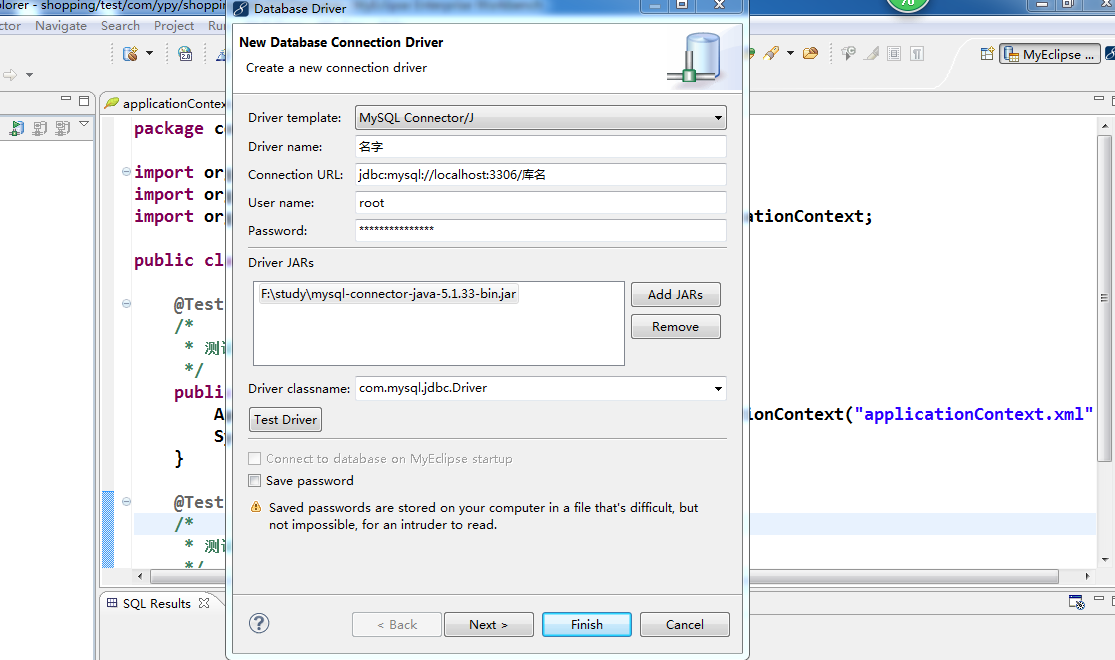
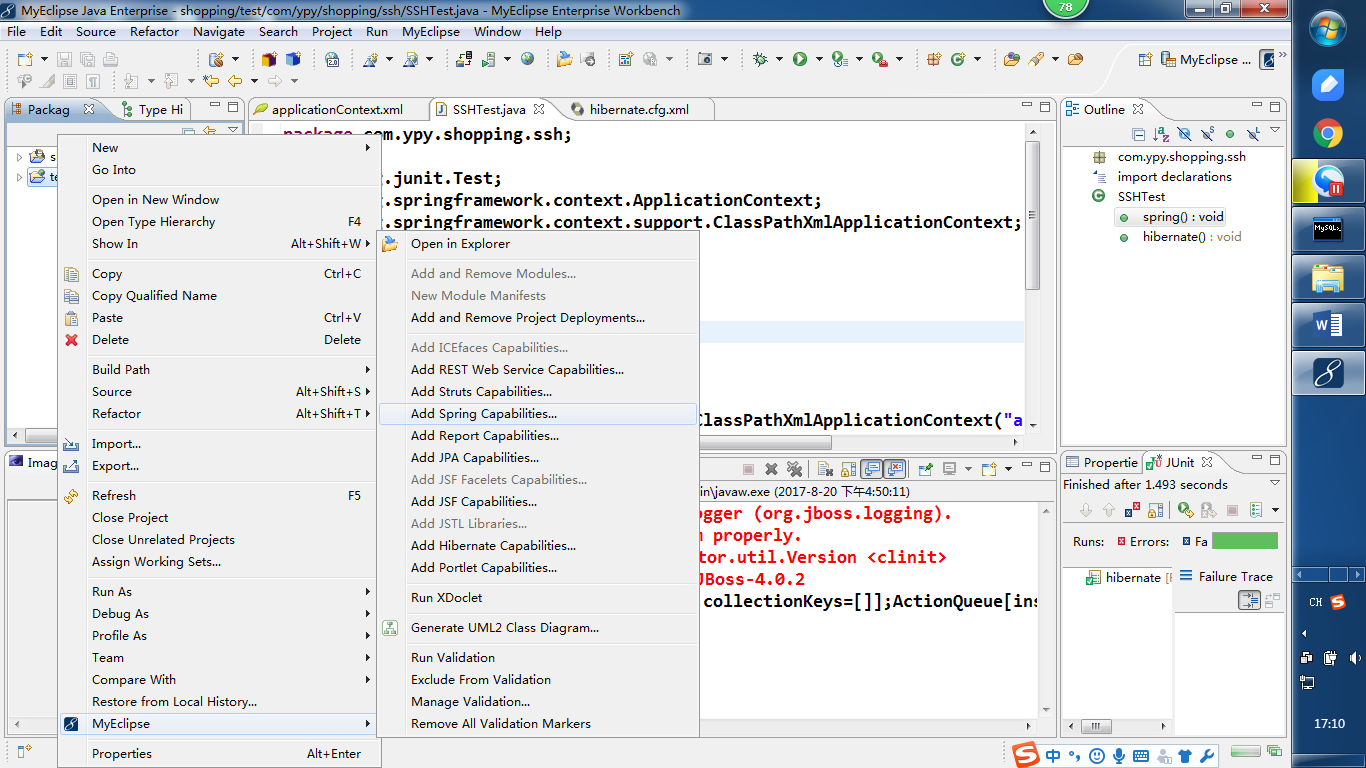
MyEclipse中整合

