软件[开发手册](http://www.cnblogs.com/oucbl/p/5928511.html)

昆山华恒工程技术中心有限公司

2017年5月10日星期三

目录

[1 软件清单 4](#_Toc482189179)

[2 JDK的安装 4](#_Toc482189180)

[2.1 新建变量名：JAVA\_HOME 4](#_Toc482189181)

[2.2 编辑变量名：Path 4](#_Toc482189182)

[2.3 新建变量名：CLASSPATH 4](#_Toc482189183)

[3 Eclipse 4.6(Neon)的安装 5](#_Toc482189184)

[4 Tomcat8安装及环境变量配置 6](#_Toc482189185)

[4.1 变量名：CATALINA\_BASE 6](#_Toc482189186)

[4.2 变量名：CATALINA\_HOME 6](#_Toc482189187)

[4.3 变量名：classpath 6](#_Toc482189188)

[4.4 变量名：path 7](#_Toc482189189)

[5 Eclipse配置Tomcat 7](#_Toc482189190)

[5.1 配置路径 7](#_Toc482189191)

[5.2 选择对应版本 7](#_Toc482189192)

[5.3 选择Tomcat路径、JRE版本 8](#_Toc482189193)

[6 Maven安装配置 9](#_Toc482189194)

[6.1 maven配置环境变量 9](#_Toc482189195)

[6.2 Maven数据仓库的配置 9](#_Toc482189196)

[6.2.1 数据仓库的路径配置 9](#_Toc482189197)

[6.2.2 解除注释自定义路径 10](#_Toc482189198)

[6.3 Eclipse中集成Maven 10](#_Toc482189199)

[6.3.1 maven配置完成查看 10](#_Toc482189200)

[6.3.2 修改默认maven管理路径 11](#_Toc482189201)

[7 SVN安装与配置 11](#_Toc482189202)

[7.1 SVN安装 11](#_Toc482189203)

[7.2 SVM导入Maven工程 12](#_Toc482189204)

[7.2.1 SVN的忽略target 12](#_Toc482189205)

[7.2.2 Eclipse添加SVN资源 14](#_Toc482189206)

[8 7. Eclipse小技巧 15](#_Toc482189207)

[8.1 7.1 关闭/开启欢迎界面 15](#_Toc482189208)

[8.2 Java代码提示 16](#_Toc482189209)

[8.3 设置默认编码 17](#_Toc482189210)

[8.4 快捷键 17](#_Toc482189211)

[9 开发手册 18](#_Toc482189212)

[9.1 代码导入 18](#_Toc482189213)

[9.2 导入Maven工程 19](#_Toc482189214)

[9.3 更新Maven库 21](#_Toc482189215)

[9.4 Tomcat配置文件路径 22](#_Toc482189216)

[9.5 生成DAO、PO、xml文件 23](#_Toc482189217)

[9.6 编写Manage业务层 28](#_Toc482189218)

[9.7 常用方法案例： 29](#_Toc482189219)

[9.7.1 \*以Object的形式返回 30](#_Toc482189220)

[9.7.2 \*以List的形式返回 31](#_Toc482189221)

[9.7.3 \*以Map的形式返回 32](#_Toc482189222)

[9.7.4 update修改方法 37](#_Toc482189223)

[9.7.5 insert方法 38](#_Toc482189224)

[9.7.6 查单个对象方法 39](#_Toc482189225)

[9.8 批处理实现： 40](#_Toc482189226)

[9.9 8.9数据库切换实现： 42](#_Toc482189227)

[9.10 获取请求方IP地址实现： 45](#_Toc482189228)

# 软件清单

* jdk-8u102-windows-x64.exe
* eclipse-inst-win64.exe (Eclipse4.6 Neon)
* apache-tomcat-8.5.5-windows-x64.zip
* apache-maven-3.3.9-bin.zip
* site-1.10.13-1.8.x.zip(svn-eclipse-plug-in)

# JDK的安装

安装过程就是next一路到底；安装完成之后要配置环境变量:

## 新建变量名：JAVA\_HOME

变量值：C:\Program Files\Java\jdk1.8.0\_102（这是我的jdk安装路径）

## 编辑变量名：Path

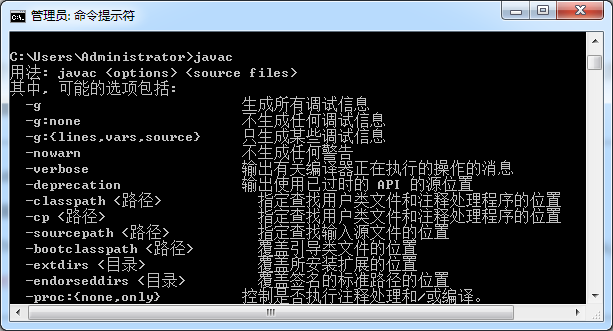
变量值：%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin

## 新建变量名：CLASSPATH

变量值： .;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar

（注意：CLASSPATH变量值前面有个"." 、 在设置变量的末尾时不要加上“;”）

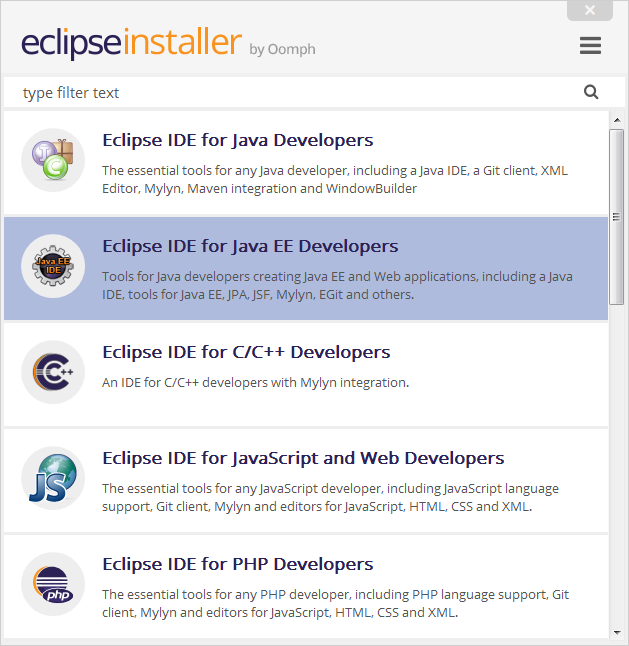
设置完成之后测试是否安装成功；控制台输入命令：java -version、javac  
打印出版本号或者命令提示则说明配置成功。如下图：



# Eclipse 4.6(Neon)的安装

安装提示：

* 要求jdk1.8+
* 在线安装可能需要用vpnFQ



# Tomcat8安装及环境变量配置

将压缩文件解压至自定义路径(我的路径: D:\Program Files\Apache\apache-tomcat-8.5.5)；(安装Tomcat时，在其字母周围不要存在空格，否则可能导致配置不成功)

在系统变量里配置环境变量(不区分大小写)

## 变量名：CATALINA\_BASE

变量值：D:\Program Files\Apache\apache-tomcat-8.5.5

## 变量名：CATALINA\_HOME

变量值：D:\Program Files\Apache\apache-tomcat-8.5.5

## 变量名：classpath

变量值：%CATALINA\_HOME%\lib\servlet-api.jar;

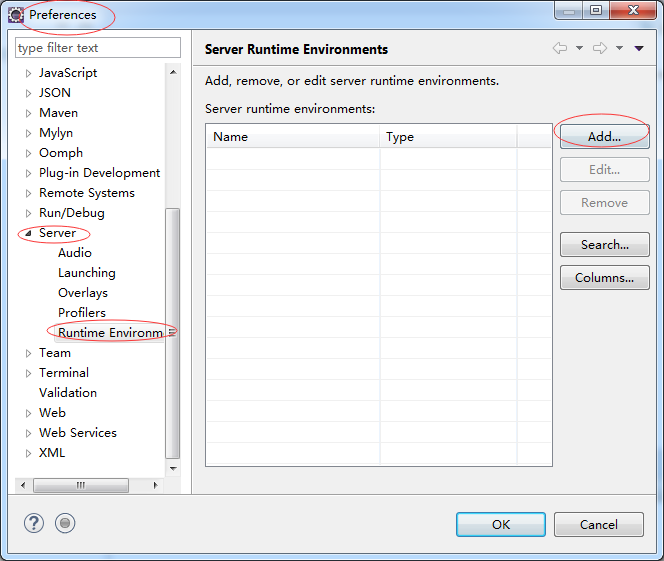
## 变量名：path

变量值：%CATALINA\_HOME%\lib;

# Eclipse配置Tomcat

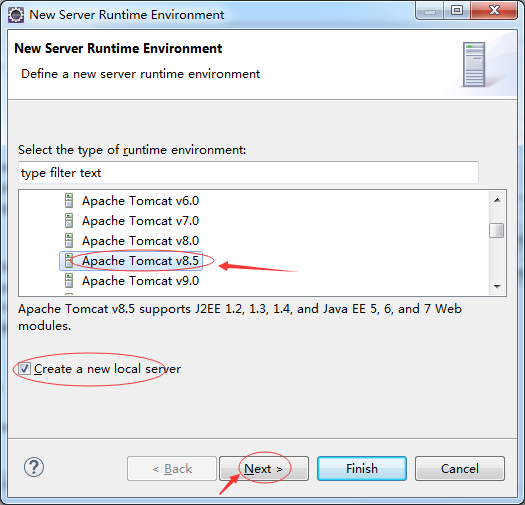
## 配置路径

Window->Preferences->Server->Runtime Environments ->Add

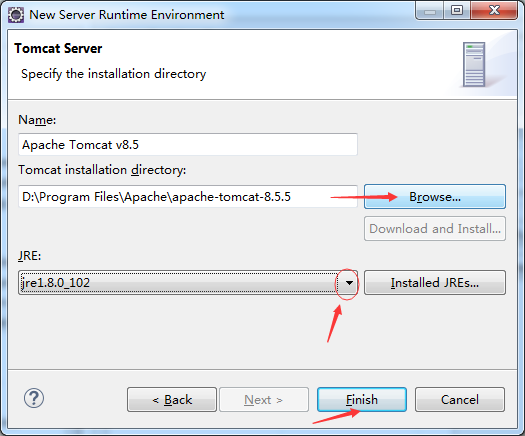


## 选择对应版本

新建一个本机服务；下一步；



## 选择Tomcat路径、JRE版本



# Maven安装配置

## maven配置环境变量

将apache-maven-3.3.9-bin.zip解压到一个路径(我的是：D:\Program Files\Apache\apache-maven-3.3.9)，然后配置环境变量：

新建变量名：MAVEN\_HOME

变量值：D:\Program Files\Apache\apache-maven-3.3.9

编辑变量名：Path

在最前面加上：%MAVEN\_HOME%\bin;

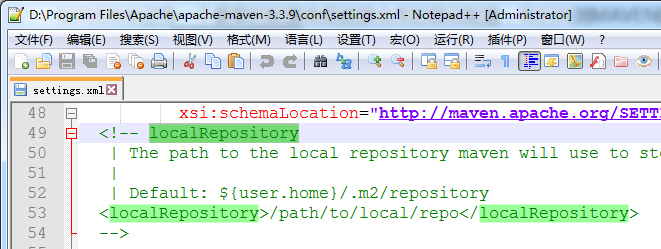
（注意，最后要有个";"作为分隔符）

完成之后，在命令行输入：mvn -version 查看是否有以下内容，如果有表示配置成功。



## Maven数据仓库的配置

### 数据仓库的路径配置

MAVEN中还有一个重要的配置就是数据仓库的路径配置，我们找到MAVEN的安装路径，进入conf-->打开settings.xml，找到localRepository标签，此时是被注释掉的；

### 解除注释自定义路径

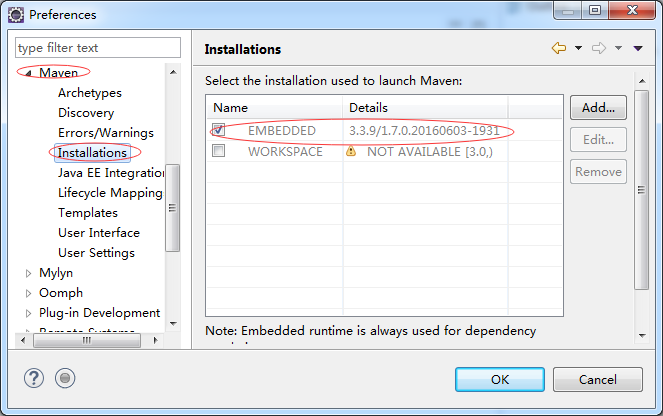
例如我的是：D:\Program Files\jee-neon\maven-server\MavenRepository\maven\_jar，这样以后MAVEN管理下载的jar包都会在这个路径下。当然我们需要建这样一个目录结构，然后还要讲settings.xml复制一份到 D:\Program Files\jee-neon\maven-server\MavenRepository下，这个在与Eclipse集成时会用到。

## Eclipse中集成Maven

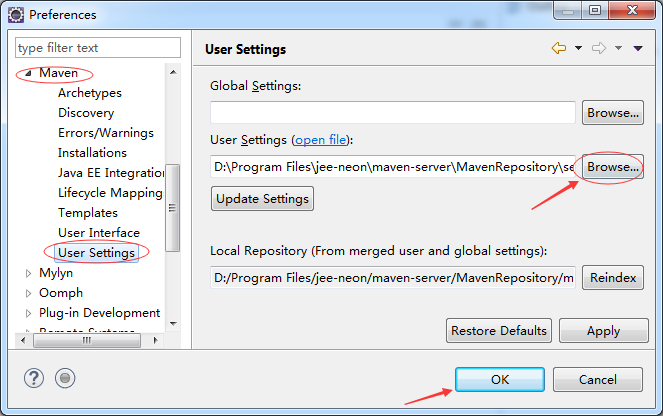
其实这个版本已经集成Maven插件，(配置好maven后)可以新建Maven项目；我们只是需定制自己的路径；

