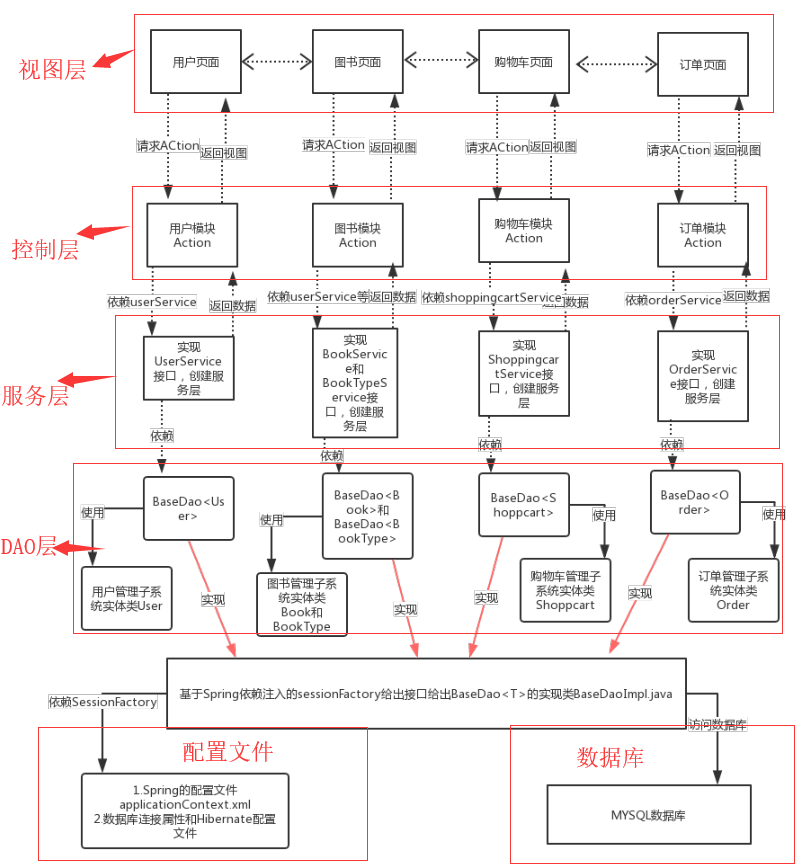
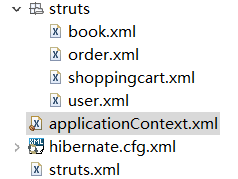
Struts2+Hibernate+Spring设计网上图书系统

