

12.–15.09.2010  
in Nürnberg



# Herbstcampus

Wissenstransfer  
par excellence

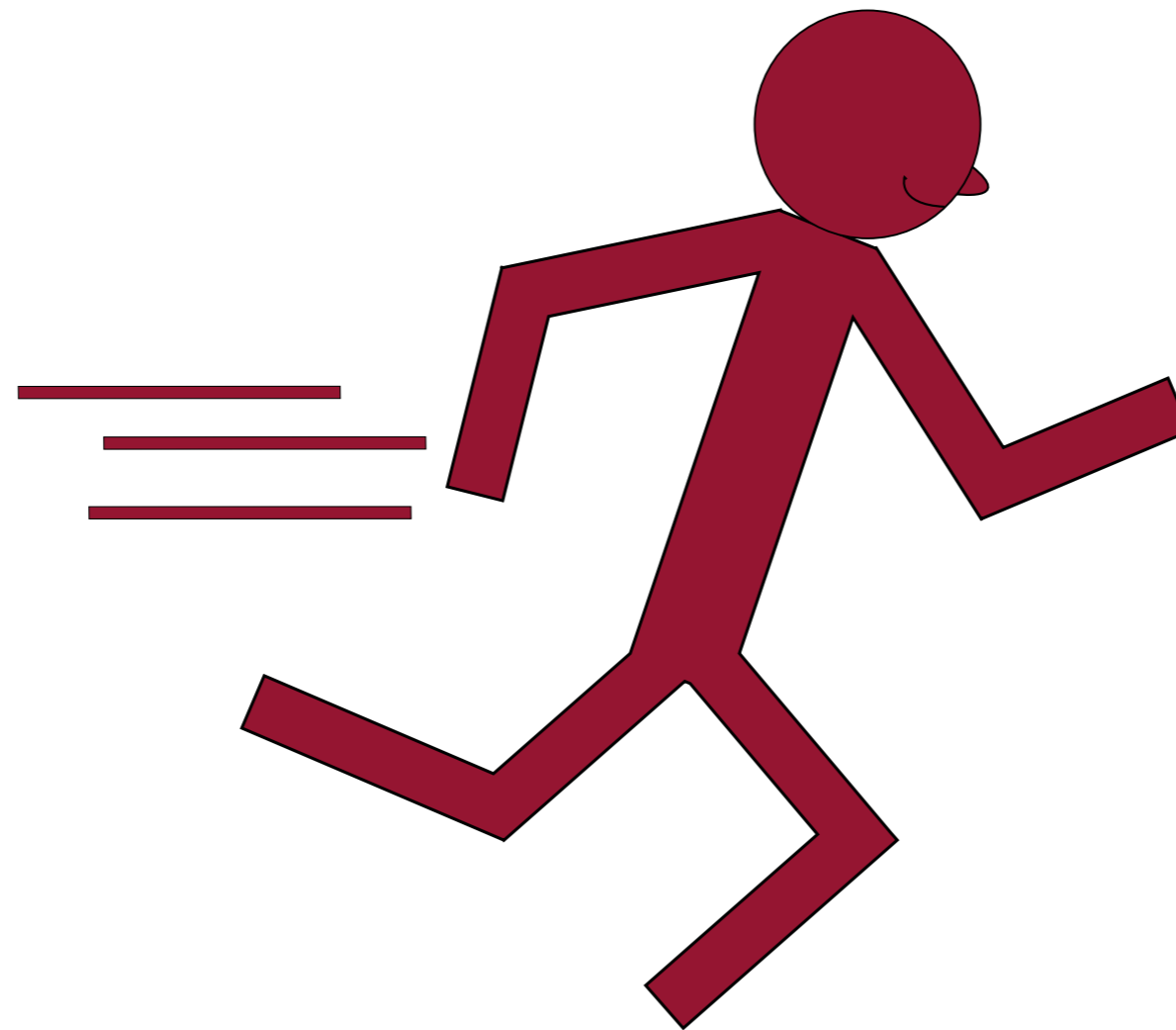
## OSGI plus X in Eclipse

Equinox

Erwin Hösch  
MATHEMA Software GmbH

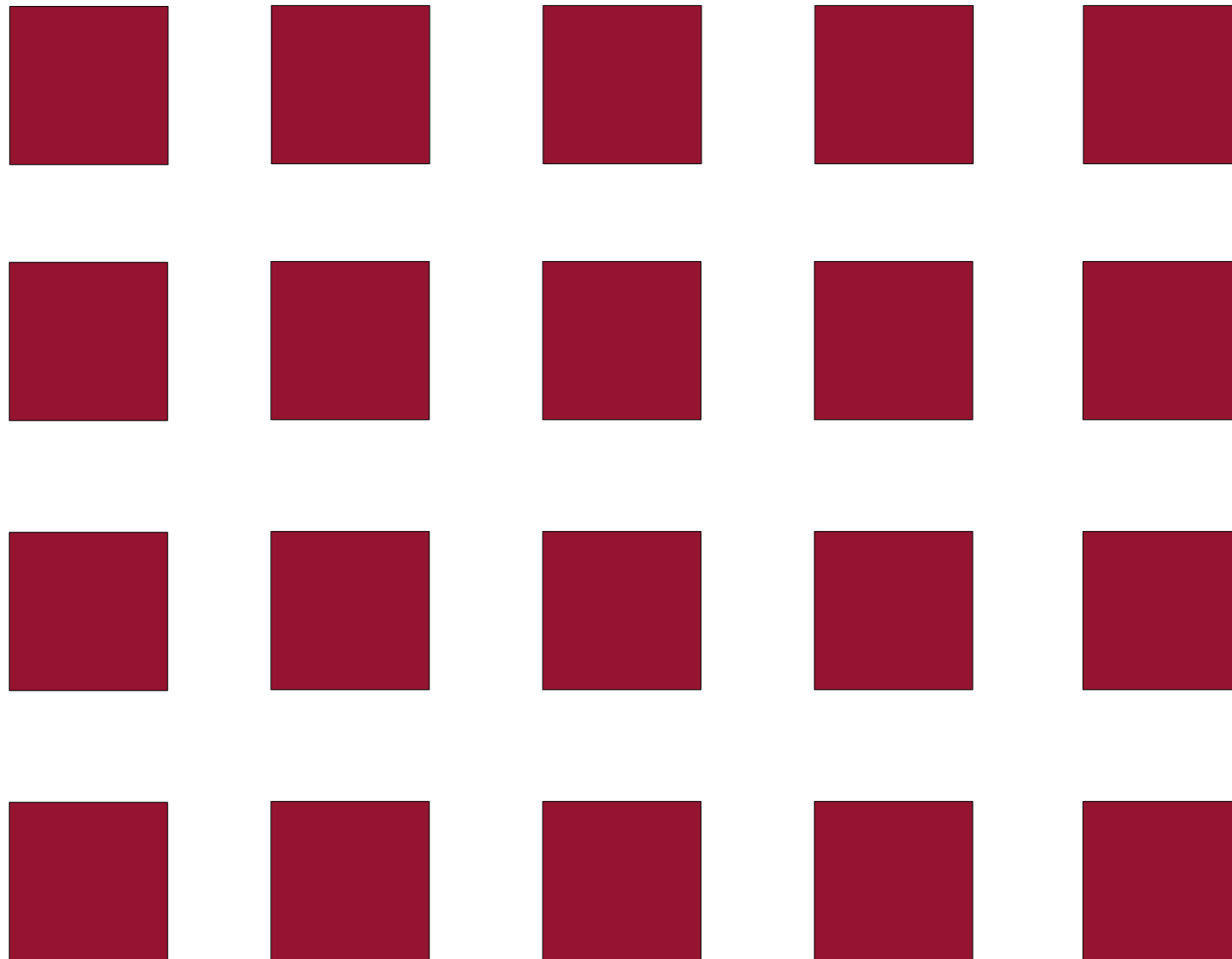
# Dynamik

---



# Modularisierung

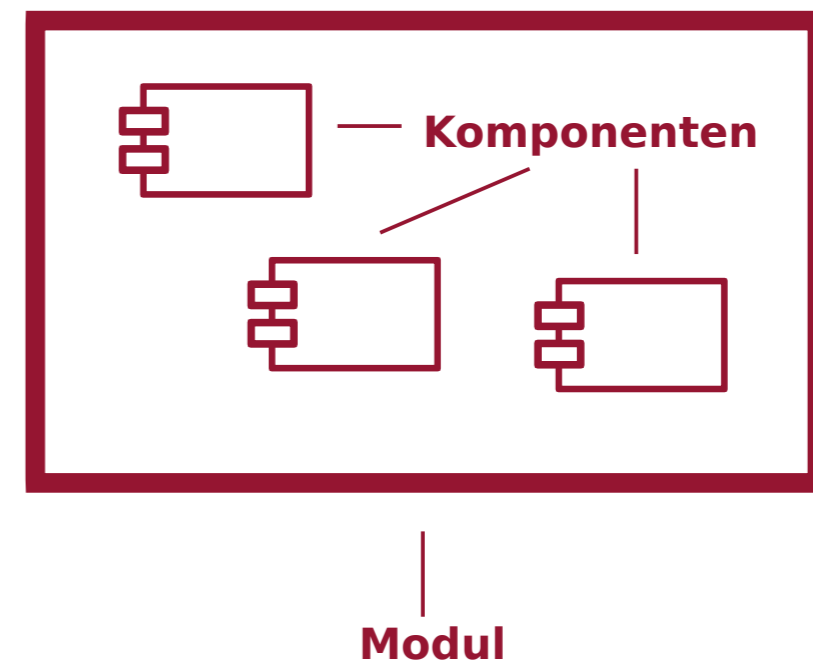
---



# Modul versus Komponente

---

- Modul
  - Deploymentartefakt
  - Abhängigkeiten
  - Konfiguration
- Komponente
  - Threading
  - Transaktion
  - Schnittstellen
  - Programmiermodell



# Modularisierung in Java

---

- Möglichkeiten
  - Klassen
  - Packages
