Mit **Java 7 und Jigsaw** werden wir alle schon über die Plattform mit einem Modularisierungskonzept und passendem Build-Management versorgt.

Bisher etablierte Konzepte und Vorgehensweisen zur Modularisierung wie **OSGi und JEE werden hierdurch auf den Prüfstand gestellt.**

"Ist doch viel zu monolitisch", "wird doch nicht testbar"... – Wo aber bleibt die Architektur bei all der Technologie in der Debatte?

Wir geben Hinweise, wie man sich anständig durch das Chaos schlägt.
Session Topics

- Modularization?
  - Java & Modularization
  - A word on tooling

architecture?
As documentation produced by architects, typically drawings, plans and technical specifications, architecture defines the **structure** and/or **behavior** of a building or any other kind of system that is to be or has been constructed.

**Wikipedia**

technologies ain‘t no architecture
XML is not extensible

XML is no fun

creativity???
XML used in XML
Analogy?

- Technologies as meta-language
  - you can do everything
  - technologies are fun
  - you must be creative
- Architecture
  - is how we finally structure our thing and use the technologies
  - stops being fun

left-hand-right-hand situation

- Should architectures be neutral to technologies?
- When do we find out if a given architecture fits a given technology stack?
- Who is in charge here?
super-architect

modularization?

Spring

JSF

JPA 2.0
Divide & Conquer

break it into pieces

- components
- layers
- modules
- aspects?
I am amonolithical thing!
I am monolithic thing!

I am not known for reusability, maintainability, etc..

bad guy

see the duplicate code?
re-use it!

lifecycle? state? runtime?
managed bean!

component based development
break it into pieces

✓ components
• layers
• modules
• aspects?
modules

- components
- layers
- modules?

break it into pieces
modules?

allowed intimacy level?
break it into pieces

modules?

aspects?

modules

layers

components
break it into pieces

minimum aop?
integration?

simple 2 client situation
... with versions ...

and 3rd party libs
and runtimes

How?

• Release management?
• Versioning?
• Backwards compatibility?
• Dependency resolution & mediation?
• Compiletime vs. runtime?
Session Topics

- Modularization?
- Java & Modularization
- A word on tooling

Java EE
Java EE modules

- ear
  - packaging, class loaders, ...

- ejb-jar
  - "instance sharing"
- ejb-jar

- util-jar
  - "class sharing"

Dienstag, 7. Dezember 2010
allowed intimacy level?

modules?
we can...

- package modules
- do class sharing
- do instance sharing
- no versions
- no visibility constraints
- no module lifecycle

DI container/ Spring
runtime dependency scopes

- compile time dependencies handled by compiler, IDE, classpath
- runtime dependency declared/ managed in context

Maven (ivy too...)
dependency mediation for poor EE people

- because we have versions
- solve version problems at compile time
- no fancy-OSGi features at runtime

and 3rd party libs
transitive dependencies

dead end with Java EE
Bundle

OSGi
OSGi

OSGi Runtime
packaging, class loaders,...

"instance sharing"

v1.0

v1.1

"class sharing"

OSGi bundles
we can...

• do class sharing
• do instance sharing
• do runtime loading/unloading
• do multi-version-loading
• define visibility constraints
• hard to do cross-module-container-ee-services

Java EE APIs

don't fit into OSGi

e.g. ....

monolithic configs
classloading nightmares
don't work by design
Java 7, 8, ...

super packages?
modules?
simplified module system?
dynamic modules?

==> Java EE ?!!

All this works fine as long as a project fits into a single package. Once the project grows beyond that, you may find yourself forced to make implementation classes public in order to access them from multiple packages in the project.

That loses the benefits of information hiding, which is clearly suboptimal and a long-standing complaint that often comes up in language discussions in forums such as JavaOne.

Blog: Superpackage strawman and the JSR 294 mailing list
as long as a project fits into a single package. forced to make implementation classes public

loses the benefits of information hiding, clearly suboptimal

Blog: Superpackage strawman and the JSR 294 mailing list

However, if a project is too large to fit into a single package, Java does not provide a suitable encapsulation mechanism.

Strawman Proposal for JSR 294 Superpackages
Session Topics

- Modularization?
- Java & Modularization
- A word on tooling

tools: a matter of context
develop modular software

- APIs
- IDEs
- Test-Frameworks

build modular software

- integration server
- automated builds
- distributed builds?
- provisioning?
- release management?
run modular software

- runtime for modules
- module/app deployment
- intelligent clustering?

each tool is great

- and does (mostly) not work with the others
- no standard? no standard tooling.
Was tun?

Problem

Technologien können einem das noch so schönes und ausgeklügeltes Architekturvorhaben ordentlich versaubeuteln.
Zauberhafte Technologien

- Sind nicht mehr so zauberhaft wenn man versucht modular zu entwickeln.
- Viele Komponentenmodelle, keine Lösungsansätze für Modularisierung
- Zaubermtrank wird schnell zum Gift

Thanks!
Schulung: Einführung in die OSGi Service Platform

http://www.oio.de/seminar/java/seminar-osgi-schulung-equinox-training.htm

Schulung: Eclipse als Rich Client Plattform

http://www.oio.de/eclipse.swt-jface-plugin-rcp-osgi-schulung.htm

Mehr von OIO zum Thema