### maven配置完成查看

maven配置完成了就可以看到回自动选择自己配置的版本



### 修改默认maven管理路径

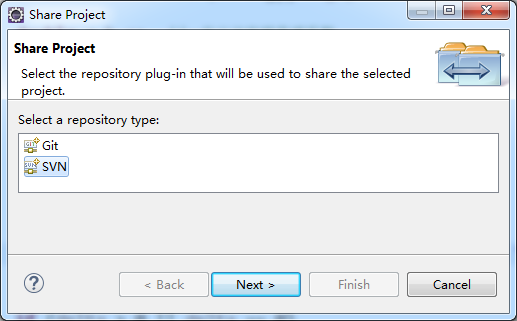


# SVN安装与配置

## SVN安装

SVN是代码版本管理器，首先在本地安装一个SVN管理器，然后配置Eclipse的SVN插件:

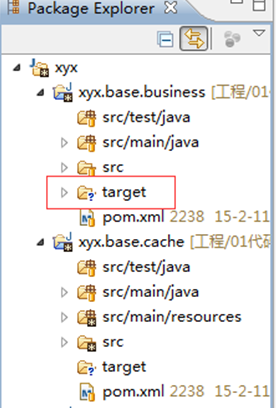
将插件(svn-eclipse-plug-in/site-1.10.13-1.8.x.zip)下载下来解压后，直接复制到Eclipse安装目录下dropins中(重启Eclipse)即可。任意建立一个项目，右键项目-->team-->Share Project..，打开后如果看到了SVN代表插件安装成功,如下图：



## SVM导入Maven工程

### SVN的忽略target

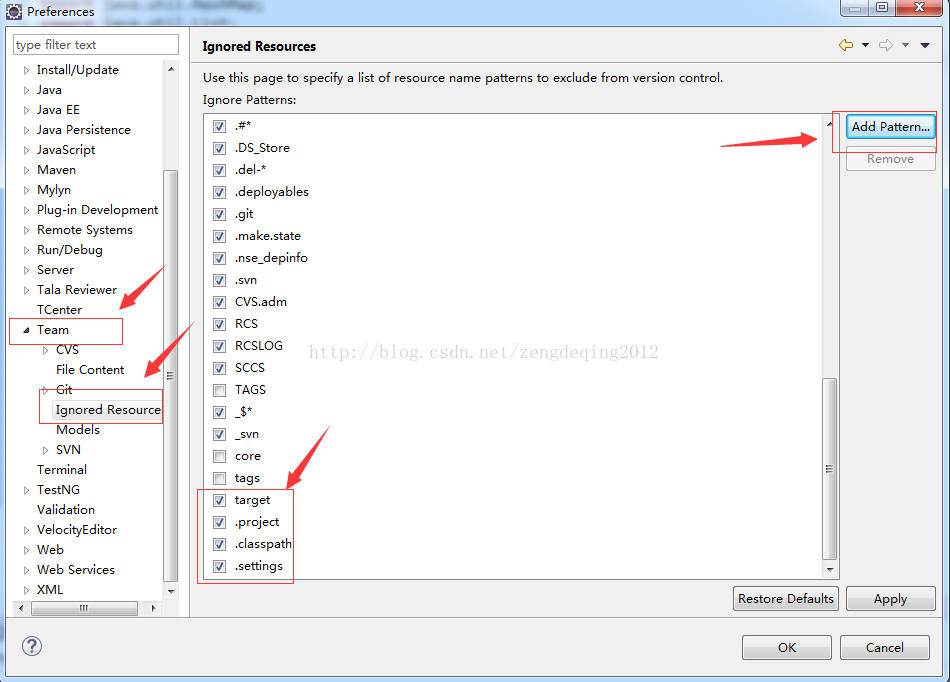
eclipse刚开始导入的项目:



修改方式：

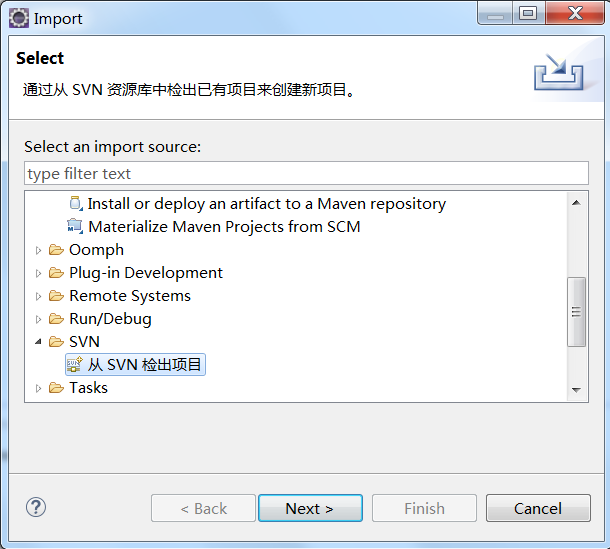
在eclipse中,Windows -> Preferences -> Team -> Ignored Resources里点 “Add Pattern”。

添加：target  \*.settings  .classpath  .project ，然后再"运用"-->"确定",重新编译代码即可



### Eclipse添加SVN资源

第一步：

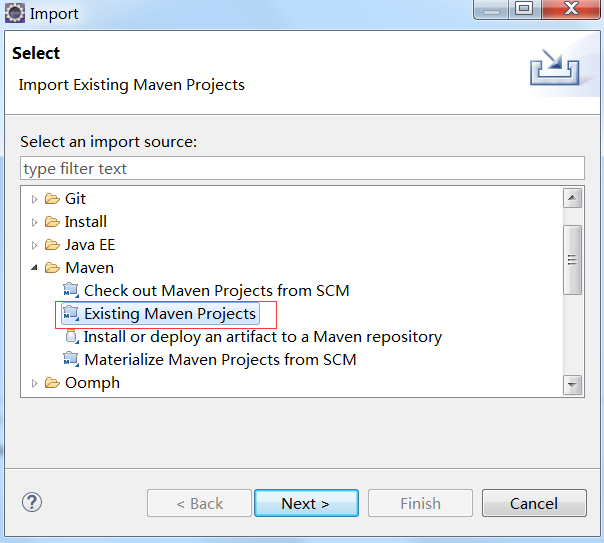


第二步：

删除Eclipse已下载Maven工程（当前不是Maven格式）

第三步：

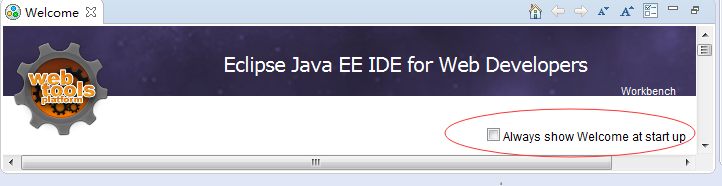
按照Maven工程导入Eclipse中SVN检出资源



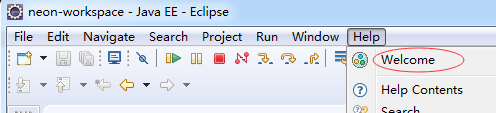
# Eclipse小技巧

## 关闭/开启欢迎界面

关闭默认的欢迎界面：将欢迎界面右下角的"Always show Welcome at start up"选项取消(不选)，以后就不会默认显示。

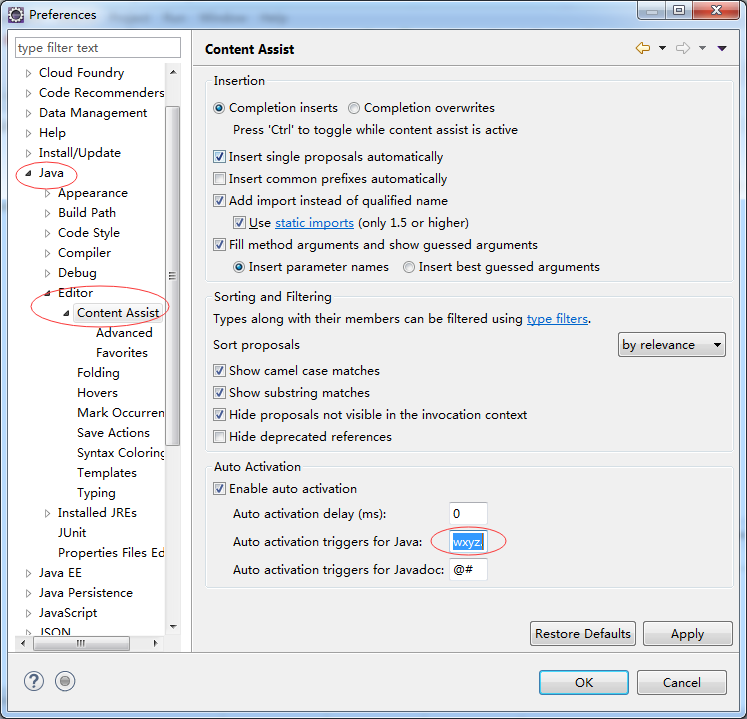


开启欢迎界面：菜单栏 -> Help -> Welcome。



## Java代码提示

复制：abcdefghijklmnopqrstuvwxyz. 到如图文本框，"."代码提示换成任意字母加"."出现时的代码提示

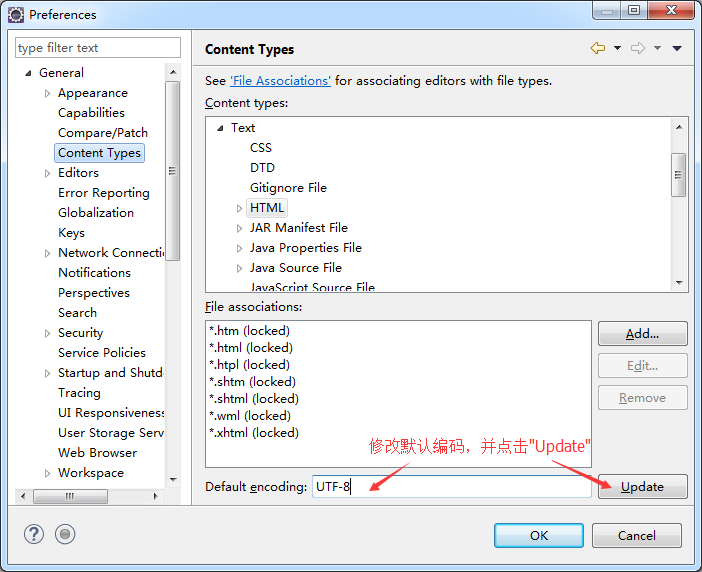


## 设置默认编码

设置workspace的编码格式为UTF-8  
Window->Preferences->General->Workspace 面板Text file encoding ->Other 选择UTF-8

设置JSP文件的编码格式为UTF-8  
Window->Preferences->Web->JSP Files 面板选择 ISO 10646/Unicode(UTF-8)

设置其他文件的编码格式：如下图  
Window->Preferences->General ->Content Type。



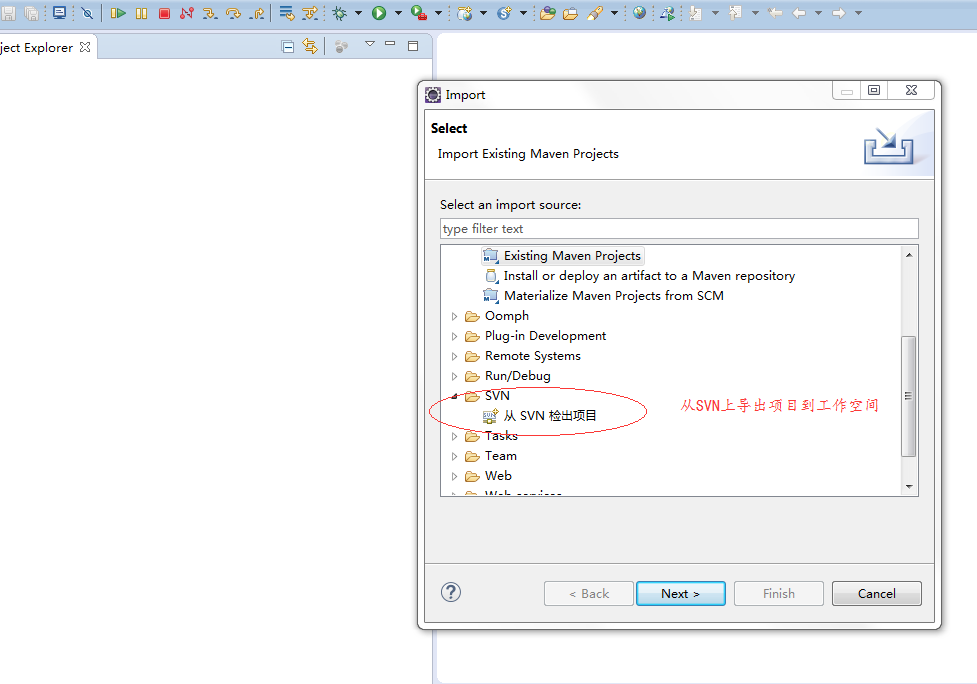
## 快捷键

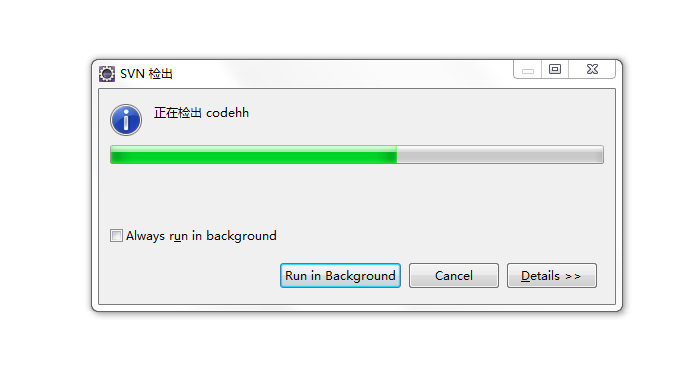
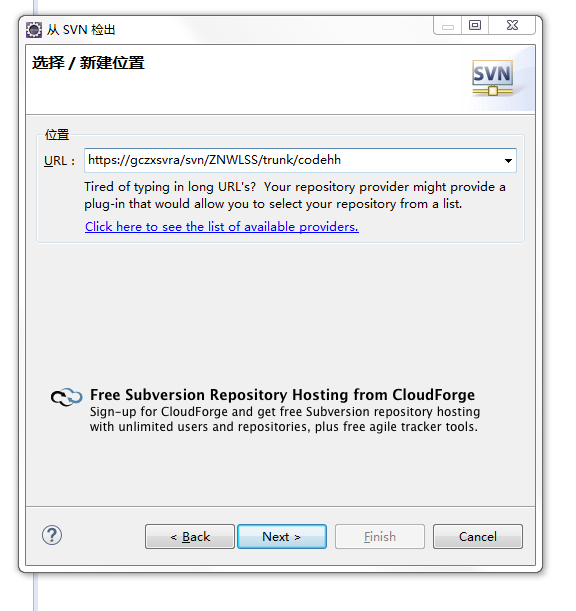
* 常用快捷键：  
  Ctrl+F11 运行  
  Ctrl+Shift+/ 在代码窗口中加入/\* ... */注释  
  Ctrl+Shift+\ 消除/* ... \*/注释  
  Ctrl+Shift+O 将缺少的import语句加入Ctrl+1 快速修复(最经典的快捷键,就不用多说了)  
  Ctrl+L 定位在某行 (对于程序超过100的人就有福音了)  
  Ctrl+M 最大化当前的Edit或View (再按则反之)  
  Ctrl+/ 注释当前行,再按则取消注释  
  Ctrl+Shift+F 格式化当前代码  
  F3 跳到声明或定义的地方  
  F4 显示类图  
  Ctrl + Alt + ↓(↑) : 向下(上)复制本行
* 断点调试中的快捷键：  
  F5 Step Into 单步进入  
  F6 Step Over 单步执行跳过  
  F7 Step Return 单步跳出

