【配置项目】

【struts2】

[struts.xml]

**2，新建源码包Source Folder与xml配置文件储存struts2的配置(dtd)**

**dtd的配置可以从文档里拷贝，也可以在struts2-core核心包里找到(打开dtd文件)**

**<?xml version="1.0" encoding="UTF-8"?>**

**<!--dtd文件路徑獲取 -->**

**<!DOCTYPE struts PUBLIC**

**"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"**

**"http://struts.apache.org/dtds/struts-2.0.dtd">**

**<!--配置根源路徑 ，扩展struts-default默认包-->**

<struts>

<constant name="struts.action.extension" value="do,action"/>

<constant name="struts.devMode" value="true" />

<constant name="struts.configuration.xml/reload" value="true"/>

<package name="surveySystem" namespace="/" extends="struts-default">

</package>

</struts>

**【hibernate.cfg.xml的dtd可以在core包的hibernate-configuration-3.0.dtd文件里拷贝】**

[web.xml]

**1，过滤器**

**<filter>**

**<filter-name>action</filter-name>**

**<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>**

**</filter>**

**<filter-mapping>**

**<filter-name>action</filter-name>**

**<url-pattern>/\*</url-pattern>**

**</filter-mapping>**

数据采集系统:

ssh:struts2 spring hibernate.

struts2:开发web程序的框架,是更加整洁的mvc结构.

分离关注：拦截器.

action:原型.线程安全性.

耦合度低:和原生的servlet api，

hibernate:持久化技术,替代jdbc,封装了数据的访问细节,体现了oop的思想.

spring:业务层框架,管理bean.

ioc:inverse of control,反转控制.

aop:aspect oriented program,不改变源代码,还给类增加新的功能.对oop进行增强.

代理.

前置

后置

环绕:事务管理.

异常

引入

1.创建web项目

2.创建各种包.

com.atguigu.surveypark.dao.impl

com.atguigu.surveypark.model

com.atguigu.surveypark.service.impl

com.atguigu.surveypark.struts2.action

com.atguigu.surveypark.util

3.引入类库

[struts2]

asm-3.3.jar

asm-commons-3.3.jar

asm-tree-3.3.jar

commons-fileupload-1.3.jar

commons-io-2.0.1.jar

commons-lang3-3.1.jar

commons-logging-1.1.3.jar

freemarker-2.3.19.jar

javassist-3.11.0.GA.jar

log4j-1.2.17.jar

ognl-3.0.6.jar

struts2-core-2.3.15.1.jar

xwork-core-2.3.15.1.jar

[hibernate]

antlr-2.7.7.jar

hibernate-commons-annotations-4.0.2.Final.jar

hibernate-core-4.2.3.Final.jar

hibernate-entitymanager-4.2.3.Final.jar

hibernate-jpa-2.0-api-1.0.1.Final.jar

javassist-3.15.0-GA.jar

hibernate-ehcache-4.2.3.Final.jar

[spring]

org.springframework.aop-3.1.0.CI-1162.jar

org.springframework.asm-3.1.0.CI-1162.jar

org.springframework.aspects-3.1.0.CI-1162.jar

org.springframework.beans-3.1.0.CI-1162.jar

org.springframework.context-3.1.0.CI-1162.jar

org.springframework.context.support-3.1.0.CI-1162.jar

org.springframework.core-3.1.0.CI-1162.jar

org.springframework.expression-3.1.0.CI-1162.jar

org.springframework.jdbc-3.1.0.CI-1162.jar

org.springframework.orm-3.1.0.CI-1162.jar

org.springframework.transaction-3.1.0.CI-1162.jar

org.springframework.web-3.1.0.CI-1162.jar

com.springsource.net.sf.cglib-2.2.0.jar

com.springsource.org.aopalliance-1.0.0.jar

com.springsource.org.aspectj.tools-1.6.6.RELEASE.jar

com.springsource.org.aspectj.weaver-1.6.8.RELEASE.jar

[struts2-spring插件]

struts2-spring-plugin-2.3.15.1.jar

[数据源]

com.springsource.com.mchange.v2.c3p0-0.9.1.2.jar

[驱动程序]

mysql-connector-java-5.0.8-bin.jar

3.配置项目

[struts2 + web]

[web-inf/web.xml]

<!-- 配置struts2的过滤器 -->

<filter>

<filter-name>action</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>action</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

[confi/struts.xml]

<?xml version="1.0"?>

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"

