# [Spring整合Hibernate图文步骤](http://blog.csdn.net/qq7342272/article/details/7928814)

标签： [hibernate](http://www.csdn.net/tag/hibernate)[spring](http://www.csdn.net/tag/spring)[数据库连接池](http://www.csdn.net/tag/%e6%95%b0%e6%8d%ae%e5%ba%93%e8%bf%9e%e6%8e%a5%e6%b1%a0)[user](http://www.csdn.net/tag/user)[class](http://www.csdn.net/tag/class)

2012-08-31 15:34 28016人阅读 [评论](http://blog.csdn.net/qq7342272/article/details/7928814#comments)(2) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/qq7342272/article/details/7928814#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

JAVA（65） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

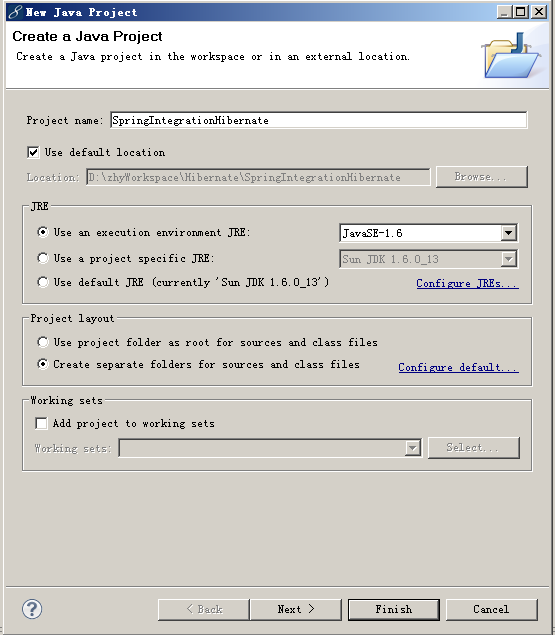
工具：myeclipse9.0

Spring版本：2.5.6

Hibernate版本：3

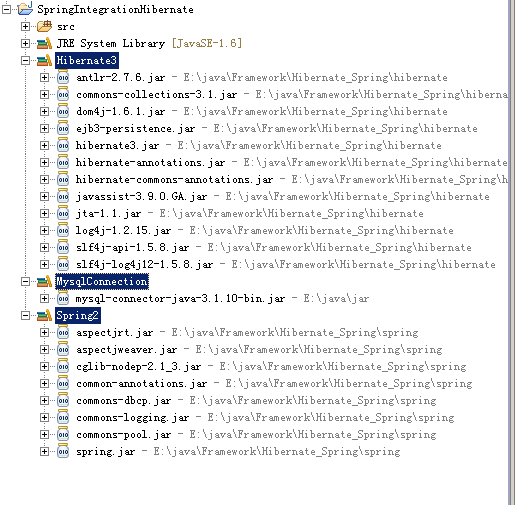
昨天花了一下午时间把Spring和Hibernate整合到了一起，今天做一个笔记。

首先建立java Project工程



点击Finish完成

添加Hibernate和Spring所需要的jar包还有Mysql连接的jar包



创建Dao层，Dao层实现，Model层，Service层

DAO层代码：IUserDao

**[java]** [view plain](http://blog.csdn.net/qq7342272/article/details/7928814) [copy](http://blog.csdn.net/qq7342272/article/details/7928814)

1. **package** org.zhy.demo.dao;
3. **import** java.sql.SQLException;
5. **import** org.zhy.demo.dao.model.UserInfo;
6. /\*\*
7. \* DAO层简单的增删改查方法
8. \* @author Administrator
9. \*
10. \*/
11. **public** **interface** IUserDao {
13. **public** **void** saveUser(UserInfo user) **throws** SQLException;
15. **public** **void** delUser(UserInfo user) **throws** SQLException;
17. **public** **void** editUsre(UserInfo user) **throws** SQLException;
19. **public** UserInfo getUserById(**int** id) **throws** SQLException;
21. }

