<http://www.blogjava.net/rongxh7/archive/2009/03/23/261577.html>

# [基于Annotation的Struts2.0+Hibernate3.3+Spring2.5整合开发 (1)](http://www.blogjava.net/rongxh7/archive/2009/03/23/261577.html)

        写这个主题的文章在网络上应该暂时没有吧，也看过许多SSH整合开发的书，许多都是旧版本的框架整合，而且大部分都是以XML配置文件的方式，即使谈到Annotation也是针对其中一个框架而言的。随着开源框架更新版本中不断引进对Annotation的支持，使得我们可以用单方面去管理我们的类与关系，而不用像以往那样，一旦遇到问题，就要不断在Java视图和XML视图中切换。好，开始吧，既然网络中暂时没有这方面的文章，我就来开创个先例。自己探索学习一下，也让大家来学习和提出批评和建议。        首选说一下我们的开发准备吧！  
        开发环境：Tomcat 6.0，JDK1.6，MySQL5.0  
        开发工具：Eclipse-jee(Eclipse的官网上下载)  
        所要下载的框架包：struts-2.0.11.2-all.zip，spring-framework-2.5.5-with-dependencies.zip，hibernate-distribution-3.3.1.GA-dist.zip，这三个压缩包在其官方网站都可以下载。当然，这个小项目所要用到不止这些，我们用到的时候再去下载。  
        好，我们来开始第一阶段的探索——Hibernate3.3的Annotation的实现。  
        新建我们的Dynamic Web Project，取名为"ssh\_annotation”，配置Tomcat6.0的支持，打开MySQL，建立一个空数据库，取名为"ssh\_annotation"。记得，要引进MySQL的JDBC驱动包mysql-connector-java-5.0.8-bin.jar。  
        在解压缩的hibernate-distribution-3.3.1.GA目录中，把hibernate.jar，和\lib\required\下的所用jar包引进到我们项目的lib目录下。  
        不知道用过Hibernate3.3开发的朋友会不会遇到过这样的问题：把上面的包都准备好的，开发Hibernate测试类时，会报Caused by: java.lang.NoClassDefFoundError: org/slf4j/impl/StaticLoggerBinder 或者Caused by: java.lang.NoClassDefFoundError: org/apache/log4j/LogManager的错误。这是因为Hibernate3.3提供的jar包，缺少了slf4j-log4j12.jar这个包。解决办法是：在Hibernate的官方网站上下载hibernate-annotations-3.4.0.GA.zip，解压缩，把其\lib\下的slf4j-log4j12.jar引进我们项目的lib目录下。而我们引进这个jar包后，若我们没有引进log4j.jar的话，就会报java.lang.NoClassDefFoundError: org/apache/log4j/LogManager的错误，可能是slf4j-log4j12.jar要用调用log4j的类吧。我们也引进log4j.jar即行。

        当然，hibernate-annotations-3.4.0.GA.zip 是我们开发Hibernate Annotation所要用到的，除以上面那些jar包外，我们还要引进其hibernate-annotations.jar，及其\lib\下的hibernate-commons-annotations.jar，ejb3-persistence.jar。好，暂时就这些，我们接下来需要用到哪些jar包时再引进。一步，来开发我们的实体类User。代码如下：

@Entity  
@Table(name="tb\_user")  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class User {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Id    @GeneratedValue(strategy=GenerationType.IDENTITY)  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private int id;  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Column(name="username", length=15)  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private String username;  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Column(name="password", length=15)  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private String password;http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

        简单解释一下吧：  
        @Entity定义User类为实体类，@Id定义该属性对应的列为主键，@GeneratedValue(strategy=GenerationType.IDENTITY)定义主键生成策略为Indentity，适用于MySQL和MS SQL等数据库。@Column(name="username", length=15)定义列名和取值长度。  
        应该不难的，更多有关Hibernate的注解，请下载并参考我整理的Hibernate Annotation帮助文档 [/Files/rongxh7/hibernateAannotations.rar](http://www.blogjava.net/Files/rongxh7/hibernateAannotations.rar) 。慢慢学吧！  
          
