软件工程 ---- Junit测试

    由于我们的项目是基于ssh框架做的，我现在做junit单元测试，就在做的过程中遇到很多问题：

1. 测试Service的方法的时候，service都是调用Dao的，在service里面Dao的问题。用Junit测试dao，可以写测试类DaoTest继承TestCase，测试方法的声明一般是这样的：public void testAdd(){}，测试方法里，dao的c、r、u、d操作都可以测试。由于J2EE,是面向接口编程的，我们定义了 DAO 接口和 Service 接口，在测试时我们也可以构造实现接口的 Mock 类。通过spring依赖注入，才能在测试时 set Mock 对象来模拟对象的行为。

2. 如：view层，测试Struts中的action,在action里request和response的模拟。网上说要用StrutsTestCase（下载<http://sourceforge.net/projects/strutstestcase/>

 AND 简介<http://strutstestcase.sourceforge.net/>

 AND 类<http://strutstestcase.sourceforge.net/api/index.html> ）。

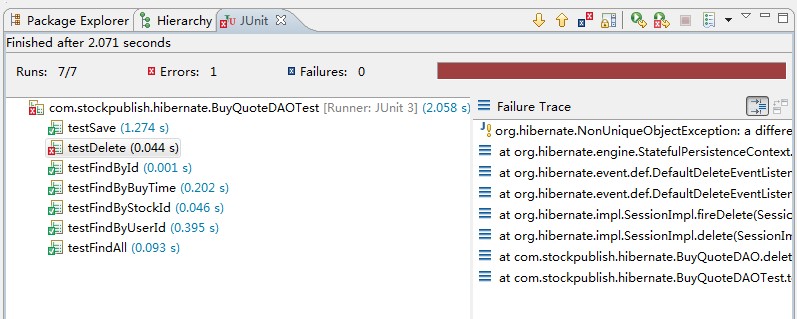
3.coverageEclEmma是一个基于EMMA的Java代码覆盖工具。它的目的是让你可以在Eclipse工作平台中使用强大的Java代码覆盖工具EMMA。EclEmma是非侵入式的不需要修改你的项目或执行其它任何安装，它能够在工作平台中启动像运行JUnit测试一样直接对代码覆盖进行分析。覆盖结果将立即被汇总并在Java源代码编辑器中高亮显示。由于网上(<http://www.eclemma.org/> )都是在讲Eclipse怎么用这个，我用MyEclipse在线装了这个插件就出现了各种错误，比如说不能正常使用，比如MyEclipse sever不能正常工作，比如说不用MyEclipse的web，比如说我两个没有错的网站一装就错了，后来重装了MyEclipse8.5,然后我试了试在6.5上装又可以，可是工程不晓得怎么回事在6.5又运行不了。另一个Coverlipse这个Eclipse插件能够把JUnit测试的代码覆盖直观化。它的独到之处就是与Eclipse开发平台无缝结合。Coverlipse(<http://coverlipse.sourceforge.net/index.php> )能够在一个JUnit运行之后马上显示覆盖结果并可把结果导成XML文件。但是它也是个eclipse plugin，在MyEclipse里也不是很好弄，最后还是没有使用成功。

Junit测试:

由于本项目除了比较完善的Exception处理机制，而且有比较完善的系统运行日志，另外，为了让客户由一个比较直观的，对系统运行的正确性看法，我们对系统的DAO类进行JUnit测试，JUint实际上会有比较多的地方好处。Junit框架，继承Test Case类，可以用Junit进行自动测试。Junit有断言功能，如果测试结果不通过会那个测试不通过Junit的好处就是这个结果是否正确的判断是它来完成的，只需要看看它告诉结果是否正确就可以了，在一般情况下会大大提高效率。一个TestSuite对像可以运行一组测试。TestSuite和TestCase都实现了Test接口，而Test接口定义了运行测试所需的方法。这就允许用TestCase和TestSuite的组合创建一个TestSuite。

**※BuyQuoteDAO.java**

对BuyQuoteDAO.java进行测试：



![C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\N[I{WOVGU`U45@5R660F](5.jpg](data:image/jpeg;base64,)测试 C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\PJ$39LSK6X4(0F}NFGD9FRU.jpg

（用来save BuyQuote instance至数据库）

C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\87VKW$64W9SLV7IBZO(X9QN.jpg测试 C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\DYI_(E5@3_(Q(Y_G5[C0S`S.jpg

（用来get BuyQuote instance with id，用在相应B数据库uyId里面的id和不在里面的id进行测试都是正常的）

C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\L`TR$N59EL@)UBY_)Y~$}7K.jpg测试 C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\WN]BHO@}_UU(Q]W92EQ2~5C.jpg