IUuserDao层实现（暂时空实现）

**[html]** [view plain](http://blog.csdn.net/qq7342272/article/details/7928814) [copy](http://blog.csdn.net/qq7342272/article/details/7928814)

1. package org.zhy.demo.dao.impl;
3. import java.sql.SQLException;
5. import org.zhy.demo.dao.IUserDao;
6. import org.zhy.demo.dao.model.UserInfo;
8. /\*\*
9. \* IUserDao实现，配置完Spring及Hibernate文件后实现此类
10. \* @author Administrator
11. \*
12. \*/
13. public class IUserDaoImpl implements IUserDao {
15. @Override
16. public void saveUser(UserInfo user) throws SQLException {
18. }
20. @Override
21. public void delUser(UserInfo user) throws SQLException {
23. }
25. @Override
26. public void editUsre(UserInfo user) throws SQLException {
28. }
30. @Override
31. public UserInfo getUserById(int id) throws SQLException {
32. return null;
33. }
35. }

Model代码

**[java]** [view plain](http://blog.csdn.net/qq7342272/article/details/7928814) [copy](http://blog.csdn.net/qq7342272/article/details/7928814)

1. **package** org.zhy.demo.dao.model;
3. **import** javax.persistence.Entity;
4. **import** javax.persistence.GeneratedValue;
5. **import** javax.persistence.Id;
7. @Entity
8. **public** **class** UserInfo {

11. **private** **int** id;
12. **private** String name;
13. **private** String title;
15. @Id
16. @GeneratedValue
17. **public** **int** getId() {
18. **return** id;
19. }
21. **public** **void** setId(**int** id) {
22. **this**.id = id;
23. }
25. **public** String getName() {
26. **return** name;
27. }
29. **public** **void** setName(String name) {
30. **this**.name = name;
31. }
33. **public** String getTitle() {
34. **return** title;
35. }
37. **public** **void** setTitle(String title) {
38. **this**.title = title;
39. }
41. }

下面添加Spring配置文件

在Src目录下添加SpringContext.xml

**[html]** [view plain](http://blog.csdn.net/qq7342272/article/details/7928814) [copy](http://blog.csdn.net/qq7342272/article/details/7928814)