  - Archive (.jar, .ear, .war, .zip)
- Probleme
  - JAR hell
  - Unkontrollierbare Abhängigkeiten
- Es fehlt eine Klammerung zwischen Package und Archive

# Modul

---

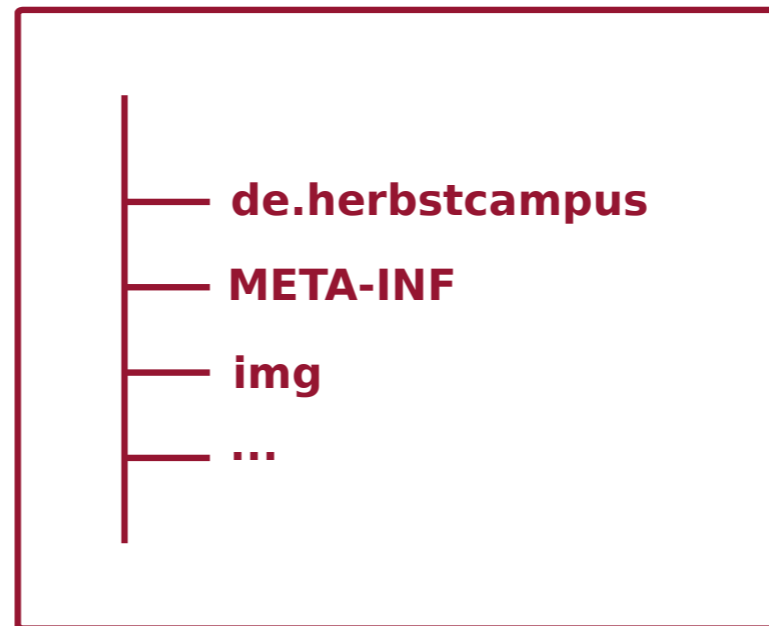
- Lose Kopplung
- Öffentliche Schnittstelle
- Definiert und verwaltet Abhängigkeiten
- Deployment Format
- Lebenszyklus
- Definiert seine Bestandteile

# Modul - Bundle

---

- Symbolic Name + Id
- Klassenpfad
- Inhalt/Resourcen

de.herbstcampus



herbst.jar

# MANIFEST.MF

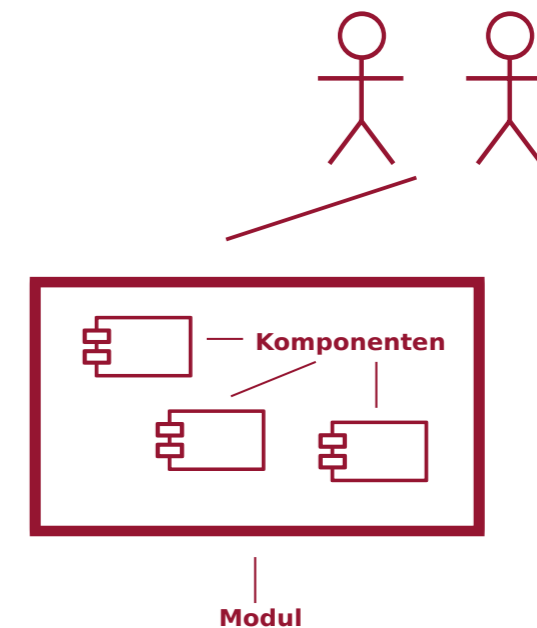
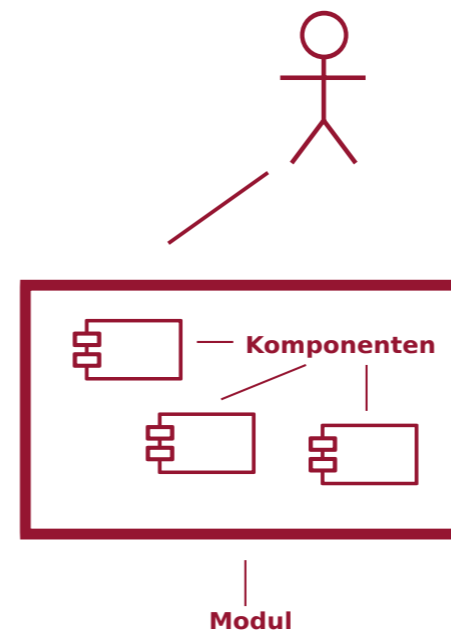
---

