Spring & OSGi kombiniert: Plattform der Zukunft

Martin Lippert, akquinet agile GmbH
Bernd Kolb, kolbware
Gerd Wütherich, freier Berater

Agenda

- OSGi – Ein Überblick
- Die Kombination von Spring und OSGi

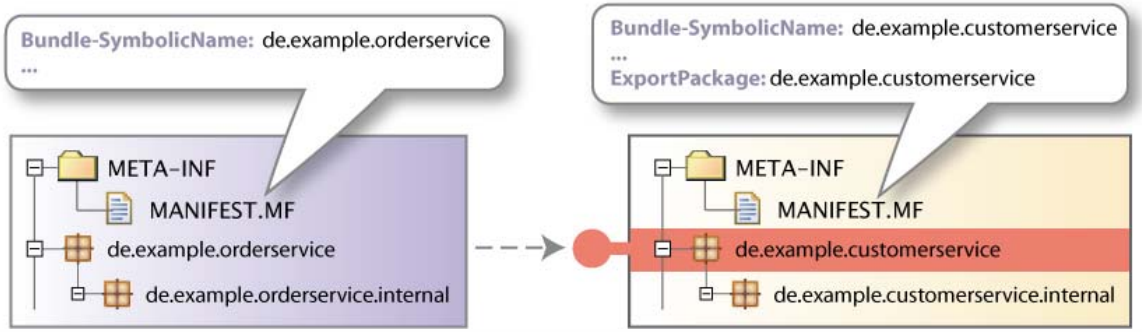
Die Ausgangsbasis

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

OSG – was?

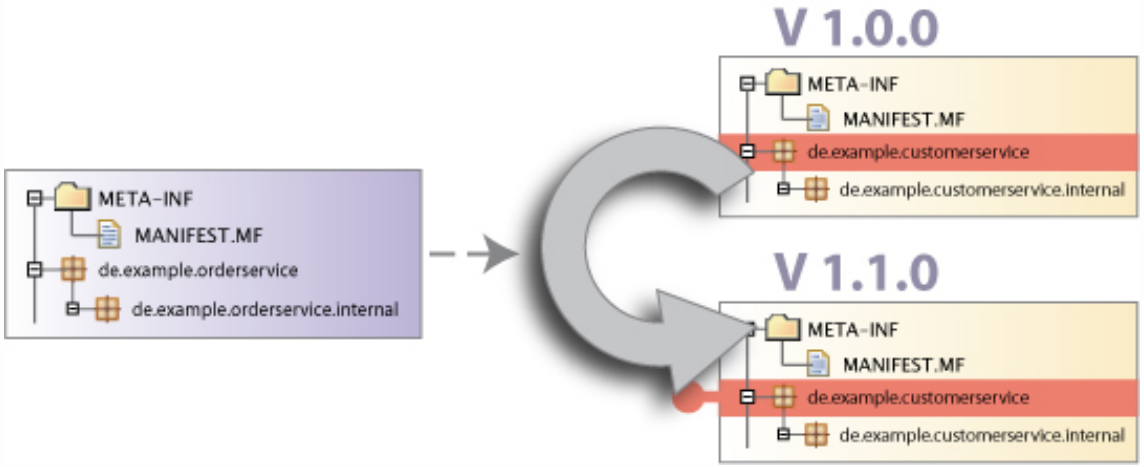
- OSGi™:
  - „A dynamic module system for Java“