        当然，必要的XML配置文件还是要有的：

http://www.blogjava.net/Images/OutliningIndicators/None.gif<?xml version='1.0' encoding='utf-8'?>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif<!DOCTYPE hibernate-configuration PUBLIC  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif<hibernate-configuration>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <session-factory>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="connection.driver\_class">com.mysql.jdbc.Driver</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="connection.url">jdbc:mysql://localhost:3306/ssh\_annotation</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="connection.username">root</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="connection.password">root</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <!-- 将Session扔到线程里去处理 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="current\_session\_context\_class">thread</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="show\_sql">true</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <!-- 自动把实体类与属性映射成数据库中的表与列 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="hbm2ddl.auto">update</property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <!--   <mapping resource="com/rong/entity/User.hbm.xml"/> -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif         <!-- 在Hibernate中注册User实体类,区别于上面注释掉的resource写法 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif         <mapping class="com.rong.entity.User"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </session-factory>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif</hibernate-configuration>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif

        到这里，我们可以运行Tomcat，Hibernate会根据hbm2ddl.auto自动地把实体类与属性转变成数据库中的表与列。但这一步，要事先在数据库中建立一个名为"ssh\_annotation"的空数据库。好了，看一下，你的数据库中是否有"tb\_user"的表生成了？！  
        接下来，我们来开发一个包装器类，取名HibernateUtil，代码如下：

http://www.blogjava.net/Images/OutliningIndicators/None.gifpackage test.rong.hibernate;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class HibernateUtil {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private static final SessionFactory sessionFactory;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    static {  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif        try {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            //用xxx.hbm.xml形式配置实体映射  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            //sessionFactory = new Configuration().configure().buildSessionFactory();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            //用Annotation形式配置实体映射  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif        } catch (Throwable ex) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            System.err.println("Initial SessionFactory creation failed." + ex);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            throw new ExceptionInInitializerError(ex);  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public static SessionFactory getSessionFactory() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return sessionFactory;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

        学过Hibernate的朋友们，应该对这个类不陌生吧。在此，我就不多解释啦，注意一下我注释的地方就行。下面，我们来进行测试，下载最新版本的Junit4.5.jar，试一下感觉如何。当然，旧版本的应该也没问题的啦！写测试代码如下：

public class EntityTest {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private Session session;  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Before  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void initSession(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.session = HibernateUtil.getSessionFactory().getCurrentSession();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Test    //测试添加用户  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void testSave(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.beginTransaction();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        User user = new User();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        user.setUsername("rongxh7");  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        user.setPassword("blogjava");  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.save(user);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.getTransaction().commit();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Test    //测试查询用户列表  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @SuppressWarnings("unchecked")  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void testQuery(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.beginTransaction();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        Query query = session.createQuery("from User u");  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        List<User> users = query.list();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif        for(Iterator<User> it = users.iterator() ; it.hasNext() ; ){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            User user = it.next();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            System.out.println("username = " + user.getUsername());  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.beginTransaction().commit();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Test    //测试查询单个用户  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void testGet(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.beginTransaction();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        User user = (User)session.get(User.class, 1);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        System.out.println("username = " + user.getUsername());  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.getTransaction().commit();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Test    //测试查询修改用户  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void testUpdate(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.beginTransaction();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        User user = (User)session.load(User.class, 1);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        user.setPassword("hello blogjava");  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.update(user);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.getTransaction().commit();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Test    //测试删除用户  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void testDelete(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.beginTransaction();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.delete(session.load(User.class, 1));  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        session.getTransaction().commit();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.blogjava.net/Images/OutliningIndicators/None.gif测试用例说明: 示例中对事务的控制在各个方法中手动编码.开启事务,操作,关闭事务.

        到此，我们的Hibernate Annotation开发完成了。你成功了没？如果有什么问题，请回复，我尽力为你解答！好，第一阶段暂告一段落。不要忘记，我们是要做的是基于Annotatioln的SSH整合开发，让我们一起期待下一阶段吧！

<http://www.blogjava.net/rongxh7/archive/2009/03/25/261782.html>

# [基于Annotation的Struts2.0+Hibernate3.3+Spring2.5整合开发 (2)](http://www.blogjava.net/rongxh7/archive/2009/03/25/261782.html)

