亚太研究院

Spark 官方文档翻译

sbt-assembly

翻译者　俞杭军

Spark 官方文档翻译团成员

前 言

世界上第一个Spark 1.1.0 中文文档问世了！

伴随着大数据相关技术和产业的逐步成熟，继Hadoop之后，Spark技术以集大成的无可比拟的优势，发展迅速，将成为替代Hadoop的下一代云计算、大数据核心技术。

Spark是当今大数据领域最活跃最热门的高效大数据通用计算平台，基于RDD，Spark成功的构建起了一体化、多元化的大数据处理体系，在“One Stack to rule them all”思想的引领下，Spark成功的使用Spark SQL、Spark Streaming、MLLib、GraphX近乎完美的解决了大数据中Batch Processing、Streaming Processing、Ad-hoc Query等三大核心问题，更为美妙的是在Spark中Spark SQL、Spark Streaming、MLLib、GraphX四大子框架和库之间可以无缝的共享数据和操作，这是当今任何大数据平台都无可匹敌的优势。

在实际的生产环境中，世界上已经出现很多一千个以上节点的Spark集群，以eBay为例，eBay的Spark集群节点已经超过2000个，Yahoo 等公司也在大规模的使用Spark，国内的淘宝、腾讯、百度、网易、京东、华为、大众点评、优酷土豆等也在生产环境下深度使用Spark。2014 Spark Summit上的信息，Spark已经获得世界 20 家顶级公司的支持，这些公司中包括Intel、IBM等，同时更重要的是包括了最大的四个Hadoop发行商，都提供了对Spark非常强有力的支持。

与Spark火爆程度形成鲜明对比的是Spark人才的严重稀缺，这一情况在中国尤其严重，这种人才的稀缺，一方面是由于Spark技术在 2013、2014 年才在国内的一些大型企业里面被逐步应用，另一方面是由于匮乏Spark相关的中文资料和系统化的培训。为此，Spark亚太研究院和 51CTO联合推出了“Spark亚太研究院决胜大数据时代 100 期公益大讲堂”，来推动Spark技术在国内的普及及落地。

具体视频信息请参考 http://edu.51cto.com/course/course_id-1659.html

与此同时，为了向Spark学习者提供更为丰富的学习资料，Spark亚太研究院发起并号召，结合网络社区的力量构建了Spark中文文档专家翻译团队，历经 1 个月左右的艰苦努力和反复修改，Spark中文文档V1.1 终于完成。尤其值得一提的是，在此次中文文档的翻译期间，Spark官方团队发布了Spark 1.1.0 版本，为了让学习者了解到最新的内容，Spark中文文档专家翻译团队主动提出基于最新的Spark 1.1.0 版本，更新了所有已完成的翻译内容，在此，我谨代表Spark亚太研究院及广大Spark学习爱好者向专家翻译团队所有成员热情而专业的工作致以深刻的敬意！

当然，作为世界上第一份相对系统的Spark中文文档，不足之处在所难免，大家有任何建议或者意见都可以发邮件到marketing@sparkinchina.com ;同时如果您想加入Spark中文文档翻译团队，也请发邮件到marketing@sparkinchina.com进行申请；Spark中文文档的翻译是一个持续更新的、不断版本迭代的过程，我们会尽全力给大家提供更高质量的Spark中文文档翻译。

最后，也是最重要的，请允许我荣幸的介绍一下我们的Spark中文文档第一个版本翻译的专家团队成员，他们分别是（排名不分先后）:

- 傅智勇, 《快速开始(v1.1.0)》（和唐海东翻译的是同一主题，大家可以对比参考）
- 吴洪泽,《Spark机器学习库 (v1.1.0)》（其中聚类和降维部分是蔡立宇翻译）
- 武扬，《在Yarn上运行Spark (v1.1.0)》《Spark 调优(v1.1.0)》
- 徐骄，《Spark配置(v1.1.0)》《Spark SQL编程指南(v1.1.0)》（Spark SQL和韩保礼翻译的是同一主题，大家可以对比参考）
- 蔡立宇，《Bagel 编程指南(v1.1.0)》
- harli，《Spark 编程指南 (v1.1.0)》
- 吴卓华，《图计算编程指南(1.1.0)》
- 樊登贵，《EC2(v1.1.0)》《Mesos(v1.1.0)》
- 韩保礼，《Spark SQL编程指南(v1.1.0)》（和徐骄翻译的是同一主题，大家可以对比参考）
- 颜军，《文档首页(v1.1.0)》
- Jack Niu，《Spark实时流处理编程指南(v1.1.0)》
- 俞杭军，《sbt-assembly》《使用Maven编译Spark(v1.1.0)》
- 唐海东，《快速开始(v1.1.0)》（和傅智勇翻译的是同一主题，大家可以对比参考）
- 刘亚卿，《硬件配置(v1.1.0)》《Hadoop 第三方发行版(v1.1.0)》《给Spark提交代码(v1.1.0)》
- 耿元振《集群模式概览(v1.1.0)》《监控与相关工具(v1.1.0)》《提交应用程序(v1.1.0)》
- 王庆刚，《Spark作业调度(v1.1.0)》《Spark安全(v1.1.0)》
- 徐敬丽，《Spark Standalone 模式 (v1.1.0)》

另外关于Spark API的翻译正在进行中，敬请大家关注。

Life is short, You need Spark!

Spark亚太研究院院长 王家林
2014 年 10 月

Spark 亚太研究院决胜大数据时代 100 期公益大讲堂

简　介

作为下一代云计算的核心技术，Spark性能超Hadoop百倍，算法实现仅有其 1/10 或 1/100,是可以革命Hadoop的目前唯一替代者，能够做Hadoop做的一切事情，同时速度比Hadoop快了 100 倍以上。目前Spark已经构建了自己的整个大数据处理生态系统，国外一些大型互联网公司已经部署了Spark。甚至连Hadoop的早期主要贡献者Yahoo现在也在多个项目中部署使用Spark；国内的淘宝、优酷土豆、网易、Baidu、腾讯、皮皮网等已经使用Spark技术用于自己的商业生产系统中，国内外的应用开始越来越广泛。Spark正在逐渐走向成熟，并在这个领域扮演更加重要的角色，刚刚结束的 2014 Spark Summit上的信息，Spark已经获得世界 20 家顶级公司的支持，这些公司中包括Intel、IBM等，同时更重要的是包括了最大的四个Hadoop发行商都提供了对非常强有力的支持Spark的支持.

鉴于Spark的巨大价值和潜力，同时由于国内极度缺乏Spark人才，Spark亚太研究院在完成了对Spark源码的彻底研究的同时，不断在实际环境中使用Spark的各种特性的基础之上，推出了Spark亚太研究院决胜大数据时代 100 期公益大讲堂，希望能够帮助大家了解Spark的技术。同时，对Spark人才培养有近一步需求的企业和个人，我们将以公开课和企业内训的方式，来帮助大家进行Spark技能的提升。同样，我们也为企业提供一体化的顾问式服务及Spark一站式项目解决方案和实施方案。

Spark亚太研究院决胜大数据时代 100 期公益大讲堂是国内第一个Spark课程免费线上讲座，每周一期，从 7 月份起，每周四晚 20:00-21:30，与大家不见不散！老师将就Spark内核剖析、源码解读、性能优化及商业实战案例等精彩内容与大家分享，干货不容错过！

时间：从 7 月份起，每周一期，每周四晚 20:00-21:30
形式：腾讯课堂在线直播
学习条件：对云计算大数据感兴趣的技术人员
课程学习地址：http://edu.51cto.com/course/course_id-1659.html

sbt-assembly

（翻译者：俞杭军）

sbt/sbt-assembly，原文档链接：

https://github.com/sbt/sbt-assembly

目录

1. sbt-assembly

Deploy fat JARs. Restart processes.（这句不翻译）

sbt-assembly 是一个原本从 codahale's assembly-sbt 独立出来的 sbt 插件，笔者猜测其灵感来自于 Maven 编译插件。它的目标是尽可能简单：为自建的工程创建带所有依赖库的 fat JAR 包

1.1 准备

- sbt
- 具有期望实现简单部署的强烈欲望

1.2 Reporting Issues & Contributing

Before you email me, please read Issue Reporting Guideline carefully. Twice. (Don't email me)

（本段不翻译）

1.3 安装

1.3.1 使用插件

sbt0.13 版本增加 sbt-assembly作为依赖库存放在 project/assembly.sbt:
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.11.2")
sbt 0.12，请参见 sbt-assembly 0.9.2.
(可能需要检查这个工程标志（project's tags）了解最新发布的版本的情况)