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:aop="http://www.springframework.org/schema/aop"
5. xmlns:tx="http://www.springframework.org/schema/tx"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
8. http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
9. http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
10. "**>**
11. **<aop:config></aop:config>**
13. **<bean** class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor" **/>**
15. **<bean** id="datasource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close"**>**
16. **<property** name="driverClassName" value="com.mysql.jdbc.Driver" **/>**
17. **<property** name="url" value="jdbc:mysql://localhost/student"**></property>**
18. **<property** name="username" value="root" **/>**
19. **<property** name="password" value="root" **/>**
20. <!-- 数据库连接池保持的最小连接数 -->
21. **<property** name="minIdle" value="5" **/>**
22. <!-- 数据库连接池保持的最大连接数 -->
23. **<property** name="maxIdle" value="30" **/>**
24. <!--
25. 当数据库连接因为某种原因断掉之后，再重新从连接池中拿另外一个连接时实际上这个连接可能
26. 已经无效，所以为了确保所拿到的连接全都有效需要在获取连接，返回连接以及连接空闲时进行
27. 有效性验证 下面3个设置为ture时进行验证，默认为false
28. --**>**
29. <!-- 取得连接时是否进行有效性验证 -->
30. **<property** name="testOnBorrow" value="true" **/>**
31. <!-- 返回连接时是否进行有效性验证 -->
32. **<property** name="testOnReturn" value="true" **/>**
33. <!-- 连接空闲时是否进行有效性验证 -->
34. **<property** name="testWhileIdle" value="true" **/>**
36. **</bean>**
38. **<bean** id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"**>**
39. **<property** name="dataSource" ref="datasource" **/>**
40. <!-- 注意：我用的是Annotation的方式配置的Hibernate，这里的property的name是annotatedClasses -->
41. **<property** name="annotatedClasses"**>**
42. **<list>**
43. **<value>**org.zhy.demo.dao.model.UserInfo**</value>**
44. **</list>**
45. **</property>**
46. **<property** name="hibernateProperties"**>**
47. **<props>**
48. <!-- 设置Hibernate方言 -->
49. **<prop** key="hibernate.dialect"**>**org.hibernate.dialect.MySQLDialect**</prop>**
50. <!-- 是否打印sql -->
51. **<prop** key="hibernate.show\_sql"**>**true**</prop>**
52. <!-- 格式化sql -->
53. **<prop** key="hibernate.format\_sql"**>**true**</prop>**
54. <!-- 是否自动更新表 -->
55. **<prop** key="hibernate.hbm2ddl.auto"**>**update**</prop>**
56. **<prop** key="hibernate.current\_session\_context\_class"**>**thread**</prop>**
57. <!-- 最大抓取深度，如果为0，则关闭默认的外连接抓取。建议值为0-3 -->
58. **<prop** key="hibernate.max\_fetch\_depth"**>**1**</prop>**
59. <!-- 用于生成有助于调试的注释信息，默认为关闭 -->
60. **<prop** key="hibernate.use\_sql\_comments"**>**true**</prop>**
61. **</props>**
62. **</property>**
63. **</bean>**
65. **<bean** id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager" **>**
66. **<property** name="sessionFactory" ref="sessionFactory"**></property>**
67. **</bean>**
69. **<aop:config>**
70. **<aop:pointcut** id="txMethod" expression="execution(\* org.zhy.demo.dao.impl.\*DaoImpl.\*(..))" **/>**
71. **<aop:advisor** advice-ref="txAdvice" pointcut-ref="txMethod"**/>**
72. **</aop:config>**
73. <!-- AOP切面声明事务管理 -->
74. **<tx:advice** id="txAdvice" transaction-manager="transactionManager"**>**
75. **<tx:attributes>**
76. **<tx:method** name="save\*" propagation="REQUIRED" **/>** <!-- 支持当前事务，如果执行到save开头的任何方法时没有事务则开启一个事务 这是最常见的方式-->
77. **<tx:method** name="update\*" propagation="REQUIRED" **/>**<!-- 支持当前事务，如果执行到save开头的任何方法时没有事务则开启一个事务 这是最常见的方式-->
78. **<tx:method** name="add\*" propagation="REQUIRED" **/>**<!-- 支持当前事务，如果执行到save开头的任何方法时没有事务则开启一个事务 这是最常见的方式-->
79. **<tx:method** name="delete\*" propagation="REQUIRED" **/>**<!-- 支持当前事务，如果执行到save开头的任何方法时没有事务则开启一个事务 这是最常见的方式-->
80. **<tx:method** name="find\*" propagation="SUPPORTS" read-only="true"**/>** <!-- 支持当前事务，如果当前没有事务，就以非事务方式执行。只读 -->
81. **<tx:method** name="get\*" propagation="SUPPORTS" read-only="true"**/>**<!-- 支持当前事务，如果当前没有事务，就以非事务方式执行。只读 -->
82. **<tx:method** name="\*" **/>**
83. **</tx:attributes>**
84. **</tx:advice>**

87. **<bean** name="userDao" class="org.zhy.demo.dao.impl.IUserDaoImpl" **>**
88. **<property** name="sessionFactory" ref="sessionFactory"**></property>**
89. **</bean>**
90. **<bean** name="userService" class="org.zhy.demo.service.UserService" **/>**
92. **</beans>**

修改DAO实现类

**[java]** [view plain](http://blog.csdn.net/qq7342272/article/details/7928814) [copy](http://blog.csdn.net/qq7342272/article/details/7928814)

