[Maven搭建SpringMVC+Hibernate项目详解](http://blog.csdn.net/fengshizty/article/details/43635305)

分类： [maven项目搭建](http://blog.csdn.net/fengshizty/article/category/2854337)2015-02-08 11:39 877人阅读 [评论](http://blog.csdn.net/fengshizty/article/details/43635305#comments)(9) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/fengshizty/article/details/43635305#report)

[SpringMVC](http://www.csdn.net/tag/SpringMVC) [Hibernate](http://www.csdn.net/tag/Hibernate) [Ehcache](http://www.csdn.net/tag/Ehcache) [Maven](http://www.csdn.net/tag/Maven) [SpringMVC+Hibernate](http://www.csdn.net/tag/SpringMVC%2bHibernate)

**前言**

     今天复习一下SpringMVC+Hibernate的搭建，本来想着将Spring-Security权限控制框架也映入其中的，但是发现内容太多了，Spring-Security的就留在下一篇吧，这篇主要搭建SpringMVC4.1.4和Hibernate4.3.8，之前也打了好多SpringMVC的，这部分已经非常的熟悉了，毕竟业开发过一年多SpringMVC的，这次持久层采用Hibernate，数据源采用c3p0，数据库暂采用MySQL，主要是想复习一下Hibernate。搭建SpringMVC+Hibernate的框架的思路如下：

     1、设计数据库：设计好表结构，最好符合3NF，采用Hibernate tools将设计好的表自动生成对应的实体entity。

     1、创建Maven项目，按需映入Maven包依赖。

     2、搭建Spring：配置Spring对控件层Bean的注入。

     3、搭建Hibernate：配置数据源，配置SessionFactory，配置事务，以及二级缓存ehcache的配置。

     4、测试Spring+Hibernate的框架搭建，写单元测试JUnit，测试事务配置等。

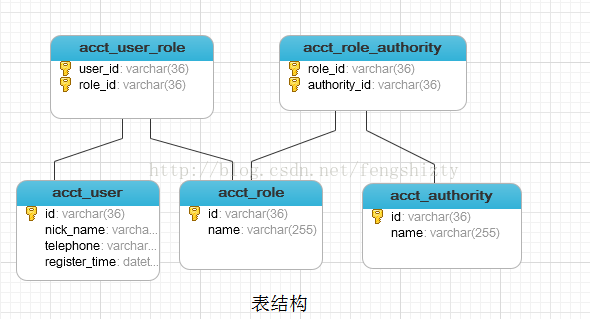
     5、映入SpringMVC：配置SpringMVC配置信息。

     6、配置web.xml容器

     7、测试三个框架的整合：Maven编译打包部署服务器，测试。

**1、数据库设计**

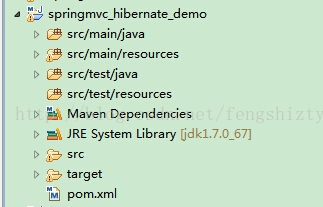
       设计一个权限、角色、用户的表。自然如果一个用户可以有多个角色，一个角色多个用户所拥有；一个角色拥有多个权限，一个权限对应多个角色。所以按照数据库表结构设计，符合3NF的话我们需要5张表进行存在。表的具体信息就不列出了（可以下在源码，里面还有数据库），表关系如下：



      这里面ID才用到是java的UUID（36位）

**2、创建Maven项目，编写pom.xml文件**

      创建一个Maven的webapp项目，编写pom.xml文件，引入所需的包依赖，我这里就将所需的全部的映入了。



      pom.xml文件内容如下:

**[html]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **<project** xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"**>**
3. **<modelVersion>**4.0.0**</modelVersion>**
4. **<groupId>**org.andy.sh**</groupId>**
5. **<artifactId>**springmvc\_hibernate\_demo**</artifactId>**
6. **<packaging>**war**</packaging>**
7. **<version>**0.0.1-SNAPSHOT**</version>**
9. **<name>**springmvc\_hibernate\_demo Maven Webapp**</name>**
10. **<url>**http://maven.apache.org**</url>**
12. **<properties>**
13. **<project.build.sourceEncoding>**UTF-8**</project.build.sourceEncoding>**
14. **<spring.version>**4.1.4.RELEASE**</spring.version>**
15. **<hibernate.version>**4.3.8.Final**</hibernate.version>**
16. **<jackson.version>**2.5.0**</jackson.version>**
17. **</properties>**
19. **<dependencies>**
21. <!-- junit -->
22. **<dependency>**
23. **<groupId>**junit**</groupId>**
24. **<artifactId>**junit**</artifactId>**
25. **<version>**4.12**</version>**
26. **<scope>**test**</scope>**
27. **</dependency>**
29. <!-- spring -->
30. **<dependency>**
31. **<groupId>**org.springframework**</groupId>**
32. **<artifactId>**spring-core**</artifactId>**
33. **<version>**${spring.version}**</version>**
34. **</dependency>**
36. **<dependency>**
37. **<groupId>**org.springframework**</groupId>**
38. **<artifactId>**spring-beans**</artifactId>**
39. **<version>**${spring.version}**</version>**
40. **</dependency>**
42. **<dependency>**
43. **<groupId>**org.springframework**</groupId>**
44. **<artifactId>**spring-context**</artifactId>**
45. **<version>**${spring.version}**</version>**
46. **</dependency>**
48. **<dependency>**
49. **<groupId>**org.springframework**</groupId>**
50. **<artifactId>**spring-tx**</artifactId>**
51. **<version>**${spring.version}**</version>**
52. **</dependency>**
54. **<dependency>**
55. **<groupId>**org.springframework**</groupId>**
56. **<artifactId>**spring-web**</artifactId>**
57. **<version>**${spring.version}**</version>**
58. **</dependency>**
60. **<dependency>**
61. **<groupId>**org.springframework**</groupId>**
62. **<artifactId>**spring-test**</artifactId>**
63. **<version>**${spring.version}**</version>**
64. **<scope>**test**</scope>**
65. **</dependency>**
67. <!-- 使用SpringMVC需配置 -->
68. **<dependency>**
69. **<groupId>**org.springframework**</groupId>**
70. **<artifactId>**spring-webmvc**</artifactId>**
71. **<version>**${spring.version}**</version>**
72. **</dependency>**
74. <!-- 关系型数据库整合时需配置 如hibernate jpa等 -->
75. **<dependency>**
76. **<groupId>**org.springframework**</groupId>**
77. **<artifactId>**spring-orm**</artifactId>**
78. **<version>**${spring.version}**</version>**
79. **</dependency>**
81. <!-- hibernate -->
82. **<dependency>**
83. **<groupId>**org.hibernate**</groupId>**
84. **<artifactId>**hibernate-core**</artifactId>**
85. **<version>**${hibernate.version}**</version>**
86. **</dependency>**
88. **<dependency>**
89. **<groupId>**org.hibernate**</groupId>**
90. **<artifactId>**hibernate-ehcache**</artifactId>**
91. **<version>**${hibernate.version}**</version>**
92. **</dependency>**
94. <!-- 二级缓存ehcache -->
95. **<dependency>**
96. **<groupId>**net.sf.ehcache**</groupId>**
97. **<artifactId>**ehcache**</artifactId>**
98. **<version>**2.9.0**</version>**
99. **</dependency>**
101. <!-- log4j -->
102. **<dependency>**
103. **<groupId>**log4j**</groupId>**
104. **<artifactId>**log4j**</artifactId>**
105. **<version>**1.2.17**</version>**
106. **</dependency>**
108. <!-- mysql连接 -->
109. **<dependency>**
110. **<groupId>**mysql**</groupId>**
111. **<artifactId>**mysql-connector-java**</artifactId>**
112. **<version>**5.1.34**</version>**
113. **</dependency>**
115. <!-- c3p0数据源 -->
116. **<dependency>**
117. **<groupId>**com.mchange**</groupId>**
118. **<artifactId>**c3p0**</artifactId>**
119. **<version>**0.9.5-pre10**</version>**
120. **</dependency>**
122. <!-- json -->
123. **<dependency>**
124. **<groupId>**com.alibaba**</groupId>**
125. **<artifactId>**fastjson**</artifactId>**
126. **<version>**1.2.3**</version>**
127. **</dependency>**
129. **<dependency>**
130. **<groupId>**com.fasterxml.jackson.core**</groupId>**
131. **<artifactId>**jackson-annotations**</artifactId>**
132. **<version>**${jackson.version}**</version>**
133. **</dependency>**
135. **<dependency>**
136. **<groupId>**com.fasterxml.jackson.core**</groupId>**
137. **<artifactId>**jackson-core**</artifactId>**
138. **<version>**${jackson.version}**</version>**
139. **</dependency>**
141. **<dependency>**
142. **<groupId>**com.fasterxml.jackson.core**</groupId>**
143. **<artifactId>**jackson-databind**</artifactId>**
144. **<version>**${jackson.version}**</version>**
145. **</dependency>**
147. <!-- aop -->
148. **<dependency>**
149. **<groupId>**org.aspectj**</groupId>**
150. **<artifactId>**aspectjweaver**</artifactId>**
151. **<version>**1.8.4**</version>**
152. **</dependency>**
154. <!-- servlet -->
155. **<dependency>**
156. **<groupId>**javax.servlet**</groupId>**
157. **<artifactId>**servlet-api**</artifactId>**
158. **<version>**3.0-alpha-1**</version>**
159. **<scope>**provided**</scope>**
160. **</dependency>**
162. **<dependency>**
163. **<groupId>**javax.servlet**</groupId>**
164. **<artifactId>**jstl**</artifactId>**
165. **<version>**1.2**</version>**
166. **</dependency>**
168. **</dependencies>**
170. **<build>**
171. **<finalName>**springmvc\_hibernate\_demo**</finalName>**
173. **<plugins>**
174. <!-- Run the JUnit unit tests in an isolated classloader -->
175. **<plugin>**
176. **<groupId>**org.apache.maven.plugins**</groupId>**
177. **<artifactId>**maven-surefire-plugin**</artifactId>**
178. **<version>**2.4.2**</version>**
179. **<configuration>**
180. **<skipTests>**true**</skipTests>**
181. **</configuration>**
183. **</plugin>**
185. **<plugin>**
186. **<groupId>**org.apache.maven.plugins**</groupId>**
187. **<artifactId>**maven-war-plugin**</artifactId>**
188. **<version>**2.3**</version>**
189. **<configuration>**
190. **<webXml>**src/main/webapp/WEB-INF/web.xml**</webXml>**
191. **</configuration>**
192. **</plugin>**
194. <!-- generate java doc -->
195. **<plugin>**
196. **<groupId>**org.apache.maven.plugins**</groupId>**
197. **<artifactId>**maven-javadoc-plugin**</artifactId>**
198. **<version>**2.9.1**</version>**
199. **<configuration>**
200. **<javadocDirectory>**target/javadoc**</javadocDirectory>**
201. **<reportOutputDirectory>**target/javadoc**</reportOutputDirectory>**
202. **<charset>**UTF-8**</charset>**
203. **<encoding>**UTF-8**</encoding>**
204. **<docencoding>**UTF-8**</docencoding>**
205. **<show>**private**</show>**
206. **</configuration>**
207. **</plugin>**
209. <!-- 部署至本机 -->
210. **<plugin>**
211. **<groupId>**org.codehaus.cargo**</groupId>**
212. **<artifactId>**cargo-maven2-plugin**</artifactId>**
213. **<version>**1.0**</version>**
214. **<configuration>**
215. **<container>**
216. **<containerId>**tomcat6x**</containerId>**
217. **<home>**D:\WebServer\apache-tomcat-6.0.39**</home>**
218. **</container>**
219. **<configuration>**
220. **<type>**existing**</type>**
221. **<home>**D:\WebServer\apache-tomcat-6.0.39**</home>**
222. **</configuration>**
223. **</configuration>**
224. **</plugin>**
225. **</plugins>**
227. **</build>**
228. **</project>**