- Bundle-SymbolicName
- **Export/Import-Package**
- Require-Bundle
  
- Bundle-Activator
- Bundle-Classpath
- Bundle-ActivationPolicy
  
- X-



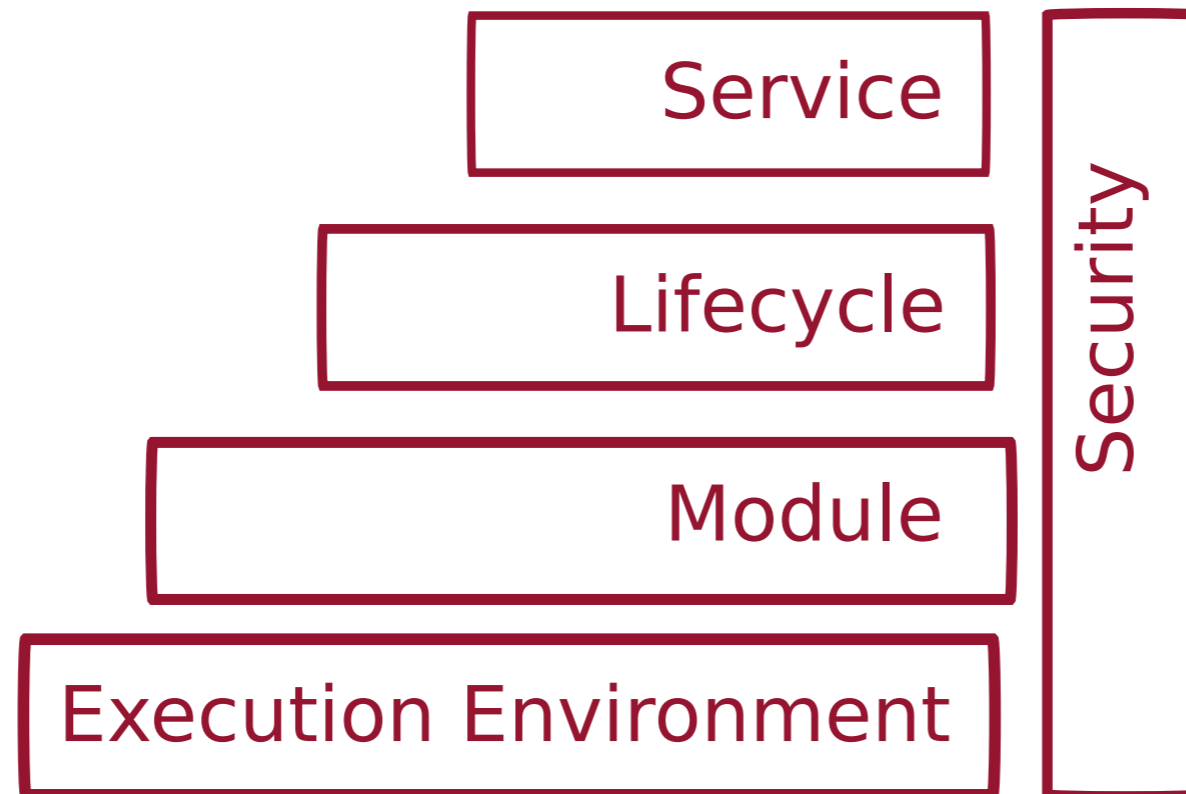
# OSGi - Projektmanagement

- Projektleiter
  - Abgrenzung zwischen Teams und Entwicklern
  - Abhängigkeiten verwalten (Import/Export)
  - Verbessert die Testbarkeit
- Entwickler
  - Lernaufwand
  - Mehraufwand
  - Abgrenzung
- Habe fertig



