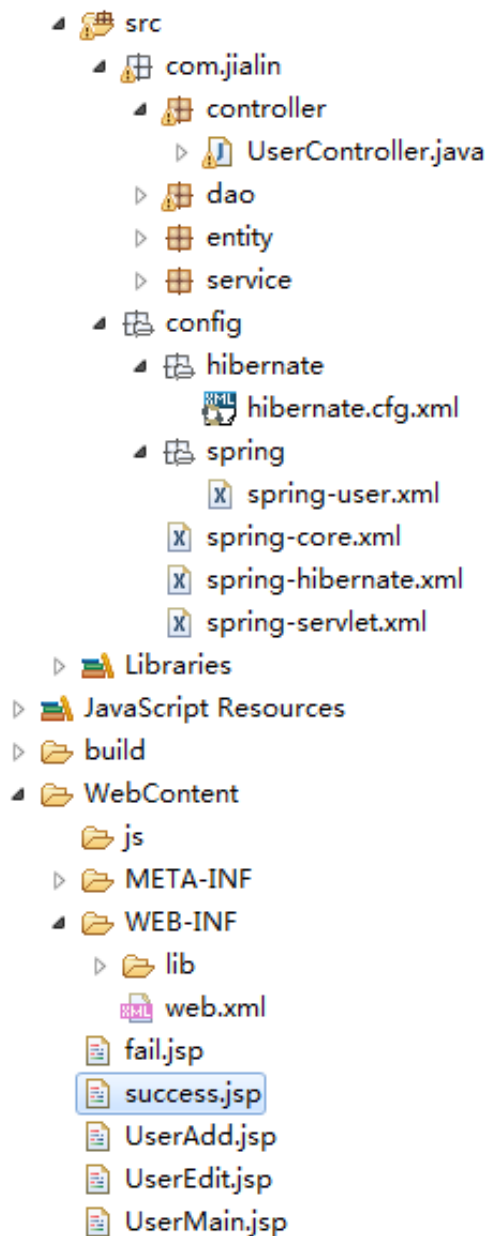


早期的项目比较简单，多是用JSP、Servlet + JDBC 直接搞定，后来使用

Struts1(Struts2)+Spring+Hibernate, 严格按照分层概念驱动项目开发，这次又使用 Spring MVC取代Struts来进行开发。

MVC已经是现代Web开发中的一个很重要的部分，下面介绍一下SpringMVC+Spring3+Hibernate4的开发环境搭建

先大致看一下项目结构：



具体的代码不再演示，主要是走了一个很平常的路线，mvc-servcie-dao-hibernate的结构，[源码可以下载](#)，主要看一下配置文件。解释见注释

web.xml

```
[html] view plain copy print ?
01. <?xml version="1.0" encoding="UTF-8"?>
02. <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/
app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml.
```

```
app_2_5.xsd" id="WebApp_ID" version="2.5">
03.     <display-name>SpringMVC</display-name>
04.     <welcome-file-list>
05.         <welcome-file>index.jsp</welcome-file>
06.     </welcome-file-list>
07.
08.     <!-- 配置Spring -->
09.     <context-param>
10.         <param-name>contextConfigLocation</param-name>
11.         <param-value>classpath*:config/spring-*.xml</param-value>
12.     </context-param>
13.
14.
15.     <listener>
16.         <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
17.     </listener>
18.
19.
20.     <!-- 配置SpringMVC -->
21.     <servlet>
22.         <servlet-name>springMVC</servlet-name>
23.         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
24.         <init-param>
25.             <param-name>contextConfigLocation</param-name>
26.             <param-value>classpath*:config/spring-servlet.xml</param-value>
27.         </init-param>
28.         <load-on-startup>1</load-on-startup>
29.     </servlet>
30.
31.     <servlet-mapping>
32.         <servlet-name>springMVC</servlet-name>
33.         <url-pattern>/</url-pattern>
34.     </servlet-mapping>
35.
36.     <!-- 设置字符集 -->
37.     <filter>
38.         <filter-name>encodingFilter</filter-name>
39.         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
40.         <init-param>
41.             <param-name>encoding</param-name>
42.             <param-value>UTF-8</param-value>
43.         </init-param>
44.         <init-param>
45.             <param-name>forceEncoding</param-name>
46.             <param-value>true</param-value>
47.         </init-param>
48.     </filter>
49.     <filter-mapping>
50.         <filter-name>encodingFilter</filter-name>
51.         <url-pattern>/*</url-pattern>
52.     </filter-mapping>
53.
54.
```

下载

```
55. <!-- 控制Session的开关 -->
56. <filter>
57.     <filter-name>openSession</filter-name>
58.     <filter-
59. class>org.springframework.orm.hibernate4.support.OpenSessionInViewFilter</filter-class>
60. </filter>
61.
62. <filter-mapping>
63.     <filter-name>openSession</filter-name>
64.     <url-pattern>/*</url-pattern>
65. </filter-mapping>
66. </web-app>
```

config

- hibernate
 - hibernate.cfg.xml
- spring
 - spring-user.xml
 - spring-core.xml
 - spring-hibernate.xml
 - spring-servlet.xml

spring-servlet.xml

[html] view plain copy print ?

