

14.–17. 09. 2009
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Gut verpackt

Das Java Module System und Super-Packages

Michael Wiedeking

MATHEMA Software GmbH

Überblick

- Was ist ein Modul?
 - „Eine technische oder organisatorische Einheit...“
 - Ein Feature von Java 7
 - Eine Einheit über den Java-Packages
- Warum brauchen wir Module?
 - Vereinfachung
 - Verbesserung

Das Module System (JSR 277 und JSR 294)

- JSR-277 Early Draft seit 02.10.2006 (Stanley Ho)
- JSR-294 Strawman Proposal seit 16.03.2007
(Andreas Sterbenz und Alex Buckley)
- Expert Group:

Apache Software Found.
Google Inc.
IBM
Jayasoft
Neward, Ted
Red Hat Middleware LLC

BEA Systems
Hall, Richard S.
Intel Corp.
Lea, Doug
Oracle
SAP AG

Bock, David W
Halloway, Stuart
Ironflare AB
Leuck, Daniel
Pullara, Samuel
SAS Institute Inc.

Das JAR-Format

- Java Archive (JARs) gibt es bereits seit Mitte der 90er
- Distribution
- Ausführung
- ZIP-Datei
- Zusätzliche Metadaten „META-INF/MANIFEST.MF“

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.6.5
Created-By: 1.4.2_12-b03 (Sun Microsystems Inc.)
Main-Class: org.apache.catalina.startup.Bootstrap
Specification-Title: Catalina
Specification-Version: 1.0
Class-Path: jmx.jar commons-daemon.jar
commons-logging-api.jar tomcat-juli.jar
```

Schwierigkeiten mit JARs

- Versionierung
- Schwierigkeit andere JARs zu referenzieren
 - relative Pfadangaben
 - absolute Pfadangaben
- Namensraumkonflikte
- Ineffizient bei der Ausführung

Schwierigkeiten beim Information Hiding

- Zugriffs-Modifikatoren
 - private
 - protected
 - public
- Keine sprachlichen Möglichkeiten über die Package-Grenzen hinaus

Überblick über die Architektur

- Distributionsformat
- Versionierungsschema
- Depot (Repository)
- Laufzeitunterstützung
- Einfachheit
- Reflective API

Unterscheidung von JSR 294 und JSR 277

- Development Module (JSR 294)
 - superpackage-Konstrukt
 - sprachliche Einheit
 - Unterstützung der VM bei der Zugriffskontrolle
- Deployment Module (JSR 277)
 - wiederverwendbare Einheit
 - enthält Metadaten
 - Repository-Infrastruktur

Was ist ein Superpackage

- Sprachliches Konstrukt
 - Listet eine Menge von Member-Packages auf
 - Öffentliche (public) Members sind in allen Member-Packages zugänglich
 - Nur Mitglieder die explizit exportiert werden sind außerhalb der Packages zugänglich
 - Kann kompiliert als Metainformation für das Deployment-Module verwendet werden

Beispiel ohne Annotations

```
superpackage de.mathema.einModul {  
  
    de.mathema.einModul.meineSachen.*;  
    de.mathema.einModul.deineSachen.*;  
    de.mathonix.etwasAnderes.andereSachen.*;  
  
    export de.mathema.einModul.meineSachen.*;  
    export de.mathema.einModul.deineSachen.*;  
  