**3、引入Spring配置**

    编写配置文件，配置文件放在src/main/resources资源目录下（下同）。

   项目所需的配置信息config.properties

**[plain]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. #application configs
3. #jdbc c3p0 config
4. jdbc.driver = com.mysql.jdbc.Driver
5. jdbc.url = jdbc:mysql://localhost:3306/work?useUnicode=true&characterEncoding=utf-8
6. jdbc.username = root
7. jdbc.password = 12345
9. #hibernate config
10. hibernate.dialect = org.hibernate.dialect.MySQLDialect
11. hibernate.show\_sql = true
12. hibernate.format\_sql = false
13. hibernate.hbm2ddl.auto = update
14. hibernate.cache.use\_second\_level\_cache = true
15. hibernate.cache.use\_query\_cache = true
16. hibernate.cache.region.factory\_class = org.hibernate.cache.ehcache.EhCacheRegionFactory
17. hibernate.cache.provider\_configuration\_file\_resource\_path = ehcache.xml

**3.1、spring配置**

spring.xml的配置文件如下：

**[html]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

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" xmlns:context="http://www.springframework.org/schema/context"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans
5. http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
6. http://www.springframework.org/schema/context
7. http://www.springframework.org/schema/context/spring-context-4.1.xsd"**>**
8. <!-- 加载配置文件 -->
10. **<context:property-placeholder** location="classpath:config.properties"**/>**
11. <!-- 扫描service自动注入为bean -->
12. **<context:component-scan** base-package="org.andy.work.service.impl,org.andy.work.dao.impl" **/>**
14. **</beans>**

上面spring配置了对应包下组件bean自动注入管理，我们需要将dao,service的实现类交由spring管理，如（@Repository，@Service）。

**3.2、引入日志文件配置Log4j**

  log4j.properties的配置文件如下：

**[plain]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. ### set log levels ###
2. log4j.rootLogger = INFO , C , D , E
4. ### console ###
5. log4j.appender.C = org.apache.log4j.ConsoleAppender
6. log4j.appender.C.Target = System.out
7. log4j.appender.C.layout = org.apache.log4j.PatternLayout
8. log4j.appender.C.layout.ConversionPattern = [springmvc\_hibernate\_demo][%p] [%-d{yyyy-MM-dd HH:mm:ss}] %C.%M(%L) | %m%n
10. ### log file ###
11. log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
12. log4j.appender.D.File = ../logs/springmvc\_hibernate\_demo.log
13. log4j.appender.D.Append = true
14. log4j.appender.D.Threshold = INFO
15. log4j.appender.D.layout = org.apache.log4j.PatternLayout
16. log4j.appender.D.layout.ConversionPattern = [springmvc\_hibernate\_demo][%p] [%-d{yyyy-MM-dd HH:mm:ss}] %C.%M(%L) | %m%n
18. ### exception ###
19. log4j.appender.E = org.apache.log4j.DailyRollingFileAppender
20. log4j.appender.E.File = ../logs/springmvc\_hibernate\_demo\_error.log
21. log4j.appender.E.Append = true
22. log4j.appender.E.Threshold = ERROR
23. log4j.appender.E.layout = org.apache.log4j.PatternLayout
24. log4j.appender.E.layout.ConversionPattern = [sspringmvc\_hibernate\_demo][%p] [%-d{yyyy-MM-dd HH:mm:ss}] %C.%M(%L) | %m%n

**4、Hibernate配置**

      Hibernate的配置主要包括：配置数据源c3p0，配置SessionFactory，配置事务管理器，配置事务管理，其spring-hibernate.xml文件如下：