        基于Annotation的SSH整合开发，其实，并没有我当初想像中那么顺利。真正去做的时候，才发觉有许多问题。但不要紧，探索一下吧。在探索过程中学到知识，才是最重要的。  
        言归正传，现在，我们加入Spring的支持：把spring-framework-2.5.5\dist中的spirng.jar引进我们项目的lib目录来，还要添加\lib\aspectj\下的两个jar包，以支持切面编程。  
        必要的配置文件还是要的：  
        applicationContext-common.xml

http://www.blogjava.net/Images/OutliningIndicators/None.gif<?xml version="1.0" encoding="UTF-8"?>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif<beans xmlns="http://www.springframework.org/schema/beans"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   
http://www.blogjava.net/Images/OutliningIndicators/None.gif    xmlns:context="http://www.springframework.org/schema/context"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    xmlns:aop="http://www.springframework.org/schema/aop"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    xmlns:tx="http://www.springframework.org/schema/tx"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    xsi:schemaLocation="  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif              
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 配置SessionFactory,由Spring容器来管理Hibernate -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 非Annotation时,使用org.springframework.orm.hibernate3.LocalSessionFactoryBean,  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        它注入实体类的方式是setMappingResources(),而Hibernate Annotation所用的映射方式  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        不是mapping resource,而是mapping class,这就要用到LocalSessionFactoryBean的子类  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        AnnotationSessionFactoryBean了.因为AnnotationSessionFactoryBean它支持实体的注入  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        方式setAnnotatedClasses,即对应Hibernate中的mapping class.参见这两个类的源代码. -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <bean id="sessionFactory"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="configLocation">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <value>classpath:hibernate.cfg.xml</value>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </bean>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 配置事务管理器 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <bean id="transactionManager"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        class="org.springframework.orm.hibernate3.HibernateTransactionManager">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="sessionFactory">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <ref bean="sessionFactory" />  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </bean>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif      
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 配置事务的传播特性 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <tx:advice id="txAdvice" transaction-manager="transactionManager">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <tx:attributes>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <tx:method name="save\*" propagation="REQUIRED" />  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <tx:method name="update\*" propagation="REQUIRED" />  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <tx:method name="delete\*" propagation="REQUIRED" />  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <tx:method name="\*" read-only="true" />  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </tx:attributes>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </tx:advice>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif      
http://www.blogjava.net/Images/OutliningIndicators/None.gif      
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 那些类的哪些方法参与事务 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <aop:config>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <aop:pointcut id="allServiceMethod" expression="execution(\* com.rong.**dao**.\*.\*.\*(..))" />  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <aop:advisor pointcut-ref="allServiceMethod" advice-ref="txAdvice" />  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </aop:config>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif      
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 使Spring关注Annotation -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <context:annotation-config/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif      
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 让Spring通过自动扫描来查询和管理Bean -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <context:component-scan base-package="com.rong"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif      
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!--   
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <bean id="userDao" class="com.rong.dao.UserDaoBean">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="sessionFactory" ref="sessionFactory"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </bean>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <bean id="userService" class="com.rong.service.UserServiceBean">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="userDao" ref="userDao"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </bean>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif     -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif</beans>

        这样配置之后，就省去了上面注释掉的DAO层和Service层等配置代码。是不是很方便呢。  
       关于这一部分的XML代码，我们下面还会作解释。来开发我们的DAO层吧,接口如下：

public interface UserDao {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public void save(User user);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public void delete(int id);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public void update(User user);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public List<User> query();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public User get(int id);  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

        DAO层的实现类：

**@Repository("userDao")**        //声明此类为数据持久层的类  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class UserDaoBean extends MyHibernateDaoSupport implements UserDao {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void save(User user){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        super.getHibernateTemplate().save(user);  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void delete(int id){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif         getHibernateTemplate().delete(super.getHibernateTemplate().load(User.class, id));  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void update(User user){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        super.getHibernateTemplate().update(user);  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @SuppressWarnings("unchecked")  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public List<User> query(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return super.getHibernateTemplate().find("from User");  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public User get(int id){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return (User)super.getHibernateTemplate().get("from User", id);  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

        大家可以看到，我们这里继承的不是HibernateDaoSupport,而是我自己编写的一个类MyHibernateDaoSupport。其代码如下：

public class MyHibernateDaoSupport extends HibernateDaoSupport {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Resource(name="sessionFactory")    //为父类HibernateDaoSupport注入sessionFactory的值  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setSuperSessionFactory(SessionFactory sessionFactory){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        super.setSessionFactory(sessionFactory);  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

        我们之所以要改写HibernateDaoSupport，是因为，我们要**为DAO层的类注入SessionFactory这个属性**。以后，我们开发的DAO类，就可以直接重用这个MyHibernateDaoSupport了。其实，这样做是相当于配置文件方式的代码：

http://www.blogjava.net/Images/OutliningIndicators/None.gif    <bean id="userDao" class="com.rong.dao.UserDaoBean">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="sessionFactory" ref="sessionFactory"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </bean>