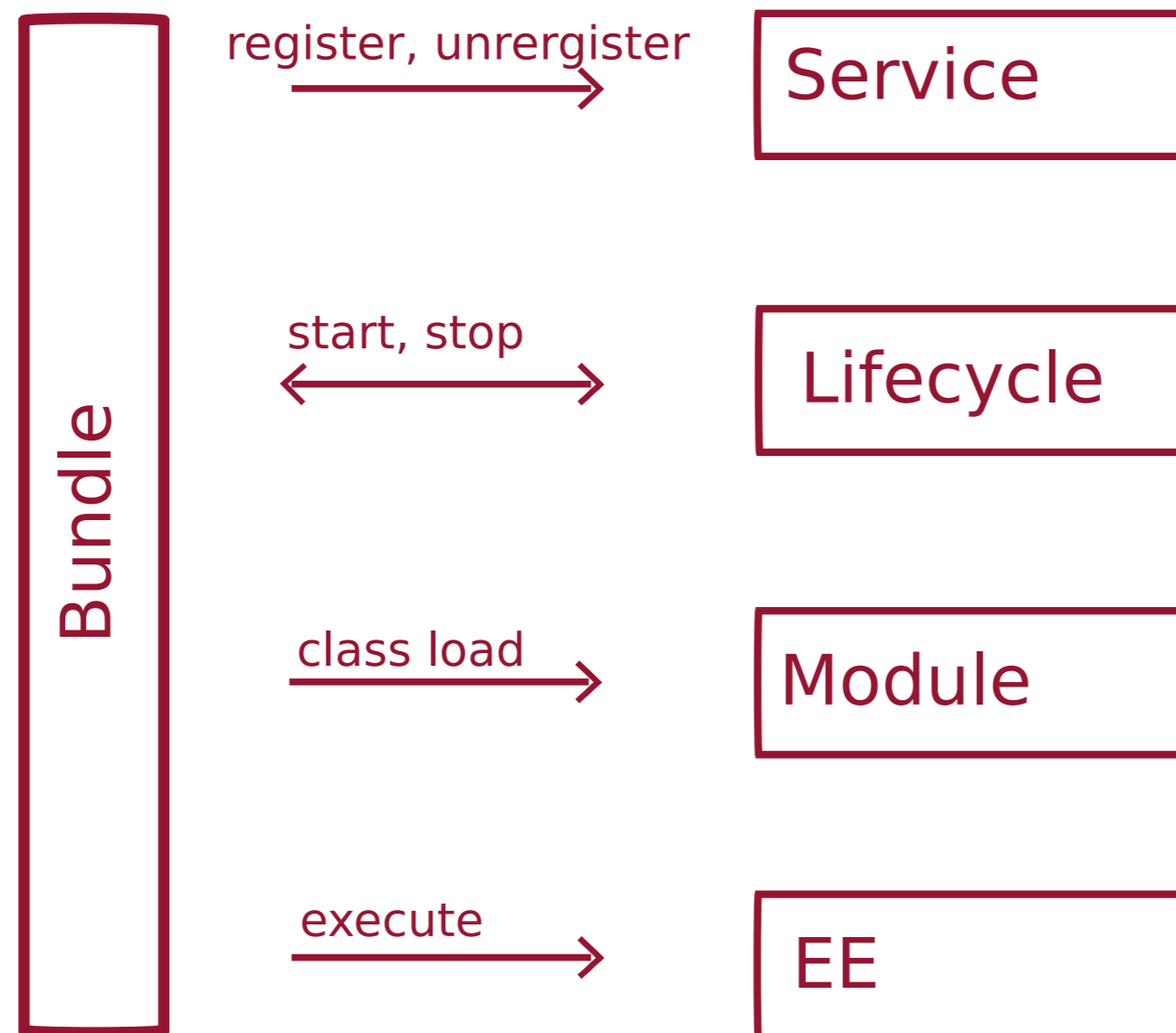
# OSGi - Architektur

---

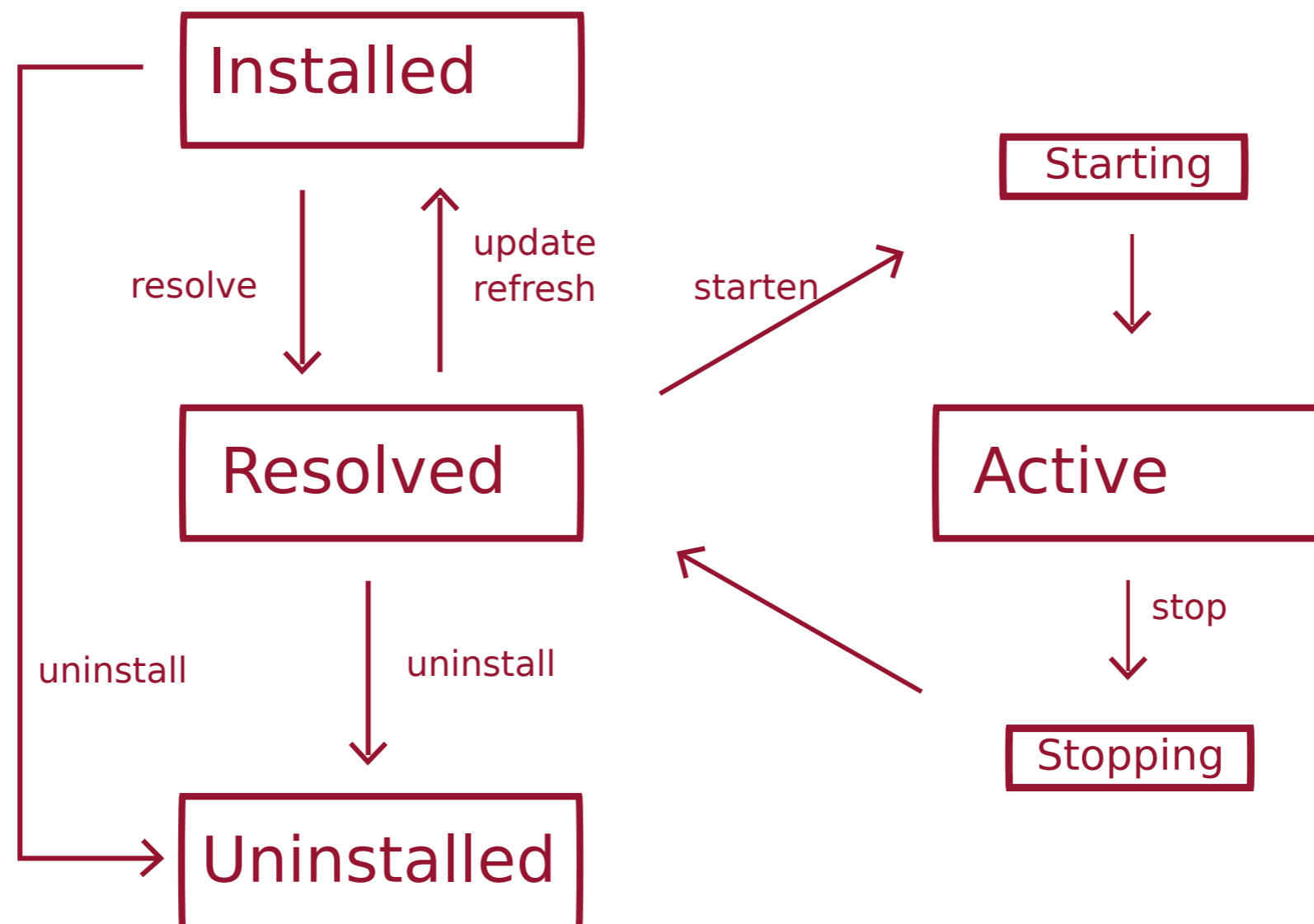


# OSGi - Architektur

---



# Bundle - Lebenszyklus



# Activator - starten und stoppen

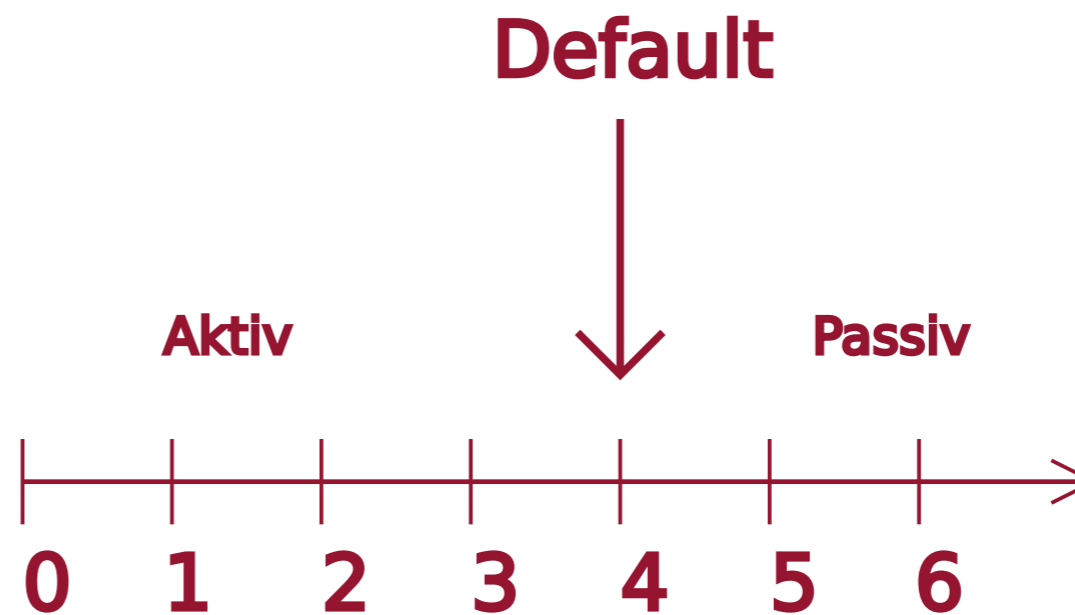
---

- **public void** start(BundleContext bc) throws Exception {}
- **public void** stop(BundleContext bc) throws Exception {}