"http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

<package name="surveyparkPkg" extends="struts-default" namespace="/">

</package>

</struts>

[spring--config/beans.xml]

1.创建数据库:lsn\_surveypark001

2.配置config/bean.xml

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.1.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.1.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.1.xsd">

<!-- 分散配置 -->

<context:property-placeholder location="classpath:jdbc.properties"/>

<!-- 配置数据源 -->

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="${jdbc.driverclass}" />

<property name="jdbcUrl" value="${jdbc.url}" />

<property name="user" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

<property name="maxPoolSize" value="${c3p0.pool.size.max}" />

<property name="minPoolSize" value="${c3p0.pool.size.min}" />

<property name="initialPoolSize" value="${c3p0.pool.size.ini}" />

<property name="acquireIncrement" value="${c3p0.pool.size.increment}" />

</bean>

</beans>

[jdbc.properties]

jdbc.driverclass=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://locallhost:3306/lsn\_surveypark001

jdbc.username=root

jdbc.password=root

c3p0.pool.size.max=10

c3p0.pool.size.min=2

c3p0.pool.size.ini=3

c3p0.pool.size.increment=2

4.测试数据源

public class TestDataSource {

@Test

public void getConnection() throws SQLException{

ApplicationContext ac = new ClassPathXmlApplicationContext("beans0.xml");

DataSource ds = (DataSource) ac.getBean("dataSource");

System.out.println(ds.getConnection());

}

}

5.实体关系分析

1.类结构:带箭头是单线关联,不带箭头是双向关联

----------------------------------------

class User (1)<------(\*) class Survey (1)-------(\*) class Page (1)-------(\*) class Question

{ { { {

Integer id ; Integer id ; Integer id ; Integer id ;

... ... ... ...

User user ; Survey survey ; Page page ;

Set<Page> pages ; Set<Question> questions ;

} } } }

2.表结构

------------------------------------------------------------------

[users]

+----------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+----------+-------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| email | varchar(50) | YES | | NULL | |

| password | varchar(50) | YES | | NULL | |

| nickname | varchar(50) | YES | | NULL | |

| regdate | datetime | YES | | NULL | |

+----------+-------------+------+-----+---------+----------------+

[surveys]

+---------------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+---------------+--------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| title | varchar(200) | YES | | NULL | |

| pretext | varchar(50) | YES | | NULL | |

| nexttext | varchar(50) | YES | | NULL | |

| exittext | varchar(50) | YES | | NULL | |

| donetext | varchar(50) | YES | | NULL | |

| createtime | datetime | YES | | NULL | |

| userid | int(11) | YES | MUL | NULL | |

+---------------+--------------+------+-----+---------+----------------+

[pages]

+-------------+---------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------------+---------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| title | varchar(200) | YES | | NULL | |

| description | varchar(200) | YES | | NULL | |

| surveyid | int(11) | YES | MUL | NULL | |

+-------------+---------------+------+-----+---------+----------------+

[questions]

+---------------------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+---------------------+--------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| questiontype | int(11) | YES | | NULL | |

| title | varchar(200) | YES | | NULL | |

| options | varchar(200) | YES | | NULL | |

| other | bit(1) | YES | | NULL | |

| otherstyle | int(11) | YES | | NULL | |

| otherselectoptions | varchar(200) | YES | | NULL | |

| matrixrowtitles | varchar(200) | YES | | NULL | |

| matrixcoltitles | varchar(200) | YES | | NULL | |

| matrixselectoptions | varchar(200) | YES | | NULL | |

| pageid | int(11) | YES | MUL | NULL | |

+---------------------+--------------+------+-----+---------+----------------+

3.映射文件

------------------------------------------

[User.hbm.xml]

<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.atguigu.surveypark.model.User" table="users">

<id name="id" column="id" type="integer">

<generator class="identity" />

</id>

<property name="email" column="email" type="string" length="50" />

<property name="password" column="password" type="string" length="50" />

<property name="nickName" column="nickname" type="string" length="50" />

<property name="regDate" column="regdate" type="timestamp" update="false"/>

</class>

</hibernate-mapping>

[Survey.hbm.xml]