SSH设计思想



1整合Spring配置文件



1. applicationContext.xml主要代码

<bean id=*"sessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

<property name=*"configLocation"* value=*"classpath:hibernate.cfg.xml"*/>

<!-- 自动扫描hibernate类文件 -->

<property name=*"packagesToScan"*>

<list>

<value>com.java1234.entity</value>

</list>

</property>

</bean>

<!-- 配置事务切面 -->

<aop:config>

<aop:pointcut id=*"serviceOperation"*

expression=*"execution(\* com.java1234.service.\*.\*(..))"* />

<aop:advisor advice-ref=*"txAdvice"* pointcut-ref=*"serviceOperation"* />

</aop:config>

<!-- 自动加载构建bean -->

<context:component-scan base-package=*"com.java1234"* />

2）struts.xml主要代码

<include file="struts/\*.xml"></include>

1. hibernate.cfg,xml主要代码

<session-factory>

<!--方言-->

<property name=*"dialect"*>org.hibernate.dialect.SQLServerDialect</property>

<!-- 显示sql语句 -->

<property name=*"show\_sql"*>true</property>

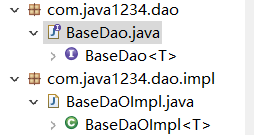
<!-- 自动更新 -->

<property name=*"hbm2ddl.auto"*>update</property>

<property name=*"hibernate.connection.autocommit"*>true</property>

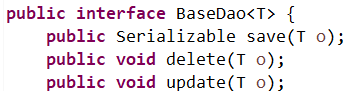
</session-factory>

1. DAO层（采用Hibernate框架实现对数据库的访问）



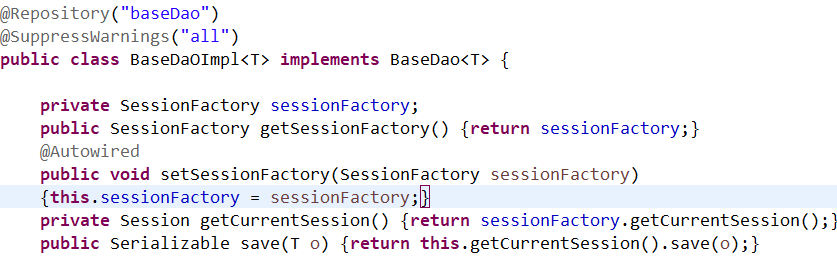
BaseDao 接口类

部分代码

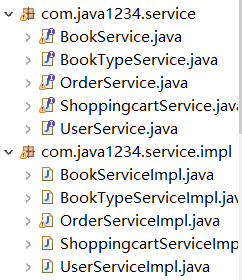


BaseDaOImpl实现类（使用Spring依赖注入的sessionFactory对象）

部分代码

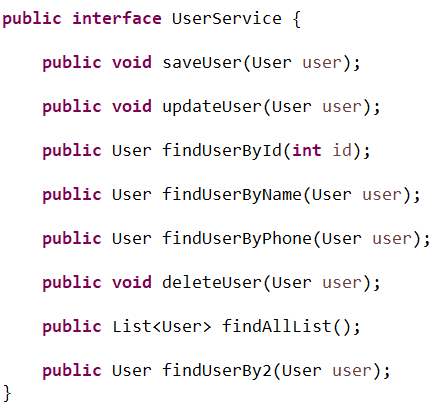


1. 业务逻辑层（Service-服务层）：位于数据访问层与控制层之间，承上启下，面向接口。



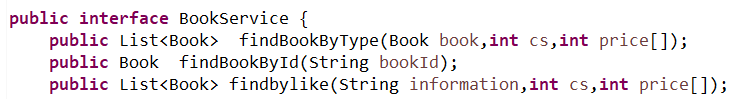
图中组件

用户管理子系统服务层接口UserService.java

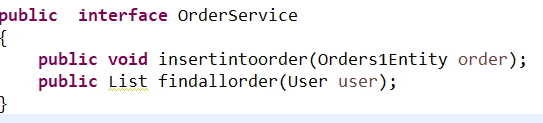


、

图书管理子系统服务层接口BookService.java



订单管理子系统服务层接口OrderService.java



购物车管理子系统服务层接口ShoppingcartService.java



用户管理服务类UserServiceImpl.java,依赖baseDao<User>,并实现BaseDaOImpl的接口

@Service("userService")

**public** **class** UserServiceImpl **implements** UserService{

@Resource

**private** BaseDao<User> baseDao;

@Override

**public** **void** saveUser(User user) {

baseDao.save(user);}

@Override

**public** **void** updateUser(User user) {

// **TODO** Auto-generated method stub

baseDao.update(user);

}

@Override

**public** User findUserById(**int** id) {

**return** baseDao.get(User.**class**, id);

}

@Override

**public** **void** deleteUser(User user) {

baseDao.delete(user);

}

@Override

**public** List<User> findAllList() {

**return** baseDao.find("from User");

}

@Override

**public** User findUserBy2(User user) {

**return** baseDao.get("from User u where u.user\_name=? and u.password=?", **new** Object[]{user.getUser\_name(),user.getPassword()});

}

**public** User findUserByPhone(User user)

{ **return** baseDao.get("from User u where u.user\_name=? and u.telephone=?", **new** Object[]{user.getUser\_name(),user.getTelephone()});

}

**public** User findUserByName(User user)

{ **return** baseDao.get("from User u where u.user\_name=?", **new** Object[]{user.getUser\_name()});

}

}

图书管理服务类口BookServiceImpl.java，依赖baseDao<Book>,并实现BaseDaOImpl的接口

@Service("bookService")

**public** **class** BookServiceImpl **implements** BookService {

@Resource

**private** BaseDao<Book> baseDao;

@Override

**public** List<Book> findBookByType(Book book,**int** cs,**int** price[]) {

List<Book> list = **null**;

String hql =**null**;

**if**(cs==0)

hql = "from Book b where b.type = '"+book.getType()+"'";

**else** **if**(cs==1) //升序

hql = "from Book b where b.type = '"+book.getType()+"' order by b.price asc";

**else** **if**(cs==2) //降序

hql = "from Book b where b.type = '"+book.getType()+"' order by b.price desc";

**else** //区间

hql = "from Book b where b.type = '"+book.getType()+"' and b.price between '"+price[0]+"' and '"+price[1]+"'";

String param[] = {};

list = baseDao.find(hql,param);

System.***out***.println("结束");