# Startlevel

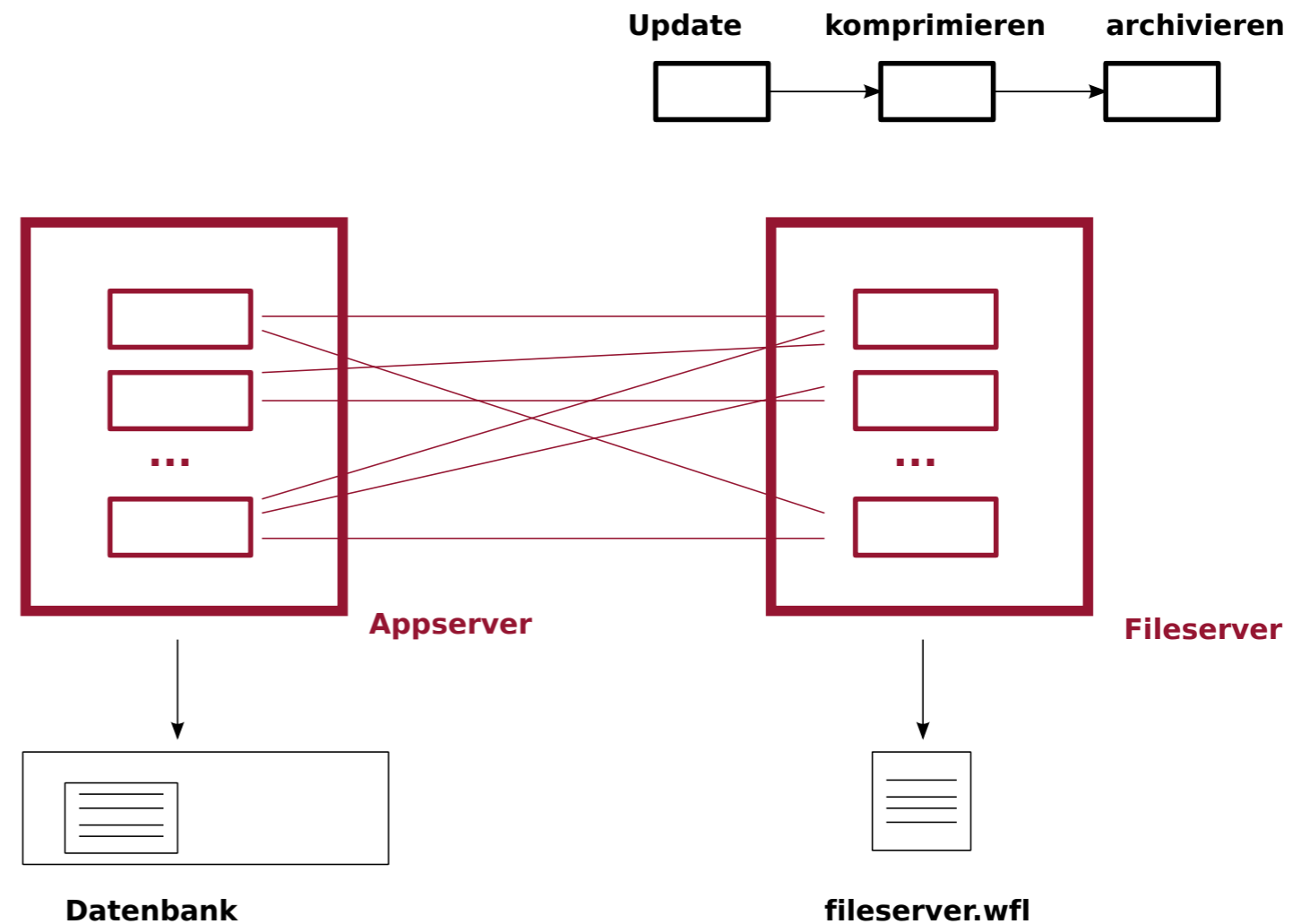
---

- Nicht manuell für jedes Bundle verwalten
- Systemberuhigung
- Subsysteme



# Starten - Design

- OSGi löst kein Designproblem
- Startreihenfolge muss geplant werden



# Versionierung

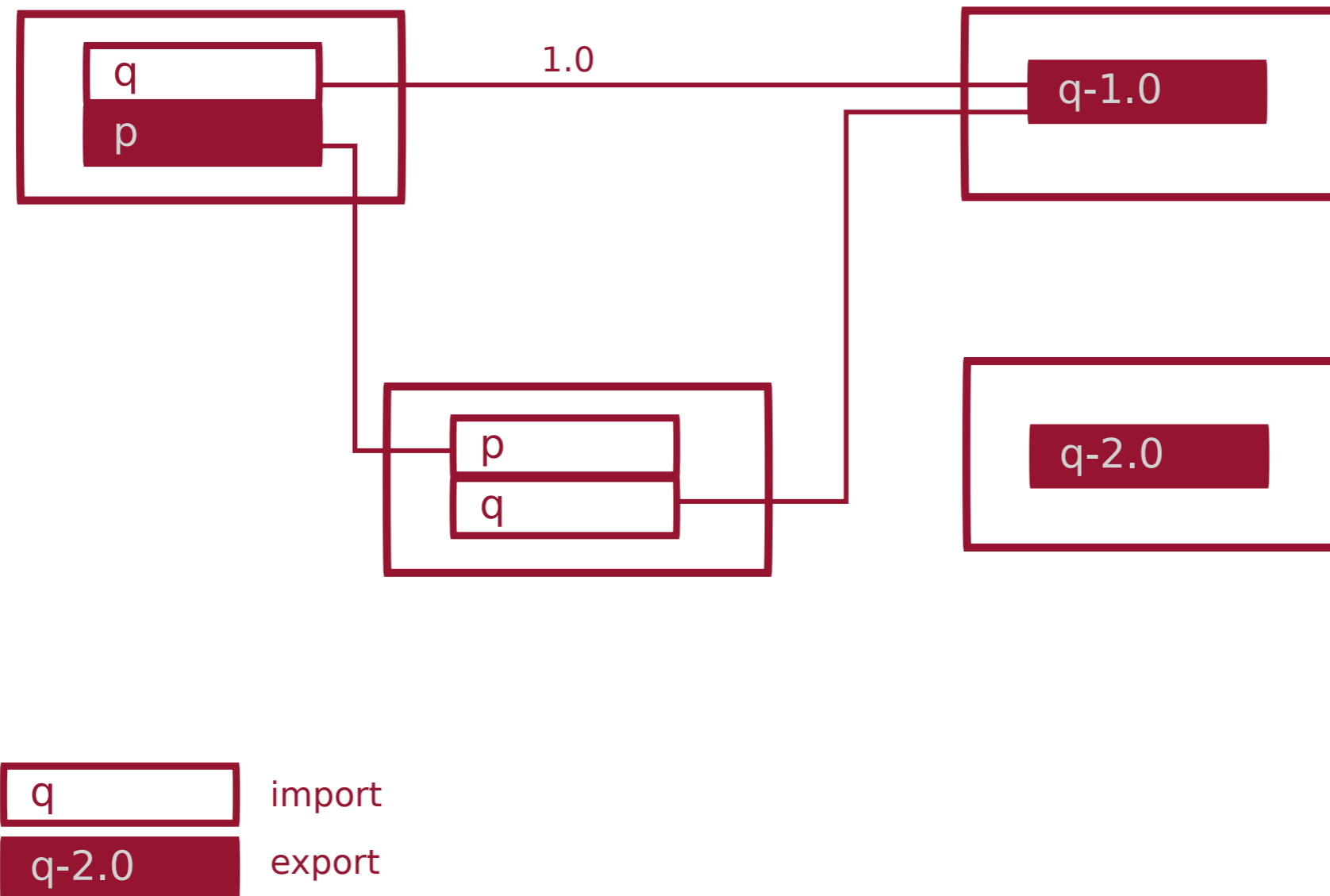
---

- Das gleiche Bundle mit unterschiedlichen Versionen
  - de.herbstcampus-1.1.0
  - de.herbstcampus-2.0
- Wiring
- Updates (Neues Resolving bzw. wiring)



# Wiring

- Exportierte Paket auch importieren



# BundleContext

---

- Properties abfragen
- Service-Behandlung
- Listeners
- Zugriff auf andere Bundles
- Dateien zum Lesen und zum Speichern (`getDataFile()`)

# Fragment

---

- **Eigenschaften**
  - Kein Bundle
  - Kein Lebenszyklus
  - Hat immer Host-Bundle (1..\*)
  - Kann exports hinzufügen
  - Wird an den Klassenpfad angefügt
  
- **Verwendung**
  - Internationalisierung
  - Testen

12.–15.09.2010  
in Nürnberg



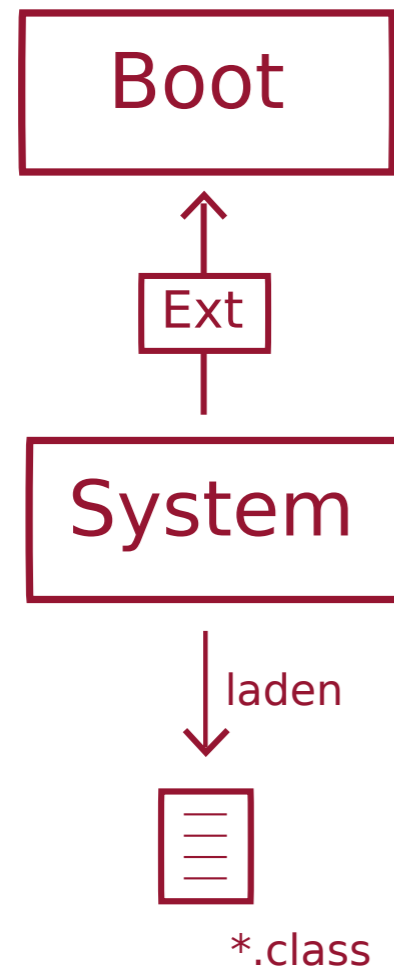
# Herbstcampus

Wissenstransfer  
par excellence

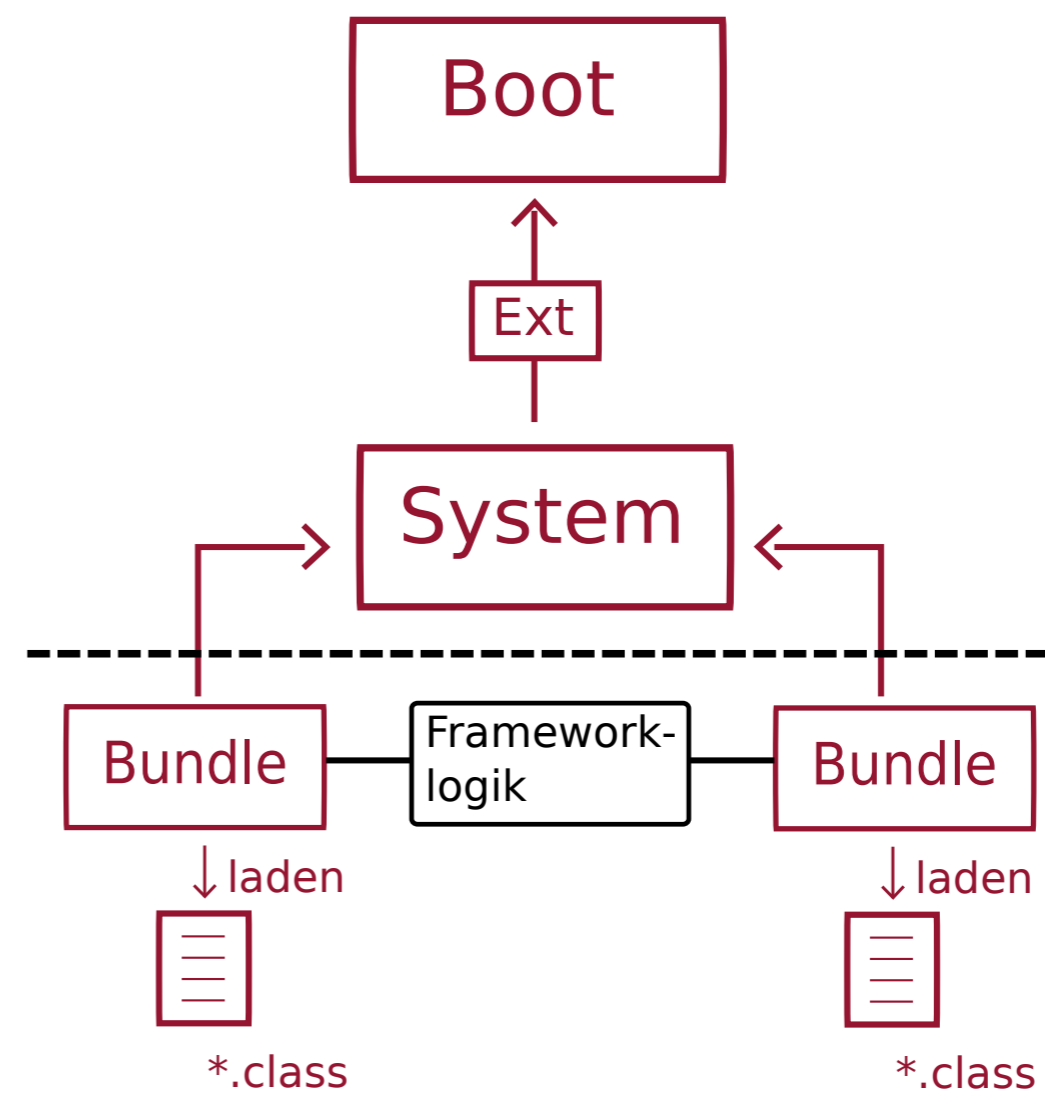
# Klassen laden

# Klassen laden

## Standard



## OSGi



# Klassen laden - OSGi Reihenfolge

---

- \*.java
- bootdelegation
- import - Delegation an Importer
- required Bundles
- Eigener Klassenpfad
- Fragmente
- Dynamic Import