**[html]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

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" xmlns:tx="http://www.springframework.org/schema/tx"
4. xmlns:aop="http://www.springframework.org/schema/aop"
5. xsi:schemaLocation="
6. http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
8. http://www.springframework.org/schema/tx
9. http://www.springframework.org/schema/tx/spring-tx-4.1.xsd
10. http://www.springframework.org/schema/aop
11. http://www.springframework.org/schema/aop/spring-aop-4.1.xsd"**>**
13. <!-- 配置数据源 c3p0 -->
14. **<bean** id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
15. destroy-method="close"**>**
16. **<property** name="driverClass" value="${jdbc.driver}" **/>**
17. **<property** name="jdbcUrl" value="${jdbc.url}" **/>**
18. **<property** name="user" value="${jdbc.username}" **/>**
19. **<property** name="password" value="${jdbc.password}" **/>**
21. <!-- 请求超时时间 -->
22. **<property** name="checkoutTimeout" value="30000" **/>**
23. <!-- 每60秒检查所有连接池中的空闲连接。默认值: 0，不检查 -->
24. **<property** name="idleConnectionTestPeriod" value="30" **/>**
25. <!-- 连接数据库连接池最大空闲时间 -->
26. **<property** name="maxIdleTime" value="30" **/>**
27. <!-- 连接池初始化连接数 -->
28. **<property** name="initialPoolSize" value="5" **/>**
29. **<property** name="minPoolSize" value="5" **/>**
30. **<property** name="maxPoolSize" value="20" **/>**
31. <!--当连接池中的连接耗尽的时候c3p0一次同时获取的连接数。默认值: 3 -->
32. **<property** name="acquireIncrement" value="5" **/>**
33. **</bean>**
35. <!-- 配置hibernate的SessionFactory -->
36. **<bean** id="sessionFactory"
37. class="org.springframework.orm.hibernate4.LocalSessionFactoryBean"**>**
38. <!-- 注入数据源 相关信息看源码 -->
39. **<property** name="dataSource" ref="dataSource" **/>**
40. <!-- hibernate配置信息 -->
41. **<property** name="hibernateProperties"**>**
42. **<props>**
43. **<prop** key="hibernate.dialect"**>**${hibernate.dialect}**</prop>**
44. **<prop** key="hibernate.show\_sql"**>**${hibernate.show\_sql}**</prop>**
45. **<prop** key="hibernate.format\_sql"**>**${hibernate.format\_sql}**</prop>**
46. **<prop** key="hibernate.hbm2ddl.auto"**>**${hibernate.hbm2ddl.auto}**</prop>**
48. <!-- 开启二级缓存 ehcache -->
49. **<prop** key="hibernate.cache.use\_second\_level\_cache"**>**${hibernate.cache.use\_second\_level\_cache}**</prop>**
50. **<prop** key="hibernate.cache.use\_query\_cache"**>**${hibernate.cache.use\_query\_cache}**</prop>**
51. **<prop** key="hibernate.cache.region.factory\_class"**>**${hibernate.cache.region.factory\_class}**</prop>**
52. **<prop** key="hibernate.cache.provider\_configuration\_file\_resource\_path"**>**${hibernate.cache.provider\_configuration\_file\_resource\_path}
53. **</prop>**
54. **</props>**
55. **</property>**
56. <!-- 扫描hibernate注解配置的entity -->
57. **<property** name="packagesToScan" value="org.andy.work.entity" **/>**
58. **</bean>**
60. <!-- 配置事务管理器 -->
61. **<bean** id="transactionManager"
62. class="org.springframework.orm.hibernate4.HibernateTransactionManager"**>**
63. **<property** name="sessionFactory" ref="sessionFactory" **/>**
64. **</bean>**
66. <!-- 配置事务增强处理Bean，指定事务管理器 -->
67. **<tx:advice** id="transactionAdvice" transaction-manager="transactionManager"**>**
68. <!-- 配置详细事务处理语义 -->
69. **<tx:attributes>**
70. **<tx:method** name="insert\*" propagation="REQUIRED" **/>**
71. **<tx:method** name="update\*" propagation="REQUIRED" **/>**
72. **<tx:method** name="delete\*" propagation="REQUIRED" **/>**
74. **<tx:method** name="get\*" propagation="SUPPORTS" read-only="true" **/>**
75. **<tx:method** name="find\*" propagation="SUPPORTS" read-only="true" **/>**
76. **<tx:method** name="select\*" propagation="SUPPORTS" read-only="true" **/>**
77. **<tx:method** name="load\*" propagation="SUPPORTS" read-only="true" **/>**
79. <!-- 其他采用默认事务方式 -->
80. **<tx:method** name="\*" **/>**
82. **</tx:attributes>**
83. **</tx:advice>**
85. <!-- Spring aop事务管理 -->
86. **<aop:config>**
87. <!-- 配置切入点 -->
88. **<aop:pointcut** id="transactionPointcut"
89. expression="execution(\* org.andy.work.service..\*Impl.\*(..))" **/>**
90. <!-- 指定在txAdvice切入点应用txAdvice事务增强处理 -->
91. **<aop:advisor** pointcut-ref="transactionPointcut"
92. advice-ref="transactionAdvice" **/>**
93. **</aop:config>**
95. **</beans>**

     上面配置了二级缓存，ehcache，相关的具体信息查看上一篇**[Hibernate二级缓存以及ehcache的搭建配置](http://blog.csdn.net/fengshizty/article/details/43603611" \t "_blank)**，下面是二级缓存的配置。

**4.1、ehcache.xml的配置**

**[html]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<ehcache** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../config/ehcache.xsd"**>**
3. **<diskStore** path="D:/ehcache" **/>**
4. <!-- DefaultCache setting. -->
5. **<defaultCache**
6. maxElementsInMemory="1000"
7. eternal="false"
8. timeToIdleSeconds="120"
9. timeToLiveSeconds="120"
10. maxElementsOnDisk="1000000"
11. overflowToDisk="true"
12. memoryStoreEvictionPolicy="LRU"**>**
14. **</defaultCache>**
16. <!-- Special objects setting. -->
18. **<cache**
19. name="org.andy.work.entity.AcctUser"
20. maxElementsInMemory="2"
21. memoryStoreEvictionPolicy="LRU"
22. eternal="true"
23. diskPersistent="false"
24. overflowToDisk="false"
25. maxElementsOnDisk="1000000" **/>**


