ARS

Member of TIMETOACT GROUP SOFTWARE & CONSULTING

Paketierung & Classloader

Java EE Grundlagen

ARS

Inhalte dieses Kapitels

Paketierung

Classloader

Zusammenfassung

ARS

Paketierung [1|8]

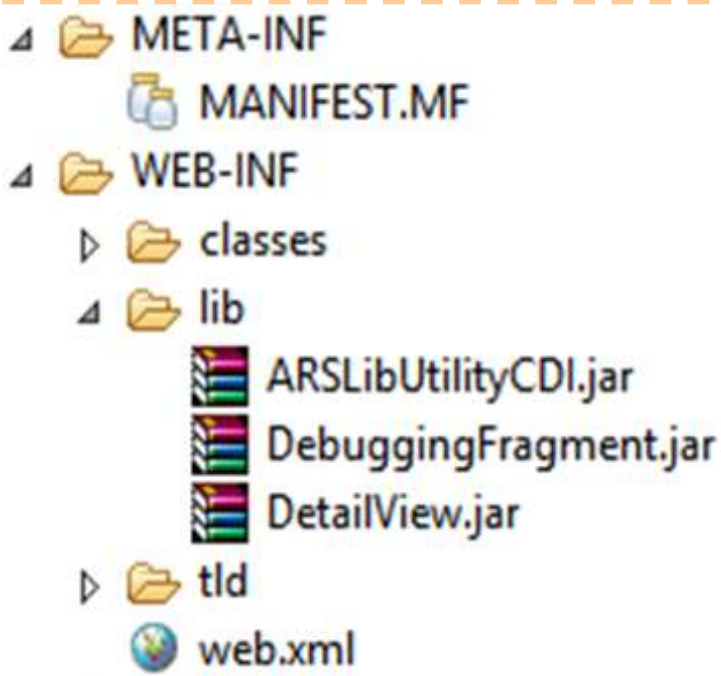
Aufbau von Enterprise-Anwendungen (*.ear)

ARS

Paketierung [2|8]

Webmodule (`*.war`)

internal resources



Manifest-Datei

Java-Klassen, Anwendungsressourcen

Webbibliotheken

Web Deployment Descriptor

Öffentliche Webressourcen

ARS

Paketierung [3|8]

Webmodule – Ablage besonderer Artefakte

- Java Server Pages
  - unter `WEB-INF` (z.B. `WEB-INF/jsp`)
    → Keine direkte Adressierung per URL möglich
    → Erreichbarkeit über interne Weiterleitung
    → Erreichbarkeit über URL-Mapping (Web Deployment Descriptor)
  - außerhalb von `WEB-INF`
    → Direkte Adressierung per URL möglich
    → Interne Weiterleitung und URL-Mapping zusätzlich möglich
- Tag Library Descriptors
  - unter `WEB-INF` (direkt oder in Unterordner, z.B. `WEB-INF/tld`)
    → Keine direkte Adressierung per URL (empfohlen)
  - Implizites Mapping (Vorraussetzung: `<uri>` in TLD)
  - Explizites Mapping (Eintrag in Web DD)

ARS

Paketierung [4|8]

Webmodule – Webbibliotheken

- Ablage unter `WEB-INF/lib/*.jar`
- Inhalte
  - Java-Klassen
  - Anwendungsressourcen
  - Erweiterung von Metadaten (Pluggability)
    - Webmodul-Fragmente
      - Servlets, Filter u.a.
      - Webressourcen (HTML, CSS, JSPs)
      - Tag Libraries
    - EJBs
    - JSF-Komponenten
    - (Weitere Framework-Komponenten)

ARS

Paketierung [5|8]

