MT AG

data . cloud . mobile

Java 9: Endlich Jigsaw!

Salem Ben Nasr, DOAG 2017

22.11.2017

Facts & Figures

Technologie-orientiert
Branchen-unabhängig

Hauptsitz
Ratingen

240
Beschäftigte

MT AG

Zertifizierter
Partner von
Oracle,
Microsoft
und SAP

Inhabergeführt

24 Mio. Euro
Umsatz

Gründung
1994

Ausbildungs-
betrieb

kununu
TOP COMPANY
www.kununu.com
VON MITARBEITERN EMPFOHLEN!

Niederlassung
Frankfurt am Main

Agenda

1. Motivation
2. History
3. Modules
4. Types of Modules
5. Tools
    1. jdep
    1. jlink
6. Migration

History

- planned for Java 7 on August 2008
- postponed to Java 8
- postponed to Java 9
- Java 9 postponed for Jigsaw
- finally released September 2017

Motivation

- Monolith

- standard Module System for Java SE
- scalable
- security
- maintainability
- performance
- better tools for programming in the large
- strong encapsulation (declare which public types are accessible)

JEPs

- JEP 200 – The Modular JDK
- JEP 201 – Modular Source Code
- JEP 220 – Modular Run-Time Images
- JSR 376 – Java Platform Module System

Modules

Module

- named
- self-describing
- collection of code and data
- organized in packages (classes, interfaces)

Module declaration

- file named „module-info.java“
- module names must not conflict (reverse-domain-name-pattern)
- placed in root

```
module org.doag2017.jigsaw.one {
}
```

Requires

- declares that the module depends on other modules

```
module org.jdoag2017.jigsaw.one {
    requires org.doag2017.jigsaw.two;
    requires java.sql;
}
```

Export

- only exported packages can be used by other modules

```
module org.doag2017.jigsaw.two {
    exports org.doag2017.jigsaw.two;
    exports org.doag2017.jigsaw.two.api;
}
```

Qualified Export

- exports can be used only by the specified modules

```
module org.doag2017.jigsaw.two {
    exports org.doag2017.jigsaw.two to org.jigsaw2017.jigsaw.one;
    exports org.doag2017.jigsaw.two.api
}
```

Transitive Require

- depends on an module and exports this module

```
module org.doag2017.jigsaw.one {
    exports org.doag2017.jigsaw.one;
    requires transitive org.doag2017.jigsaw.two;
}
```

Service Provider

- uses <service>
- provides <service> with <implementation>

```
module org.doag2017.jigsaw.one {
    exports org.doag2017.jigsaw.one;
    uses org.doag2017.jigsaw.one.MyService;
}
```

```
module org.doag2017.jigsaw.three {
    requires org.doag2017.jigsaw.one;
    provides org.doag2017.jigsaw.one.MyService
        with org.doag2017.jigsaw.three.MyServiceProvider;
}
```

Accessibility

- module a exports package
- module b requires module a
- class and method must be public

Module-Path

```
javac -d modules --module-path "automaticModules"
--module-source-path src $(find src -name "*.java")
```

```
java --module-path "automaticModules;otherModules“
-m org.doag2017.moduleOne/org.doag2017.moduleOne
```

Types of Modules

Types of Modules

- Platform Explicit Modules
- Application Explicit Modules
- Automatic Modules
- Unnamed Module

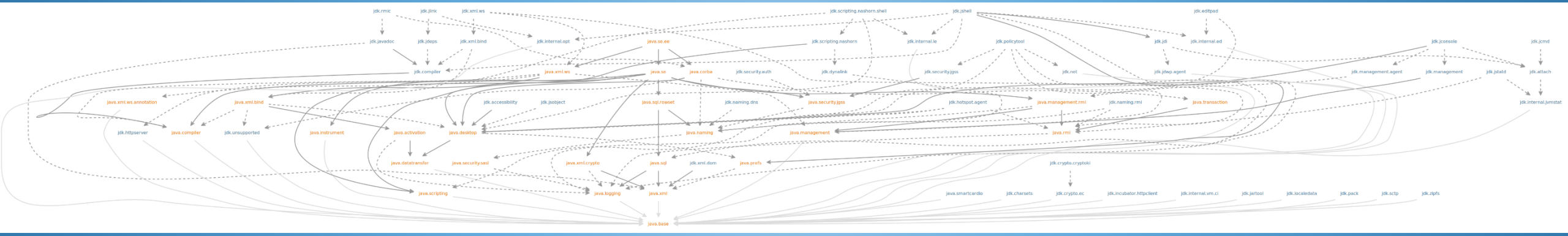
Platform Explicit Modules

- Base module
  - java.base
  - always present
  - every module depends implicit on it
  - depends on no other module

