原 Struts2 + Spring + Hibernate 通用 Service 和 DAO

2013年10月11日 19:32:06 阅读数：6943

我在 Struts2 + Spring + Hibernate 项目开发中总结出了一个Service 和 DAO ，可以用于处理任何的pojo(bean)。使用这两个Service 和 DAO 可以极大地提高开发的效率，不必再分别针对不同的pojo编写对应的Service 和 DAO。内容如下：

DAO：

接口：BaseDao.java

```java
package dao;

import java.util.List;


/**
 * @author 雷霄骅
 * 对Object的DAO操作
 * 提供了通用的一些方法
 */

public interface BaseDao {
    public void save(Object object);
    public void delete(Object object);
    public void update(Object object);
    public Object ReadSingle(String targetName,String propertyName,Object value);
    public List<Object> ReadByProperty(String targetName,String propertyName,Object value);
    public List<Object> ReadAll(String targetName);
    public List<Object> ReadAllByOrder(String targetName,String propertyName,String order);
    public Object get(int id);
    public List<Object> ReadByPropertyList(String targetName,List<String> propertyName, List<Object> value);
    public Integer ReadCount(String targetName);
    public List<Object> ReadLimitedByOrder(String targetName, String propertyName,int num, String order);
}
```

实现：BaseDaoImpl.java

```java
package dao;

import java.sql.SQLException;
import java.util.List;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;




/**
 * @author 雷霄骅
 * HibernateTemplate提供非常多的常用方法来完成基本的操作，比如通常的增加、删除、修改、查询等操作，
 * Spring 2.0更增加对命名SQL查询的支持，也增加对分页的支持。大部分情况下，就可完成大多数DAO对象的CRUD操作。
 */
public class BaseDaoImpl extends HibernateDaoSupport implements BaseDao{

    @Override
    public void save(Object object) {
        getHibernateTemplate().save(object);
        //System.out.println("save "+object.toString());
    }

    @Override
    public void delete(Object object) {
        getHibernateTemplate().delete(object);
    }

    @Override
    public void update(Object object) {
        getHibernateTemplate().update(object);
        //改用saveOrUpdate，在评价的时候，第一次创建的时候Save，其他时候Update
        //getHibernateTemplate().saveOrUpdate(object);
        //System.out.println("update "+object.toString());
    }

    @SuppressWarnings("unchecked")
    @Override
    public Object ReadSingle(final String targetName,final String propertyName, final Object value) {
        // TODO Auto-generated method stub
```

```
        // TODO Auto-generated method stub
        return (Object) getHibernateTemplate().execute(new HibernateCallback() {
            /*doInHibernate()。session的创建和销毁，一切都在程序内部完成。*/
            public Object doInHibernate(Session session)
                    throws HibernateException, SQLException {
                String hql = "from "+targetName+" as "+targetName+" where "+targetName+"." + propertyName + "=:value";
                Query query = session.createQuery(hql);
                query.setParameter("value", value);
                return query.uniqueResult();
            }
        });
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<Object> ReadAll(String targetName) {
        // TODO Auto-generated method stub
        String hql="from "+targetName;
        return getHibernateTemplate().find(hql);
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<Object> ReadByProperty(final String targetName, final String propertyName,
            final Object value) {
        // TODO Auto-generated method stub
        return (List<Object>) getHibernateTemplate().execute(new HibernateCallback() {
            /*doInHibernate()。session的创建和销毁，一切都在程序内部完成。*/
            public Object doInHibernate(Session session)
                    throws HibernateException, SQLException {
                String hql = "from "+targetName+" as "+targetName+" where "+targetName+"." + propertyName + "=:value";
