JEP 200: The Modular JDK

JEP 200：模块化 JDK

Owner 所有者	Mark Reinhold 马克·莱因霍尔德
Type 类型	Feature 特征
Scope 范围	SE
Status 地位	Closed / Delivered 已关闭 / 已交付
Release 释放	9
JSR	376
Discussion 讨论	jigsaw dash dev at openjdk dot java dot net
Effort 努力	XL
Duration 期间	XL
Depends 取决于	JEP 201: Modular Source Code JEP 201：模块化源代码 JEP 220: Modular Run-Time Images JEP 220：模块化运行时映像 JEP 261: Module System JEP 261： 模块系统
Reviewed by 校订者	Alan Bateman, Alex Buckley, Paul Sandoz 艾伦·贝特曼、亚历克斯·巴克利、保罗·山德士
Endorsed by 通过	Brian Goetz 布莱恩·戈茨
Created 创建	2014/07/22 14:08
Updated 更新	2017/09/21 14:30
Issue 问题	8051618

Summary 总结

Use the Java Platform Module System, specified by JSR 376 and implemented by JEP 261, to modularize the JDK.

使用由 JSR 376 指定并由 JEP 261 实现的 Java 平台模块系统来模块化 JDK。

Goals 目标

Divide the JDK into a set of modules that can be combined at compile time, build time, and run time into a variety of configurations including, but not limited to:

将 JDK 划分为一组模块，这些模块可以在编译时、构建时和运行时组合成各种配置，包括但不限于：

- Configurations corresponding to the full Java SE Platform, the full JRE, and the full JDK;

  对应于完整 Java SE 平台、完整 JRE 和完整 JDK 的配置;

- Configurations roughly equivalent in content to each of the Compact Profiles defined in Java SE 8; and

  配置在内容上大致相当于每个 Java SE 8 中定义的 Compact Profile;和

- Custom configurations which contain only a specified set of modules possibly augmented by external library and application modules, and the modules transitively required by all of these modules.

  自定义配置，其中仅包含一组指定的模块，这些模块可能由外部库和应用程序模块扩充，以及所有这些模块传递所需的模块。

The definition of the modular structure should make a clear distinction between standard modules, whose specifications are governed by the Java Community Process, and modules that are specific to the JDK. It should also distinguish

JDK (..., 23, 24, 25)

JDK (..., 23、 24、 25)

JDK Updates JDK 更新
JMC 江铃汽车
Jigsaw 拼图
Kona 科纳
Kulla 库拉
Lanai 拉奈岛
Leyden 莱 顿
Lilliput 小人 国
Locale Enhancement 区域设置增强
Loom 织布机
Memory Model Update 内存模型更新
Metropolis 都市
Multi-Language VM 多语言 VM
Nashorn 纳斯霍恩
New I/O 新 I/O
OpenJFX OpenJFX 公司
Panama 巴拿马
Penrose 彭罗斯
Port: AArch32 端口: AArch32
Port: AArch64 端口: AArch64
Port: BSD 端口: BSD
Port: Haiku 港口: Haiku
Port: Mac OS X 端口: Mac OS X
Port: MIPS 端口: MIPS
Port: Mobile 端口: 移动
Port: PowerPC/AIX 端口: PowerPC/AIX
Port: RISC-V 端口: RISC-V
Port: s390x 端口: s390x
SCTP SCTP 系列
Shenandoah 谢南多厄
Skara 人群
Sumatra 苏门答腊岛
Tsan TSA 餐厅
Valhalla 瓦尔哈拉
Verona 维罗纳
VisualVM 可视化虚拟机
Wakefield 维克菲尔德
Zero 零
ZGC 中关村

modules that are included in the Java SE Platform Specification, and thereby made mandatory in every Platform Implementation, from all other modules.

模块化结构的定义应明确区分 在标准模块之间，其规范受 Java Community Process 和特定于 JDK 的模块。它还应该将包含在 Java SE 平台规范中的模块与所有其他模块区分开来，从而在每个平台实现中都成为强制性的模块。

Motivation 赋予动机

Project Jigsaw aims to design and implement a standard module system for the Java SE Platform and to apply that system to the Platform itself, and to the JDK. Its primary goals are to make implementations of the Platform more easily scalable down to small devices, improve security and maintainability, enable improved application performance, and provide developers with better tools for programming in the large.

Jigsaw 项目旨在为 Java SE 平台设计和实现一个标准模块系统，并将该系统应用于平台本身和 JDK。其主要目标是使平台的实现更容易扩展到小型设备，提高安全性和可维护性，提高应用程序性能，并为开发人员提供更好的大型编程工具。

Description 描述

Design principles 设计原则

The modular structure of the JDK implements the following principles:

JDK 的模块化结构实现了以下原则：

1. Standard modules, whose specifications are governed by the JCP, have names starting with the string `"java."`.

   其规范由 JCP 管理的标准模块的名称以字符串 “`java.`” 开头。

2. All other modules are merely part of the JDK, and have names starting with the string `"jdk."`.

   所有其他模块只是 JDK 的一部分，其名称以字符串 “`jdk.`” 开头。

3. If a module exports a package that contains a type that contains a public or protected member that, in turn, refers to a type from some other module, then the first module must grant implied readability to the second, via `requires transitive`. (This ensures that method-invocation chaining works in the obvious way.)