        我们既然要用annotation代替XML文件的，就要让它也能像原来那样使用sessionFactory,故为MyHibernateDaoSupport注入SessionFactory。子类继承这个类时，也继承其Annotation。这样，我们就可以实现SessionFactory的注入了。  
         到现在，我们再回过头来看applicationContext-common.xml中的

http://www.blogjava.net/Images/OutliningIndicators/None.gif    <bean id="sessionFactory"  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <property name="configLocation">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <value>classpath:hibernate.cfg.xml</value>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </property>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </bean>

        我们平时开发Hibernate与Spring整合时，常常会用到org.springframework.orm.hibernate3.LocalSessionFactoryBean来提供SessionFactory，而我们这里却要改成AnnotationSessionFactoryBean。其实是这样的，我们在Hibernate.cfg.xml中配置的实体类映射的方式如下：（详见[基于Annotation的Struts2.0+Hibernate3.3+Spring2.5整合开发 (1)](http://www.blogjava.net/rongxh7/archive/2009/03/23/261577.html) ）

http://www.blogjava.net/Images/OutliningIndicators/None.gif        <!-- mapping resource="com/rong/entity/User.hbm.xml"/> ->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif           
http://www.blogjava.net/Images/OutliningIndicators/None.gif         <!-- 在Hibernate中注册User实体类,区别于上面注释掉的resource写法 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif         <mapping class="com.rong.entity.User"/>

        要使Hibernate的实体类支持注解，去掉xxx.hbm.xml的文件，故我们所用的是mapping class方式，不是mapping resource的方法。然而，LocalSessionFactoryBean这个类，它采用的实体类映射方式是mapping resource，（详情可参见LocalSessionFactoryBean这个类的源代码）。如果我们在配置中仍然用这个类的话，Hibernate与Spring整合时，就会报错。而AnnotationSessionFactoryBean这个类在LocalSessionFactoryBean的基础上添加了mapping class方式实现实体类映射（详见AnnotationSessionFactoryBean类的源代码）。  
        我们再来看Service层的代码：（接口比较简单，节约篇幅就不列出了）

**@Service("userService")**        //声明此类为业务逻辑层的类  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class UserServiceBean implements UserService {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Autowired  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private UserDao userDao;  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void save(User user){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        userDao.save(user);  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

        我们用到的注解上面一般都作了注释，就不多叙。@Autowired和@Resource功能差不多，就是把对象注入，相当于<bean>配置的功能。  
        好，就开发到这样，是不是忘记了什么？记得要配置web.xml，部分代码如下：

http://www.blogjava.net/Images/OutliningIndicators/None.gif      <!-- 修改Spring配置文件的路径 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <context-param>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <param-name>contextConfigLocation</param-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <param-value>classpath\*:applicationContext-\*.xml</param-value>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </context-param>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif      
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 配置Spring -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <listener>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </listener>

        是不是真的成功了？用Junit测试一下吧，我测试过是没问题的，由于篇幅，Junit的测试代码就不贴出来了。自己练习一下吧！  
        其实，到现在为止，我们发觉我们的XML配置文件还是很多。其实，这样想想，上一阶段我们省去了xxx.hbm.xml这类的文件，这一阶段，我们少去了<bean id="" class=""><property name="" ref="">这样的配置项。而这些，正是我们项目开发中，大量使用的配置。而只要书写简单的Annotation注解，就可以省去这样，我们何乐而不用。而那些我们保留的XML配置文件（如：数据库连接，事务），这样是写死的，一个项目就写一次或复制过来用，我们保留它又何妨？

        好，暂时到这里，我们还有下一阶段的基于Annotation的SSH整合开发，我们将会以一个用户注册的例子，把Struts2的注解带到我们的整合开发中来。一起期待吧！

# [基于Annotation的Struts2.0+Hibernate3.3+Spring2.5整合开发 (3)](http://www.blogjava.net/rongxh7/archive/2009/03/27/262260.html)