# Klassen laden - Probleme

---

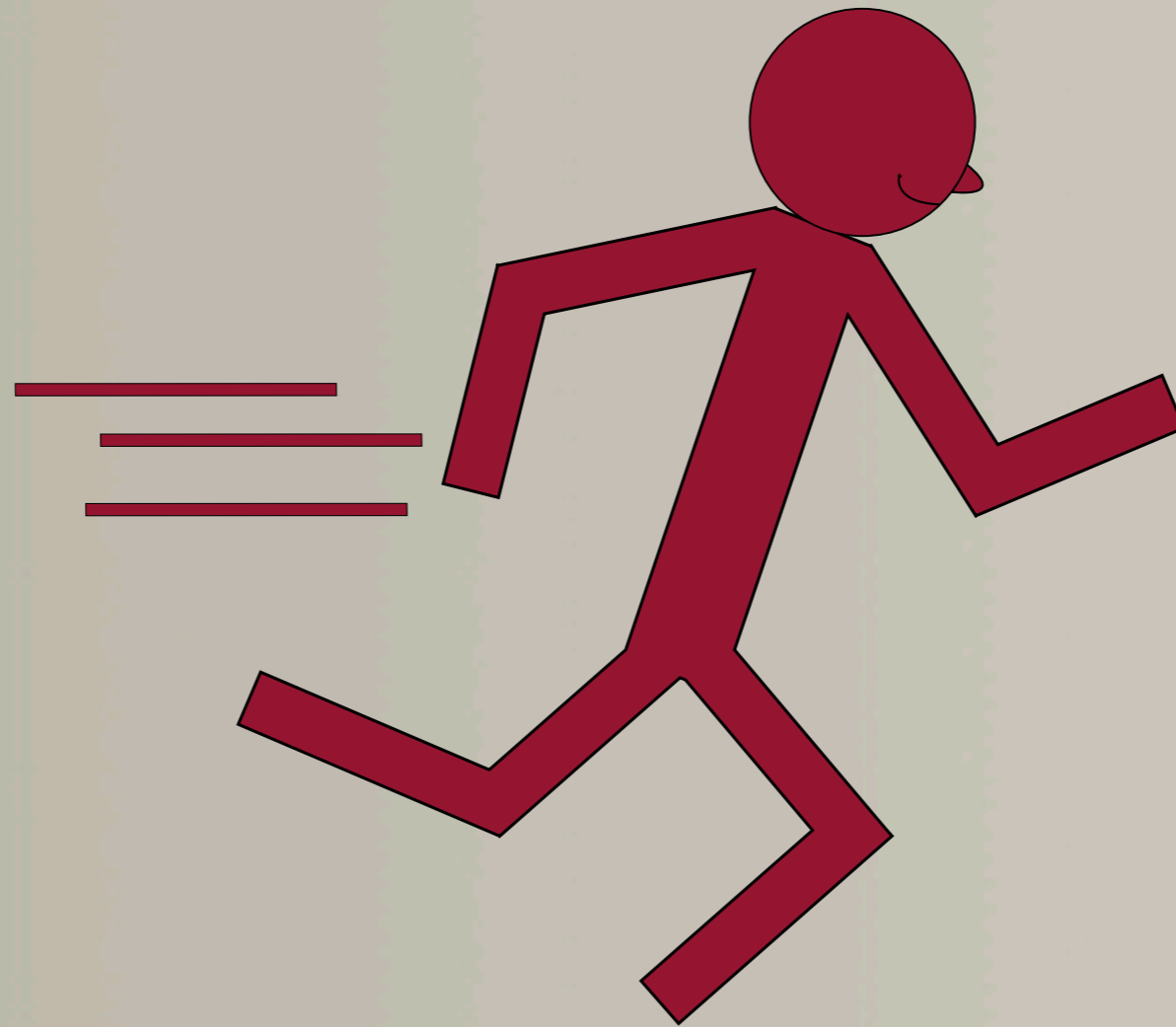
- Eigene Classloader
- `Class.forName()`
- `ContextClassLoader`

