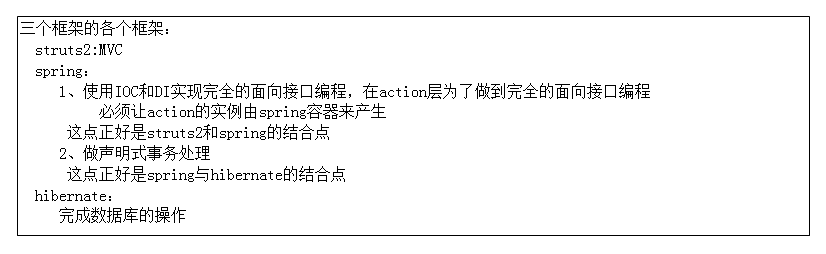
# SSH整合点：



# 整合的原则

谁能先测试，先写谁

1、持久化类、映射文件

2、引入sessionFactory

3、dao层和service层

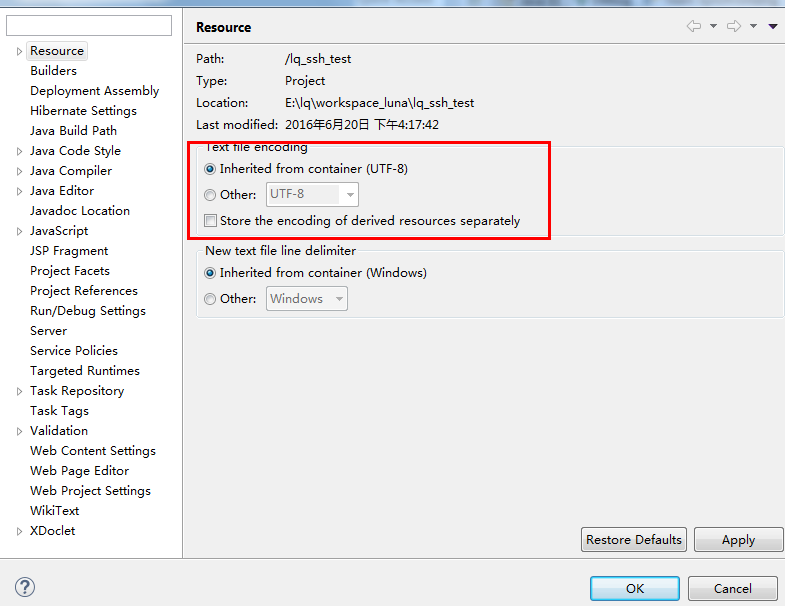
4、写spring的声明式事务处理

5、写action和前台

# 步骤

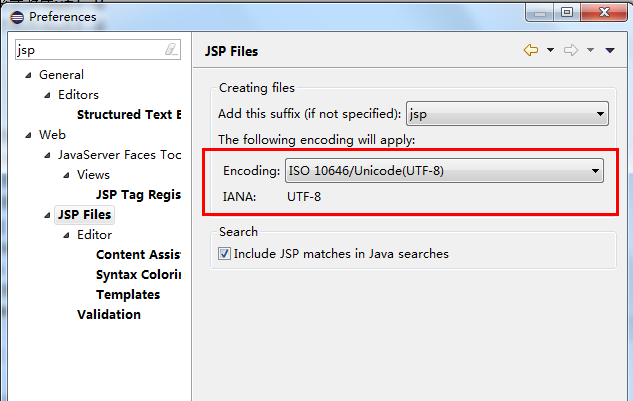
## 1、创建一个web project

## 2、把该工程的编码形式改成utf-8



## 3、把jsp的编码形式也改成utf-8

Windows🡪Peferences



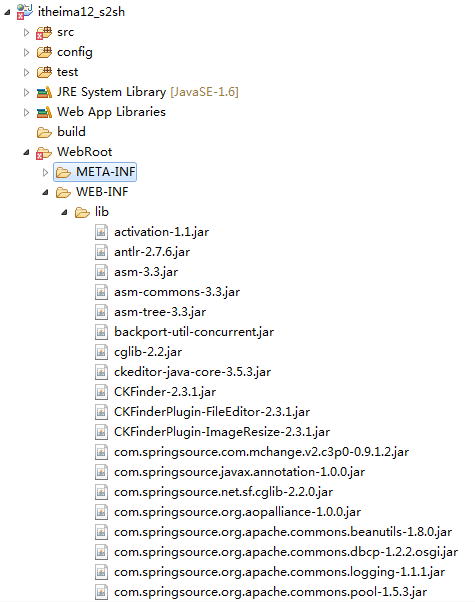
## 4、导入包

struts2

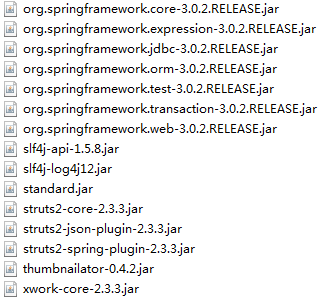
hibernate

spring

数据库的驱动包







## 5、创建三个source folder

src 存放源代码

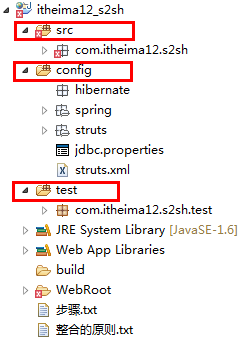
config 存放配置文件

spring

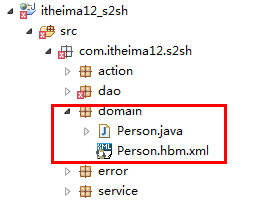
hibernate

struts

test 存放测试类



## 6、创建持久化类和映射文件



Person:

**public** **class** Person **implements** Serializable{

**private** Long pid;

**private** String name;

**private** String description;

**public** Person(String name){

**this**.name = name;

}

**public** Person(){}

**public** Long getPid() {

**return** pid;

}

**public** **void** setPid(Long pid) {

**this**.pid = pid;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

}

Person.hbm.xml:

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

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<!--

class用来描述一个类

name 类的全名

table 该持久化类对应的表名 可以不写，默认值为类名

catalog 数据库的名称

-->

<class name=*"com.itheima12.s2sh.domain.Person"* table=*"person"*>

<!--

用来描述主键

name 属性的名称

column 属性的名称对应的表的字段 可以不写 默认值就是属性的名称

length 属性的名称对应的表的字段的长度 如果不写，默认是最大的长度

-->

<id name=*"pid"* column=*"pid"* length=*"5"*>

<!--

主键的产生器

-->

<generator class=*"increment"*></generator>

</id>

<property name=*"name"* length=*"20"* type=*"string"*></property>

<property name=*"description"* length=*"50"* type=*"java.lang.String"*></property>

</class>

</hibernate-mapping>

## 7、写spring的配置文件

config/spring/

applicationContext.xml

applicationContext-db.xml

applicationContext.xml:

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

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

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

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

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

*http://www.springframework.org/schema/beans/spring-beans-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/tx*

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

<import resource=*"applicationContext-db.xml"*/>

<import resource=*"applicationContext-person.xml"*/>

</beans>

applicationContext-db.xml:

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

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

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

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

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

*http://www.springframework.org/schema/beans/spring-beans-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/tx*

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

<!--

引入prperties配置文件

-->

<bean

class=*"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"*>

<property name=*"locations"*>

<value>classpath:jdbc.properties</value>

</property>

</bean>

<bean id=*"dataSource"* destroy-method=*"close"*

class=*"org.apache.commons.dbcp.BasicDataSource"*>

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

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

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

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

</bean>

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

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

<!--

映射文件所在的路径

-->

<property name=*"mappingDirectoryLocations"*>

<list>

<!--

spring容器会去该包及子包下搜索所有的映射文件

-->

<value>classpath:com/itheima12/s2sh/domain</value>

</list>

</property>

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>org.hibernate.dialect.MySQL5InnoDBDialect</prop>