        进入我们基于Annotation的Struts2.0+Hibernate3.3+Spring2.5整合开发的最后一个阶段了，这阶段主要为我们的项目引进Struts2的Annotation。相对于带Annotation的Hibernate与带Annotation的Spring整合来说，带Annotation的Struts与带Annotation的Spring整合还是比较容易实现的。  
        我们先来引进Struts2.0的相关jar包：xwork-2.0.5.jar，struts2-core-2.0.11.2.jar，ognl-2.6.11.jar，commons-logging.jar(这个前面已引进)，freemarker-2.3.8.jar，还有Struts2.0与Spring整合所需的struts2-spring-plugin-2.0.11.2.jar。好，马上来开发我们的用户注册小例子。 register.jsp注册页的部分代码如下：

http://www.blogjava.net/Images/OutliningIndicators/None.gif    <s:form action="register">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <s:textfield name="username"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <s:password name="password"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <s:submit value="注册"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </s:form>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <s:fielderror></s:fielderror>

        success.jsp注册成功页部分代码如下：

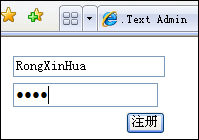
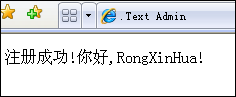
http://www.blogjava.net/Images/OutliningIndicators/None.gif注册成功!你好,<s:property value="username"/>!

        好，来开发我们的控制层吧。写个RegisterAction.java的类：

**@Controller    @Scope("prototype")**        //声明此类为控制层的类,且为prototype模式调用  
@ParentPackage(value="struts-default")  
@Results({    @Result(name="success",value="success.jsp"),@Result(name="input",value="register.jsp")})  
public class RegisterAction extends ActionSupport {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif   **@Resource(name="userService")  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private UserService userService;**  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private String username;  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private String password;  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Override  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public String execute() throws Exception {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        User user = new User();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        user.setUsername("rongxinhua");  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        user.setPassword("blogjava");  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        userService.save(user);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return SUCCESS;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public String getUsername() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return username;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @RequiredStringValidator(message = "请输入用户名!")  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setUsername(String username) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.username = username;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public String getPassword() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return password;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @RequiredStringValidator(message = "请输入密码!")  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setPassword(String password) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.password = password;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.blogjava.net/Images/OutliningIndicators/None.gif

        **Spring2.5通过@Controller @Scope("prototype") 这样的注解，将Struts的Action纳为自己的控制范围之内**。Struts2.0自带的@ParentPackage和@Results等注解，注明了要继承的父包和响应结果。还有@RequiredStringValidator，@RequiredFieldValidator等注解，大大方便了我们处理表单信息的校验。大家有没有发觉少了什么？我们不需要Struts.xml的配置文件了，也不需要xxx-validation.xml，也不需要写烦杂的验证代码了。岂不是很方便吗？  
        不过，不要忘记了，我们还得在web.xml中配置Struts2:

http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 配置Struts 2.0 -->  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <filter>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <filter-name>struts2</filter-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <init-param>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <param-name>actionPackages</param-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <param-value>com.rong.action</param-value>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </init-param>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </filter>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    <filter-mapping>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <filter-name>struts2</filter-name>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <url-pattern>/\*</url-pattern>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif    </filter-mapping>

        我们init了一个参数actionPackages,表示我们的请求到哪里寻找我们的action。举例说，我们的用户注册action="register"，它默认会在com.rong.action包下寻找RegisterAction类。经过action的处理后，若返回success，则会跳到success.jsp，若校验不通过，则返回input，则请求转发回register.jsp。  
        好了，看看运行效果吧！  
                  
        成功啦！我们基于Annotation的SSH整合开发终于完成了。我所介绍的Annotation注解其实并不多，还得靠大家找资料去学习。而我们侧重点是在用Annotation时三大框架是如何整合起来的。  
        说点题外话吧，我们在开发过程中，究竟用Annotation还是XML，许多人曾问这样的问题。有培训机构的老师说过一句话，Annotation可以用，但XML配置一定要懂，因为XML配置现在还是主流。他说得也挺有道理的。有些人可能还不能接受Annotation这东西，就像我当初不大喜欢它一样。我的一位教我们Java中间件的老教师说：“Struts,Spring,Hibernate算什么技术？学EJB才是有用的。”他教我们的好像是EJB2版本的，他年轻时用Jbuilder开发过EJB的项目。如果我问他“为什么不选EJB3.0”，他会不会说：“EJB3被改得一塌糊涂，什么注解啊，改得狗屁不通，还是EJB2的最好……”。当然，我没有再问他。有时候，新事物要被人们所接受，是要经历一段时间的，世界在变，知识在变，我们何不用更宽广的心态，去接受那些新生的事物？

