所需架包   
aopalliance.jar   
asm-1.5.3.jar   
aspectjrt.jar   
aspectjweaver.jar   
c3p0-0.9.1.jar   
cglib-2.1\_3.jar   
com.springsource.org.apache.commons.dbcp-1.2.2.osgi.jar   
com.springsource.org.apache.commons.logging-1.1.1.jar   
com.springsource.org.apache.commons.pool-1.5.3.jar   
commons-fileupload-1.2.1.jar   
commons-io-1.3.2.jar   
freemarker-2.3.15.jar   
junit-3.8.1.jar   
log4j-1.2.15.jar   
mybatis-3.0.3-SNAPSHOT.jar   
mybatis-3.0.4.jar   
mybatis-spring-1.0.0-RC2.jar   
ognl-2.7.3.jar   
ojdbc14.jar   
org.springframework.aop-3.0.4.RELEASE.jar   
org.springframework.asm-3.0.4.RELEASE.jar   
org.springframework.context-3.0.4.RELEASE.jar   
org.springframework.expression-3.0.4.RELEASE.jar   
org.springframework.test-3.0.4.RELEASE.jar   
org.springframework.web-3.0.4.RELEASE.jar   
slf4j-api-1.5.10.jar   
slf4j-log4j12-1.5.10.jar   
spring-beans-3.0.4.RELEASE.jar   
spring-core-3.0.4.RELEASE.jar   
spring-jdbc-3.0.4.RELEASE.jar   
spring-tx-3.0.4.RELEASE.jar   
struts2-convention-plugin-2.1.8.1.jar   
struts2-core-2.1.8.1.jar   
struts2-spring-plugin-2.1.8.1.jar   
struts-getterAndSetter-plugin.jar   
xwork-core-2.1.6.jar   
  
  
对应的配置文件   
applicationContext.xml:   
<beans xmlns="http://www.springframework.org/schema/beans"   
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jee="http://www.springframework.org/schema/jee"   
xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context"   
xmlns:aop="http://www.springframework.org/schema/aop"   
xsi:schemaLocation="http://www.springframework.org/schema/beans   
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd   
http://www.springframework.org/schema/tx   
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd   
http://www.springframework.org/schema/jee   
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd   
http://www.springframework.org/schema/aop   
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd   
http://www.springframework.org/schema/context   
http://www.springframework.org/schema/context/spring-context-2.5.xsd"   
default-autowire="byName" default-lazy-init="true">   
  
<!-- 定义受环境影响易变的变量 -->   
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">   
<property name="locations">   
<list>   
<!-- 标准配置 -->   
<value>classpath:com/medbri/mss/config/jdbc.properties</value>   
</list>   
</property>   
</bean>   
  
<!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->   
<context:component-scan base-package="com.medbri.mss" />   
  
<!-- 数据源配置,使用应用内的DBCP数据库连接池 -->   
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">   
<!-- Connection Info -->   
<property name="driverClassName" value="${jdbc.driver}" />   
<property name="url" value="${jdbc.url}" />   
<property name="username" value="${jdbc.username}" />   
<property name="password" value="${jdbc.password}" />   
  
<!-- Connection Pooling Info -->   
<property name="initialSize" value="5" />   
<property name="maxActive" value="100" />   
<property name="maxIdle" value="30" />   
<property name="maxWait" value="500" />   
<property name="defaultAutoCommit" value="false" />   
</bean>   
  
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">   
<property name="configLocation" value="classpath:com/medbri/mss/config/mybatis-config.xml" />   
<property name="dataSource" ref="dataSource" />   
</bean>   
  
<!-- 事务管理器配置,单数据源事务 -->   
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">   
<property name="dataSource" ref="dataSource" />   
</bean>   
  
<bean id="iUsers" class="org.mybatis.spring.MapperFactoryBean">   
<property name="sqlSessionFactory" ref="sqlSessionFactory" />   
<property name="mapperInterface" value="com.medbri.mss.dao.IUsers" />   
</bean>   
  
<!-- 使用annotation定义事务 -->   
<!-- <tx:annotation-driven transaction-manager="transactionManager" proxy-target-class="true"/> -->   
  