OSGi ist …



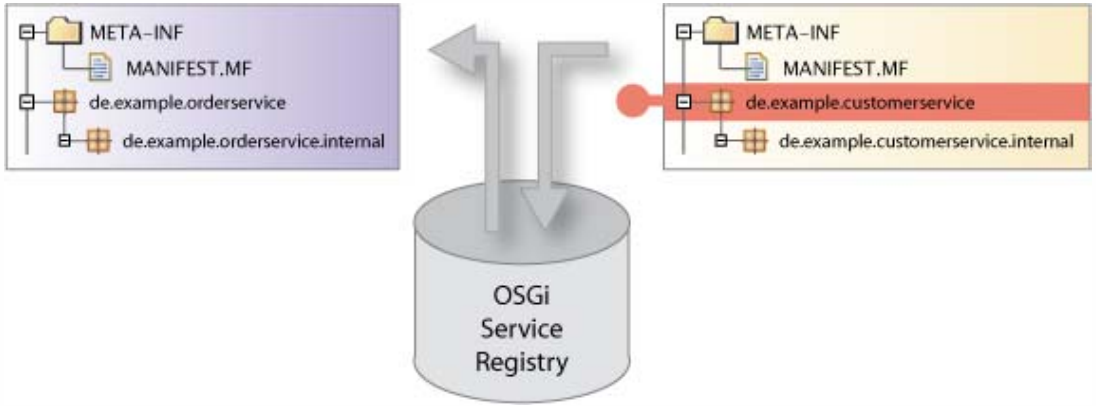
- … ein Modulsystem für Java, das die Definition von
  - **Modulen** (genannt „Bundles“),
  - **Sichtbarkeiten** von Modul-Bestandteilen (public-API vs. private-API)
  - **Abhängigkeiten** zwischen Modulen, sowie
  - **Versionen** von Modulen erlaubt.

OSGi ist …



- … dynamisch:
  - Module können dynamisch zur Laufzeit installiert, gestartet, gestoppt, deinstalliert und aktualisiert werden.

OSGi ist …



- … service-orientiert:
  - Bundles können Services veröffentlichen (dynamisch)
  - Bundles können über eine Service-Registry Services finden und verwenden
  - Die Runtime erlaubt, dass Services zur Laufzeit kommen und gehen

Woher kommt OSGi?

- OSGi-Alliance:
  - http://www.osgi.org/

- Wird seit 1999 mit einem Fokus auf Leichtgewichtigkeit und Dynamik entwickelt
  - Ursprünglich für Embedded-Systeme
  - Inzwischen verbreitet für Server- und Client-Systeme

Woher kriege ich OSGi?

- Open-Source-Implementationen:
  - Eclipse Equinox (http://www.eclipse.org/equinox/)
  - Apache Felix (http://cwiki.apache.org/FELIX/index.html)
  - Knopflerfish (http://www.knopflerfish.org/)
  - ProSyst mBedded Server Equinox Edition (http://www.prosyst.com/products/osgi_se_equi_ed.html)

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

(kein Anspruch auf Vollständigkeit)

Wo wird OSGi 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 und Oracle interessiert, JBoss prototypt OSGi-Umstellung
- Adobe
- …

Spring und OSGi kombiniert

- Spring-OSGi: Ein neues Mitglied der Spring-Familie
  - http://www.springframework.org/osgi

  - Keine eigene OSGi-Implementation, sondern eine Brücke zwischen Spring und OSGi-Implementationen
  - Erlaubt es, Spring-Anwendungen mit OSGi zu implementieren
  - Ist kompatibel zu Equinox, Felix und Knopflerfish

Das Ziel:

- Die Möglichkeiten von OSGi nutzen!
  - Anwendungen in Bundles aufteilen können, inkl.
    - Dependency Management
    - Sichtbarkeits-Definitionen
    - Versions-Management
    - Dynamic Management
  - Spring selbst in Form von Bundles zur Hand zu haben

