Custom Specification tools

Người viết: Vũ Văn Thưởng

Người kiểm thử: Vũ Văn Thưởng

# Giới thiệu sơ lược

Một đoạn code trợ giúp xây dựng câu lệnh hql trong jpa/hibernate

Có thể ứng dụng trong việc tạo dựng lệnh(query) lọc( filter).

Mục lục

# Class:

## class CustomSpecification

Contructor:

**public** CustomSpecification(Object filter)

filter chứa các field lọc.

Static method:

**public** **static** Predicate toCustomPredicate(From<?,?> root, CriteriaBuilder cb, Object filter)

Trả về điều kiện tạo ra bởi filter

**@param** root

**@param** cb

**@param** filter

**@return Predicate**

**@exception** common.filter.SearchException

**public** **static** Expression<Boolean> conditionWithColumn(From<?,?> root, CriteriaBuilder cb, String fieldName, Object fieldValue, CompareType compare, SearchArrayType arrayType)

Trả về điều kiện tạo ra bởi 1 giá trị fieldValue

**@param** root

**@param** cb

**@param** fieldname // Tên field

**@param** fieldValue // Giá trị so sánh

**@param** compare // Phép toán so sánh

**@param** arrayType // Kiểu mảng. Nếu không phải mảng thì có thể nhập SearchArrayType.All

**@return** Expression<Boolean>

**public** **static** Expression<Boolean> conditionWithJoinColumn(From<?,?> root, CriteriaBuilder cb, String fieldName, Object fieldValue, CompareType compare, SearchArrayType arrayType, JoinType joinType)

**@param** root

**@param** cb

**@param** fieldname // Tên field

**@param** fieldValue // Giá trị so sánh

**@param** compare // Phép toán so sánh

**@param** arrayType // Kiểu mảng. Nếu không phải mảng thì có thể nhập SearchArrayType.All

**@param** joinType // Kiểu liên kết (left, right, inner join)

**@return** Expression<Boolean>

**@exception** common.filter.SearchException

**public** **static** Predicate joinList(CriteriaBuilder cb, List<Expression<Boolean> > expressions, SearchArrayType type )

Liên kết các câu điều kiện

**@param** cb

**@param** expressions // Kiểu List, là các điều kiện cần liên kết

**@param** type // Kiểu liên kết (all, not all, any, not any, in , not in)

**@return** Predicate

Instance method:

**public** Predicate toPredicate(Root<T> root, CriteriaQuery<?> query, CriteriaBuilder cb)

Trả về điều kiện ứng với filter

**@param** root

**@param** query

**@param** cb

**@return** Predicate

**@exception** common.filter.SearchException

**public** FetchAttribute addSubFetch(String attributeName, JoinType joinType)

Thêm fetch, trả về Fetch vừa tạo ra

**@param** attributeName

**@param** joinType

**@return** FetchAttribute

**public** Object getFilter()

Trả về object filter

**@return** Object

**public** **void** setFilter(Object filter)

Thay đổi filter

**@param** filter

**public** Set<FetchAttribute> getFetchFields()

Lấy danh sách fetch

**@return** Set<FetchAttribute>

## public class SearchException extends RuntimeException

Trả về khi quá trình xây dựng điều kiện xảy ra lỗi

## public class FetchAttribute

Lưu trữ các bảng fetch

Contructor

**public** FetchAttribute(String attribute, JoinType joinType)

Method:

**public** String getAttribute()

Trả về tên bảng fetch

**public** Set<FetchAttribute> getSubFetchAttribute()

Trả về danh sách fetch với các bảng khác từ fetch này

**public** FetchAttribute addSubFetch(String name, JoinType joinType)

Thêm fetch từ fetch này. Trả về fetch vừa tạo ra

**public** JoinType getJoinType()

Trả về kiều liên kết

# Enum:

## public enum CompareType

**Equal:** So sánh bằng

**NotEqual:** So sánh khác

**Like:** So sánh biểu thức chính quy

**GreaterThan:** So sánh lớn hơn

**LessThan:** So sánh nhỏ hơn

**EqualLessThan:** So sánh nhỏ hơn hoặc bắng

**EqualGreaterThan:** So sánh lớn hơn hoặc bằng

**HasContains:** Kiểm tra có chuỗi con hay không

## public enum SearchArrayType

***Any***, Thoả mãn một điều kiện trong dãy(array hoặc list)

***NotAny***, Không thoả mãn bất kỳ một điều kiện trong dãy(array hoặc list)

***All*** Thoả mãn tất cả điều kiện trong dãy(array hoặc list)

***NotAll*** Không thoả mãn tất cả điều kiện trong dãy(array hoặc list)

***In*** tương đương Any

***NotIn*** tương đương notAny

# Annotation

## public @interface SearchColumn

String name() **default** "": tên field trong entity

CompareType compare() **default** CompareType.***Equal***: Cách so sánh

SearchArrayType arrayType() **default** SearchArrayType.***All***: kiểu kết hợp mảng

## public @interface SearchJoinColumn

String name() **default** "": tên field trong entity

CompareType compare() **default** CompareType.***Equal***: Cách so sánh

SearchArrayType arrayType() **default** SearchArrayType.***All***: kiểu kết hợp mảng

JoinType joinType() **default** JoinType.***LEFT***: Kiểu liên kết bảng

# Ví dụ:

## Entity

@Entity

@Table(name="employee")

@Data

**public** **class** Employee {

@Id

**private** Integer id;

**private** String name;

**private** **int** age;

@ManyToOne

@JoinColumn(name="department\_id")

**private** Department department;

}

@Entity

@Table(name = "department")

@Data

**public** **class** Department {

@Id

**private** Integer id;

**private** String name;

}

## Filter:

@Data

**public** **class** EmployeeFilter {

@SearchColumn(compare = CompareType.***Equal***)

**private** Integer id;

@SearchColumn(compare = CompareType.***HasContains***)

**private** String name;

@SearchColumn(name="age",compare = CompareType.***EqualGreaterThan***)

**private** Integer ageMin;

@SearchColumn(name="age",compare = CompareType.***EqualLessThan***)

**private** Integer ageMax;

@SearchColumn()

**private** DepartmentFilter department;

}

*id: Tìm kiếm với id = {id}*

*name: Tìm kiếm tên có chứa câu {name}*

*ageMin: tìm kiếm tuổi trên {ageMin}*

*ageMax: Tìm kiếm tuổi dưới {ageMax}*

*department: Tìm kiếm bằng vị trí công việc*

@Data

**public** **class** DepartmentFilter {

@SearchColumn

**private** Integer id;

@SearchColumn(compare = CompareType.***HasContains***)

**private** String name;

}

*id: Tìm kiếm với id = {id}*

*name: Tìm kiếm với tên có chứa câu {name}*

## Hibernate

Dựa vào filter trên ta có thể tạo query bằng cách

CriteriaBuilder cb= entityManager.getCriteriaBuilder();

CriteriaQuery< Employee > query= cb.createQuery(Employee.**class**);

Root< Employee > root = query.from(Employee.**class**);

EmployeeFilter filter = new EmployeeFilter();

filter.setName(“john”);

filter.setAgeMin(25);

filter.setAgeMax(30)

CustomSpecification<Employee> specification = new CustomSpecification<>(filter);

query.where(specification.toPredicate(root,query,cb));

List<Employee> results = entityManager.createQuery(query).getResultList();

## JPA

Hoặc trong jpa ta có thể viết

@Repository

**public** **interface** EmployeeRepository **extends** JpaRepository<Employee, Integer>, JpaSpecificationExecutor<Employee>{

}

EmployeeFilter filter = new EmployeeFilter();

filter.setName(“john”);

filter.setAgeMin(25);

filter.setAgeMax(30);

CustomSpecification<Employee> specification=**new** CustomSpecification<Employee>(filter);

List<Employee> list = employeeRepository.findAll(specification);