<aop:config proxy-target-class="true">   
<aop:advisor pointcut="execution(\* com.medbri.mss.service..\*Service.\*(..))"   
advice-ref="txAdvice" />   
</aop:config>   
  
<tx:advice id="txAdvice">   
<tx:attributes>   
<tx:method name="get\*" read-only="true" />   
<tx:method name="find\*" read-only="true" />   
<tx:method name="query\*" read-only="true" />   
<tx:method name="is\*" read-only="true" />   
<tx:method name="\*" propagation="REQUIRED" />   
</tx:attributes>   
</tx:advice>   
</beans>   
  
struts.xml   
<?xml version="1.0" encoding="GBK" ?>   
<!DOCTYPE struts PUBLIC   
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"   
    "http://struts.apache.org/dtds/struts-2.0.dtd">   
<struts>   
<!-- 设置 编码 -->   
<constant name="struts.i18n.encoding" value="GBK" />   
<!-- 与spring集成 -->   
<constant name="struts.objectFactory" value="spring" />   
<!-- 指定默认的父包 -->   
<constant name="struts.convention.default.parent.package"   
value="medbri-default" />   
<!-- 设置convention插件默认的根包 -->   
<constant name="struts.convention.package.locators" value="web" />   
<!-- 搜索此配置下的所有包 -->   
<constant name="struts.convention.package.locators.medbriPackage"   
value="com.medbri.mss" />   
<!-- 继承convention-default包，定义一个默认的拦截器，根据需要还可扩展 -->   
<package name="medbri-default" extends="convention-default">   
<interceptors>   
<interceptor-stack name="medbriStack">   
<interceptor-ref name="paramsPrepareParamsStack"></interceptor-ref>   
</interceptor-stack>   
</interceptors>   
<default-interceptor-ref name="medbriStack" />   
<global-results>   
<result name="suc">/common/sucess.jsp</result>   
<result name="error">/common/error.jsp</result>   
</global-results>   
</package>   
</struts>   
  
  
mybatis-config.xml   
<?xml version="1.0" encoding="UTF-8" ?>   
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">   
<configuration>   
<typeAliases>   
<typeAlias alias="Users" type="com.medbri.mss.model.Users" />   
</typeAliases>   
<mappers>   
<mapper resource="com/medbri/mss/dao/IUsers.xml" />   
</mappers>   
</configuration>   
  
代码   
action   
@Results({   
@Result(name = "userListWindow", type = UserAction.DISPATHCER, location = "/userList.jsp"),   
@Result(name = "userUpdateWindow", type = UserAction.DISPATHCER, location = "/updateUser.jsp")   
})   
public class UserAction extends BaseAction<Users> {   
@Autowired   
private IUserService userService;   
private Users users;   
  
public Users getUsers() {   
return users;   
}   
  
public void setUsers(Users users) {   
this.users = users;   
}   
  
/\*\*   
\*   
\*/   
private static final long serialVersionUID = 1L;   
  
/\*\*   
\* 用户添加   
\*   
\* @return 返回转向信息   
\* @throws Exception   
\*             采用struts2异常处理抛出所有异常   
\*/   
public String addUser() throws Exception {   
userService.addUser(users);   
getRequest().setAttribute("url","user!queryPageUser.action");   
return "suc";   
  
}   
  
  
/\*\*   
\* 用户修改   
\*   
\* @return 返回转向信息   
\* @throws Exception   
\*             采用struts2异常处理抛出所有异常   
\*/   
public String updateUser() throws Exception {   
userService.updateUser(users);   
getRequest().setAttribute("url","user!queryPageUser.action");   
return "suc";   
  
}   
  
/\*\*   
\* 用户删除   
\*   
\* @return 返回转向信息   
\* @throws Exception   
\*             采用struts2异常处理抛出所有异常   
\*/   
public String deleteUser() throws Exception {   
userService.deleteUser(id);   
getRequest().setAttribute("url","user!queryPageUser.action");   
return "suc";   
}   
  
/\*\*   
\* 用户修改查询   
\*   
\* @return 返回转向信息   
\* @throws Exception   
\*             采用struts2异常处理抛出所有异常   
\*/   
public String findById() throws Exception {   
users= userService.findById(id);   
return "userUpdateWindow";   
}   
  