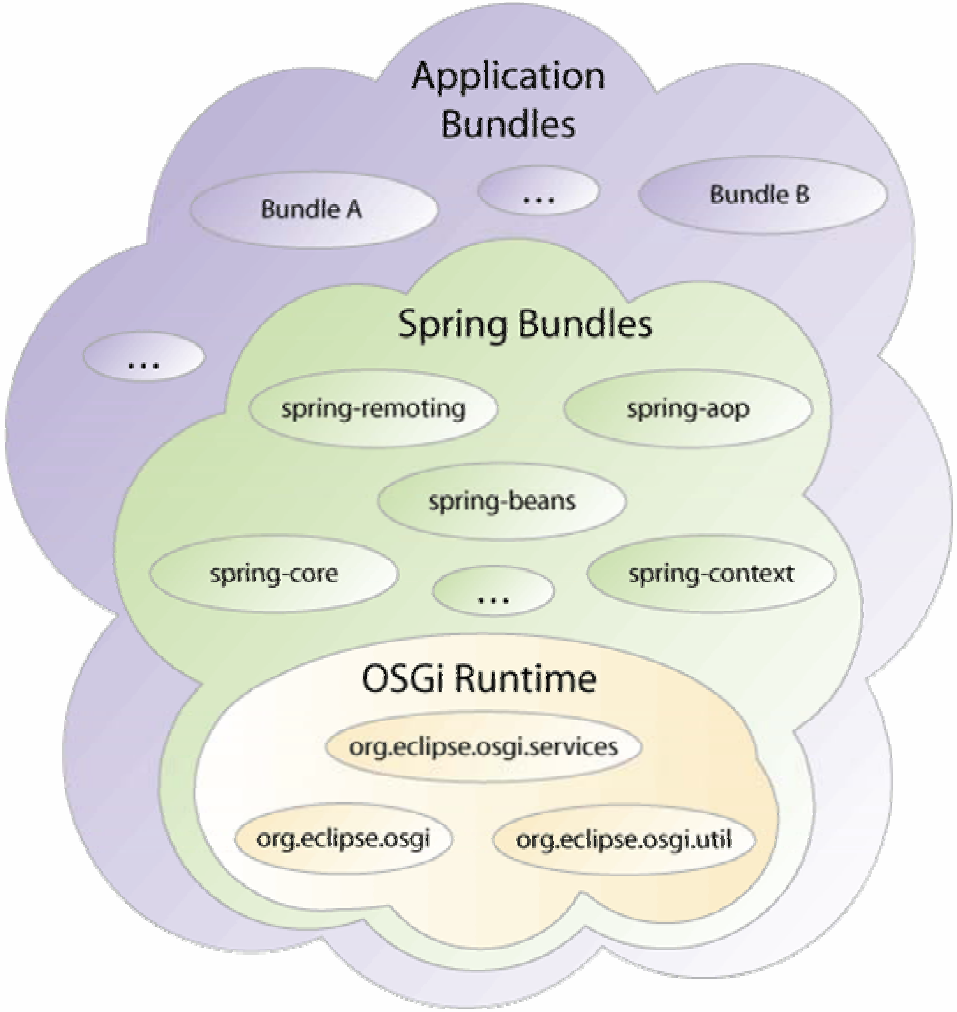
Randbedingungen

- 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 genauso nutzbar sein

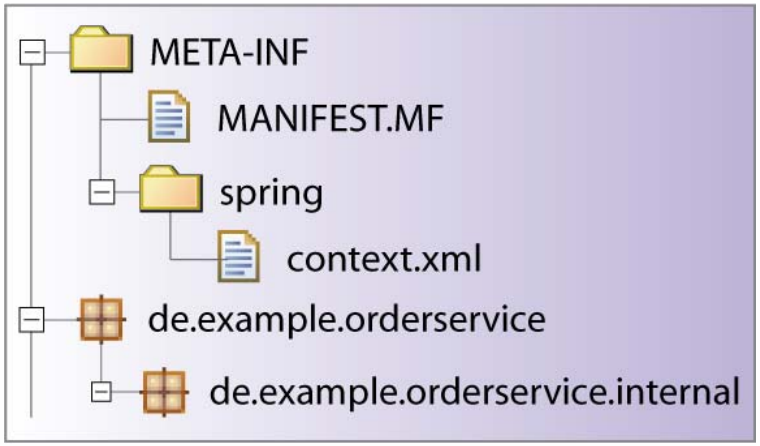
Spring als Bundles

(hier am Beispiel von Equinox)