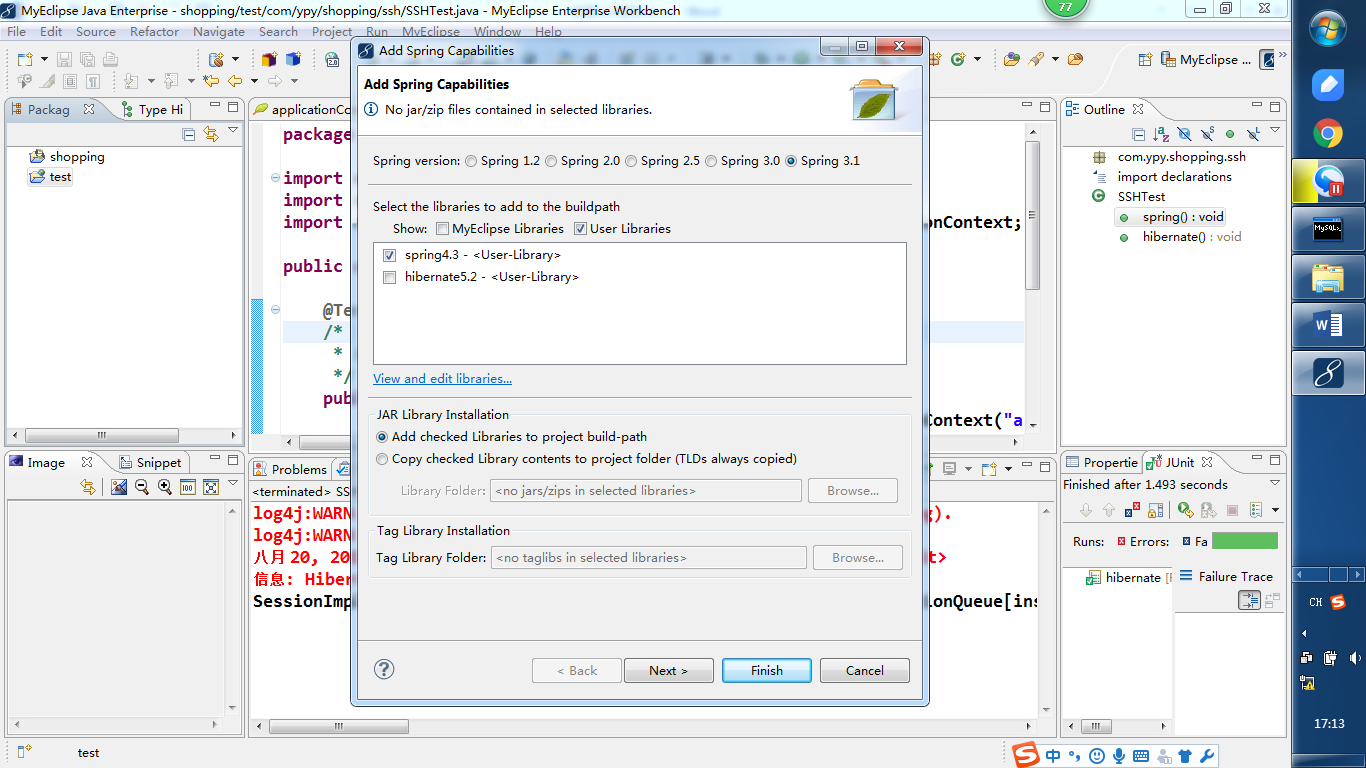
## 先创建一个datebase driver

右上角切换到datebase界面，新建



## 配置spring





其中，spring4.3的Library需要自己新建并导入spring所需jar包。

然后next->finish。

此时src文件中多出一个applicationContext.xml文件，在其中添加一句测试信息：<bean id=*"date"* class=*"java.util.Date"*/>。

