**Jigsaw项目简介**

本文通过收集一些与Jigsaw项目（Project Jigsaw）相关的报道，介绍了这个Jigsaw项目是如何随着Java 7（JDK7）越来越近而逐渐清晰。Java的模块化在几年间由OSGi建立了规则，现在，是否将由Jigsaw取而代之呢？

本文收集了一些与Jigsaw项目（Project Jigsaw）相关的报道，旨在让读者们更加清楚的了解这个**[Java 7](http://developer.51cto.com/art/201001/175830.htm)**中的新玩意儿是做什么用的。

**在Jigsaw项目成为OpenJDK的一部分时的声明**

为了在接下来的几年中能够模块化JDK 7，同时也为了更好地宣传JSR 294的工作，Sun打算不久之后在OpenJDK社区中创建Jigsaw项目。

该努力必然会创建一个简单、低层次的模块系统，其设计将聚焦于模块化JDK这个目标。开发者可以在自己的代码中使用该模块系统，Sun也会全力支持该系统，但它并不会成为官方Java SE 7平台规范的一部分，也不一定会得到其他SE 7实现的支持。

如果未来的Java SE平台包含了特定的模块系统，那么Sun将提供一种方式来移植Jigsaw模块以使之符合相应的标准。同时，我们还会积极地寻求与其他模块系统交互的方式，尤其是OSGi。

模块化JDK这个目标应该是可以实现的，尤其是Apache Harmony已经证明了JDK是可以被模块化的（使用OSGi）。然而这个目标却因另一个原因而变得更加有趣：它将在JCP外实现。

由于缺少可视化的发展过程，再加上闭门造车的实现，对于大多数人来说很难参与进JSR 277，更别提对其献计献策了。通常这会导致标准的误入歧途，使得标准过于关注实现的细节而忽视了通用性。不仅如此，这还会导致标准成为事后诸葛亮——想解决问题时（例如奇数的版本号）却发现已经太迟了。根据声明所述，JSR277还处于“领导地位”，但它实际已死。

从JSR277中分离出的JSR294（以前可是个庞然大物，不过现在已经模块化了）是个很大的进步；它将引入一些语言上的变化，这会使得模块化的实现独立于任何特定的模块化进程。

从声明中看到的激动人心的事情还有OSGi已经成为模块化的事实上的标准了，这种合作是件好事。但这不一定意味着它将基于OSGi实现，事实上，它只是建议最好与OSGi而不是其他方式合作：

JSR 277的JAM模块系统并不适合作为模块化JDK的基础，那么OSGi框架如何呢？该模块系统非常成熟、稳定和健壮。其核心甚至已经在Java虚拟机中实 现出来了，也就是Apache Harmony。OSGi并没有与Java语言集成，相反，它构建在Java SE平台之上而不是其中。

这最后的问题也能解决。Sun现在计划直接与OSGi联盟合作以便OSGi框架的未来版本能充分利用JSR 294的特性进而能和语言集成的更加紧密。

**08年底，Java 7新特性展望**

现在的Java程序员，或者说所有语言的程序员，都面临着日益增多的开源和商业类库，往往要花费很长时间来管理其依赖关系。今天的一个普通企业应用程序往往要依赖数十个外部JAR文件，其本身往往就能包含数十个不同团队开发的更小内部工程。我们一直在坚持寻找更好的方式来管理日益复杂的依赖关系，以使我们的开发更具重用性，部署更加完整。现在出现了越来越多的类似Maven的依赖关系管理系统，以及诸如OSGi之类的运行时部署系统，这一点正是反应了这种需求的增长趋势。

在Java SE 7发展初期，两个重要的JSR曾经试图解决依赖关系管理问题，分别是JSR 294：Java编程语言中的改进模块性支持(Improved Modularity Support in the Java Programming Language)和JSR 277：Java模块系统(Java Module System)，两者分别关注Java模块概念的开发和部署方面。一个模块(module)就是多个实现相同目标且相互存在联系的类的集合，与JAR类似，但是，根据开发和部署的需要，一个模块的范围可以是一个JAR的一部分，也可以是几个JAR的集合。在2008年中期，JSR 294被简化并合并到JSR 277中，以便同一个专家组能够先后研究这两个方面。

在2008年12月份，Sun再次重新审视这一计划，宣布在OpenJDK社区中创建Jigsaw项目，以在明年实现JDK 7模块化。JSR 277和Java模块系统的研究将被放到Java SE 7推出之后进行，而JSR 294将被重新恢复研究。Sun已经声明了此举的意图是，与OSGi联盟更紧密的配合，以便JSR 294模块可以被OSGi所使用。

在Java SE 7中有一个问题将得到解答，即Sun将如何来使用module关键字，它是最初的JSR 294中的一个重要概念，预计将包含在下一平台版本中。

假定有一个名为Flapjack的项目由几个Java包(package)组成，该项目包含在基包(base package)中的一个public APIs，和实现这个API的几个内部包：

