrays.asList(1L, 2L, 3L, 4L, 5L)))

.and(hasType(NORMAL))

.or(hasId(10L)))

.forEach(System.out::println);

**Kết**

Tới đây bạn đã có thể sử dụng Specification để thực hiện các truy vấn phức tạp và tái sử dụng được trong nhiều trường hợp. Đón đọc các bài sau về các phần nâng cao hơn.