                Query query = session.createQuery(hql);
                query.setParameter("value", value);
                return query.list();
            }
        });
    }
    //比ReadByProperty简单很多
    @Override
    public Object get(int id) {
        // TODO Auto-generated method stub
        return getHibernateTemplate().get(Object.class, id);
    }

    @Override
    public List<Object> ReadByPropertyList(final String targetName,
            final List<String> propertyName, final List<Object> value) {
        // TODO Auto-generated method stub
        return (List<Object>) getHibernateTemplate().execute(new HibernateCallback() {
            /*doInHibernate()。session的创建和销毁，一切都在程序内部完成。*/
            public Object doInHibernate(Session session) throws HibernateException, SQLException {
                String hql = "from "+targetName+" as "+targetName;

                //------------------------------
                for(int i=0;i<propertyName.size();i++){
                    String propertynametemp= propertyName.get(i);
                    Object propertyvaluetemp= value.get(i);
                    if(propertynametemp!=null){
                     if(i==0){
                      hql=hql+" where "+targetName+"." + propertynametemp + "=" + propertyvaluetemp +" ";
                     }else{
                         hql=hql+" and "+targetName+"." +propertynametemp + "=" + propertyvaluetemp +" ";
                     }
                    }
                }
                //------------------------------
                    Query query = session.createQuery(hql);
                //当返回的数据不是一条的时候，不用uniqueresult()，而用list()
                return query.list();
            }
        });
    }
//这里在Hibernate2.0之前版本list.get(0)返回的是Integer类型.
//但是在Hibernate3.0以后版本list.get(0)返回的是Long类型.
//所以在这里不可以由Long型强转成Integer类型.
//Integer属于不可更改类型，而且Long和Integer没有任何继承关系，当然不能这样转换。
    @Override
    public Integer ReadCount(final String targetName) {
        // TODO Auto-generated method stub
        return (Integer) getHibernateTemplate().execute(new HibernateCallback() {
            /*doInHibernate()。session的创建和销毁，一切都在程序内部完成。*/
            public Object doInHibernate(Session session)
                    throws HibernateException, SQLException {
                String hql = "select count(*) from "+targetName;
                //System.out.println(hql);
                //注:java.lang.Number是Integer,Long的父类.
                return ((Number)session.createQuery(hql).iterate().next()).intValue();
            }
        });
    }

    @Override
```

```
    public List<Object> ReadLimitedByOrder(final String targetName,
            final String propertyName, final int num, final String order) {
        // TODO Auto-generated method stub
        return (List<Object>) getHibernateTemplate().execute(new HibernateCallback() {
            /*doInHibernate()。session的创建和销毁，一切都在程序内部完成。*/
            public Object doInHibernate(Session session) throws HibernateException, SQLException {
                String hql ="from "+targetName+" as "+targetName+ " order by "+targetName+"." + propertyName+ " " + order;
                Query query = session.createQuery(hql);
                query.setMaxResults(num);
                //当返回的数据不是一条的时候，不用uniqueresult()，而用list()
                return query.list();
            }
        });
    }

    @Override
    public List<Object> ReadAllByOrder(String targetName, String propertyName,
            String order) {
        // TODO Auto-generated method stub
        String hql="from "+targetName+" as "+targetName+ " order by "+targetName+"." + propertyName+ " " + order;
        return getHibernateTemplate().find(hql);
    }


}
```