测试：项目右击，new->Source Folder，写Junit测试文件

@Test

/\*

\* 测试spring

\*/

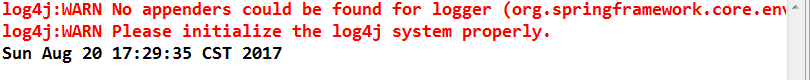
public void spring() {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

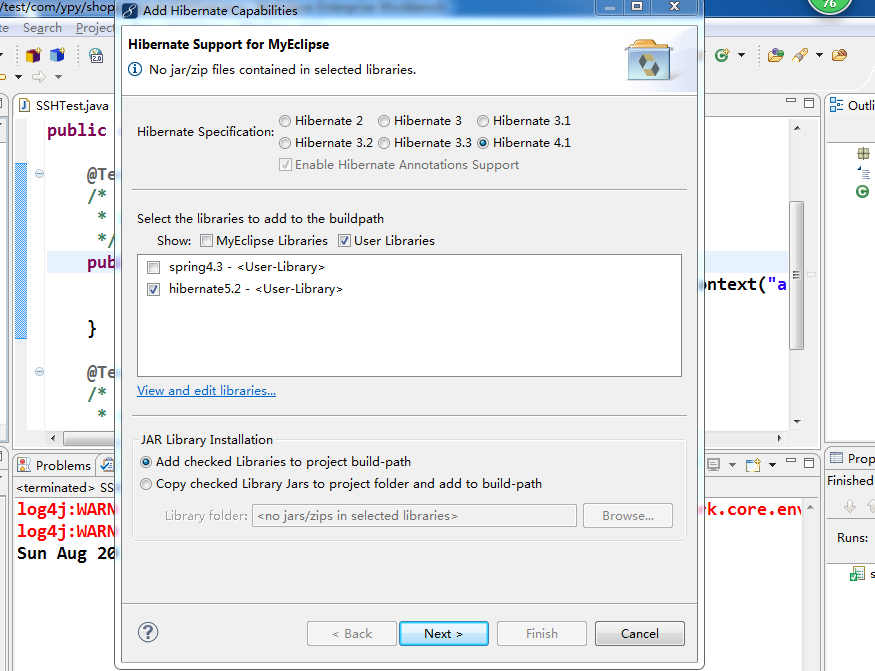
System.out.println(context.getBean("date"));

