Gerd Wütherich | freiberuflicher Software-Architekt
Bernd Kolb | KolbWare
Martin Lippert | akquinet agile GmbH

Spring Dynamic Modules for OSGi Service Platforms

Agenda

OSGi Technologie:

- OSGi™ Technologie im Überblick
- Ein praktisches Beispiel

Spring Dynamic Modules:

- Spring Dynamic Modules im Überblick
- Ein praktisches Beispiel

Ausblick auf Release 1.1 – Web-Support:

- Web-Support im Überblick
- Ein praktisches Beispiel

OSG – was?

Die OSGi™ Service Platform:

- Java-basierte Software-Plattform, die dynamische Integration und (Fern-) Management von Modulen (Bundles) und Diensten (Services) ermöglicht.
- „A dynamic module system for Java“

Besteht aus:

- OSGi™ Framework (Container für Bundles und Services)
- OSGi™ Standard Services (verschiedene, horizontale Dienste)

Woher kommt OSGi™ Service Platform?

OSGi-Alliance:

- http://www.osgi.org/

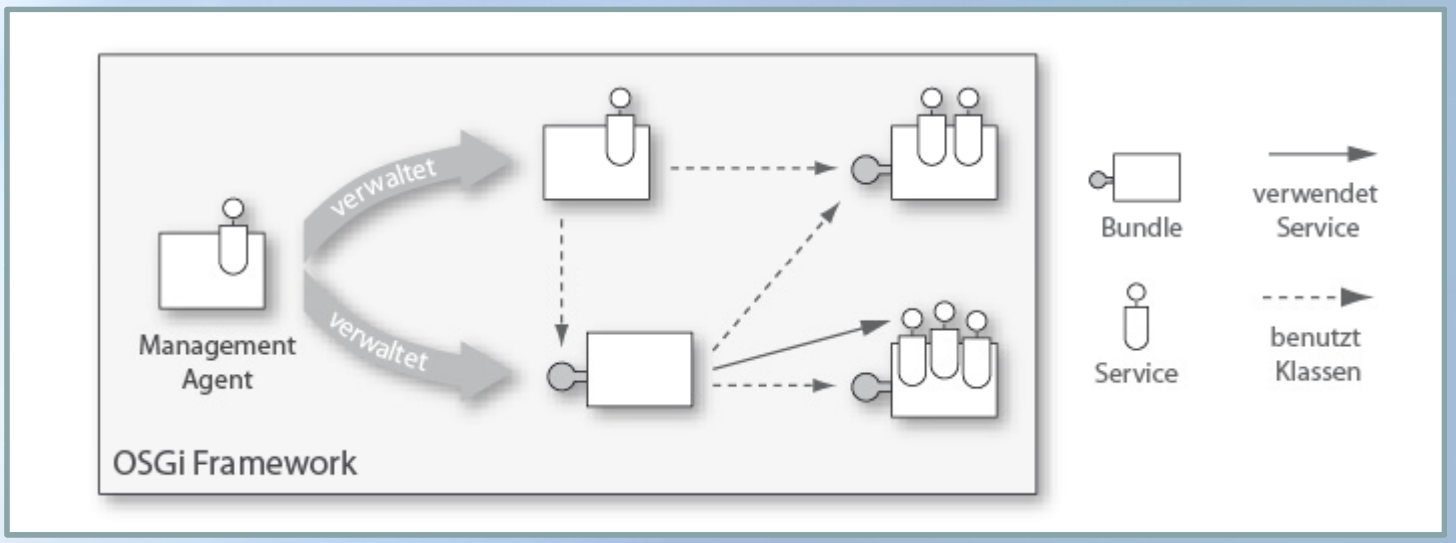
Wird seit 1999 mit Fokus auf Leichtgewichtigkeit und Dynamik entwickelt:

- Ursprünglich für Embedded-Systeme
- Inzwischen verbreitet für Server- und Client-Systeme

Wo wird OSGi Service Platform heute verwendet?

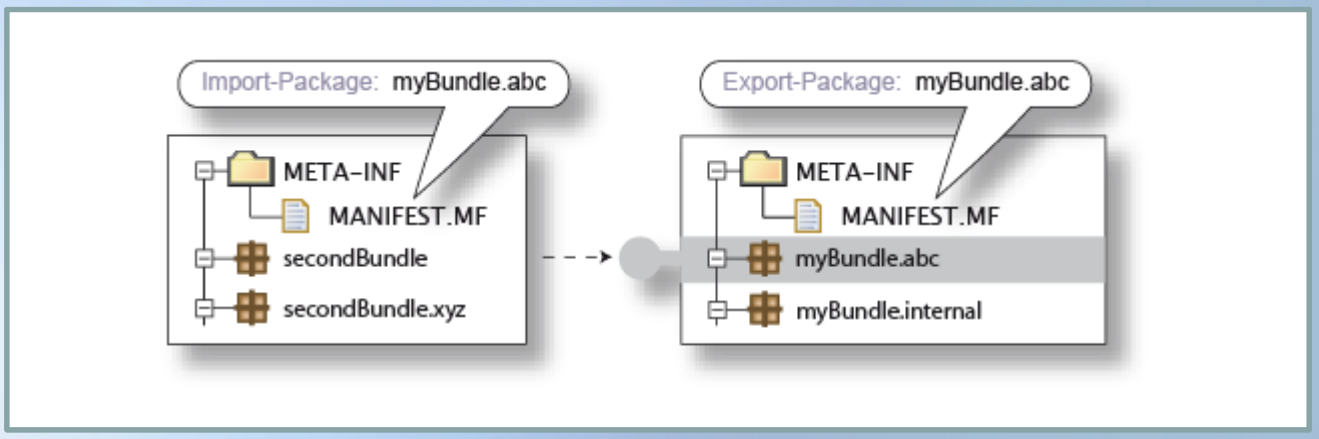
- **Die Eclipse-Plattform**
  - Eclipse-SDK (IDEs), Server-Side-Eclipse, eRCP, …
- **IBM**
  - Websphere App Server 6.1 (basiert auf OSGi)
  - Lotus (basiert auf Eclipse-RCP, damit auch OSGi)
  - Jazz (basiert auf Server-Side-Eclipse)
- **BEA/Oracle, Adobe**
- **JBoss prototypt OSGi-Umstellung**
- **...**