Service：

接口：BaseService.java

```java
package service;

import java.util.List;

/**
 * @author 雷霄骅
 * 对Object的Service
 * 提供了一些通用的方法
 */
public interface BaseService {
     public void save(Object object);
     public void update(Object object);
     public void delete(Object object);
     public Object ReadByID(String targetName,int id);
     @SuppressWarnings("rawtypes")
     public List ReadAll(String targetName);
     public List ReadAllByOrder(String targetName,String propertyName,String order);
     @SuppressWarnings("rawtypes")
     public List ReadByProperty(String targetName,String propertyName,Object propertyValue);
     public List ReadByPropertyList(String targetName,List<String> propertyName,List<Object> propertyValue);
     public List ReadLimitedByOrder(String targetName,String propertyName,int num,String order);
     public Object ReadSingle(String targetName,String propertyName,Object propertyValue);
     public int ReadCount(String targetName);

}
```

实现：BaseServiceImpl.java

```java
package service;

import java.util.ArrayList;
import java.util.List;

import dao.BaseDao;
/**
 * @author 雷霄骅
 * 对Object的Service
 * 提供了一些通用的方法
 */
public class BaseServiceImpl implements BaseService {

    private BaseDao baseDao;
    @Override
    public void save(Object object) {
        // TODO Auto-generated method stub
        baseDao.save(object);
    }

    @Override
    public void update(Object object) {
        // TODO Auto-generated method stub
        baseDao.update(object);
    }
```

```
26.
27.     @Override
28.     public void delete(Object object) {
29.         // TODO Auto-generated method stub
30.         baseDao.delete(object);
31.     }
32.
33.     @Override
34.     public Object ReadByID(String targetName,int id) {
35.         // TODO Auto-generated method stub
36.         return baseDao.ReadSingle(targetName,"id", id);
37.     }
38.
39.     @SuppressWarnings("rawtypes")
40.     @Override
41.     public List ReadAll(String targetName) {
42.         // TODO Auto-generated method stub
43.         return baseDao.ReadAll(targetName);
44.     }
45.
46.     @SuppressWarnings("rawtypes")
47.     @Override
48.     public List ReadAllByOrder(String targetName,String propertyName,String order) {
49.         // TODO Auto-generated method stub
50.         return baseDao.ReadAllByOrder(targetName,propertyName,order);
51.     }
52.
53.     public BaseDao getBaseDao() {
54.         return baseDao;
55.     }
56.
57.     public void setBaseDao(BaseDao baseDao) {
58.         this.baseDao = baseDao;
59.     }
60.
61.     @Override
62.     public List ReadByProperty(String targetName, String propertyName,
63.             Object propertyValue) {
64.         // TODO Auto-generated method stub
65.         return baseDao.ReadByProperty(targetName, propertyName, propertyValue);
66.     }
67.
68.     @Override
69.     public Object ReadSingle(String targetName, String propertyName,
70.             Object propertyValue) {
71.         // TODO Auto-generated method stub
72.         return baseDao.ReadSingle(targetName, propertyName, propertyValue);
73.     }
74.
75.     @Override
76.     public int ReadCount(String targetName) {
77.         // TODO Auto-generated method stub
78.         return baseDao.ReadCount(targetName);
79.     }
80.
81.     @Override
82.     public List ReadLimitedByOrder(String targetName, String propertyName,
83.             int num, String order) {
84.         return baseDao.ReadLimitedByOrder(targetName,propertyName,num,order);
85.     }
86.
87.     @Override
88.     public List ReadByPropertyList(String targetName,
89.             List<String> propertyName, List<Object> propertyValue) {
90.         // TODO Auto-generated method stub
91.         return baseDao.ReadByPropertyList(targetName,propertyName,propertyValue);
92.     }
93.
94. }
```

这样，在Action层调用方法的时候，可以直接调用BaseService相应的方法完成操作。

举一个例子：

有这么一个名字叫Blog的pojo：

```
[java]
1.  package bean;
2.
3.  import java.sql.Timestamp;
4.  import javax.persistence.Column;
5.  import javax.persistence.Entity;
6.  import javax.persistence.FetchType;
7.  import javax.persistence.GeneratedValue;
8.  import static javax.persistence.GenerationType.IDENTITY;
9.  import javax.persistence.Id;
10. import javax.persistence.JoinColumn;
11. import javax.persistence.ManyToOne;
```

```java
import javax.persistence.Table;

/**
 * Blog entity. @author MyEclipse Persistence Tools
 */
@Entity
@Table(name = "blog", catalog = "vqe")
public class Blog implements java.io.Serializable {

    // Fields

    private Integer id;
    private Admin admin;
    private String title;
    private Timestamp modifytime;
    private String content;

    // Constructors

    /** default constructor */
    public Blog() {
    }

    /** full constructor */
    public Blog(Admin admin, String title, Timestamp modifytime, String content) {
        this.admin = admin;
        this.title = title;
        this.modifytime = modifytime;
        this.content = content;
    }

    // Property accessors
    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "id", unique = true, nullable = false)
    public Integer getId() {
        return this.id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "adminid")
    public Admin getAdmin() {
        return this.admin;
    }

    public void setAdmin(Admin admin) {
        this.admin = admin;
    }

    @Column(name = "title", length = 200)
    public String getTitle() {
        return this.title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @Column(name = "modifytime", length = 19)
    public Timestamp getModifytime() {
        return this.modifytime;
    }

    public void setModifytime(Timestamp modifytime) {
        this.modifytime = modifytime;
    }

    @Column(name = "content", length = 10000)
    public String getContent() {
        return this.content;
    }

    public void setContent(String content) {
        this.content = content;
    }

}
```