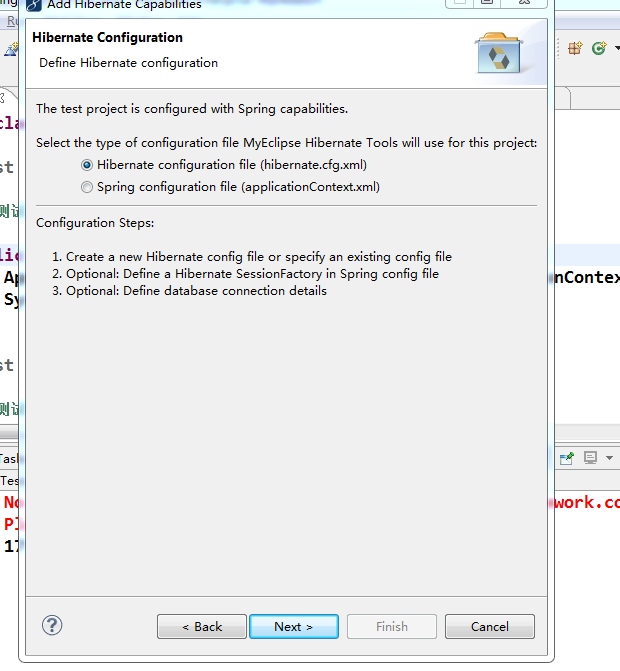
}

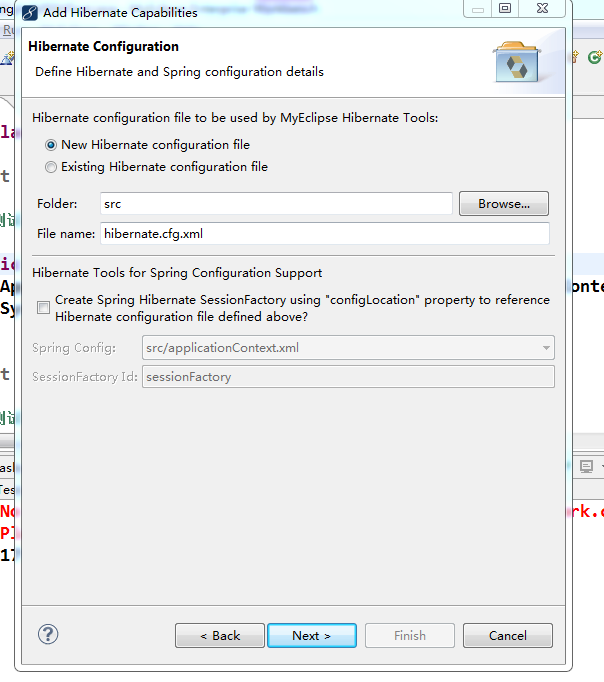
运行结果：

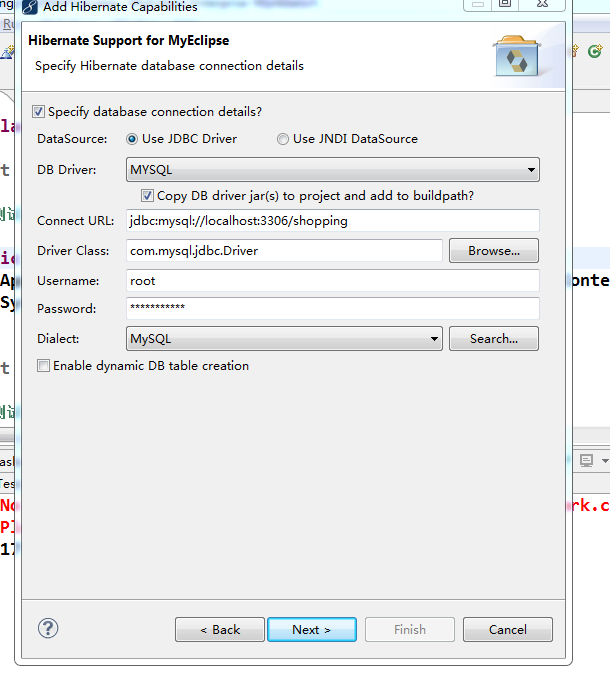


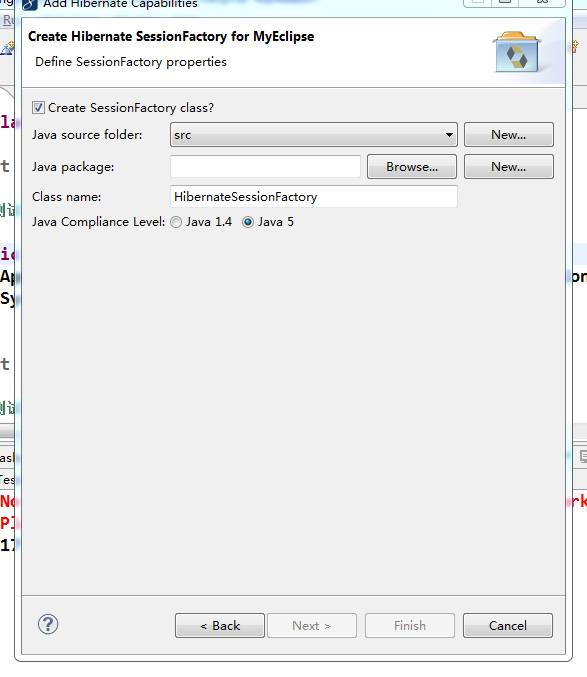
## 配置hibernate











这里要选择测试文件所在包。

……finish。

此时，src文件夹中会出现hibernate.cfg.xml文件，测试文件同目录下会出现HibernateSessionFactory.java文件，如果是hibernate5以上版本，需要更改HibernateSessionFactory.java文件中的serviceRegistry = new Configuration().configure().buildSessionFactory(); 为 sessionFactory = new Configuration().configure().buildSessionFactory();。

测试：

在前面写的测试文件中写：

@Test

/\*

\* 测试hibernate

\*/

public void hibernate() {

System.out.println(HibernateSessionFactory.getSession());

}

## 整合spring和hibernate（注：hibernate4前后的版本配置方式不同，hibernate4以后版本必须在applicationContext.xml文件中配置数据库）

1. 新建资源文件夹config，将配置文件都放在此文件夹中。
2. 在applicationContext.xml文件中添加：

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:tx=<http://www.springframework.org/schema/tx>

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-4.3.xsd

http://www.springframework.org/schema/tx

<http://www.springframework.org/schema/tx/spring-tx.xsd>

如果是hibernate4以后的版本，还需要添加：

xmlns:context=<http://www.springframework.org/schema/context>

http://www.springframework.org/schema/context

<http://www.springframework.org/schema/context/spring-context.xsd>

按注释提示添加以下内容：

<!--

1:Spring来管理Hibernate配置文件，就是取代了HibernateSessionFactory

2:通过SessionFactory，返回session，在spring由hibernate封装session

3:如果要实现Spring的AOP声明式事务，则需要Spring提供事务管理器，所有session都是从SessionFactory创建的，

事务管理器，需要控制SessionFactory

4:配置事物通知(通知就是运行时动态切入的方法,Spring已经把事物模块化,只需要配置那些方法被切入那种事务类型即可)