1.4 使用

1.4.1 在工程中使用插件（添加 assembly Task）

首先确保已经将插件添加到编译环境中（不管是发行版版本还是 Git 源代码）
将 assembly.sbt 放在根目录下

```
import AssemblyKeys._ // put this at the top of the file
assemblySettings
```

```
// your assembly settings here
```

1.4.2 在多个工程中使用插件（添加 assembly 任务）

如果使用多个工程编译配置文件 build.sbt:

```
import AssemblyKeys._
lazy val buildSettings = Seq(
  version := "0.1-SNAPSHOT",
  organization := "com.example",
  scalaVersion := "2.10.1"
)
val app = (project in file("app")).
  settings(buildSettings: _*).
  settings(assemblySettings: _*).
  settings(
    // your settings here
  )
```

1.4.3 在多个工程中使用插件 build.scala

如果使用多个工程编译配置文件 build.scala:

```
import sbt._
import Keys._
import sbtassembly.Plugin._
import AssemblyKeys._
object Builds extends Build {
  lazy val buildSettings = Defaults.defaultSettings ++ Seq(
    version := "0.1-SNAPSHOT",
    organization := "com.example",
    scalaVersion := "2.10.1"
  )
  lazy val app = Project("app", file("app"),
    settings = buildSettings ++ assemblySettings) settings(
      // your settings here
    )
}
```

1.4.4 编译集成任务(assembly task)

新的 assembly task 非常令人震撼，可以用来编译工程、运行测试，然后将 class 文件和所有依赖库打包到一个 JAR 文:target/scala_X.X.X/projectname-assembly-X.X.X.jar.

```
> assembly
```

如果在 build.sbt 文件中指定 mainClass in assembly(或只是自动检测代码是否改变)，就能自动生成一个完整的可执行 JAR 包，并且可以直接运行。

以下是能够重新连接 assembly task 的关键字列表

注：任何定制化参数必须添加到 assemblySettings 之后

jarName	test	mainClass
outputPath	mergeStrategy	assemblyOption
excludedJars	assembledMappings	

例如：在 build.sbt 文件中可以通过下面的语句重新设置 jar 包名：

```
jarName in assembly := "something.jar"
```

assembly 过程中要忽略测试的话，可以：

```
test in assembly := {}
```

设置显式的主函数，可以：

```
mainClass in assembly := Some("com.example.Main")
```

1.4.5 合并策略(Merge Strategy)

如果多个文件共享同一个相对路径(例如：文件名为 application.conf 的资源文件存在于多个依赖包中)，否则默认策略是校验所有备选策略是否有相同的内容和错误。这种方式能够通过路径前置(a per-path basis)的设置方式使用下文中某个内建策略(built-in strategies)中或自定义策略：

- MergeStrategy.deduplicate :即上文描述的默认选项
- MergeStrategy.first:提取classpath下第一个匹配的文件
- MergeStrategy.last：提取最后匹配的文件
- MergeStrategy.singleOrError：提取冲突时的错误消息(bails out with an error message on conflict)
- MergeStrategy.concat 仅连接(concatenate)所有匹配的文件并包括策略执行结果
- MergeStrategy.filterDistinctLines 同样也是连接,但是忽略过程中的重复文件
- MergeStrategy.rename 对jar包中的文件重命名
- MergeStrategy.discard 仅丢弃匹配的文件

对合并策略进行映射方式命名是通过设置 assembly-merge-strategy实现的，其可以通过下列方式扩展：(The mapping of path names to merge strategies is done via the setting assembly-merge-strategy which can be augmented as follows:)

```
mergeStrategy in assembly <<= (mergeStrategy in assembly) { (old) =>
  {
    case PathList("javax", "servlet", xs @ _*)           => MergeStrategy.first
    case PathList(ps @ _*) if ps.last endsWith ".html" => MergeStrategy.first
    case "application.conf" => MergeStrategy.concat
    case "unwanted.txt"       => MergeStrategy.discard
    case x => old(x)
  }
}
```

注意：

- mergeStrategy in assembly 需要通过方法实现，而不能直接设置 mergeStrategy in assembly := MergeStrategy.first!
- 某些文件必须丢弃或重命名，否则尽量避免破坏压缩文件(主要是由于重复的文件名)或经授权的许可证。如上文中的模式匹配案例默认处理(mergeStrategy in assembly)(原文：Delegate default handling to (mergeStrategy in assembly) as the above pattern matching example.)

