Kettle学习简介

叶盛

2017

目录

[0.概念介绍 2](#_Toc476749026)

[1.UI界面 2](#_Toc476749027)

[2.核心功能： 3](#_Toc476749028)

[2.1 core 3](#_Toc476749029)

[2.2 dbdialog 3](#_Toc476749030)

[2.3 engine 4](#_Toc476749031)

[3.表数据迁移 8](#_Toc476749032)

[3.1目标库建表 8](#_Toc476749033)

[3.2数据插入 9](#_Toc476749034)

[3.3数据类型映射 10](#_Toc476749035)

# 0.概念介绍

Job：一个作业，由不同逻辑功能的entry组件构成，数据从一个entry组件传递到另一个entry组件，并在entry组件中进行相应的处理。多表迁移主要是由job来完成。

Entry：实体，即Job型的组件。用来完成特定功能单元，用来完成整个转换过程的每一个特定步骤。

Transformation：完成一个数据的转换过程。主要指的是从各类含有数据的文件中提取数据。

Step：步骤，是Transformation的功能单元，用来完成整个转换过程的一个特定步骤。

Hop：工作流或者转换过程的流向指示，从一个组件指向另一个组件，有三种hop：无条件流向、判断为真流向、判断为假流向。

Kettle的源码配置可以按照教程：<http://blog.csdn.net/zutsoft/article/details/52998148>

我配置的工程：<https://github.com/ysaicll/kettle-6.1.git>

按教程，6.1可以成功，7.0未成功，调试环境为MyEclipse，在eclipse中配置也未成功。配置出来的Kettle工程与原始工程的目录结构有一些差别，现有工程的主要文件夹有core,dbdialog,engine,ui,plugin.

# 1.UI界面

**UI布局的实现在ui文件夹中实现**

主要包括主界面的实现，控件的实现，基本上就是一个控件一个package。ui文件夹中包含图形界面的主程序spoon.java, 文件位于ui/org/pentaho/di/ui/spoon/spoon.java.

菜单栏布局在ui/org/pentaho/xul/swt/tab/menubar.xul(注：xul为可扩展[标记语言](http://baike.baidu.com/view/329009.htm)用户[界面](http://baike.baidu.com/view/240704.htm)语言。它是为了支持[Mozilla](http://baike.baidu.com/view/393243.htm)系列的应用程序（如Mozilla Firefox和Mozilla Thunderbird）而开发的使用者界面标示语言)中被定义。菜单布局以及选项可以在其中修改。从menubar.xul中的command命令中可以调动Spoon.java类中相应的方法，所以开发可以以此作为一个起点。如图1中，红色方框内就是菜单栏“help”的布局选项。