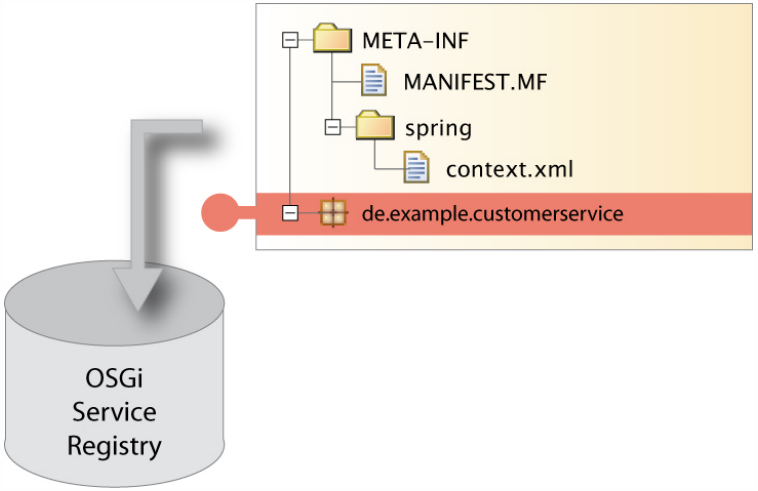
Bundles und Spring

- Ein Application-Context pro Bundle:
  - Spring-OSGi 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