<prop key=*"hibernate.show\_sql"*>true</prop>

<prop key=*"hibernate.hbm2ddl.auto"*>update</prop>

</props>

</property>

</bean>

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

<property name=*"sessionFactory"*>

<ref bean=*"sessionFactory"*/>

</property>

</bean>

<tx:advice transaction-manager=*"transactionManager"* id=*"tx"*>

<tx:attributes>

<tx:method name=*"save\*"* read-only=*"false"*/>

<tx:method name=*"update\*"* read-only=*"false"*/>

<tx:method name=*"delete\*"* read-only=*"false"*/>

<!--

除了save开头的方法以外，其他的方法用只读事务

-->

<tx:method name=*"\*"* read-only=*"true"*/>

</tx:attributes>

</tx:advice>

<aop:config>

<aop:pointcut

expression=*"execution(\* com.itheima12.s2sh.service.impl.\*.\*(..))"*

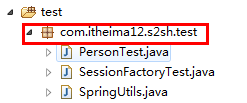
id=*"perform"*/>

<aop:advisor advice-ref=*"tx"* pointcut-ref=*"perform"*/>

</aop:config>

</beans>

## 8、在test创建一个包com.itheima12.s2sh.test



## 9、在上面的包中创建一个类SessionFactoryTest,SpringUtils

SpringUtils:

**public** **class** SpringUtils {

**public** **static** ApplicationContext *context*;

**static**{

*context* = **new** ClassPathXmlApplicationContext("spring/applicationContext.xml");

}

}

SessionFactoryTest:

**public** **class** SessionFactoryTest **extends** SpringUtils{

@Test

**public** **void** testSessionFactory(){

*context*.getBean("sessionFactory");

}

}

## 10、创建dao层和service层的类和接口

PersonDao:

**public** **interface** PersonDao {

**public** **void** savePerson(Person person);

**public** Person getPersonById(Serializable id);

}

PersonDaoImpl:

**public** **class** PersonDaoImpl **extends** HibernateDaoSupport **implements** PersonDao{

@Override

**public** **void** savePerson(Person person) {

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

WebApplicationContextUtils.getWebApplicationContext(**null**);

**this**.getHibernateTemplate().save(person);

}

@Override

**public** Person getPersonById(Serializable id) {

**return** **this**.getHibernateTemplate().load(Person.**class**, id);

}

}

PersonService:

**public** **interface** PersonService {

**public** **void** savePerson(Person person);

**public** Person getPersonById(Serializable id);

}

PersonServiceImpl:

**public** **class** PersonServiceImpl **implements** PersonService{

**private** PersonDao personDao;

**public** PersonDao getPersonDao() {

**return** personDao;

}

**public** **void** setPersonDao(PersonDao personDao) {

**this**.personDao = personDao;

}

@Override

**public** **void** savePerson(Person person) {

**this**.personDao.savePerson(person);

}

@Override

**public** Person getPersonById(Serializable id) {

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

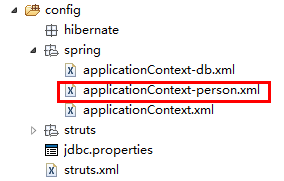
Person person = **this**.personDao.getPersonById(id);

**return** person;

}

}

## 11、写spring的声明式事务处理的配置和把dao层，service层所有的类放入到spring容器中



applicationContext-person.xml:

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

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

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

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

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

*http://www.springframework.org/schema/beans/spring-beans-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/tx*

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

<bean id=*"personDao"* class=*"com.itheima12.s2sh.dao.impl.PersonDaoImpl"*>

<property name=*"sessionFactory"*>

<ref bean=*"sessionFactory"*/>

</property>

</bean>

<bean id=*"personService"* class=*"com.itheima12.s2sh.service.impl.PersonServiceImpl"*>

<property name=*"personDao"*>

<ref bean=*"personDao"*/>

</property>

</bean>

<bean id=*"personAction"* class=*"com.itheima12.s2sh.action.PersonAction"*

scope=*"prototype"*>

<property name=*"personService"*>

<ref bean=*"personService"*/>

</property>

</bean>

</beans>

## 12、进行测试声明式事务处理

**public** **class** PersonTest **extends** SpringUtils{

@Test

**public** **void** testSavePerson(){

PersonService personService = (PersonService)*context*.getBean("personService");