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <beans xmlns="http://www.springframework.org/schema/beans"
03.     xmlns:context="http://www.springframework.org/schema/context" xmlns:p="http://www.spr
04.     xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/200
instance"
05.     xsi:schemaLocation="http://www.springframework.org/schema/beans
06.         http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
07.         http://www.springframework.org/schema/context
08.         http://www.springframework.org/schema/context/spring-context.xsd
09.         http://www.springframework.org/schema/mvc
10.         http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">
11.
12.     <!-- 注解扫描的包 -->
13.     <context:component-scan base-package="com.jialin" />
14.
15.     <!-- 开启注解方案1 -->
16.     <!-- 注解方法处理 -->
17.     <!--
18. - <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapt
19.         /> -->
20.     <!-- 注解类映射处理 -->
21.     <!--
22. - <bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapp
23. </bean> -->
24.
25.     <!-- 开启注解方案2 -->
```

```
23.      <mvc:annotation-driven />
24.
25.      <!-- 静态资源访问, 方案1 -->
26.      <mvc:resources location="/img/" mapping="/img/**" />
27.      <mvc:resources location="/js/" mapping="/js/**" />
28.
29.      <!-- 静态资源访问, 方案2 -->
30.      <!-- <mvc:default-servlet-handler/> -->
31.
32.      <!-- 视图解释类 -->
33.      <bean id="viewResolver"
34.          class="org.springframework.web.servlet.view.InternalResourceViewResolver">
35.          <property name="prefix" value="/"></property>
36.          <!-- 可为空,方便实现自己的依据扩展名来选择视图解释类的逻辑 -->
37.          <property name="suffix" value=".jsp"></property>
38.      </bean>
39.
40.      <!-- 上传文件bean -->
41.      <!--
42.      - <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMul
43.          <property name="defaultEncoding" value="utf-
44.          8" /> <property name="maxUploadSize"
45.          value="1048576000" /> <property name="maxInMemorySize" value="40960" />
46.          </bean> -->
47.      </beans>
```

spring-hibernate.xml

```
[html] view plain copy print ?
01.      <?xml version="1.0" encoding="UTF-8"?>
02.      <beans xmlns="http://www.springframework.org/schema/beans"
03.          xmlns:context="http://www.springframework.org/schema/context" xmlns:p="http://www.spr
04.          xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/200
instance"
05.          xsi:schemaLocation="http://www.springframework.org/schema/beans
06.              http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
07.              http://www.springframework.org/schema/context
08.              http://www.springframework.org/schema/context/spring-context.xsd
09.              http://www.springframework.org/schema/mvc
10.              http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">
11.
12.      <!-- 配置数据源 -->
13.      <bean id="dataSource"
14.          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
15.          <property name="driverClassName" value="com.mysql.jdbc.Driver" />
16.          <property name="url" value="jdbc:mysql://127.0.0.1/springmvc" />
17.          <property name="username" value="root" />
18.          <property name="password" value="123456" />
19.      </bean>
20.
21.      <!-- 配置hibernate SessionFactory-->
```

```
22. <bean id="sessionFactory"
23.     class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
24.     <property name="dataSource" ref="dataSource" />
25.     <property name="hibernateProperties">
26.         <props>
27.             <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
28.             <prop key="hibernate.hbm2ddl.auto">update</prop>
29.             <prop key="hibernate.show_sql">true</prop>
30.             <prop key="hibernate.format_sql">true</prop>
31.         </props>
32.     </property>
33.     <property name="configLocations">
34.         <list>
35.             <value>
36.                 classpath*:config/hibernate/hibernate.cfg.xml
37.             </value>
38.         </list>
39.     </property>
40. </bean>
41.
42. <!-- 事务管理器 -->
43. <bean id="transactionManager"
44.     class="org.springframework.orm.hibernate4.HibernateTransactionManager">
45.     <property name="sessionFactory" ref="sessionFactory"></property>
46. </bean>
47.
48. <!-- 事务代理类 -->
49. <bean id="transactionBese"
50.     class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean"
51.     lazy-init="true" abstract="true">
52.     <property name="transactionManager" ref="transactionManager"></property>
53.     <property name="transactionAttributes">
54.         <props>
55.             <prop key="add*">PROPAGATION_REQUIRED,-Exception</prop>
56.             <prop key="update*">PROPAGATION_REQUIRED,-Exception</prop>
57.             <prop key="insert*">PROPAGATION_REQUIRED,-Exception</prop>
58.             <prop key="modify*">PROPAGATION_REQUIRED,-Exception</prop>
59.             <prop key="delete*">PROPAGATION_REQUIRED,-Exception</prop>
60.             <prop key="del*">PROPAGATION_REQUIRED,-Exception</prop>
61.             <prop key="get*">PROPAGATION_NEVER</prop>
62.         </props>
63.     </property>
64. </bean>
65.
66. </beans>
```

spring-core.xml

[html] view plain copy print ?

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

