Flink快速上手之Scala API使用

本文将介绍如何通过简单地几步来开始编写你的 Flink Scala 程序。

构建工具

Flink工程可以使用不同的工具进行构建，为了快速构建Flink工程，Flink为下面的构建工具分别提供了模板：

1、SBT
2、Maven

这些模板可以帮助我们组织项目结构并初始化一些构建文件。

SBT

创建工程

1、使用Giter8

可以使用下面命令插件一个Flink工程：

```
$ g8 tillrohrmann/flink-project
```

这将在 指定 的工程目录下，用 flink-project 模版创建一个 Flink 工程。 如果你没有安装 giter8, 请参照此 安装指南。

2、克隆repository

```
$ git clone https://github.com/tillrohrmann/flink-project.git
```

这将在 flink-project 目录下创建 Flink 工程。

3、运行 quickstart 脚本

```
$ bash <(curl https://flink.apache.org/q/sbt-quickstart.sh)
```

这将在 指定 的工程目录下创建 Flink 工程

构建工程

为了构建工程， 我们只需要简单地执行 sbt clean assembly 命令。
这将会在target/scala_your-major-scala-version/目录里面创建 fat-jar your-project-name-assembly-0.1-SNAPSHOT.jar。

运行工程

我们可以使用sbt run 命令来运行之前构建的工程。

默认情况下，作业将会和sbt运行在同一个 JVM 上。 如果你想让作业运行在不同的 JVM 上，将以下代码加入至 build.sbt：

```
fork in run := true
```

IntelliJ

我们推荐使用 IntelliJ 作为你的 Flink job的开发环境。 首先，你需要将新创建的工程导入至 Intellij。 操作步骤依次：File -> New -> Project from Existing Sources... ，
然后选择你的工程目录来导入工程。 IntelliJ 会检测到 build.sbt 文件并自动导入。

如果你想要运行 Flink 作业, 建议将 mainRunner 模块 作为 Run/Debug Configuration 的 classpath 路径。 这将保证在执行过程中， 所有标识为”provided”的依赖都可用。
你可以通过打开 Run -> Edit Configurations... 来配置 Run/Debug Configurations， 然后从 Use classpath of module 下拉框中选择 mainRunner。

Eclipse

如果你想要将新建的工程导入至 Eclipse, 首先你得为它创建 Eclipse 工程文件。
这些工程文件可以通过 sbteclipse 插件来创建。 将以下代码加入至 PROJECT_DIR/project/plugins.sbt 文件:

```
addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse-plugin" % "4.0.0")
```

在 sbt 交互式环境下使用下列命令创建 Eclipse 工程文件：

```
> eclipse
```

现在你可以通过打开 File -> Import... -> Existing Projects into Workspace 并选择你的工程目录，将之导入至 Eclipse。

Maven

要求（Requirements）

只需要安装Maven 3.0.4（或更高版本）和Java 7.x（或更高版本）即可。

创建工程

使用下列其中一个命令即可创建工程:

1、使用Maven archetypes

```
$ mvn archetype:generate                               ₩
    -DarchetypeGroupId=org.apache.flink            ₩
    -DarchetypeArtifactId=flink-quickstart-scala   ₩
    -DarchetypeVersion=1.0.0
```

这种创建方式允许你 给新创建的工程命名。它会提示你输入 groupId、 artifactId，以及 package name。

2、运行quickstart脚本

```
$ curl https://flink.apache.org/q/quickstart-scala.sh | bash
```

上面两个命令（使用Maven创建的时候需要输入artifactId、groupId等相关信息）运行完之后都会产生类似于以下的目录结构：

```
.
├── pom.xml
```

```
└── src
    └── main
        ├── resources
        │   └── log4j.properties
        └── scala
            └── com
                └── iteblog
                    ├── Job.scala
                    ├── SocketTextStreamWordCount.scala
                    └── WordCount.scala

6 directories, 5 files
```

这里使用Maven创建Scala工程的方式和之前《Flink快速上手之Java API使用》文章里面的方式很类似。

检查工程（Inspect Project）

运行完上面的脚本或者命令之后，在你的工作目录下将会出现一个新的目录。如果你使用了 curl 建立工程, 这个目录就是 quickstart。 否则， 就以你输入的 artifactId 命名。

这个示例工程是一个包含三个类的 Maven工程。 Job 是一个基本的框架程序， SocketTextStreamWordCount和WordCount是一个简单的单词次数计算示例。需要注意的是，这三个类的main 方法都允许你在开发/测试模式下启动Flink。

推荐 把这个工程导入你的 IDE 进行测试和开发。 如果用的是 Eclipse， 你需要从常用的 Eclipse 更新站点上下载并安装以下插件:

1、Eclipse 4.x
    Scala IDE
    m2eclipse-scala
    Build Helper Maven Plugin
2、Eclipse 3.7
    Scala IDE
    m2eclipse-scala
    Build Helper Maven Plugin

IntelliJ IDE 也支持 Maven 并提供了一个用于 Scala 开发的插件。

编译工程

如果想要 构建你的工程， 进入工程目录并输入 mvn clean package -Pbuild-jar 命令。 你会找到一个jar包: target/your-artifact-id-1.0-SNAPSHOT.jar , 它可以在任意 Flink 集群上运行。还有一个 fat-jar, target/your-artifact-id-1.0-SNAPSHOT-flink-fat-jar.jar ，包含了所有添加到 Maven 工程的依赖。

下一步Next Steps

开始编写我们自己的程序

Quickstart 工程包含了一个 WordCount 的实现，也就是大数据处理系统的 Hello World。WordCount 的目标是计算文本中单词出现的频率。比如： 单词 “the” 或者 “house” 在所有的Wikipedia文本中出现了多少次。

样本输入

```
big data is big
```

样本输出

```
big 2
data 1
is 1
```

下面的代码就是 Quickstart 工程的 WordCount 实现，它使用两种操作(FlatMap 和 Reduce)处理了一些文本，并且在标准输出中打印了单词的计数结果。

```
object WordCountJob {
  def main(args: Array[String]) {

    // set up the execution environment
    val env = ExecutionEnvironment.getExecutionEnvironment

    // get input data
    val text = env.fromElements("To be, or not to be,--that is the question:--",
      "Whether 'tis nobler in the mind to suffer",
      "The slings and arrows of outrageous fortune",
      "Or to take arms against a sea of troubles,")

    val counts = text.flatMap { _.toLowerCase.split("\W\W\W+") }
```

```
      .map { (_, 1) }
      .groupBy(0)
      .sum(1)

    // emit result
    counts.print()
  }
}
```

请到https://github.com/apache/flink/blob/master//flink-examples/flink-scala-examples/src/main/scala/org/apache/flink/examples/scala/wordcount/WordCount.scala里面查看完整的代码。