Person person = **new** Person();

person.setName("adsf");

personService.savePerson(person);

}

## 13、创建一个action

**public** **class** PersonAction **extends** ActionSupport{

**private** PersonService personService;

**public** PersonService getPersonService() {

**return** personService;

}

**public** **void** setPersonService(PersonService personService) {

**this**.personService = personService;

}

**public** String savePerson(){

Person person = **new** Person();

person.setName("adsf");

personService.savePerson(person);

**return** **null**;

}

**public** String showPerson(){

**int** a = 1/0;

Person person = **this**.personService.getPersonById(1L);

ServletActionContext.getRequest().setAttribute("person", person);

**return** "index";

}

}

## 14、把action放入到spring容器中

<bean id="personAction" class="com.itheima12.s2sh.action.PersonAction"

scope="prototype">

<property name="personService">

<ref bean="personService"/>

</property>

</bean>

**注意事项：scope="prototype"必须写**

<bean id=*"personAction"* class=*"com.itheima12.s2sh.action.PersonAction"*

scope=*"prototype"*>

<property name=*"personService"*>

<ref bean=*"personService"*/>

</property>

</bean>

## 15、测试action的创建

**public** **class** PersonTest **extends** SpringUtils{

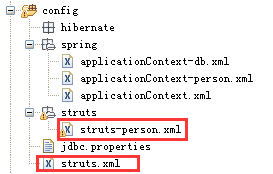
@Test

**public** **void** testPersonAction(){

*context*.getBean("personAction");

}

## 16、写struts2的配置文件



struts.xml:

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

<!DOCTYPE struts PUBLIC

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

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

<struts>

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

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

<include file=*"struts/struts-person.xml"*></include>

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

<global-results>

<result name=*"errHandler"* type=*"chain"*>

<param name=*"actionName"*>errorProcessor</param>

</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception=*"java.lang.Exception"*

result=*"errHandler"* />

</global-exception-mappings>

<action name=*"errorProcessor"* class=*"com.itheima12.s2sh.error.ErrorProcessor"*>

<result name=*"error"*>error.jsp</result>

</action>

</package>

</struts>

struts-person.xml:

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

<!DOCTYPE struts PUBLIC

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

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

<struts>

<package name=*"person"* namespace=*"/"* extends=*"struts-global"*>

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

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

</action>

</package>

</struts>

## 17、写web.xml文件

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

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* id=*"WebApp\_ID"* version=*"3.0"*>

<display-name>itheima11\_ssh</display-name>

<listener>

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

</listener>

<context-param>

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

<param-value>classpath:spring/applicationContext.xml</param-value>

</context-param>

<filter>

<filter-name>openSessionInViewFilter</filter-name>

<filter-class>org.springframework.orm.hibernate3.support.OpenSessionInViewFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>openSessionInViewFilter</filter-name>