·org.flapjack - public API classes

·org.flapjack.impl - 实现类

·org.flapjack.util - 实用类

在Java SE 6中，如果你需要在基包中放置一个工厂类(factory class)，以实例化内部执行包中的API类，你需要将这个实现类设为public，这样它们才可以从API包中被看到。由于跨越了不同的包，没有办法既允许API以factory方法对类实例化，又不允许外部类直接执行它。

JSR 294模块将允许你声明整个包集合为一个模块，你只需要在源程序中加入以下一个新的声明：

module org.flapjack;你可以将这个声明加在你的项目中每一个源程序文件中，也可以将其增加到package-info.java文件中，然后一次将其应用到整个包。虽然module是一个新关键字，它是一个“限制性”关键字，只有在特定位置时才被作为关键字来处理;因此，它可以在任何其它地方作为普通Java标识符来使用。这使得它扩展了语言的功能，同时又保持了其向后兼容性。

除了新的声明外，你还可以把module关键字当作一个新的可见性修饰符使用，你可以用它来定义一个类，使其仅对同一个模块中的其它类可见，Listing 1演示了module关键字的这种用法。

1. Listing 1
2. module org.flapjack;
3. **package** org.flapjack.impl;
4. **import** org.flapjack.Flapjack;
5. module **class** FlapjackImpl **implements** Flapjack {
6. }

最后，你可以定义一个新的module-info.java伪类，使用元数据来注释该模块，增加诸如版本、主类、导入的依赖模块、导出资源和许多其它预定义或特定的模块注释等。值得注意的是，与现有的package-info.java文件一样，这个新的module-info.java文件使用了一个无效Java源文件名称，可以避免与已经存在的文件可能发生冲突。

在编译时，JSR 294让你可以使用javac来编译你的类。至于在JVM中，Jigsaw项目将如何规定模块的组成、加载和验证，尚需拭目以待。

**09年5月，JavaOne前夕对Jigsaw的展望**

……这个新的模块系统的第一部分就是JSR-294，即所谓的超级包。也正是这个规范阐释了Java语言的模块部分的概念。

JSR-294引入了新的可见性关键字“module”。如果一个成员拥有这样的可见性，那就意味着它只对同一模块中的成员可见。它会创建一个内部的 API，只有模块本身能调用。就此看来，“public”关键字应当只在声明一个公共的API时才用。而在其他情况下，应当使用“module”或者有更多限制的可见性关键字。当然，一旦语言中有了“module”关键字，那么模块之间的可见性限制将会由编译器来负责检查。

JSR-294也允许定义依赖性。你可以在某个给定版本中，定义某个模块依赖于另一模块。比如：

1. //org/netbeans/core/module-info.java
2. @Version("7.0")
3. @ImportModule(name="java.se.core", version="1.7+")
4. module org.netbeans.core;

最后一句表明“org.netbeans.core”模块依赖“java.se.core”的1.7版本或者更高。这类似于Maven的依赖性或者 OSGi的导入。你也可以暂时不要管这些语法，因为将来语法可能会另有变化。重要的是，这儿的依赖是在module-info.java中定义的，会被编译成class文件。而OSGi中，依赖则是在普通的文本文件中定义的。

Jigsaw项目是这个新模块系统的第二部分。我预计它会是JSR-294特定于Sun的实现，也会是Sun JDK的模块化实现。既然创建完整的JDK模块化是有必要的，Sun就希望把标准库分装成模块。这直接简化了JRE中的内容整合。整个JRE除了Swing之外的所有内容因此都能够在移动设备上运行。它还有可能为语言引入新的标准API，而无需再等待整个平台的新版本发布。目前看起来，这个项目绝对有希望实现。

但我对此还有个担忧，那就是，专有的Jigsaw和JSR标准之间的关系并不清晰，正如Mark Reinhold所说的：

对Jigsaw的投入无疑会创建出一个简单的、低层次的模块系统，它的设计会严格地朝着JDK模块化的目标而发展。开发人员可以把这个模块系统运用到他们的代码中去，Sun对这个模块系统也会是绝对的支持，但它不会是Java SE 7平台规范的官方部分，也可能不会被其他SE 7实现所支持。

这段话说的不是很清楚，当中有很多疑问。他的意思是说创建的模块只能在Sun JRE中运行吗?还是想说，如果开发者写了“@ImportModule(name="java.se.core", version="1.7+")”，那么这个模块只能在Sun JRE中运行，而不能在IBM JRE环境中运行吗?或者他的意思是不是说Sun会以某种方式把它的JRE分割成许多模块，而Oracle会选择另外的方式去分割吗?(译者注：至少现在看来，不太会有这样的可能了，因为Oracle刚刚收购了Sun)。我们希望都不是，因为还有“编写一次，到处运行”的原则。

细究起来问题更多。我们并不清楚Jigsaw项目的主要目标是什么。据项目本身所宣布的主要目标来看，它要实现的是Sun JRE的模块化，但如果纯粹是要实现模块化的话，就不需要对语言做任何改变。Sun可以对JRE进行模块化，而不修改Java语言本身。

这些语言上的变化会不会成为Sun JRE模块化带来的副产品?如果是，那就彻底错了！语言变化必须是一等公民，而不是专属的副产品。