用buyTime找到一个所对应元组（相应的也用了在其中和不在其中的值来找都正确）

C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\%I0K%E0_9TZ_`QLDX[ML6~9.jpg测试 C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\P~BJOYKI]_TUGK@5W7@9OGI.jpg

用stockId找到一个所对应元组（相应的也用了在其中和不在其中的值来找都正确）

C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\W)XJPNF[%7$3[9~I5SO06HC.jpg测试C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\~TUFX1R{P9Q{$[BO{U5YX5D.jpg

用userId找到一个所对应元组（相应的也用了在其中和不在其中的值来找都正确）

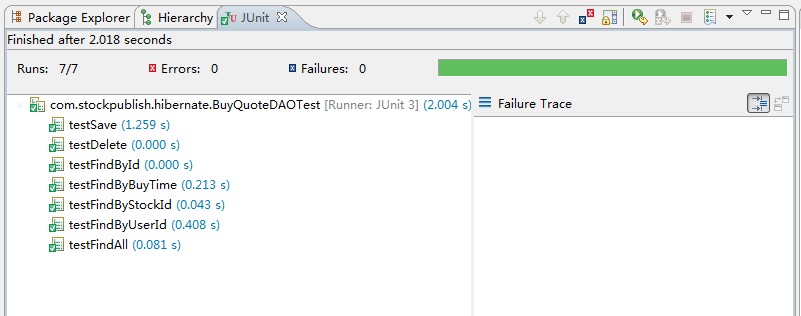
C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\0HNWDYHXDX06AAUY~RHQKPG.jpg测试 C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\~H1%WGKHO2W8DSF)LXO[C~P.jpg

用来find all BuyQuote instances（测试是都通过了的，我看了看源码源码，其中也是用catch (RuntimeException re) {

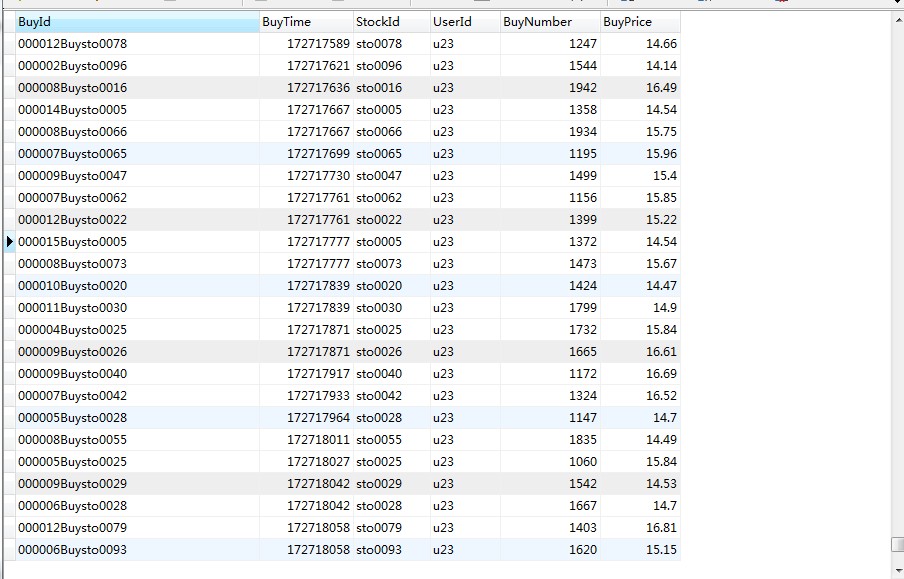
log.error("find all failed", re);

throw re;}来处理异常的）

C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\{_]C}D8OHLRPR%%VET5BD46.jpg测试 C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\N5{LOCG)_JS%3Z2}O(5[X]P.jpg（由于没有考虑删除一个不存在在数据库中的异常处理的情况，出错了。然后我去改正了一下，后面再测试就成功了）



