**代码走查会议纪要及走查结论**

时间：2016-8-16 16:00-18:30

地点：软件园科技蓝色会议室

会议主题：跟单融资系统代码走查

参会人员：夏可、顾也科、张帆、海波、李冬翔、左志超等

记录人：李冬翔

**会议内容：**   
跟单融资系统核心代码走查。

**代码走查问题：**

1. **性能隐患问题，危害程度：高**

**问题1**

**问题描述：**在做银联和商户数据比对时，程序先取出了银联数据进行循环，每次循环再通过一次SQL查询进行数据的对比，如此一来，循环次数为n，即会执行n次查询，当数据量十分巨大时，会出现性能瓶颈。

**改进方案：**用两次SQL查询分别获取商户和银联的数据，且查询只取出需比较的必要字段，获取结果集后，在内存中进行数据比对。

从该业务需要来看，需要比对的字段在4到5个左右，每个字段平均长度在50个字节以下，即每条记录在200-250个字节。假设我行该业务数据量在10万量级，那单次取到内存中的数据量为100,000×250 = 25,000,000 ≈ 25MB，即在该数量级下，内存中数据在25MB以下，因此对内存不会造成过多的负担。

**问题2**

**问题描述：**项目中用到厂商开发的框架，该框架中很多SQL语句直接使用String拼接，并以Statement形式直接交给数据库执行，存在性能隐患。由于项目组无法确定该框架代码是否被引用到，因此会上决定由项目组进行会后确认，如该类代码被使用，则项目组列出整改时间。

**改进方案：**调用SQL都需使用预编译的方式，且不许采用String多次拼接SQL语句，以免产生性能问题。

1. **开发规范，危害程度：中**

**问题3**

**问题描述：**项目中使用的数据库用户名密码都以明文形式保存在配置文件中，存在一定安全隐患。

**改进方案：**行内需制定密码管理办法，统一如SFTP、数据库等环境的用户名密码配置方案。

**问题4**

**问题描述：**项目中使用System.out.println()、System.err.println()，该打印输入方法会将内容统一输出至WAS的System.out及System.err文件，如果WAS中同时挂载多个应用，那么所有应用的日志都将打印到这两个文件中，对后续各项目问题的排查及定位将造成巨大的负面影响。

**改进方案：**每个应用将日志打印至应用自身的日志文件中，不许使用 System.out.println()、System.err.println()作为日志输出，以规避上述问题。

1. **编码规范，危害程度：低**

**问题5**

**问题描述：**在ZipParse.java的条件判断语句中，多次出现和数字“6”直接做比较判断的语句，经询问该数字代表的是文件个数。此类编码方式不符合编码规范，改进的方法是定义一个常量，并将“6”赋值给该常量，且常量的命名需能让代码的阅读者明白该常量所代表的含义。

**改进方案：**

1. 将多次出现的数字（字符串）定义为常量，且该常量的命名需具有可读性；
2. 如果该数字（字符串）仅出现一次，可以辅以注释以说明该处数字（字符串）的含义；
3. 如果该常量可能会因项目的变化而经常修改，则需将该常量维护在配置文件中，以尽量减少程序的改动。
4. **补充问题：**

项目之前有过一个生产问题，该问题很具代表性，因此代码走查会议上和项目组进行了详细沟通，现附上该问题和解决方案

**问题描述：**项目在做商户及银联数据对比时，采用的数据类型都为浮点型，比较的方式使用了简单的等值比较，即用等号来判断两个浮点数是否一致。浮点数在计算机当中是二进制的，因此实际上大多数浮点数都是无法精确的表达的，所以会产生如下的问题:

double a = 0.02； double b = 0.02;

此时 a和b相同；

double a = 0.02; double b = 1/50f;

此时 a 和 b并不相同；

所以就产生了项目中使用等值比较失效的情况，此情况无论是使用double亦或是BigDecimal中都无法直接规避。

**改进方案：**

**有两种方案供参考：**

1、可以确定绝对的误差值

当项目的业务逻辑能够确定该项目的绝对误差值时，如，跟单项目的贷款误差可以到1元，则可以采用绝对误差的方式，

abs(A-B) < epsilon

即两个浮点数之差的绝对值小于一个可确定的误差值时，则认为该两个浮点数相同。

2、项目无法确定绝对误差值时，可以采用相对误差

相对误差即指以运算数作为基准，设定一定的比例，当该误差小于该比例，则认定两个运算数相等。

附上该考虑绝对误差及相对误差完整方案的伪代码：

/\*\*

\*

\* **@param** a

\* **@param** b

\* **@param** absError 绝对误差

\* **@param** relError 相对误差

\* **@return**

\*/

**boolean** IsEqual(**float** a, **float** b, **float** absError, **float** relError )

{

**if** (a==b)

{

**return** **true**;

}

**if** (Math.*abs*(a-b) < absError )

{

**return** **true**;

}

**if** (Math.*abs*(a) > Math.*abs*(b))

{

**return** Math.*abs*((a-b)/a) < relError ? **true**: **false**;

}

**else**

{

**return** Math.*abs*((a-b)/b) < relError ? **true**: **false**;

}

}

**结论及代码评分：**

虽然本次跟单融资系统核心代码走查出5个问题，但其整体代码质量良好，编码习惯也较好，注释完整，一些细节的处理也十分到位，如事务处理、异常错误处理等。整体代码质量评分为90分。