Module-based development with Spring and Maven 2

Anticipating the emerging module technologies

Philipp H. Oser
ELCA
3502

Martin Zeltner

JAZOON08
THE INTERNATIONAL CONFERENCE ON JAVA TECHNOLOGY
JUNE 23 - 26, 2008 ZURICH

ELCA

Sun microsystems

ELCA

“Jar-files”

OSGi

Spring dynamic modules

Why modules?

JSR 277: Java Modules

“Maven 2”

A module is a collection of code, configuration and metadata that implements some responsibilities. Modules can have “depends on” relations to other modules.

Splitting your application in separate modules can

> reduce complexity
> reduce undesired coupling
> simplify team development
> decrease execution size by using only the required modules
> …

Modules can help for (1) project organization, during (2) development and (3) run time

What module abstraction is suitable?

We answer that question based on our experience with modules over the last 6 years. 4 big frameworks (2 in Java, 2 in .NET), around 70 applications developed with them.

AGENDA

- Intro
  - What we would like from a module support
  - Our context
- Improved dependency management
- What we can get with this improved dependency management
  - Configuration scenarios
  - Parameters for Multi-Environments
  - Execution order of DB scripts
  - Recursive Maven executions

ELCA portrait

One of the main Swiss independent companies in the IT development and system integration field.

We develop, integrate, operate, and maintain IT solutions using custom developed applications.

Founded in	1968
Employees	500+
Offices	Lausanne, Zurich, Geneva, Bern, London, Paris, Madrid, Ho Chi Minh City
Turnover	CHF 63M, uninterrupted positive results for 20 years
Certification	ISO9001 (since 1993), CMMI Level 3 (since 2007)
Awards	SWISS TECHNOLOGY AWARD; CMMI; THE EUROPEAN IST PRIZE WINNER; SOFI SWISS ORGANISATION FOR FACILITATING INVESTMENTS

Features simultaneously for project organization, development, and run time

Definition of parameters in modules, with ability to overload

> Convention over configuration, but be able to define exceptions
> Provide ability to *share* some parameters (avoid duplication):
  – among modules
  – between development (Maven 2) and run time (Spring)

Clean support for multiple environments and technologies

Cartesian product

> databases x web containers x JDK versions x security or not x web vs gui
  – => allow definitions for multiple environments *on the level of the module*

Automation

> Run some tasks on a *group of modules*
  – Build, launch, make a release, create & set up the database, run the automatic tests, generate some reports, deploy via single-jar or webstart, …

Features

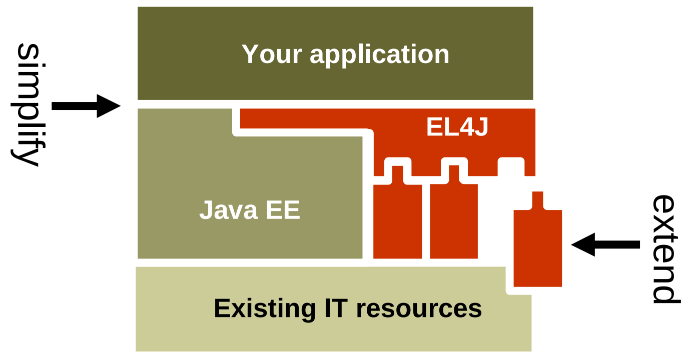
Project organization:

> Binary modules & “local” modules
> A repository of all modules
> Module versioning, with a version resolution scheme
> Transitive dependency management

Run time:

> Presence of a module => allow it to run some initialization code
> A “space” of components / services => for indirect collaboration of modules (modules collaborate via components in this “space”)
> Interception/ AOP mechanism => for a module to “implicitly” change behavior of other modules

Our situation



EL4J.sf.net combines Java frameworks (Spring, Maven 2, Hibernate, …) to simplify and extend the Java EE.

Challenges:

> Organize EL4J itself & projects built with it

> 100+ modules

> Support for organization of many different environments

> Get towards an intelligent application management, covering
>
> – Project organization
>
> – Development/ build time
>
> – Run time

Overview of our approach

We leverage existing technology:

> Maven 2 for builds

> Spring for run time support

> Some conventions and light complements

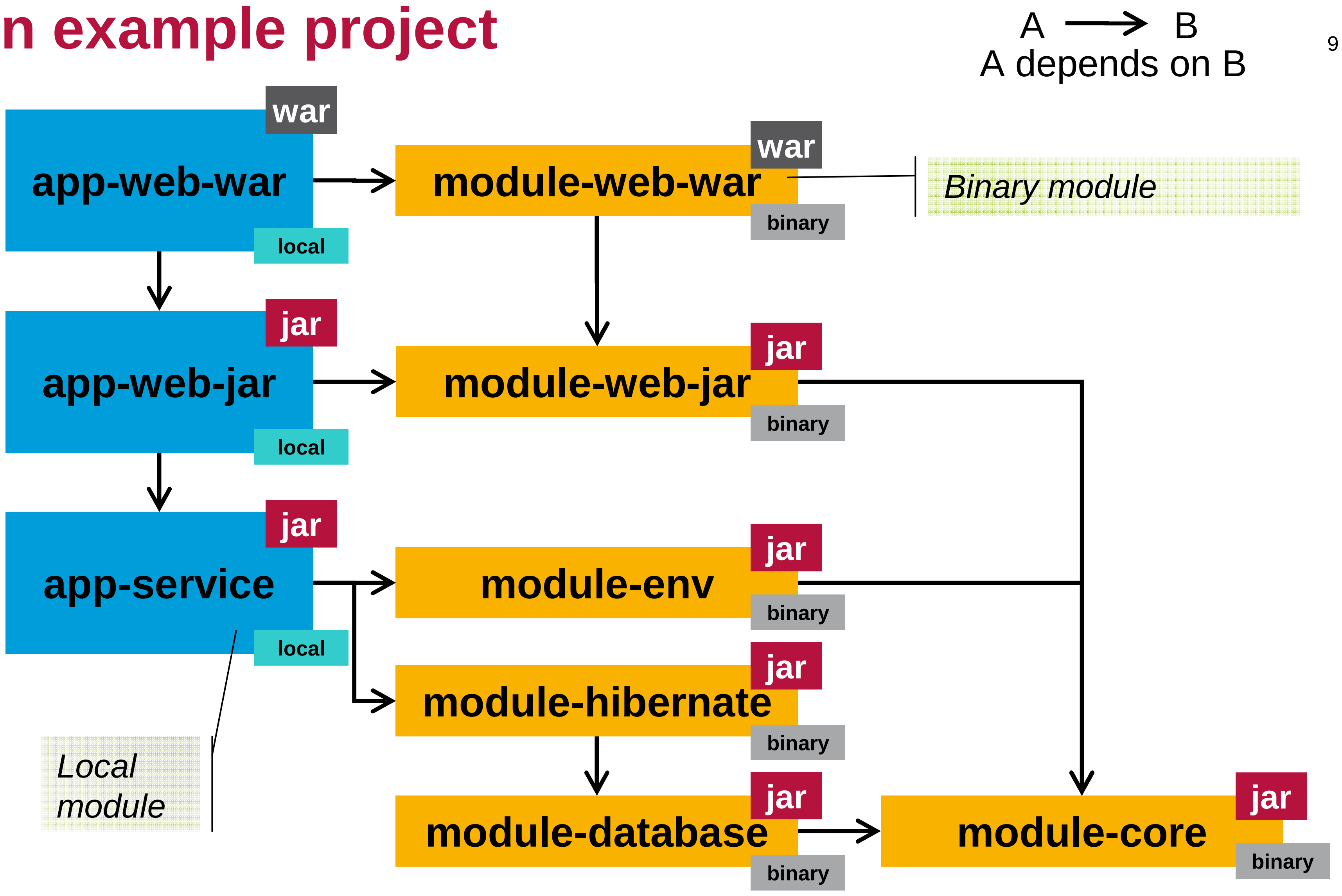
Key elements of our approach:

> Give each present module the possibility to “add Spring beans” to the global Spring configuration

> Dependency management
  – Controlled order of resources for build & runtime

> Share parameters between Maven and Spring

An example project



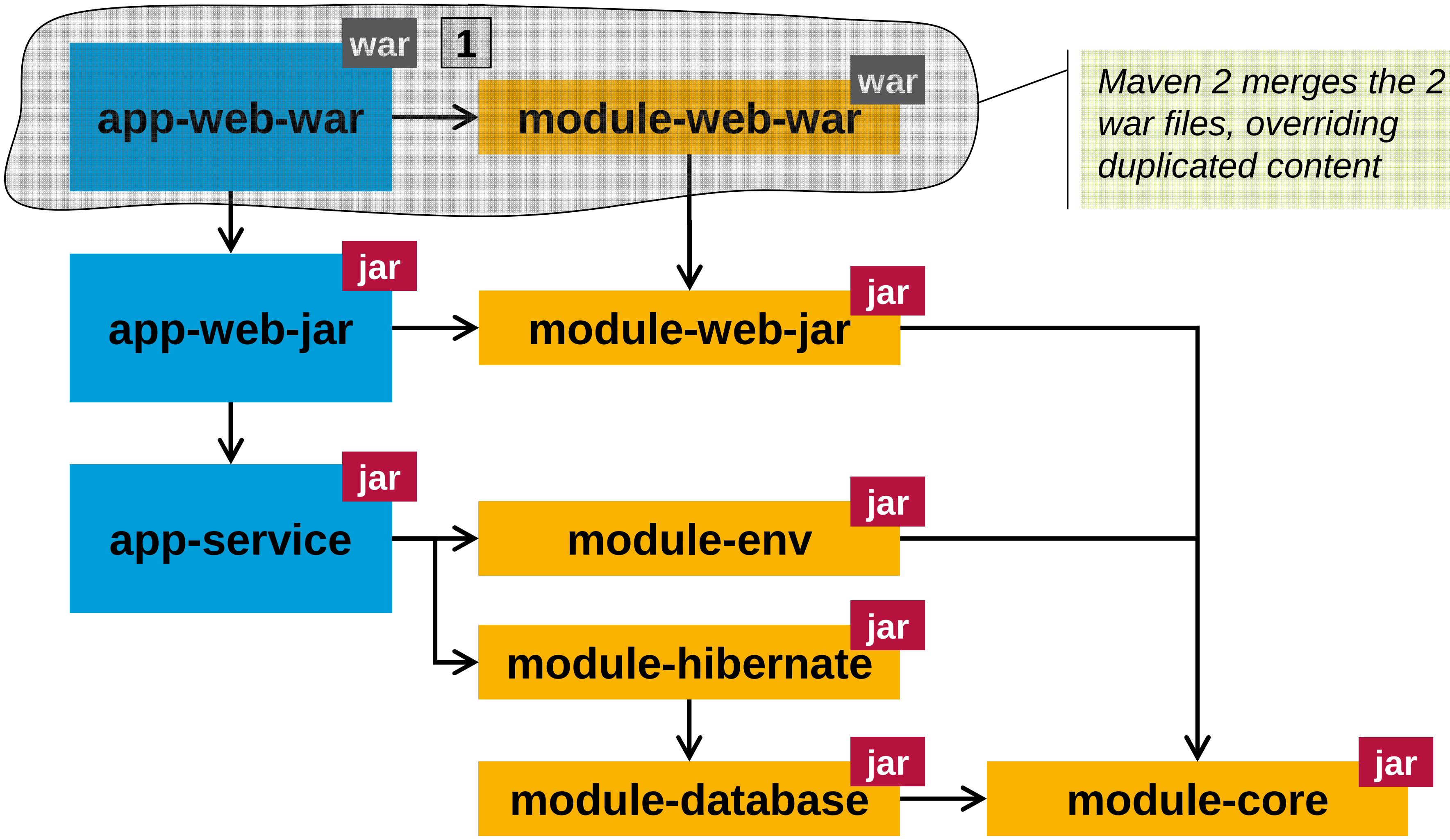
Dependency management

Idea:

> A depends on B (A $\longrightarrow$ B)
> implies also: A should be able to override resources of B