Webmodule – Webbibliotheken

	Webanwendung (`.war`)	**Webbibliothek** (`.jar` in `WEB-INF\lib`)
Bytecode, Anw.-Ressourcen	`WEB-INF/classes/**`	direkt im JAR
Webressourcen (per URL erreichbar)	direkt im WAR (Ausnahme: `WEB-INF/*`)	`META-INF/resources/**`
Manifest	`META-INF/MANIFEST.MF`	
TLDs	`WEB-INF/**/*.tld`	`META-INF/**/*.tld`
Web (Fragment) DD	`WEB-INF/web.xml`	`META-INF/web-fragment.xml`
JSF Konfiguration	`WEB-INF/faces-config.xml`	`META-INF/faces-config.xml`
EJB DD	`WEB-INF/ejb-jar.xml`	

ARS

Paketierung [6|8]

Aufbau weiterer Module

- Connector-Module: `.rar`-Dateien (*Resource Adapter*)
- Sonstige Module: `.jar`-Dateien
  - EJB-Module
  - Application-Client-Module
  - Utility JARs
- Aufbau analog
  - Bytecode und Anwendungsressourcen direkt im Archiv
  - Manifest-Datei und Deskriptoren unter `META-INF`

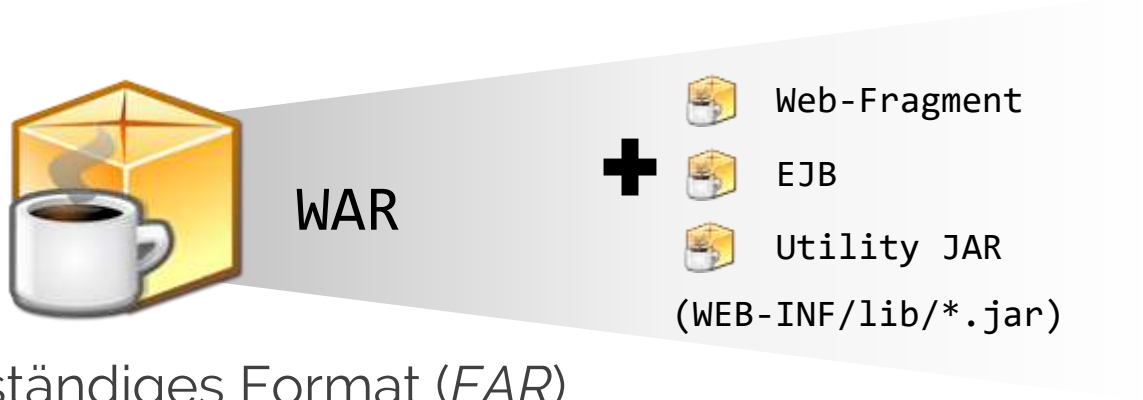
Modul-Typ	Deployment Deskriptor
EJB	`META-INF/ejb-jar.xml`
Application Client	`META-INF/application-client.xml`
Connector	`META-INF/ra.xml`

ARS

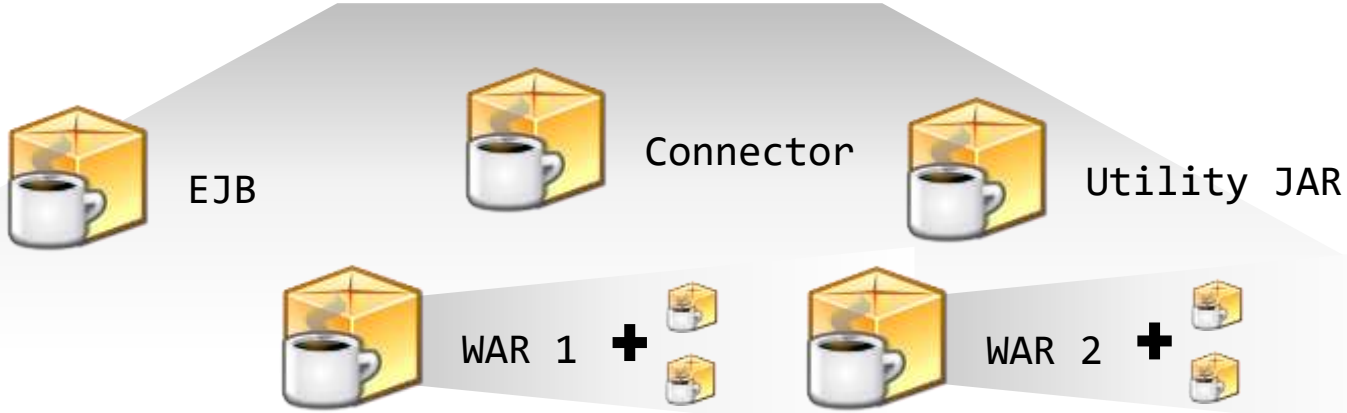
Paketierung [7|8]