    import org.openSource.andereModule;  
}
```

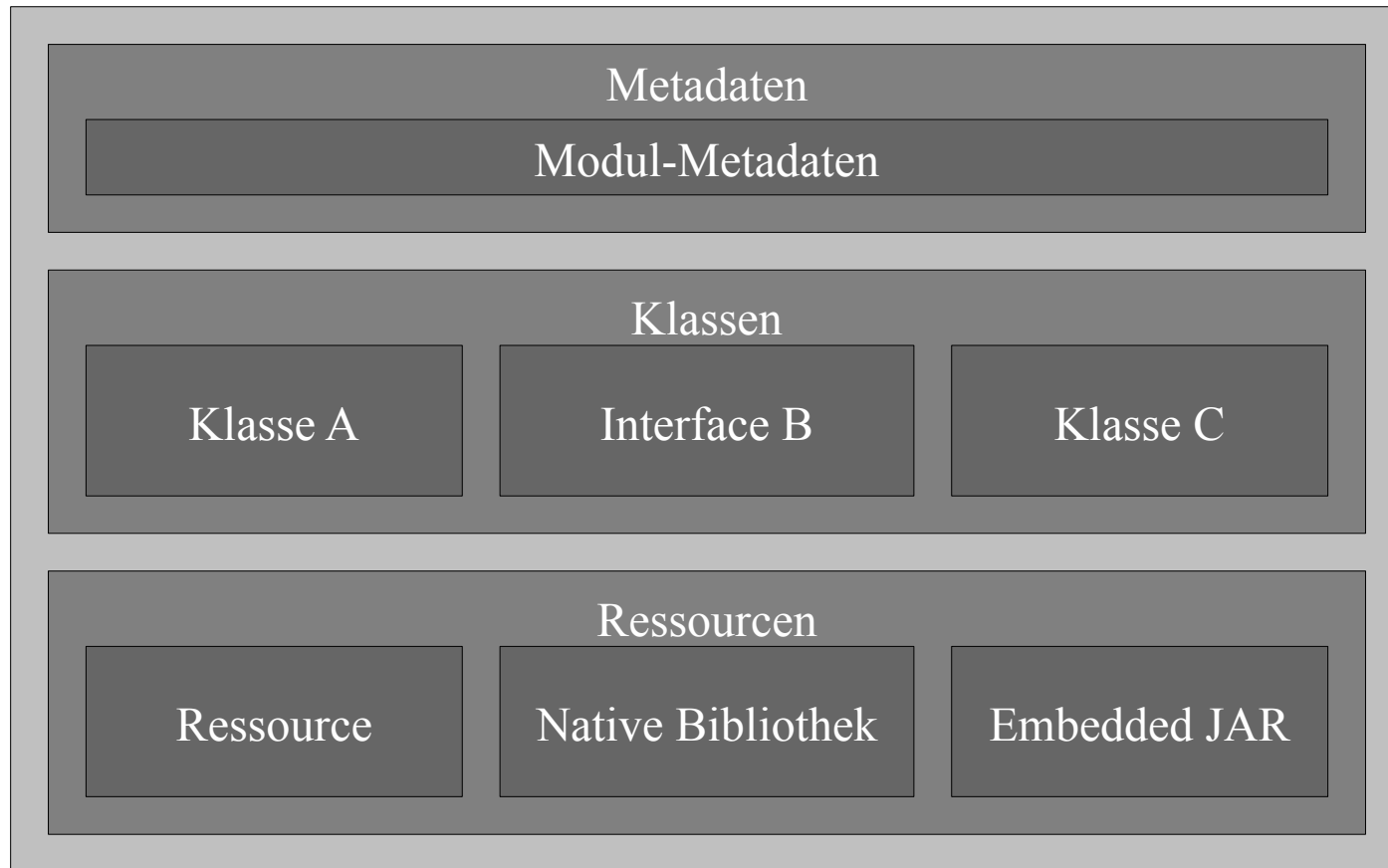
Beispiel mit Annotations

```
@Version("1.0.0")
@ExportResources({
    "bilder/bild1.jpg",
    "bilder/bild2.jpg"}) ;
@ModuleAttributes({
    @ModuleAttribute("mein.text", "Hallo Welt!")
    @ModuleAttribute("secret.id", "42")});
superpackage de.mathema.einModul {
    de.mathema.einModul.meineSachen.*;
    de.mathema.einModul.deineSachen.*;
    de.mathonix.etwasAnderes.andereSachen.*;
    export de.mathema.einModul.meineSachen.*;
    export de.mathema.einModul.deineSachen.*;
    import org.openSource.andereModule;
}
```

Die Module-Datei (1)

- Die binäre Definition eines Moduls
- VM verwendet Module-Datei zur Zugriffskontrolle
- Jede Klasse muss die Zugehörigkeit zu einem Modul bekannt geben
- Entspricht den Metadaten im Deployment-Modul
- Ermöglicht Tool-Unterstützung

Die Module-Datei (2)



Versionierung (1)

- Beispiel für eine Version
 - 1.5.0.1-beta-2
- Versionierungsrichtlinie:
 - Major
 - Minor
 - Micro
 - Update
 - Qualifier

Versionierung (2)

- Grammatik für Versionsnummern:

```
version := simple-version ('-' qualifier)?  
simple-version :=  
    major ('.' minor ('.' micro ('.' update)?)?)?  
major := digit+  
minor := digit+  
micro := digit+  
update := digit+  
qualifier := (alpha | digit | '-' | '_')+
```

- Vergleich von Versionen:
 - $1 < 2 < 2.1 < 2.1.4 == 2.1.4.0 > 2.1.4.0\text{-irgendwas}$

Intervalle von Versionen

- offene Intervalle
 - **a+**
 - **a.b+**
 - **a.b.c+**
 - **a.b.c.d+**
- Familien von Versionen
 - **a***
 - **a.b***
 - **a.b.c***
 - **a.b.c.d***
- Offene Intervalle in Familien
 - **a.[b.c.d+]**
 - **a.b.[c.d+]**
 - **a.b.c.[d+]**
- Beispiel:
$$1.[2.4.2];2^*;3^*$$

~

$$[1.2.4.2, 4.0.0.0]$$

Export

- Definiert welche Klassen und Ressourcen für externe Module sichtbar sind.
- Das Einhalten der Exportliste wird zur Kompilierzeit und zur Laufzeit sichergestellt

```
@Version("2.5.6")
@ExportResources({
    "bilder/bild1.jpg",
    "bilder/bild2.jpg"}) ;
superpackage de.abc.eins {
    export org.open.das.* ;
}
```

Import

- Import definiert die Abhängigkeiten von anderen Modulen
- Statisch

```
superpackage de.abc.bsp {  
    @VersionConstraint("1.3")  
    import de.mathema.eins;  
}  
  
superpackage de.abc.bsp {  
    @VersionConstraint("2.0+")  
    import de.mathema.eins;  
}
```

- Flexibler durch Verwendung einer Import-Policy
- Kann in den Metadaten angegeben werden

Import-Policy

```
public class CustomImportPolicy extends ImportPolicy{
    private Module self;
    public CustomImportPolicy(Module self) {
        this.self = self;
    }
    public void prepare() {
        ModuleDefinition mdef = self.getModuleDefinition();
        Repository repos = mdef.getRepository();
        ModuleDefinition xml = repos.find("org.foo.xml",
                                           Query.parse("1.0.0"));
        ModuleDefinition soap = null;
        if (System.getProperty("os.name").equals("Linux"))
            soap = repos.find("org.foo.soap",
                              Query.parse("2.3.4"));
        else
            soap = repos.find("org.foo.soap", Query.parse("3.0+"));
        self.addImports(xml.getInstance());
        self.addImports(soap.getInstance());
    } }
```

Metadata

- Main Class
 - java -module de.mathema.abc:1.0.0
- Plattform-Bindung
- Attribute

```
@MainClass("de.mathema.abc.Main")
@PlatformBinding(Platform="win32", arch="x86")

superpackage de.abc.eins {
    ...
    export org.open.das.*;
}
```

Was ist ein JAM File

- Ein Modul-Archiv ist die physikalische Repräsentation eines Moduls
- Der empfohlene Name ist:
`<name>-<version>[-<platform>-<arch>].jam`
- Die Teile einer JAM-Datei:
 - Metadaten
 - Klassen
 - Ressourcen
 - Eingebettete JAR-Dateien
 - Native Bibliotheken

Signierung und Komprimierung von JAM-Dateien

- Eine JAM-Datei kann beliebig oft signiert werden
 - Beachtung der Konsistenz
 - Ignorieren der JARs
- Komprimierung mit pack200 ermöglicht weitere Komprimierung
 - Direkte Unterstützung
 - **<name>-<ver>-[-<plat>-<arch>].jam.pack.gz**

Aufbau (Verzeichnisstruktur)

- Physikalische Ansicht eines Java Moduls