Das OSGi Framework: Überblick



- Kern der OSGi Service Platform
- Ermöglicht die Installation und Administration von Bundles und Services
- Verwaltet (Klassen-)Abhängigkeiten zwischen Bundles
- Kann „von außen“ administriert werden

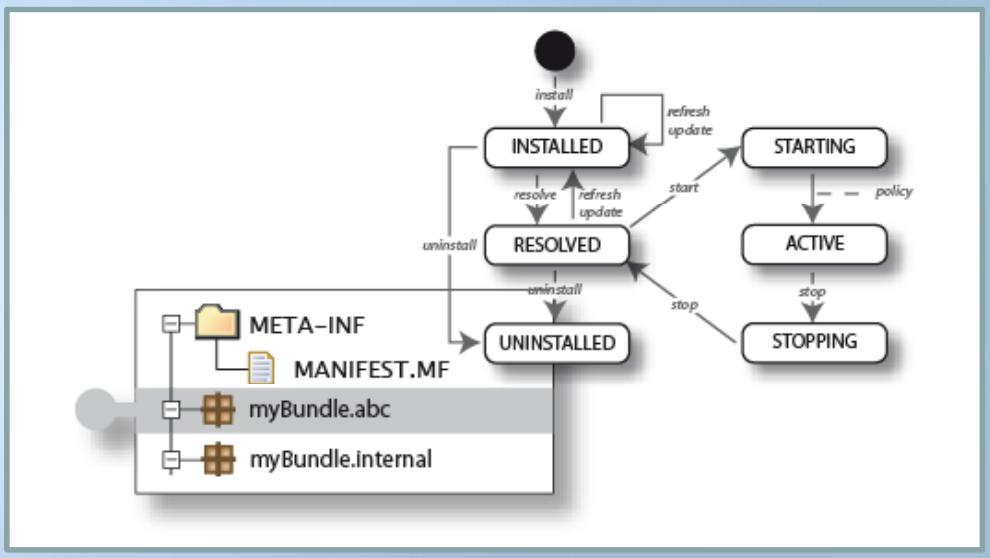
Das OSGi Framework: Bundles



Das Framework erlaubt die Definition von

- Modulen (genannt „Bundles“),
- Sichtbarkeiten von Modul-Bestandteilen (public-API vs. private-API)
- Abhängigkeiten zwischen Modulen, sowie
- Versionen von Modulen.

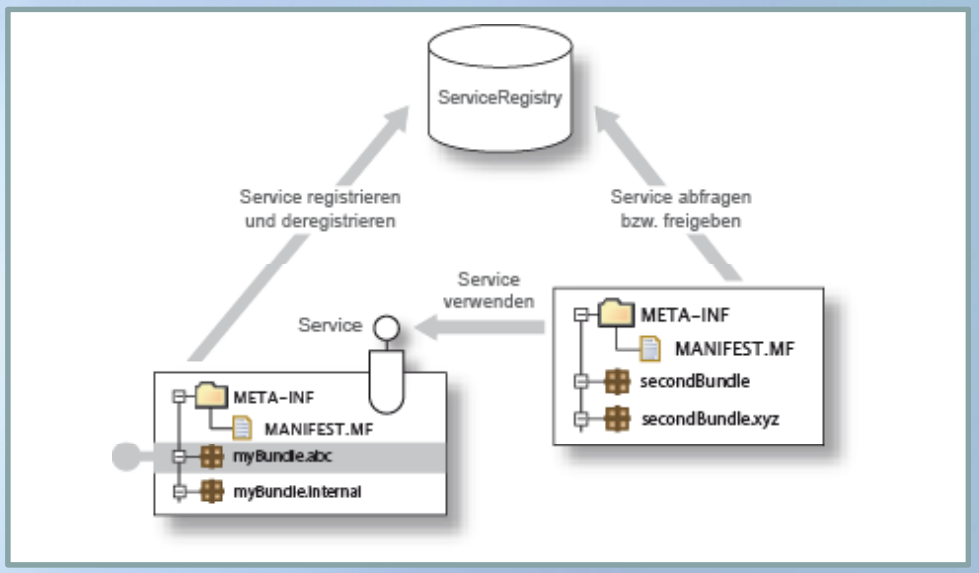
Das OSGi Framework: Lebenszyklus von Bundles



Das OSGi™ Framework ist dynamisch:

- Bundles können dynamisch zur Laufzeit installiert, gestartet, gestoppt, deinstalliert und aktualisiert werden.
- Bei Start und Stopp können Bundles über den Bundle-Activator Code ausführen.