   如果模块导出的包包含的类型包含 public 或 protected 成员，而该成员又引用其他模块中的类型，则第一个模块必须通过 `requires transitive` 向第二个模块授予隐式可读性。 （这确保了方法调用链接在显而易见的 方式。

4. A standard module may contain both standard and non-standard API packages. If a standard module exports a standard API package then the export may be qualified; if a standard module exports a non-standard API package then the export must be qualified. In either case, if a standard module exports a package with qualification then the export must be to some subset of the modules in the JDK. If a standard module is a Java SE module, *i.e.*, is included in the Java SE Platform Specification, then it must not export any non-SE API packages, at least not without qualification.

   标准模块可以同时包含标准和非标准 API 包。 如果标准模块导出 standard API 包，则导出可能为 qualify;如果标准 module 导出一个非标准的 API 包，那么导出的必须是 合格。 在任何一种情况下，如果标准模块导出一个 package 通过 Qualification，则导出必须到 模块。 如果标准模块是 Java SE 模块，则 *即* ，包含在 Java SE 平台规范中，则它不得导出任何非 SE API 包，至少不能无条件导出。

5. A standard module may depend upon one or more non-standard modules. It must not grant implied readability to any non-standard

module. If it is a Java SE module then it must not grant implied readability to any non-SE module.

标准模块可能依赖于一个或多个非标准模块。 它不得授予任何非标准模块的隐含可读性。如果它是一个 Java SE 模块，则它不得向任何非 SE 模块授予隐含可读性。

6. A non-standard module must not export any standard API packages. A non-standard module may grant implied readability to a standard module.

   非标准模块不得导出任何标准 API 包。 非标准模块可能会授予标准模块隐含的可读性。

An important consequence of principles 4 and 5 is that code that depends only upon Java SE modules will depend only upon standard Java SE types, and thus be portable to all Implementations of the Java SE Platform.

原则 4 和 5 的一个重要结果是，仅依赖于 Java SE 模块的代码将仅依赖于标准 Java SE 类型，因此可以移植到 Java SE 平台的所有实现中。

The module graph 模块图

The modular structure of the JDK can be visualized as a graph: Each module is a node, and there is a directed edge from one module to another if the first depends upon the second. The full module graph has too many edges to be displayed easily; here is the *transitive reduction* of the graph, in which redundant edges are omitted (click to enlarge):

JDK 的模块化结构可以可视化为图形：每个模块都是一个节点，如果一个模块依赖于第二个模块，则从一个模块到另一个模块有一条有向边。完整的模块图有太多的边，无法轻松显示;这是 *transitive reduction* ，其中省略了多余的边（单击可放大）：

Herewith a guided tour of the module graph:

以下是模块图的指导教程：

- Standard Java SE modules are colored orange; non-SE modules are colored blue.

  标准 Java SE 模块为橙色;非 SE 模块为蓝色。

- If one module depends upon another, and it grants implied readability to that module, then the edge from the first module to the second is solid; otherwise, the edge is dashed.

  如果一个模块依赖于另一个模块，并且它授予该模块隐含的可读性，那么从第一个模块到第二个模块的边是实心的;否则，边缘为虚线。

- At the very bottom is the `java.base` module, which contains essential classes such as `java.lang.Object` and `java.lang.String`. The base module depends upon no module, and every other module depends upon the base module. Edges to the base module are lighter than other edges.

  最底部是 `java.base` 模块，其中包含 `java.lang.Object` 和 `java.lang.String` 等基本类。base module 不依赖于 module，而其他每个模块都依赖于 base module。基本模块的边线比其他边线浅。

- Near the top is the `java.se.ee` module, which gathers together all of the modules that comprise the Java SE Platform, including modules that overlap with the Java EE Platform Specification. This is an example of an *aggregator* module, which collects and re-exports the content of other

modules but adds no content of its own. A run-time system configured to contain the `java.se.ee` module will contain all of the API packages of the Java SE Platform. A module is included in the Java SE Platform Specification if, and only if, it is a standard module reachable from the `java.se.ee` module.

顶部附近是 `java.se.ee` 模块，它汇集了构成 Java SE 平台的所有模块，包括与 Java EE 平台规范重叠的模块。这是一个*聚合器*模块的示例，该模块收集并重新导出其他模块的内容，但不添加自己的内容。配置为包含 `java.se.ee` 模块的运行时系统将包含 Java SE 平台的所有 API 包。当且仅当模块是可从 `java.se.ee` 访问的标准模块时，该模块才包含在 Java SE 平台规范中 模块。

- The `java.se` aggregator module gathers together those parts of the Java SE Platform that do not overlap with Java EE.

