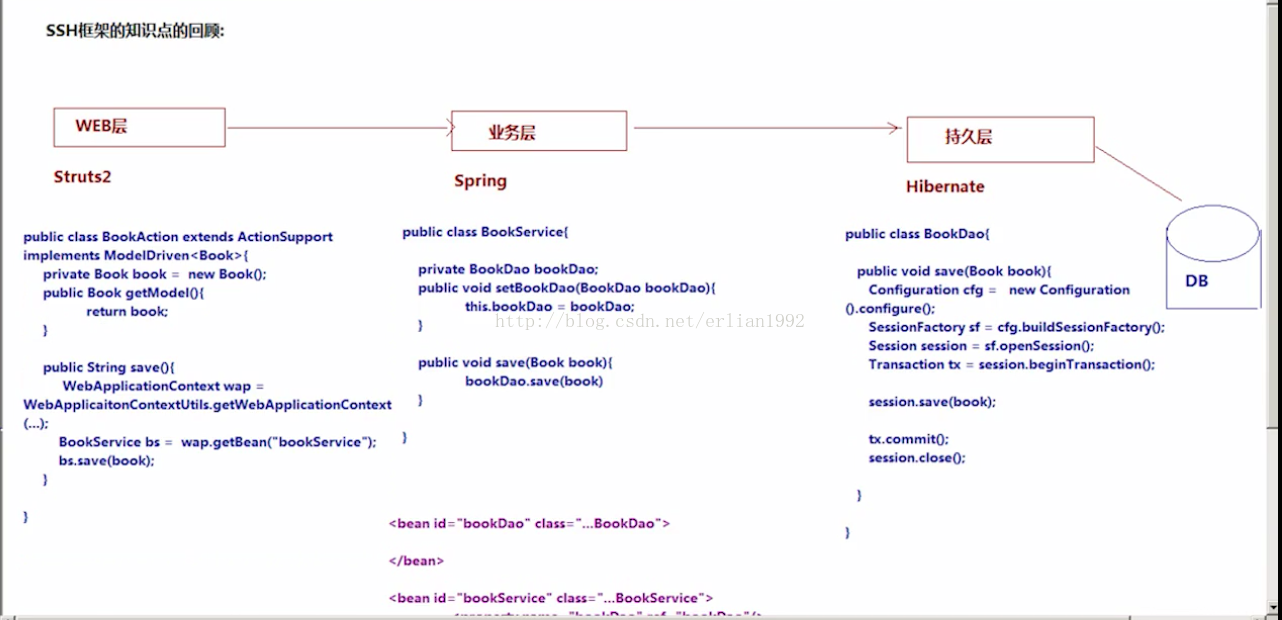
# [基于Struts2+HIbernate3+Spring3实现员工管理系统之框架整合篇](https://blog.csdn.net/erlian1992/article/details/54375833)

## 1.SSH知识点回顾

这个不用多说，老师画的一张图还是可以的：



## 2.搭建SSH开发环境

版本比较旧，但是方法还是好的，给初学者还是很大帮助的。SSH整合的三种方式：

1、带有HIbernate配置文件的方式

2、不带有HIbernate配置文件的方式

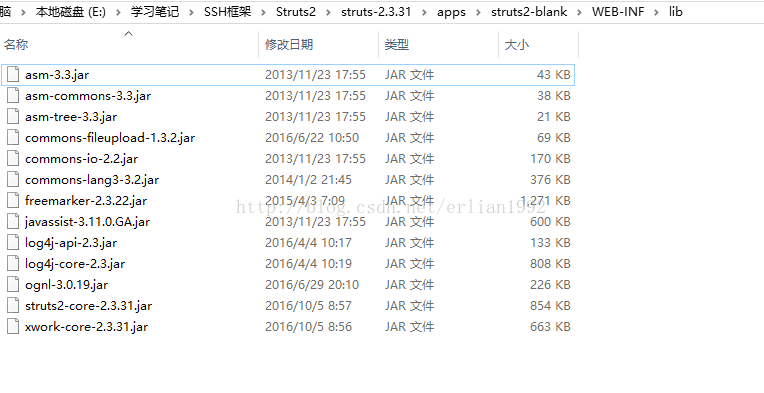
3、纯注解的

这一次我们采用第二种方式来进行SSH框架整合。

### 步骤一：创建web项目并引入相应的jar包

#### Struts2框架开发的相应的jar包

一般我们都去这个路径下struts-2.3.31\apps\struts2-blank\WEB-INF\lib索取开发的jar包：



然后再去struts-2.3.31\lib索取其它比较重要的jar包：

struts2-convention-plugin-2.3.31.jar：Struts2的注解开发的jar包

struts2-spring-plugin-2.3.31.jar：Struts2用于整合Spring的jar包

#### Hibernate框架开发的相应的jar包

我们使用的Hibernate版本是hibernate-release-3.3.1.Final。

hibernate-release-3.3.1.Final\lib\required\\*.jar

hibernate-release-3.3.1.Final\lib\jpa\\*.jar

#### Spring框架开发的相应的jar包

spring-framework-3.2.0.RELEASE

做Spring基本开发需要引入的jar包：

IoC

spring-beans-3.2.0.RELEASE.jar

spring-context-3.2.0.RELEASE.jar

spring-core-3.2.0.RELEASE.jar

spring-expression-3.2.0.RELEASE.jar

依赖的jar包

com.springsource.org.apache.log4j-1.2.15.jar

com.springsource.org.apache.commons.logging-1.1.1.jar

AOP

spring-aop-3.2.0.RELEASE.jar

spring-aspects-3.2.0.RELEASE.jar

依赖的jar包

com.springsource.org.aopalliance-1.0.0.jar

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

事务管理的jar包

spring-tx-3.2.0.RELEASE.jar

整合HIbernate的包

spring-orm-3.2.0.RELEASE.jar

spring-jdbc-3.2.0.RELEASE.jar

整合Web项目

spring-web-3.2.0.RELEASE.jar

整合Junit单元测试

spring-test-3.2.0.RELEASE.jar

连接池：

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

有关日志记录的jar包

slf4j-log4j12-1.7.5.jar： slf4j整合log4j的jar包

commons-logging-1.1.3.jar

mysql-connector-java-5.1.22-bin.jar：MySQL数据库驱动包

### 步骤二：引入相关配置文件

这里将已经配置好的配置文件贴出来，后面很多的配置连接池和事务管理以及注入方式步骤省略。

WebContent/WEB-INF/web.xml配置文件的内容：

1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4. xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5. http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"
6. id="WebApp\_ID"
7. version="3.1">
8. <display-name>struts2\_spring3\_hibernate3</display-name>
10. <welcome-file-list>
11. <welcome-file>index.jsp</welcome-file>
12. </welcome-file-list>
14. <filter>
15. <filter-name>struts2</filter-name>
16. <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
17. </filter>
19. <filter-mapping>
20. <filter-name>struts2</filter-name>
21. <url-pattern>/\*</url-pattern>
22. </filter-mapping>
24. <listener>
25. <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
26. </listener>
28. <context-param>
29. <param-name>contextConfigLocation</param-name>
30. <param-value>classpath:applicationContext.xml</param-value>
31. </context-param>
33. </web-app>

struts.xml配置文件

在src目录下引入struts.xml

1. <?xml version="1.0" encoding="UTF-8" ?>
2. <!DOCTYPE struts PUBLIC
3. "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
4. "http://struts.apache.org/dtds/struts-2.1.dtd">

7. <struts>
8. <span style="white-space:pre;"> </span><package name="ssh" namespace="/" extends="struts-default">
9. <span style="white-space:pre;">     </span><action name="product\_\*" class="productAction" method="{1}">
10. <span style="white-space:pre;">     </span></action>
11. <span style="white-space:pre;"> </span></package>
12. </struts>