# 开发手册

## 代码导入

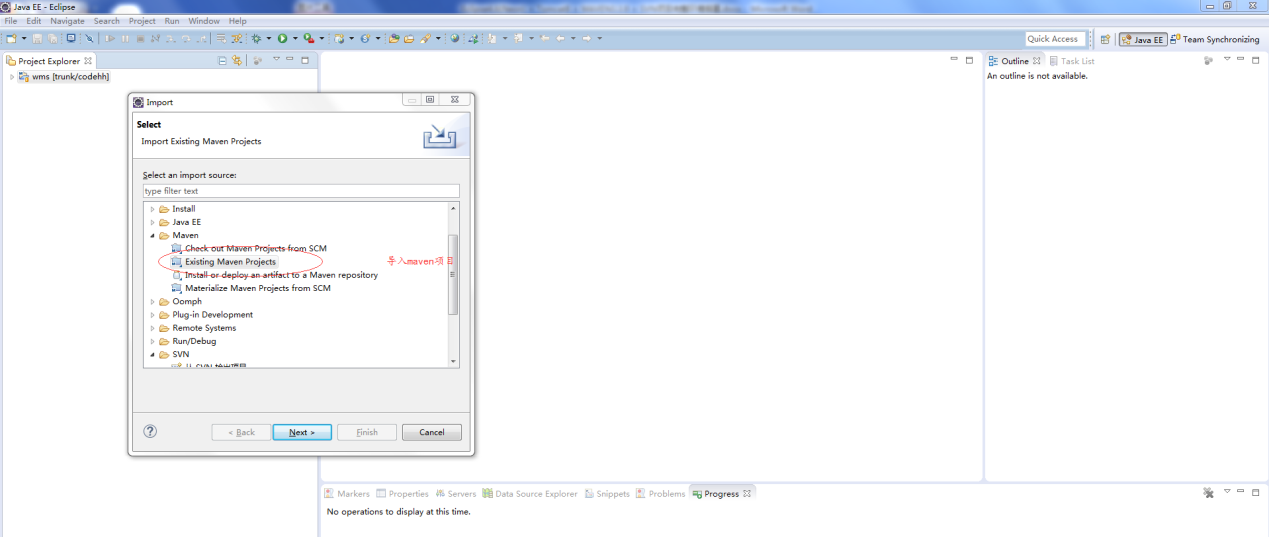
从SVN上导出项目：地址https://gczxsvra/svn/ZNWLSS/trunk/codehh

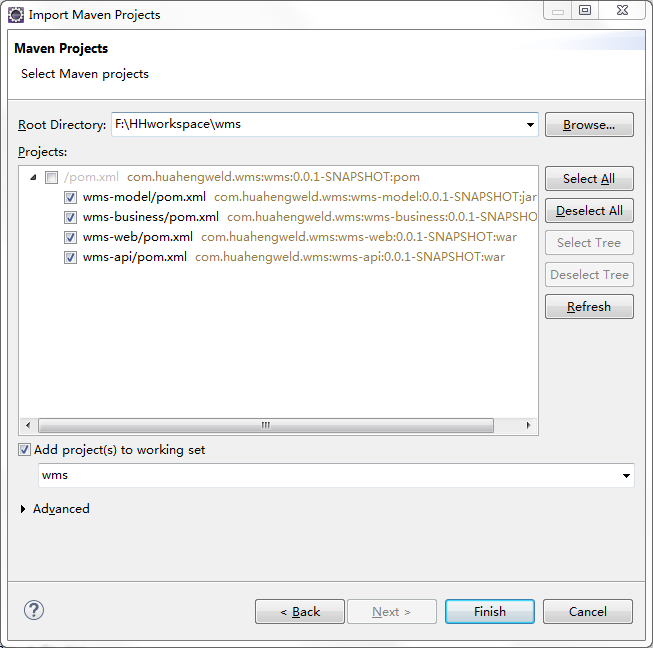


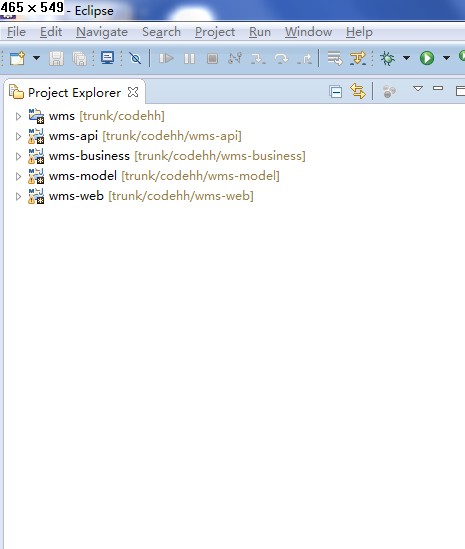


## 导入Maven工程

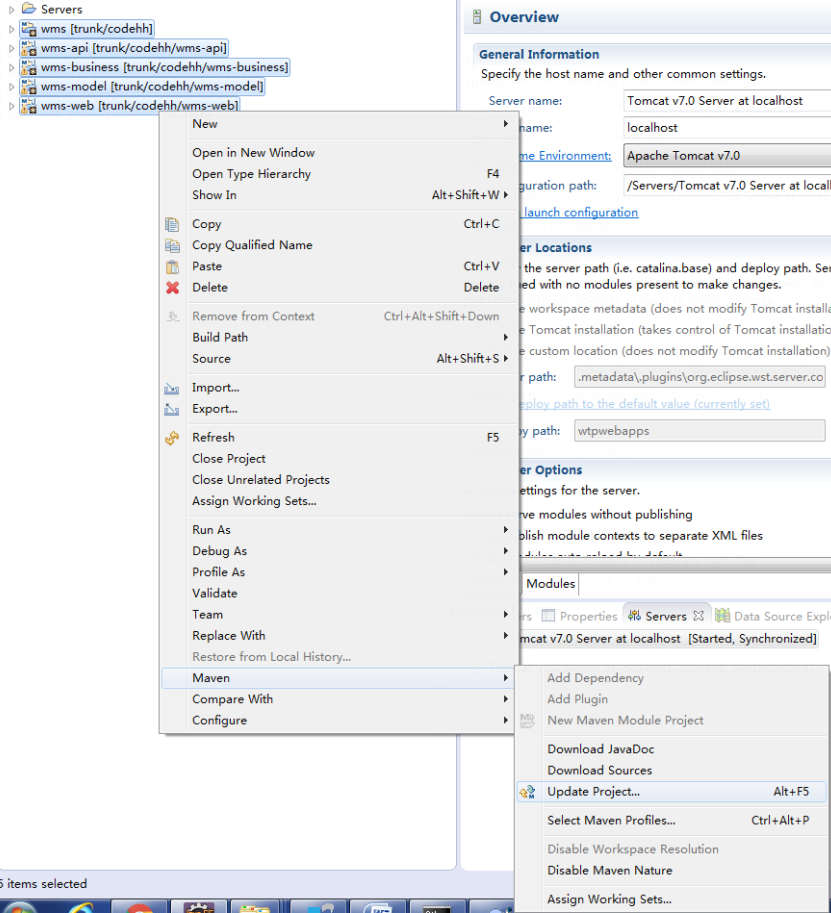
选中导入的项目，右键import选择maven导入。导入完毕后全选所有项目，右键→Maven→Update project→选择Offline→确定，更新Maven工程所需的jar包

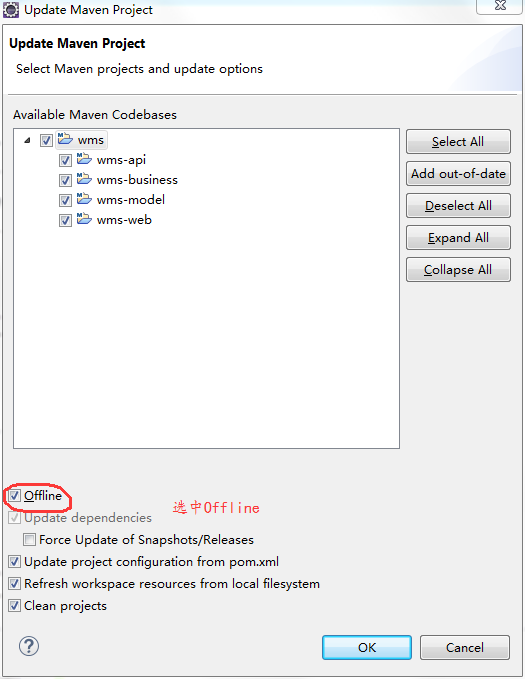






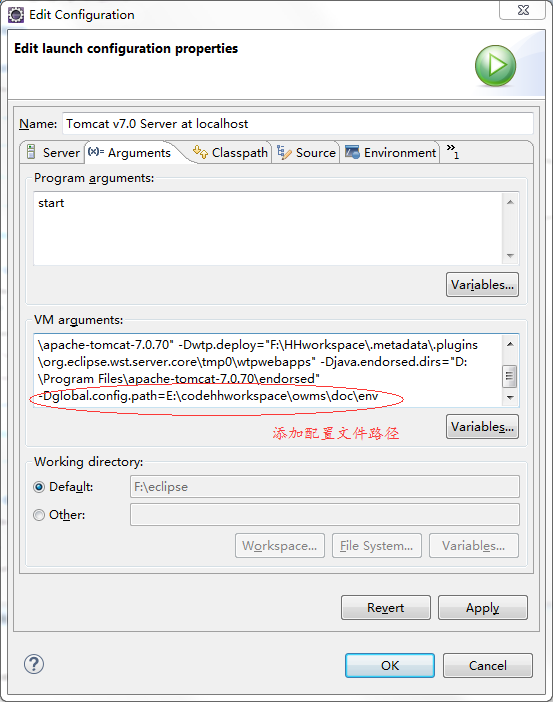
## 更新Maven库





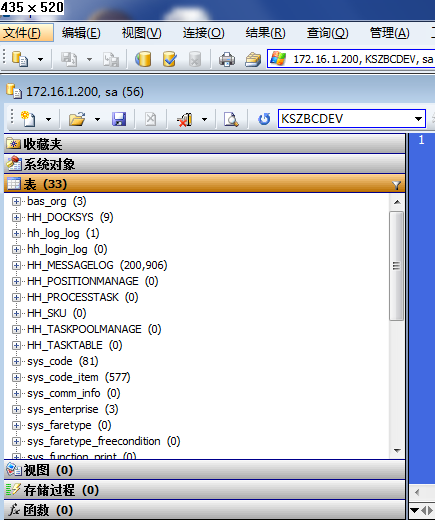
## Tomcat配置文件路径

双击已添加的Tomcat v7.0 Servier at localhost，点击Open launch configuration，在Arguments标签下最后一行添加-Dglobal.config.path=E:\codehhworkspace\wms\doc\env（路径改为当前项目下doc文件夹下的env文件夹路径）



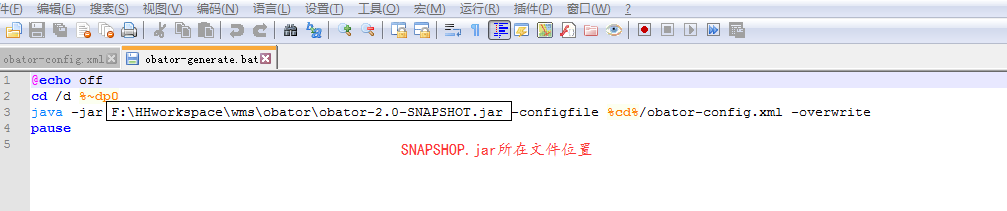
## 生成DAO、PO、xml文件

连接数据库，找到需要自动生成文件的表；复制表名；

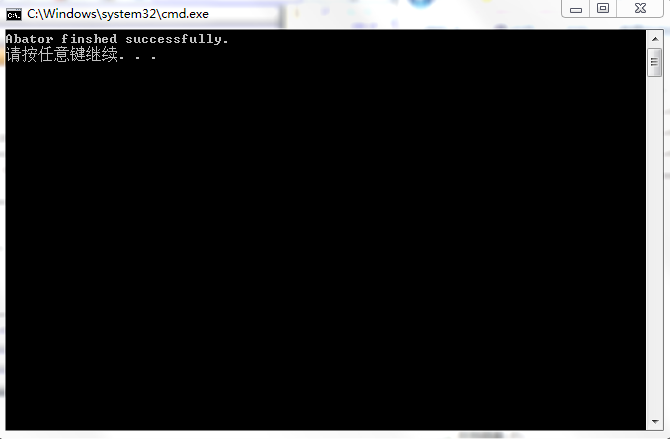


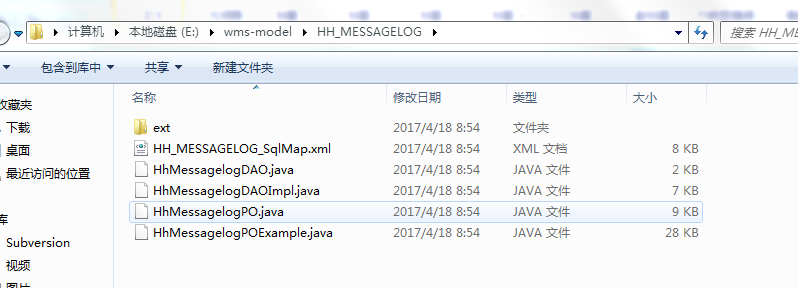
在当前项目下obator文件夹中，编辑obator-config.xml文件，修改classPathEntry标签下的location属性，把路径改为本机sqljdbc4-4.0.jar文件（location="C:\Users\0290\.m2\repository\com\microsoft\sqlserver\sqljdbc4\4.0\sqljdbc4-4.0.jar"）的路径；修改table标签下tableName属性，把表名改为需要生成文件的表名（HH\_MESSAGELOG）；修改targetProject属性，为生成文件的文件夹路径；修改generatedKey标签下的column属性，把表结构主键改为关联的主键名称