  `java.se` 聚合器模块将 Java SE 平台中与 Java EE 不重叠的部分收集在一起。

- The non-standard modules include debugging and serviceability tools and APIs (*e.g.*, `jdk.jdi`, `jdk.jcmd` and `jdk.jconsole`), development tools (*e.g.*, `jdk.compiler`, `jdk.javadoc`, and `jdk.xml.bind`), and various service providers (*e.g.*, `jdk.charsets`, `jdk.scripting.nashorn`, and `jdk.crypto.ec`), which are made available to other modules via the existing `java.util.ServiceLoader` mechanism.

  非标准模块包括调试和可维护性工具以及 API（*例如* `jdk.jdi`、`jdk.jcmd` 和 `jdk.jconsole`）、开发工具（*例如* `jdk.compiler`、`jdk.javadoc` 和 `jdk.xml.bind`）和各种服务提供商（*例如* `jdk.charsets`、`jdk.scripting.nashorn` 和 `jdk.crypto.ec`），其中 通过现有的 `java.util.ServiceLoader` 机制。

- The `java.smartcardio` module is standard but not part of the Java SE Platform Specification, hence its name starts with the string `"java."` but it is colored blue, and it is not reachable from the `java.se` module.

  `java.smartcardio` 模块是标准的，但不是 Java SE 平台规范的一部分，因此它的名称以字符串“`java.`”开头， 但它是蓝色的，无法从 `java.se` 模块访问。

The module graph is, in effect, a new kind of API, and is specified and evolved as such. The subgraph of the module graph rooted at the `java.se.ee` module, with all non-SE modules and corresponding edges removed, is specified in the Java SE Platform Specification; its evolution will hereinafter be governed by the JCP. The evolution of the remainder of the graph will be covered by future JEPs. In either case, if a module is specified as being available for general use then it will be subject to the same evolutionary constraints as other APIs. Removing such a module or changing it incompatibly, in particular, will require public notice at least one major release in advance.

模块图实际上是一种新型的 API，并且被指定并 进化成这样。 以 `java.se.ee` 模块（删除了所有非 SE 模块和相应的边缘）在 Java SE 平台规范中指定;其演变在下文中将由 JCP 管理。图形其余部分的演变将在未来的 JEP 中涵盖。在任何一种情况下，如果一个模块被指定为可用于一般用途，那么它将受到与其他 API 相同的进化约束。特别是，删除此类模块或更改不兼容的模块需要至少提前一个主要版本发布公告。

A tabular summary of all of the modules, including footprint metrics for a Linux/AMD64 build, is available here.

所有模块的表格摘要，包括 Linux/AMD64 构建的占用空间指标， 可在此处获得。

Testing 测试

The unit and regression tests in the JDK and `jtreg`, the harness used to run them, now allow tests to be selected on the basis of the modules which they test and upon which they depend, so that arbitrary configurations of JDK modules can be

tested.

JDK 和 jtreg 中的单元和回归测试（用于运行它们的工具）现在允许根据它们测试的模块和它们所依赖的模块来选择测试，以便可以测试 JDK 模块的任意配置。

The primary functional test of this enhancement inspects a configured set of modules to ensure that it is a valid combination of the modules defined herein, that each module has the expected content and exports the expected API packages, and that the modules have the expected dependence relationships.

此增强功能的主要功能测试检查一组已配置的模块，以确保它是此处定义的模块的有效组合，每个模块都具有预期的内容并导出预期的 API 包，并且模块具有预期的依赖关系。

The JCK can now test those aspects of the module graph which become part of the Java SE Platform Specification. This includes the names of the SE modules, their exported API packages, and the dependences amongst them that cause SE API packages to be re-exported. The JCK can also test arbitrary configurations of the SE modules present in a Platform Implementation.

JCK 现在可以测试模块图的那些方面，这些方面将成为 Java SE 平台规范的一部分。这包括 SE 模块的名称、它们导出的 API 软件包，以及它们之间导致 SE API 软件包重新导出的依赖关系。JCK 还可以测试平台实现中存在的 SE 模块的任意配置。

Risks and Assumptions 风险和假设

The modular structure defined here does not support at least one known use case, namely that of wanting to use the `java.beans` package without having to require the very large `java.desktop` module. It might not address other use cases, as yet unknown. If a critical use case is not supported in the final implementation of this JEP then we expect to be able address it in a later release by refactoring the module graph.

此处定义的模块化结构不支持至少一个已知的用例，即希望使用 `java.beans` 包而不必使用非常大的 `java.desktop` 模块。它可能无法解决其他未知的使用案例。如果此 JEP 的最终实现不支持关键用例，那么我们希望能够在以后的版本中通过重构模块图来解决该问题。

Dependences 依赖项

This JEP is the one of several for Project Jigsaw. The other JEPs are:

此 JEP 是 Jigsaw 项目的几个 JEP 之一。其他 JEP 是：

- 201: Modular Source Code 201： 模块化源代码
- 220: Modular Run-Time Images

  220：模块化运行时映像
- 260: Encapsulate Most Internal APIs

  260：封装大多数内部 API
- 261: Module System; see also JSR 376

  261 ： 模块系统 ;另请参见 JSR 376
- 282: jlink: The Java Linker

  282： jlink： Java 链接器