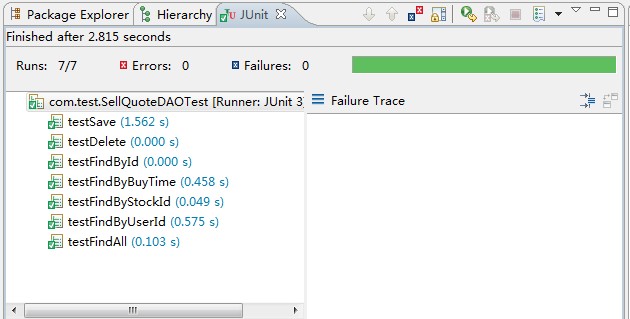
**对应数据库：**



**※SellQuoteDAO.java**

相应也地又对SellQuoteDAO进行测试：

与上面的BuyQuoteDAO比较类似，再此就不详细说明了。



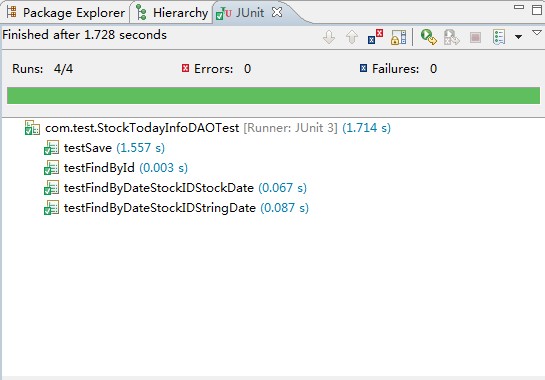
**※StockTodayInfo.java**

![D:\My Documents\Tencent Files\584183338\Image\[X684J](G[2`P6G8VYE`R1X.jpg](data:image/jpeg;base64,)

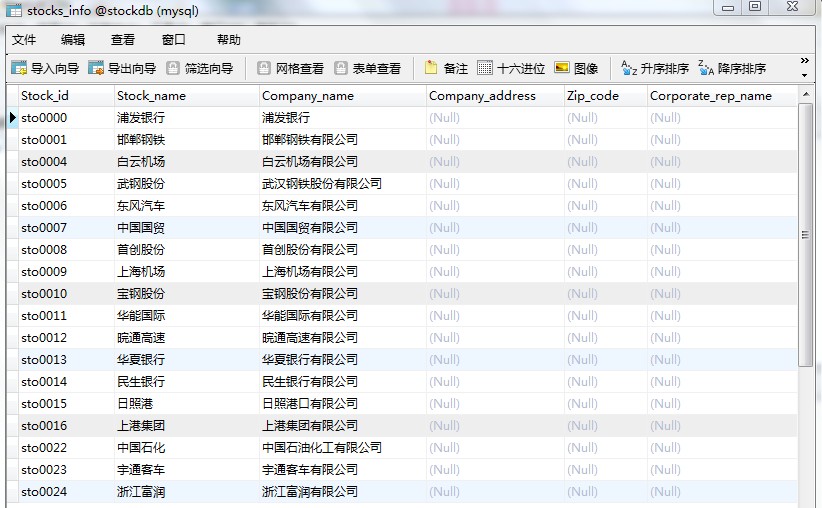
C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\Z9A(MT8G2EXL)%ESQAXE~$U.jpg测试

C:\Documents and Settings\Administrator\Application Data\Tencent\Users\584183338\QQ\WinTemp\RichOle\TCITGXYC3CVI{}H158CC)FV.jpg

因为前面测试中把这个删除了，所以测试的时候出现了问题。然后又回头去看源代码，我也记得是处理了的，因为直接用Exception，throw了。所以没有显式地返回什么提示，但实际上是处理了。然后测试另一个参数，结果如下:



**对应数据库：**



**测试感想：**

JUnit接触得很少，这次来测试ssh遇到了很多困难，很多测试的概念也不清楚，好多工具不能体会其中的妙处，以后要好好学习。测试的过程其实考虑到更多的异常的东西和各个方面的需求，更多需要完善的地方。ssh框架很难测，那些dao应该用mock+junit测试更有效率，这次是用JUnit做。struts也应该用用StrutsTestCase试试，看了看它的模拟器，可以用模拟器来模拟view层的输入输出，这次由于不熟悉StrutsTestCase类的方法，没有做成功。

*参考资料：*

1. Unit\_test\_with\_JUnit&easyMork: <http://www.michaelminella.com/testing/unit-testing-with-junit-and-easymock.html>

<http://www.51cto.com/art/200604/24841.htm>

<http://www.51cto.com/art/200604/24841.htm>

1. easyMock: <http://easymock.org/>
2. StrutsTestCase：­ <http://strutstestcase.sourceforge.net/api/index.html>
3. JUnit&ssh: <http://topic.csdn.net/u/20100902/15/5973355c-5882-4faa-9505-46a01a51b4c7.html>

<http://fantlam.javaeye.com/blog/236300>

<http://xuyufei1985.blog.163.com/blog/static/13164348200911212942824/> (DAO)

<http://www.docin.com/p-2889514.html> (DAO)

<http://junit.org/index.htm>