Auslieferung von Enterprise-Anwendungen

- Vereinfachtes Format für Webanwendungen (*Single WAR*)



- Vollständiges Format (*EAR*)



ARS

Paketierung [8|8]

Vollständiges Format (`.ear`)

- ZIP-Format
  - Module (Web-, EJB-, Application-Client-, Connector)
  - Gemeinsam genutzte Bibliotheken (Utility JARs)
    - unter `/lib` (implizite Abhängigkeit aller Module)
    - andere Position (explizite Abhängigkeit über Manifest-Datei)
  - Application Deployment Descriptor (`META-INF/application.xml`)
    - Auflistung der Module (bei Webmodulen: *Context Root*)
    - Security Roles
    - Application Level Resources (JNDI Namespace `java:app/env`)
      - Data Sources, JMS Destinations
      - EJBs, JPA Persistence Contexts
      - Webservices
      - Umgebungsvariablen

ARS

Classloader-Hierarchien [1|5]

Classloader

- Teil der JVM zum Laden von
  - Java-Bytecode
  - Anwendungsressourcen

```
Konto.class
   .getResourceAsStream("/de/ars/bank/ui/konto.gif")
Classloader.getSystemClassloader()
   .getResourceAsStream("de/ars/bank/ui/konto.gif")
```

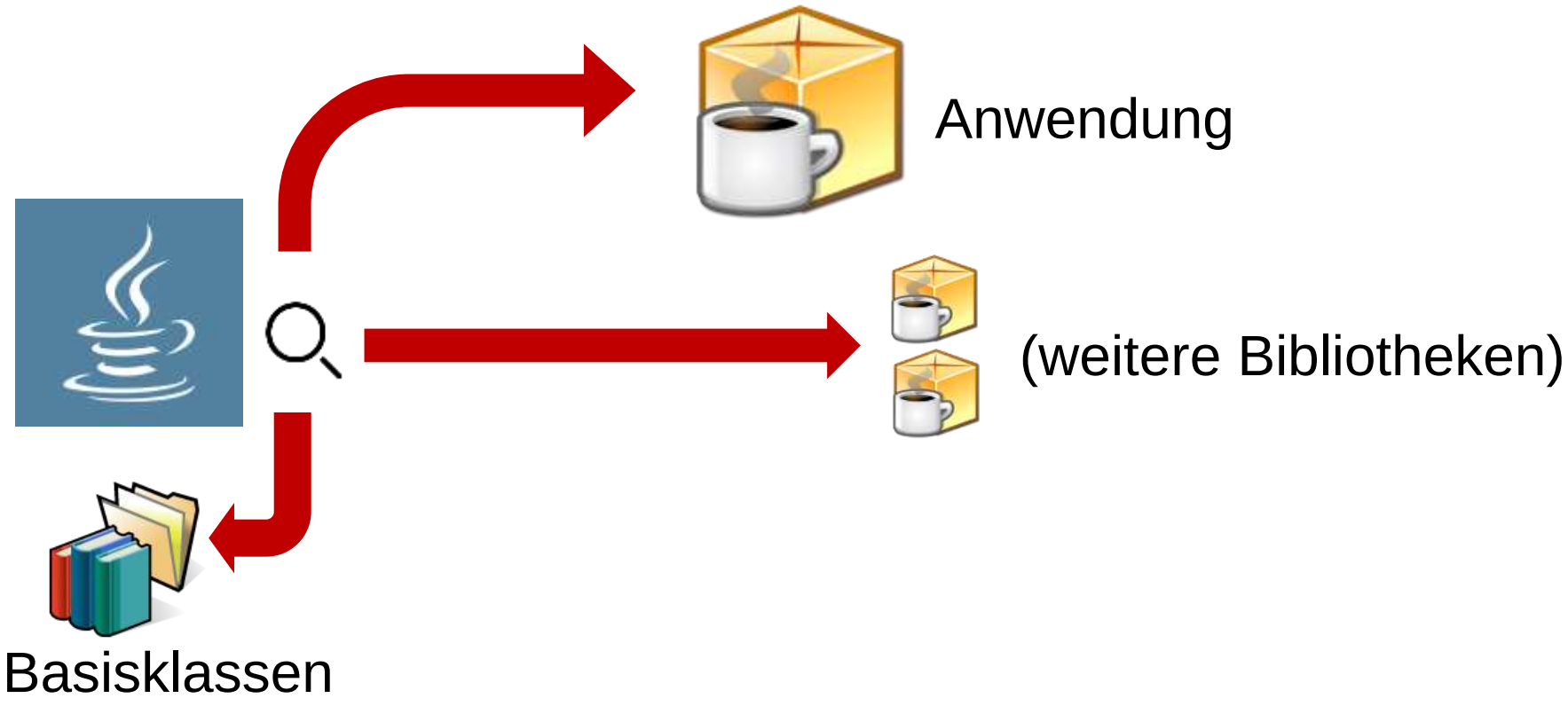
- Konfiguration über *Classpath*
  - Liste von Standard-Verzeichnissen und JARs
  - Erweiterung über Manifest-Dateien, Kommandozeilenparameter, …
- Anwendungsfall: *Service Provider Mechanism*