顺便提一下，上文提到使用 PathList(...)进行 首次模式匹配，主要是关于如何从第一个jar中抽取javax/servlet/*。如果默认的MergeStrategy.deduplicate参数不起作用，那么很有可能意味着库依赖层级 (dependency graph) 中的某些类库有多个版本。真正的解决

方式是修复库依赖层级（dependency graph）。可以使用 MergeStrategy.first参数作为变通方法，但如果出现ClassNotFoundException也不必惊讶。

以下是默认配置：

```
  val defaultMergeStrategy: String => MergeStrategy = {
    case x if Assembly.isConfigFile(x) =>
      MergeStrategy.concat
    case PathList(ps @ _*) if Assembly.isReadme(ps.last) ||
Assembly.isLicenseFile(ps.last) =>
      MergeStrategy.rename
    case PathList("META-INF", xs @ _*) =>
      (xs map {_.toLowerCase}) match {
        case ("manifest.mf" :: Nil) | ("index.list" :: Nil) | ("dependencies" :: Nil) =>
          MergeStrategy.discard
        case ps @ (x :: xs) if ps.last.endsWith(".sf") || ps.last.endsWith(".dsa") =>
          MergeStrategy.discard
        case "plexus" :: xs =>
          MergeStrategy.discard
        case "services" :: xs =>
          MergeStrategy.filterDistinctLines
        case ("spring.schemas" :: Nil) | ("spring.handlers" :: Nil) =>
          MergeStrategy.filterDistinctLines
        case _ => MergeStrategy.deduplicate
      }
    case _ => MergeStrategy.deduplicate
  }
```

自定义 MergeStrategyS参数能够找出某个来自使用基于sbtassembly.AssemblyUtils的thesourceOfFileForMerge方法的特殊文件， 该方法将临时文件夹和其中一个文件作为参数传递给策略。

1.5 排除 JAR 包及文件(Excluding JARs and files)

如果要让 sbt-assembly 忽略 JAR 包，可能错误地执行 assembly task 导致从工程的 classpath 搜集依赖包。可以尝试先修正 classpath。

1.5.1 % "provided" 配置

如果试着排除已经是容器一部分的 JAR 文件（如 Spark），考虑通过"provided"配置项限定依赖库的范围：

```
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % "0.8.0-incubating" % "provided",
 "org.apache.hadoop" % "hadoop-client" % "2.0.0-cdh4.4.0" % "provided"
```

Maven 是这样定义"provided"的：

这和 compile 很相似，但暗示着预期 JDK 或者容器在运行时提供依赖库。例如当为 J2EE 构建一个 WEB 应用时，需要设置 Servlet API 和相关的 JavaEE API 来限定 provided 范围，因为 web 容器提供了这些类。这个限定范围只在编译和测试的 classpath 有效，而且不是间接的。

这个依赖包将成为编译和测试的一部分，但是运行时会排除在外。如果Spark用户想把 "provided" 的依赖库添加到运行阶段(run)。 @douglaz在StackOverflow上提出一个一行代码的解决方案：sbt: how can I add "provided" dependencies back to run/test tasks' classpath?:

```
run in Compile <<= Defaults.runTask(fullClasspath in Compile, mainClass in
(Compile, run), runner in (Compile, run))
```

1.5.2 排除特定的间接依赖(Exclude specific transitive deps)

由于合并带来的冲突，你可能会想到排除 JAR 文件。*.class 文件的合并冲突意味着 classpath 有问题，通常因非模块化的 JAR 包文件或 SLF4J 所导致，而不是集成的问题。当尝试创建一个包含 Spark 的 fat JAR 包，将发生以下错误：

```
[error] (*:assembly) deduplicate: different file contents found in the following:
[error]
/Users/foo/.ivy2/cache/org.eclipse.jetty.orbit/javax.servlet/orbits/javax.servlet-2.5.
0.v201103041518.jar:javax/servlet/SingleThreadModel.class
[error]
/Users/foo/.ivy2/cache/org.mortbay.jetty/servlet-api/jars/servlet-api-2.5-20081211
.jar:javax/servlet/SingleThreadModel.class
```