12.–15.09.2010  
in Nürnberg



# Herbstcampus

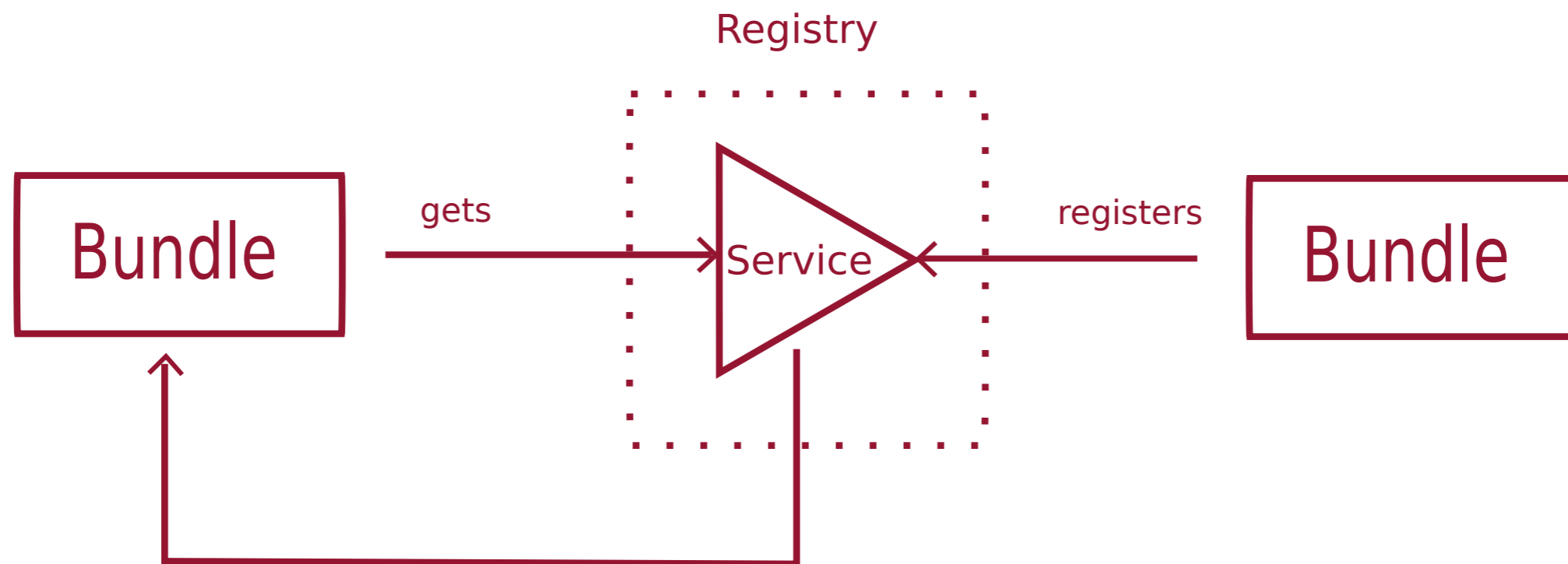
Wissenstransfer  
par excellence





# Service und ServiceRegistration

- Service: Javaklasse + Properties



# Services können kommen und gehen

---



# Services verwalten

---

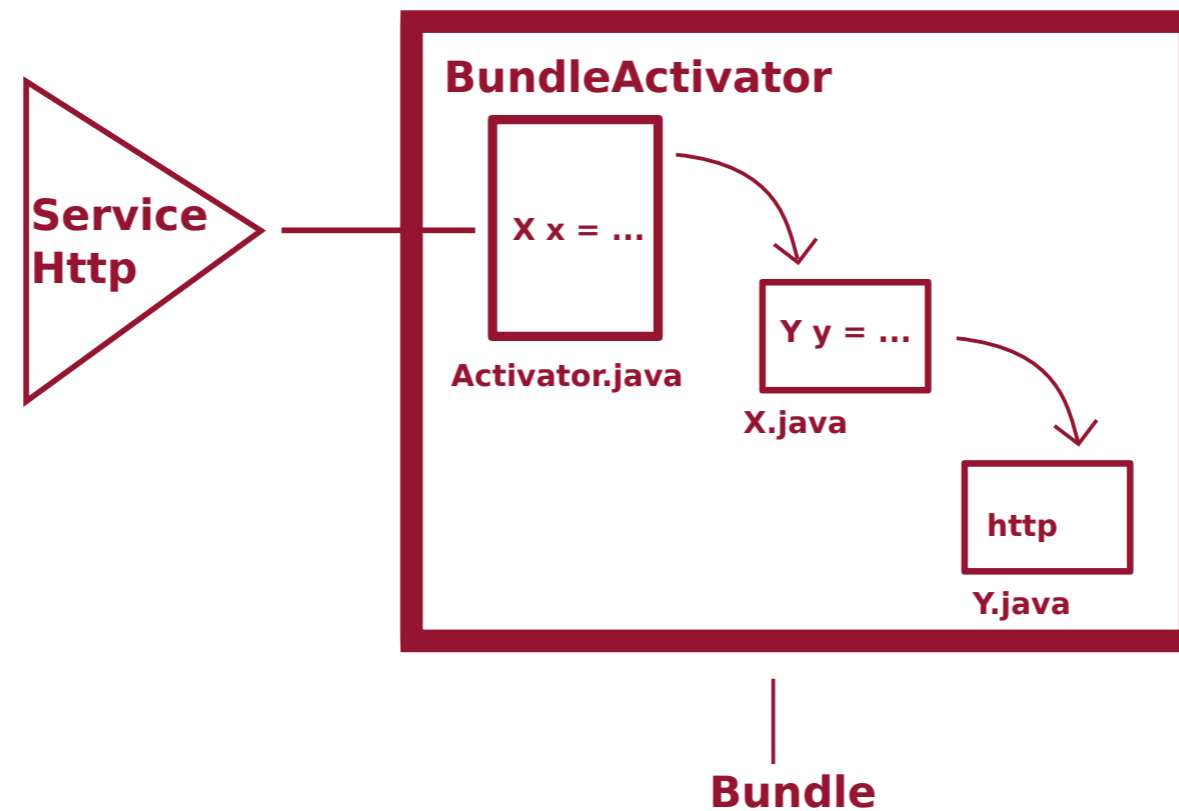
- Service registrieren
  - Normalerweise in der start-Methode des BundleActivators
  - BundleContext nutzen: `bc.registerService(clazz, service, properties);`
- Service erfragen
  - Normalerweise in der start-Methode des BundleActivators
  - BundleContext nutzen: `bc.getService(clazz, service, properties);`
- ServiceTracker
- Declarative Services - DS

# ServiceTracker

---

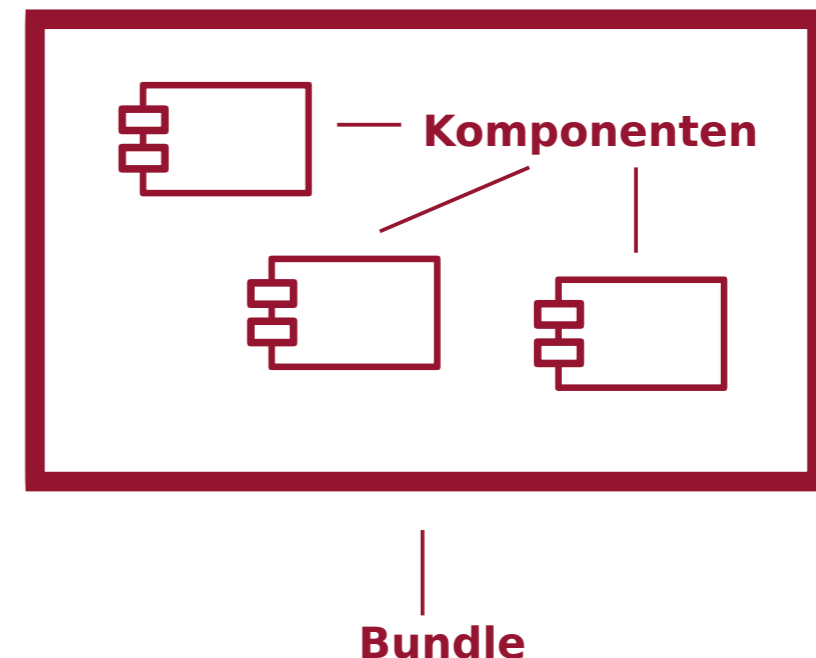
- Activator wird komplex, wenn viele Services kommen und gehen können

# Activator - Referenzen weiterreichen



# DS - Declarative Services

- Die zweite Antwort auf komplexe BundleActivators
- Konfigurationsdatei in Manifest eintragen
  - Manifesteintrag:  
**Service-Component:\*.xml**
- Dependency Injection für Service
  - `setHttp(Http http)`
- Lazy Evaluation



# DS - Konfigurationsdatei \*.xml

---

- Implementation class
- Lebenszyklus
  - activate
  - deactivate
- Service
  - provide
  - reference
- Properties

# DS - activate und deactivate

---

- `void <method-name>(<arguments>)`
- `<arguments>`:
  - `ComponentContext`
  - `BundleContext`
  - `Map`
- `<method-name>`
  - in `*.xml`: `<activate="startup">`
- Bei überladenen Methoden gilt die Reihenfolge
  - `public void startup(ComponentContext cc) {}`
  - `public void startup(BundleContext cc) {}`
  - `public void startup(Map map) {}`
  - `public void startup() {}`



## DS - bind, unbind (Service auflösen)

---

- `void <method-name>(ServiceReference);`
- `void <method-name>(<parameter-type>);`
- `void <method-name>(<parameter-type>, Map);`
  
- `*.xml`
  - `<reference bind="setHttp" interface="herbstcampus.Http">`
  
- Bei überladenen Methoden gilt die Reihenfolge
  - `public void setHttp(ServiceReference sr) {}`
  - `public void setHttp(Http http, Map map) {}`
  - `public void setHttp(Http http, Map map) {}`

