# HOMEWOKK01 设计文档

1562810322 王归航

## 实现说明

本作业在eclipse-jee-oxygen-M3-win32-x86\_64版本下进行开发实现。

已经实现了作业要求中如下部分的内容：

1. 接收用户从控制台输入任意字符串(例如java.lang.String)，利用Class.forName装载对应的类，如果装载成功打印方法信息到控制台
2. 定义自定义类型类ClassNotFindException，当1中的装载失败时抛出。
3. 将查询到的类、方法和方法参数信息新增到ClassInfo、MethodInfo和ParamInfo表(如果相关信息在表里已经存在，则更新相关的表)
4. 将3个表数据导出到1个XML文件中
5. 将XML文件中的数据导入数据库
6. 将3个表数据导出到1个JSON文件中
7. 将JSON文件的数据导入数据库

## 具体流程

以下为各实现的方法与具体说明：

1. 使用Scanner类型读取用户输入的类的字符串，通过Class.forName 装载相对应的类，如果装载成功，在console栏中输出成功信息。
2. 定义自定义类型ClassNotFindException，在调用 Class.forName 方法时系统将会抛出ClassNotFoundException 方法，在catch{}处理中抛出自定义的ClassNotFindException异常，在main方法中catch并输出相应提示信息，并开始新一轮的输入等待。
   1. 通过依次执行三个SQL语句classInsertSQL，methodInsertSQL和paramInsertSQL将相应的属性插入数据库的对应表中。因为3个表之间有主外键的关联，因此采用如下的方法：执行完classInsertSQL后存储返回的主键classId,在执行methodInsertSQL语句时将classId插入到MethodInfo表中的classId 属性中，并存储返回的主键methodId。在执行paramInsertSQL语句时将methodId插入到ParamInfo表中的methodId属性中，由此完成数据库中主外键之间的对应关联。
   2. 在执行insert操作前，将会执行定义的checkSQL语句，即在ClassInfo表中查找将要插入的类名，若在数据库中已经存在，说明此类的信息事先已经插入，程序将放弃执行3.(a中的操作，输出提示信息并开始新一轮的输入等待。
3. 通过依次执行查询语句classSelectSQL，methodSelectSQL，paramSelectSQL并将查询结果转换为javabean对象列表。依次遍历列表并通过XStream对象将Javabean转换为XML对象并以字符串的形式存储。为了导入方便，在设置导出排版的时候，用“ClassInfo:”，“MethodInfo:”，“ParamInfo”各占一行将xml对象间隔开来。最后通过系统输出流对象实现将存储结果以xml文件的形式导出。
   1. 由于在4中的导出操作中对于xml文件做了规范，因此只需要在读取xml的时候根据得到的xml字符串所处的不同位置调用XStream类方法生成不同类型的JAVABEAN对象即可。
   2. 在执行插入操作前，会通过classDeleteSQL，methodDeleteSQL，paramDeleteSQL语句清空数据库。
   3. 由于数据库中的主键设置为自增，因此存储在文件中的classId,methodId信息在插入语句中将无法使用。为此，通过两个hashMap对象hashMap\_classId和hashMap\_methodId实现。以hashMap\_classId为例，当向数据库中插入一个classInfo对象的信息后，存储返回的主键值methodId。在hashMap\_classId添加一对键值对methodInfo.getClassId()：classId。这样，当插入文件中以methodInfo.getMethodId()作为外键的methodInfo对象时，就可以通过hashMap\_classId.get(paramInfo.getClassId())获取到新的classId作为其外键。由此插入的三张表就可以重新建立联系。
4. 通过依次执行查询语句classSelectSQL，methodSelectSQL，paramSelectSQL并将查询结果转换为javabean对象列表。依次遍历列表并通过Gson对象将Javabean转换为JSON对象并以字符串的形式存储。为了导入方便，在设置导出排版的时候，用“ClassInfo:”，“MethodInfo:”，“ParamInfo”各占一行将json对象间隔开来。最后通过系统输出流对象实现将存储结果以json文件的形式导出。
   1. 由于在6中的导出操作中对于json文件做了规范，因此只需要在读取json的时候根据得到的json字符串所处的不同位置调用Gson类方法生成不同类型的JAVABEAN对象即可。
   2. 在执行插入操作前，会通过classDeleteSQL，methodDeleteSQL，paramDeleteSQL语句清空数据库。
   3. 由于数据库中的主键设置为自增，因此存储在文件中的classId,methodId信息在插入语句中将无法使用。为此，通过两个hashMap对象hashMap\_classId和hashMap\_methodId实现。以hashMap\_classId为例，当向数据库中插入一个classInfo对象的信息后，存储返回的主键值methodId。在hashMap\_classId添加一对键值对methodInfo.getClassId()：classId。这样，当插入文件中以methodInfo.getMethodId()作为外键的methodInfo对象时，就可以通过hashMap\_classId.get(paramInfo.getClassId())获取到新的classId作为其外键。由此插入的三张表就可以重新建立联系。

## 运行说明

本项目共提供5个运行类，分别为UpdateDatebase, JsonExport, JsonImport, XmlExport, XMLImport。

UpdateDatebase获取用户输入并将相应的类的信息存储到数据库中。

JsonExport将数据库中的信息以Json文件的形式导出。

JsonImport将导出的Json文件导入数据库中。

XmlExport将数据库中的信息以Xml文件的形式导出。

XmlImport将导出的Xml文件导入数据库中。