<http://www.php.cn/java-article-360939.html>

spring Data JPA大大的简化了我们持久层的开发，但是实际应用中，我们还是需要动态查询的。

比如，前端有多个条件，这些条件很多都是可选的，那么后端的SQL，就应该是可以定制的，在使用hibernate的时候，可以通过判断条件来拼接SQL(HQL)，当然，Spring Data JPA在简化我们开发的同时，也是提供了支持的。

通过实现Criteria二实现的动态查询，需要我们的Repo接口继承JpaSpecificationExecutor接口，这是个泛型接口。

然后查询的时候，传入动态查询参数，分页参数等即可。

使用起来很简单，不过为了知其所以然，先介绍一下Criteria API。

**Criteria API**

如果编译器能够对查询执行语法正确性检查，那么对于 Java 对象而言该查询就是类型安全的。Java™Persistence API (JPA) 的 2.0 版本引入了 Criteria API，这个 API 首次将类型安全查询引入到 Java 应用程序中，并为在运行时动态地构造查询提供一种机制。本文介绍如何使用 Criteria API 和与之密切相关的 Metamodel API 编写动态的类型安全查询。

在使用Spring Data JPA的时候，只要我们的Repo层继承JpaSpecificationExecutor接口就可以使用Specification进行动态查询了，我们先看下JpaSpecificationExecutor接口：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public interface JpaSpecificationExecutor<T> {   T findOne(Specification<T> spec);   List<T> findAll(Specification<T> spec);   Page<T> findAll(Specification<T> spec, Pageable pageable);   List<T> findAll(Specification<T> spec, Sort sort);   long count(Specification<T> spec);  } |

可以看到提供了5个方法，方法的参数和返回值已经很明显的表达了其意图。其中的参数，Pageable和Sort应该是比较简单的，分别是分页参数和排序参数，而重点就是Specification参数，先看下这个接口的定义：

|  |  |
| --- | --- |
| 1  2  3 | public interface Specification<T> {   Predicate toPredicate(Root<T> root, CriteriaQuery<?> query, CriteriaBuilder cb);  } |

其中就一个方法，返回的是动态查询的数据结构。

|  |  |
| --- | --- |
| 1  2  3 | javax.persistence.criteria.Predicate toPredicate(javax.persistence.criteria.Root<T> root,  javax.persistence.criteria.CriteriaQuery<?> query,  javax.persistence.criteria.CriteriaBuilder cb); |

这里使用的都是Java EE中的规范，具体实现本人采用的是Hibernate，当然也可以选择其他实现了JPA规范的数据持久层框架。  
这里需要再次回过头看看Criteria API中的一些东西：

Criteria 查询是以元模型的概念为基础的，元模型是为具体持久化单元的受管实体定义的，这些实体可以是实体类，嵌入类或者映射的父类。

CriteriaQuery接口：代表一个specific的顶层查询对象，它包含着查询的各个部分，比如：select 、from、where、group by、order by等注意：CriteriaQuery对象只对实体类型或嵌入式类型的Criteria查询起作用

Root接口：代表Criteria查询的根对象，Criteria查询的查询根定义了实体类型，能为将来导航获得想要的结果，它与SQL查询中的FROM子句类似

1：Root实例是类型化的，且定义了查询的FROM子句中能够出现的类型。

2：查询根实例能通过传入一个实体类型给 AbstractQuery.from方法获得。

3：Criteria查询，可以有多个查询根。

4：AbstractQuery是CriteriaQuery 接口的父类，它提供得到查询根的方法。CriteriaBuilder接口：用来构建CritiaQuery的构建器对象Predicate：一个简单或复杂的谓词类型，其实就相当于条件或者是条件组合

其中支持的方法非常之强大，下面给出一个示例，大家可以参考一下，同样的，可以根据示例，自己可以写出更为复杂的查询：

Repo接口：

|  |  |
| --- | --- |
| 1  2 | public interface DevHREmpConstrastDao   extends JpaRepository<DevHREmpConstrast, Long>,JpaSpecificationExecutor<DevHREmpConstrast> |