```
de.mathema.abc-1.0.1.jam:  
  /META-INF/MANIFEST.MF  
  /MODULE-INF/METADATA.module  
  /MODULE-INF/bin/abc-win.dll  
  /MODULE-INF/bin/abc-linux.so  
  /ClassA.class  
  /ClassB.class  
  /InterfaceC.class  
  /image/graphics.jpg
```

jam-Tool Komandozeilenbeispiel

- Die -jam Option des Java-Launchers
 - `java -jam .../org.abc.Xml-2.1.jam`
 - `java -jam .../org.abc.Xml-2.1.jam.pack.gz`
 - `java -cp abc-path -module org.abc.Xml:2.1+
-modulemain AbcClass`
- Das jam Verpackungs-Tool
 - `jam cfs org.foo.mymodule-1.0.jam
org.foo.mymodule .`

Was ist ein Repository

- Mechanismus des Module-Systems zum:
 - abspeichern
 - auffinden
 - abrufen
- Erlaubt eine oder mehrere Modul-Definitionen



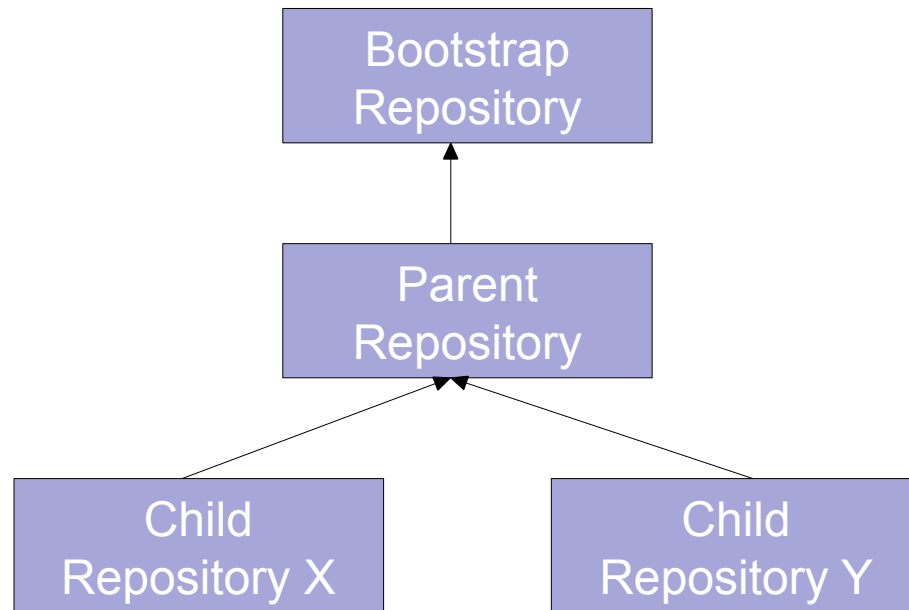
- Bootstrap-Repository

Möglichkeiten beim Deployment und Optimierung

- Möglichkeiten beim Deployment
 - „Shared Modules“ in einem globalen Repository
 - private Module in einem Anwendungs-Repository
 - als „Download-Modul“ in einem URL-Repository
- Möglichkeiten zur Optimierung
 - Generierung eines Indexes für Klassen und Ressourcen