Das OSGi Framework: Services



Das OSGi Framework ist service-orientiert:

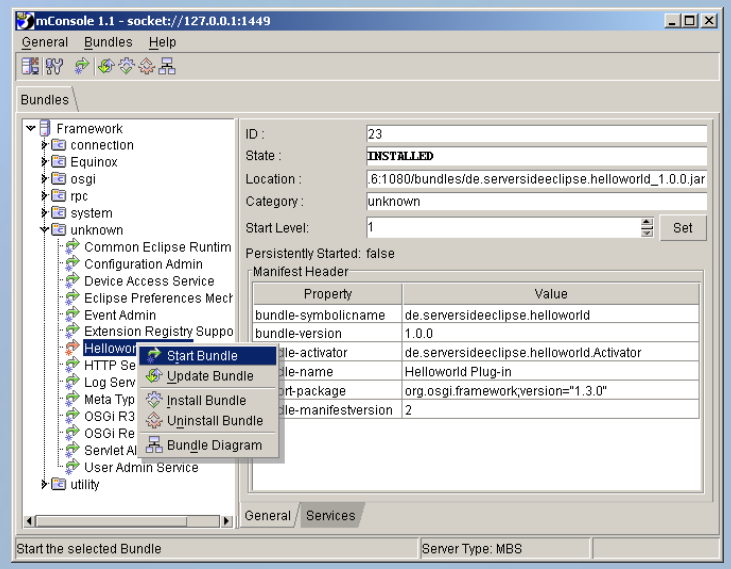
- Services können zur Laufzeit über die Service-Registry veröffentlichen und wieder entfernen
- Bundles können über die Service-Registry Services finden und verwenden

Das OSGi Framework: Management Agents I

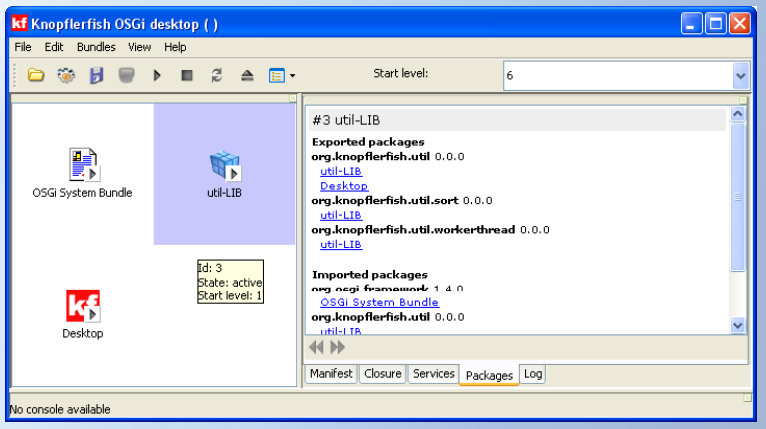
- Management Agents ermöglichen die Administration eines OSGi Frameworks.
- Breites Spektrum verfügbar:
  - Kommandobasierte Konsole (Equinox Konsole)
  - Grafisch-interaktive Anwendungen (Knopflerfish Desktop, Prosyst mConsole)
  - Web-basierte Oberfläche (Knopflerfish Web-Konsole)
- Nicht standardisiert, aber Zugriff auf das Framework über definierte Schnittstellen.

Das OSGi Framework: Management Agents II

Eclipse Equinox Konsole



Prosyst mConsole



Knopferfish Desktop

Implementierungen der OSGi Service Platform

Open-Source-Implementierungen:

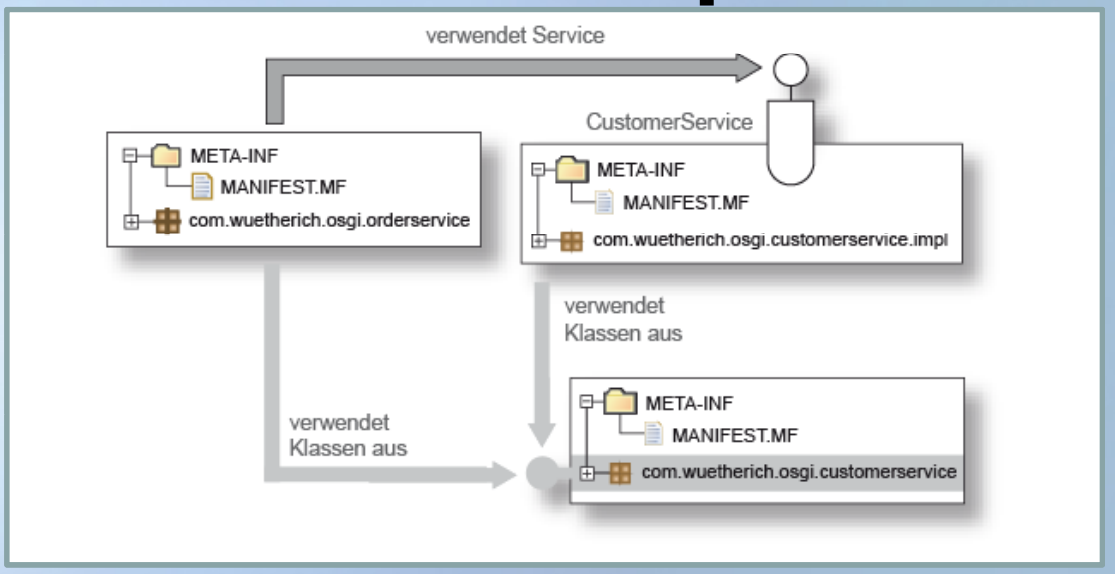
- Eclipse Equinox (http://www.eclipse.org/equinox/)
- Apache Felix (http://felix.apache.org/)
- Knopflerfish (http://www.knopflerfish.org/)
- ProSyst mBedded Server Equinox Edition (http://www.prosyst.com/products/osgi_se_equi_ed.html)

Kommerzielle Implementierungen:

- ProSyst (http://www.prosyst.com/)
- Knopflerfish Pro (http://www.gatespacetelematics.com/)

(kein Anspruch auf Vollständigkeit)

Die OSGi Service Platform : Ein praktisches Beispiel



Der OrderService…

- …ermöglicht die Order von Aktien für einen Kunden
- …ist Remote über RMI verfügbar

Der CustomerService…

- …liefert dem OrderService die passenden Kundendaten

Zwischenbilanz

- OSGi Service Platform ermöglicht Modularisierung von Anwendungen.
- Module können zur Laufzeit installiert, gestartet, gestoppt, deinstalliert werden.