查询实例1：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | /\*\*   \* 条件查询时动态组装条件   \*/  private Specification<DevHREmpConstrast> where(    final String corg,final String name,final String type,final String date,final String checker){   return new Specification<DevHREmpConstrast>() {    @Override    public Predicate toPredicate(Root<DevHREmpConstrast> root, CriteriaQuery<?> query, CriteriaBuilder cb) {     List<Predicate> predicates = new ArrayList<Predicate>();     //机构     if(corg!=null&&!corg.equals("")){      List<String> orgIds = organizationDao.findByName("%"+corg+"%");      if(orgIds.size()>0&&orgIds.size()<1000)       predicates.add(root.<String>get("confirmOrgNo").in(orgIds));//confirmOrgNo     }     //名字     if(name!=null&&!name.equals("")){      List<String> userIds = userDao.findByName(name);      if(userIds.size()>0&&userIds.size()<1000)//如果太多就不管了这个条件了       predicates.add(root.<String>get("hrUserName").in(userIds));     }     //类型     if(type!=null&&!type.equals(""))      predicates.add(cb.equal(root.<String>get("hrUpdateType"),type));     //日期     if(date!=null&&!date.equals("")){      //处理时间      SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");      Date startDate;      Date endDate;      try {       startDate = format.parse(date);      } catch (ParseException e) {       startDate = new Date(946656000000L);//2000 01 01      }      endDate = startDate;      Calendar calendar = Calendar.getInstance() ;      calendar.setTime(endDate);      calendar.add(Calendar.DATE, 1);      endDate = calendar.getTime();      calendar = null;      predicates.add(cb.between(root.<Date>get("insDate"),startDate,endDate));     }     //审核人     if(checker!=null&&!checker.equals("")){      List<String> userIds = userDao.findByName(checker);      if(userIds.size()>0&&userIds.size()<1000)//如果太多就不管了这个条件了       predicates.add(root.<String>get("confirmUserId").in(userIds));     }     return query.where(predicates.toArray(new Predicate[predicates.size()])).getRestriction();    }   };  } |

查询实例2：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | /\*\*   \* 条件查询时动态组装条件   \*/   private Specification<DevHREmpConstrast> where(     final String corg,final String name,final String type,final String date,final String checker){    return new Specification<DevHREmpConstrast>() {     @Override     public Predicate toPredicate(Root<DevHREmpConstrast> root, CriteriaQuery<?> query, CriteriaBuilder cb) {      List<Predicate> predicates = new ArrayList<Predicate>();      //机构      if(corg!=null&&!corg.equals("")){       List<String> orgIds = organizationDao.findByName("%"+corg+"%");       if(orgIds.size()>0&&orgIds.size()<1000)        predicates.add(root.<String>get("confirmOrgNo").in(orgIds));//confirmOrgNo      }      //名字      if(name!=null&&!name.equals("")){       List<String> userIds = userDao.findByName(name);       if(userIds.size()>0&&userIds.size()<1000)//如果太多就不管了这个条件了        predicates.add(root.<String>get("hrUserName").in(userIds));      }      //类型      if(type!=null&&!type.equals(""))       predicates.add(cb.equal(root.<String>get("hrUpdateType"),type));      //日期      if(date!=null&&!date.equals("")){       //处理时间       SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");       Date startDate;       Date endDate;       try {        startDate = format.parse(date);       } catch (ParseException e) {        startDate = new Date(946656000000L);//2000 01 01       }       endDate = startDate;       Calendar calendar = Calendar.getInstance() ;       calendar.setTime(endDate);       calendar.add(Calendar.DATE, 1);       endDate = calendar.getTime();       calendar = null;       predicates.add(cb.between(root.<Date>get("insDate"),startDate,endDate));      }      //审核人      if(checker!=null&&!checker.equals("")){       List<String> userIds = userDao.findByName(checker);       if(userIds.size()>0&&userIds.size()<1000)//如果太多就不管了这个条件了        predicates.add(root.<String>get("confirmUserId").in(userIds));      }      return query.where(predicates.toArray(new Predicate[predicates.size()])).getRestriction();     }    };   } |

然后调用dao层方法传入where()方法返回的参数即可。

以上就是JAVA Spring Data JPA中的动态查询代码详解的详细内容，更多请关注php中文网其它相关文章！