1. **package** org.zhy.demo.dao.impl;
3. **import** java.sql.SQLException;
5. **import** org.hibernate.SessionFactory;
6. **import** org.springframework.orm.hibernate3.HibernateTemplate;
7. **import** org.zhy.demo.dao.IUserDao;
8. **import** org.zhy.demo.dao.model.UserInfo;
10. /\*\*
11. \* IUserDao实现
12. \*
13. \* @author Administrator
14. \*
15. \*/
16. **public** **class** IUserDaoImpl **implements** IUserDao {
18. **private** HibernateTemplate hibernateTemplate;
20. **public** **void** setSessionFactory(SessionFactory sessionFactory) {
21. **this**.hibernateTemplate = **new** HibernateTemplate(sessionFactory);
22. }
24. @Override
25. **public** **void** saveUser(UserInfo user) **throws** SQLException {
26. hibernateTemplate.save(user);
27. }
29. @Override
30. **public** **void** delUser(UserInfo user) **throws** SQLException {
31. hibernateTemplate.delete(user);
32. }
34. @Override
35. **public** **void** editUsre(UserInfo user) **throws** SQLException {
36. hibernateTemplate.update(user);
37. }
39. @Override
40. **public** UserInfo getUserById(**int** id) **throws** SQLException {
41. UserInfo user = (UserInfo) hibernateTemplate.get(UserInfo.**class**, id);
42. **return** user;
43. }
45. }

junit测试

**[java]** [view plain](http://blog.csdn.net/qq7342272/article/details/7928814) [copy](http://blog.csdn.net/qq7342272/article/details/7928814)

1. @Test
2. **public** **void** saveUserTest() **throws** SQLException{
3. ApplicationContext context= **new** ClassPathXmlApplicationContext("/SpringContext.xml");
4. UserService service =(UserService) context.getBean("userService");
6. UserInfo user = **new** UserInfo();
7. user.setName("T`");
8. user.setTitle("CSDN BLOG");
9. service.addUserInfo(user);
10. }

# [使用注解方式进行spring和hibernate整合](http://blog.csdn.net/witsmakemen/article/details/43151989)

2015-01-26 14:17 8450人阅读 [评论](http://blog.csdn.net/witsmakemen/article/details/43151989#comments)(3) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/witsmakemen/article/details/43151989#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

struts+hibernate+spring 学习（10） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

整合spring和hibernate需要五个要素，分别包括持久化的类， 数据源，SessionFactory， TransactionManager和持久化操作的DAO类。

持久化类：

**[java]** [view plain](http://blog.csdn.net/witsmakemen/article/details/43151989) [copy](http://blog.csdn.net/witsmakemen/article/details/43151989)

[在CODE上查看代码片](https://code.csdn.net/snippets/589286)

1. @Entity
2. **public** **class** Spitter {
3. **private** **long** id;
4. **private** String userName, passWord, fullName;
6. **public** Spitter(**long** id, String n, String p, String f){
7. **this**.id = id;
8. **this**.userName = n;
9. **this**.passWord = p;
10. **this**.fullName = f;
11. }
12. **public** Spitter(){}
13. **public** **void** setId(**long** id){
14. **this**.id = id;
15. }
16. @Id
17. **public** **long** getId(){
18. **return** id;
19. }
20. **public** String getUserName(){
21. **return** **this**.userName;
22. }
23. **public** **void** setUserName(String n){
24. **this**.userName = n;
25. }
26. **public** String getPassWord(){
27. **return** **this**.passWord;
28. }
29. **public** **void** setPassWord(String p){
30. **this**.passWord = p;
31. }
32. **public** String getFullName(){
33. **return** **this**.fullName;
34. }
35. **public** **void** setFullName(String f){
36. **this**.fullName = f;
37. }
39. }

数据源（在spring配置文件中配置）：

**[html]** [view plain](http://blog.csdn.net/witsmakemen/article/details/43151989) [copy](http://blog.csdn.net/witsmakemen/article/details/43151989)

[在CODE上查看代码片](https://code.csdn.net/snippets/589286)

1. **<bean** id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" **>**
2. **<property** name="driverClassName" value="com.mysql.jdbc.Driver" **/>**
3. **<property** name="url" value="jdbc:mysql://localhost:3306/spitter" **/>**
4. **<property** name="username" value="root" **/>**
5. **<property** name="password" value="root" **/>**
6. **<property** name="initialSize" value="5"**/>**
7. **<property** name="maxActive" value="10" **/>**
8. **</bean>**