- Platform modules
  - java.*
  - depends implicit on base module
  - java.sql, java.transaction…

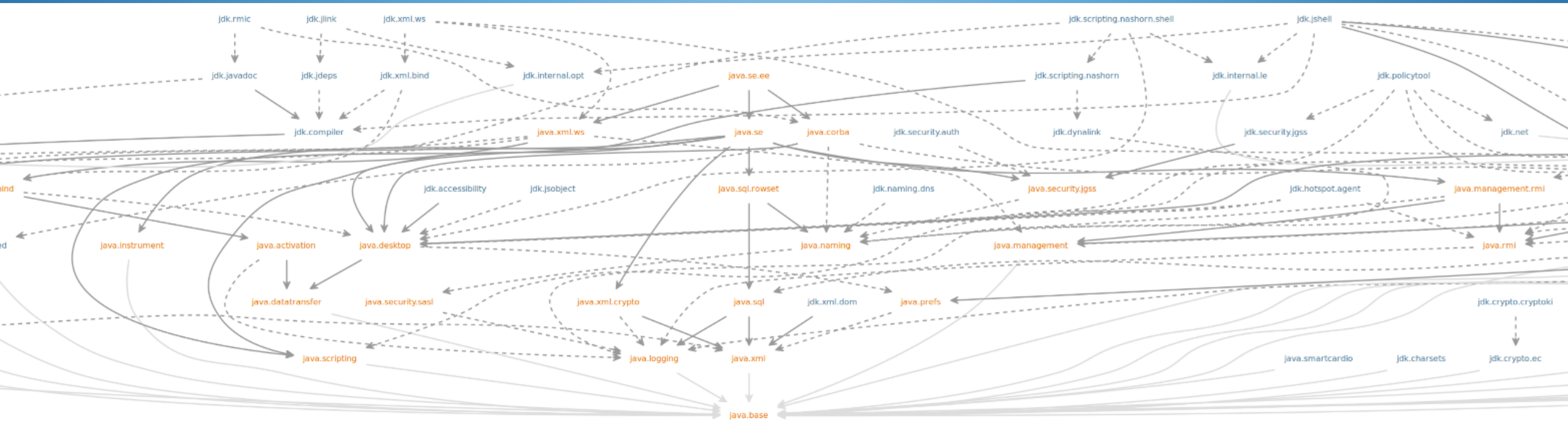
- Specific to the JDK
  - jdk.*
  - depends implicit on base module
  - jdk.jlink, jdk.jconsole…

JEP 200 – The Modular JDK



Quelle: http://openjdk.java.net/jeps/200

JEP 200 – The Modular JDK



Quelle: http://openjdk.java.net/jeps/200

Jmods

Application Explicit Modules

- Jar-Libs in modul-path
- module-descriptor (module-info.class) exists

- only Modules declared with required are accessible
- no access to unnamed module

Automatic Modules

- Jar-Lib in module-path
- no module-descriptor

- named module defined implicit

- reads implicit all automatic modules and explicit modules
- every package is implicit exported

Unnamed Module

- consists of eveything in the class-path

- reads every other module
- exports all packages

- explicit and automatic modules can not declare a dependency to it

JLink

Modular Runtime Images with JLink

- specific for OS
- Input: modular JAR, JAR, JMOD, .class
- only for explicit modules
- minimum options:
  - module-path
  - add-modules
  - output