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

</filter-mapping>

<filter>

<filter-name>struts2</filter-name>

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

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

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

</filter-mapping>

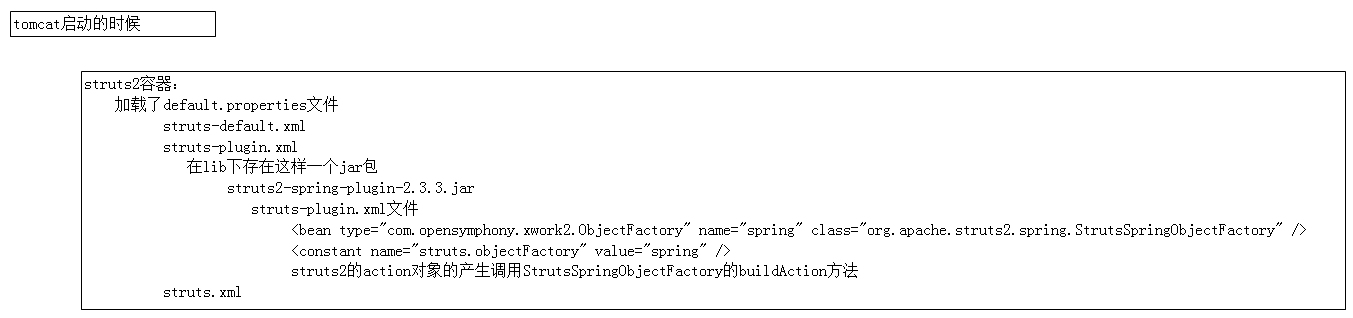
<welcome-file-list>