编辑obator-generate.bat，把路径改为obator-2.0-SNAPSHOT.jar所在的路径（示例：E:\DBCode\1.0\obator\obator-2.0-SNAPSHOT.jar）

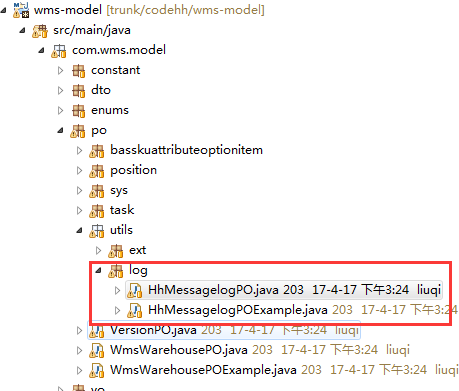


运行obator-generate.bat即可生成DAO层、DAO层的实现类、PO层、PO层的实现类和对应的sqlMap\_xml文件。

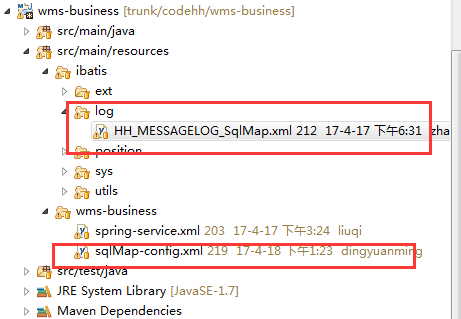




把PO文件放到wms-model工程下的src/main/java/com.wms.model.po下;



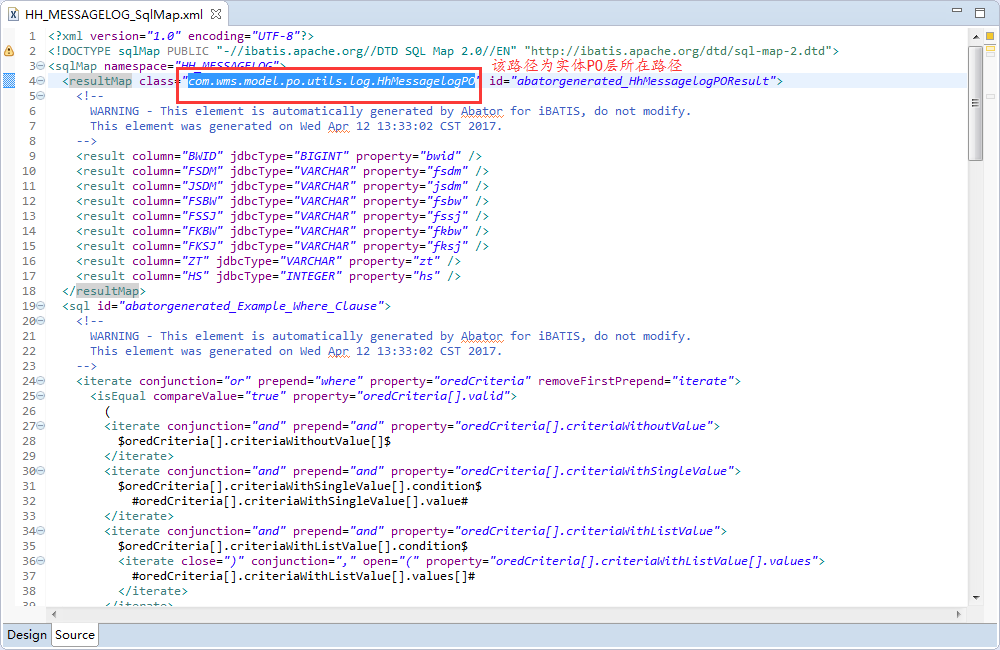
把sqlMap.xml放到wms-business工程下的src/main/resources/ibatis下，配置SqlMap.xml的文件路径；



打开sqlMap-config.xml,修改sqlMap的文件路径；

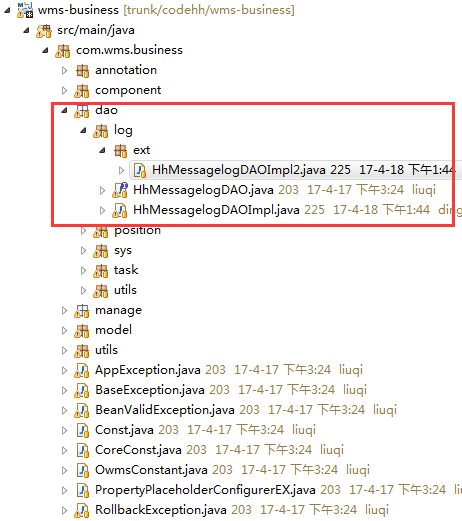


打开HH\_MESSAGELOG\_SqlMap.xml,修改文件中的实体类路径；所有涉及PO层都需要更改文件路径，批量修改。



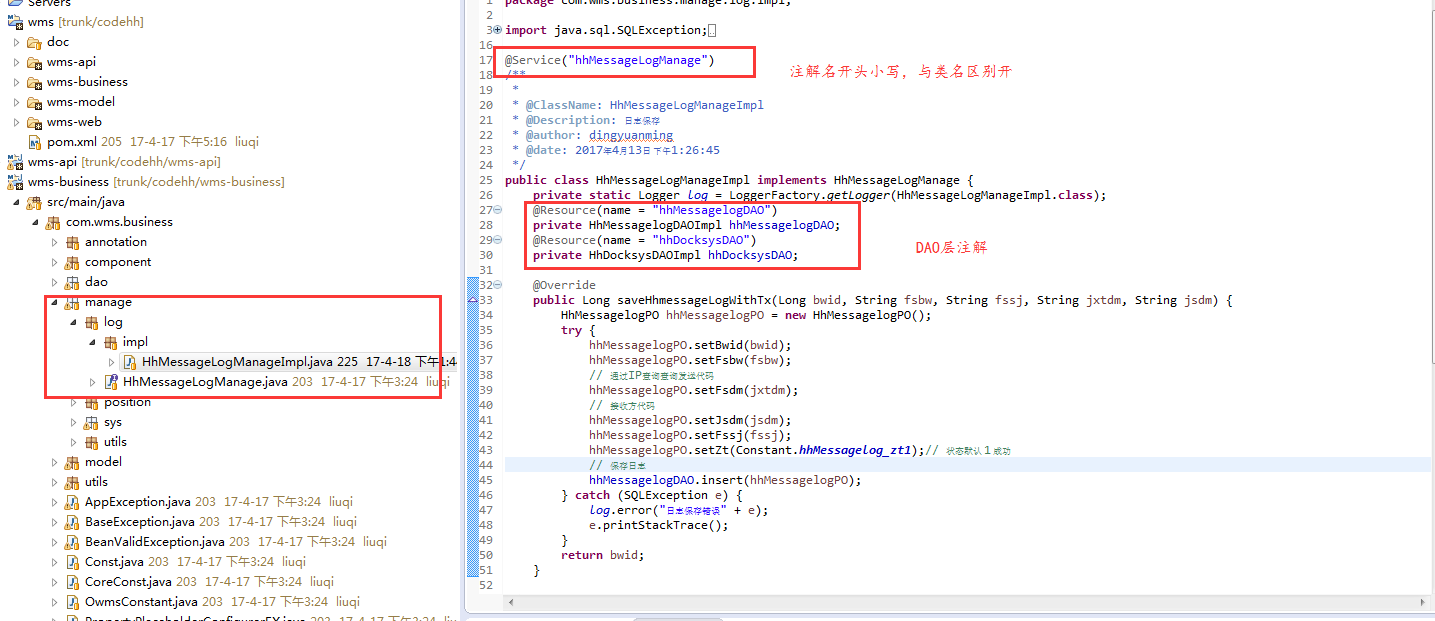
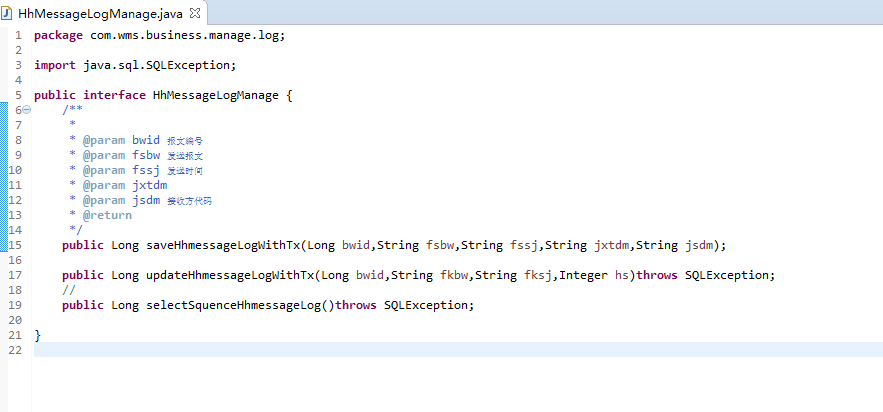
把DAO.java和DAOImpl.java放到wms-business工程下的src/main/java/com.wms.business.dao；

扩展文件DAOImpl2.java文件放到wms-business工程下的src/main/java/com.wms.business.dao.ext下。



## 编写Manage业务层

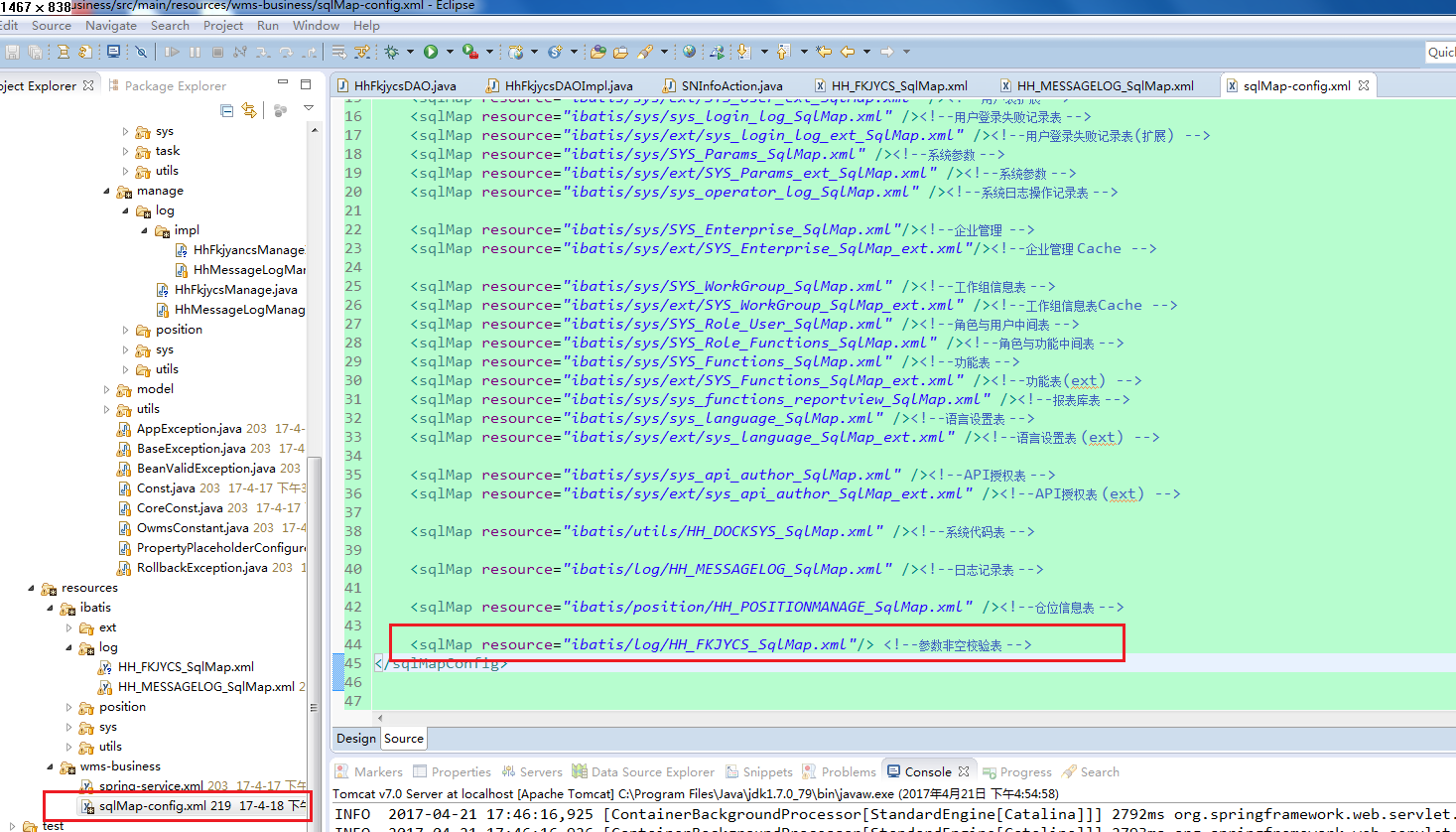
新建目录结构com.wms.business.manage.log，新建接口层及实现层；



