Jpa 列表 Specification 多条件查询

作者：zhangzeshan
原文链接：https://ld246.com/article/1602748716561
来源网站：链滴

前景

从php转java,发现JPA的持久层有点类似自己接触的php的ORM写法,于是用这个持久层框架去写了一统计后台,但是发现我错了,这个持久层我用的一点都不习惯,单纯局限在一些简单的增删改,之后在网上种查询,发现需要多继承一个Specification就可以实现一些复杂的查询

实体类

也就是对应一个表的结构

这里的注解就不再去赘述,只要知道这个是一个表

```
package com.center.entity.admin;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.*;
import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

/**
 * 统计表实体类
 *
 * @author Administrator
 */
@Entity
@Table(name = "t_cashout_count")
```

```
@EntityListeners(AuditingEntityListener.class)
@Setter
@Getter
@ToString
public class CashoutCount implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Column(name = "id", nullable = false, length = 11)
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    private Long id;//唯一id


    @Column(name = "cashoutcount")
    private Long cashoutCount;

    @Column(name = "cashoutsuccesscount")
    private Long cashoutSuccessCount;

    @Column(name = "cashoutfailcount")
    private Long cashoutFailCount;

    @Column(name = "cashoutprice", columnDefinition = "decimal(15,6)")
    private BigDecimal cashoutPrice;

    @Column(name = "cashoutsuccessprice", columnDefinition = "decimal(15,6)")
    private BigDecimal cashoutSuccessPrice;

    @Column(name = "cashoutfailprice", columnDefinition = "decimal(15,6)")
    private BigDecimal cashoutFailPrice;

    @Column(name = "type")
    private Byte type;

    @Column(name = "timetype")
    private Byte timeType;

    @Column(name = "source")
    private Byte source;

    @Column(name = "time")
    private Date createTime;

    @Column(name = "updatetime")
    private Date updateTime;

    @Transient
    private Date endTime;

    @Transient
```

```
    private String sourceName;
}
```

创建上述表对应的接口

这个接口就是我们文章的重点,继承基础的接口JpaRepository之外,

还要继承JpaSpecificationExecutor

```
package com.center.dao.admin;

import com.center.entity.admin.CashoutCount;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import org.springframework.stereotype.Repository;

@Repository
public interface CashoutCountDao extends JpaRepository<CashoutCount, Long>, JpaSpecific
tionExecutor<CashoutCount> {
}
```

创建sevice层,编写继承接口之后的代码

注释将写在代码上

```
package com.center.service.admin;

import com.center.bean.PageBean;
import com.center.common.enumtype.Constants;
import com.center.dao.admin.CashoutCountDao;
import com.center.entity.admin.CashoutCount;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;

import javax.persistence.criteria.Predicate;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Service
public class CashoutCountService {
    @Autowired
    private CashoutCountDao cashoutCountDao;

    /**
     * 日/月
     *
     * @param cashoutCount
     * @param pageBean  这个是分页 先不管
```

```
     * @return
     */
    public List<CashoutCount> findByDays(CashoutCount cashoutCount, PageBean<CashoutC
unt> pageBean) {
        //分页 每页7条 按照创建时间去做倒序
        Pageable pageable = PageRequest.of(pageBean.getCurrentPage() - 1, 7, Sort.Direction.D
SC, "createTime");
        //这个就是主代码块
        Specification<CashoutCount> tvSpecification = (Specification<CashoutCount>) (root, q
ery, cb) -> {
            List<Predicate> predicateList = new ArrayList<>();
            //相当于 where source = ?
            if (cashoutCount.getSource() != null) {
                predicateList.add(cb.equal(root.get("source"), cashoutCount.getSource()));
            }
            //相当于 where timeType  = ?
            if (cashoutCount.getTimeType() != null) {
                predicateList.add(cb.equal(root.get("timeType"), cashoutCount.getTimeType()));
            }
            //相当于 where type = ?
            if (cashoutCount.getType() != null) {
                predicateList.add(cb.equal(root.get("type"), cashoutCount.getType()));
            }
            //相当于 where createtime <= ?  (createTime是实体那边的属性 不要和表字段混淆)
            if (cashoutCount.getCreateTime() != null) {
                predicateList.add(cb.lessThanOrEqualTo(root.get("createTime").as(Date.class), casho
tCount.getCreateTime()));
            }
            Predicate[] predicates = new Predicate[predicateList.size()];
            return query.where(predicateList.toArray(predicates)).getRestriction();
        };
        Page<CashoutCount> findAll = cashoutCountDao.findAll(tvSpecification, pageable);
        return findAll.getContent();
    }
```

这样就能实现列表页的一些查询了