[**Hibernate+Spring+Struts2整合开发中的一个分页显示方案**](http://www.blogjava.net/rongxh7/archive/2008/11/29/243456.html)

      分页显示一直是web开发中一大烦琐的难题，传统的网页设计只在一个JSP或者ASP页面中书写所有关于数据库操作的代码，那样做分页可能简单一点，但当把网站分层开发后，分页就比较困难了，下面是我做Spring+Hibernate+Struts2项目时设计的分页代码，与大家分享交流。  
1、DAO层接口的设计，在MemberDao接口中定义了如下两个方法：

http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic interface MemberDao{  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    //省略了其他的代码  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 分页查询  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param hql 查询的条件  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param offset 开始记录  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param length 一次查询几条记录  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public List queryForPage(final String hql,final int offset,final int length);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 查询所有记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param hql 查询的条件  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return 总记录数  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public int getAllRowCount(String hql);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

2、DAO层实现类MemberDaoImpl对上面两个方法的实现如下：

http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class MemberDaoImpl extends HibernateDaoSupport implements MemberDao {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif       //省略了其他的代码      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 分页查询  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param hql 查询的条件  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param offset 开始记录  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param length 一次查询几条记录  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public List queryForPage(final String hql,final int offset,final int length){  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif        List list = getHibernateTemplate().executeFind(new HibernateCallback(){  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif            public Object doInHibernate(Session session) throws HibernateException,SQLException{  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                Query query = session.createQuery(hql);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                query.setFirstResult(offset);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                query.setMaxResults(length);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                List list = query.list();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                return list;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        });  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return list;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 查询所有记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return 总记录数  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public int getAllRowCount(String hql){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return getHibernateTemplate().find(hql).size();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

        细心的读者会发现，这个类继承了HibernateDaoSupport类，HibernateDaoSupport是Spring提供的对Hibernate支持的类，getHibernateTemplate().executeFind(new HibernateCallback(){....})方法中的参数，我们使用了接口回调，在其参数内，我们能像原生的Hibernate一样调用query.setFirstResult(offset)和query.setMaxResults(length)来实现分页查询功能。  
  
3、下面我们来新建一个保存分页信息的类PageBean,具体代码如下：

http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class PageBean {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private List list;        //要返回的某一页的记录列表  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private int allRow;         //总记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private int totalPage;        //总页数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private int currentPage;    //当前页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private int pageSize;        //每页记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private boolean isFirstPage;    //是否为第一页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private boolean isLastPage;        //是否为最后一页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private boolean hasPreviousPage;    //是否有前一页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private boolean hasNextPage;        //是否有下一页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public List getList() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return list;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setList(List list) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.list = list;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public int getAllRow() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return allRow;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setAllRow(int allRow) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.allRow = allRow;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public int getTotalPage() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return totalPage;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setTotalPage(int totalPage) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.totalPage = totalPage;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public int getCurrentPage() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return currentPage;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setCurrentPage(int currentPage) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.currentPage = currentPage;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public int getPageSize() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return pageSize;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setPageSize(int pageSize) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.pageSize = pageSize;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 初始化分页信息  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void init(){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.isFirstPage = isFirstPage();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.isLastPage = isLastPage();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.hasPreviousPage = isHasPreviousPage();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.hasNextPage = isHasNextPage();  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 以下判断页的信息,只需getter方法(is方法)即可  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public boolean isFirstPage() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return currentPage == 1;    // 如是当前页是第1页  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public boolean isLastPage() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return currentPage == totalPage;    //如果当前页是最后一页  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public boolean isHasPreviousPage() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return currentPage != 1;        //只要当前页不是第1页  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public boolean isHasNextPage() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return currentPage != totalPage;    //只要当前页不是最后1页  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 计算总页数,静态方法,供外部直接通过类名调用  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param pageSize 每页记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param allRow 总记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return 总页数  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public static int countTotalPage(final int pageSize,final int allRow){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        int totalPage = allRow % pageSize == 0 ? allRow/pageSize : allRow/pageSize+1;  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return totalPage;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 计算当前页开始记录  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param pageSize 每页记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param currentPage 当前第几页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return 当前页开始记录号  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public static int countOffset(final int pageSize,final int currentPage){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        final int offset = pageSize\*(currentPage-1);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return offset;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 计算当前页,若为0或者请求的URL中没有"?page=",则用1代替  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param page 传入的参数(可能为空,即0,则返回1)  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return 当前页  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public static int countCurrentPage(int page){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        final int curPage = (page==0?1:page);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return curPage;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