## 常用方法案例：

1.配置文件设置

（1）全局配置文件:SqlMap-config.xml

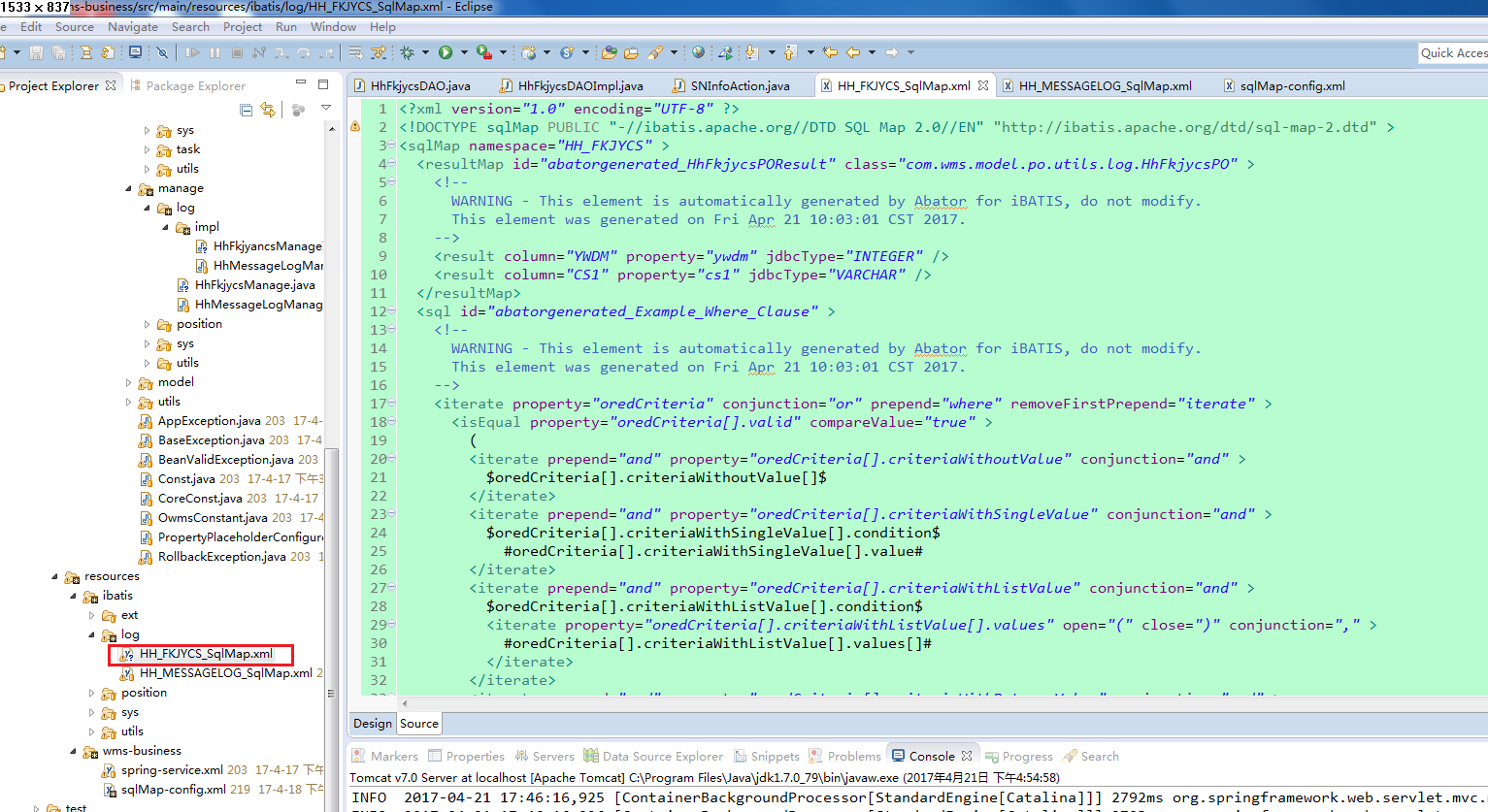


在全局配置文件SqlMap-config.xml中添加字段

<sqlMap resource="ibatis/log/HH\_FKJYCS\_SqlMap.xml"/><!--参数非空校验表 -->

即要关联的参数非空校验表HH\_FKJYCS

（2）映射关系配置文件：HH\_FKJYCS\_SqlMap.xml



备注：xml文件里parameterClass是传入参数类型resultMap为对象类型为返回结果类型



3.对数据库CURD详解

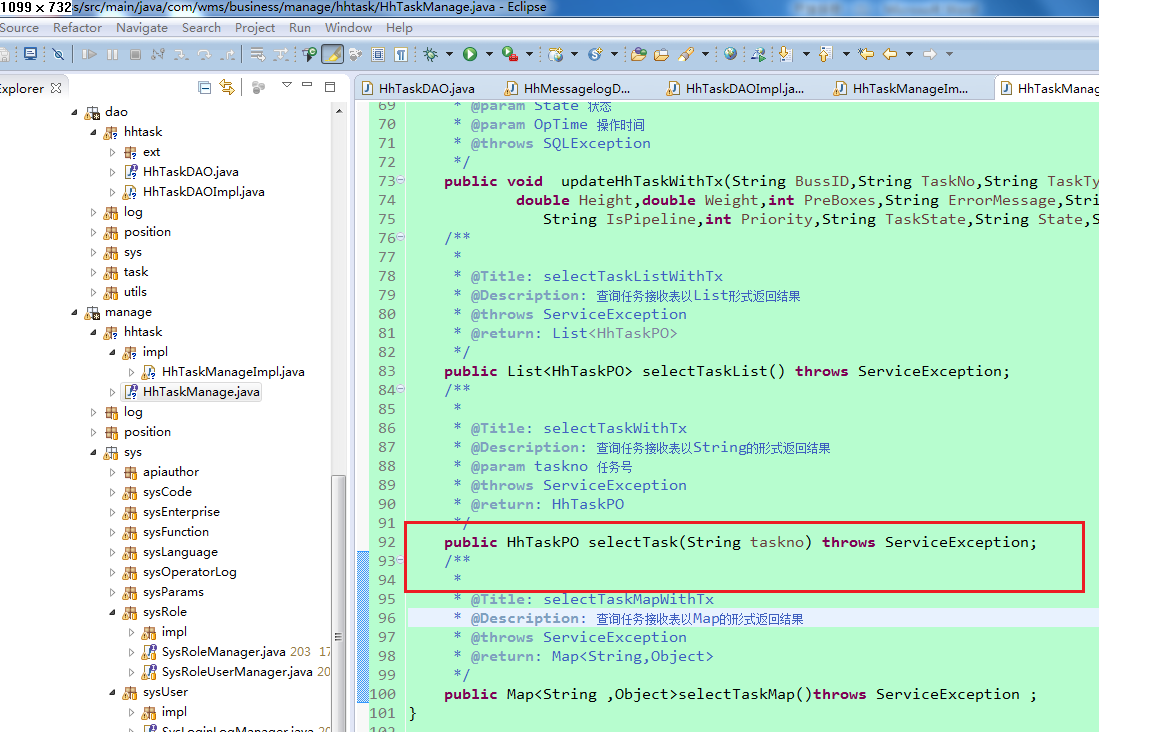
（1）insert向数据库中插入数据

（2）delete删除数据库中数据

（3）select查询数据库，以三种方式返回数据(Object、List、Map)

### \*以Object的形式返回

HhTaskManage.java



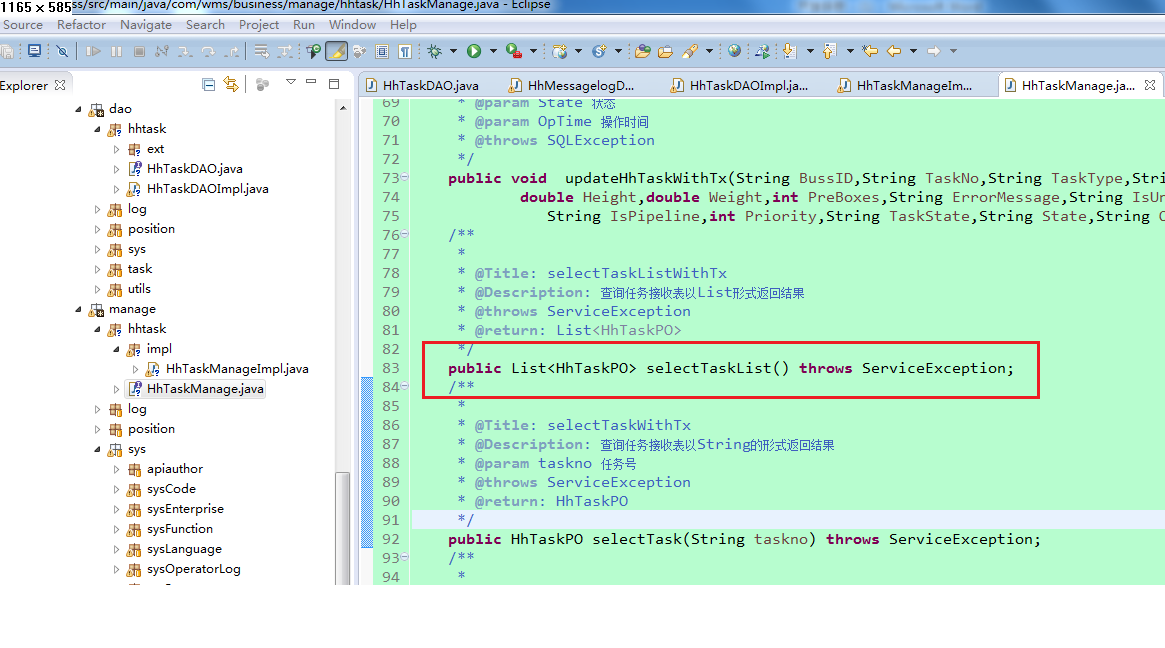
HhTaskManageImpl.java



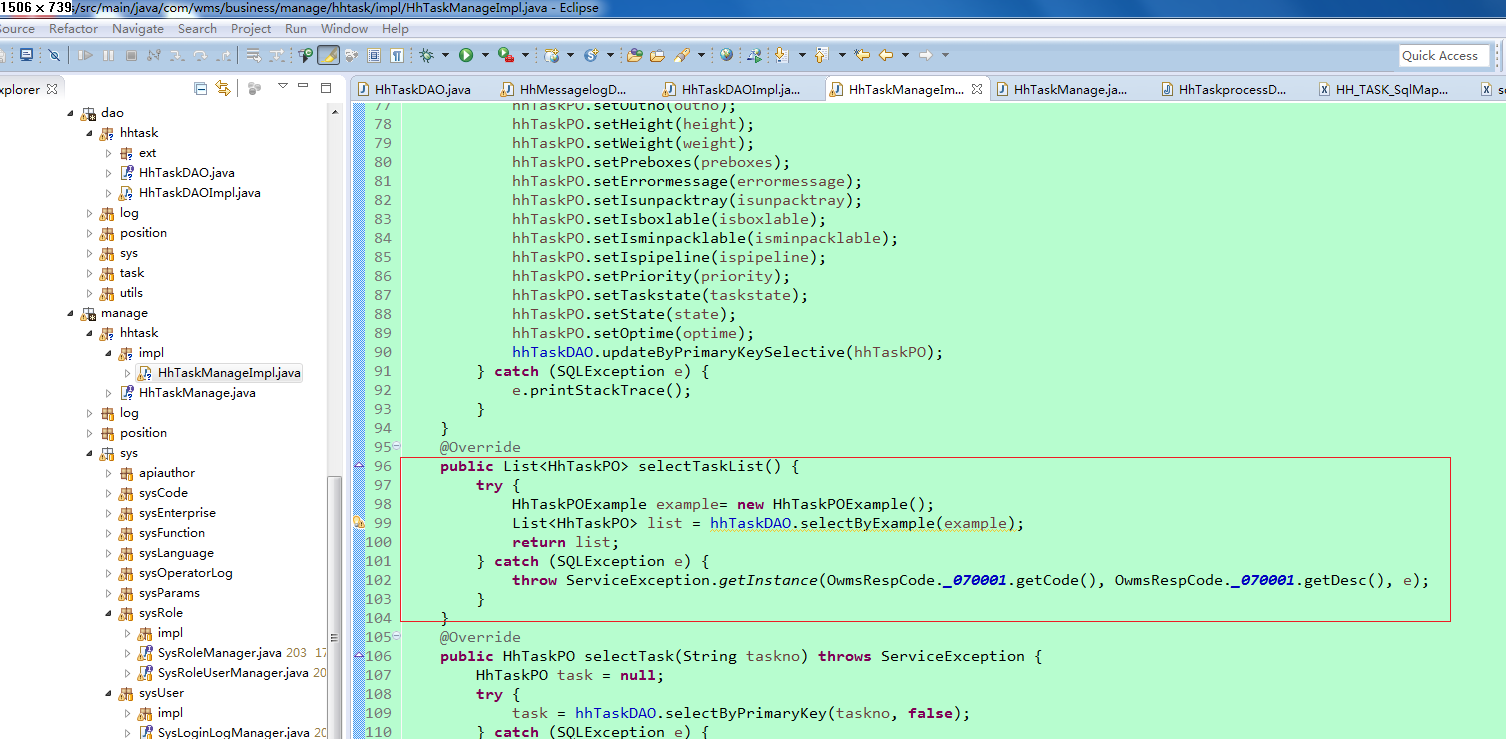
### \*以List的形式返回

查询List:条件传进去的参数必须是对象;