该类代表博客的一篇文章。

在Action层只需调用BaeService对应的方法就能完成相应的操作。换句话说，只要把pojo的类的名字当一个字符串传递给ReadByID这种的函数，就可以实现相应的功能。

```java
//根据ID读取：
Blog blog=(Blog) baseService.ReadByID("Blog", blogid);
//添加：
baseService.save(blog);
//修改：
baseService.update(blog);
//删除：
baseService.delete(blog);
//读取所有（根据时间降序）
List<Blog> resultblog=baseService.ReadAllByOrder("Blog","modifytime","desc");
//读取num条（根据时间降序）
List<Blog> resultblog=baseService.ReadLimitedByOrder("Blog","modifytime",num,"desc");
```

完整的实现Blog（博客）的增删改查的Action示例：

```java
package action;


import java.sql.Timestamp;
import java.util.Date;
import java.util.List;
import java.util.Map;

import service.BaseService;


import bean.Admin;
import bean.Blog;

import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
/**
 * @author 雷霄骅
 * Action
 */
public class BlogAct extends ActionSupport {
    private int blogid;
    private int num;
    private Blog blog;
    private List<Blog> resultblog;
    private BaseService baseService;

    public int getBlogid() {
        return blogid;
    }

    public void setBlogid(int blogid) {
        this.blogid = blogid;
    }

    public Blog getBlog() {
        return blog;
    }

    public void setBlog(Blog blog) {
        this.blog = blog;
    }

    public BaseService getBaseService() {
        return baseService;
    }

    public void setBaseService(BaseService baseService) {
        this.baseService = baseService;
    }

    public List<Blog> getResultblog() {
        return resultblog;
    }

    public void setResultblog(List<Blog> resultblog) {
        this.resultblog = resultblog;
    }

    public int getNum() {
        return num;
    }

    public void setNum(int num) {
        this.num = num;
    }

    public String Add(){
        try{
            //------------------------------
            ActionContext context = ActionContext.getContext();
```

```
            Map sessionMap = context.getSession();
            Admin admin=(Admin)sessionMap.get("admin");
            //------------------------------
            blog.setModifytime( new Timestamp(new Date().getTime()));
            blog.setAdmin(admin);
            baseService.save(blog);
            return SUCCESS;
        }
        catch(Exception ex){
            ex.printStackTrace();
            return ERROR;
        }
    }

    public String Delete(){
        try{
            blog=(Blog) baseService.ReadByID("Blog", blogid);
            baseService.delete(blog);
            return SUCCESS;
        }
        catch(Exception ex){
            ex.printStackTrace();
            return ERROR;
        }
    }

    public String Read(){
        try{
            blog=(Blog) baseService.ReadByID("Blog", blogid);
            return SUCCESS;
        }
        catch(Exception ex){
            ex.printStackTrace();
            return ERROR;
        }
    }

    public String Update(){
        try{
            //------------------------------
            ActionContext context = ActionContext.getContext();
            Map sessionMap = context.getSession();
            Admin admin=(Admin)sessionMap.get("admin");
            //------------------------------
            blog.setModifytime( new Timestamp(new Date().getTime()));
            blog.setAdmin(admin);
            baseService.update(blog);
            return SUCCESS;
        }
        catch(Exception ex){
            ex.printStackTrace();
            return ERROR;
        }
    }

    public String ReadAll(){
        try{
            resultblog=baseService.ReadAllByOrder("Blog","modifytime","desc");
            return SUCCESS;
        }
        catch(Exception ex){
            ex.printStackTrace();
            return ERROR;
        }
    }

    public String ReadLimitedByOrder(){
        try{
            resultblog=baseService.ReadLimitedByOrder("Blog","modifytime",num,"desc");
            return SUCCESS;
        }
        catch(Exception ex){
            ex.printStackTrace();
            return ERROR;
        }
    }
}
```


文章标签： hibernate struts2 spring entity dao

个人分类： J2EE