29. **</ehcache>**

   上面Hibernate二级缓存ehcache存放磁盘是D:/ehcache

**5、生成业务entity实体**

      设计好数据表后，Hibernate的实体有工具Hibernate tools自动生成，Eclipse需要安装Hibernate tools插件（我在线安装过几次都不成功，推荐使用links离线安装，附录将介绍），所以不需要手工的书写，下面就贴一个用户信息的实体，AcctUser.java

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.entity;
3. // Generated 2015-2-3 10:43:00 by Hibernate Tools 4.0.0
5. **import** java.util.Date;
6. **import** java.util.HashSet;
7. **import** java.util.Set;
9. **import** javax.persistence.Column;
10. **import** javax.persistence.Entity;
11. **import** javax.persistence.FetchType;
12. **import** javax.persistence.Id;
13. **import** javax.persistence.JoinColumn;
14. **import** javax.persistence.JoinTable;
15. **import** javax.persistence.ManyToMany;
16. **import** javax.persistence.Table;
17. **import** javax.persistence.Temporal;
18. **import** javax.persistence.TemporalType;
20. **import** org.hibernate.annotations.Cache;
21. **import** org.hibernate.annotations.CacheConcurrencyStrategy;
23. **import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;
25. /\*\*
26. \* AcctUser generated by hbm2java
27. \*/
28. @Entity
29. @Table(name = "acct\_user", catalog = "work")
30. @Cache(usage = CacheConcurrencyStrategy.READ\_WRITE)
31. **public** **class** AcctUser **implements** java.io.Serializable {
33. /\*\*
34. \*
35. \*/
36. **private** **static** **final** **long** serialVersionUID = 6980093847795726310L;
37. **private** String id;
38. **private** String nickName;
39. **private** String telephone;
40. **private** Date registerTime;
41. **private** Set<AcctRole> acctRoles = **new** HashSet<AcctRole>(0);
43. **public** AcctUser() {
45. }
47. **public** AcctUser(String id, String nickName) {
48. **this**.id = id;
49. **this**.nickName = nickName;
50. }
52. **public** AcctUser(String id, String nickName, String telephone,
53. Date registerTime, Set<AcctRole> acctRoles) {
54. **this**.id = id;
55. **this**.nickName = nickName;
56. **this**.telephone = telephone;
57. **this**.registerTime = registerTime;
58. **this**.acctRoles = acctRoles;
59. }
61. @Id
62. @Column(name = "id", unique = **true**, nullable = **false**, length = 36)
63. **public** String getId() {
64. **return** **this**.id;
65. }
67. **public** **void** setId(String id) {
68. **this**.id = id;
69. }
71. @Column(name = "nick\_name", nullable = **false**)
72. **public** String getNickName() {
73. **return** **this**.nickName;
74. }
76. **public** **void** setNickName(String nickName) {
77. **this**.nickName = nickName;
78. }
80. @Column(name = "telephone")
81. **public** String getTelephone() {
82. **return** **this**.telephone;
83. }
85. **public** **void** setTelephone(String telephone) {
86. **this**.telephone = telephone;
87. }
89. @Temporal(TemporalType.TIMESTAMP)
90. @Column(name = "register\_time", length = 19)
91. **public** Date getRegisterTime() {
92. **return** **this**.registerTime;
93. }
95. **public** **void** setRegisterTime(Date registerTime) {
96. **this**.registerTime = registerTime;
97. }
99. @JsonIgnoreProperties(value={"acctUsers", "acctAuthorities"})
100. @ManyToMany(fetch = FetchType.LAZY)
101. @Cache(usage = CacheConcurrencyStrategy.READ\_WRITE)
102. @JoinTable(name = "acct\_user\_role", catalog = "work", joinColumns = { @JoinColumn(name = "user\_id", nullable = **false**, updatable = **false**) }, inverseJoinColumns = { @JoinColumn(name = "role\_id", nullable = **false**, updatable = **false**) })
103. **public** Set<AcctRole> getAcctRoles() {
104. **return** **this**.acctRoles;
105. }
107. **public** **void** setAcctRoles(Set<AcctRole> acctRoles) {
108. **this**.acctRoles = acctRoles;
109. }
111. }

**上面有几个需要注意的**：

     上面大多数都是Hibernate tools自动生成的，但是有几个需要手动添加：

              1、如果这个实体使用二级缓存的话，需要添加@cache注解；

              2、如果该实体里面还有集合元素（set， map，list），如果实体要采用二级缓存，**那么这些集合元素必须也添加@cache注解**。

              3、@JsonIgnoreProperties注解是为了防止SpringMVC在json返回时产生循环输出使用，如果不配置的话就会**出现json死循环**（还有多对多，一对多的属性时）。

             4、@JsonIgnore这个是在转换成json是忽略该属性，而@JsonIgnoreProperties(value={"acctUsers", "acctAuthorities"})则是忽略acctRoles里面的acctUsers和acctAuthorites属性。

**6、创建Dao层**

**6.1Dao接口**

              Dao层和Service层我们肯定采用的是面相接口编程的思想，所以，我们先定义一个通用的Dao接口，GenericDao.java

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.dao;
3. **import** java.io.Serializable;
4. **import** java.util.List;
6. /\*\*
7. \* 创建时间：2015-2-6 下午2:26:42
8. \*
9. \* @author andy
10. \* @version 2.2
11. \*
12. \* Dao通用接口
13. \*/
15. **interface** GenericDao<T, PK **extends** Serializable> {
17. T load(PK id);
19. T get(PK id);
21. List<T> findAll();
23. **void** persist(T entity);
25. PK save(T entity);
27. **void** saveOrUpdate(T entity);
29. **void** delete(PK id);
31. **void** flush();
32. }