HhTaskManage.java

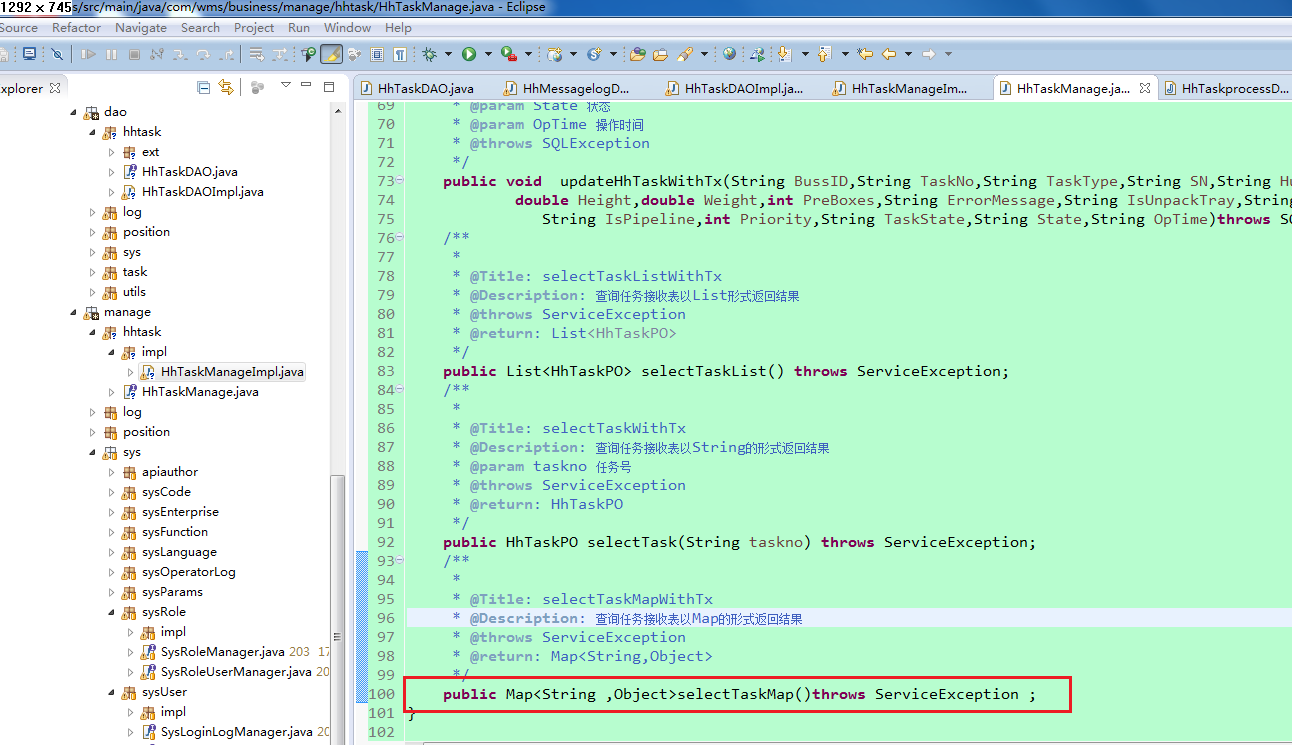


HhTaskManageImpl.java

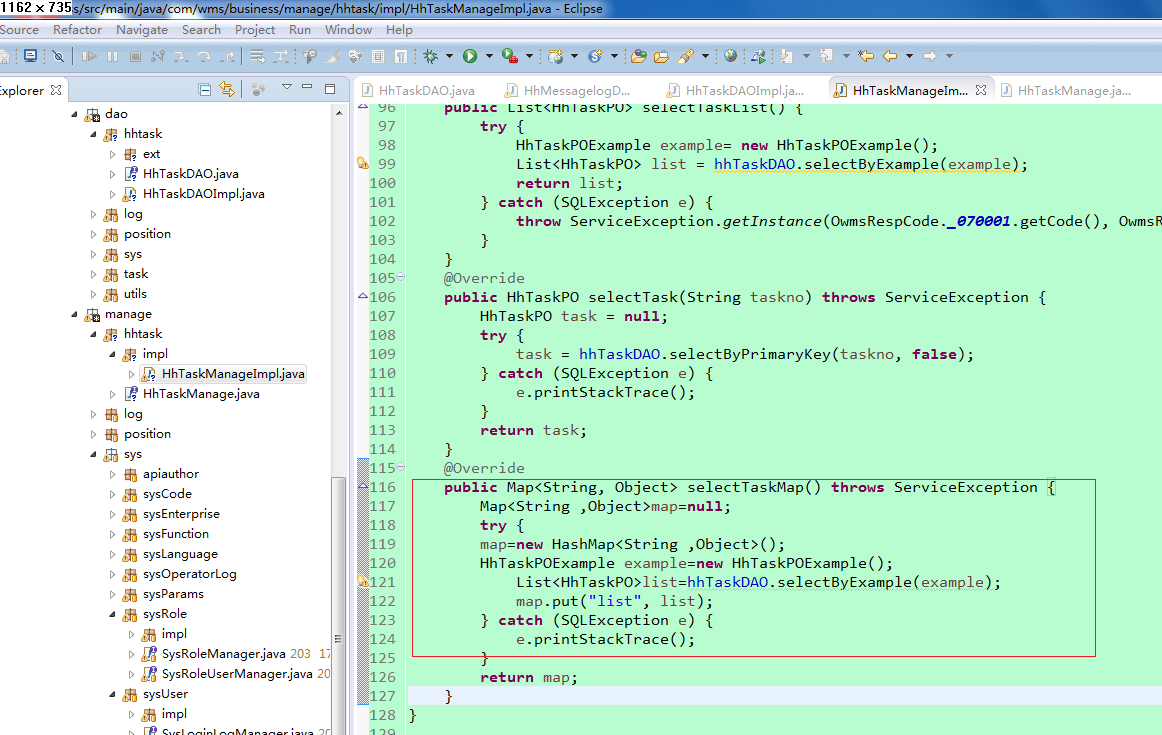


### \*以Map的形式返回

HhTaskManage.java



HhTaskManageImpl.java



resultMap为返回结果类型resultMap

sql语句为select YWDM, CS1 from HH\_FKJYCS表名需查出的为多个对象

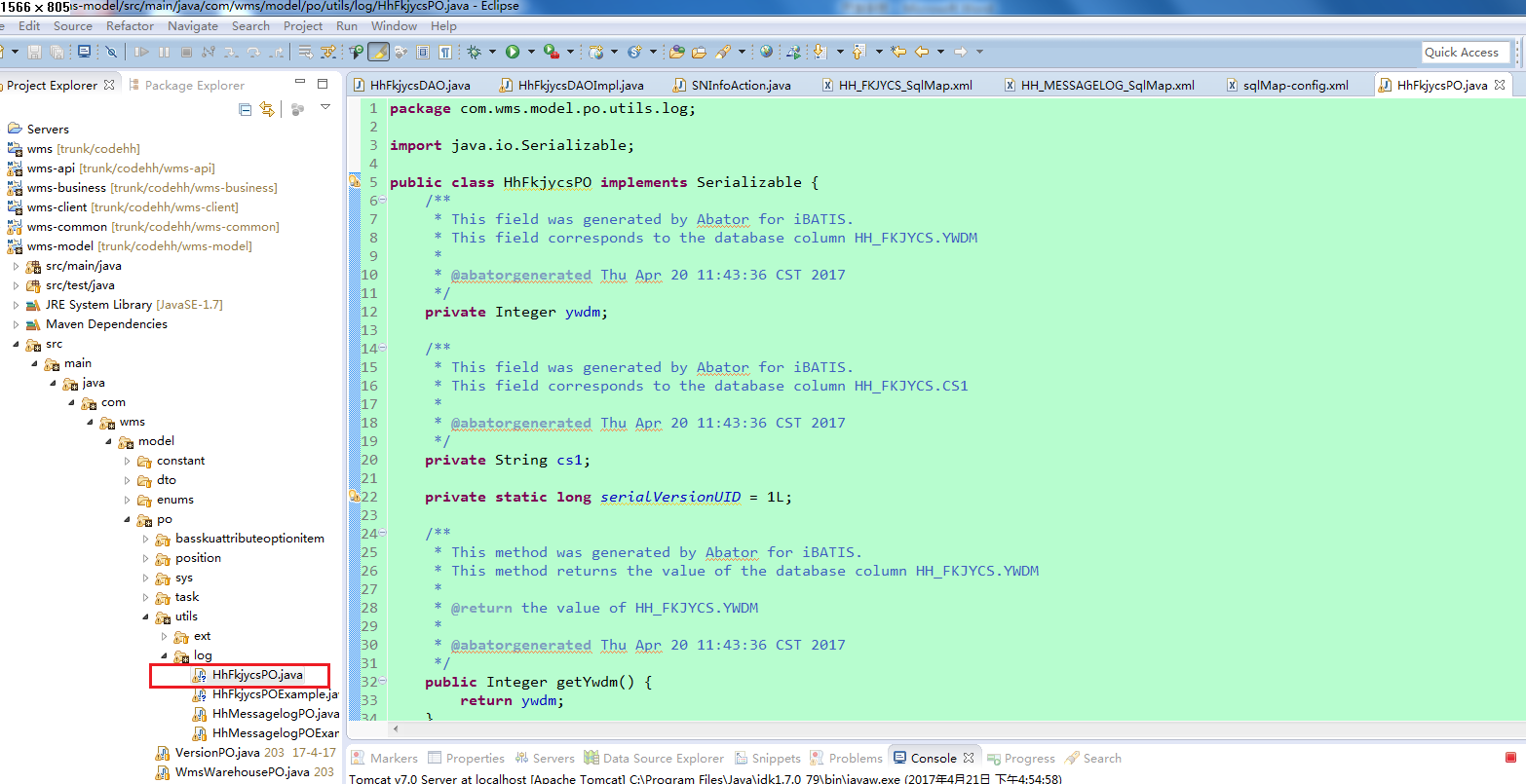
service层：HhTaskManage.java

HhTaskManageImpl.java

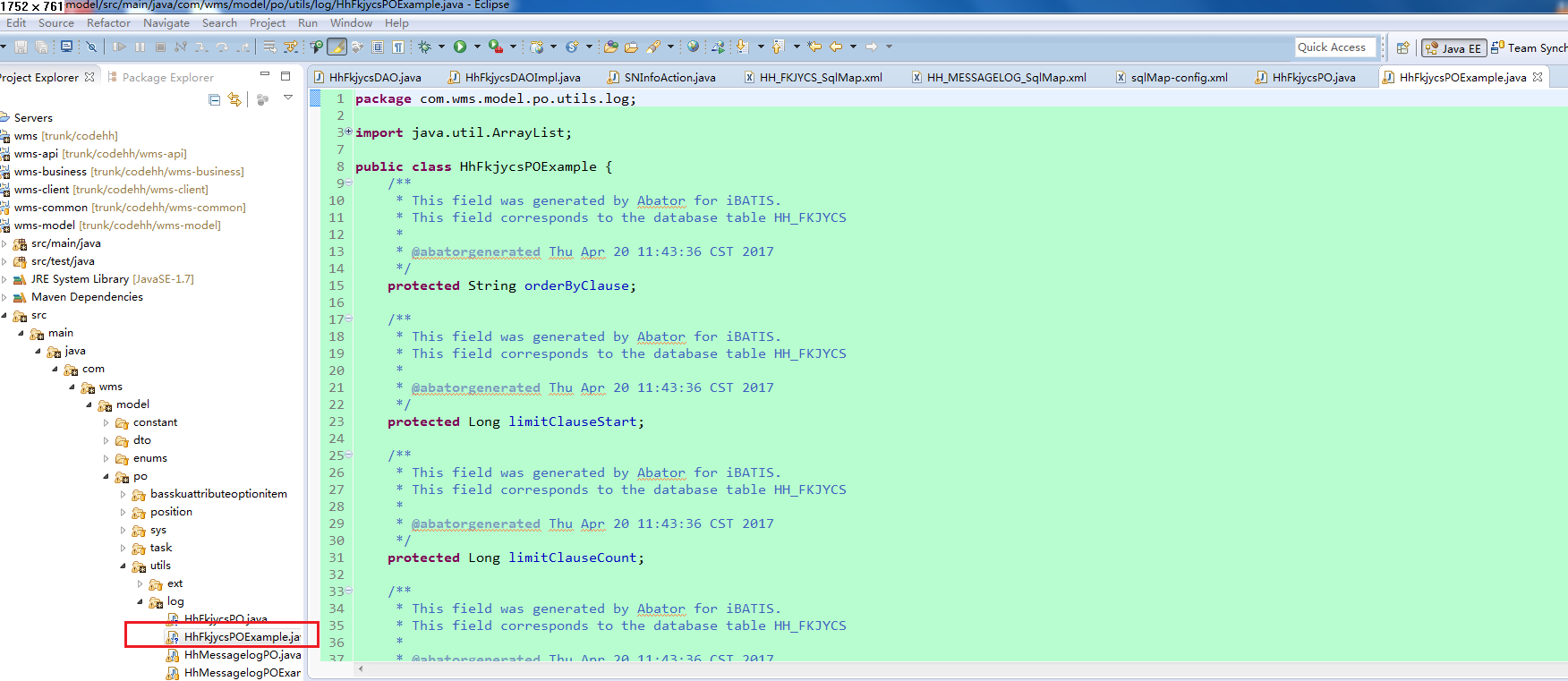


PO层：

HhFkjycsPO

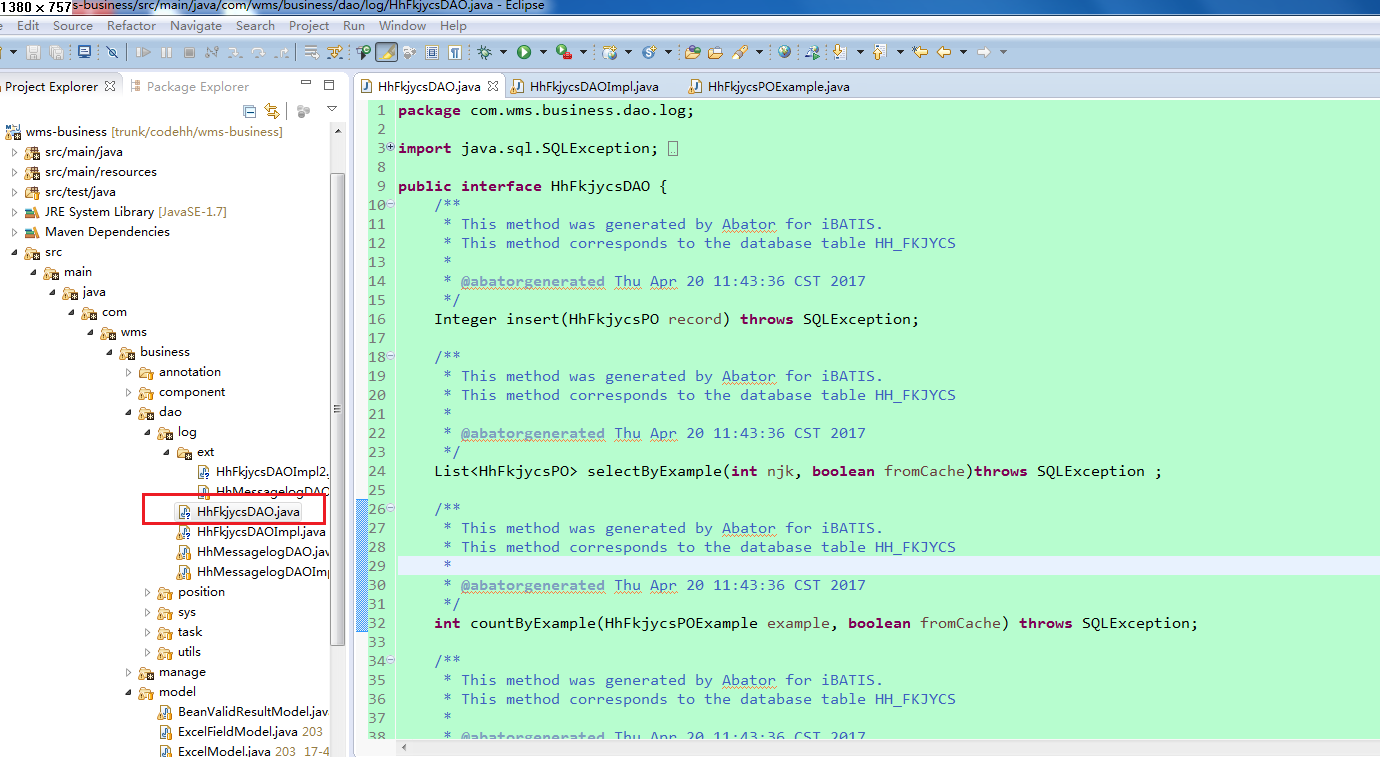


HhFkjycsPOExample



Dao层：

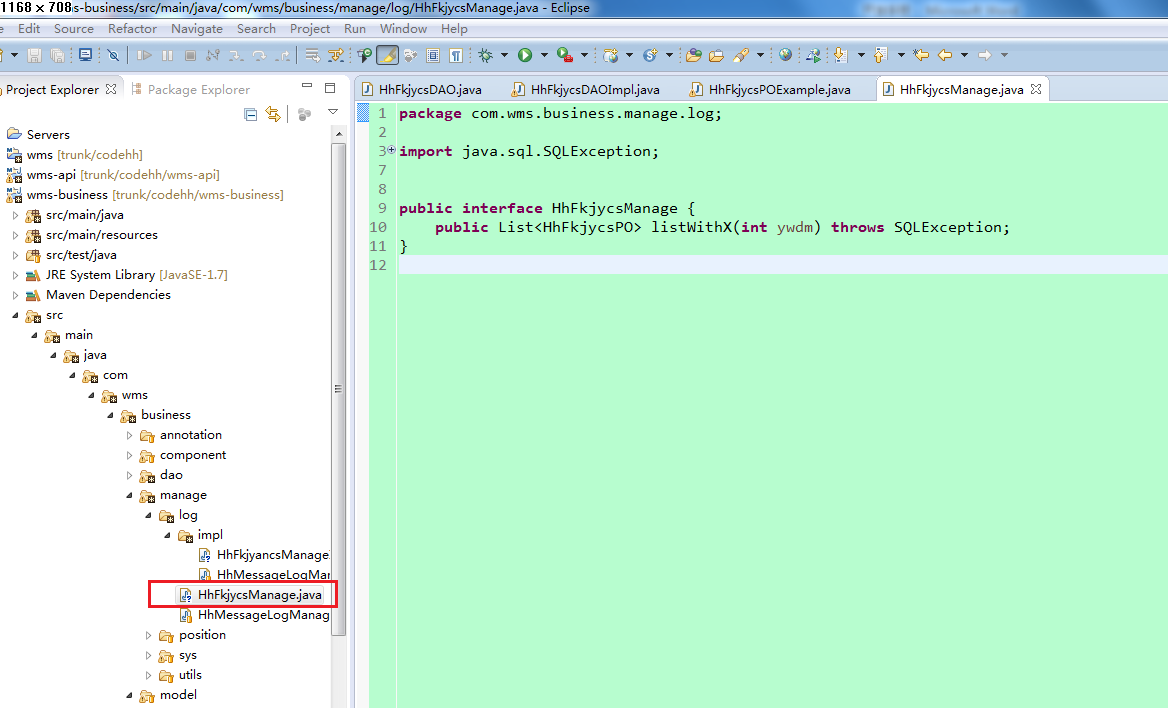
HhFkjycsDAO



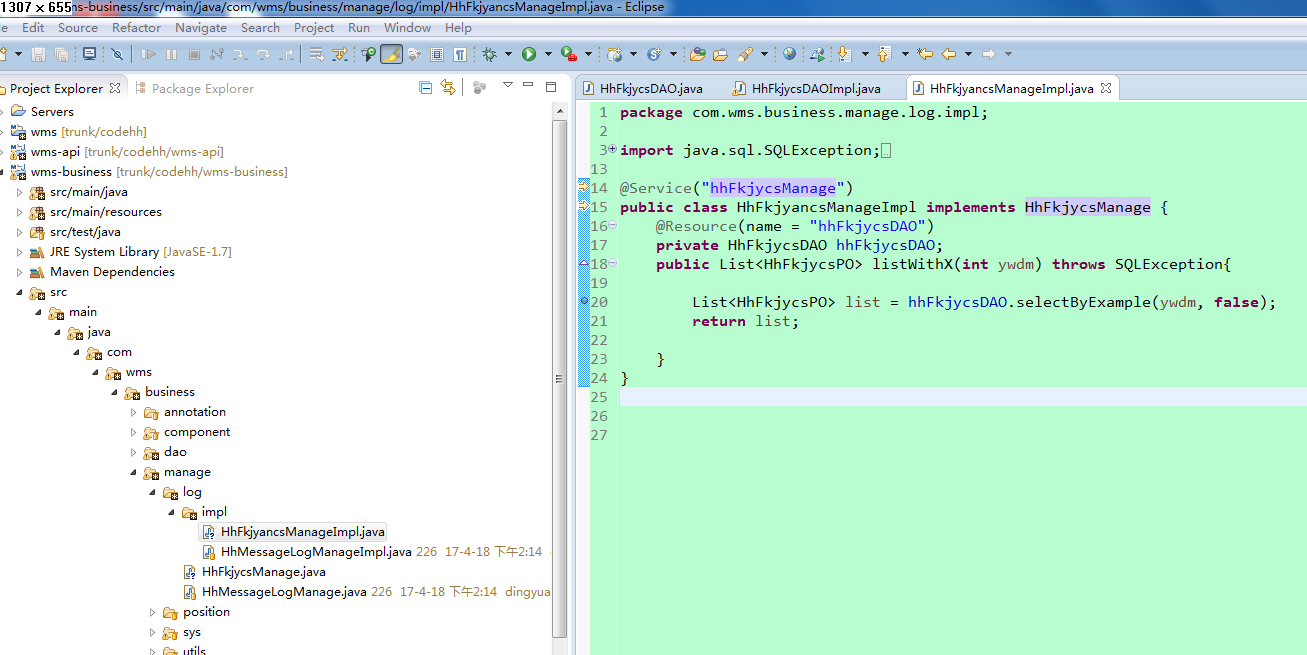
HhFkjycsDAOImpl



Service层:

HhFkjycsManage层：

HhFkjyancsManageImpl



1用开发工具自动生成po层,dao层,配置文件(表名+xml);