ARS

Classloader-Hierarchien [2|5]

Classloader in Standalone-Anwendungen

- Ein Classloader für eine Anwendung in einer JVM (*vereinfacht*)

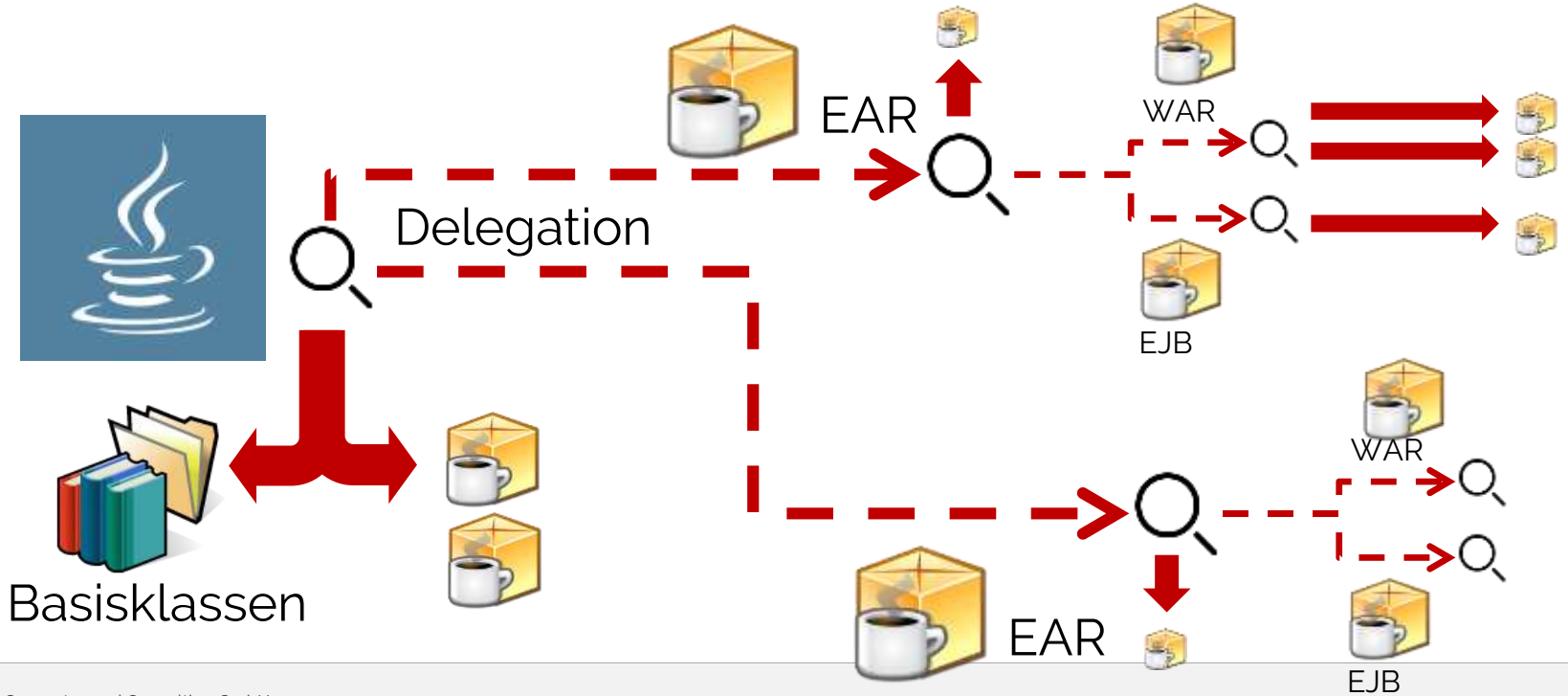


ARS

Classloader-Hierarchien [3|5]

Classloader beim Application Server