上文案例中两个独立的JAR文件 javax.servlet-2.5.0.v201103041518.jar 和 servlet-api-2.5-20081211.jar 都定义了 javax/servlet/SingleThreadModel.class! 同样地，common-beanutils 和 EsotericSoftware/minlog 也会冲突。下文展示了如何排除指定的间接依赖

```
libraryDependencies ++= Seq(
  ("org.apache.spark" %% "spark-core" % "0.8.0-incubating").
    exclude("org.mortbay.jetty", "servlet-api").
    exclude("commons-beanutils", "commons-beanutils-core").
    exclude("commons-collections", "commons-collections").
    exclude("commons-collections", "commons-collections").
    exclude("com.esotericsoftware.minlog", "minlog")
)
```

参考sbt：排除间接依赖（Exclude Transitive Dependencies） 获取更多细节。

有时需要仔细观察来解决需要排除的间接依赖问题。Play框架自带 dist Task，所以 assembly 不是必须的，但可以假设我们希望运行 assembly。通过引入 signpost-commonshttp4 包，从而引入commons-logging。这会和重新实现了日志API的jcl-over-slf4j包冲突。只要通过build.sbt和 playScalaSettings添加依赖库 ，就可以有一种变通方法：

```
libraryDependencies ~= { _ map {
  case m if m.organization == "com.typesafe.play" =>
    m.exclude("commons-logging", "commons-logging").
      exclude("com.typesafe.play", "sbt-link")
  case m => m
}}
```

1.5.3 排除指定的文件(Excluding specific files)

要排除指定文件，可以自定义合并策略：

```
mergeStrategy in assembly <<= (mergeStrategy in assembly) { (old) =>
  {
    case PathList("about.html") => MergeStrategy.rename
    case x => old(x)
  }
```

```
}
```

1.5.4 排除 Scala 库、自建工程或者依赖 JAR 包(Excluding Scala library, your project, or deps JARs)

要排除 Scala 类库，可以：

```
assemblyOption in assembly ~= { _.copy(includeScala = false) }
```

排除从主源文件生成的类文件和内部依赖，可以：

```
assemblyOption in assembly ~= { _.copy(includeBin = false) }
```

编译只包含外部依赖的 JAR 文件，输入：

```
> assemblyPackageDependency
```

注意：如果使用-jar选项，则将忽略-cp，因此如果有多个JAR文件则必须使用-cp并传递给主类： java -cp "jar1.jar:jar2.jar" Main

1.5.5 被排除在外的 JAR 包(excludedJars)

如果所有的努力都失败了，还有一个方法来排除 JAR 包

```
excludedJars in assembly <<= (fullClasspath in assembly) map { cp =>
  cp filter {_.data.getName == "compile-0.1.0.jar"}
}
```

1.6 其他事项(Other Things)

1.6.1 内容哈希化（Content hash）

```
同样可以添加 SHA-1 指纹加密算法到编辑文件文件名，例如如果有必要部署依赖的类库，用它来判断文件是否被改变：
assemblyOption in packageDependency ~= { _.copy(appendContentHash = true) }
```

1.6.2 缓存(Caching)

考虑到性能原因，默认每一步解压缩任何依赖库的 JAR 文件到磁盘操作将会被缓存。这个特征可以通过以下设置禁用：

```
assemblyOption in assembly ~= { _.copy(cacheUnzip = false) }
```

另外，仅当输入文件改变导致 fat JAR 的时间戳也会变化，fat JAR 则会被缓存。这个特征需要检测所有*.class 文件的 SHA-1 hash 值 以及所有依赖库的*.jar 文件的 hash 值。如果有大量的 class 文件会非常耗时，尽管 hash 的是 JAR 文件，而不是文件的内容，hash 速度最近（版本中）有所提升。这个特征可以通过以下设置禁用：

```
assemblyOption in assembly ~= { _.copy(cacheOutput = false) }
```

1.6.3 发布（不推荐）Publishing (Not Recommended)

不建议对外发布 far JAR，因为非模块化的 JAR 包会引起很多问题。其中一个可能考虑是非模块化 JAR 包很方便但是当涉及复杂应用时将会令人头痛。如果你仍然希望通过执行 publish 任务发布集成 jar 包和其他 jar 包 那么就添加一个 assembly 标识符(或者其他)：

```
artifact in (Compile, assembly) ~= { art =>
  art.copy(`classifier` = Some("assembly"))
}
addArtifact(artifact in (Compile, assembly), assembly)
```