12.–15.09.2010  
in Nürnberg



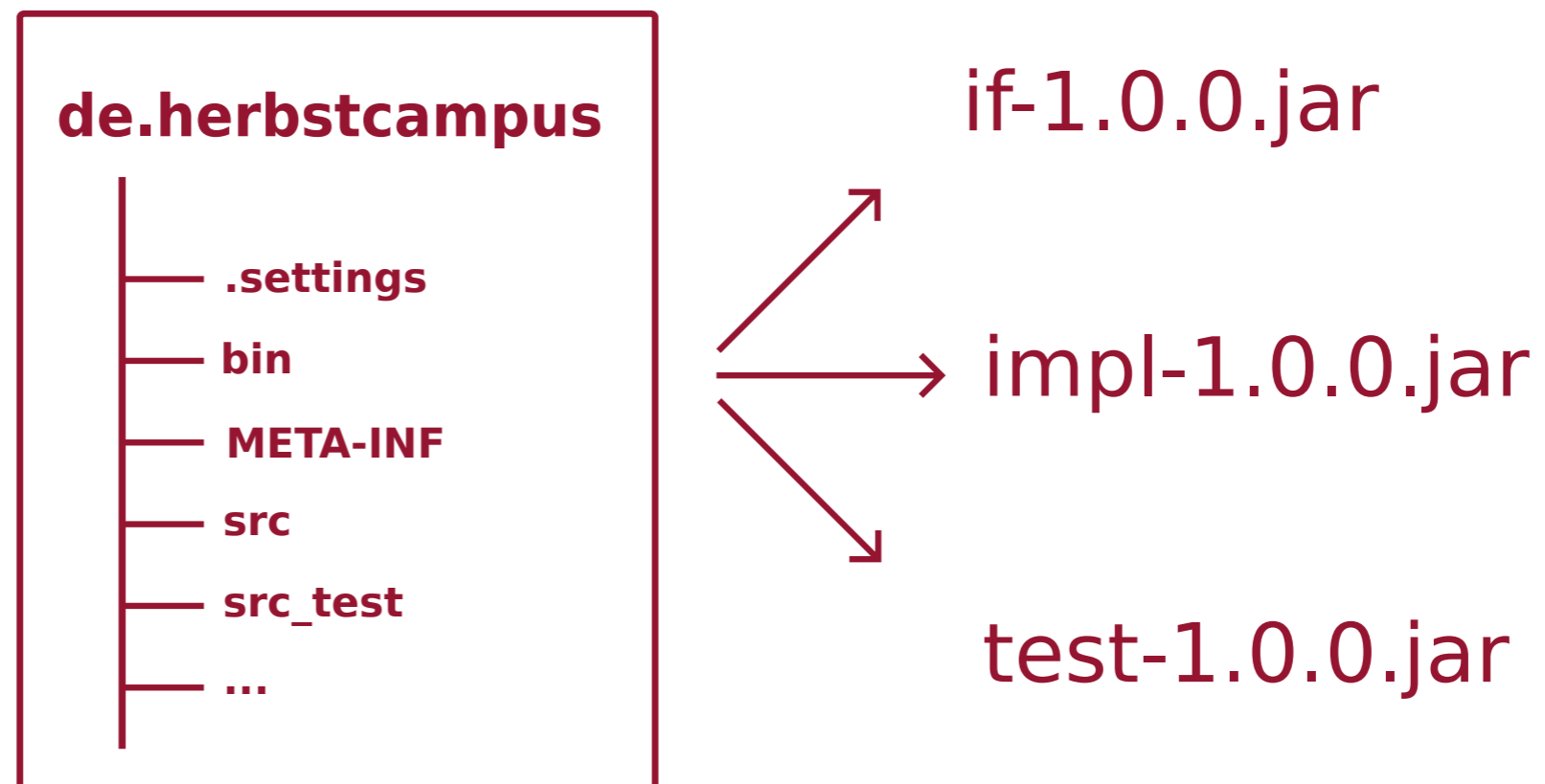
# Herbstcampus

Wissenstransfer  
par excellence

# Buildsystem

# Buildsystem - Knopflerfish

- Ein Projekt produziert drei Bundles



# Buildsystem - Equinox

---

- Plugins bauen
- Mehrere Plattformen werden unterstützt

12.–15.09.2010  
in Nürnberg



# Herbstcampus

Wissenstransfer  
par excellence

# Testen

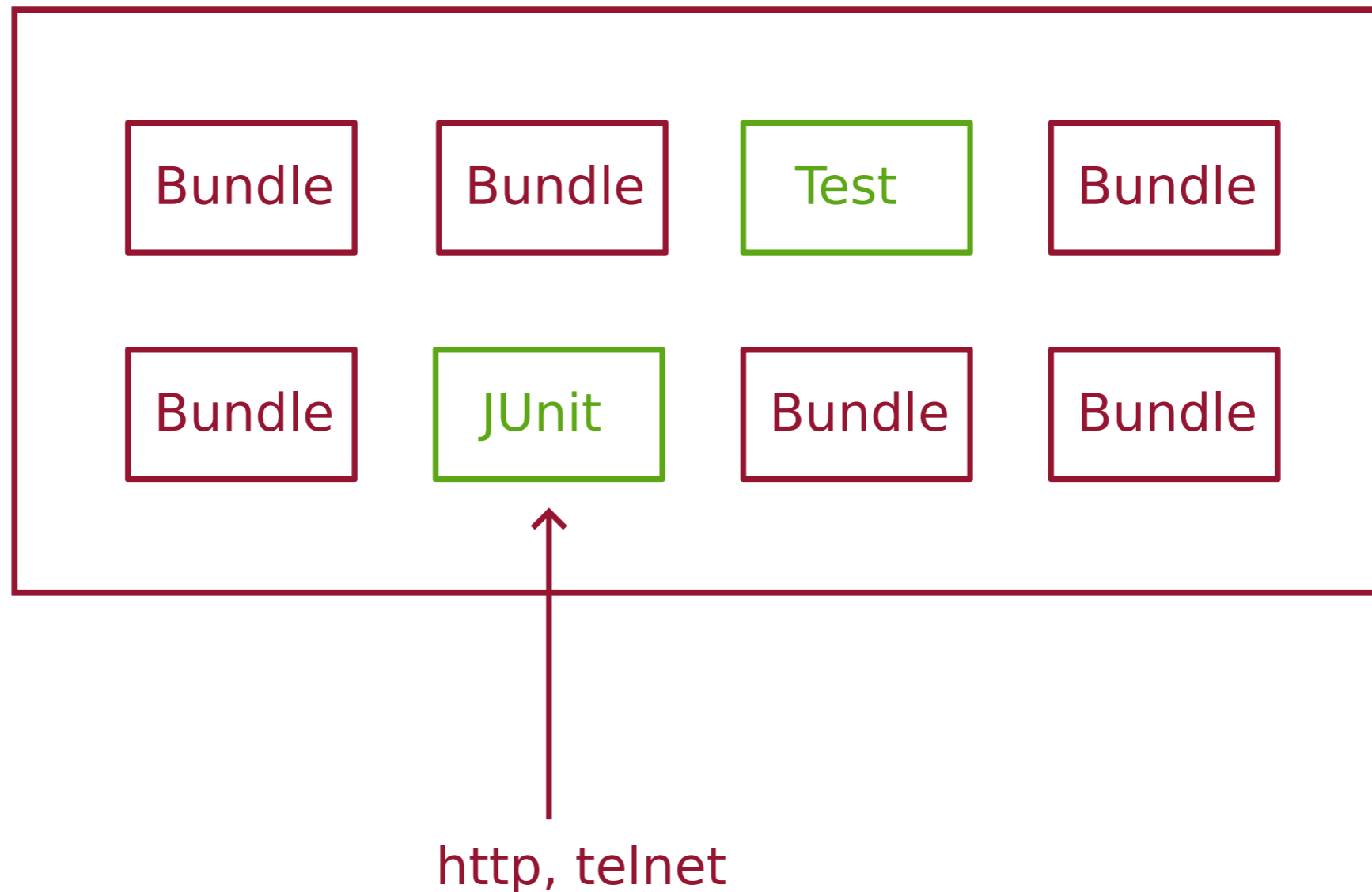
# Testen - Bundle starten?

---

- Klassenebene
  - OSGi-Code nur zum starten verwenden
  - Standardtests mit JUnit
- Integration
  - Test-Treiber in das System installieren (JUnit)
  - Integrationstests und -mocks in das System installieren
- Dynamik
  - DA Framework (<http://www.dynamicjava.org/>)
  - Annotationen
- System
  - Testabteilung

# Testen - Integration - Mock

- Knopflerfish



12.–15.09.2010  
in Nürnberg



# Herbstcampus

Wissenstransfer  
par excellence

# Equinox



# Erweiterungen

---

- PDE - Buildsystem
- P2 Deployment (Update-Site)
- Buddy - Classloading
- Extension-Registry
  - plugin.xml
  - extension point
  - extension

12.–15.09.2010  
in Nürnberg



# Herbstcampus

Wissenstransfer  
par excellence

- Integraler Bestandteil einer Programmiersprache
- Grüne Wiese
- Externe Bibliotheken

12.–15.09.2010  
in Nürnberg



# Herbstcampus

Wissenstransfer  
par excellence

Vielen Dank!

Erwin Hösch

MATHEMA Software GmbH

# Firma

---

- MATHEMA Software GmbH  
91052 Erlangen, Henkestraße 91