 - Verifizierung von Klassen
 - Extrahieren und verifizieren der digitalen Signatur
 - Kompilierung des Moduls in Maschinencode
- Weitere Implementierungen können weitere Möglichkeiten eröffnen

Repository-Delegationsdiagramm

- Delegation
- Konstruktion
- Lifecycle
 - initialisation
 - shutdown
- Zustände
 - aktiv
 - inaktiv



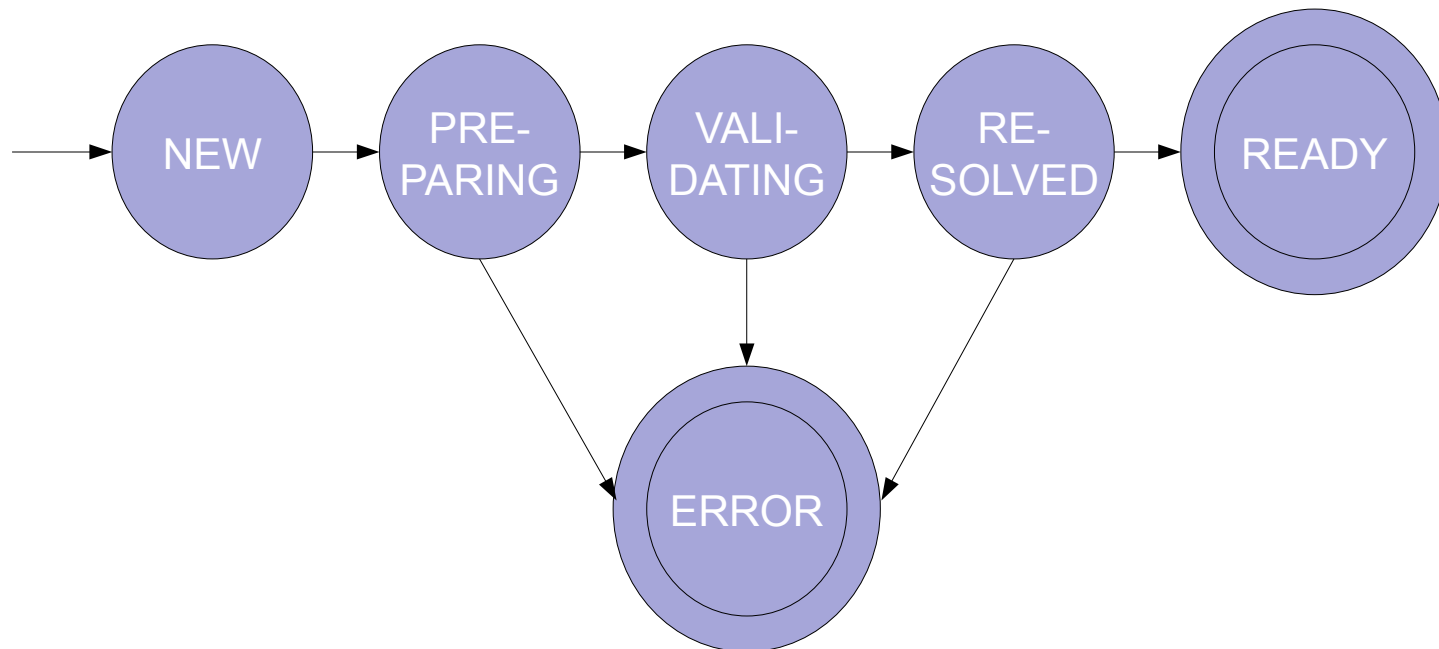
JRepo Repository-Management-Tool

- JRepo erlaubt Inspektion und Modifikation von Repositories
- `jrepo <command> <flags> <options>`
 - `list`
 - `install`
 - `uninstall`
 - `dependencies`
 - `contents`
 - `get`
 - `check`

Einfluss auf die Java Runtime

- Das Bootstrap-Repository enthält Standard-Module
 - java.se Modul
- Individuelles Erweiterungsmodul
- „classpath“-Modul

Zustände einer Modul-Instanz

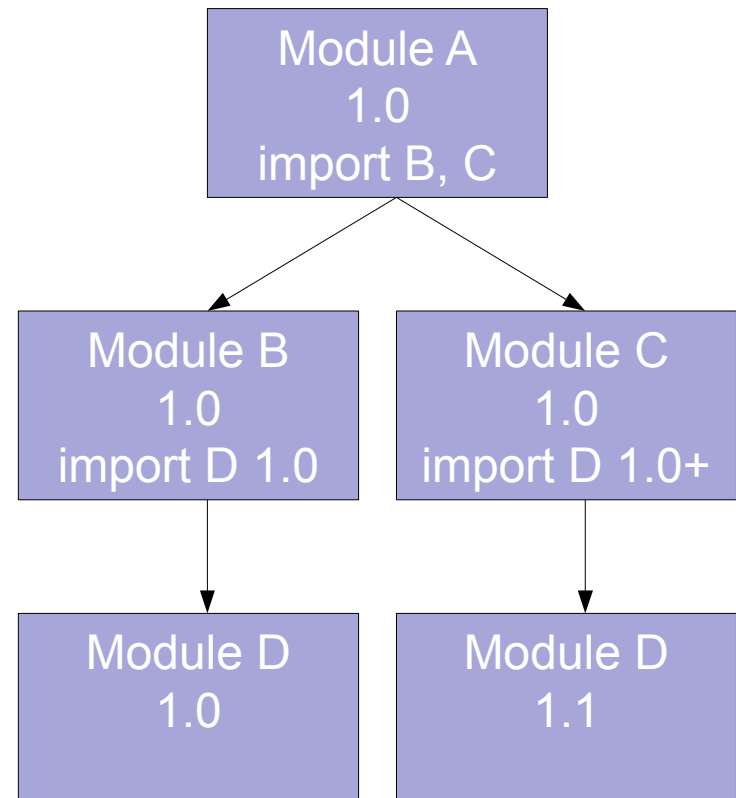


Auflösen (Resolving)

- Was versteht man unter Resolving?
- Vorbereitung
 - Import Policy
- Validierung
 - oberflächlich (Shallow Validation)
 - tief (Deep Validation)

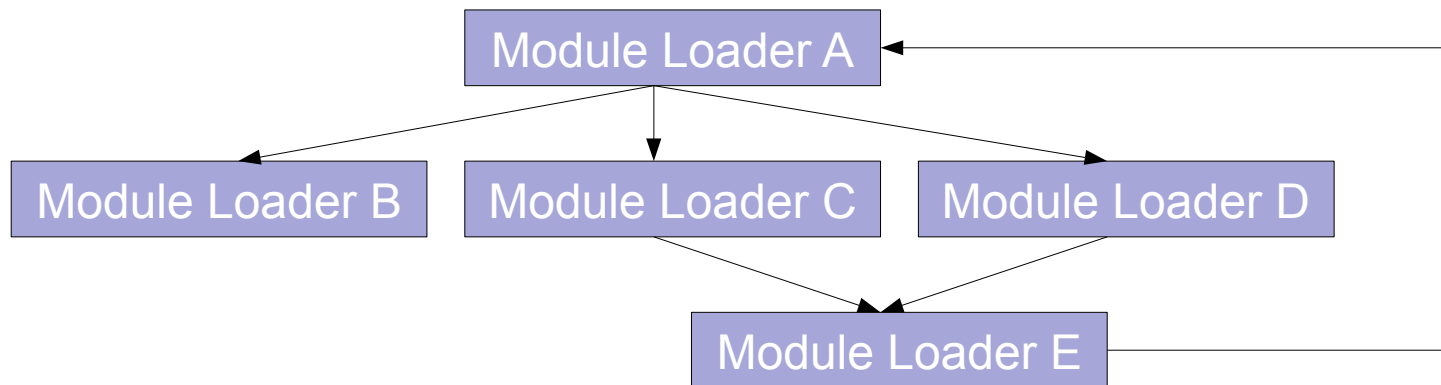
Resolving (Beispiele)