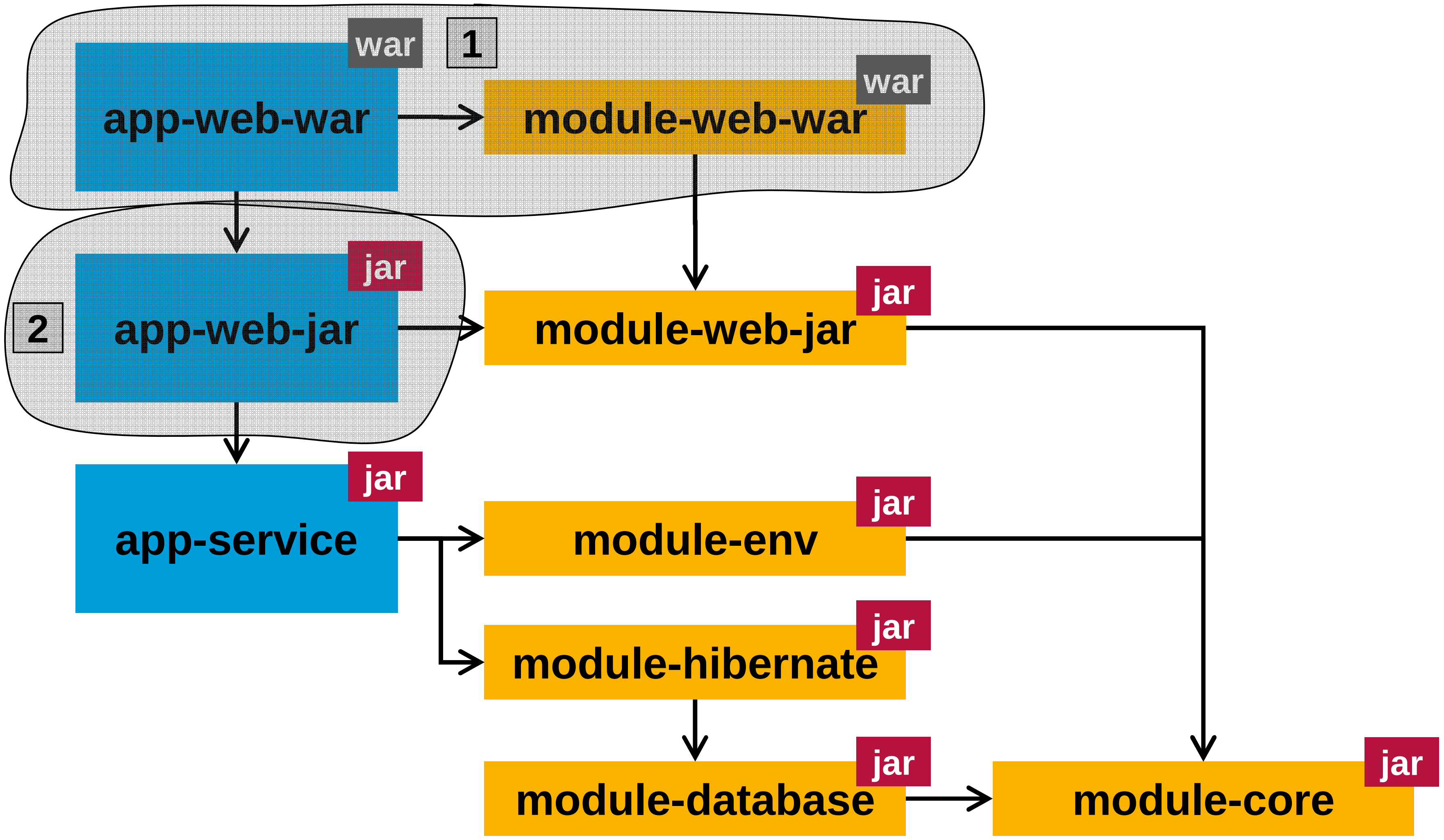
The next slides illustrate this with the example project