Aber:

- Abhängigkeit zu OSGi Framework-Bibliotheken
- Services müssen programmtisch verwaltet werden
- Keine Unterstützung für „Enterprise-Technologien“

Spring Dynamic Modules for OSGi Service Platforms

- Formerly known as „Spring-OSGi“
- Ein neues Mitglied der Spring-Familie
  - http://www.springframework.org/osgi
  - Keine eigene OSGi-Implementierung, sondern eine Brücke zwischen Spring und OSGi-Implementierung
  - Erlaubt es, Spring-Anwendungen mit OSGi zu implementieren
  - Ist kompatibel zu Equinox, Felix und Knopflerfish

Die Ausgangsbasis

- Das Spring-Framework
  - Einfaches Programmiermodell (POJOs)
  - Dependency Injection
  - AOP
  - Viele Technologie-Vereinfachungen und -Abstraktionen
- Inzwischen ein de-facto-Standard für JEE-Systeme

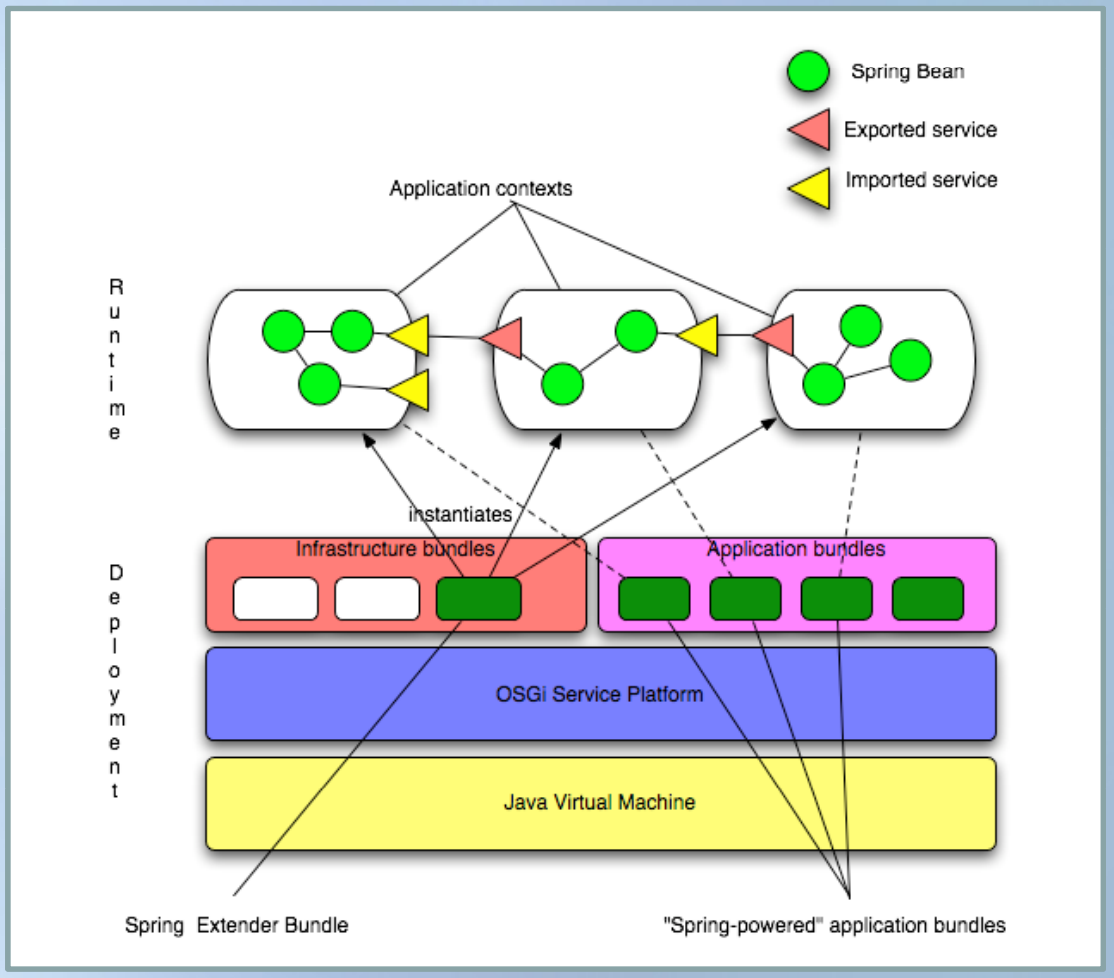
Das Ziel

- Die Möglichkeiten von OSGi nutzen!
  - Modularisierung inkl.
    - Dependency Management
    - Sichtbarkeits-Definitionen
    - Versions-Management
    - Dynamic Management
- Spring selbst steht in Form von Bundles zur Verfügung