2在dao层写实现方法:

3List:条件传进去的参数必须是对象;xml文件里parameterClass是传入参数类型

为对象类型

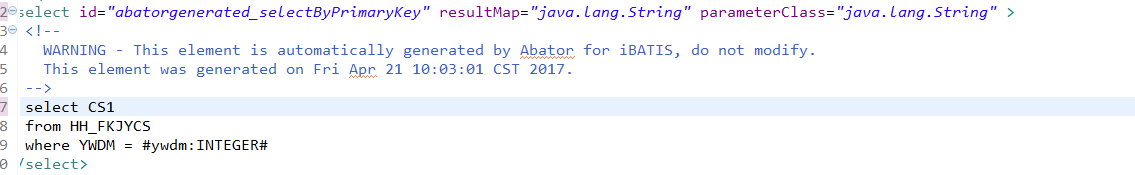
4resultMap为返回参数类型

5sql语句为select \* from 表名需查出的为多个对象

修改update 添加insert 需要提交数据不许在方法名后+WithTx 查询可以不加

1 单个对象的查询:

2 在dao曾写业务需要的是查出一个值在配置文件中parameterClass是传入参数类型resultMap为输出参数的类型.



3如果为对象类型会报错

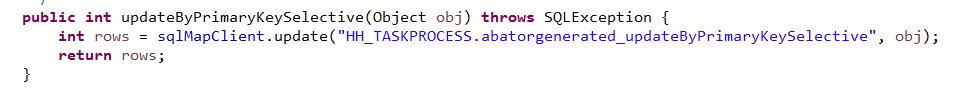
### update修改方法

1要实现修改方法,在传参事要为对象,在对象中封装要传入的参数,主键是必须要有的,实现方法如下

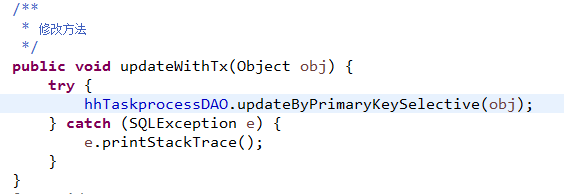


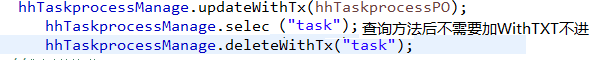
* dao层

将参数类型定义为Object,使方法通用,在dao层,manage层里面只写方法不封装参数

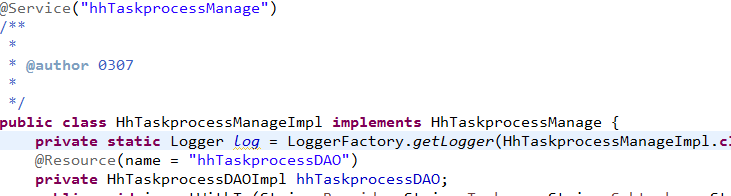


* manage层



* **api**
* 
* spring-service.xml中配置了事务提交方法名后面必须加WithTx，查询不需要。
* 
* api 和manage层注释