            在定义具体的UserDao.java接口

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.dao;
3. **import** org.andy.work.entity.AcctUser;
5. /\*\*
6. \* 创建时间：2015-2-6 下午2:43:50
7. \*
8. \* @author andy
9. \* @version 2.2
10. \*
11. \* 用户Dao接口
12. \*/
14. **public** **interface** UserDao **extends** GenericDao<AcctUser, String> {
16. }

**6.2、Dao层实现类**

    我们需要将Dao层的实现类注入为bean，所以需要添加@Repository注解，UserDaoImpl如下：

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.dao.impl;
3. **import** java.util.List;
5. **import** org.andy.work.dao.UserDao;
6. **import** org.andy.work.entity.AcctUser;
7. **import** org.hibernate.Session;
8. **import** org.hibernate.SessionFactory;
9. **import** org.springframework.beans.factory.annotation.Autowired;
10. **import** org.springframework.stereotype.Repository;
12. /\*\*
13. \* 创建时间：2015-2-6 下午2:45:14
14. \*
15. \* @author andy
16. \* @version 2.2
17. \*/
18. @Repository("userDao")
19. **public** **class** UserDaoImpl **implements** UserDao {
21. @Autowired
22. **private** SessionFactory sessionFactory;
24. **private** Session getCurrentSession() {
25. **return** **this**.sessionFactory.getCurrentSession();
26. }
28. @Override
29. **public** AcctUser load(String id) {
30. **return** (AcctUser) **this**.getCurrentSession().load(AcctUser.**class**, id);
31. }
33. @Override
34. **public** AcctUser get(String id) {
35. **return** (AcctUser) **this**.getCurrentSession().get(AcctUser.**class**, id);
36. }
38. @SuppressWarnings("unchecked")
39. @Override
40. **public** List<AcctUser> findAll() {
41. List<AcctUser> acctUsers = **this**.getCurrentSession().createQuery("from AcctUser").setCacheable(**true**).list();
42. **return** acctUsers;
43. }
45. @Override
46. **public** **void** persist(AcctUser entity) {
47. **this**.getCurrentSession().persist(entity);
49. }
51. @Override
52. **public** String save(AcctUser entity) {
53. **return** (String) **this**.getCurrentSession().save(entity);
54. }
56. @Override
57. **public** **void** saveOrUpdate(AcctUser entity) {
58. **this**.getCurrentSession().saveOrUpdate(entity);
59. }
61. @Override
62. **public** **void** delete(String id) {
63. AcctUser entity = **this**.load(id);
64. **this**.getCurrentSession().delete(entity);
65. }
67. @Override
68. **public** **void** flush() {
69. **this**.getCurrentSession().flush();
71. }
73. }

**7、创建Service层**

**7.1、Service层接口**

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.service;
3. **import** java.util.List;
5. **import** org.andy.work.entity.AcctUser;
7. /\*\*
8. \* 创建时间：2015-2-6 下午3:18:57
9. \*
10. \* @author andy
11. \* @version 2.2
12. \*  userService接口
13. \*/
15. **public** **interface** UserService {
16. AcctUser load(String id);
18. AcctUser get(String id);
20. List<AcctUser> findAll();
22. **void** persist(AcctUser entity);
24. String save(AcctUser entity);
26. **void** saveOrUpdate(AcctUser entity);
28. **void** delete(String id);
30. **void** flush();
31. }

**7.2、service层实现类**

   UserServiceImpl就需要注入上面定义的Dao层bean。

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.service.impl;
3. **import** java.util.List;
5. **import** org.andy.work.dao.UserDao;
6. **import** org.andy.work.entity.AcctUser;
7. **import** org.andy.work.service.UserService;
8. **import** org.springframework.beans.factory.annotation.Autowired;
9. **import** org.springframework.stereotype.Service;
11. /\*\*
12. \* 创建时间：2015-2-6 下午3:24:16
13. \*
14. \* @author andy
15. \* @version 2.2 UserService 的实现
16. \*/
17. @Service("userService")
18. **public** **class** UserServiceImpl **implements** UserService {
20. @Autowired
21. **private** UserDao userDao;
23. @Override
24. **public** AcctUser load(String id) {
25. **return** userDao.load(id);
26. }
28. @Override
29. **public** AcctUser get(String id) {
30. **return** userDao.get(id);
31. }
33. @Override
34. **public** List<AcctUser> findAll() {
35. **return** userDao.findAll();
36. }
38. @Override
39. **public** **void** persist(AcctUser entity) {
40. userDao.persist(entity);
41. }
43. @Override
44. **public** String save(AcctUser entity) {
45. **return** userDao.save(entity);
46. }
48. @Override
49. **public** **void** saveOrUpdate(AcctUser entity) {
50. userDao.saveOrUpdate(entity);
51. }
53. @Override
54. **public** **void** delete(String id) {
55. userDao.delete(id);
56. }
58. @Override
59. **public** **void** flush() {
60. userDao.flush();
61. }
63. }