Jlink Demo

Migration

Migration

- No-Migration
- Top-down-Migration
- Bottom-up-Migration

No-Migration

No-Migration

modular system ist NOT mandatory

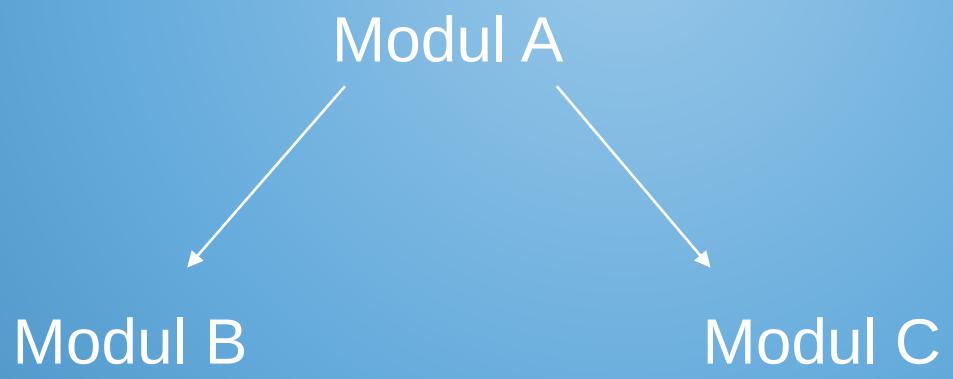
just use the class-path

Bottom-up-Migration

Bottom-up-Migration

- use unnamed module
- all jars in class-path

- write module-descriptor from the bottom up
- move jar by jar to module-path

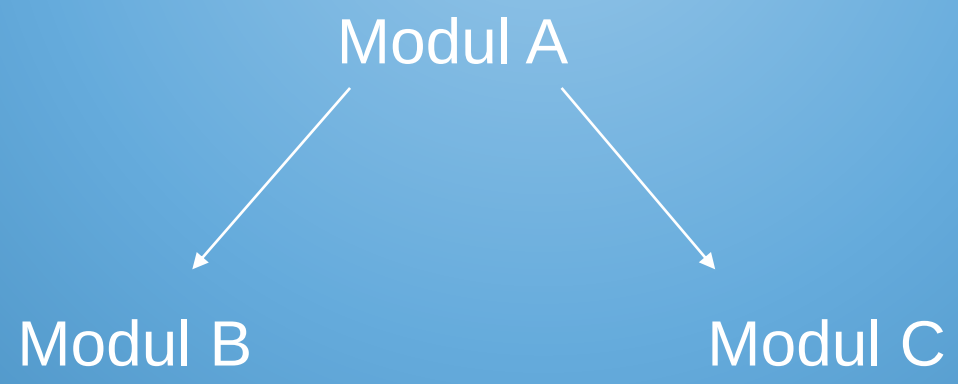


Top-down-Migration

Top-down-Migration (1)

- use automatic modules
- put all jars in module path

- Advantages:
  - requires all modules
  - exports all packages
  - works also for libs without module-descriptor



Top-down-Migration (2)

1. generate module-descriptors with jdeps
2. remove not needed exports

```
jdeps --generate-module-info path “hamcrest-core-1.3.jar“ “junit-4.12.jar"
```

Jdeps Demo

*examples by **Alexandru Jecan**

That‘s it?

No

Internal JDK-APIs

- internal JDK-APIs no longer available for modules (sun.*)
- exceptions jdk.unsupported

- Solution:
  - requires jdk.unsupported
  - use unnamed module

- Tool:
  - jdeps –jdkinternals <jar>

Jdeps Demo

*examples by **Alexandru Jecan**

java.se.ee and unnamed module

- modules can‘t be found
  - java.activation
  - java.cobra
  - java.transaction
  - java.xml.bind
  - java.xml.ws
  - java.xml.annotation

Error:
- Package does not exist

Solution:
- add module with –add-modules <module>
- must be set on compile- and runtime

Fragen?

Vorträge der MT AG

Dienstag, 21. 11.17

JavaScript Tuning in modernen Web-Applikationen

16.00 Uhr | Foyer Tokio
Till Albert

Mittwoch, 22. 11.17

Ein Snapshot ist kein Backup

8.00 Uhr | Raum Shanghai
Angelina Weinschenk

Pimp my iGrid

15.00 Uhr | Raum Kopenhagen
Moritz Klein

Jetlag: Oracle JET und APEX

9.00 Uhr | Raum Kopenhagen
Oliver Lemm

APEX (Hoch)verfügbar? Darf etwas Open Source sein?

15.00 Uhr | Raum Seoul
Ernst Leber

Mit Augenhöhe und Aufmerk-samkeit Projekte zum Erfolg führen

13.00 Uhr | Raum Kiew
Carsten Firus

OWB-ODI Migration: Fallstricke & Lösungen im Praxisbericht

16.00 Uhr | Raum Oslo
Jürgen Günter

Java Script und Offline First

15.00 Uhr | Raum Kopenhagen
Kai Donato

Wieder verschätzt?

17.00 Uhr | Raum Singapur
Oliver Lemm

APEX open Mic Night

20.30 Uhr | Raum Istanbul
Niels de Bruijn

Donnerstag, 23. 11.17

Besserer Java Code, außerhalb der Automatismen

9.00 Uhr | Raum Helsinki
Wolfgang Nast

Java 9: Endlich Jigsaw!

10.00 Uhr | Raum Budapest
Salem Ben Nasr

So bringen Sie Ihr DWH Projekt zum Scheitern

12.00 Uhr | Raum Stockholm
Irina Gotlibovych

Java Script und PL/SQL – das dynamische Duo für APEX

12.00 Uhr | Raum St. Petersburg
Moritz Klein

JSON in Java mit Schema und JsonPath

14.00 Uhr | Raum Oslo
Wolfgang Nast

Freitag, 24. 11.17

Ihre Datenbank startet nicht? Oder Anatomie des Startup-Prozesses einer Oracle-Datenbank

09.00 Uhr | NCC
Ernst Leber
Workshop

MT AG