src/applicationContext.xml配置文件的内容：

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:aop="http://www.springframework.org/schema/aop"
6. xmlns:tx="http://www.springframework.org/schema/tx"
7. xsi:schemaLocation="http://www.springframework.org/schema/beans
8. http://www.springframework.org/schema/beans/spring-beans.xsd
9. http://www.springframework.org/schema/context
10. http://www.springframework.org/schema/context/spring-context.xsd
11. http://www.springframework.org/schema/aop
12. http://www.springframework.org/schema/aop/spring-aop.xsd
13. http://www.springframework.org/schema/tx
14. http://www.springframework.org/schema/tx/spring-tx.xsd"><span style="white-space:pre;">  </span>
15. <span style="white-space:pre;"> </span>
16. <span style="white-space:pre;"> </span><!-- 引入外部的属性文件 -->
17. <span style="white-space:pre;"> </span><context:property-placeholder location="classpath:jdbc.properties"/>
18. <span style="white-space:pre;"> </span>
19. <span style="white-space:pre;"> </span><!-- 配置连接池 -->
20. <span style="white-space:pre;"> </span><bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
21. <span style="white-space:pre;">     </span><property name="driverClass" value="${jdbc.driverClass}"></property>
22. <span style="white-space:pre;">     </span><property name="jdbcUrl" value="${jdbc.url}"></property>
23. <span style="white-space:pre;">     </span><property name="user" value="${jdbc.username}"></property>
24. <span style="white-space:pre;">     </span><property name="password" value="${jdbc.password}"></property>
25. <span style="white-space:pre;"> </span></bean>
26. <span style="white-space:pre;"> </span>
27. <span style="white-space:pre;"> </span><!-- 配置Hibernate的相关属性 -->
28. <span style="white-space:pre;"> </span><bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
29. <span style="white-space:pre;">     </span><!-- 注入连接池 -->
30. <span style="white-space:pre;">     </span><property name="dataSource" ref="dataSource"></property>
31. <span style="white-space:pre;">     </span><!-- 加载Hibernate相关属性 -->
32. <span style="white-space:pre;">     </span><property name="hibernateProperties">
33. <span style="white-space:pre;">         </span><props>
34. <span style="white-space:pre;">             </span><prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
35. <span style="white-space:pre;">             </span><prop key="hibernate.show\_sql">true</prop>
36. <span style="white-space:pre;">             </span><prop key="hibernate.format\_sql">true</prop>
37. <span style="white-space:pre;">             </span><prop key="hibernate.hbm2ddl.auto">update</prop>
38. <span style="white-space:pre;">         </span></props>
39. <span style="white-space:pre;">     </span></property>
40. <span style="white-space:pre;">     </span><!-- 加载hibernate中的映射文件 -->
41. <span style="white-space:pre;">     </span><property name="mappingResources">
42. <span style="white-space:pre;">         </span><list>
43. <span style="white-space:pre;">             </span><value>com/demo/domain/Product.hbm.xml</value>
44. <span style="white-space:pre;">         </span></list>
45. <span style="white-space:pre;">     </span></property>
46. <span style="white-space:pre;"> </span></bean>
47. <span style="white-space:pre;"> </span>
48. <span style="white-space:pre;"> </span><!-- 配置Action的类 -->
49. <span style="white-space:pre;"> </span><bean id="productAction" class="com.demo.action.ProductAction" scope="prototype">
50. <span style="white-space:pre;">     </span><property name="productService" ref="productService"></property>
51. <span style="white-space:pre;"> </span></bean>
52. <span style="white-space:pre;"> </span>
53. <span style="white-space:pre;"> </span><!-- 配置业务层的类 -->
54. <span style="white-space:pre;"> </span><bean id="productService" class="com.demo.service.ProductService">
55. <span style="white-space:pre;">     </span><property name="productDao" ref="productDao"></property>
56. <span style="white-space:pre;"> </span></bean>
57. <span style="white-space:pre;"> </span>
58. <span style="white-space:pre;"> </span><!-- 配置DAO的类 -->
59. <span style="white-space:pre;"> </span><bean id="productDao" class="com.demo.dao.ProductDao">
60. <span style="white-space:pre;">     </span><property name="sessionFactory" ref="sessionFactory"></property>
61. <span style="white-space:pre;"> </span></bean><span style="white-space:pre;">     </span>
62. <span style="white-space:pre;"> </span>
63. <span style="white-space:pre;"> </span><!-- 配置事务管理 -->
64. <span style="white-space:pre;"> </span><bean id="transaction" class="org.springframework.orm.hibernate3.HibernateTransactionManager">
65. <span style="white-space:pre;">     </span><property name="sessionFactory" ref="sessionFactory"></property>
66. <span style="white-space:pre;"> </span></bean>
67. <span style="white-space:pre;"> </span>
68. <span style="white-space:pre;"> </span><!-- 开启注解事务 -->
69. <span style="white-space:pre;"> </span><tx:annotation-driven transaction-manager="transaction"></tx:annotation-driven>
70. <span style="white-space:pre;"> </span>
71. </beans>

 src/jdbc.properties属性配置文件的内容：

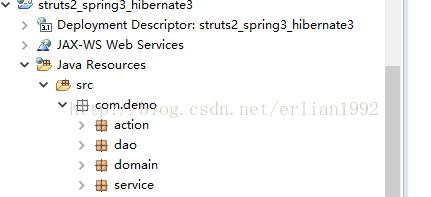
1. jdbc.driverClass=com.mysql.jdbc.Driver
2. jdbc.url=jdbc:mysql://localhost:3306/ssh
3. jdbc.username=root
4. jdbc.password=root

       src/log4j.properties属性配置文件的内容：

1. log4j.rootLogger = debug, console, E
2. #level=INFO,all can be output
3. #console is set to be a ConsoleAppender
4. log4j.appender.console = org.apache.log4j.ConsoleAppender
5. log4j.appender.console.layout = org.apache.log4j.PatternLayout
6. log4j.appender.console.layout.ConversionPattern = [%p] %d{yyyy-MM-dd HH:mm:ss} - %m%n

9. #file is set to output to a extra file
10. log4j.appender.E = org.apache.log4j.DailyRollingFileAppender
11. log4j.appender.E.layout = org.apache.log4j.PatternLayout
12. log4j.appender.E.Threshold=error
13. log4j.appender.E.Encoding=UTF-8
14. log4j.appender.E.layout.ConversionPattern=[%p] %d{yyyy-MM-dd HH:mm:ss} - %m%n
15. log4j.appender.E.DatePattern='.'yyyy-MM-dd
16. log4j.appender.E.File = ${catalina.base}/logs/dd/error.log

### 步骤三：创建包结构



### 步骤四：创建页面（保存商品）

 addProduct.jsp

1. <%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
5. <html>
6. <head>
7. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8. <title>添加商品页面</title>
9. </head>
10. <body>
11. <h1>保存商品的页面</h1>
12. <s:form action="product\_save" method="post" theme="simple" namespace="/">
13. <table border="1" width="400">
14. <tr>
15. <td>商品名称:</td>
16. <td><s:textfield name="pname"></s:textfield></td>
17. </tr>
18. <tr>
19. <td>商品价格:</td>
20. <td><s:textfield name="price"></s:textfield></td>
21. </tr>
22. <tr>
23. <td colspan="2"><input type="submit" value="添加" /></td>
24. </tr>
25. </table>
26. </s:form>
27. </body>
28. </html>

### 步骤五：编写实体类、对象关系映射文件、Action、Service和DAO

Product.java

1. package com.demo.domain;
3. /\*\*
4. \* 商品实体类
5. \* @author Administrator
6. \* @date 2016年12月24日
7. \*/
8. public class Product {
9. private Integer pid;
10. private String pname;
11. private Double price;
13. public Integer getPid() {
14. return pid;
15. }
17. public void setPid(Integer pid) {
18. this.pid = pid;
19. }
21. public Double getPrice() {
22. return price;
23. }
25. public String getPname() {
26. return pname;
27. }
29. public void setPname(String pname) {
30. this.pname = pname;
31. }
33. public void setPrice(Double price) {
34. this.price = price;
35. }
37. }

       Product.hbm.xml

1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE hibernate-mapping PUBLIC
3. "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
4. "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
5. <hibernate-mapping>
6. <class name="com.demo.domain.Product" table="product">
7. <id name="pid" column="pid">
8. <generator class="native"></generator>
9. </id>
10. <property name="pname" column="pname" length="20"></property>
11. <property name="price" column="price"></property>
12. </class>
13. </hibernate-mapping>

       ProductAction.java

1. package com.demo.action;
3. import com.demo.domain.Product;
4. import com.demo.service.ProductService;
5. import com.opensymphony.xwork2.ActionSupport;
6. import com.opensymphony.xwork2.ModelDriven;
8. /\*\*
9. \* 商品管理的action类
10. \* @author Administrator
11. \* @date 2016年12月24日
12. \*/
13. @SuppressWarnings("serial")
14. public class ProductAction extends ActionSupport implements ModelDriven<Product>{
15. //模型驱动使用的类
16. private Product product = new Product();
18. @Override
19. public Product getModel() {
20. return product;
21. }
23. //Struts2和Spring整合过程中按名称注入的业务层的类
24. private ProductService productService;
26. public void setProductService(ProductService productService) {
27. this.productService = productService;
28. }
30. /\*\*
31. \* action保存商品的执行方法：save()
32. \* @return
33. \*/
34. public String save(){
35. System.out.println("Action中的save()方法执行了");
36. productService.save(product);
37. return NONE;
38. }
40. }

       ProductService.java

1. package com.demo.service;
3. import org.springframework.transaction.annotation.Transactional;
5. import com.demo.dao.ProductDao;
6. import com.demo.domain.Product;
8. /\*\*
9. \* 商品管理的service类
10. \* @author Administrator
11. \* @date 2016年12月24日
12. \*/
13. @Transactional
14. public class ProductService {
15. //业务层注入DAO的类
16. private ProductDao productDao;
18. public void setProductDao(ProductDao productDao) {
19. this.productDao = productDao;
20. }
22. /\*\*
23. \* 业务层保存商品的方法
24. \* @param product
25. \*/
26. public void save(Product product) {
27. System.out.println("Service中的save()方法执行了");
28. productDao.save(product);
29. }
31. }

