链滴

java 调度 Kettle 时使用 jndi 连接数据库

作者：someone9891
原文链接：https://ld246.com/article/1561815630619
来源网站：链滴

看了Kettle好久了，不得不说，国内资料真的是少啊，又是买书，又是自己翻译官方文档，又是看源，好歹算是有些眉目了，今天有解决一大难题，在此记录下来，算是为国内Kettle圈奉献点力量吧。

Java调度Kettle

关于Java调度Kettle，在网上还是能搜到一些内容的，其实还是比较简单的，Kettle在Spoon中设计后，会生成.ktr 和.kjb 文件，分别对应Kettle的转换和作业。Java正是通过Kettle提供的API调用这些件的。

准备工作

在通过Java调用Kettle时，需要下面几个jar包：

```
<dependency>
        <groupId>pentaho-kettle</groupId>
        <artifactId>kettle-engine</artifactId>
        <version>${kettle.version}</version>
    </dependency>
    <dependency>
        <groupId>pentaho</groupId>
        <artifactId>metastore</artifactId>
        <version>${kettle.version}</version>
    </dependency>
    <dependency>
        <groupId>pentaho-kettle</groupId>
        <artifactId>kettle-core</artifactId>
        <version>${kettle.version}</version>
        <exclusions>
            <exclusion>
                <groupId>jug-lgpl</groupId>
                <artifactId>jug-lgpl</artifactId>
```

```
        </exclusion>
        <exclusion>
          <groupId>secondstring</groupId>
          <artifactId>secondstring</artifactId>
        </exclusion>
        <exclusion>
          <artifactId>xercesImpl</artifactId>
          <groupId>xerces</groupId>
        </exclusion>
        <exclusion>
          <groupId>org.apache.xmlgraphics</groupId>
          <artifactId>batik-js</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
```

这些jar 包在maven 仓库是没有的，所以还需要配上kettle 的maven仓库地址：

```
<!-- kettle中央仓库 -->
  <repositories>
    <repository>
      <id>pentaho-public</id>
      <name>Pentaho Public</name>
      <url>http://nexus.pentaho.org/content/groups/omni</url>
      <releases>
        <enabled>true</enabled>
        <updatePolicy>always</updatePolicy>
      </releases>
      <snapshots>
        <enabled>true</enabled>
        <updatePolicy>always</updatePolicy>
      </snapshots>
    </repository>
  </repositories>
```

配置好maven依赖以后，就可以通过Java来操作kettle的.kjb 和.ktr 文件了。

连接资源库

在kettle中，有一个很重要的概念就是资源库，资源库通俗理解就是管理kettle元数据，在多人开发时候尤为重要，可以把资源库放在公共的地方供大家访问，目前kettle支持三种资源库：

- 数据库资源库
- 文件资源库
- Pentaho 资源库

本文主要记录Java调用时使用jndi 连接数据库的问题，Kettle概念就不多赘述了如果需要了解，可以看本人呢还未翻译完全的非官方文档.

在Java调用时，首先就是连接资源库，本人为了方便svn 版本控制，使用的是文件资源库。

获取资源库

以下代码仅仅是获取文件资源库的:

```
/**
 *
 * @param basePath 主路径
 * @return KettleFileRepository
 */
public static KettleFileRepository getFileRepository(String basePath){

    KettleFileRepositoryMeta meta = new KettleFileRepositoryMeta();
    URL url = KettleUtils.class.getClassLoader().getResource(basePath);
    String baseDirectory = url == null ? null : url.getPath();
    meta.setBaseDirectory(baseDirectory);
    KettleFileRepository kettleFileRepository = new KettleFileRepository();
    kettleFileRepository.init(meta);
    return kettleFileRepository;
}
```

其中，通过

```
URL url = KettleUtils.class.getClassLoader().getResource(basePath);
    String baseDirectory = url == null ? null : url.getPath();
```