SessionFactory类（在spring配置文件中配置）：

**[html]** [view plain](http://blog.csdn.net/witsmakemen/article/details/43151989) [copy](http://blog.csdn.net/witsmakemen/article/details/43151989)

[在CODE上查看代码片](https://code.csdn.net/snippets/589286)

1. **<bean** id="sessionFactory"
2. class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"**>**
3. **<property** name="dataSource" ref="dataSource" **/>**
4. **<property** name="annotatedClasses"**>**
5. **<list>**
6. **<value>**Spitter.spitterOne.Spitter**</value>**
7. **</list>**
8. **</property>**
10. **<property** name="hibernateProperties"**>**
11. **<props>**
12. **<prop** key="dialect"**>**org.hibernate.dialect.MySQLInnoDBDialect**</prop>**
13. **<prop** key="hibernate.show\_sql"**>**true**</prop>**
14. **<prop** key="hibernate.format\_sql"**>**true**</prop>**
15. **<prop** key="hibernate.hbm2ddl.auto"**>**update**</prop>**
17. **</props>**
18. **</property>**
20. **</bean>**

配置hibernate事务：

**[html]** [view plain](http://blog.csdn.net/witsmakemen/article/details/43151989) [copy](http://blog.csdn.net/witsmakemen/article/details/43151989)

[在CODE上查看代码片](https://code.csdn.net/snippets/589286)

1. <!-- 设定transactionManager -->
2. **<bean** id="txManager"
3. class="org.springframework.orm.hibernate3.HibernateTransactionManager"**>**
4. **<property** name="sessionFactory" ref="sessionFactory" **/>**
5. **</bean>**
7. <!--启动spring事务注解功能-->
8. **<tx:annotation-driven** transaction-manager="txManager"**/>**

进行持久化操作的DAO类：

**[java]** [view plain](http://blog.csdn.net/witsmakemen/article/details/43151989) [copy](http://blog.csdn.net/witsmakemen/article/details/43151989)

[在CODE上查看代码片](https://code.csdn.net/snippets/589286)

1. @Repository
2. **public** **class** HibernateSpitterDao **implements** SpitterDAO {
3. **private** SessionFactory sessionFactory;
5. @Autowired
6. **public** HibernateSpitterDao(SessionFactory sessionFactory){
7. **this**.sessionFactory = sessionFactory;
8. }
9. **private** Session currentSession(){
10. **return** **this**.sessionFactory.getCurrentSession();
11. }
12. /\*\*
13. \* 进行持久化的方法需要使用@Transactional进行事务管理
14. \*/
15. @Transactional(readOnly = **false**, rollbackFor = RuntimeException.**class**)
16. **public** **void** addSpitter(Spitter spitter){
17. **this**.currentSession().save(spitter);
18. }
19. **public** Spitter getSpitterById(**long** id){
20. **return** (Spitter)**this**.currentSession().get(Spitter.**class**, id);
21. }
22. @Transactional(readOnly = **false**, rollbackFor = RuntimeException.**class**)
23. **public** **void** saveSpitter(Spitter spitter){
24. **this**.currentSession().update(spitter);
25. }
26. **public** **static** **void** main(String [] args){
27. Spitter ss = **new** Spitter(103,"zhang sfdasf454352an", "cccninini", "zhang shan fdasfsdfewe");
29. ApplicationContext ctx = **new** ClassPathXmlApplicationContext("Spitter/spitterOne/spring-idol.xml");
30. <span style="color:#ff0000;">SpitterDAO dao = (SpitterDAO) ctx.getBean("hibernateSpitterDao");</span>
31. dao.addSpitter(ss);
33. }
34. }

因为Spring只能对接口进行aop操作，所以红色代码部分只能将hibernateSpitterDao强制转换成SpitterDAO接口。