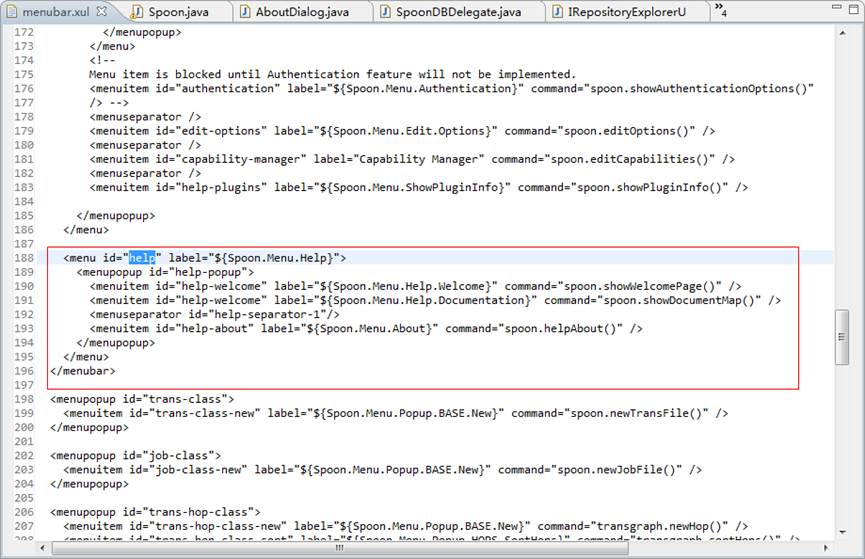


图1

/Kettle6.1/ui/org/pentaho/di/ui/spoon/messages/messages\_zh\_CN.properties中的信息为xul文件label的对应中文列表，messages文件夹主要目的是实现语言的国际化。各国语言的描述信息在各个包对应的.message中的properties文件中。ui/org/pentaho/di/ui/i18n及相关的包中实现界面的国际化功能可以直接复用。

在/Kettle6.1/ui/org/pentaho/di/ui/core/dialog/Splash.java中实现加载页面时的动画。

# 2.核心功能：

## 2.1 core

**core文件夹主要实现数据库连接并与之进行交互的功能**

在/Kettle6.1/core/org/pentaho/di/core/database/Database.java中连接数据库，拼凑SQL语句来获取数据库中的数据

## 2.2 dbdialog

**dbdialog文件夹实现连接数据库时的弹窗引导**

/Kettle6.1/dbdialog/org/pentaho/ui/database/event/DataHandler.java处理数据库连接的高级选项,如图2就是一个连接数据库的高级向导

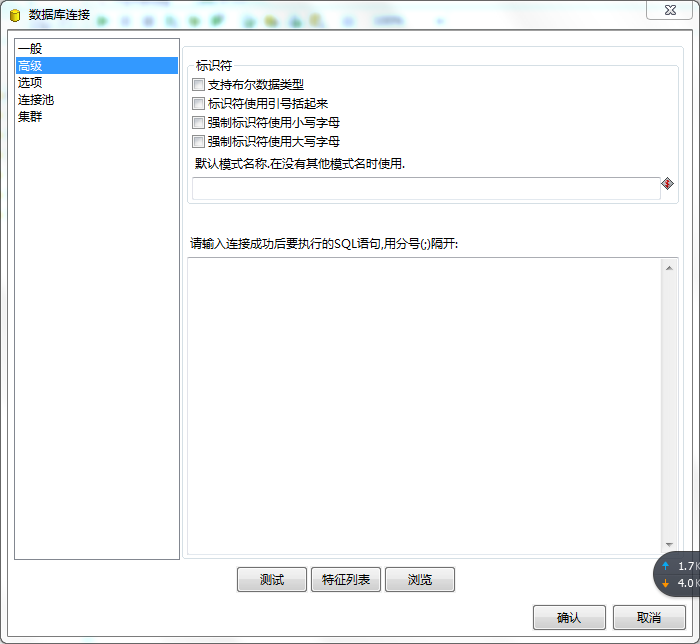


图2

## 2.3 engine

**Engine文件夹实现最重要的数据提取和导出功能，它是kettle的功能实现的核心**

/Kettle6.1/engine/org/pentaho/di/job/entries路径下的package为每一个工作流的实体对象进行操作定义

以“复制多表向导”的一般过程为例：

1.ui/org/pentaho/di/ui/spoon/spoon.java类加载主界面，windows下以图形界面的形式启动的代码的main函数就在这个类中。

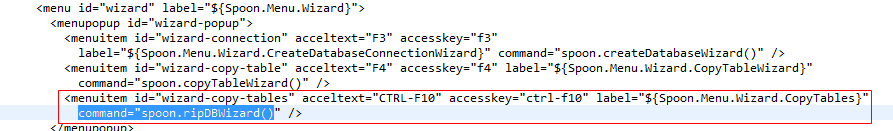
2. 从xul文件中发现“复制多表向导”的command为spoon.ripDBWizard()。点击工具-向导-复制多表向导，调用spoon.ripDBWizard()方法。如图3,4所示