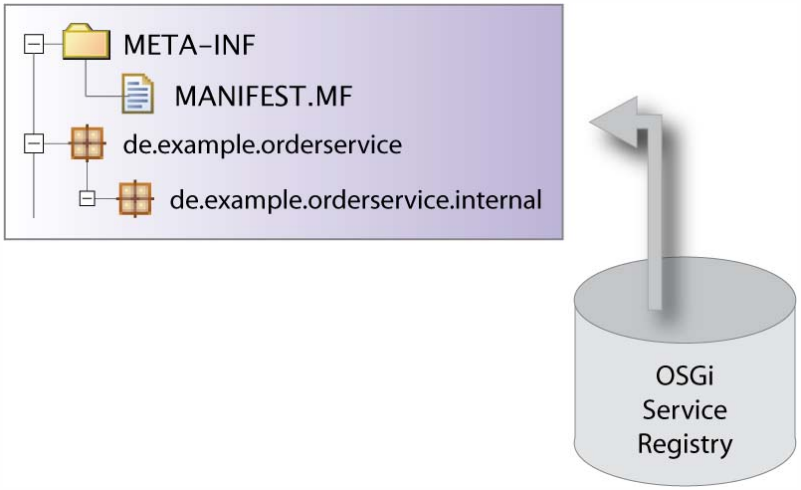



```
<beans>
  <bean name="customerService"
        class="de.example.customerservice.internal.CustomerServiceImpl" />

  <osgi:service id="customerServiceOsgi"
                ref="customerService"
                interface="de.example.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="de.example.customerservice.CustomerService"/>

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

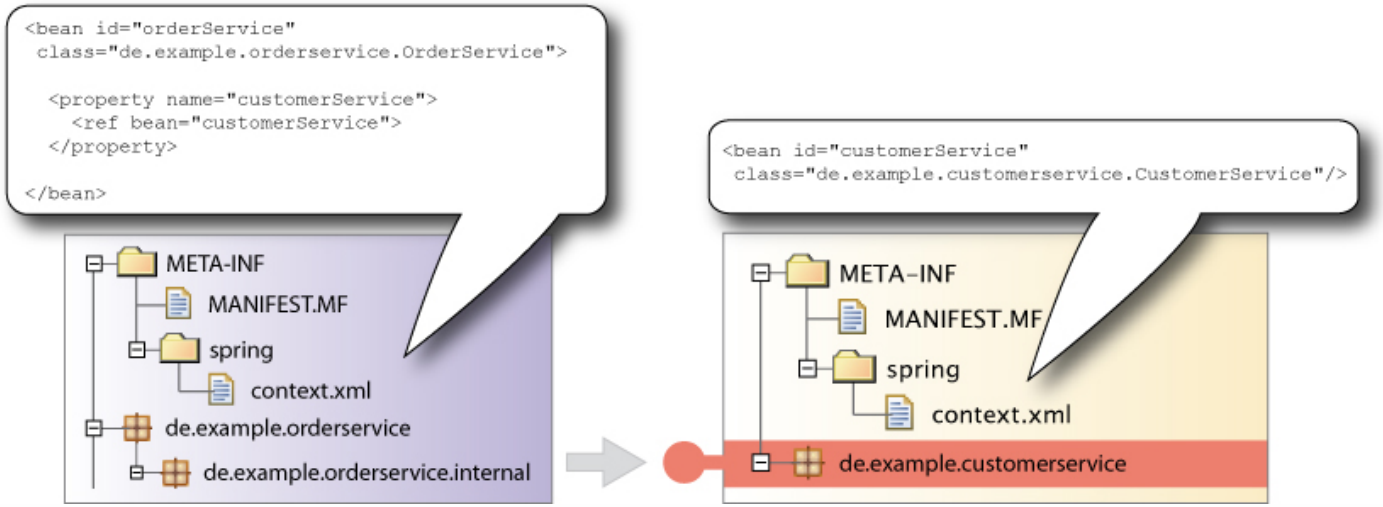
Resultat

- 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**

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"

Ein praktisches Beispiel



Der OrderService…

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

Der CustomerService…

- … stellt dem OrderService Kundenddaten zur Verfügung

Stand der Dinge

- Spring-OSGi 1.0 M1
  - am 5. April 2007 veröffentlicht
  - Enthält Basis-OSGi-Support (<osgi:…>-Namespace)
  - Context-Classloader-Handling
  - Unterstützung für Integrations-Tests
- Was noch fehlt:
  - Unterstützung für Web-Anwendungen (ist derzeit in Arbeit)
  - ???

Ausblick

- Release geplant zum Spring-Release 2.1
- Weitere Informationen:
  - http://www.springframework.org/osgi
  - http://groups.google.com/group/spring-osgi
  - http://www.springframework.org/osgi/specification
- Spring-OSGi-Tutorial (für Eclipse-Developer)
  - http://www.eclipsecon.org/2007/index.php?page=sub/&id=3632

Ausblick: Web-Anwendungen

- Standalone-Server-Applikation:
  - Läuft auf Basis von OSGi
  - Enthält beipspielsweise Jetty als Http-Service und Servlet-Container
- OSGi innerhalb eines Web- oder App-Servers:
  - OSGi innerhalb eines WAR deployen
  - Servlet-Bridge sorgt dafür, dass Requests an das entsprechende Bundle weitergeleitet werden

Ausblick: Web-Anwendungen

- In beiden Fällen:
  - Anwendung wird aus Bundles zusammengesetzt
  - Spring kann genutzt werden, inkl. Spring-Web-Support oder Remoting oder ähnlichem
- Deployment-Modell ist unabhängig davon
  - Im Entwicklungs-Modus der embedded-Jetty
  - Beim Deployment im Test oder Produktion dann die Bridge

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)

Ein Beispiel

- … aus der Praxis:
  - Anwendungen als Menge von Bundles
  - Remoting mit Spring-Remoting per HTTP
  - Client basiert beispielsweise auf Eclipse-RCP und Spring
  - Server basiert auf Spring und OSGi innerhalb eines Tomcat

Vielen Dank

- … für die Aufmerksamkeit!!!

- Vielen Dank an *Adrian Colyer*, *Costin Leau* und *BJ Hargrave* !!!

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