       ProductDao.java

1. package com.demo.dao;
3. import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
5. import com.demo.domain.Product;
7. /\*\*
8. \* 商品管理的dao类
9. \* @author Administrator
10. \* @date 2016年12月24日
11. \*/
12. public class ProductDao extends HibernateDaoSupport {
14. /\*\*
15. \* DAO中保存商品的方法
16. \* @param product
17. \*/

public void save(Product product) { System.out.println("DAO中的save()方法执行了"); this.getHibernateTemplate().save(product); } }

### 步骤六：配置Action、Service和DAO

       Struts2和Spring整合的两种方式：

       1、Action的类由Struts2自身去创建  
       Action的配置：struts.xml

1. <package name="ssh" namespace="/" extends="struts-default">
2. action name="product\_\*" class="com.demo.action.ProductAction" method="{1}"></action>
3. </package>

       Service和DAO是如何配置的  
       直接在applicationContext.xml中进行配置

1. <!-- 配置业务层的类 -->
2. <bean id="productService" class="com.demo.service.ProductService">
3. <property name="productDao" ref="productDao"></property>
4. </bean>
6. <!-- 配置DAO的类 -->
7. <bean id="productDao" class="com.demo.dao.ProductDao" />

       2、Action的类交给Spring框架创建  
       Action、Service和DAO的配置：applicationContext.xml

1. <!-- 配置Action的类 -->
2. <bean id="productAction" class="com.demo.action.ProductAction" scope="prototype">
3. <property name="productService" ref="productService"></property>
4. </bean>
6. <!-- 配置业务层的类 -->
7. <bean id="productService" class="com.demo.service.ProductService">
8. <property name="productDao" ref="productDao"></property>
9. </bean>
11. <!-- 配置DAO的类 -->
12. <bean id="productDao" class="com.demo.dao.ProductDao">
13. <property name="sessionFactory" ref="sessionFactory"></property>
14. </bean>

       Action配置

1. <package name="ssh" namespace="/" extends="struts-default">
2. <action name="product\_\*" class="productAction" method="{1}">
3. </action>
4. lt;/package>

### 步骤七：创建数据库

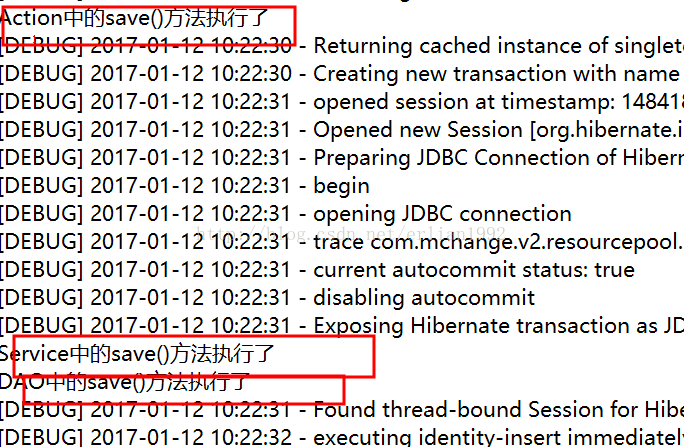
       create database ssh charset utf8;

### 步骤八：Spring整合Hibernate

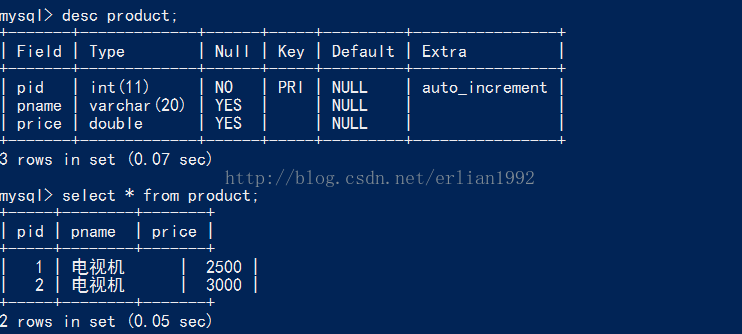
       在上面的Applicatiom.xml文件已经配置。不管是注入方式还是事务管理，在上面的步骤基本都包括了，就不再一一赘述。运行项目：



       控制台输出：



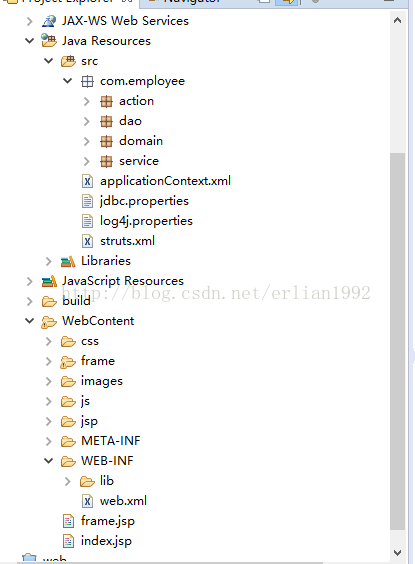
       数据库的数据：



       上述的流程基本通过，下一篇就是实现案例：开发员工管理系统（一个小的示例）

## 3.开发员工管理系统

由于上一篇已经搭建了SSH开发环境，这一篇就来完成案例。过程不再赘述。创建的项目目录为：



       具体实现

JSP页面

       1、index.jsp与frame.jsp及frame/\*

       index.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>
2. <%@ taglib uri="/struts-tags" prefix="s" %>
3. <%
4. String path = request.getContextPath();
5. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
6. %>
8. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
9. <html>
10. <head>
11. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12. <title>登陆</title>
13. <link type="text/css" rel="stylesheet" href="<%=basePath %>css/reset.css">
14. <link type="text/css" rel="stylesheet" href="<%=basePath %>css/main.css">
15. </head>
17. <body>
18. <div class="headerBar">
19. <div class="logoBar login\_logo">
20. <div class="comWidth">
21. <div class="logo fl">
22. <a href="#"><img src="<%=basePath %>images/logo.jpg" alt="慕课网" /></a>
23. </div>
24. <h3 class="welcome\_title">欢迎登陆</h3>
25. </div>
26. </div>
27. </div>
28. <s:form action="employee\_login" method="post" namespace="/" theme="simple">
29. <div class="loginBox">
30. <h3>
31. <!-- 显示错误信息 -->
32. <s:actionerror />
33. </h3>
34. <div class="login\_cont">
35. <ul class="login">
36. <li class="l\_tit">用户名：</li>
37. <li class="mb\_10"><input type="text" name="username" class="login\_input user\_icon"></li>
38. <li class="l\_tit">密码：</li>
39. <li class="mb\_10"><input type="text" name="password" class="login\_input user\_icon"></li>
40. <li><input type="submit" value="" class="login\_btn"></li>
41. </ul>
43. <div class="login\_partners">
44. <p class="l\_tit">使用合作方账号登陆网站</p>
45. <ul class="login\_list clearfix">
46. <li><a href="#">QQ</a></li>
47. <li><span>|</span></li>
48. <li><a href="#">网易</a></li>
49. <li><span>|</span></li>
50. <li><a href="#">新浪微博</a></li>
51. <li><span>|</span></li>
52. <li><a href="#">腾讯微薄</a></li>
53. <li><span>|</span></li>
54. <li><a href="#">新浪微博</a></li>
55. <li><span>|</span></li>
56. <li><a href="#">腾讯微薄</a></li>
57. </ul>
58. </div>
59. </div>
61. </div>
62. </s:form>
63. <div class="hr\_25"></div>
64. <div class="footer">
65. <p>
66. <a href="#">慕课简介</a><i>|</i>
67. <a href="#">慕课公告</a><i>|</i>
68. <a href="#">招纳贤士</a><i>|</i>
69. <a href="#">联系我们</a><i>|</i>
70. 客服热线：400-675-1234
71. </p>
72. <p>Copyright &copy; 2006 - 2014
73. 慕课版权所有
74. 京ICP备09037834号
75. 京ICP证B1034-8373号
76. 某市公安局XX分局备案编号：123456789123
77. </p>
78. <p class="web">
79. <a href="#"><img src="<%=basePath %>images/webLogo.jpg" alt="logo"></a>
80. <a href="#"><img src="<%=basePath %>images/webLogo.jpg" alt="logo"></a>
81. <a href="#"><img src="<%=basePath %>images/webLogo.jpg" alt="logo"></a>
82. <a href="#"><img src="<%=basePath %>images/webLogo.jpg" alt="logo"></a>
83. </p>
84. </div>
85. </body>
86. </html></span>

       frame.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%
4. String path = request.getContextPath();
5. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
6. %>
8. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
9. <html>
10. <head>
11. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12. <title>主页</title>
13. </head>
15. <frameset rows="80,\*">
16. <frame name="top" src="<%=basePath %>frame/top.jsp">
17. <frameset cols="150,\*" id="main">
18. <frame src="<%=basePath %>frame/left.jsp">
19. <frame name="right" src="<%=basePath %>frame/right.jsp">
20. </frameset>
21. </frameset>
22. </span>

       frame/top.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>

10. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
11. <html>
12. <head>
13. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14. <style type="text/css">
15. .div1 {
16. <span style="white-space:pre;"> </span>margin-top: 50px;
17. <span style="white-space:pre;"> </span>margin-left: 600px;
18. <span style="white-space:pre;"> </span>font-size: 14px;
19. <span style="white-space:pre;"> </span>color: white
20. }
21. </style>
22. </head>