C:\Users\0307\Desktop\Feige图片20170426114543.png

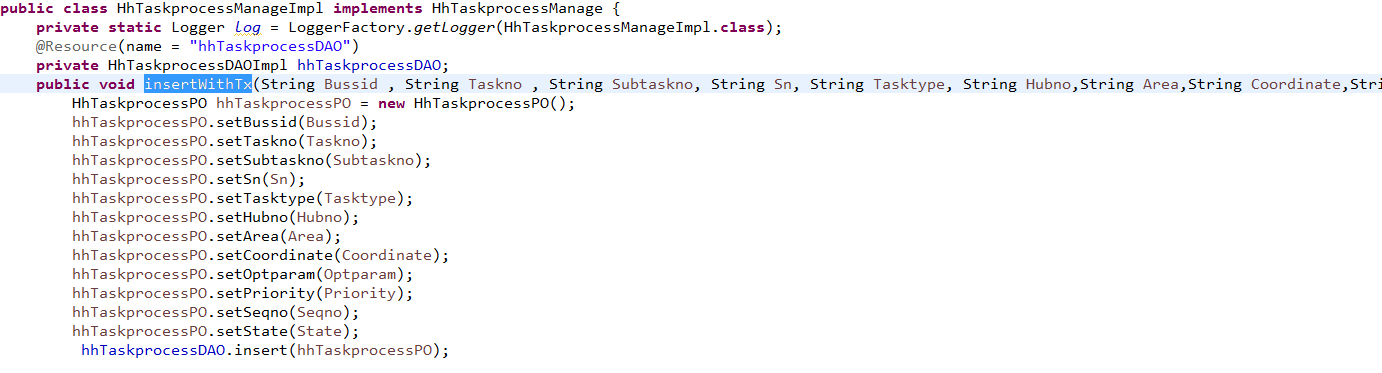


### insert方法

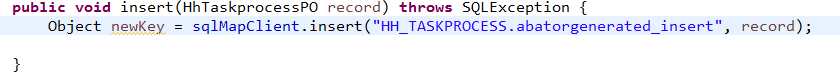
api: JSON解析前台传过来的的参数传入参数

C:\Users\0307\Desktop\Feige图片20170426115033.png

* manage层对参数进行封装

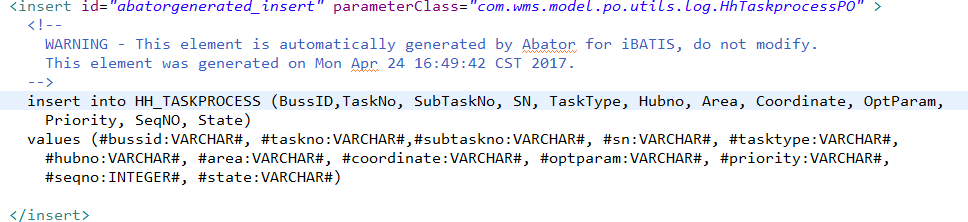


* dao层事务提交



XML配置文件

由于传入的是一个对象parameterClass为对象类型



### 查单个对象方法

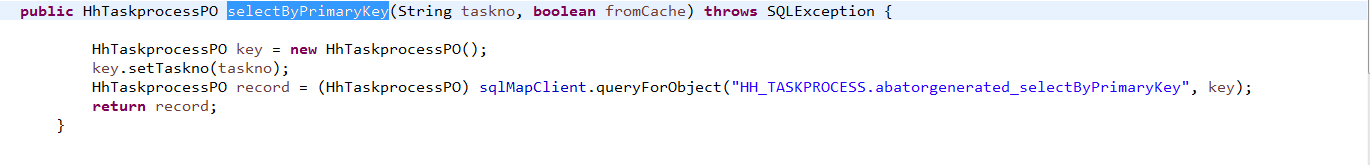
api: 根据主键进行查询

C:\Users\0307\Desktop\Feige图片20170426130450.png

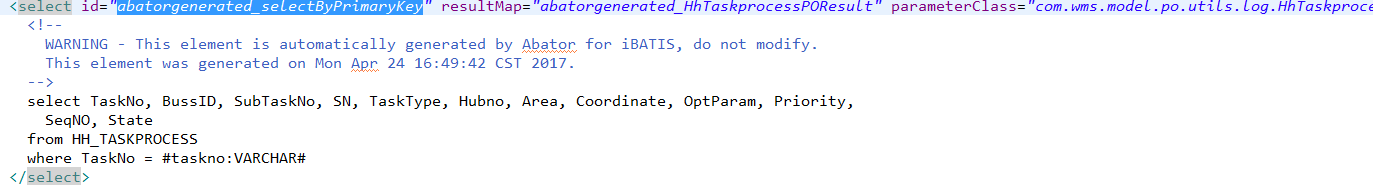
Manage调用dao实现方法传入参数



dao：根据XML命名空间和ID操作数据库



XML



## 批处理实现：

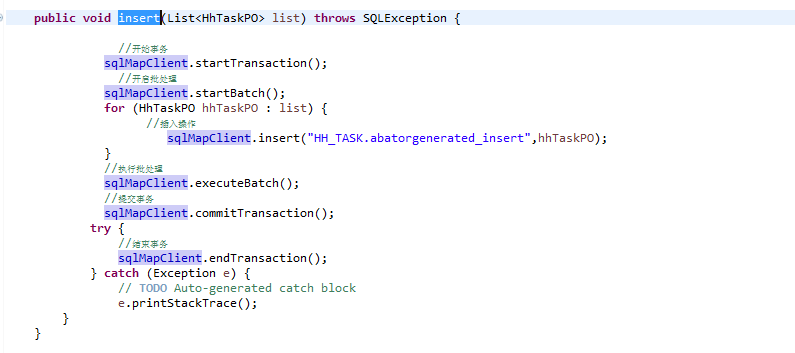
实现方式：（具体实现见 /wms-api/src/main/java/com/wms/api/wmsapi/LoadTasksApi.java、/wms-business/src/main/java/com/wms/business/dao/hhtask/HhTaskDAOImpl.java等文件）

1  非空等校验通过后，将调用者传的json数据转成json数组后，在循环遍历中将其包装成相应对象后添加到List中，如 List<HhTaskPO> list；代码截图如下：

[](http://172.16.1.201/pro/data/upload/1/201705/08153936059o58.png)

2  在DAO层实现层，新增参数类型为List<HhTaskPO>的 insert方法；实现原理可简要概括为：通过foreach遍历list对象，遍历完成后，调用框架原有的批处理方法统一进行提交，具体可见注释

代码截图如下：

[](http://172.16.1.201/pro/data/upload/1/201705/0815415604151587.png)

3 验证方法

批处理验证：

   1) 在soapui添加测试链接，以本地启动的服务为例，链接为 http://localhost:8080/wms-api/api/wms/loadTasks

   2) 在soapui中添加参数，参数为：[{'taskNo':'1';'SN':'1';'BDNO':'3';'pn':'4';'weight':'5';'height':'6';'preBoxes':'7';'opTime':'2017-04-10 13:52:36';'planningNo':'1'},{'taskNo':'2';'SN':'2';'BDNO':'3';'pn':'4';'weight':'5';'height':'6';'preBoxes':'7';'opTime':'2017-04-10 13:52:36';'planningNo':'1'}]

   3) 可在debug模式下结合数据库，看到当未执行executeBatch()操作时，数据库中并没有相应数据

回滚验证：

 1)对以上数据再次执行插入操作，由于业务流水表(HH\_MSGINFOLOG)优先于任务插入执行,但由于任务接收表(HH\_TASK)中的主键冲突，亦可看到业务流水表中数据已经回滚（即并未有此条数据）；

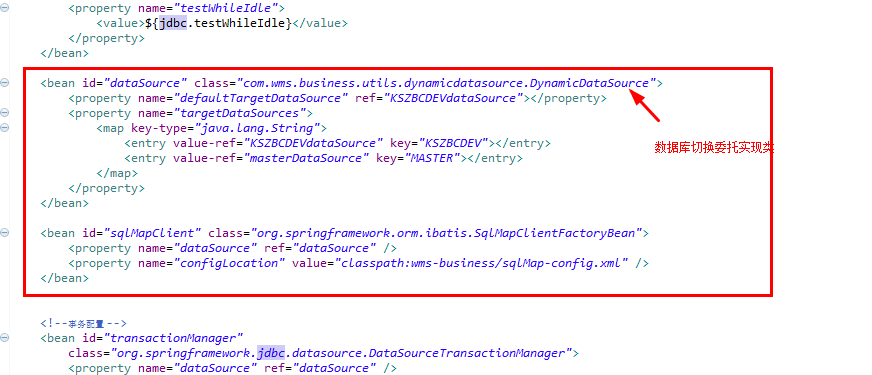
## 8.9数据库切换实现：

实现方式：

1 实现原理：通过配置business工程下的spring-service.xml将数据切换委托给对应的实现类。当在业务中需要进行数据库切换时，通过类中的静态方法进行切换（说明：需在api层进行数据切换；原因：当业务流程执行到business层时，数据源已确定，若在business内切换数据源，将引起冲突）

2 具体实现：

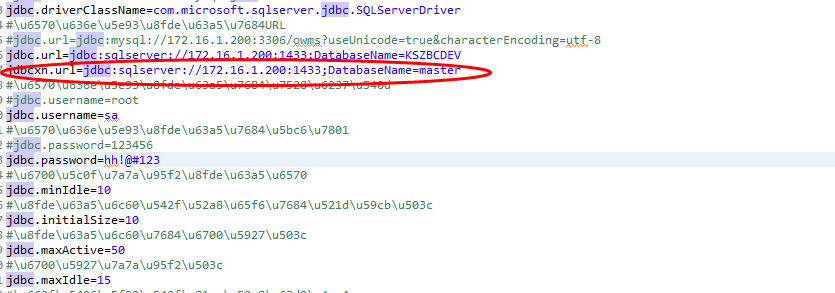
   1）配置business工程下的spring-service.xml文件的委托实现类，配置截图如下：

[](http://172.16.1.201/pro/data/upload/1/201705/090852080930fhf.png)

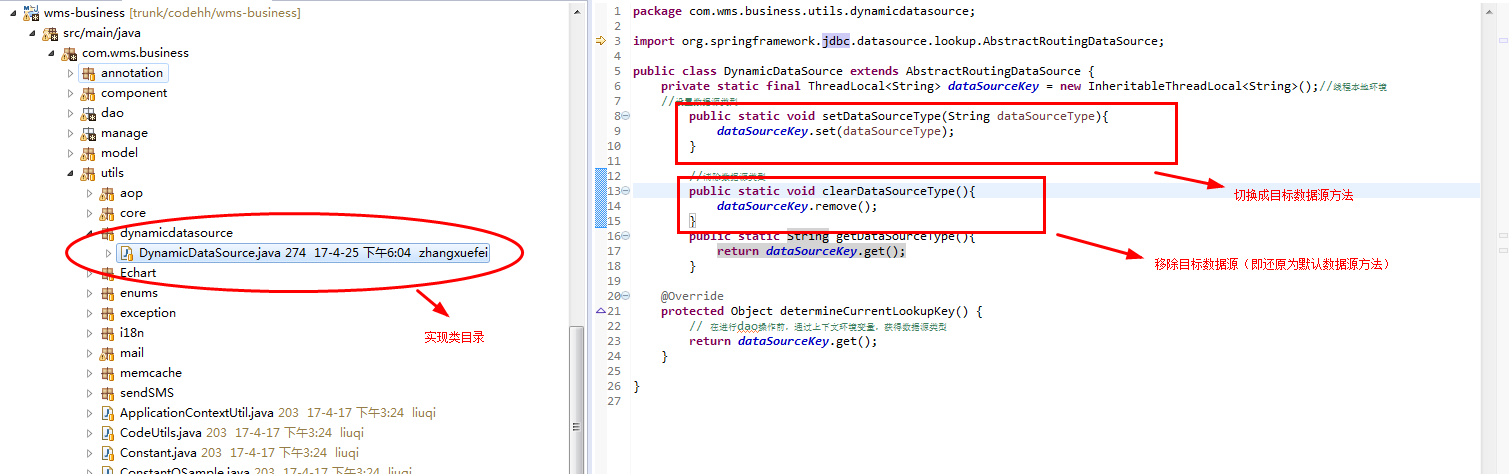
  2）配置bussiness工程下的spring-service.xml文件的要切换的数据库配置项，其中${jdbcxn.url}配置项是在jdbc.properties文件中添加了相应配置引用而来，截图如下：

[](http://172.16.1.201/pro/data/upload/1/201705/0908554402662pit.png)

 jdbc.properties配置截图如下：

[](http://172.16.1.201/pro/data/upload/1/201705/0909080607112b9b.png)

3）编写数据库切换实现类，实现类目录为：/wms-business/src/main/java/com/wms/business/utils/dynamicdatasource/DynamicDataSource.java，目录截图为：

[](http://172.16.1.201/pro/data/upload/1/201705/0909144407606587.png)

3验证方法：

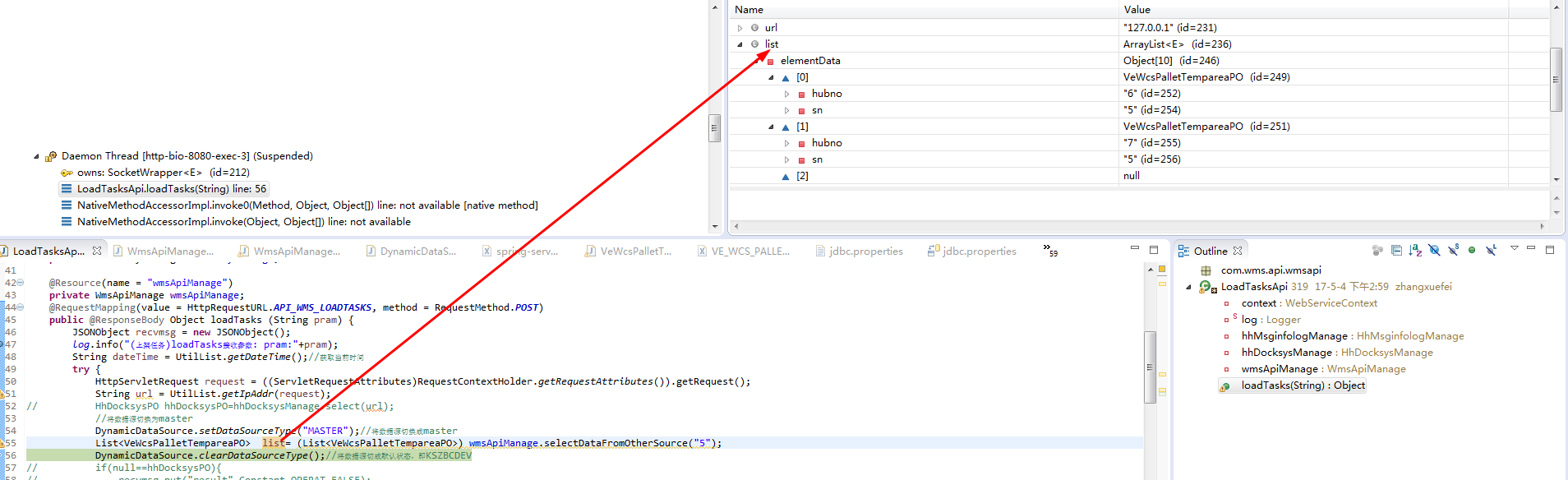
目标：在进行数据插入前查询其他库中的数据，以在任务接收表插入数据之前查询master库下的VE\_WCS\_PALLET\_TEMPAREA（此表为自己创建的测试表）表中数据为例

1）在soapui中调用http://localhost:8080/wms-api/api/wms/loadTasks接口，添加参数（[{'taskNo':'1';'SN':'1';'BDNO':'3';'pn':'4';'weight':'5';'height':'6';'preBoxes':'7';'opTime':'2017-04-10 13:52:36';'planningNo':'1'},{'taskNo':'2';'SN':'2';'BDNO':'3';'pn':'4';'weight':'5';'height':'6';'preBoxes':'7';'opTime':'2017-04-10 13:52:36';'planningNo':'1'}]）

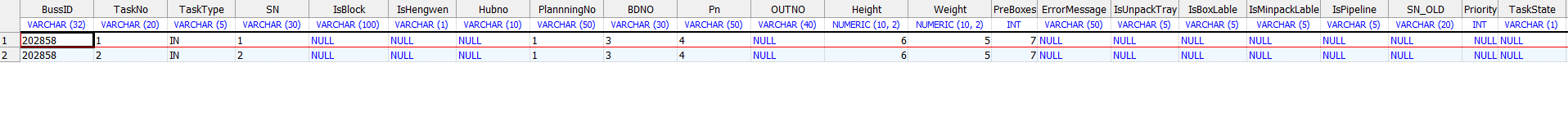
2 ) 在api层调用数据源切换方法及具体业务实现

3 ) 可在debug模式下观察到在master库中VE\_WCS\_PALLET\_TEMPAREA表的相应查询数据，截图如下

4）可在KSZBCDEV数据库的HH\_TASK表中看到插入的数据

[](http://172.16.1.201/pro/data/upload/1/201705/0909335009796hf9.png)

  HH\_TASK表中的数据：

[](http://172.16.1.201/pro/data/upload/1/201705/09095347043b2o.png)

## 获取请求方IP地址实现：

rest接口：

实现原理：通过HttpServletRequest得到请求头（Header）,对Header处理后（处理方式已封装为getIpAddr），获取请求方IP；

调用方式：截图如下



验证方式

1）在soapui中调用http://localhost:8080/wms-api/api/wms/loadTasks接口

2）在debug模式下，可看到请求方Ip,截图如下（此处为本地调用的地址）：