图3

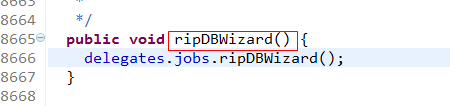


图4

该方法的实现在：/Kettle6.1/ui/org/pentaho/di/ui/spoon/delegates/SpoonJob-Delegate.java

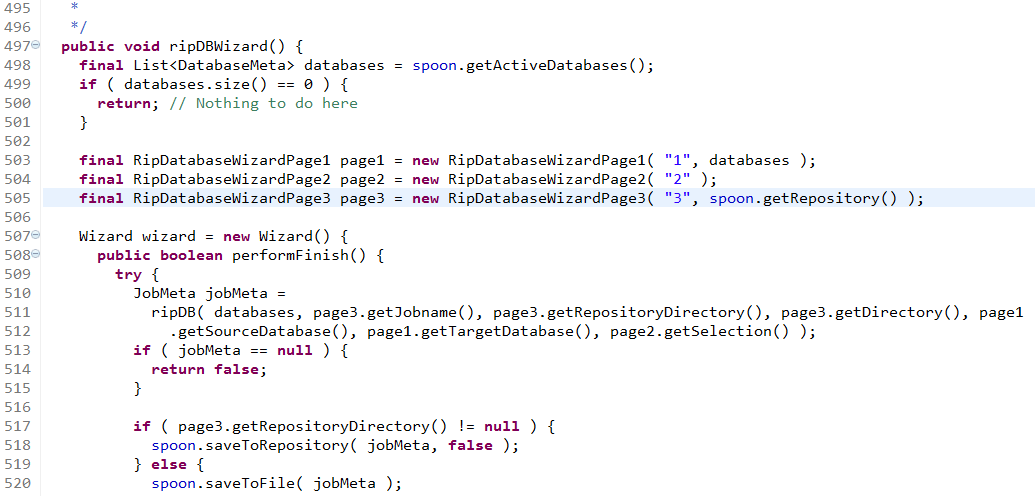


图5

/Kettle6.1/core/org/pentaho/di/core/database/DatabaseMeta.java中设置数据库连接时的参数

/Kettle6.1/core/kettle-database-types.xml有数据库元类型的清单

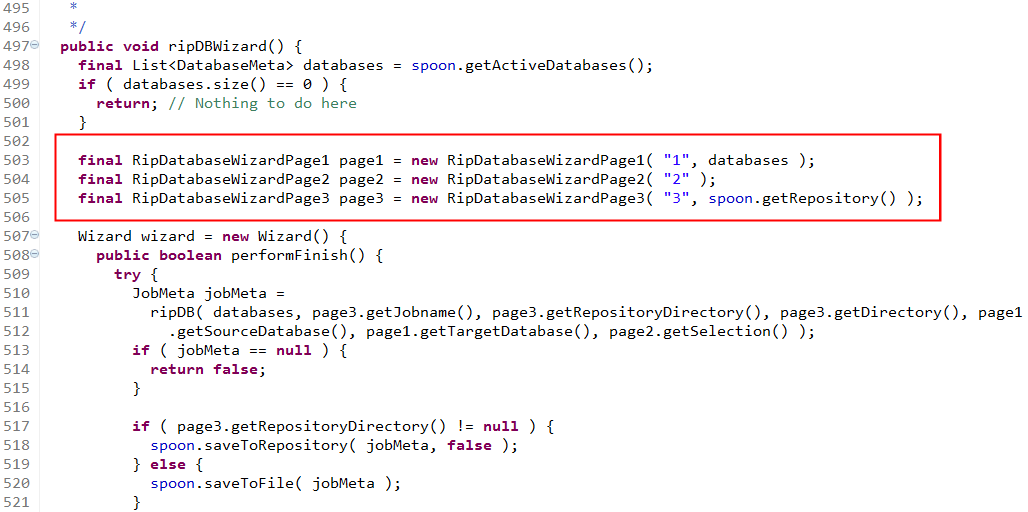


图6

在复制多表向导中分别需要经历的三个页面，之后判断page3页面是否能够保存，（即该job能否保存），如果能保存则进行下一步。