Randbedingungen

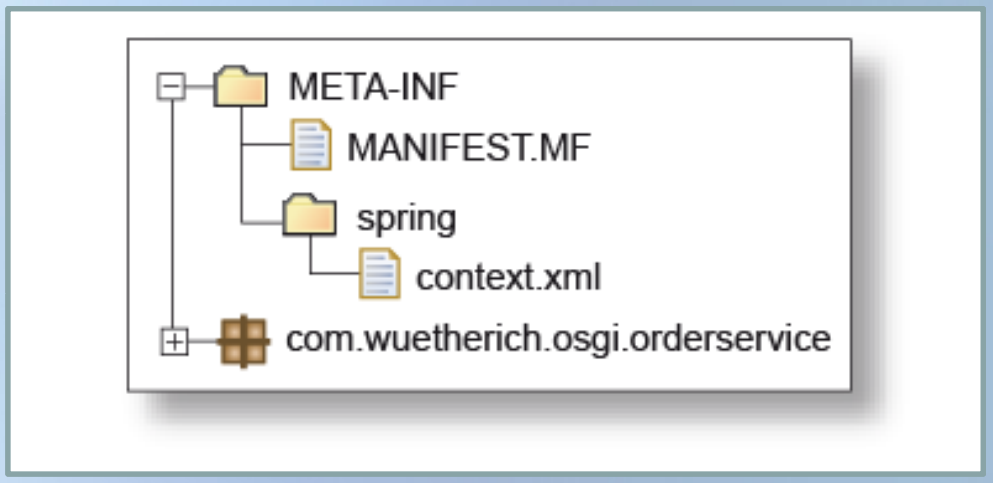
- Komplexität
  - Darf nicht erhöht werden, weiterhin POJO-Programmiermodell
- OSGi-Service-Modell integrieren
  - OSGi Services als Beans und umgekehrt
- Testing
  - muss auch außerhalb eines OSGi Containers möglich sein
  - Möglichst keine Abhängigkeiten zu OSGi APIs
- Alle Enterprise-Features von Spring müssen weiterhin nutzbar sein

Das Spring-OSGi-Modell

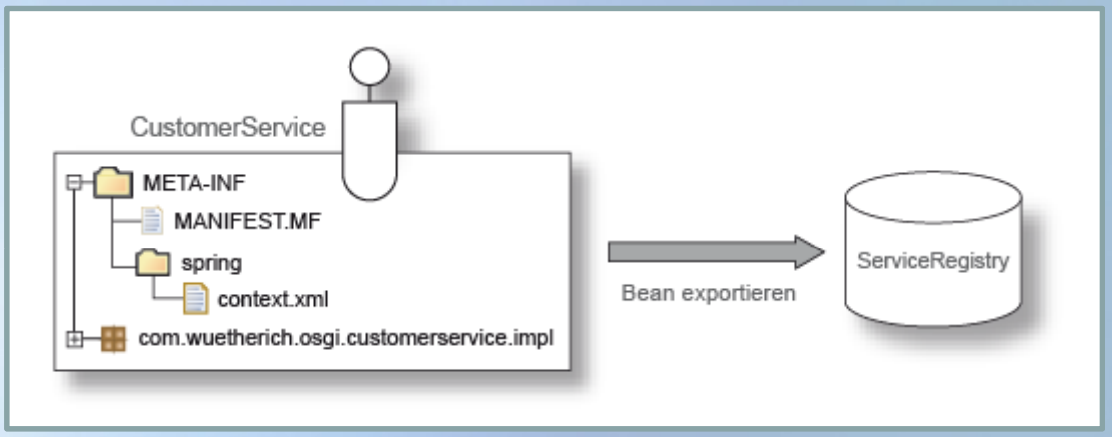


Bundles und Spring

- Ein Application Context pro Bundle:
  - Spring DM erzeugt und zerstört den Context automatisch, wenn das Bundle aktiviert bzw. deaktiviert wird
  - Definition über XML-Dateien unterhalb von ‚META-INF/spring‘



Beans als OSGi Services

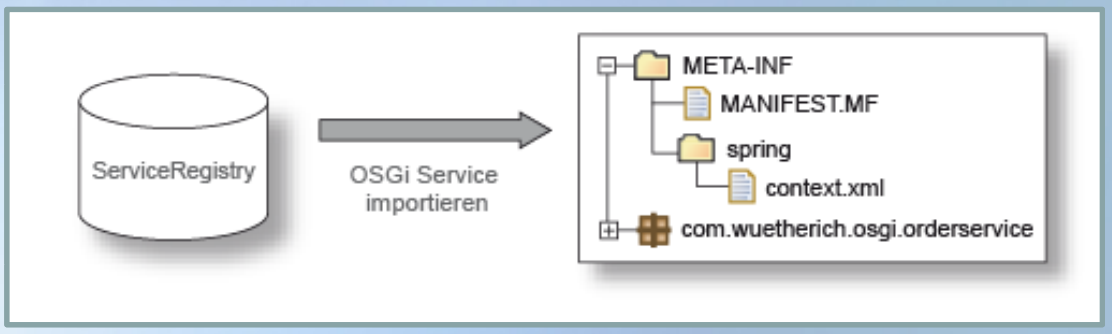


- Spring Beans können als OSGi Services exportiert werden
- OSGi Service sind bundle-übergreifend sichtbar

```
<beans>
  <bean name="customerService"
    class="com.wuetherich.osgi.customerservice.impl.CustomerServiceImpl"/>

  <osgi:service id="customerServiceOsgi"
        ref="customerService"
        interface="com.wuetherich.osgi.customerservice.CustomerService"/>
</beans>
```

OSGi Services als Beans



- Existierende OSGi Services können als Beans in den Spring Context integriert werden