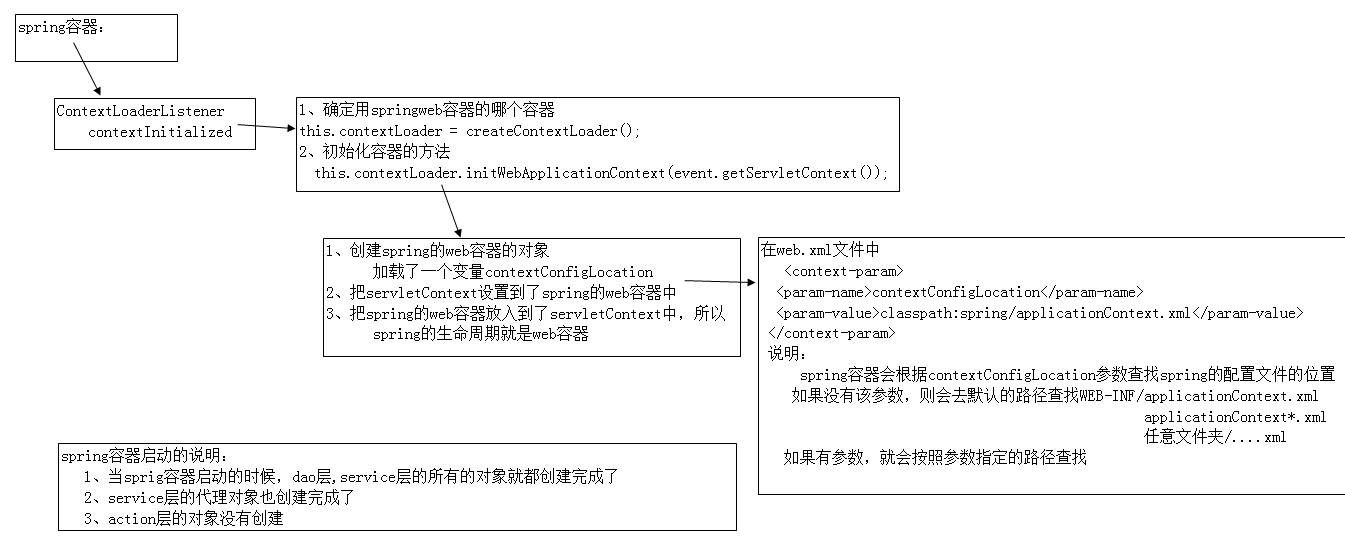
<welcome-file>index.jsp</welcome-file>

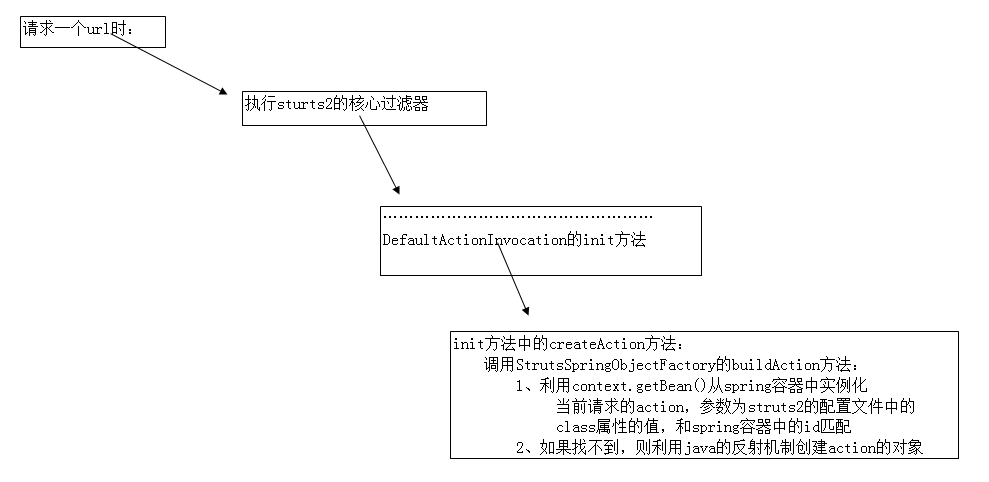
</welcome-file-list>

</web-app>

# 原理:







# spring与hibernate整合后的session的问题

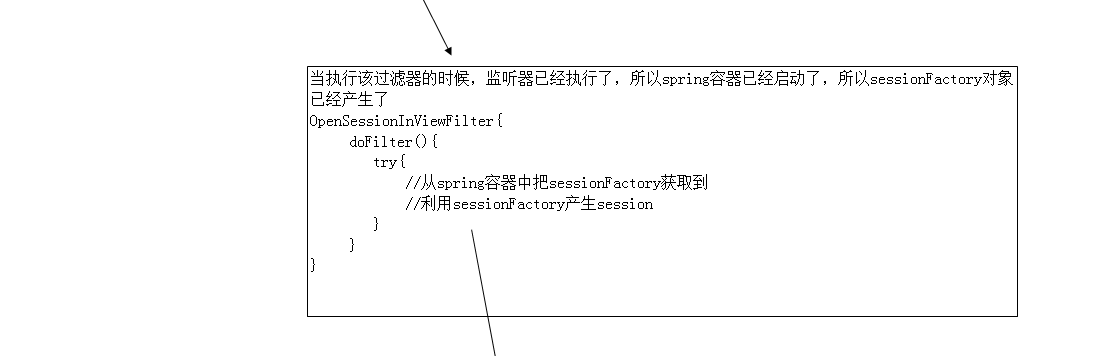
## 关于session的问题：

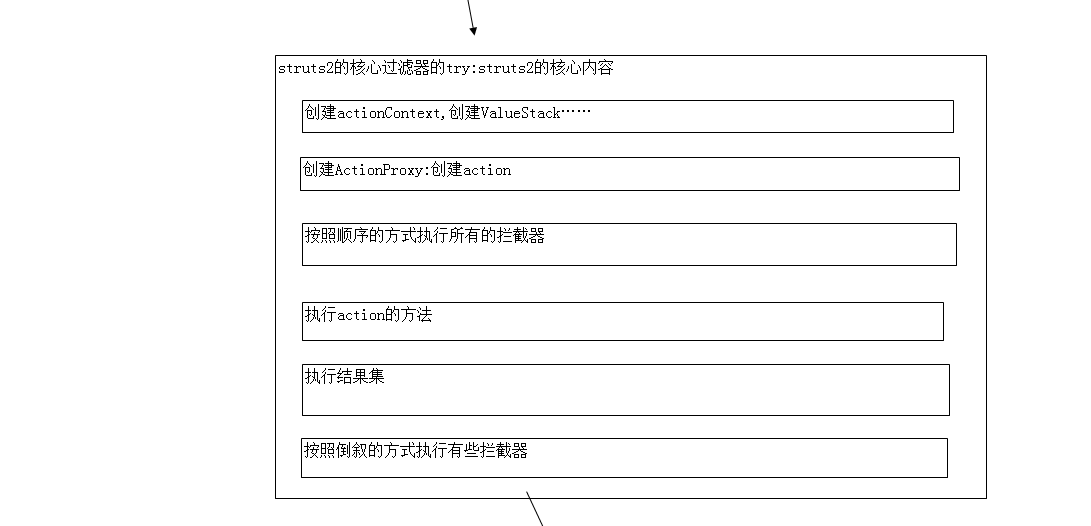
1、如果当前的方法不存在事务，则hibernateTemplate一执行完毕以后,session关闭。