1.6.4 Prepending Shebang

可以通过以下方式将 shell 脚本添加到 fat jar 的开头：
译注：prepend 通常是指在对象的开头插入指定内容

```
assemblyOption in assembly ~= { _.copy(prependShellScript =
Some(defaultShellScript)) }
jarName in assembly := { s"${name.value}-${version.value}" }
```

下列语句将以下 shell 脚本添加到 jar 包的头部

```
#!/usr/bin/env sh
exec java -jar "$0" "$@"
```

Spark 亚太研究院

Spark 亚太研究院是中国最专业的一站式大数据 Spark 解决方案供应商和高品质大数据企业级完整培训与服务供应商，以帮助企业规划、架构、部署、开发、培训和使用 Spark 为核心，同时提供 Spark 源码研究和应用技术训练。针对具体 Spark 项目，提供完整而彻底的解决方案。包括 Spark 一站式项目解决方案、Spark 一站式项目实施方案及 Spark 一体化顾问服务。

官网： www.sparkinchina.com

近期活动

- 2014 年亚太地区规格最高的 Spark 技术盛会！
- 面向大数据、云计算开发者、技术爱好者的饕餮盛宴！
- 云集国内外 Spark 技术领军人物及灵魂人物！
- 技术交流、应用分享、源码研究、商业案例探询！

时间：2014 年 12 月 6-7 日

地点：北京珠三角万豪酒店

Spark亚太峰会网址：http://www.sparkinchina.com/meeting/2014yt/default.asp

- 如果你是对 Spark 有浓厚兴趣的初学者，在这里你会有绝佳的入门和实践机会！
- 如果你是 Spark 的应用高手，在这里以“武”会友，和技术大牛们尽情切磋！
- 如果你是对 Spark 有深入独特见解的专家，在这里可以尽情展现你的才华！

比赛时间：

2014 年 9 月 30 日—12 月 3 日

Spark开发者大赛网址：http://www.sparkinchina.com/meeting/2014yt/dhhd.asp

视频课程：

《大数据 Spark 实战高手之路》 国内第一个 Spark 视频系列课程

从零起步，分阶段无任何障碍逐步掌握大数据统一计算平台 Spark，从 Spark 框架编写和开发语言 Scala 开始，到 Spark 企业级开发，再到 Spark 框架源码解析、Spark 与 Hadoop 的融合、商业案例和企业面试，一次性彻底掌握 Spark，成为云计算大数据时代的幸运儿和弄潮儿，笑傲大数据职场和人生！

- 第一阶段：熟练的掌握 Scala 语言
  课程学习地址：http://edu.51cto.com/pack/view/id-124.html
- 第二阶段：精通 Spark 平台本身提供给开发者 API
  课程学习地址：http://edu.51cto.com/pack/view/id-146.html
- 第三阶段：精通 Spark 内核
  课程学习地址：http://edu.51cto.com/pack/view/id-148.html
- 第四阶段：掌握基于 Spark 上的核心框架的使用
  课程学习地址：http://edu.51cto.com/pack/view/id-149.html
- 第五阶段：商业级别大数据中心黄金组合：Hadoop+ Spark
  课程学习地址：http://edu.51cto.com/pack/view/id-150.html
- 第六阶段：Spark 源码完整解析和系统定制
  课程学习地址：http://edu.51cto.com/pack/view/id-151.html

近期公开课：

《决胜大数据时代：Hadoop、Yarn、Spark 企业级最佳实践》

集大数据领域最核心三大技术：Hadoop 方向 50%：掌握生产环境下、源码级别下的 Hadoop 经验，解决性能、集群难点问题；Yarn 方向 20%：掌握最佳的分布式集群资源管理框架，能够轻松使用 Yarn 管理 Hadoop、Spark 等；Spark 方向 30%：未来统一的大数据框架平台，剖析 Spark 架构、内核等核心技术，对未来转向 SPARK 技术，做好技术储备。课程内容落地性强，即解决当下问题，又有助于驾驭未来。

开课时间： 2014 年 10 月 26-28 日北京、2014 年 11 月 1-3 日深圳

咨询电话：4006-998-758

QQ 交流群： 1 群：317540673 （已满）
2 群：297931500

微信公众号：spark-china