<hibernate-mapping package="com.atguigu.surveypark.model">

<class name="Survey" table="surveys">

<id name="id" column="id" type="integer">

<generator class="identity" />

</id>

<property name="title" column="title" type="string" length="200" />

<property name="preText" column="pretext" type="string" length="50" />

<property name="nextText" column="nexttext" type="string" length="50" />

<property name="doneText" column="donetext" type="string" length="50" />

<property name="exitText" column="exittext" type="string" length="50" />

<property name="createTime" column="createtime" type="string" length="200" />

<!-- 映射从Survey到User之间多对一关联关系 -->

<many-to-one name="user" class="User" column="userid" />

<!-- 映射从Survey到Page之间一对多关联关系 -->

<set name="pages" inverse="true">

<key column="surveyid" />

<one-to-many class="Page"/>

</set>

</class>

</hibernate-mapping>

[Page.hbm.xml]

<hibernate-mapping package="com.atguigu.surveypark.model">

<class name="Page" table="pages">

<id name="id" column="id" type="integer">

<generator class="identity" />

</id>

<property name="title" column="title" type="string" length="100" />

<property name="description" column="description" type="string" length="200" />

<!-- 映射从Page到Survey之间多对一关联关系 -->

<many-to-one name="survey" class="Survey" column="surveyid" />

<!-- 映射从Page到Question之间一对多关联关系 -->

<set name="questions" inverse="true">

<key column="pageid" />

<one-to-many class="Question"/>

</set>

</class>

</hibernate-mapping>

[Question.hbm.xml]

<hibernate-mapping package="com.atguigu.surveypark.model">

<class name="Question" table="questions">

<id name="id" column="id" type="integer">

<generator class="identity" />

</id>

<property name="questionType" column="questiontype" type="integer" />

<property name="title" column="title" type="string" length="100" />

<property name="options" column="options" type="string" length="200" />

<property name="other" column="other" type="boolean"/>

<property name="otherStyle" column="otherstyle" type="integer" />

<property name="otherSelectOptions" column="otherselectoptions" type="string" length="200" />

<property name="matrixRowTitles" column="maxtrixrowtitles" type="string" length="200" />

<property name="matrixColTitles" column="matrixcoltitles" type="string" length="200" />

<property name="matrixSelectOptions" column="matrixselectoptions" type="string" length="200" />

<!-- 映射从Question到Page之间多对一关联关系 -->

<many-to-one name="page" class="Page" column="pageid" />

</class>

</hibernate-mapping>

rose安装以及破解

----------------------

1.解压rose.zip

2.运行setup.exe文件 -->进入安装界面

3.选择J Edition 选项 --> next --> next ..

4.选择安装目录:d:\rational

5.打开授权key的窗口(破解)

1.解压CRACK.rar文件

2.选择 import a rational licenses file选项 --> next

3.进入import license file窗口 --> browse --> 定位D:\Download\rose\CRACK\license.upd文件 --> import --> import --> ok

dao和实现类

-----------------------------------------

1.dao

[BaseDao<T>接口]

public interface BaseDao<T> {

//写操作

public void saveEntity(T t);

public void saveOrUpdateEntity(T t);

public void updateEntity(T t);

public void deleteEntity(T t);

public void batchEntityByHQL(String hql,Object...objects);

//读操作

public T loadEntity(Integer id);

public T getEntity(Integer id);

public List<T> findEntityByHQL(String hql,Object...objects);

}

[BaseDaoImpl<T>实现类]

public abstract class BaseDaoImpl<T> implements BaseDao<T> {

//注入sessionFactory

@Resource

private SessionFactory sf ;

private Class<T> clazz ;

public BaseDaoImpl(){

//得到泛型话超类

ParameterizedType type = (ParameterizedType) this.getClass().getGenericSuperclass();

clazz = (Class<T>) type.getActualTypeArguments()[0];

}

public void saveEntity(T t) {

sf.getCurrentSession().save(t);

}

public void saveOrUpdateEntity(T t) {

sf.getCurrentSession().saveOrUpdate(t);

}

...

}

[UserDaoImpl实现类]

@Repository("userDao")

public class UserDaoImpl extends BaseDaoImpl<User> {

}

[SurveyDaoImpl实现类]

@Repository("surveyDao")

public class SurveyDaoImpl extends BaseDaoImpl<Survey> {

}

...

2.service

[BaseService<T>接口]

public interface BaseService<T> {

//写操作

public void saveEntity(T t);

public void saveOrUpdateEntity(T t);

public void updateEntity(T t);

public void deleteEntity(T t);

public void batchEntityByHQL(String hql,Object...objects);

//读操作

public T loadEntity(Integer id);

public T getEntity(Integer id);

public List<T> findEntityByHQL(String hql,Object...objects);

}

[BaseServiceImpl<T>实现类]

public abstract class BaseServiceImpl<T> implements BaseService<T> {

private BaseDao<T> dao ;

//注入dao

@Resource

public void setDao(BaseDao<T> dao) {

this.dao = dao;

}

public void saveEntity(T t) {

dao.saveEntity(t);

}

public void saveOrUpdateEntity(T t) {

dao.saveOrUpdateEntity(t);

}

...