/\*\*   
\* 用户分页查询   
\*   
\* @return 返回转向信息   
\* @throws Exception   
\*             采用struts2异常处理抛出所有异常   
\*/   
public String queryPageUser() throws Exception {   
pagenation= userService.queryPage(pageNum,PAGE\_SIZE);   
return "userListWindow";   
}   
  
}   
  
  
dao   
public interface IUsers {   
/\*\*   
\* 添加用户   
\*   
\* @param users   
\*            用户model   
\*/   
public void addUser(Users users);   
  
/\*\*   
\* 删除用户   
\*   
\* @param userId   
\*            用户id   
\*/   
public void deleteUser(Long userId);   
  
/\*\*   
\* 分页时获取总记录数   
\* @param map 查询条件   
\* @return 返回总记录数   
\*/   
public int getCount(Map map);   
  
/\*\*   
\* 用户分页查询   
\* @param map 查询条件   
\* @return 返回用户列表   
\*/   
public List<Users> queryPage(Map map);   
  
/\*\*   
\* 修改用户信息   
\* @param users   
\*/   
    public void updateUser(Users users);   
      
    /\*\*   
     \* 根据id查找用户   
     \* @param id 用户id   
     \* @return 返回指定用户   
     \*/   
    public Users findById(Long id);   
}   
  
mybatis配置文件   
<?xml version="1.0" encoding="UTF-8" ?>   
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">   
<mapper namespace="com.medbri.mss.dao.IUsers">   
  
<resultMap type="Users" id="sUsersResultMap">   
<id property="userId" column="userId" />   
<result property="userName" column="userName" />   
<result property="userPassword" column="userPassword" />   
</resultMap>   
<insert id="addUser" parameterType="Users">   
insert into users values(seq\_user\_id.nextval,#{userName},#{userPassword})   
</insert>   
  
<delete id="deleteUser" parameterType="Long">   
delete from users where userId=#{userId}   
</delete>   
<update id="updateUser" parameterType="Users">   
update users set userName=#{userName},userPassword=#{userPassword} where   
userId=#{userId}   
    </update>   
  
<select id="getCount" parameterType="map" resultType="int">   
select count(1)from users   
    </select>   
<select id="queryPage" resultType="Users" parameterType="map">   
select \* from(select a.\*,rownum r from(select   
t.userId,t.userName,t.userPassword from users t)   
a ) where r &gt; #{startRow} and rownum &lt;= #{pageSize}   
       
    </select>   
<select id="findById" resultType="Users" parameterType="Long">   
select t.userId,t.userName,t.userPassword from users t where   
t.userId=#{userId}   
    </select>   
</mapper>   
  
  
service   
  
service接口省略（版面有限）   
  
@Service   
public class UserServiceImpl implements IUserService {   
@Autowired   
private IUsers iUsers;   
  
/\*\*   
\* 用户管理添加   
\*   
\* @param users   
\*            用户实体   
\*/   
public void addUser(Users users) {   
iUsers.addUser(users);   
}   
  
/\*\*   
\* 删除用户   
\*   
\* @param userId   
\*            用户id   
\*/   
public void deleteUser(Long userId) {   
iUsers.deleteUser(userId);   
}   
  
/\*\*   
\* 分页时获取总记录数   
\* @param map 查询条件   
\* @return 返回总记录数   
\*/   
public int getCount(Map map){   
return iUsers.getCount(map);   
}   
  
/\*\*   
\* 用户分页查询   
\* @param map 查询条件   
\* @return 返回用户列表   
\*/   
public  Pagenation queryPage(int pageNum,int pageSize){   
Map map=new HashMap();   
int count=iUsers.getCount(map);   
Pagenation p = new Pagenation(pageSize,pageNum, count);   
map.put("startRow",p.getStartRow());   
map.put("pageSize",pageSize);   
List<Users>list=iUsers.queryPage(map);   
p.setList(list);   
return p;   
}   
  
/\*\*   
\* 修改用户信息   
\* @param users   
\*/   
    public void updateUser(Users users){   
    iUsers.updateUser(users);   
    }   
      
    /\*\*   
     \* 根据id查找用户   
     \* @param id 用户id   
     \* @return 返回指定用户   
     \*/   
    public Users findById(Long id){   
    return iUsers.findById(id);   
    }   
  
}