4、Service层接口的设计：

http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic interface MemberService {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    //省略其他的代码  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 分页查询  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param currentPage 当前第几页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param pageSize 每页大小  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return 封闭了分页信息(包括记录集list)的Bean  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public PageBean queryForPage(int pageSize,int currentPage);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

5、Service层实现类的部分内码如下：

http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class MemberServiceImpl implements MemberService {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  //通过applicationContext.xml配置文件注入MemberDao的值  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private MemberDao memberDao;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setMemberDao(MemberDao memberDao) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.memberDao = memberDao;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    /\*\*  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 分页查询  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param currentPage 当前第几页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param pageSize 每页大小  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @return 封闭了分页信息(包括记录集list)的Bean  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public PageBean queryForPage(int pageSize,int page){  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        final String hql = "from Member";        //查询语句  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        int allRow = memberDao.getAllRowCount(hql);    //总记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        int totalPage = PageBean.countTotalPage(pageSize, allRow);    //总页数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        final int offset = PageBean.countOffset(pageSize, page);    //当前页开始记录  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        final int length = pageSize;    //每页记录数  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        final int currentPage = PageBean.countCurrentPage(page);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        List<Member> list = memberDao.queryForPage(hql,offset, length);        //"一页"的记录  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif          
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        //把分页信息保存到Bean中  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        PageBean pageBean = new PageBean();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        pageBean.setPageSize(pageSize);      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        pageBean.setCurrentPage(currentPage);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        pageBean.setAllRow(allRow);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        pageBean.setTotalPage(totalPage);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        pageBean.setList(list);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        pageBean.init();  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return pageBean;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }

6、在Struts2中调用queryForPageMemberService层的queryForPage（）方法即可return一个包含分页信息、符合条件的结果集list， 代码如下：

http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class ListMember extends ActionSupport{  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    //通过applicationContext.xml配置文件注入memberService的值  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private MemberService memberService;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setMemberService(MemberService memberService) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.memberService = memberService;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private int page;    //第几页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private PageBean pageBean;    //包含分布信息的bean  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public int getPage() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return page;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setPage(int page) {        //若URL中无此参数,会默认为第1页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.page = page;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public PageBean getPageBean() {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return pageBean;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public void setPageBean(PageBean pageBean) {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.pageBean = pageBean;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Override  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gif    public String execute() throws Exception {  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        //分页的pageBean,参数pageSize表示每页显示记录数,page为当前页  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.pageBean = memberService.queryForPage(2, page);  
http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return SUCCESS;  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}

7、最后在listMember.jsp页面中，用到了Struts2标签：

http://www.blogjava.net/Images/OutliningIndicators/None.gif<s:iterator value="pageBean.list">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <s:property value="title"/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <a href="getArticle.action?id=<s:property value="id"/>">modify</a>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <a href="deleteArticle.action?id=<s:property value="id"/>" onclick="return askDel()"/>delete</a><br/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </s:iterator>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        共<s:property value="pageBean.allRow"/> 条记录  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        共<s:property value="pageBean.totalPage"/> 页  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        当前第<s:property value="pageBean.currentPage"/>页<br/>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif          
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <s:if test="%{pageBean.currentPage == 1}">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            第一页 上一页  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </s:if>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <s:else>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <a href="listMyArticle.action?page=1">第一页</a>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <a href="listMyArticle.action?page=<s:property value="%{pageBean.currentPage-1}"/>">上一页</a>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </s:else>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <s:if test="%{pageBean.currentPage != pageBean.totalPage}">  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <a href="listMyArticle.action?page=<s:property value="%{pageBean.currentPage+1}"/>">下一页</a>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            <a href="listMyArticle.action?page=<s:property value="pageBean.totalPage"/>">最后一页</a>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </s:if>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        <s:else>  
http://www.blogjava.net/Images/OutliningIndicators/None.gif            下一页 最后一页  
http://www.blogjava.net/Images/OutliningIndicators/None.gif        </s:else>

       到这里，Hibernate+Spring+Struts2整合开发中的分页问题就已经解决了，在我上述过程中，省略了许多Hibernate,Spring,Struts2的配置，那不是本文的重点，大家可以参考有关的书与资料，由于篇幅有限，在此就不一一列举。在以后的文章中，我也会详细地跟大家讲述SSH整合开发。