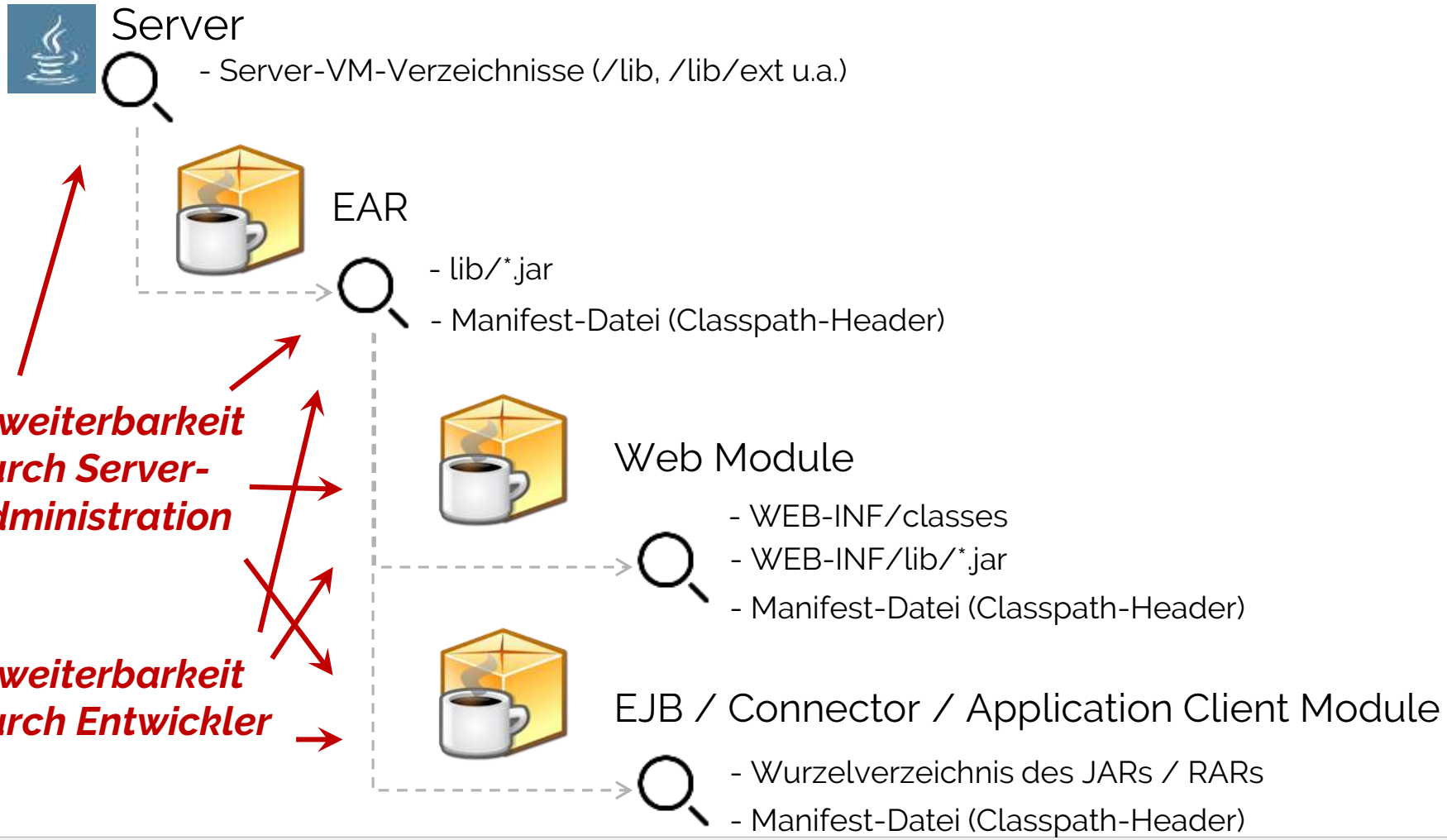
- Eine JVM für mehrere Enterprise-Anwendungen gemeinsam
- Fokus: Isolation der Anwendungen über getrennte Classloader
  - Classpath-Konfiguration
  - Laden von Klassen → statische Variablen (z.B. Singletons)