25. <body bgcolor="#0099FF">
26. <span style="white-space:pre;"> </span><div class="div1">
27. <span style="white-space:pre;">     </span>欢迎您：<s:property value="#session.exsitEmploy.ename" />
28. <span style="white-space:pre;"> </span></div>
29. </body>
30. </html></span>

       frame/left.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. <link href="<%=basePath %>css/dtree.css" rel="stylesheet" type="text/css">
14. <script type="text/javascript" src="<%=basePath %>js/dtree.js"></script>
15. </head>
17. <body bgColor="#DDF0FB" leftMargin="0" topMargin="0" marginwidth="0" marginheight="0">
18. <table width="90%" border="0" cellspacing="1" cellpadding="2" align="center">
19. <div class="dtree">
20. <script type="text/javascript">
21. d = new dTree('d');
22. d.add('01', '-1', '员工管理系统');
23. d.add('0101', '01', '人力资源部');
24. d.add('010101', '0101', '部门管理', 'department\_findAll.action','', 'right');
25. d.add('010102', '0101', '员工管理', 'employee\_findAll.action','','right');
26. document.write(d);
27. </script>
28. </div>
29. </table>
30. </body>
31. </html></span>

       frame/right.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. </head>
15. <body>
16. <div align="center">
17. <table border="0" width="60%" cellspacing="0" cellpadding="0" id="table1">
18. <tr>
19. <td height="93"></td>
20. </tr>
21. <tr>
22. <td>
23. <img src="<%=basePath %>images/bg.jpg" style="background-repeat: no-repeat" width="1500px" height="600px">
24. </td>
25. </tr>
26. </table>
27. </div>
28. </body>
29. </html></span>

       2、jsp/department/\*

       listDepartment.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. <style type="text/css">
14. .table1 {
15. border: 1px solid #ddd;
16. width: 900px;
17. }
19. thead {
20. background-color: lightblue;
21. }
22. </style>
23. </head>
24. <body>
25. <table border="0" width="900px">
26. <tr>
27. <td align="center" style="font-size: 24px; color: #666">部门管理</td>
28. </tr>
29. <tr>
30. <td align="right">
31. <a href="department\_saveUI.action">添加</a>
32. </td>
33. </tr>
34. </table>
35. <br/>
36. <table cellspacing="0" border="1" class="table1">
37. <thead>
38. <tr>
39. <th width="300">部门名称</th>
40. <th width="300">编辑</th>
41. <th width="300">删除</th>
42. </tr>
43. </thead>
44. <tbody>
45. <s:iterator value="list" var="d">
46. <tr>
47. <td align="center"><s:property value="#d.dname" /></td>
48. <td align="center">
49. <a href="department\_editById.action?did=<s:property value="#d.did"/>">
50. <img src="<%=basePath %>images/编辑.png" />
51. </a>
52. </td>
53. <td align="center">
54. <a href="department\_delete.action?did=<s:property value="#d.did"/>">
55. <img src="<%=basePath %>images/trash.gif" />
56. </a>
57. </td>
58. </tr>
59. </s:iterator>
60. </tbody>
61. </table>
62. <br/>
63. <table border="0" cellspacing="0" cellpadding="0" width="900px">
64. <tr>
65. <td align="right">
66. <span>
67. 第<s:property value="currentPage" />/<s:property value="pageSize" />页
68. </span>
70. <span>
71. 总记录数：<s:property value="totalSize" />
72. 每页显示:<s:property value="pageCount" /></span>
73. <span>
74. <s:if test="currentPage != 1">
75. <a href="department\_findAll.action?currentPage=1">[首页]</a>
76. <a href="department\_findAll.action?currentPage=<s:property value="currentPage-1"/>">[上一页]</a>
77. </s:if>
78. <s:if test="currentPage != pageSize">
79. <a href="department\_findAll.action?currentPage=<s:property value="currentPage+1"/>">[下一页]</a>
80. <a href="department\_findAll.action?currentPage=<s:property value="pageSize"/>">[尾页]</a>
81. </s:if>
82. </span>
83. </td>
84. </tr>
85. </table>
86. </body>
87. </html></span>

       addDepartment.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. <title>添加部门</title>
14. </head>
16. <body>
17. <table border="0" width="600px">
18. <tr>
19. <td align="center" style="font-size: 24px; color: #666">部门添加</td>
20. </tr>
21. <tr>
22. <td align="right">
23. <a href="javascript:document.getElementById('saveForm').submit()">保存</a>
25. <a href="javascript:history.go(-1)">退回 </a>
26. </td>
27. </tr>
28. </table>
29. <br/>
30. <br/>
31. <!-- action对应一个action标签，id对应提交时的对应关系 -->
32. <s:form id="saveForm" action="department\_save" method="post" namespace="/" theme="simple">
33. <table style="font-size:16px"width="600px">
34. <tr>
35. <td width="30%" align="right">部门名称：</td>
36. <td><s:textfield name="dname" /></td>
37. </tr>
38. <tr>
39. <td width="30%" align="right">部门介绍：</td>
40. <td><s:textarea rows="5" cols="50" name="ddesc"></s:textarea></td>
41. </tr>
42. </table>
43. </s:form>
44. </body>
45. </html></span>

       editDepartment.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. </head>
15. <body>
16. <table border="0" width="600px">
17. <tr>
18. <td align="center" style="font-size: 24px; color: #666">编辑部门</td>
19. </tr>
20. <tr>
21. <td align="right">
22. <a href="javascript:document.getElementById('saveForm').submit()">保存</a>
24. <a href="javascript:history.go(-1)">退回 </a>
25. </td>
26. </tr>
27. </table>
28. <br/>
29. <br/>
30. <!-- action对应一个action标签，id对应提交时的对应关系 -->
31. <s:form id="saveForm" action="department\_update" method="post" namespace="/" theme="simple">
32. <!-- 隐藏域，指明修改的是哪个部门 -->
33. <s:hidden name="did" value="%{model.did}"></s:hidden>
34. <table style="font-size:16px" width="600px">
35. <tr>
36. <td width="30%" align="right">部门名称：</td>
37. <td><s:textfield name="dname" value="%{model.dname}" /></td>
38. </tr>
39. <tr>
40. <td width="30%" align="right">部门介绍：</td>
41. <td></td>
42. </tr>
43. <tr>
44. <td width="30%" align="right"></td>
45. <td><s:textarea rows="5" cols="50" name="ddesc" value="%{model.ddesc}"></s:textarea></td>
46. </tr>
47. </table>
48. </s:form>
49. </body>
50. </html> </span>

       3、jsp/employee/\*

       listEpartment.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. <style type="text/css">
14. .table1 {
15. border: 1px solid #ddd;
16. width: 900px;
17. }
19. thead {
20. background-color: lightblue;
21. }
22. </style>
23. </head>
24. <body>
25. <table border="0" width="900px">
26. <tr>
27. <td align="center" style="font-size: 24px; color: #666">员工管理</td>
28. </tr>
29. <tr>
30. <td align="right">
31. <a href="employee\_saveUI.action">添加</a>
32. </td>
33. </tr>
34. </table>
35. <br />
36. <table cellspacing="0" border="1" class="table1">
37. <thead>
38. <tr>
39. <th width="300">员工编号</th>
40. <th width="300">员工姓名</th>
41. <th width="100">性别</th>
42. <th width="500">出生日期</th>
43. <th width="500">入职时间</th>
44. <th width="300">所在部门</th>
45. <th width="300">编辑</th>
46. <th width="300">删除</th>
47. </tr>
48. </thead>
49. <tbody>
50. <s:iterator value="list" var="e">
51. <tr>
52. <td align="center"><s:property value="#e.eno" /></td>
53. <td align="center"><s:property value="#e.ename" /></td>
54. <td align="center"><s:property value="#e.sex" /></td>
55. <td align="center"><s:date format="yyyy-MM-dd" name="#e.birthday" /></td>
56. <td align="center"><s:date name="#e.joinDay" format="yyyy-MM-dd" /></td>
57. <td align="center"><s:property value="#e.department.dname" /></td>
58. <td align="center">
59. <a href="employee\_edit.action?eid=<s:property value="#e.eid"/>">
60. <img src="<%=basePath %>images/编辑.png" />
61. </a>
62. </td>
63. <td align="center">
64. <a href="employee\_delete.action?eid=<s:property value="#e.eid"/>">
65. <img src="<%=basePath %>images/trash.gif" />
66. </a>
67. </td>
68. </tr>
69. </s:iterator>
70. </tbody>
71. </table>
72. <br/>
73. <table border="0" cellspacing="0" cellpadding="0" width="900px">
74. <tr>
75. <td align="right">
76. <span>
77. 第<s:property value="currentPage" />/<s:property value="pageSize" />页
78. </span>
80. <span>
81. 总记录数：<s:property value="totalSize" />
82. 每页显示:<s:property value="pageCount" /></span>
83. <span>
84. <s:if test="currentPage != 1">
85. <a href="employee\_findAll.action?currentPage=1">[首页]</a>
86. <a href="employee\_findAll.action?currentPage=<s:property value="currentPage-1"/>">[上一页]</a>
87. </s:if>
88. <s:if test="currentPage != pageSize">
89. <a href="employee\_findAll.action?currentPage=<s:property value="currentPage+1"/>">[下一页]</a>
90. <a href="employee\_findAll.action?currentPage=<s:property value="pageSize"/>">[尾页]</a>
91. </s:if>
92. </span>
93. </td>
94. </tr>
95. </table>
96. </body>
97. </html></span>