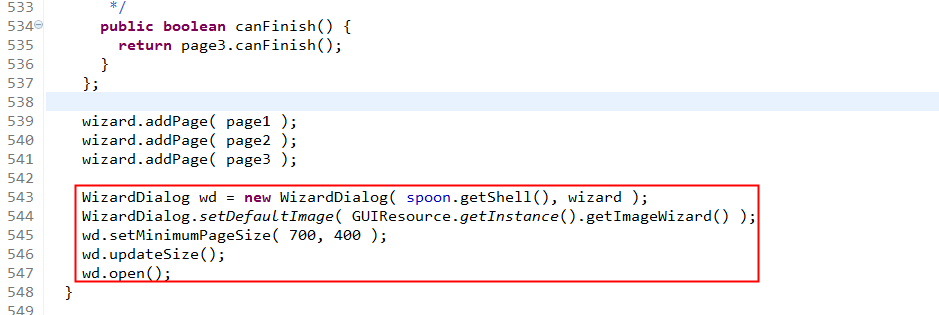


图7

spoon.ripDBWizard()方法执行完毕后，迁移的配置过程就结束了，job也作为jobmeta被保存下来，接下来就要执行迁移过程。

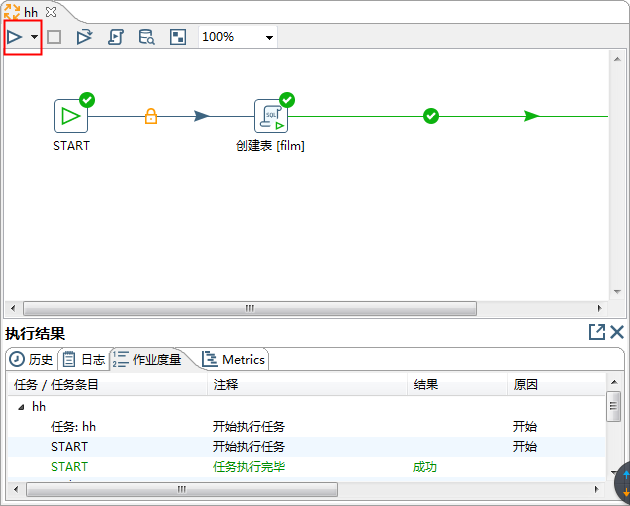


图8

点击图8中红色方框的执行按钮，Spoon.java文件的executeFile()方法执行，如图9，如果说job是存在的，就可以执行executeJob()方法了，该方法的实现在/Kettle6.1/ui/org/pentaho/di/ui/spoon/delegates/SpoonJobDelegate.java中

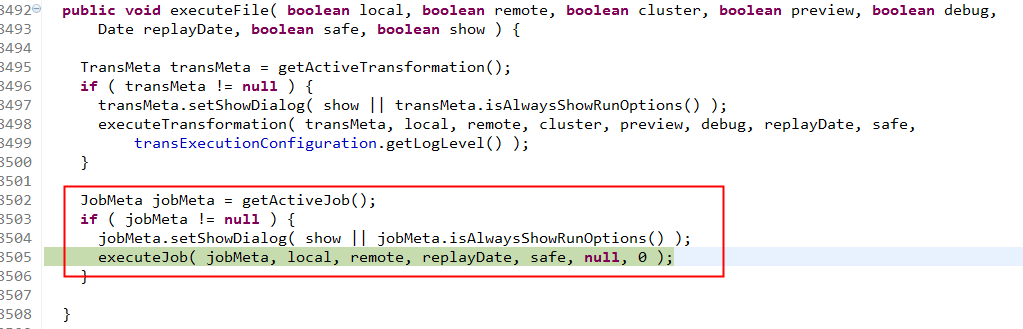


图9

所以一个完整的多表复制过程大致就是以上的过程，两个最重要的类分别为Spoon.java和SpoonJobDelegate.java，一般的过程都会通过这两个类去调度实现。