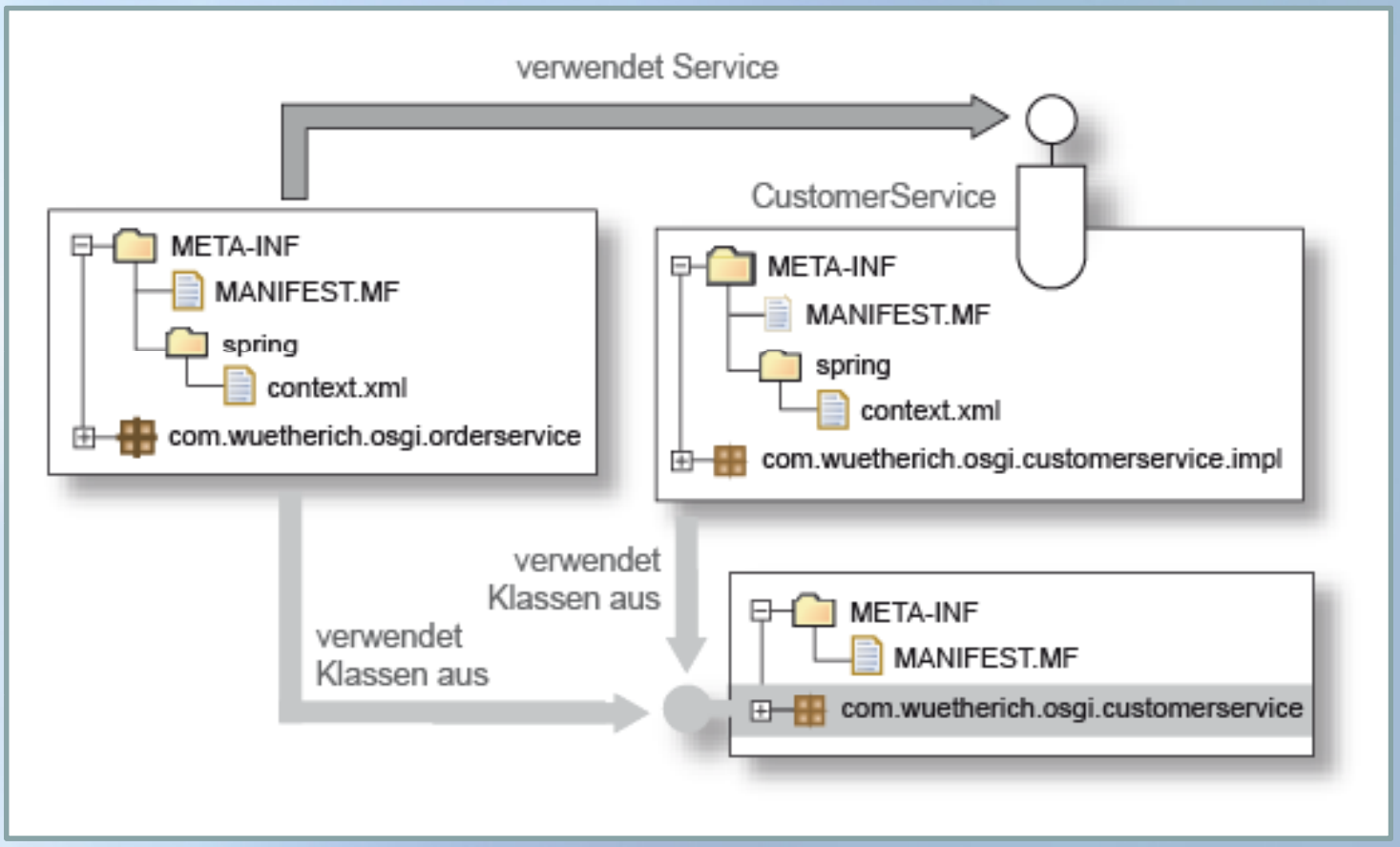
```
<beans>
  <osgi:reference id="customerServiceOsgi"
        interface="com.wuetherich.osgi.customerservice.CustomerService"/>

  <bean id="orderService"
     class="com.wuetherich.osgi.orderservice.internal.OrderServiceImpl">
    <property name="customerService">
      <ref local="customerServiceOsgi"/>
    </property>
  </bean>
</beans>
```

Weitere Möglichkeiten

- Cardinality
  - Beziehungen zwischen importierten OSGi-Services und der repräsentierenden Bean (1..1, 0..1, 1..n, 0..n)
- Service Listener
  - Information einer Bean über Service-Änderungen
- <osgi:property-placeholder>
  - Liest Properties aus einer Properties-Datei
- <osgi:bundle>
  - Stellt ein Bundle-Objekt als Bean bereit
- <osgi:virtual-bundle>
  - erlaubt die Installation eines JARs als Bundle "on-the-fly"

Spring DM in Action: Ein praktisches Beispiel



Resultat

- Keine OSGi API nötig
  - (ganz im Spring-Sinne)
- Bean-Sichtbarkeiten zwischen Bundles können definiert werden
  - nur was für andere Bundles sichtbar sein soll, wird als OSGi-Service exportiert
- Dependency Injection über Bundle-Grenzen hinweg