Module dependency graph – Layer 1

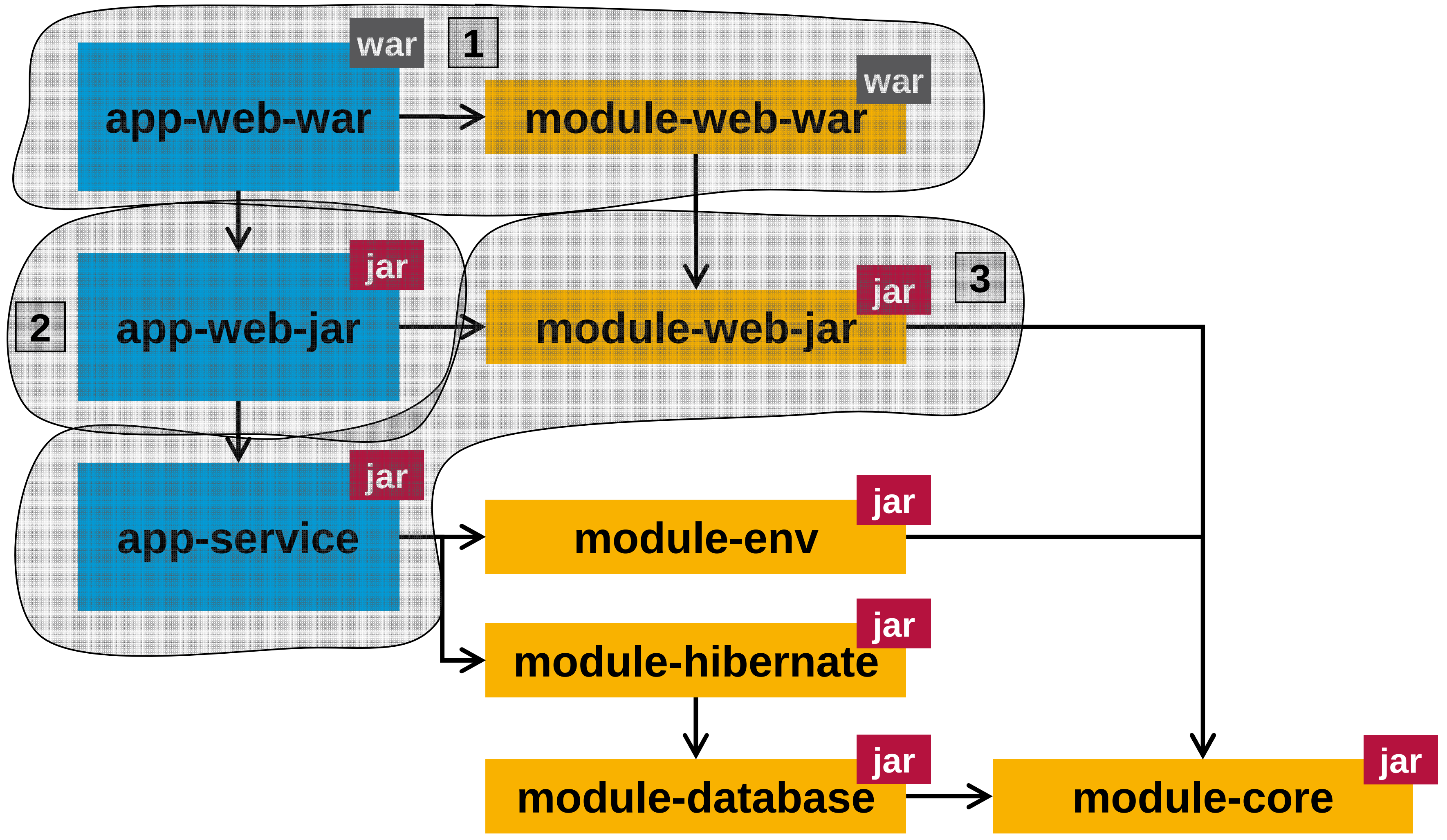


Maven 2 merges the 2 war files, overriding duplicated content

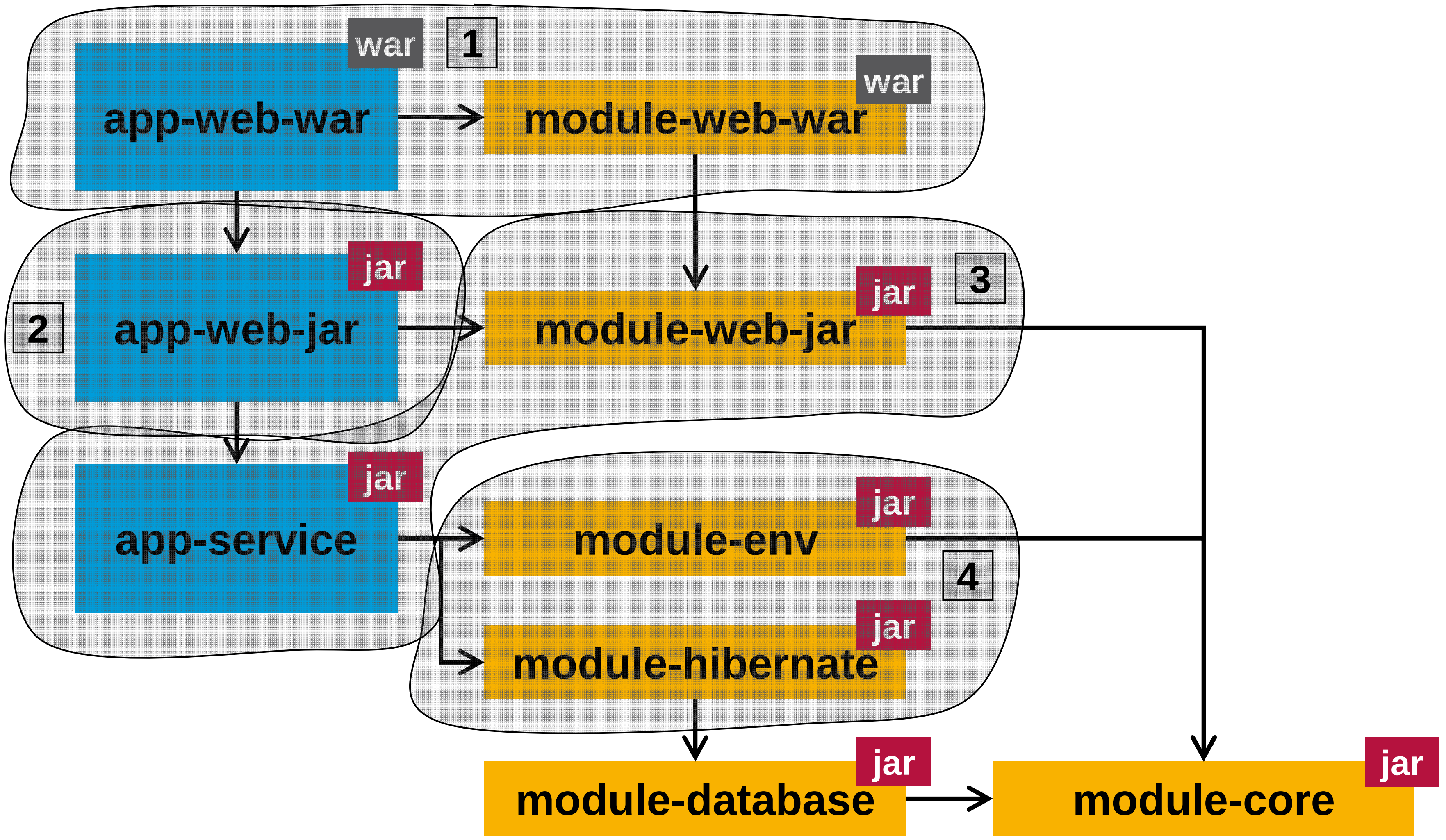
Module dependency graph – Layer 1 to 2



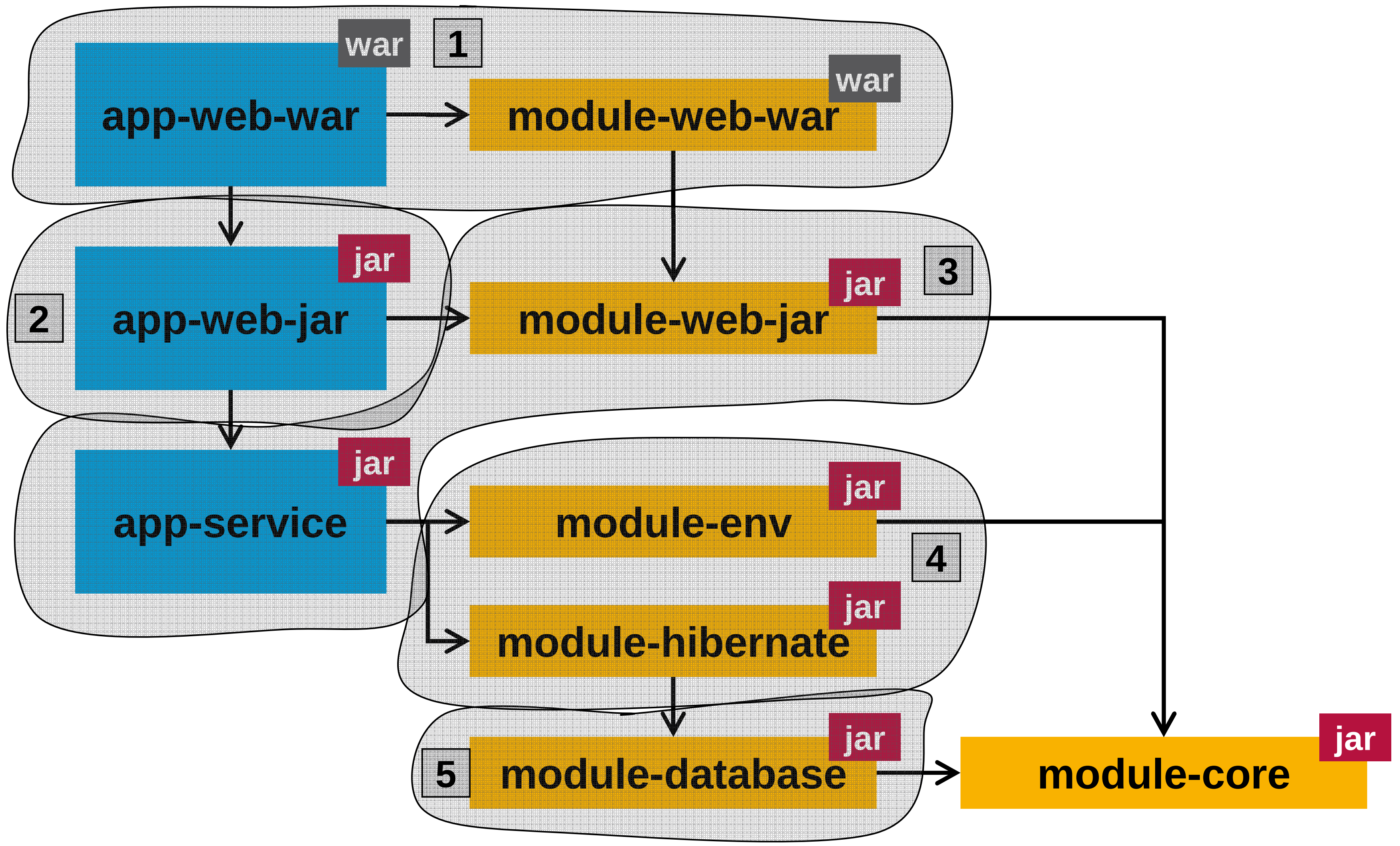
Module dependency graph – Layer 1 to 3



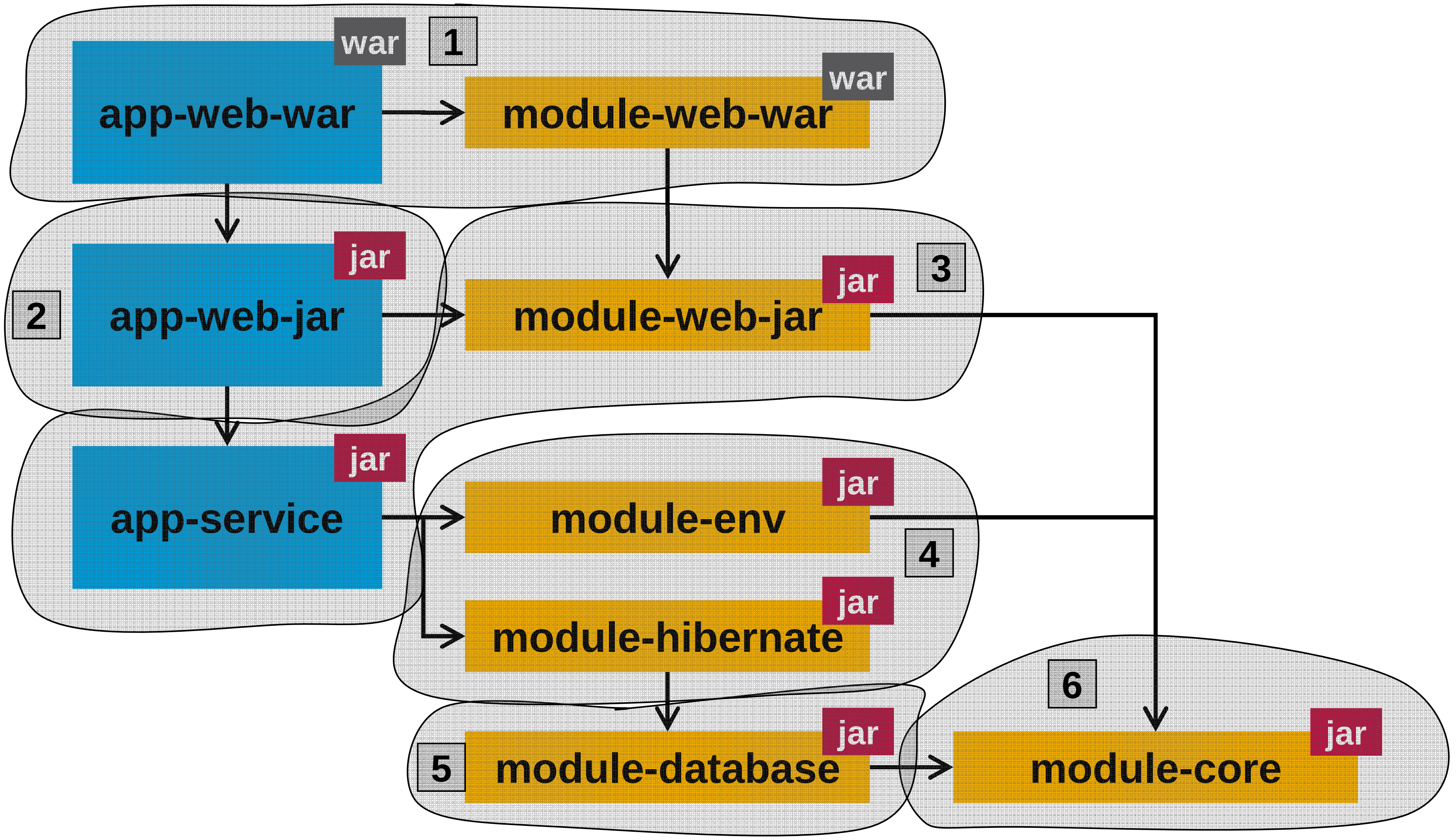
Module dependency graph – Layer 1 to 4



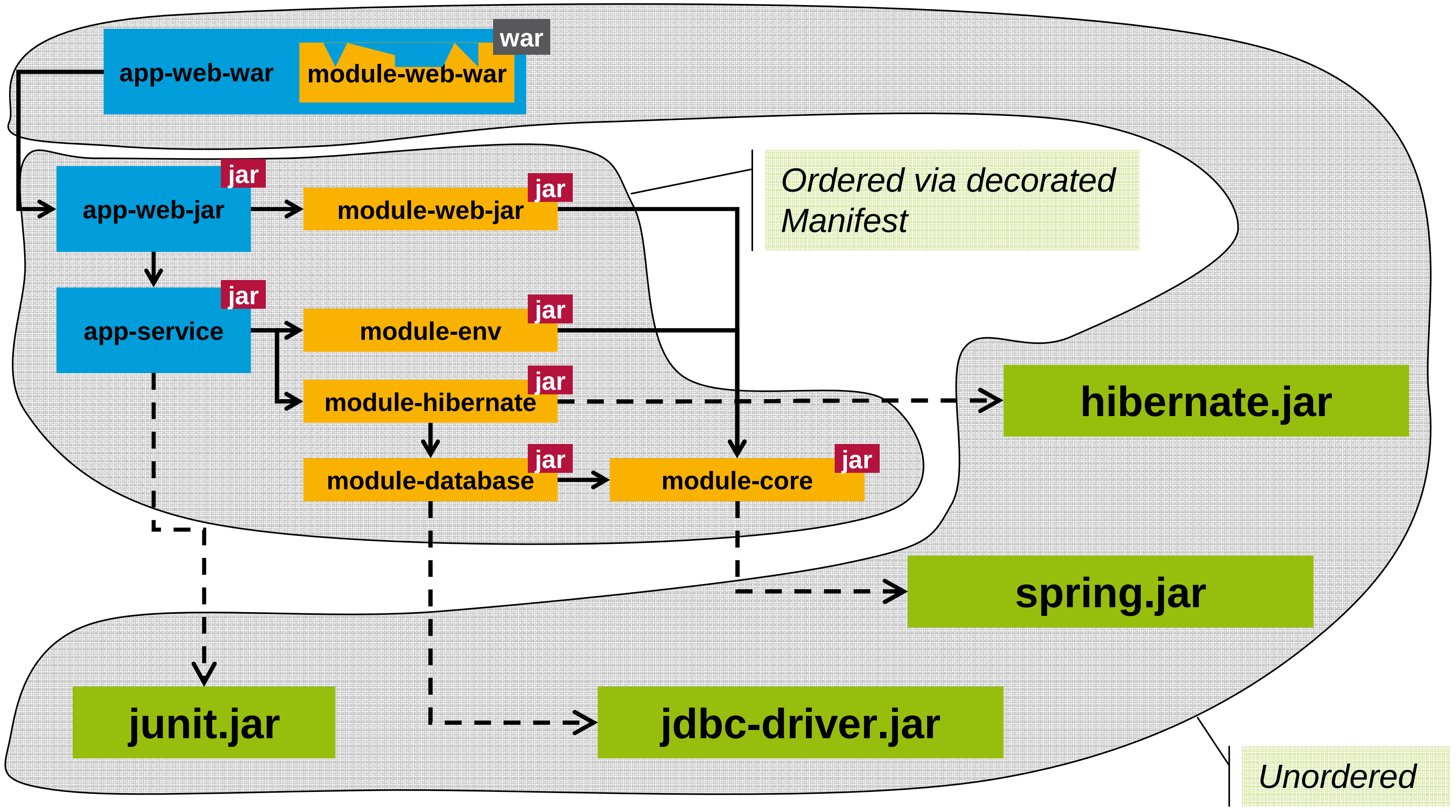
Module dependency graph – Layer 1 to 5



Module dependency graph – Layer 1 to 6



Order of resources from classpath



Structure for resources inside a module

Spring configuration files

mandatory/
    main-app-services.xml
scenarios/
    db/raw/
        simple-db-config.xml
    dataaccess/hibernate
        central-hibernate-config.xml
optional/
    interception/
        java5-transaction-annotations.xml

Other resources

etc/
    sql/oracle/
        create-app-core.sql
        drop-all.sql

Application Context for Modules in NON-WEB (i.e. unit-tests)

```
ApplicationContext ctx = new ModuleApplicationContext(new String[] {
    "classpath*:mandatory/*.xml",
    "classpath*:scenarios/db/raw/*.xml",
    "classpath*:scenarios/dataaccess/hibernate/*.xml",
    "classpath*:optional/interception/java5-transaction-annotations.xml“}, …);