# 3.表数据迁移

/Kettle6.1/core/kettle-database-types.xml中有各类数据库的名字，按照需求，将会显示在备选框中，只要留下v7和v8的引导窗口即可，其他的数据库类型可以删除。但Ingres相关数据库不能删除，不知道为什么，删除Ingres之后就导致新建数据连接不能弹出候选框。

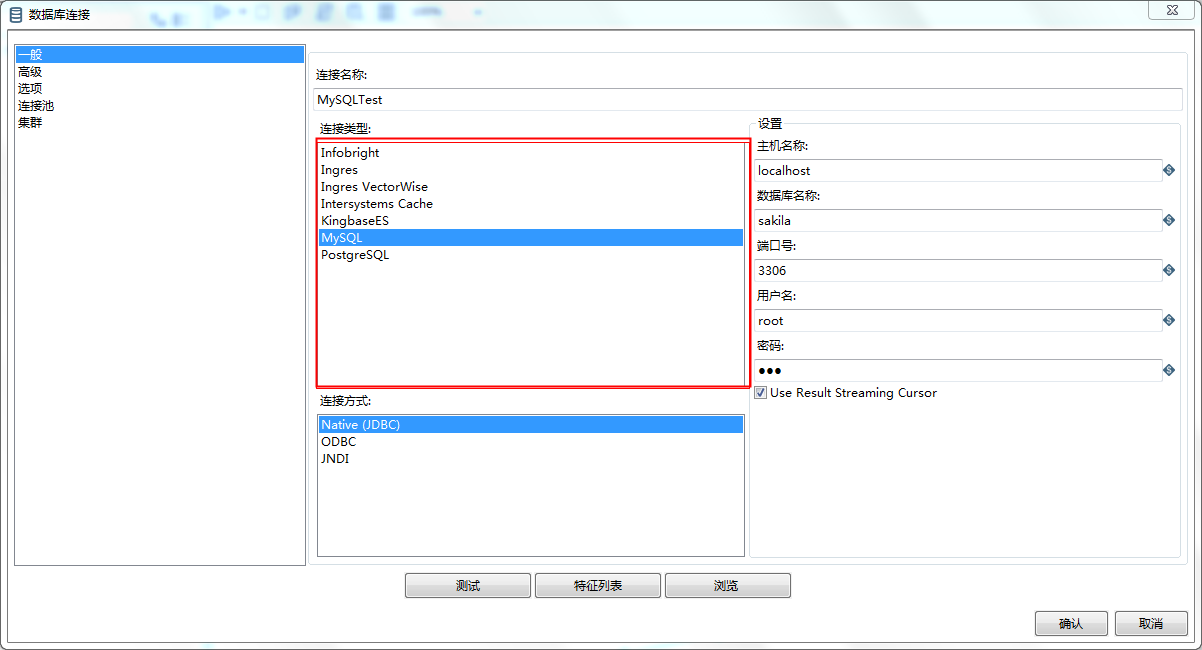


图10 数据库类型选择

## 3.1目标库建表

建表语句的实现在/core/org/pentaho/di/core/database/Database.java中

在Database.java类中getCreateTableStatement()方法实现建表语句。在kettle处理表的时候就已经开始获取建表语句了。即，当配置好源库和目标库，选择好jobMeta的存储路径的时候。此时会遇到三个问题

问题一：不能正确匹配数据类型（主要是TIMESTAMP的类型，/Kettle6.1/core/org/pentaho/di/core/database/\*这个包下面的类为指定的目标数据库创建建表sql语句，比如，要为pg创建建表语句，就是去找/Kettle6.1/core/org/pentaho/di/core/database/PostgreSQLDatabaseMeta.java）。易解决