**return** list;

}

**public** List<Book> findbylike(String information,**int** cs,**int** price[]){

List <Book> list = **null**;

// String hql = "from Book b ";

String hql = "from Book b ";

String Field[] = {"b.bookName","b.type","b.discription"};

list = baseDao.findByFields(hql,Field,information,cs,price);

**return** list;

}

@Override

**public** Book findBookById(String bookId) {

String hql = "from Book b where b.bookId=?";

String param[] = {bookId};

Book book=baseDao.get(hql, param);

// **TODO** Auto-generated method stub

**return** book;

}

}

订单管理服务类OrderServiceImpl.java, 依赖baseDao<Order>,并实现BaseDaOImpl的接口

@Service("OrderService")

**public** **class** OrderServiceImpl **implements** OrderService{

@Resource

**private** BaseDao<Orders1Entity> baseDao;

@Override

**public** **void** insertintoorder(Orders1Entity order) {

baseDao.save(order);

}

@Override

**public** List findallorder(User user) {

String hql = "from Orders1Entity o where o.userId = ?";

String param[] = {user.getUser\_id()};

List list = baseDao.find(hql,param);

**return** list;

}

}

购物车管理服务类ShoppingcartServiceImple.java, 依赖baseDao<Shoppingcart>,并实现BaseDaOImpl的接口

@Service("ShoppingcartService")

**public** **class** ShoppingcartServiceImpl **implements** ShoppingcartService{

@Resource

**private** BaseDao<Shoppingcart> baseDao;

@Resource

**private** BaseDao<Book> baseDao1;

**public** List<Shoppingcart> Findlist(String userId){ //通过用户ID获取购物车表

List<Shoppingcart> list = **null**;

String hql = "from Shoppingcart s where s.user\_id = ?";

String param[] = {userId};

list = baseDao.find(hql,param);

**return** list;

}

//通过userid和bookid获得一个shoppingcart对象

**public** Shoppingcart findshoppingcart(String user\_id,String book\_id)

{

String hql = "from Shoppingcart s where s.user\_id = ? and book\_id=?";

String param[] = {user\_id,book\_id};

**return** baseDao.get(hql, param);

}

//删除一个shoppingcat记录

**public** **boolean** detelebook(Shoppingcart shoppingcart)

{

**try**{

baseDao.delete(shoppingcart);

**return** **true**;

}

**catch**(Exception e)

{

e.printStackTrace();}

**return** **false**;

}

//插入到购物车

**public** **boolean** insert(Shoppingcart shoppingcart){

**try**{

Shoppingcart sp1=**null**;

String hql="from Shoppingcart s where s.user\_id =? and s.book\_id =?";

**if**(baseDao.get(hql, **new** Object[]{shoppingcart.getUser\_id(),shoppingcart.getBook\_id()})!=**null**)

{

// System.out.println("2");

sp1=baseDao.get(hql, **new** Object[]{shoppingcart.getUser\_id(),shoppingcart.getBook\_id()});

sp1.setCount(shoppingcart.getCount()+sp1.getCount());

baseDao.save(sp1);

// System.out.println("1");

}

**else**

baseDao.save(shoppingcart);

**return** **true**;

}**catch**(Exception e){

e.printStackTrace();

**return** **false**;

}

}

@Override

**public** List<Book> findbyid(List<Shoppingcart> list){ //通过购物车列表获取图书表

ArrayList<Book> list1=**new** ArrayList<Book>();

String hql = "from Book b where b.bookId =?";

**try** { **for** (**int** i = 0; i < list.size(); i++){

String param[] = {list.get(i).getBook\_id()};

list1.add(i,baseDao1.get(hql,param));

}

}

**catch**(Exception e)

{

e.printStackTrace();

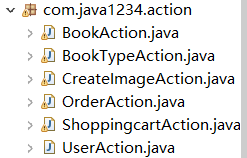
}

**return** list1;

}

}

4.控制层，由struts2的Action实现该层的功能



组件

用户模块Action类（登录包括发送短信验证码，注册，找回密码，修改资料等）

UserAction.java

图书模块Action类（查询显示等）

BookAction.java

图书类型Action类（显示图书类型，分类显示等）

BookTypeAction.java

生成验证码图片Action类

CreateImageAction.java

订单类型Action类（增加订单，显示订单等）

OrderAction.java

购物车类型Action类（增加到购物车，显示购物车删除等）

ShoppingcartAction.java

5.视图层，用Jsp,Struts标签，Js和HTML等技术设计的页面