```
@Version("1.0")
super package A {
import B; import C; ...
}
@Version("1.0")
super package B {
@VersionConstraint("1.0+")
import D; ...
}
@Version("1.0")
super package C {
@VersionConstraint("1.0")
import D; ...
}
@Version("1.0")
super package D { ... }
@Version("1.1")
super package D { ... }
```



ClassLoading in Classloadern

- Jedes Modul hat einen Modul-Classloader
- Die Suchreihenfolge wird bestimmt durch die Exporte der importierten Module
- Erleichtert paralleles Classloading



Migration

- Modul-Abhängigkeiten von JARs
 - importiere die Classpath-Modul-Definition
- JAR-Abhängigkeiten von Modulen
- JAR-Abhängigkeiten auf Erweiterungen (Optionale Pakete)
- Benutzerdefinierte Class-Loader
 - Aufruf der Reflection API
- Beschränkungen
 - nur eine Version der JARs
 - nur eine Version der Module

Erweiterbarkeit

- Die Architektur von JSR-277 ist erweiterbar:
 - alternative Repository-Implementierungen können verwendet werden
 - Datenbank
 - Maven Repositories
 - Ivy Repositories
 - andere Modulsysteme können ihre Module als JSR-277 Module herausgeben
 - OSGi
 - NetBeans

Module