**8、测试Spring和Hibernate的整合**

    我们可以在src/test/java编写测试类，测试上面的配置，如果测试成功已经基本大功告成。

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.service;
3. **import** java.util.Date;
4. **import** java.util.List;
5. **import** java.util.UUID;
7. **import** org.andy.work.entity.AcctUser;
8. **import** org.apache.log4j.Logger;
9. **import** org.junit.Test;
10. **import** org.junit.runner.RunWith;
11. **import** org.springframework.beans.factory.annotation.Autowired;
12. **import** org.springframework.test.context.ContextConfiguration;
13. **import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
15. **import** com.alibaba.fastjson.JSON;
17. /\*\*
18. \* 创建时间：2015-2-6 下午3:31:07
19. \*
20. \* @author andy
21. \* @version 2.2
22. \*/
23. @RunWith(SpringJUnit4ClassRunner.**class**)
24. @ContextConfiguration(locations = { "classpath:spring.xml",
25. "classpath:spring-hibernate.xml" })
26. **public** **class** TestUserService {
28. **private** **static** **final** Logger LOGGER = Logger
29. .getLogger(TestUserService.**class**);
31. @Autowired
32. **private** UserService userService;
34. @Test
35. **public** **void** save() {
36. AcctUser acctUser = **new** AcctUser();
37. acctUser.setId(UUID.randomUUID().toString());
38. acctUser.setNickName("andy");
39. acctUser.setRegisterTime(**new** Date());
40. acctUser.setTelephone("13022221111");
41. String id = userService.save(acctUser);
42. LOGGER.info(JSON.toJSONString(id));
43. }
45. }

**9、引入SpringMVC**

     添加spring-mvc.xml文件

**[html]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:p="http://www.springframework.org/schema/p" xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="http://www.springframework.org/schema/beans
6. http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
7. http://www.springframework.org/schema/context
8. http://www.springframework.org/schema/context/spring-context-4.1.xsd
9. http://www.springframework.org/schema/mvc
10. http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd"**>**
12. <!-- 自动扫描@Controller注入为bean -->
13. **<context:component-scan** base-package="org.andy.work.controller" **/>**
15. <!-- 以下为SpringMVC配置 -->
16. **<mvc:annotation-driven>**
17. <!-- 返回json数据，@response使用 -->
18. **<mvc:message-converters** register-defaults="true"**>**
19. **<bean**
20. class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter"**>**
21. **<property** name="supportedMediaTypes"**>**
22. **<list>**
23. **<value>**text/html;charset=UTF-8**</value>**
24. **<value>**application/json;charset=UTF-8**</value>**
25. **</list>**
26. **</property>**
28. **</bean>**
29. **</mvc:message-converters>**
30. **</mvc:annotation-driven>**
32. <!-- 对模型视图名称的解析，即在模型视图名称添加前后缀 -->
33. **<bean**
34. class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**
35. **<property** name="viewClass"
36. value="org.springframework.web.servlet.view.JstlView" **/>**
37. **<property** name="prefix" value="/WEB-INF/views" **/>**
38. **<property** name="suffix" value=".jsp" **/>**
39. **</bean>**
41. **</beans>**

   上面配置了包扫描Controller，将@Controller注入为spring的bean。以及springmvc的json处理，和视图层的配置。

**10、配置web.xml容器**

      web容器是web项目的大脑，所以web容器，首先需要引入spring，让spring管理各个框架，并将其注入为bean。然后配置控制层的filter信息。并且配置了springmvc拦截的url为以.hmls结尾的请求。

web.xml文件如下：

**[html]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"
5. id="WebApp\_ID" version="2.5"**>**
7. **<display-name>**springmvc\_demo**</display-name>**
9. **<context-param>**
10. **<param-name>**contextConfigLocation**</param-name>**
11. **<param-value>**classpath:spring.xml,classpath:spring-hibernate.xml**</param-value>**
12. **</context-param>**
14. **<filter>**
15. **<filter-name>**encodingFilter**</filter-name>**
16. **<filter-class>**org.springframework.web.filter.CharacterEncodingFilter**</filter-class>**
17. **<init-param>**
18. **<param-name>**encoding**</param-name>**
19. **<param-value>**utf-8**</param-value>**
20. **</init-param>**
21. **<init-param>**
22. **<param-name>**forceEncoding**</param-name>**
23. **<param-value>**true**</param-value>**
24. **</init-param>**
25. **</filter>**
26. **<filter-mapping>**
27. **<filter-name>**encodingFilter**</filter-name>**
28. **<url-pattern>**/\***</url-pattern>**
29. **</filter-mapping>**
31. <!-- openSessionInView配置 作用是延迟session关闭到view层 -->
32. **<filter>**
33. **<filter-name>**openSessionInViewFilter**</filter-name>**
34. **<filter-class>**org.springframework.orm.hibernate4.support.OpenSessionInViewFilter**</filter-class>**
35. **<init-param>**
36. **<param-name>**singleSession**</param-name>**
37. **<param-value>**true**</param-value>**
38. **</init-param>**
39. **</filter>**
41. <!-- 监听servletContext，启动contextConfigLocation中的spring配置信息 -->
42. **<listener>**
43. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
44. **</listener>**
46. <!-- 防止spring内存溢出监听器 -->
47. **<listener>**
48. **<listener-class>**org.springframework.web.util.IntrospectorCleanupListener**</listener-class>**
49. **</listener>**
51. **<servlet>**
52. **<description>**spring mvc servlet**</description>**
53. **<servlet-name>**rest**</servlet-name>**
54. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
55. **<init-param>**
56. **<param-name>**contextConfigLocation**</param-name>**
57. <!-- 此处配置的是SpringMVC的配置文件 -->
58. **<param-value>**classpath:spring-mvc.xml**</param-value>**
59. **</init-param>**
60. **<load-on-startup>**1**</load-on-startup>**
61. **</servlet>**
63. **<servlet-mapping>**
64. **<servlet-name>**rest**</servlet-name>**
65. **<url-pattern>**\*.htmls**</url-pattern>**
66. **</servlet-mapping>**
68. **<filter-mapping>**
69. **<filter-name>**openSessionInViewFilter**</filter-name>**
70. **<url-pattern>**\*.htmls**</url-pattern>**
71. **</filter-mapping>**
73. <!-- 配置session超时时间，单位分钟 -->
74. **<session-config>**
75. **<session-timeout>**30**</session-timeout>**
76. **</session-config>**
78. **<welcome-file-list>**
79. **<welcome-file>**/index.jsp**</welcome-file>**
80. **</welcome-file-list>**
81. **</web-app>**