ARS

Classloader-Hierarchien [4|5]

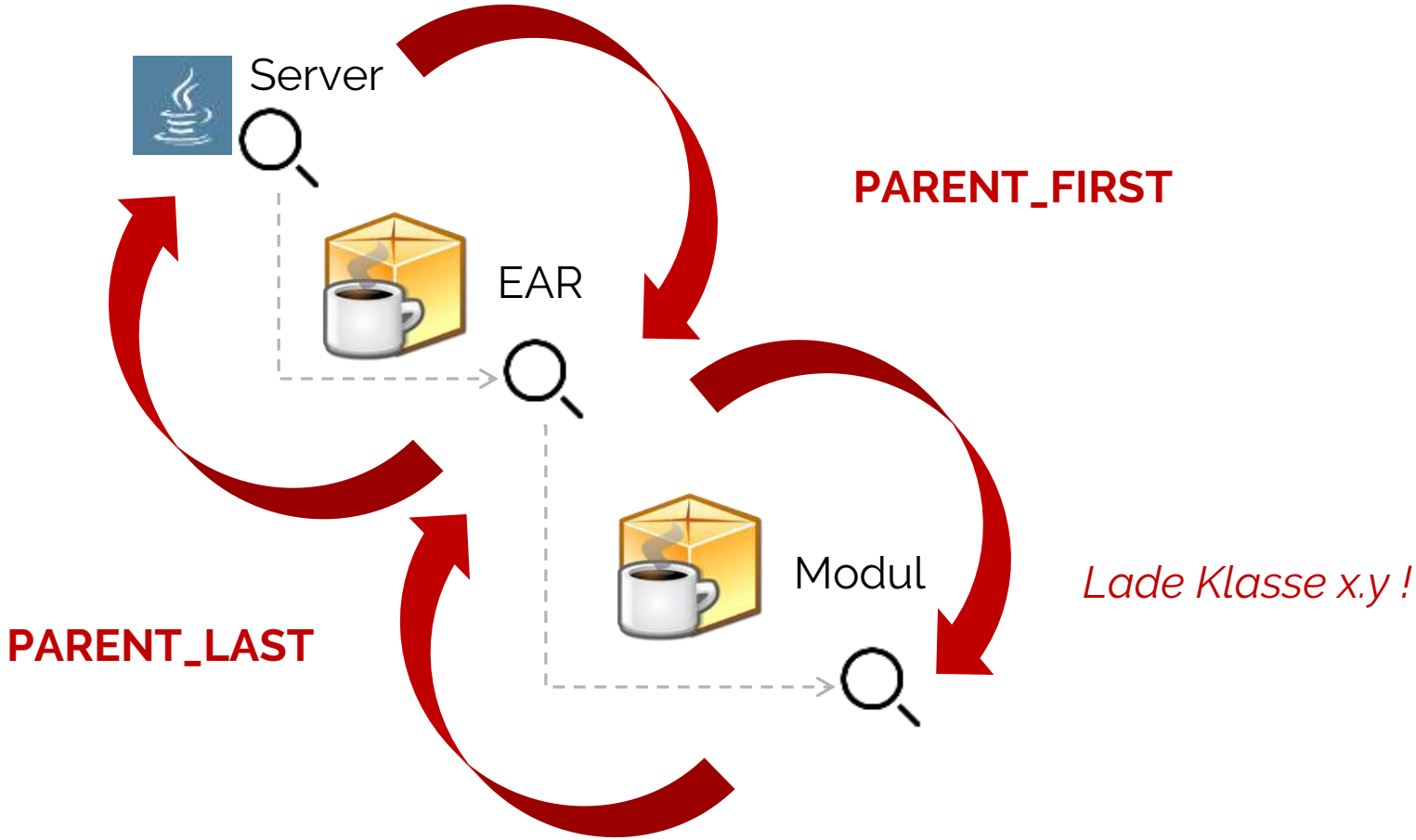
Classpath – Standardwerte



ARS

Classloader-Hierarchien [5|5]

Classloader Modes



ARS

EJBs: Paketierung im Webmodul oder separat

- EJBs in Webmodul
  - Direkt (`WEB-INF/classes`) oder als Webbibliothek (`WEB-INF/lib`)
  - Vereinfachtes Packaging (*Single WAR*)
- EJBs als eigenständiges Modul
  - Vollständiges Packaging im EAR erforderlich
  - Höherer Grad an Isolation

	Teil des Webmoduls	Eigenständiges Modul
Classloader	1 gemeinsamer Classloader für Webmodul und EJBs	Eigener Classloader für EJB-Modul (Isolation)
JNDI-Namespace	1 gemeinsamer Namespace für Webmodul und EJBs	Eigener Namespace für EJBs
EJB DD	`WEB-INF/ejb-jar.xml`	`META-INF/ejb-jar.xml`

ARS

Kontrollfragen

- Wie sind Enterprise-Anwendungen aufgebaut?
- Warum ist Classloader-Isolation auf einem Application Server wichtig?
- Sie haben am Server eine Bibliothek ausgetauscht, die in Ihrer Anwendung vom EAR-Classloader geladen wird. Welche Aussage ist korrekt?
  - Der Server muss neu gestartet werden.
  - Die Anwendung muss neu gestartet werden.