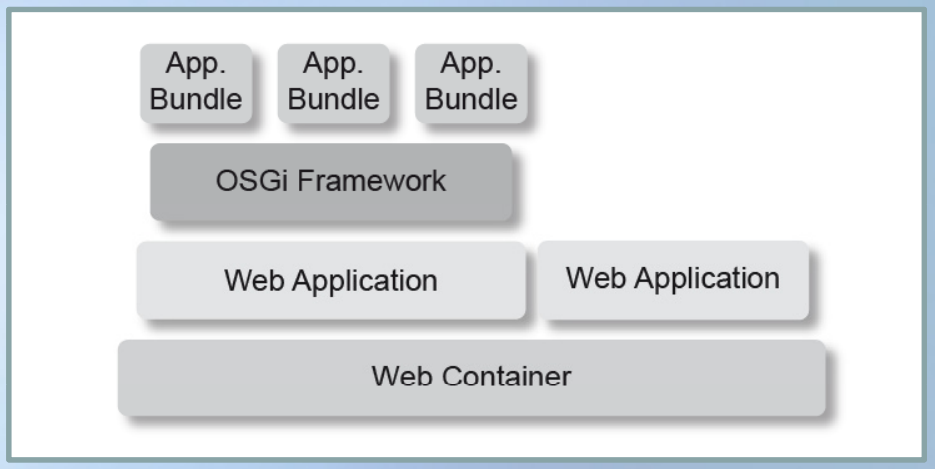
Spring-DM-Web-Unterstützung

- Fokus in Spring DM 1.1
- Derzeit Version 1.1.0.M1 verfügbar
- Integration von Spring DM mit der vorhandenen Spring-Web-Unterstützung

Web-Anwendungen in OSGi

- Generell unterschiedliche Deployment-Szenarien möglich

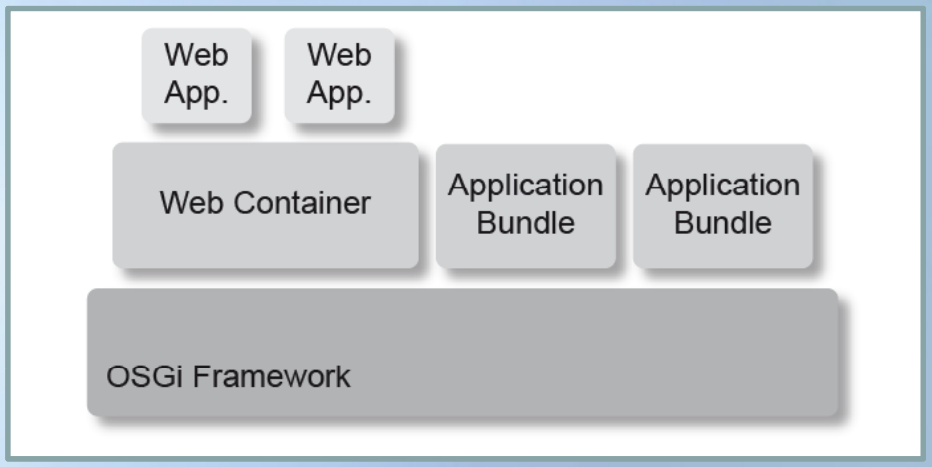
OSGi-basierte Web-Anwendungen 1



Szenario 1: OSGi Framework innerhalb einer Web-Anwendung

- OSGi Framework wird innerhalb einer Web-Anwendung ausgeführt
- Anwendungs-Bundles können über das OSGi Framework in der Web-Anwendung installiert und gestartet werden

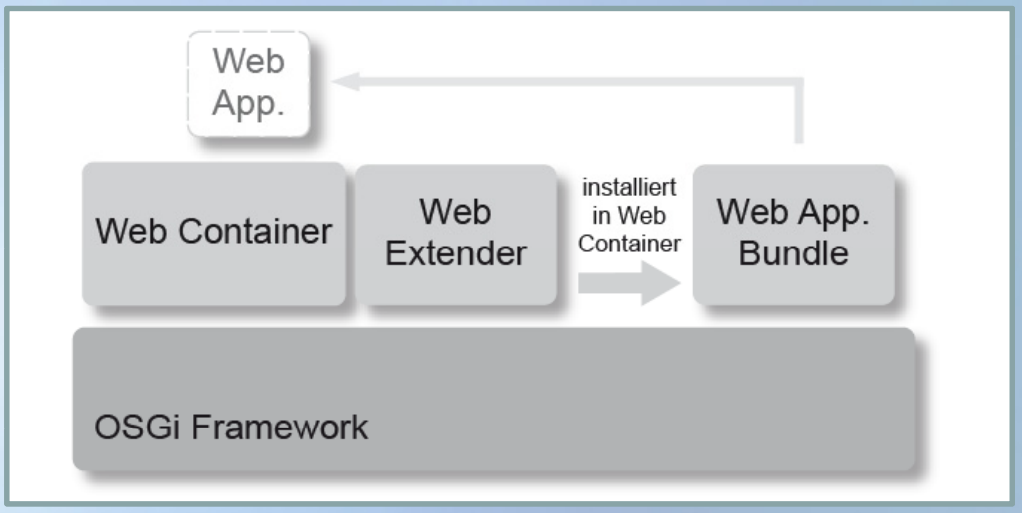
OSGi-basierte Web-Anwendungen 2



Szenario 2: Web Container eingebettet in OSGi Framework

- Web Container wird innerhalb des OSGi Frameworks als Bundle installiert und gestartet
- Anwendungs-Bundles können Web-Anwendungen programmatisch deployen

Web-Unterstützung in Spring DM



- Web-Anwendungen werden als Bundles in OSGi Framework installiert und gestartet („ bundle-fiziertes“ WAR)
- Spring DM Web Extender installiert die Web-Anwendung im Web Container, wenn das WAR-Bundle gestartet wird
- „Native“ Unterstützung verschiedener Web Container
- Volle Unterstützung des OSGi Class Loadings