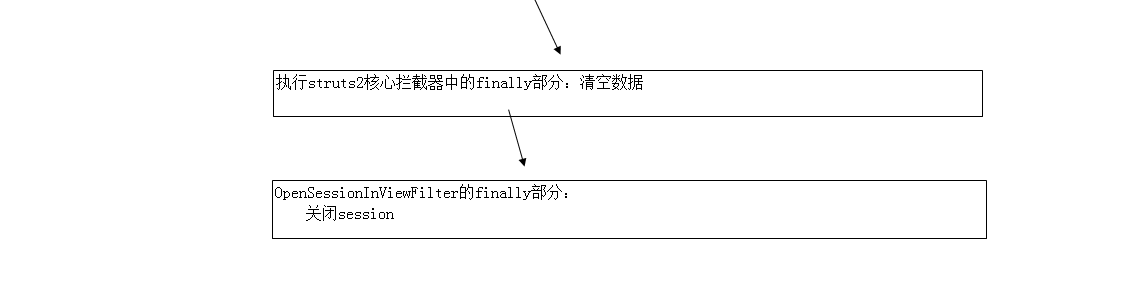
2、如果当前的方法存在事务环境，则session是没有关闭的，当事务方法结束以后,session关闭

## 执行过程：









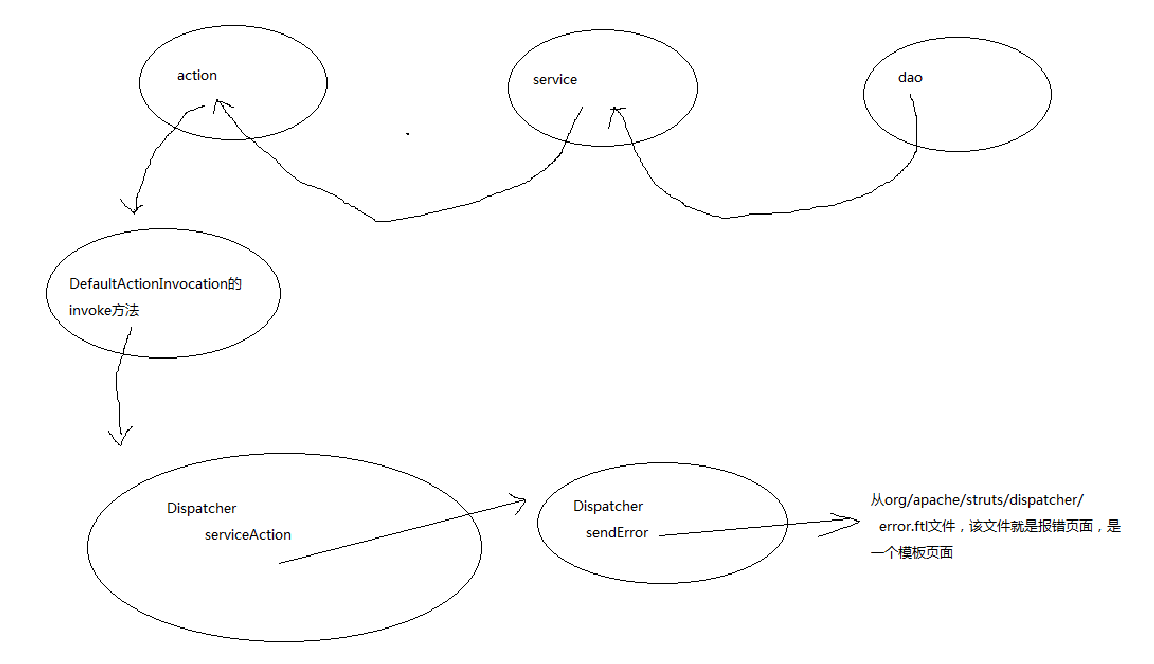
## 总结

OpenSessionInViewFilter模式通过提前打开session,延迟关闭session解决了hibernate中的懒加载的问题

说明：延迟关闭session说明一级缓存的生命周期被延长了，说明有更多的对象长时间的停留在了内存中

# struts2的错误处理

## 默认的struts2框架的错误处理



## struts2的错误处理