5:配置切入表达式,让后与事务通知整合,组成完整的切入表达式

6:hibernate4以后的版本需要在applicationContext文件中配置数据库，不再需要hibernate.cfg.xml

-->

<!-- 1 -->

<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

<!-- 加载hibernate配置文件 -->

<!-- <property name="configLocation" value="classpath:hibernate.cfg.xml"/> --> <!-- hibernate4以前版本 -->

<property name="dataSource" ref="dataSource"/> <!-- hibernate4以后版本 -->

<!-- 配置hibernate -->

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>

<prop key='hibernate.hbm2ddl.auto'>update</prop>

<prop key="hibernate.show\_sql">true</prop>

<prop key="hibernate.format\_sql">true</prop>

</props>

</property>

</bean>

<!-- 2 -->

<bean id="hibernateTemplate" class="org.springframework.orm.hibernate5.HibernateTemplate">

<!-- 需要sessionFactory -->

<property name="sessionFactory" ref="sessionFactory"/>

</bean>

<!-- 3 -->

<bean id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManager">

<property name="sessionFactory" ref="sessionFactory"/>

</bean>

<!-- 4 配置事务的管理者 -->

<!-- <tx:annotation-driven transaction-manager="transactionManager"/> --> <!-- 注释配置 -->

<tx:advice id="txAdvice" transaction-manager="transactionManager">

<tx:attributes>

<tx:method name="someOtherBusinessMethod" propagation="REQUIRES\_NEW"/>

<tx:method name="add\*" propagation="REQUIRED"/>

</tx:attributes>

</tx:advice>

<!-- 5:切入表达式 -->

<aop:config>

<aop:pointcut id="productServiceMethods"

expression="execution(public \* com.ypy.service..\*.\*(..))"/>

<aop:advisor advice-ref="txAdvice" pointcut-ref="productServiceMethods"/>

</aop:config>

<!-- 6:配置数据库（hibernate4以后版本需要） -->

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">

<property name="driverClass" value="${jdbc.driverClassName}"/>

<property name="jdbcUrl" value="${jdbc.url}"/>

<property name="user" value="${jdbc.username}"/>

<property name="password" value="${jdbc.password}"/>

</bean>

<!-- 引入数据库的具体配置 -->

<context:property-placeholder location="classpath:jdbc.properties"/>

1. 添加完成后，hibernate4以后版本，需要在config文件夹下添加一个jdbc.properties配置文件，里面写：

jdbc.driverClassName=com.mysql.jdbc.Driver

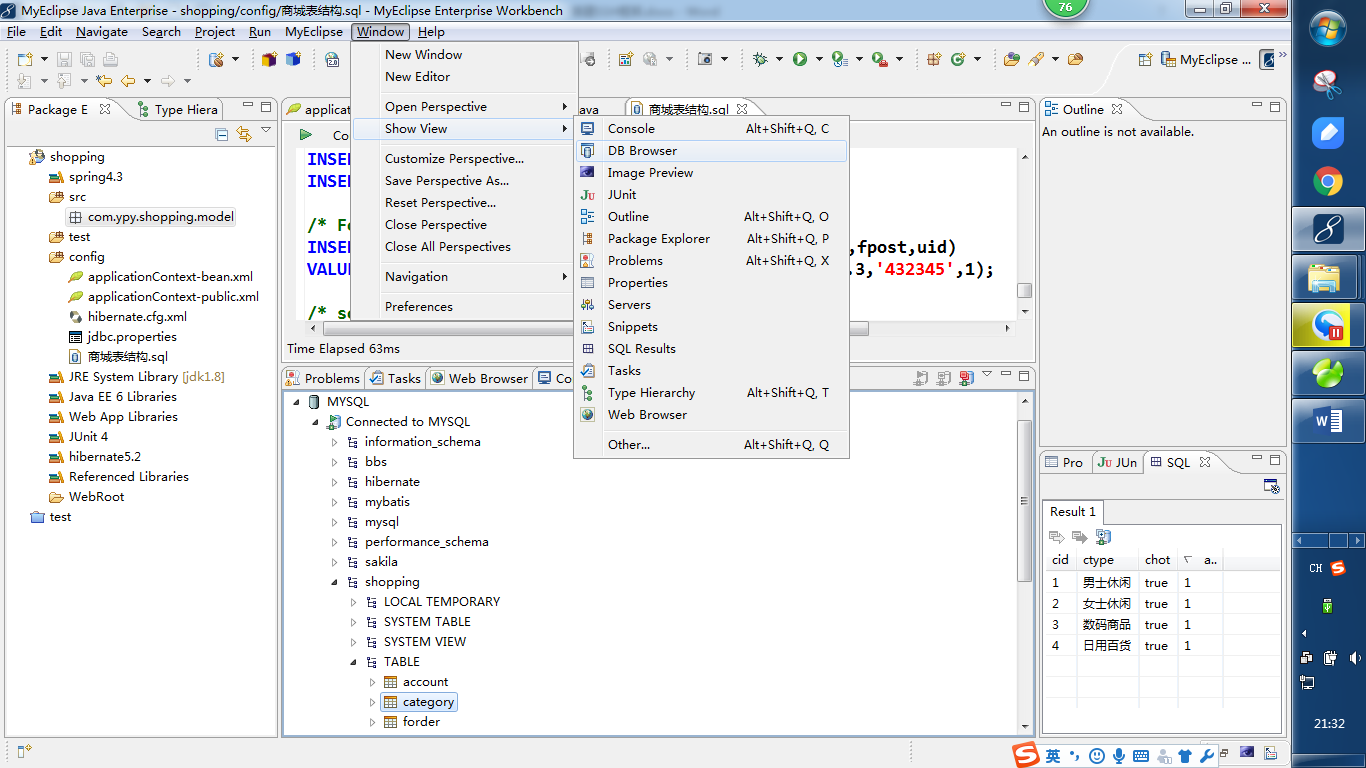
jdbc.url=jdbc\:mysql\://localhost\:3306/shopping