```

Application Context for Modules in WEB

web.xml:

```
<context-param>
      <param-name>inclusiveLocations</param-name>
      <param-value>
            classpath*:mandatory/*.xml,
            classpath*:scenarios/db/raw/*.xml,
            classpath*:scenarios/dataaccess/hibernate/*.xml,
            classpath*:optional/interception/java5-transaction-annotations.xml
      </param-value>
</context-param>

<listener>
      <listener-class>ch.elca.el4j.web.context.ModuleContextLoaderListener</listener-class>
</listener>
```

Multi-Environment configuration

Parameters for the environment (db name, db connection strings, TCP ports, hostnames, web contexts, …) are often required in Maven (build & launch time) **and** in Spring (run time).

The environment support makes this information available to both.

Locations to define properties in Maven 2 (lowest to highest precedence)

1. On command line (out of the box):
   mvn … -Drmi.host=140.211.11.130
2. pom.xml
3. pom.xml in <profile/>
4. settings.xml in <profile/>
5. On command line (with EL4J-patch):
   mvn … -D**override**.rmi.host=66.35.250.209

Sample pom.xml excerpt

pom.xml

```
…
<properties>
   <rmi.host>localhost</rmi.host>
   …
   <db.url>jdbc:oracle:thin:@${db.host}:${db.port}:${db.database-name}</db.url>