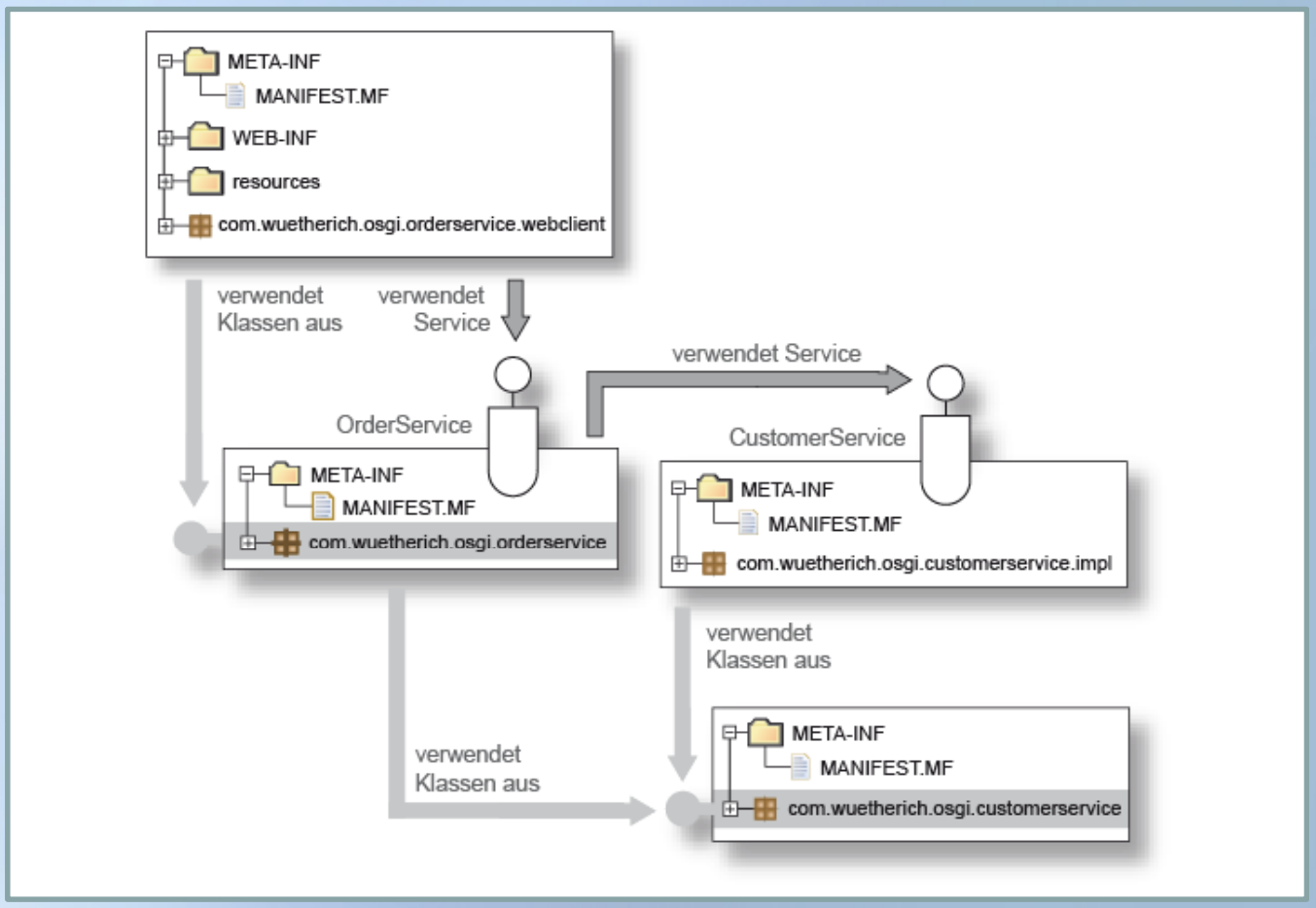
Web-Anwendungen als Bundles

- „Reguläre“ WAR-Dateien
- Zusätzlich: Bundle-Manifest
- Spring-Anbindung über `OsgiBundleXmlWebApplication-Context` und `ContextLoaderListener`
- Spezifikation erfolgt in der Datei web.xml:

```
<context-param>
  <param-name>contextClass</param-name>
  <param-value>org.springframework.osgi.web.context.
    support.OsgiBundleXmlWebApplicationContext</param-value>
</context-param>

<listener>
   <listener-class>
     org.springframework.web.context.ContextLoaderListener
   </listener-class>
</listener>
```

Spring DM Web in Action: Ein praktisches Beispiel



Ausblick

- Weitere Informationen:
  - http://www.springframework.org/osgi
  - http://groups.google.com/group/spring-osgi
  - http://www.springframework.org/osgi/specification
- Spring-OSGi-Vortrag
  - http://www.eclipsecon.org/2008/index.php?page=sub/&id=298

Ausblick: Noch mehr Spring… ☺

- Auch für Clients kann Spring verwendet werden:
  - Dependency-Injection für Eclipse-RCP-Anwendungen
  - Server-Kommunikation mit Spring
- Weiter interessant:
  - Dependency Injection für Extension-Points (z.B. Views in Eclipse-RCP über Spring erzeugen und mit Dependencies versorgen

Die OSGi Service Platform –
Eine Einführung mit Eclipse Equinox

- Detaillierte Einführung in OSGi-Technologie
- April 2008, dpunkt.verlag
- ISBN 978-3-89864-457-0
- Ab sofort erhältlich

jax 08

Vielen Dank!!!

- … für die Aufmerksamkeit!

- Fragen jederzeit gerne
  - b.kolb@kolbware.de
  - gerd@gerd-wuetherich.de
  - martin.lippert@akquinet.de