**11、创建控制层Controller**

     控制层UserController

**[java]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **package** org.andy.work.controller;
3. **import** java.util.List;
5. **import** org.andy.work.entity.AcctUser;
6. **import** org.andy.work.service.UserService;
7. **import** org.apache.log4j.Logger;
8. **import** org.springframework.beans.factory.annotation.Autowired;
9. **import** org.springframework.stereotype.Controller;
10. **import** org.springframework.ui.ModelMap;
11. **import** org.springframework.web.bind.annotation.PathVariable;
12. **import** org.springframework.web.bind.annotation.RequestMapping;
13. **import** org.springframework.web.bind.annotation.ResponseBody;
15. /\*\*
16. \* 创建时间：2015-2-7 上午11:49:00
17. \* @author andy
18. \* @version 2.2
19. \* 描述： 用户Controller
20. \*/
21. @Controller
22. @RequestMapping("/user")
23. **public** **class** UserController {
25. **private** **static** **final** Logger LOGGER = Logger.getLogger(UserController.**class**);
27. @Autowired
28. **private** UserService userService;
30. @RequestMapping("/showInfo/{userId}")
31. **public** String showUserInfo(ModelMap modelMap, @PathVariable String userId){
32. LOGGER.info("查询用户：" + userId);
33. AcctUser userInfo = userService.load(userId);
34. modelMap.addAttribute("userInfo", userInfo);
35. **return** "/user/showInfo";
36. }
38. @RequestMapping("/showInfos")
39. **public** @ResponseBody List<AcctUser> showUserInfos(){
40. LOGGER.info("查询用户全部用户");
41. List<AcctUser> userInfos = userService.findAll();
42. **return** userInfos;
43. }
44. }

**12、创建视图层**

   在src/main/webapp/WEB-INF/views下 创建user/showInfo.jsp

**[html]** [view plaincopy](http://blog.csdn.net/fengshizty/article/details/43635305)[在CODE上查看代码片](https://code.csdn.net/snippets/600252)

1. **<**%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%**>**
2. **<**%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%**>**
3. **<**%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%**>**
5. **<**%
6. String path = request.getContextPath();
7. String basePath = request.getScheme() + "://"
8. + request.getServerName() + ":" + request.getServerPort()
9. + path + "/";
10. %**>**
11. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"**>**
12. **<html>**
13. **<head>**
14. **<base** href="<%=basePath%>" **/>**
15. **<meta** http-equiv="Content-Type" content="text/html; charset=UTF-8"**>**
16. **<script** type="text/javascript" src="js/jquery-1.9.1.min.js"**></script>**
17. **<title>**userInfo**</title>**
18. **</head>**
19. **<body>**
20. 用户信息 昵称： ${userInfo.nickName} 用户id：${userInfo.id} 用户电话:${userInfo.telephone } 注册时间：
21. **<fmt:formatDate** value="${userInfo.registerTime }" pattern="yyyy-MM-dd HH:mm:ss" **/>**
22. 角色：[
23. **<c:forEach** items="${ userInfo.acctRoles}" var="role"**>**
24. ${role.name }   权限[
25. **<c:forEach** items="${ role.acctAuthorities}" var="authority"**>**
26. ${authority.name }
27. **</c:forEach>** ]
28. **</c:forEach>**
29. ]
31. **<br** **/>** ajax显示全部用户信息：
32. **<div** id="show\_all\_user"**></div>**
33. **</body>**
34. **<script** type="text/javascript"**>**
35. $.ajax({
36. type : "get",
37. url : "user/showInfos.htmls",
38. dataType : "json",
39. success : function(data) {
40. $(data).each(
41. function(i, user) {
42. var p = "<p>昵称:" + user.nickName + "    电话:"
43. + user.telephone + "    注册时间:"
44. + user.registerTime + "    id:" + user.id +
45. "**</p>**";
46. $("#show\_all\_user").append(p);
47. });
48. },
49. async : true
50. });
51. **</script>**
52. **</html>**

**13、部署服务器测试**

          使用Maven打包部署：clean compile  package

          部署到tomcat，测试

          主要测试上面的  http://localhost:8080/springmvc\_hibernate\_demo/user/showInfo/6e5afb1d-50e1-45fe-b6fe-b9e399415387.htmls

                                  和http://localhost:8080/springmvc\_hibernate\_demo/user/showInfos.htmls  （json数据返回）

          上面视图层就包含了这两条url请求的测试：



     ok，到此Spring+SpringMVC+Hibernate搭建完毕。

**博客来源：**<http://blog.csdn.net/fengshizty?viewmode=list>

**项目源码：**<http://download.csdn.net/detail/fengshizty/8432647>