问题二：不支持的数据类型BLOB，难解决

问题三：实际上源码中已经提供了主键的语句，但无法识别源库的主键，创建主键的代码一般不会被执行，难解决

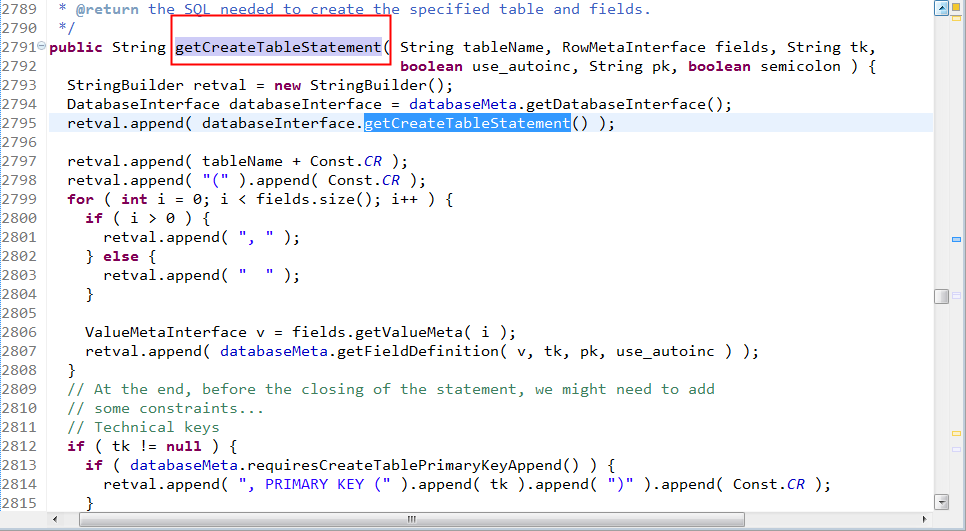


图11 getCreateTableStatement()

## 3.2数据插入

同样在这个类中getInsertStatement()方法实现插入语句。

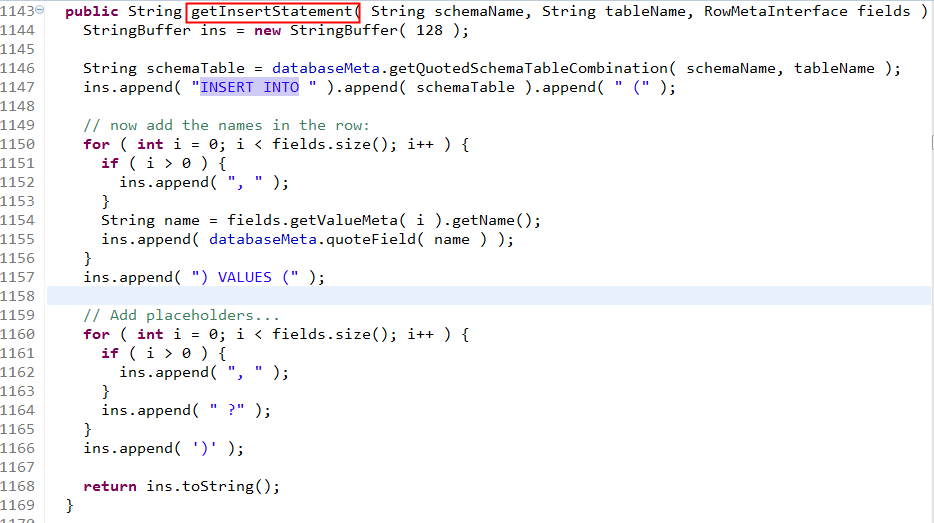


图12 getInsertStatement()