       addEmployee.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. </head>
15. <body>
16. <table border="0" width="600px">
17. <tr>
18. <td align="center" style="font-size: 24px; color: #666">员工添加</td>
19. </tr>
20. <tr>
21. <td align="right">
22. <a href="javascript:document.getElementById('saveForm').submit()">保存</a>
24. <a href="javascript:history.go(-1)">退回 </a>
25. </td>
26. </tr>
27. </table>
28. <br/>
29. <br/>
30. <!-- action对应一个action标签，id对应提交时的对应关系 -->
31. <s:form id="saveForm" action="employee\_save" method="post" namespace="/" theme="simple">
32. <table style="font-size:16px">
33. <tr>
34. <td align="right">员工姓名：</td>
35. <td><s:textfield name="ename" /></td>
36. <td align="right">性别：</td>
37. <td><s:radio list="{'男','女'}" name="sex" /></td>
38. </tr>
39. <tr>
40. <td align="right">出生日期：</td>
41. <td align="right"><s:textfield name="birthday" /></td>
42. <td align="right">入职日期：</td>
43. <td><s:textfield name="joinDay" /></td>
44. </tr>
45. <tr>
46. <td align="right">用户名：</td>
47. <td><s:textfield name="username" /></td>
48. <td align="right">密码：</td>
49. <td><s:password name="password" /></td>
50. </tr>
51. <tr>
52. <td align="right">编号：</td>
53. <td><s:textfield name="eno" /></td>
54. <td align="right">部门：</td>
55. <td><s:select name="department.did" list="list" listKey="did" listValue="dname" headerKey="0" headerValue="--请选择--" /></td>
56. </tr>
57. </table>
58. </s:form>
59. </body>
60. </html> </span>

       editEmployee.jsp

1. <span style="font-size:18px;"><%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib uri="/struts-tags" prefix="s" %>
4. <%
5. String path = request.getContextPath();
6. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
7. %>
9. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10. <html>
11. <head>
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13. </head>
15. <body>
16. <table border="0" width="600px">
17. <tr>
18. <td align="center" style="font-size: 24px; color: #666">员工修改</td>
19. </tr>
20. <tr>
21. <td align="right">
22. <a href="javascript:document.getElementById('saveForm').submit()">保存</a>
24. <a href="javascript:history.go(-1)">退回 </a>
25. </td>
26. </tr>
27. </table>
28. <br/>
29. <br/>
30. <!-- action对应一个action标签，id对应提交时的对应关系 -->
31. <s:form id="saveForm" action="employee\_update" method="post" namespace="/" theme="simple">
32. <s:hidden name="eid" value="%{model.eid}"></s:hidden>
33. <table style="font-size: :16px">
34. <tr>
35. <td align="right">员工姓名：</td>
36. <td><s:textfield value="%{model.ename}" name="ename" /></td>
37. <td align="right">性别：</td>
38. <td><s:radio list="{'男','女'}" value="%{model.sex}" name="sex" /></td>
39. </tr>
40. <tr>
41. <td align="right">出生日期：</td>
42. <td><input type="text" name="birthday" value="<s:date name="model.birthday" format="yyyy-MM-dd"/>" /></td>
43. <td align="right">入职日期：</td>
44. <td><input type="text" name="joinDay" value="<s:date name="model.joinDay" format="yyyy-MM-dd"/>" /></td>
45. </tr>
46. <tr>
47. <td align="right">用户名：</td>
48. <td><s:textfield value="%{model.username}" name="username" /></td>
49. <td align="right">密码：</td>
50. <td><s:password value="%{model.password}" name="password" showPassword="true"/></td>
51. </tr>
52. <tr>
53. <td align="right">编号：</td>
54. <td><s:textfield value="%{model.eno}" name="eno" /></td>
55. <td align="right">部门：</td>
56. <td>
57. <s:select name="department.did" headerKey="0" value="%{model.department.did}" headerValue="--请选择--" list="list" listKey="did" listValue="dname" />
58. </td>
59. </tr>
60. </table>
61. </s:form>
62. </body>
63. </html> </span>

       实体类和对象关系映射文件

       Depaerment实体类

1. <span style="font-size:18px;">package com.employee.domain;
3. import java.util.HashSet;
4. import java.util.Set;
6. /\*\*
7. \* 部门实体类
8. \* @author Administrator
9. \* @date 2016年12月24日
10. \*/
11. public class Department {
12. private Integer did;
13. private String dname;
14. private String ddesc;
15. //部门员工的集合
16. private Set<Employee> employee = new HashSet<Employee>();
18. public Integer getDid() {
19. return did;
20. }
22. public void setDid(Integer did) {
23. this.did = did;
24. }
26. public String getDname() {
27. return dname;
28. }
30. public void setDname(String dname) {
31. this.dname = dname;
32. }
34. public String getDdesc() {
35. return ddesc;
36. }
38. public void setDdesc(String ddesc) {
39. this.ddesc = ddesc;
40. }
42. public Set<Employee> getEmployee() {
43. return employee;
44. }
46. public void setEmployee(Set<Employee> employee) {
47. this.employee = employee;
48. }
50. }</span>

       Department.hbm.xml

1. <span style="font-size:18px;"><?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE hibernate-mapping PUBLIC
3. "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
4. "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
5. <hibernate-mapping>
6. <class name="com.employee.domain.Department" table="department">
7. <id name="did" column="did">
8. <generator class="native"></generator>
9. </id>
11. <property name="dname" column="dname" length="20"></property>
12. <property name="ddesc" column="ddesc" length="100"></property>
14. <!-- 一对多关联映射 -->
15. <set name="employee" cascade="all" inverse="true">
16. <key column="dno" />
17. <one-to-many class="com.employee.domain.Employee"/>
18. </set>
20. </class>
21. </hibernate-mapping></span>

       Employee实体类

1. <span style="font-size:18px;">package com.employee.domain;
3. import java.util.Date;
5. /\*\*
6. \* 员工实体
7. \* @author Administrator
8. \* @date 2016年12月24日
9. \*/
10. public class Employee {
11. private Integer eid;
12. private String ename;
13. private String sex;
14. private Date birthday;
15. private Date joinDay;
16. private String eno;
17. private String username;
18. private String password;
19. //员工的所属部门
20. private Department department;
22. public Integer getEid() {
23. return eid;
24. }
26. public void setEid(Integer eid) {
27. this.eid = eid;
28. }
30. public String getEname() {
31. return ename;
32. }
34. public void setEname(String ename) {
35. this.ename = ename;
36. }
38. public String getSex() {
39. return sex;
40. }
42. public void setSex(String sex) {
43. this.sex = sex;
44. }
46. public Date getBirthday() {
47. return birthday;
48. }
50. public void setBirthday(Date birthday) {
51. this.birthday = birthday;
52. }
54. public Date getJoinDay() {
55. return joinDay;
56. }
58. public void setJoinDay(Date joinDay) {
59. this.joinDay = joinDay;
60. }
61. public String getEno() {
62. return eno;
63. }
65. public void setEno(String eno) {
66. this.eno = eno;
67. }
69. public String getUsername() {
70. return username;
71. }
73. public void setUsername(String username) {
74. this.username = username;
75. }
77. public String getPassword() {
78. return password;
79. }
81. public void setPassword(String password) {
82. this.password = password;
83. }
85. public Department getDepartment() {
86. return department;
87. }
89. public void setDepartment(Department department) {
90. this.department = department;
91. }
93. }
94. </span>

       Employee.hbm.xml

1. <span style="font-size:18px;"><?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE hibernate-mapping PUBLIC
3. "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
4. "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
6. <hibernate-mapping>
7. <class name="com.employee.domain.Employee" table="employee">
8. <id name="eid" column="eid">
9. <generator class="native" />
10. </id>
12. <property name="ename" column="ename" length="20"/>
13. <property name="sex" column="sex" length="10" />
14. <property name="birthday" column="birthday" type="date" />
15. <property name="joinDay" column="joinDay" type="date" />
16. <property name="eno" column="eno" length="20" />
17. <property name="username" column="username" length="20" />
18. <property name="password" column="password" length="20" />
20. <!-- 多对一关联映射 -->
21. <many-to-one name="department" class="com.employee.domain.Department" cascade="save-update">
22. <column name="dno" />
23. </many-to-one>
25. </class>
26. </hibernate-mapping></span>