获取到资源库路径，我的资源库是直接放在项目目录的，也可以使用绝对路径。

连接资源库

在连接资源库进行操作之前，需要先进行初始化，而想要使用jndi连接数据库，也就需要在这个地方文章了，首先，kettle初始化的代码是

KettleEnvironment.init(true); 就这么简单一句，即可，kettle会帮你完成初始化。

在这个init 方法，需要一个boolean 类型的参数，这个参数就是告诉kettle需要初始化jndi 数据库配文件。

首先看以下这个方法的源码：

```
/**
 * Initializes the Kettle environment. This method performs the following operations:
 * <p/>
 * - Creates a Kettle "home" directory if it does not already exist - Reads in the kettle.propert
es file -
 * Initializes the logging back-end - Sets the console log level to debug - If specified by par
meter, configures
 * Simple JNDI - Registers the native types and the plugins for the various plugin types - Rea
s the list of variables
 * - Initializes the Lifecycle listeners
 *
 * @param simpleJndi true to configure Simple JNDI, false otherwise
 * @throws KettleException Any errors that occur during initialization will throw a KettleExce
tion.
 */
public static void init( boolean simpleJndi ) throws KettleException {
  init( Arrays.asList(
    RowDistributionPluginType.getInstance(),
```

```
      StepPluginType.getInstance(),
      StepDialogFragmentType.getInstance(),
      PartitionerPluginType.getInstance(),
      JobEntryPluginType.getInstance(),
      JobEntryDialogFragmentType.getInstance(),
      LogTablePluginType.getInstance(),
      RepositoryPluginType.getInstance(),
      LifecyclePluginType.getInstance(),
      KettleLifecyclePluginType.getInstance(),
      ImportRulePluginType.getInstance(),
      CartePluginType.getInstance(),
      CompressionPluginType.getInstance(),
      AuthenticationProviderPluginType.getInstance(),
      AuthenticationConsumerPluginType.getInstance(),
      EnginePluginType.getInstance()
    ), simpleJndi );
  }
```

可以看到，这个参数被传递到下个方法，我们继续往下看：

下面这个方法代码有点多，我只取主要的部分：

```
// Configure Simple JNDI when we run in stand-alone mode (spoon, pan, kitchen, carte, ... NO
 on the platform
        //
        if ( simpleJndi ) {
          JndiUtil.initJNDI();
        }
```

可以看到，在代码里，判断了前面那个参数的值，如果是true,就调用JndiUtil.initJNDI();进行jndi的始化，JndiUtil 类的代码：

```
public class JndiUtil {

  public static void initJNDI() throws KettleException {
    String path = Const.JNDI_DIRECTORY;

    if ( path == null || path.equals( "" ) ) {
      try {
        File file = new File( "simple-jndi" );
        path = file.getCanonicalPath();
      } catch ( Exception e ) {
        throw new KettleException( "Error initializing JNDI", e );
      }
      Const.JNDI_DIRECTORY = path;
    }

    System.setProperty( "java.naming.factory.initial", "org.osjava.sj.SimpleContextFactory" );
    System.setProperty( "org.osjava.sj.root", path );
    System.setProperty( "org.osjava.sj.delimiter", "/" );
  }

}
```

这里就是初始化jndi 的部分了，这里String path = Const.JNDI_DIRECTORY; 是kettle默认的jndi 配

文件的位置，所以我们只需要将这个 值 修改了，就可以让kettle加载我们的配置文件，也就可以使用jdi 数据源了。

于是我进行了尝试，直接在初始化之前，给Const.JNDI_DIRECTORY 赋值，证明猜想确实是对的。

```
URL url = KettleUtils.class.getClassLoader().getResource(JNDI_PATH);
Const.JNDI_DIRECTORY = url == null ? null : url.getPath();
```

JNDI_PATH 是我jndi 配置文件的目录名。

下面，将jndi 连接名作为变量传给kettle，就可以实现通过 通过变量连接jndi 数据库了。

```
job.setVariable("jndiName","ODS");
```

之后，在kettle中，只需要获取变量名对应的连接即可：

这样即可实现多个数据源通过变量切换。

在获取 到资源库对象，并且已经 初始化kettle以后，直接通过connect 方法就可以进行连接了，文资源库默认是没有密码的，如果是数据库资源库，默认的账户和密码是 admin/admin

```
//连接资源库
repository.connect(null,null);
```

下面，就可以开心的调用kettle文件了：

获取作业

```
RepositoryDirectoryInterface directoryInterface = repository.loadRepositoryDirectoryTree();
        JobMeta jobMeta = repository.loadJob(jobName,directoryInterface,null,null);
```

设置变量并执行kettle作业

```
Job job = new Job(repository,jobMeta);
        job.setVariable("jndiName","ODS");
```

原文链接：java 调度 Kettle 时使用 jndi 连接数据库

```
        //接口参数设置为变量
        for(Map.Entry<String,String> entry:params.entrySet()){
            job.setVariable(entry.getKey(),entry.getValue());
            System.out.println("正在设置变量，key:" + entry.getKey() + "，值为：" + entry.getVa
ue());
        }
        //日志级别
        job.setLogLevel(LogLevel.ROWLEVEL);
        //执行作业
```

现在主要做大数据，kettle只作为入门的工具，后面可能还会设计Hadoop、spark 之类的，欢迎一沟通交流。