}

[UserService接口]

public interface UserService extends BaseService<User> {

}

[UserServiceImpl实现类]

@Service("userService")

public class UserServiceImpl extends BaseServiceImpl<User> implements

UserService {

/\*\*

\* 重写该方法,目的是为了覆盖超类中该方法的注解,指明注入指定的Dao对象,否则spring

\* 无法确定注入哪个Dao---有四个满足条件的Dao.

\*/

@Resource(name="userDao")

public void setDao(BaseDao<User> dao) {

super.setDao(dao);

}

}

3.配置config/beans0.xml文件

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.1.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.1.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.1.xsd">

<!-- 分散配置 -->

<context:property-placeholder location="classpath:jdbc.properties"/>

<context:component-scan base-package="com.atguigu.surveypark.dao.impl,com.atguigu.surveypark.service.impl,com.atguigu.surveypark.struts2.action" />

<!-- 配置数据源 -->

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="${jdbc.driverclass}" />

<property name="jdbcUrl" value="${jdbc.url}" />

<property name="user" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

<property name="maxPoolSize" value="${c3p0.pool.size.max}" />

<property name="minPoolSize" value="${c3p0.pool.size.min}" />

<property name="initialPoolSize" value="${c3p0.pool.size.ini}" />

<property name="acquireIncrement" value="${c3p0.pool.size.increment}" />

</bean>

<!-- 本地回话工厂bean(spring整合hibernate的核心入口) -->

<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

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

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

<property name="mappingDirectoryLocations">

<list>

<value>classpath:com/atguigu/surveypark/model</value>

</list>

</property>

</bean>

<!-- hibnerate事务管理器,用来在service层面上实现事务管理,而且达到平台无关性 -->

<bean id="txManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory" ref="sessionFactory" />

</bean>

<!-- 事务通知 -->

<tx:advice id="txAdvice" transaction-manager="txManager">

<tx:attributes>

<!-- 写操作 -->

<tx:method name="save\*" propagation="REQUIRED" isolation="DEFAULT"/>

<tx:method name="update\*" propagation="REQUIRED" isolation="DEFAULT"/>

<tx:method name="delete\*" propagation="REQUIRED" isolation="DEFAULT"/>

<tx:method name="batch\*" propagation="REQUIRED" isolation="DEFAULT"/>

<!-- 读操作 -->

<tx:method name="load\*" propagation="REQUIRED" isolation="DEFAULT" read-only="true"/>

<tx:method name="get\*" propagation="REQUIRED" isolation="DEFAULT" read-only="true"/>

<tx:method name="find\*" propagation="REQUIRED" isolation="DEFAULT" read-only="true"/>

<tx:method name="\*" propagation="REQUIRED" isolation="DEFAULT"/>

</tx:attributes>

</tx:advice>

<!-- aop配置 -->

<aop:config>

<aop:advisor advice-ref="txAdvice" pointcut="execution(\* \*..\*Service.\*(..))"/>

</aop:config>

</beans>

4.测试插入用户

public class TestUserService {

private static UserService us ;

@BeforeClass

public static void iniUserService(){

ApplicationContext ac = new ClassPathXmlApplicationContext("beans0.xml");

us = (UserService) ac.getBean("userService");

}

/\*\*

\* 插入用户

\*/

@Test

public void insertUuser() throws SQLException{

User u = new User();

u.setEmail("xupccc@hotmail.com");

u.setPassword("123456");

u.setNickName("stone");

us.saveEntity(u);

}

}

【spring】

1.创建数据库

2.配置数据源

<?xml version="1.0" encoding="UTF-8"?>

<!-- 加aop，context，tx -->

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-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/aop

http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

<!--属相占位符扫描 -->

<context:property-placeholder location="classpath:jdbc.properties"/>

<!--dbcp:class="org.apache.commons.dbcp.BasicDataSource"-->

<!--c3p0:class="com.mchange.v2.c3p0.ComboPooledDataSource"-->

<bean id="datasource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="${driverClass}"/>