       PageBean实现类

1. <span style="font-size:18px;">package com.employee.domain;
3. import java.util.List;
5. /\*\*
6. \* 分页Bean
7. \* @author Administrator
8. \* @date 2016年12月24日
9. \* @param <T>
10. \*/
11. public class PageBean<T>{
12. private Integer currentPage;//当前页数
13. private Integer pageCount;//每页显示记录数
14. private Integer pageSize;//总页数
15. private Integer totalSize;//总记录数
16. private List<T> list;     //每页显示的数据
18. public Integer getCurrentPage() {
19. return currentPage;
20. }
22. public void setCurrentPage(Integer currentPage) {
23. this.currentPage = currentPage;
24. }
26. public Integer getPageCount() {
27. return pageCount;
28. }
30. public void setPageCount(Integer pageCount) {
31. this.pageCount = pageCount;
32. }
34. public Integer getPageSize() {
35. return pageSize;
36. }
38. public void setPageSize(Integer pageSize) {
39. this.pageSize = pageSize;
40. }
42. public Integer getTotalSize() {
43. return totalSize;
44. }
46. public void setTotalSize(Integer totalSize) {
47. this.totalSize = totalSize;
48. }
50. public List<T> getList() {
51. return list;
52. }
54. public void setList(List<T> list) {
55. this.list = list;
56. }
58. }</span>

       web层实现

       1、web.xml文件

1. <span style="font-size:18px;"><?xml version="1.0" encoding="UTF-8"?>
2. <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">
3. <display-name>ssh\_employee</display-name>
5. <welcome-file-list>
6. <welcome-file>index.jsp</welcome-file>
7. </welcome-file-list>
9. <filter>
10. <filter-name>openSessionInViewFilter</filter-name>
11. <filter-class>org.springframework.orm.hibernate3.support.OpenSessionInViewFilter</filter-class>
12. </filter>
13. <filter-mapping>
14. <filter-name>openSessionInViewFilter</filter-name>
15. <url-pattern>\*.action</url-pattern>
16. </filter-mapping>
18. <filter>
19. <filter-name>struts2</filter-name>
20. <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
21. </filter>
22. <filter-mapping>
23. <filter-name>struts2</filter-name>
24. <url-pattern>/\*</url-pattern>
25. </filter-mapping>
27. <listener>
28. <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
29. </listener>
30. <context-param>
31. <param-name>contextConfigLocation</param-name>
32. <param-value>classpath:applicationContext.xml</param-value>
33. </context-param>
35. </web-app></span>

       2、Action

       DepartmentAction类

1. <span style="font-size:18px;">package com.employee.action;
3. import com.employee.domain.Department;
4. import com.employee.domain.PageBean;
5. import com.employee.service.DepartmentService;
6. import com.opensymphony.xwork2.ActionContext;
7. import com.opensymphony.xwork2.ActionSupport;
8. import com.opensymphony.xwork2.ModelDriven;
10. /\*\*
11. \* 部门action
12. \* @author Administrator
13. \* @date 2016年12月24日
14. \*/
15. @SuppressWarnings("serial")
16. public class DepartmentAction extends ActionSupport implements ModelDriven<Department>{
18. //struts结合spring自动注入的department
19. private Department department = new Department();
21. public Department getModel() {
22. return department;
23. }
25. //spring注入departmentService
26. private DepartmentService departmentService;
28. public void setDepartmentService(DepartmentService departmentService) {
29. this.departmentService = departmentService;
30. }
32. //接收当前页面的值
33. private int currentPage = 1;
35. public void setCurrentPage(int currentPage) {
36. this.currentPage = currentPage;
37. }
39. //分页查询部门
40. public String findAll(){
41. PageBean<Department> pageBean = departmentService.findByPage(currentPage);
42. //将pageBean存入到值栈中
43. ActionContext.getContext().getValueStack().push(pageBean);
44. return "findAll";
45. }
47. //通过action返回添加部门的视图
48. public String saveUI(){
49. return "saveUI";
50. }
52. /\*\*
53. \* 保存部门
54. \* @return
55. \*/
56. //添加部门的save方法
57. public String save(){
58. departmentService.save(department);
59. return "saveSuccess";
60. }
62. /\*\*
63. \* 根据ID查询部门
64. \* @return
65. \*/
66. public String editById(){
67. department = departmentService.editById(department.getDid());
68. return "editSuccess";
69. }
71. /\*\*
72. \* 修改部门
73. \* @return
74. \*/
75. public String update(){
76. departmentService.update(department);
77. return "updateSuccess";
78. }
80. /\*\*
81. \* 删除部门
82. \* @return
83. \*/
84. public String delete(){
85. Department dep = departmentService.editById(department.getDid());
86. departmentService.delete(dep);
87. return "deleteSuccess";
88. }
90. }</span>

       EmployeeAction类

1. <span style="font-size:18px;">package com.employee.action;
3. import java.util.List;
5. import com.employee.domain.Department;
6. import com.employee.domain.Employee;
7. import com.employee.domain.PageBean;
8. import com.employee.service.DepartmentService;
9. import com.employee.service.EmployeeService;
10. import com.opensymphony.xwork2.ActionContext;
11. import com.opensymphony.xwork2.ActionSupport;
12. import com.opensymphony.xwork2.ModelDriven;
14. /\*\*
15. \* 员工管理的Action类
16. \* @author Administrator
17. \* @date 2016年12月24日
18. \*/
19. @SuppressWarnings("serial")
20. public class EmployeeAction extends ActionSupport implements ModelDriven<Employee>{
22. //是struts与spring融合后自动注入的
23. private Employee employee = new Employee();
25. @Override
26. public Employee getModel() {
27. return employee;
28. }
30. //使用spring将service注入
31. private EmployeeService employeeService;
33. public void setEmployeeService(EmployeeService employeeService) {
34. this.employeeService = employeeService;
35. }
37. //spring注入部门管理的service
38. private DepartmentService departmentService;
40. public void setDepartmentService(DepartmentService departmentService) {
41. this.departmentService = departmentService;
42. }
44. //获取当前页
45. private int currentPage=1;
47. public void setCurrentPage(int currentPage) {
48. this.currentPage = currentPage;
49. }
51. /\*\*
52. \* 登录执行的方法
53. \* @return
54. \*/
55. public String login(){
56. //调用业务层的类
57. Employee exsitEmploy = employeeService.findUserExist(employee);
58. if(exsitEmploy == null){
59. //登录失败
60. this.addActionError("用户名或者密码错误！");
61. return INPUT;
62. }else{
63. //登录成功
64. //把用户信息保存在Session域中
65. ActionContext.getContext().getSession().put("exsitEmploy",exsitEmploy);
66. return SUCCESS;
67. }
68. }
70. /\*\*
71. \* 分页查询员工
72. \* @return
73. \*/
74. public String findAll(){
75. PageBean<Employee> pageBean = employeeService.findAll(currentPage);
76. ActionContext.getContext().getValueStack().push(pageBean);
77. return "findAll";
78. }
80. /\*\*
81. \* 跳转到添加员工页面
82. \* @return
83. \*/
84. public String saveUI(){
85. //用于查询出所有的部门信息
86. List<Department> list = departmentService.findAll();
87. //保存到值栈中
88. ActionContext.getContext().getValueStack().set("list",list);
89. return "saveUI";
90. }
92. /\*\*
93. \* 保存员工
94. \* @return
95. \*/
96. public String save(){
97. employeeService.save(employee);
98. return "saveSuccess";
99. }
101. /\*\*
102. \* 编辑员工
103. \* @return
104. \*/
105. public String edit(){
106. employee = employeeService.edit(employee.getEid());
107. List<Department> list = departmentService.findAll();
108. ActionContext.getContext().getValueStack().set("list",list);
109. return "editSuccess";
110. }
112. /\*\*
113. \* 修改员工
114. \* @return
115. \*/
116. public String update(){
117. employeeService.update(employee);
118. return "updateSuccess";
119. }
121. /\*\*
122. \* 删除员工
123. \* @return
124. \*/
125. public String delete(){
126. Employee emp = employeeService.edit(employee.getEid());
127. employeeService.delete(emp);
128. return "deleteSuccess";
129. }