```
planetjdk/src/  
  org/planetjdk/aggregator/Main.java  
  module-info.java
```

module-info.java

```
module org.planetjdk.aggregator {  
    system jigsaw;  
    requires module jdom;  
    requires module tagsoup;  
    requires module rome;  
    requires module rome-fetcher;  
    requires module joda-time;  
    requires module java-xml;  
    requires module java-base;  
    class org.planetjdk.aggregator.Main;  
}
```

Übersetzen

```
planetjdk/src/  
  org/planetjdk/aggregator/Main.java  
  module-info.java
```

```
javac    module-info.java  
        -sourcepath planetjdk/src  
        -d planetjdk/cls  
        org/planetjdk/aggregator/*.java  
        module-info.java
```

```
planetjdk/cls/  
  org/planetjdk/aggregator/Main.class  
  module-info.class
```

Titel

```
module org.planetjdk.aggregator @ 1.0 {  
  
    system jigsaw;  
    requires module jdom @ 1.0;  
    requires module tagsoup @ 1.2;  
    requires module rome @ 1.0;  
    requires module rome-fetcher @ 1.0;  
    requires module joda-time @ 1.6;  
    requires module java-xml @ 7;  
    requires module java-base @ 7;  
    class org.planetjdk.aggregator.Main;  
}
```

Vergleich mit bestehenden Systemen

- Viele existierende Systeme bieten bereits Lösungen
 - OSGi, NetBeans, .NET, Maven, Ivy
- Unterstützung von JSR-294
 - weniger Daten in XML
- Erweiterbarkeit
 - Möglichkeit zur Interaktion mit anderen Systemen

Ausblick, offene Themen

- Untermodule
- Distributionsformat für mehrere Module
- Internationalisierung
- Vertragliche Abhängigkeiten
- Monitoring
- Management
- Interoperabilität mit anderen Modul-Systemen

Zusammenfassung

- Entwicklung
 - definieren von Superpackages
 - enthalten Klassen und Packages
- Deployment
 - Möglichkeit zur Versionierung
 - bieten Reflektiven Zugriff
 - Format zur Distribution
- Enthalten in Java 7

Weitere Informationen

- Offizielle JCP-Seiten:
 - JSR 277: <http://jcp.org/en/jsr/detail?id=277>
 - JSR 294: <http://jcp.org/en/jsr/detail?id=294>
- Blogs:
 - Stanley Ho: <http://weblogs.java.net/blog/stanleyh/>
 - Andreas Sterbenz:
http://blogs.sun.com/andreas/entry/openjdk_modules_source
- Interviews:
 - Ted Neward befragt Stanley Ho:
<http://www.parleys.com/display/PARLEYS/St Stanley+Ho+Java+Polis+2006+interview>

Noch mehr Informationen

- Podcasts:
 - The Java Posse:
http://javaposse.com/index.php?post_id=234692
 - BeJUG:
http://media.libsyn.com/media/javapolis/JavaPolis_2006_-_Ja
- Diskussionen
 - The real story about OSGi and JSR 277:
http://groups.google.com/group/javaposse/browse_thread/thread
 - <http://www.javalobby.org/java/forums/t82769.html>

Fragen?

14.–17.09.2009
in Nürnberg



Wissenstransfer
par excellence

Vielen Dank!

Michael Wiedeking

MATHEMA Software GmbH