  - Weder noch, die Bibliothek wird automatisch neu geladen.
- Was ist der Unterschied zwischen den *Classloader Modes* `PARENT_FIRST` und `PARENT_LAST`?

ARS

Kontrollfragen

- Als Server-Administrator bemerken Sie, dass in einer installierten Webanwendung die Servlet-API, die ja bereits vom Server bereitgestellt wird, aus Versehen in der Webanwendung als Webbibliothek mitverpackt wurde. Welche Aussage ist richtig?
  - *Classloader Mode* `PARENT_FIRST` ist erforderlich.
  - *Classloader Mode* `PARENT_LAST` ist erforderlich.
  - Der *Classloader Mode* ist in diesem Fall nicht relevant.

ARS

Kontrollfragen

- Sie entwickeln eine Webanwendung mit einigen Webbibliotheken unter `WEB-INF/lib`. Sie kompilieren und testen diese erfolgreich in Ihrer lokalen Entwicklungsumgebung. Nach Erstellen des EARs und Deployment auf einem anderen Zielsystem erhalten Sie beim Start der Anwendung jedoch einen

  `java.lang.IncompatibleClassChangeError` bzw. eine

  `java.lang.NoSuchMethodException`

  bei Verwendung einer der Klassen aus den Webbibliotheken.

  Was könnte die Ursache sein?