132. }</span>

       3、struts.xml

1. <span style="font-size:18px;"><?xml version="1.0" encoding="UTF-8" ?>
2. <!DOCTYPE struts PUBLIC
3. "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
4. "http://struts.apache.org/dtds/struts-2.1.dtd">
6. <struts>
7. <package name="ssh" namespace="/" extends="struts-default">
9. <action name="employee\_\*" class="employeeAction" method="{1}">
10. <result name="input">/index.jsp</result>
11. <result name="success" type="dispatcher">/frame.jsp</result>
12. <result name="findAll">/jsp/employee/listEmployee.jsp</result>
13. <result name="saveUI">/jsp/employee/addEmployee.jsp</result>
14. <result name="saveSuccess" type="redirect">employee\_findAll.action</result>
15. <result name="editSuccess">/jsp/employee/editEmployee.jsp</result>
16. <result name="updateSuccess" type="redirect">employee\_findAll.action</result>
17. <result name="deleteSuccess" type="redirect">employee\_findAll.action</result>
18. </action>
20. <action name="department\_\*" class="departmentAction" method="{1}">
21. <result name="findAll">/jsp/department/listDepartment.jsp</result>
22. <result name="saveUI">/jsp/department/addDepartment.jsp</result>
23. <result name="saveSuccess" type="redirect">department\_findAll.action</result>
24. <result name="editSuccess">/jsp/department/editDepartment.jsp</result>
25. <result name="updateSuccess" type="redirect">department\_findAll.action</result>
26. <result name="deleteSuccess" type="redirect">department\_findAll.action</result>
27. </action>
29. </package>
31. </struts>    </span>

       service实现

       1、接口

        DepartmentService.java

1. <span style="font-size:18px;">package com.employee.service;
3. import java.util.List;
5. import com.employee.domain.Department;
6. import com.employee.domain.PageBean;
8. /\*\*
9. \* 部门管理的service接口
10. \* @author Administrator
11. \* @date 2016年12月24日
12. \*/
13. public interface DepartmentService {
15. PageBean<Department> findByPage(int currentPage);
17. void save(Department department);
19. Department editById(Integer did);
21. void update(Department department);
23. void delete(Department department);
25. List<Department> findAll();
27. }</span>

        EmployeeService.java

1. <span style="font-size:18px;">package com.employee.service;
3. import com.employee.domain.Employee;
4. import com.employee.domain.PageBean;
6. /\*\*
7. \* 员工管理的service接口
8. \* @author Administrator
9. \* @date 2016年12月24日
10. \*/
11. public interface EmployeeService {
13. Employee findUserExist(Employee employee);
15. PageBean<Employee> findAll(int currentPage);
17. void save(Employee employee);
19. Employee edit(Integer eid);
21. void update(Employee employee);
23. void delete(Employee employee);
25. }</span>

        2、实现类

        DepartmentServiceImpl.java

1. <span style="font-size:18px;">package com.employee.service.impl;
3. import java.util.List;
5. import org.springframework.transaction.annotation.Transactional;
7. import com.employee.dao.DepartmentDao;
8. import com.employee.domain.Department;
9. import com.employee.domain.PageBean;
10. import com.employee.service.DepartmentService;
11. import com.opensymphony.xwork2.ActionContext;
13. /\*\*
14. \* 部门的service接口的实现类
15. \* @author Administrator
16. \* @date 2016年12月24日
17. \*/
18. @Transactional
19. public class DepartmentServiceImpl implements DepartmentService {
21. //spring注入部门的departmentDaoImpl
22. private DepartmentDao departmentDao;
24. public void setDepartmentDao(DepartmentDao departmentDao) {
25. this.departmentDao = departmentDao;
26. }
28. /\*\*
29. \* 分页查询部门方法
30. \*/
31. @Override
32. public PageBean<Department> findByPage(int currentPage) {
33. PageBean<Department> pageBean = new PageBean<Department>();
34. //封装当前页数
35. pageBean.setCurrentPage(currentPage);
36. //封装每页显示记录数
37. int pageCount = 3;
38. pageBean.setPageCount(pageCount);
39. //封装总记录数
40. int totalSize = departmentDao.findCount();
41. pageBean.setTotalSize(totalSize);
42. //封装总页数
43. double ts = totalSize;
44. Double num = Math.ceil(ts /  pageCount);
45. pageBean.setPageSize(num.intValue());
46. //封装每页显示的数据
47. int begin = (currentPage - 1) \* pageCount;
48. List<Department> list = departmentDao.findByPage(begin,pageCount);
49. pageBean.setList(list);
50. ActionContext.getContext().put("list", list);
51. return pageBean;
52. }
54. /\*\*
55. \* 保存部门
56. \*/
57. @Override
58. public void save(Department department) {
59. departmentDao.save(department);
60. }
62. /\*\*
63. \* 根据ID查询部门
64. \*/
65. @Override
66. public Department editById(Integer did) {
67. return departmentDao.editById(did);
68. }
70. /\*\*
71. \* 修改部门
72. \*/
73. @Override
74. public void update(Department department) {
75. departmentDao.update(department);
76. }
78. /\*\*
79. \* 删除部门
80. \*/
81. @Override
82. public void delete(Department dep) {
83. departmentDao.delete(dep);
84. }
86. /\*\*
87. \* 查询所有部门
88. \*/
89. @Override
90. public List<Department> findAll() {
91. return departmentDao.findAll();
92. }
94. }</span>

        EmployeeServiceImpl.java

1. <span style="font-size:18px;">package com.employee.service.impl;
3. import java.util.List;
5. import org.springframework.transaction.annotation.Transactional;
7. import com.employee.dao.EmployeeDao;
8. import com.employee.domain.Employee;
9. import com.employee.domain.PageBean;
10. import com.employee.service.EmployeeService;
12. /\*\*
13. \* 员工管理的service接口的实现类
14. \* @author Administrator
15. \* @date 2016年12月24日
16. \*/
17. @Transactional
18. public class EmployeeServiceImpl implements EmployeeService {
20. //使用spring注入的dao
21. private EmployeeDao employeeDao;
23. public void setEmployeeDao(EmployeeDao employeeDao) {
24. this.employeeDao = employeeDao;
25. }
27. /\*\*
28. \* 业务层的登录的执行方法
29. \* @param employee
30. \* @return
31. \*/
32. @Override
33. public Employee findUserExist(Employee employee) {
34. return employeeDao.findUserExist(employee);
35. }
37. /\*\*
38. \* 分页查询员工
39. \*/
40. @Override
41. public PageBean<Employee> findAll(int currentPage) {
42. PageBean<Employee> pageBean = new PageBean<Employee>();
43. // 封装当前页数
44. pageBean.setCurrentPage(currentPage);
45. // 封装每页显示记录数
46. int pageCount = 3;
47. pageBean.setPageCount(pageCount);
48. // 封装总记录数
49. int totalSize = employeeDao.findCount();
50. pageBean.setTotalSize(totalSize);
51. // 封装总页数
52. double ts = totalSize;
53. Double num = Math.ceil(ts / pageCount);
54. pageBean.setPageSize(num.intValue());
55. // 封装每页显示的数据
56. int begin = (currentPage - 1) \* pageCount;
57. // 这个方法用于查找出指定页数中的List<Employee>
58. List<Employee> list = employeeDao.findAll(begin,pageCount);
59. pageBean.setList(list);
60. return pageBean;
61. }
63. /\*\*
64. \* 保存员工
65. \*/
66. @Override
67. public void save(Employee employee) {
68. employeeDao.save(employee);
69. }
71. /\*\*
72. \* 编辑员工
73. \*/
74. @Override
75. public Employee edit(Integer eid) {
76. return employeeDao.edit(eid);
77. }
79. /\*\*
80. \* 修改员工
81. \*/
82. @Override
83. public void update(Employee employee) {
84. employeeDao.update(employee);
85. }
87. /\*\*
88. \* 删除员工
89. \*/
90. @Override
91. public void delete(Employee employee) {
92. employeeDao.delete(employee);
93. }
95. }</span>

       dao层实现

       1、接口DepartmentDao.java

1. package com.employee.dao;

4. import java.util.List;

7. import com.employee.domain.Department;

10. /\*\*
11. \* 员工管理的dao接口
12. \* @author Administrator
13. \* @date 2016年12月24日
14. \*/
15. public interface DepartmentDao {
16. <span style="white-space:pre">  </span>
17. <span style="white-space:pre">  </span>int findCount();
18. <span style="white-space:pre">  </span>
19. <span style="white-space:pre">  </span>List<Department> findByPage(int begin, int pageSize);
20. <span style="white-space:pre">  </span>
21. <span style="white-space:pre">  </span>void save(Department department);

24. Department editById(Integer did);

27. <span style="white-space:pre">  </span>void update(Department department);

30. <span style="white-space:pre">  </span>void delete(Department department);

33. <span style="white-space:pre">  </span>List<Department> findAll();
34. <span style="white-space:pre">  </span>
35. }

       EmployeeDao.java

1. package com.employee.dao;

4. import java.util.List;

7. import com.employee.domain.Employee;

10. /\*\*
11. \* 员工管理的dao接口
12. \* @author Administrator
13. \* @date 2016年12月24日
14. \*/
15. public interface EmployeeDao {
16. <span style="white-space:pre">  </span>
17. <span style="white-space:pre">  </span>Employee findUserExist(Employee employee);
18. <span style="white-space:pre">  </span>
19. <span style="white-space:pre">  </span>int findCount();
20. <span style="white-space:pre">  </span>
21. <span style="white-space:pre">  </span>List<Employee> findAll(int begin, int pageSize);

24. <span style="white-space:pre">  </span>void save(Employee employee);

27. <span style="white-space:pre">  </span>Employee edit(Integer eid);

30. <span style="white-space:pre">  </span>void update(Employee employee);

33. <span style="white-space:pre">  </span>void delete(Employee employee);