执行完getInsertStatement()方法后的insert语句形如：INSERT INTO TABLE VALUES (?, ?)，此时就要调用java.sql中preparedstatement的addBatch()方法进行源库的数据批量获取至缓冲区，然后直接执行executeBatch()方法将获取到缓冲区的数据进行批量插入目的数据库。最后clear batch执行下一次批处理。

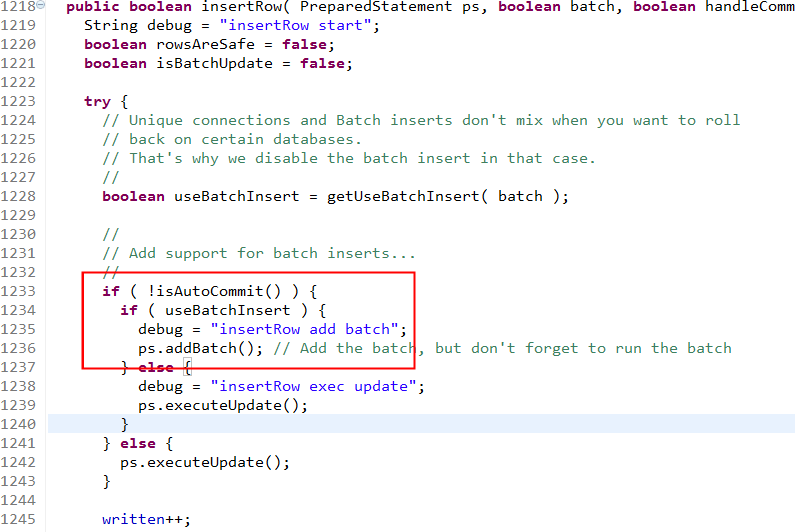
创建的sql语句将会被/Kettle6.1/engine/org/pentaho/di/trans/TransMeta.java应用，接下来就是/Kettle6.1/ui/org/pentaho/di/ui/spoon/delegates/SpoonJobDelegate.java来运行job，表处理完成。

图13 调用addBatch()方法

## 3.3数据类型映射

/Kettle6.1/core/org/pentaho/di/core/database/\*这个包下面的类在创建CREATE语句时提供各个数据库对应的数据类型，即在此完成数据类型的映射。目标库是什么，就到目标库对应的类中去确定需要转换的数据类型。

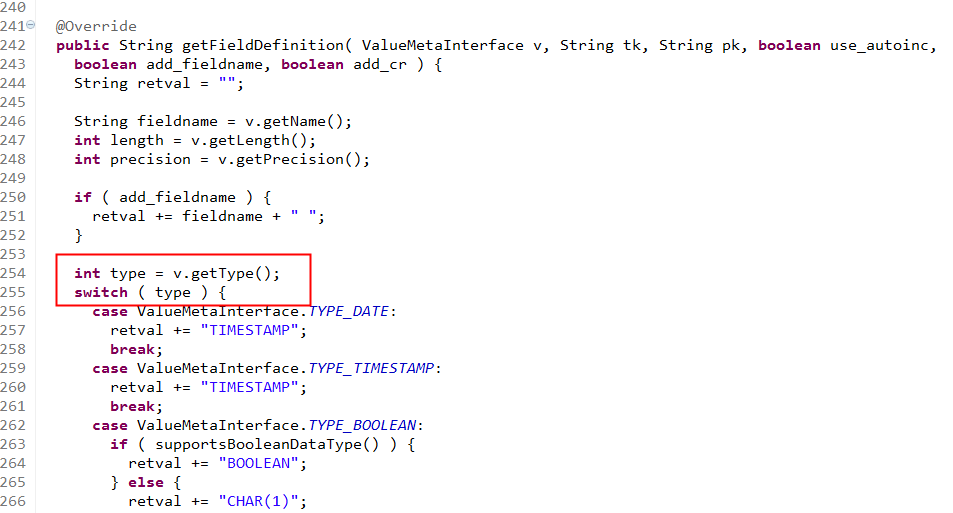


图13 getFieldDefinition()方法