jdbc.username=root

jdbc.password=12345678900

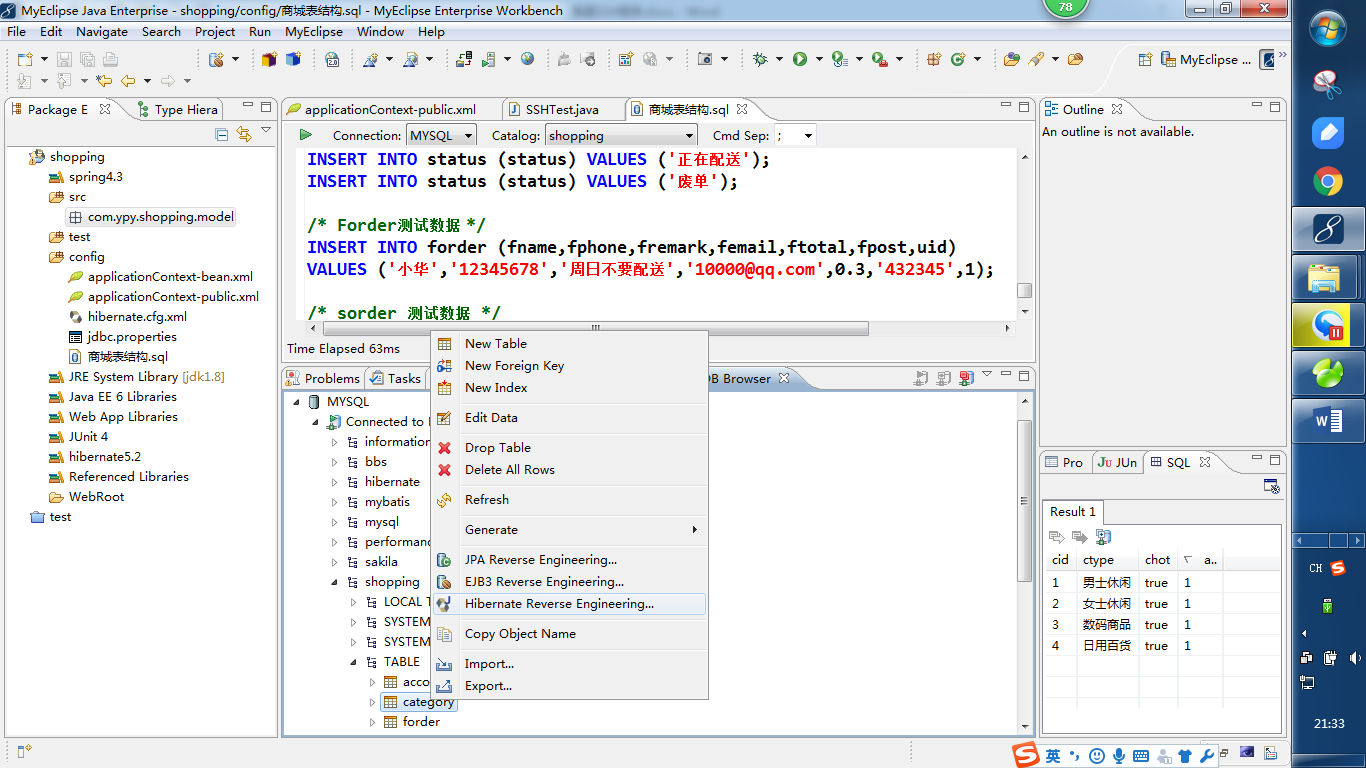
而且在applicationContext.xml文件中配置完数据库信息，就不再需要hibernate.cfg.xml文件了。