36. }

       2、实现类

       DepartmentDaoImpl.java

1. package com.employee.dao.impl;
3. import java.util.List;
5. import org.hibernate.criterion.DetachedCriteria;
6. import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
8. import com.employee.dao.DepartmentDao;
9. import com.employee.domain.Department;
11. /\*\*
12. \* 部门的dao接口的实现类
13. \* @author Administrator
14. \* @date 2016年12月24日
15. \*/
16. public class DepartmentDaoImpl extends HibernateDaoSupport implements DepartmentDao {
18. @SuppressWarnings("unchecked")
19. @Override
20. public int findCount() {
21. String hql = "select count(\*) from Department";
22. List<Long> list = this.getHibernateTemplate().find(hql);
24. if(list.size()>0){
25. return list.get(0).intValue();
26. }else{
27. return 0;
28. }
29. }
31. @SuppressWarnings("unchecked")
32. @Override
33. public List<Department> findByPage(int begin, int pageCount) {
34. //通过分离标准方法得到数据
35. DetachedCriteria criteria = DetachedCriteria.forClass(Department.class);
36. List<Department> list = this.getHibernateTemplate().findByCriteria(criteria,begin,pageCount);
37. return list;
38. }
40. /\*\*
41. \* 保存部门
42. \*/
43. @Override
44. public void save(Department department) {
45. this.getHibernateTemplate().save(department);
46. }
48. /\*\*
49. \* 根据ID查询部门
50. \*/
51. @Override
52. public Department editById(Integer did) {
53. return this.getHibernateTemplate().get(Department.class,did);
54. }
56. /\*\*
57. \* 修改部门
58. \*/
59. @Override
60. public void update(Department department) {
61. this.getHibernateTemplate().update(department);
62. }
64. /\*\*
65. \* 删除部门
66. \*/
67. @Override
68. public void delete(Department department) {
69. this.getHibernateTemplate().delete(department);
70. }
72. /\*\*
73. \* 查询所有部门
74. \*/
75. @SuppressWarnings("unchecked")
76. @Override
77. public List<Department> findAll() {
78. String hql = "from Department";
79. List<Department> list = this.getHibernateTemplate().find(hql);
80. return list;
81. }
83. }

       EmployeeDaoImpl.java

1. package com.employee.dao.impl;
3. import java.util.List;
5. import org.hibernate.criterion.DetachedCriteria;
6. import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
8. import com.employee.dao.EmployeeDao;
9. import com.employee.domain.Employee;
11. /\*\*
12. \* 员工管理的dao接口的实现类
13. \* @author Administrator
14. \* @date 2016年12月24日
15. \*/
16. public class EmployeeDaoImpl extends HibernateDaoSupport implements EmployeeDao {
18. /\*\*
19. \* 查询用户名和密码是否存在
20. \*/
21. @SuppressWarnings("unchecked")
22. @Override
23. public Employee findUserExist(Employee employee) {
24. String hql = " from Employee where username = ? and password = ?";
25. List<Employee> list = this.getHibernateTemplate().find(hql,employee.getUsername(),employee.getPassword());
27. if(list.size() > 0){
28. return list.get(0);
29. }
30. return null;
31. }
33. /\*\*
34. \* 查询总记录数
35. \*/
36. @SuppressWarnings("unchecked")
37. @Override
38. public int findCount() {
39. String hql = "select count(\*) from Employee";
40. List<Long> list = this.getHibernateTemplate().find(hql);
42. if(list.size() > 0){
43. return list.get(0).intValue();
44. }
45. return 0;
46. }
48. /\*\*
49. \* 分页查询员工
50. \*/
51. @SuppressWarnings("unchecked")
52. @Override
53. public List<Employee> findAll(int begin, int pageCount) {
54. DetachedCriteria criteria = DetachedCriteria.forClass(Employee.class);
55. List<Employee> list = this.getHibernateTemplate().findByCriteria(criteria, begin, pageCount);
56. return list;
57. }
59. /\*\*
60. \* 保存员工
61. \*/
62. @Override
63. public void save(Employee employee) {
64. this.getHibernateTemplate().save(employee);
65. }
67. /\*\*
68. \* 编辑员工
69. \*/
70. @Override
71. public Employee edit(Integer eid) {
72. return this.getHibernateTemplate().get(Employee.class, eid);
73. }
75. /\*\*
76. \* 修改员工
77. \*/
78. @Override
79. public void update(Employee employee) {
80. this.getHibernateTemplate().update(employee);
81. }
83. /\*\*
84. \* 删除员工
85. \*/
86. @Override
87. public void delete(Employee employee) {
88. this.getHibernateTemplate().delete(employee);
89. }
91. }

       配置文件

       ApplicationContext.xml

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:aop="http://www.springframework.org/schema/aop"
6. xmlns:tx="http://www.springframework.org/schema/tx"
7. xsi:schemaLocation="http://www.springframework.org/schema/beans
8. http://www.springframework.org/schema/beans/spring-beans.xsd
9. http://www.springframework.org/schema/context
10. http://www.springframework.org/schema/context/spring-context.xsd
11. http://www.springframework.org/schema/aop
12. http://www.springframework.org/schema/aop/spring-aop.xsd
13. http://www.springframework.org/schema/tx
14. http://www.springframework.org/schema/tx/spring-tx.xsd">
16. <!-- 引入外部的属性文件 -->
17. <context:property-placeholder location="classpath:jdbc.properties"/>
19. <!-- 配置连接池 -->
20. <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
21. <property name="driverClass" value="${jdbc.driverClass}"></property>
22. <property name="jdbcUrl" value="${jdbc.url}"></property>
23. <property name="user" value="${jdbc.username}"></property>
24. <property name="password" value="${jdbc.password}"></property>
25. </bean>
27. <!-- 配置Hibernate的相关属性 -->
28. <bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
29. <!-- 注入连接池 -->
30. <property name="dataSource" ref="dataSource"></property>
31. <!-- 加载Hibernate相关属性 -->
32. <property name="hibernateProperties">
33. <props>
34. <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
35. <prop key="hibernate.show\_sql">true</prop>
36. <prop key="hibernate.format\_sql">true</prop>
37. <prop key="hibernate.hbm2ddl.auto">update</prop>
38. </props>
39. </property>
40. <!-- 加载hibernate中的映射文件 -->
41. <property name="mappingResources">
42. <list>
43. <value>com/employee/domain/Department.hbm.xml</value>
44. <value>com/employee/domain/Employee.hbm.xml</value>
45. </list>
46. </property>
47. </bean>
49. <!-- 配置Action的类 -->
50. <bean id="employeeAction" class="com.employee.action.EmployeeAction" scope="prototype">
51. <property name="employeeService" ref="employeeService"></property>
52. <property name="departmentService" ref="departmentService"></property>
53. </bean>
55. <bean id="departmentAction" class="com.employee.action.DepartmentAction" scope="prototype">
56. <property name="departmentService" ref="departmentService"></property>
57. </bean>
59. <!-- 配置业务层的类 -->
60. <bean id="employeeService" class="com.employee.service.impl.EmployeeServiceImpl">
61. <property name="employeeDao" ref="employeeDao"></property>
62. </bean>
64. <bean id="departmentService" class="com.employee.service.impl.DepartmentServiceImpl">
65. <property name="departmentDao" ref="departmentDao"></property>
66. </bean>
68. <!-- 配置DAO的类 -->
69. <bean id="employeeDao" class="com.employee.dao.impl.EmployeeDaoImpl">
70. <property name="sessionFactory" ref="sessionFactory"></property>
71. </bean>
73. <bean id="departmentDao" class="com.employee.dao.impl.DepartmentDaoImpl">
74. <property name="sessionFactory" ref="sessionFactory"></property>
75. </bean>
77. <!-- 配置事务管理 -->
78. <bean id="transaction" class="org.springframework.orm.hibernate3.HibernateTransactionManager">
79. <property name="sessionFactory" ref="sessionFactory"></property>
80. </bean>
82. <!-- 开启注解事务 -->
83. <tx:annotation-driven transaction-manager="transaction"></tx:annotation-driven>
85. </beans>

       jdbc.properties

1. jdbc.driverClass=com.mysql.jdbc.Driver
2. jdbc.url=jdbc:mysql://localhost:3306/ssh
3. jdbc.username=root
4. jdbc.password=root

       log4j.properties

1. log4j.rootLogger = debug, console, E
2. #level=INFO,all can be output
3. #console is set to be a ConsoleAppender
4. log4j.appender.console = org.apache.log4j.ConsoleAppender
5. log4j.appender.console.layout = org.apache.log4j.PatternLayout
6. log4j.appender.console.layout.ConversionPattern = [%p] %d{yyyy-MM-dd HH:mm:ss} - %m%n

9. #file is set to output to a extra file
10. log4j.appender.E = org.apache.log4j.DailyRollingFileAppender
11. log4j.appender.E.layout = org.apache.log4j.PatternLayout
12. log4j.appender.E.Threshold=error
13. log4j.appender.E.Encoding=UTF-8
14. log4j.appender.E.layout.ConversionPattern=[%p] %d{yyyy-MM-dd HH:mm:ss} - %m%n
15. log4j.appender.E.DatePattern='.'yyyy-MM-dd
16. log4j.appender.E.File = ${catalina.base}/logs/dd/error.log

 运行项目：