```

02. <beans xmlns="http://www.springframework.org/schema/beans"
03.      xmlns:context="http://www.springframework.org/schema/context"
04.      xmlns:p="http://www.springframework.org/schema/p"
05.      xmlns:mvc="http://www.springframework.org/schema/mvc"
06.      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
07.      xsi:schemaLocation="http://www.springframework.org/schema/beans
08.          http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
09.          http://www.springframework.org/schema/context
10.          http://www.springframework.org/schema/context/spring-context.xsd
11.          http://www.springframework.org/schema/mvc
12.          http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">
13.
14.     <!-- 引入其他配置文件，可以为多个 -->
15.     <import resource="classpath*:config/spring/spring-user.xml"/>
16.
17. </beans>

```

spring-user.xml

```

[html] view plain copy print ?
01. <?xml version="1.0" encoding="UTF-8"?>
02. <!DOCTYPE beans PUBLIC "-//
//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.dtd" [
03. <!ENTITY contextInclude SYSTEM "org/springframework/web/context/WEB-
INF/contextInclude.xml">
04. ]>
05.
06. <beans>
07.     <!-- Spring Bean -->
08.     <bean id="userDao" class="com.jialin.dao.UserDao">
09.         <property name="sessionFactory" ref="sessionFactory"></property>
10.     </bean>
11.
12.     <bean id="userManagerBase" class="com.jialin.service.UserManager">
13.         <property name="userDao" ref="userDao"></property>
14.     </bean>
15.
16.     <!-- parent为transactionBese, 表示支持事务 -->
17.     <bean id="userManager" parent="transactionBese">
18.         <property name="target" ref="userManagerBase"></property>
19.     </bean>
20.
21. </beans>

```

hibernate.cfg.xml

```

[html] view plain copy print ?
01. <!DOCTYPE hibernate-configuration PUBLIC
02.     "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

```

```
03.         "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
04.
05.     <hibernate-configuration>
06.         <session-factory>                                下载
07.             <!-- 引入需要映射的类 -->
08.             <mapping class="com.jialin.entity.User"/>
09.         </session-factory>
10.     </hibernate-configuration>
```

下面再来看看Controller

```
[java] view plain copy print ?
01. package com.jialin.controller;
02.
03. import java.io.IOException;
04. import java.io.PrintWriter;
05. import java.util.List;
06.
07. import javax.annotation.Resource;
08. import javax.servlet.http.HttpServletRequest;
09. import javax.servlet.http.HttpServletResponse;
10.
11. import org.springframework.stereotype.Controller;
12. import org.springframework.web.bind.annotation.RequestMapping;
13.
14. import com.jialin.entity.User;
15. import com.jialin.service.IUserManager;
16.
17. @Controller    //类似Struts的Action
18. @RequestMapping("/user")
19. public class UserController {
20.
21.     @Resource(name="userManager")    // 获取spring配置文件中bean的id为userManager, 并注入
22.     private IUserManager userManager;
23.
24.     @RequestMapping("/addUser")    // 请求url地址映射, 类似Struts的action-mapping
25.     public String addUser(User user){
26.         if(userManager.addUser(user))
27.         {
28.             // 重定向
29.             return "redirect:/user/getAllUser";
30.         }else
31.         {
32.             return "/fail";
33.         }
34.
35.     }
36.
37.     @RequestMapping("/updateUser")
38.     public String updateUser(User user,HttpServletRequest request){
39.         if (userManager.updateUser(user))
```

```
40.     {
41.         user = userManager.getOneUser(user);
42.         request.setAttribute("user", user);
43.         return "/UserEdit";
44.     }else
45.     {
46.         return "/fail";
47.     }
48.
49. }
50.
51. @RequestMapping("/delUser")
52. public void delUser(User user,HttpServletResponse response){
53.     String result = "{\"result\":\"error\"}";
54.
55.     if(userManager.delUser(user)){
56.         result = "{\"result\":\"success\"}";
57.     }
58.     PrintWriter out = null;
59.     response.setContentType("application/json");
60.
61.     try {
62.         out = response.getWriter();
63.         out.write(result);
64.     } catch (IOException e) {
65.         e.printStackTrace();
66.     }
67.
68. }
69. @RequestMapping("/toAddUser")
70. public String toAddUser(){
71.     return "/UserAdd";
72. }
73.
74. @RequestMapping("/toUpdateUser")
75. public String toUpdateUser(User user,HttpServletRequest request){
76.     User user1=userManager.getOneUser(user);
77.
78.     request.setAttribute("user1", user1);
79.
80.     return "/UserEdit";
81. }
82.
83. @RequestMapping("/getAllUser")
84. public String getAllUser(HttpServletRequest request){
85.
86.     List userList=userManager.getAllUser();
87.
88.     request.setAttribute("userlist", userList);
89.
90.     return "/UserMain";
91. }
92.
93. }
```