</properties>
```

Shared environment configuration

1) Sample properties (e.g., in a pom.xml file):

pom.xml

```
…
<properties>
   <rmi.host>localhost</rmi.host>
   …
   <db.url>jdbc:oracle:thin:@${db.host}:${db.port}:${db.database-name}</db.url>
</properties>
```

2) The properties for Spring are listed in one of these 2 files:

env-placeholder.properties

```
…
rmi.host=${rmi.host}
```

env-bean-property.properties

```
…
dataSource.url=${db.url}
```

Translation of Maven properties to placeholders (filtering)

3) Spring config that will be overridden:

a-spring-config-file.xml

```
…
<bean id=…>
   <property name=“serviceHost“ value=“${rmi.host}“/>
</bean>

<bean id=“dataSource“>
  <property name=“url">
     <value>myDefaultUrl</value>
  </property>
</bean>
```

Translation of Maven properties to bean properties (filtering)

Environment configuration - access from code

To access environment parameters from your code (during runtime, but outside of Spring) there are 2 methods in the class **`EnvPropertiesUtils`**:

> **`Properties getEnvBeanPropertyProperties()`**

> **`Properties getEnvPlaceholderProperties()`**

More info:

http://el4j.sourceforge.net/framework-modules/apidocs/ch/elca/el4j/util/env/EnvPropertiesUtils.html