<property name="jdbcUrl" value="${jdbcUrl}"/>

<property name="username" value="${username}"/>

<property name="password" value="${password}"/>

<property name="maxPoolSize" value="${maxPoolSize}"/>

<property name="minPoolSize" value="${minPoolSize}"/>

<property name="initialPoolSize" value="${initialPoolSize}"/>

<property name="acquireIncrement" value="${acquireIncrement}"/><!--增量 -->

</bean>

</beans>

3.测试

4.创建数据库以及模型类

5.创建映射文件hbm.xml

**dtd文件可以在hibernate-core.jar的org.hibernate(第一个包)包的最后一个文件找到**

**spring的dtd文件可以在spring的jar的org.springframework.beans.factory.xml包的最后**

任何对象的构造必然经过构造函数，而构造函数是从根本集的，也就是说先从父类开始构造。而抽象父类无对象

因此在父类中使用this其实就是子类的对象，可以通过在构造函数中

public BaseDaoImpl(){

//通过反射获取泛型化超类

ParameterizedType type = (ParameterizedType) this.getClass().getGenericSuperclass();

//获取实参,返回的是数组

type.getActualTypeArguments()[0];

}

操作获取泛型父类的类型(参数化类型)

【MD5加密】

先将字符串转变为一个16位字节数组，然后每个字节高四位和低四位各用一个16进制数表示，这样就出现了一个32位的字节数组

做法：先将低四位与00001111进行与运算，然后将高四位向右移四位，再与00001111进行与运算

char[] chars={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

StringBuffer sb = new StringBuffer();

String str = "123456";

byte[] bytes = str.getBytes();//编程字节数组

//获取md5分解后的16位字节数组

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] tags = md.digest(bytes);

//把每个字节都分解为两个十六进制字符

for(byte t:tags){

sb.append(chars[(t>>4) & 0x0f]);

sb.append(chars[t & 0x0f]);

}

System.out.print(sb.toString());

【注册页面】

1】 功能性:

到达注册页面

1.导航栏-->用户注册链接

2.进入RegAction.toRegPage()方法

public String toRegPage(){

return "regPage" ;

}

3.struts.xml

<action name="RegAction\_\*" class="regAction" method="{1}">

<result name="regPage">/reg.jsp</result>

</action>

4.跳转到/reg.jsp

<s:form>

<s:textfield name="email" />

...

</s:form>

2 】注册监听器

配置spring的监听器(web.xml)

--------------------------

**<!-- 通过上下文参数配置spring文件的位置，提供给下面的上下文载入器监听器-->**

**<context-param>**

**<param-name>contextConfigLocation</param-name>**

**<param-value>classpath:beans.xml</param-value>**

**</context-param>**

**<!-- 上下文载入器监听器,确保web服务器启动时,直接完成spring容器的初始化，通过上面参数配置找spring配置文件 -->**

**<listener>**

**<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>**

**</listener>**

【部署测试】

界面可能会出异样：默认主题原因，修改主题：

<!--改变struts主题，默认struts.ui.theme=xhtml 以表格方式修饰页面-->

<constant name=*"struts.ui.theme"* value=*"simple"*></constant>

【用户注册功能实现】

**1.注册页面--->提交**

**2.RegAction.doReg()方法**

public String doReg(){

1.接受用户信息--model

2.userService.saveEntity(model);

3.return "success" ;

}

**3.校验**

public void validate(){ //内置的方法

1.非空

2.密码一致性

3.email是否占用

}

**4. struts.xml**

<action ...>

<result name="success">/index.jsp</result>

<result name="input">/reg.jsp</result>

</action>

**5.跳转到/reg.jsp|/index.jsp**

**【用户登录】**

【跳过校验】

//到达注册页面,跳过校验

@SkipValidation

public String toRegPage(){

return "regPage";

}

【只让某个方法校验】

有效前缀validate和validateDo

public String doLogin(){

return SUCCESS;

}

/\*

\* 校验登陆信息

\*/

public void validateDoLogin() {

super.validate();

User user = userService.validateLoginInfo(email,DataUtil.md5(password));

if(user==null){

addActionError("email/password错误！");

}else{

/\* ServletActionContext.getRequest().getSession().setAttribute("user", user);

\* 一般不采用，耦合度太高

\*/

sessionMap.put("user", user);

}

}

【sevletconfig拦截器】

在Struts2的默认拦截器栈中，servletConfig拦截器是在validate拦截器之前运行的，这保证了当action中的validate()方法运行时session对象已经设定好，从而可以在验证过程中进行Session操作

【surveyService操作多个实体，因此不需继承baseBaseServiceImpl】