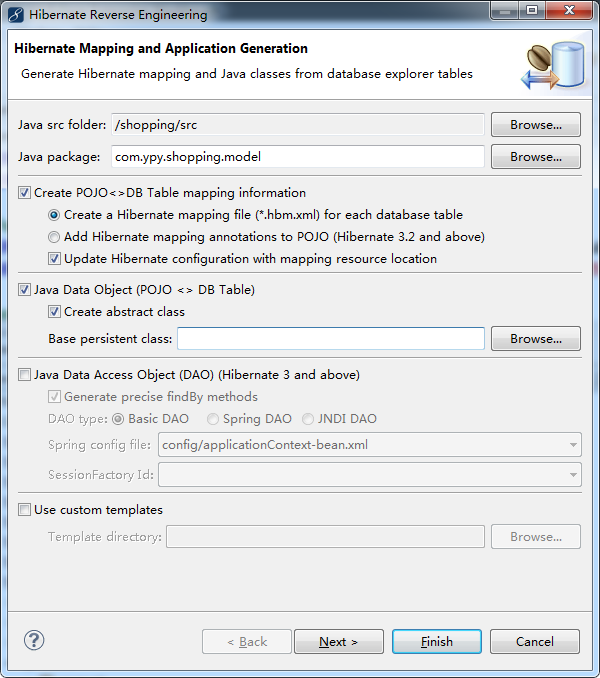
1. 配置完后，先测试文件中的spring()方法。
2. 测试成功后加入数据库测试数据，若无测试表，可按下述操作进行。
3. 无DB Browser视图的先显示出该视图

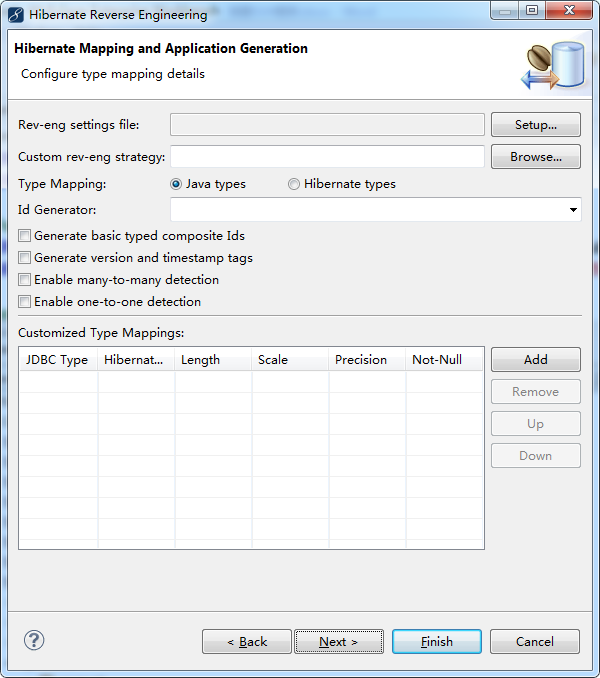


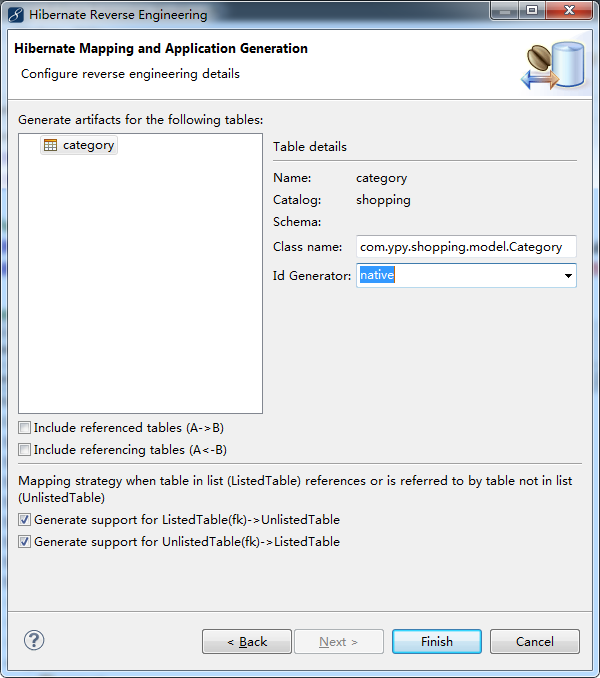
将sql语句放入config文件，打开，选择相应的数据库执行就可完成建表和添加数据。（若无对应数据库驱动则按本文第一步添加，然后在datebase explorer中双击打开）。