Structure for resources inside a module

Spring configuration files

mandatory*/*

    main-app-services.xml

scenarios*/*

    db/raw/

        simple-db-config.xml

    dataaccess/hibernate

        central-hibernate-config.xml

optional*/*

    interception/

        java5-transaction-annotations.xml

Other resources

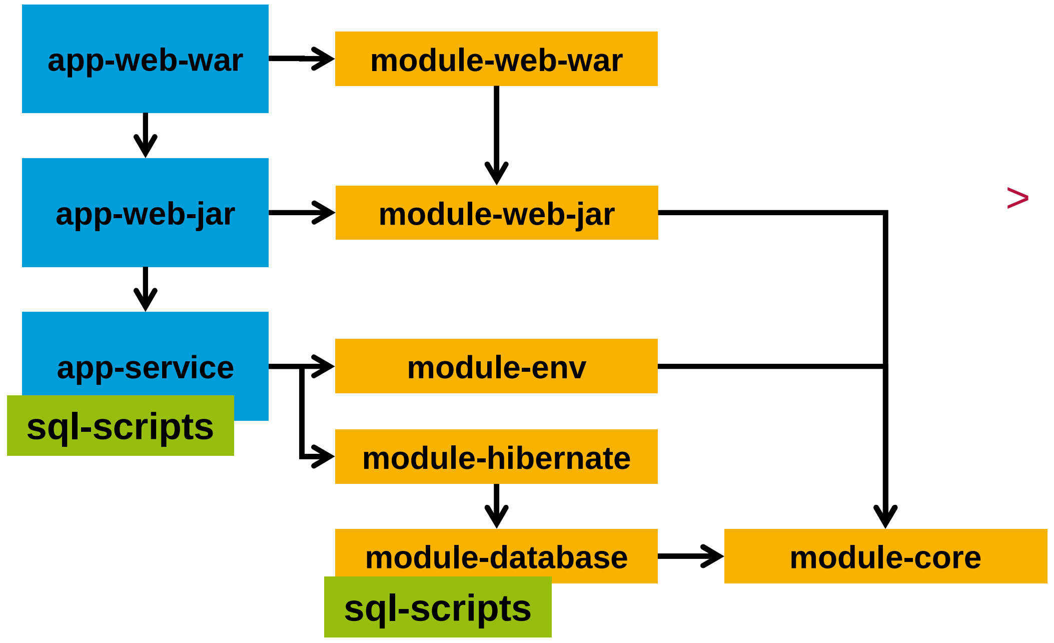
etc*/*

    sql/oracle/

        create-app-core.sql

        drop-all.sql

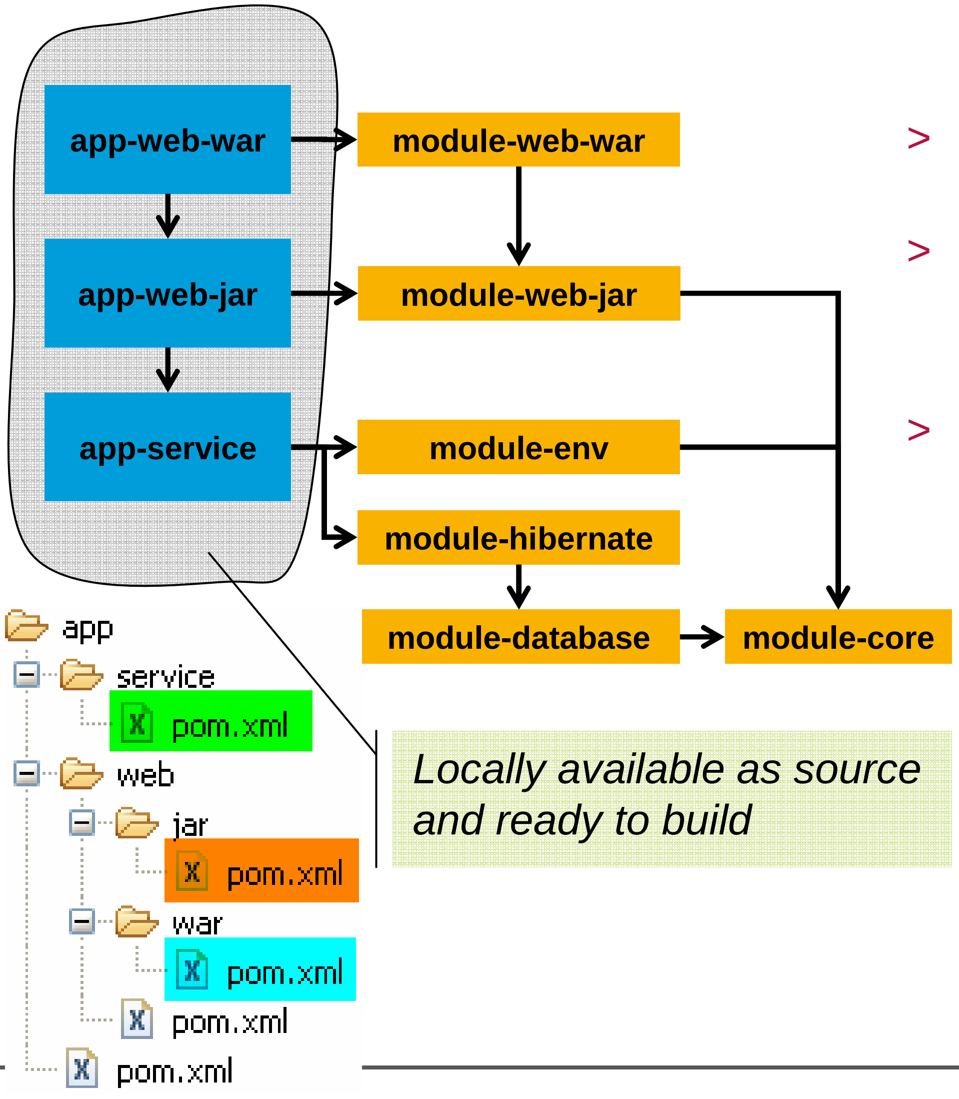
Database plugin – order of SQL script execution



Sql scripts on the same „dependency level“
are sorted alphapetically!

- Re-init the db
  - `cd app/web/war`
  - `mvn db:prepare`
    - ➔ First drop then create (see below)
- Some other mvn commands
  - `mvn db:drop`
    - ➔ Exec „drop-*.sql“ started from app-web-war
  - `mvn db:silentDrop`
    - ➔ Same as above but continues after failure
  - `mvn db:create`
    - ➔ Exec „create-*.sql“ started from module-core
  - `mvn db:insert`
    - ➔ Exec „insert-*.sql“ started from module-core
  - `mvn db:delete`
    - ➔ Exec „delete-*.sql“ started from app-web-war

Recursive Maven executions



> Executes Maven with given parameters on all dependencies that can be found locally.
> `mvnrec` command
– **`cd app/web/war`**
– **`mvnrec clean install -DskipTests=true`**
> Is equivalent to these „normal“ mvn commands:
– **`cd app/service`**
– **`mvn clean install -DskipTests=true`**
– **`cd ../web/jar`**
– **`mvn clean install -DskipTests=true`**
– **`cd ../war`**
– **`mvn clean install -DskipTests=true`**

Recursive Maven executions - Options

> Usage
  - `mvnrec [OPTIONS] MAVEN_COMMAND [MAVEN_COMMAND]`

> Options
  - **`-ff`**
    - ➔ fail-fast (interrupt the build at the first failure)
    - ➔ the default is to fail only at the end of the build (“fail at end”)
  - **`-b`**
    - ➔ force (re-)scanning of folders and creation of bootstrap-file (mvnrec caches dependencies for better performance)
  - **`-v`**
    - ➔ verbose mvnrec output

Our wish list

Maven 2 & Spring

> More seamless integration of Maven 2, Spring, multi-environment, …

> Maven 2 & its ecosystem

- More stability and fewer regressions
- Speed

> More flexible configuration overloading

More flexible module support (JSR 277 or OSGi will change this ☺)

> Allow a module to hide some classes (module-private classes)

> Just-in-time module loading/ module unloading/ module reloading/ allow conflicting module versions in one application => flexibility to adapt an application during runtime

Conclusion

We have seen a unified module support for project organization, development and run time, based on Maven 2, Spring, some helper classes and conventions

Gets us closer to an “ideal” development and run time environment

The coming module standards (OSGi on the server/ JSR 277) …

> … completes the runtime module abstraction
> … will make our benefits available to most of you (even if you can’t apply our patterns/ conventions yet)

Want to learn more/ want to try it out?

http://EL4J.sourceforge.net

More infos:

> Java One 2008 TS-6185: Modularity in the Java Platform

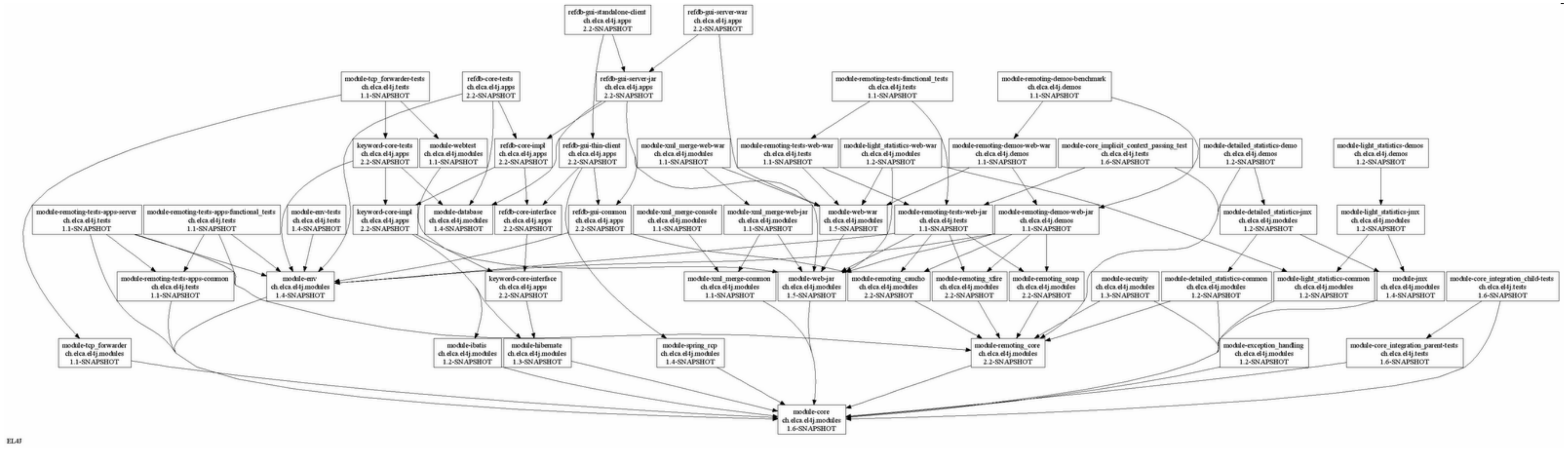
Any Questions?

Martin Zeltner
Philipp H. Oser
ELCA

http://www.elca.ch
<firstname>.<lastname>@elca.ch

EL4J http://EL4J.sourceforge.net

Dependency plugin


refdb-gui-standalone-client ch.elca.el4j.apps 2.2-SNAPSHOT
refdb-gui-server-war ch.elca.el4j.apps 2.2-SNAPSHOT
module-tcp_forwarder-tests ch.elca.el4j.tests 1.1-SNAPSHOT
refdb-core-tests ch.elca.el4j.apps 2.2-SNAPSHOT
refdb-gui-server-jar ch.elca.el4j.apps 2.2-SNAPSHOT
module-remoting-tests-functional_tests ch.elca.el4j.tests 1.1-SNAPSHOT
module-remoting-demos-benchmark ch.elca.el4j.demos 1.1-SNAPSHOT
keyword-core-tests ch.elca.el4j.apps 2.2-SNAPSHOT
module-webtest ch.elca.el4j.modules 1.1-SNAPSHOT
refdb-core-impl ch.elca.el4j.apps 2.2-SNAPSHOT
refdb-gui-thin-client ch.elca.el4j.apps 2.2-SNAPSHOT
module-xml_merge-web-war ch.elca.el4j.modules 1.1-SNAPSHOT
module-remoting-tests-web-war ch.elca.el4j.tests 1.1-SNAPSHOT
module-light_statistics-web-war ch.elca.el4j.modules 1.2-SNAPSHOT
module-remoting-demos-web-war ch.elca.el4j.demos 1.1-SNAPSHOT
module-core_implicit_context_passing_test ch.elca.el4j.tests 1.6-SNAPSHOT
module-detailed_statistics-demo ch.elca.el4j.demos 1.2-SNAPSHOT
module-light_statistics-demos ch.elca.el4j.demos 1.2-SNAPSHOT
module-remoting-tests-apps-server ch.elca.el4j.tests 1.1-SNAPSHOT
module-remoting-tests-apps-functional_tests ch.elca.el4j.tests 1.1-SNAPSHOT
module-env-tests ch.elca.el4j.tests 1.4-SNAPSHOT
keyword-core-impl ch.elca.el4j.apps 2.2-SNAPSHOT
module-database ch.elca.el4j.modules 1.4-SNAPSHOT
refdb-core-interface ch.elca.el4j.apps 2.2-SNAPSHOT
refdb-gui-common ch.elca.el4j.apps 2.2-SNAPSHOT
module-xml_merge-console ch.elca.el4j.modules 1.1-SNAPSHOT
module-xml_merge-web-jar ch.elca.el4j.modules 1.1-SNAPSHOT
module-web-war ch.elca.el4j.modules 1.5-SNAPSHOT
module-remoting-tests-web-jar ch.elca.el4j.tests 1.1-SNAPSHOT
module-remoting-demos-web-jar ch.elca.el4j.demos 1.1-SNAPSHOT
module-detailed_statistics-jmx ch.elca.el4j.modules 1.2-SNAPSHOT
module-light_statistics-jmx ch.elca.el4j.modules 1.2-SNAPSHOT
module-remoting-tests-apps-common ch.elca.el4j.tests 1.1-SNAPSHOT
module-env ch.elca.el4j.modules 1.4-SNAPSHOT
keyword-core-interface ch.elca.el4j.apps 2.2-SNAPSHOT
module-xml_merge-common ch.elca.el4j.modules 1.1-SNAPSHOT
module-web-jar ch.elca.el4j.modules 1.5-SNAPSHOT
module-remoting_caucho ch.elca.el4j.modules 2.2-SNAPSHOT
module-remoting_xfire ch.elca.el4j.modules 2.2-SNAPSHOT
module-remoting_soap ch.elca.el4j.modules 2.2-SNAPSHOT
module-security ch.elca.el4j.modules 1.3-SNAPSHOT
module-detailed_statistics-common ch.elca.el4j.modules 1.2-SNAPSHOT
module-light_statistics-common ch.elca.el4j.modules 1.2-SNAPSHOT
module-jmx ch.elca.el4j.modules 1.4-SNAPSHOT
module-core_integration_child-tests ch.elca.el4j.tests 1.6-SNAPSHOT
module-tcp_forwarder ch.elca.el4j.modules 1.1-SNAPSHOT
module-ibatis ch.elca.el4j.modules 1.2-SNAPSHOT
module-hibernate ch.elca.el4j.modules 1.3-SNAPSHOT
module-spring_rcp ch.elca.el4j.modules 1.4-SNAPSHOT
module-remoting_core ch.elca.el4j.modules 2.2-SNAPSHOT
module-exception_handling ch.elca.el4j.modules 1.2-SNAPSHOT
module-core_integration_parent-tests ch.elca.el4j.tests 1.6-SNAPSHOT
module-core ch.elca.el4j.modules 1.6-SNAPSHOT
EL4J