新建存放实体类的包，然后按照下述步骤创建并关联实体类（已经创建并关联实体类的可以省去）

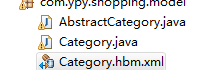








新建完成后将出现如下文件：



1. 新建service的实现包，以及对应实体类的业务实现类，本例中是CategoryServiceImpl.java文件。

内容：

package com.ypy.shopping.service.impl;

import org.springframework.orm.hibernate5.HibernateTemplate;

import com.ypy.shopping.model.Category;

public class CategoryServiceImpl {

private HibernateTemplate hibernateTemplate = null;

public void setHibernateTemplate(HibernateTemplate hibernateTemplate) {

this.hibernateTemplate = hibernateTemplate;

}

public void save(Category category) {

hibernateTemplate.save(category);

}

}

1. 在applicationContext.xml文件中添加以下内容：（对应自己的包名和实体名）

<bean id="categoryServiceImpl" class="com.ypy.shopping.service.impl.CategoryServiceImpl">

<property name="hibernateTemplate" ref="hibernateTemplate" />

</bean>

以及在id=sessionFactory的bean中添加以下内容：（hibernate4以后版本，以前版在hibernate.cfg.xml中配置）

<property name="mappingResources">

<list>

<value>com/ypy/shopping/model/Category.hbm.xml</value>

</list>

</property>

1. 在测试文件中写对spring整合hibernate的测试：

@Test

/\*

\* 测试spring与hibernate整合

\*/

public void springHibernate() {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext-\*.xml");

CategoryServiceImpl categoryServiceImpl = (CategoryServiceImpl) context.getBean("categoryServiceImpl");

Category category = new Category("test", false, 1);

categoryServiceImpl.save(category);

}

运行后查询数据库中，若添加成功则配置成功。

## 配置Struts2

1. 导入除struts2-spring-plugin-2.3.4.1.jar以外所有的Struts2中的包。
2. 在新建好的Struts.xml文件中写以下内容：

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

<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">

<struts>

<constant name="struts.configuration.xml.reload" value="true"/>

<constant name="struts.enable.DynamicMethodInvocation" value="true" />

<package name=*"default"* namespace=*"/"* extends=*"struts-default"*>

<action name=*"\*\_Action"* class=*"com.ypy.shopping.action.TestAction"* method=*"{1}"*><!-- 直接通过action的属性也能接收(注意：调的是方法，属性名不同也可以，只要方法中set,get后面的名字和参数名相同就行)，不用request.getParameter -->

<result name=*"success"*>

/success.jsp

</result>

</action>

</package>

</struts>

1. 根据配置文件的内容新建action文件：TestAction.java

package com.ypy.shopping.action;

public class TestAction {

public String test() {

System.out.println(-------------test------------);

return "success";

}

}

1. 新建jsp文件，index.jsp和返回成功的success.jsp文件

Index.jsp中写访问action的超链接/shopping/test\_Action

1. 配置web.xml文件：

<filter>

<filter-name>struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置applicationContext.xml的路径 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext-\*.xml</param-value>

</context-param>

1. 启动Tomcat，运行index.jsp进行测试。

## 整合Struts2和spring

1. 加入struts2-spring-plugin-2.3.4.1.jar包
2. 在web.xml文件中添加如下内容：

<!-- 配置spring結合Struts2 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

1. 修改TestAction.java文件：

package com.ypy.shopping.action;

import com.ypy.shopping.service.impl.CategoryServiceImpl;

public class TestAction {

private CategoryServiceImpl categoryServiceImpl=null;

public void setCategoryServiceImpl(CategoryServiceImpl categoryServiceImpl) {

this.categoryServiceImpl = categoryServiceImpl;

}

public String test() {

System.out.println(categoryServiceImpl);

return "success";

}

}

1. 在applicationContext.xml配置文件中添加：（spring中配置Struts2）

<bean id="testAction" class="com.ypy.shopping.action.TestAction">

<property name="categoryServiceImpl" ref="categoryServiceImpl" />

</bean>

1. 修改struts.xml配置文件：

<package name=*"default"* namespace=*"/"* extends=*"struts-default"*>

<action name=*"\*\_Action"* class=*"com.ypy.shopping.action.TestAction"* method=*"{1}"*>中的class的值改为在applicationContext.xml中配置Struts2的id，本例中为testAction。

1. 测试index.jsp文件，成功则配置完成！