Peter Roßbach ◆ Gerd Wütherich ◆ Martin Lippert

Mit OSGi Webanwendungen entwickeln

Was geht, was nicht?

Agenda

- Warum Webanwendungen mit OSGi?
- Architekturmodelle
  - Web-Container innerhalb von OSGi
  - OSGi innerhalb eines Web-Containers
- Ausblick

Modularisierung und Webanwendungen

- Modularisierung in Java kommt!
  - OSGi
  - Java Module System
  - Was ganz anderes?
- Webanwendungen sind ein wesentliches Einsatzgebiet von Java

- **Wir brauchen eine Antwort, wie das zusammenpasst!**

Typische Einsatzgebiete von OSGi

- Eclipse-SDK
- RCP-Anwendungen
- Standalone-Anwendungen
- Embedded-Systeme

- **Und Webanwendungen?**

Warum OSGi?

- Aus Entwicklersicht:
  - Modularisierung für Webanwendungen
  - Klares Abhängigkeitsmanagement
  - JAR-Hell endlich hinter sich lassen
  - Dynamik ermöglichen

Warum OSGi?

- Aus Betriebssicht:
  - Updates in Produktion ohne Herunterfahren einer App
  - Granularität von Updates (Bundles statt ganze App)
  - Dynamik ermöglichen
  - Flexibles Management
  - Verschiedene Versionen gleichzeitig betreiben

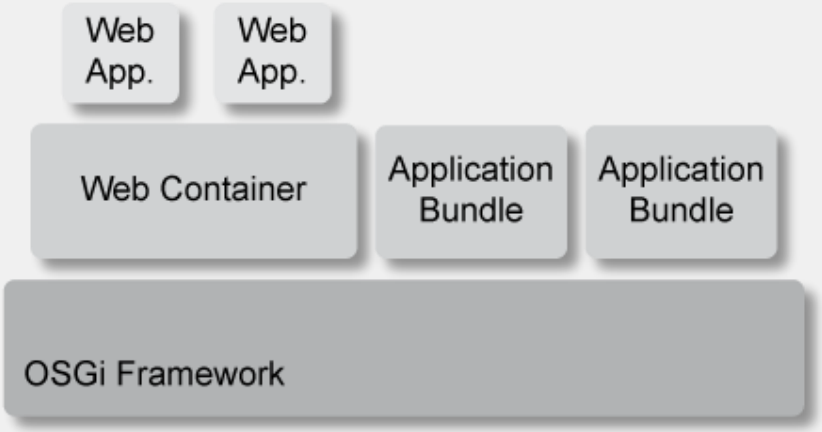
Was bedeutet das?

- **Webanwendungen bestehen aus OSGi-Bundles**
  - Keine klassischen WAR-Files mehr
  - Stattdessen Standard-OSGi-Bundles
  - Libraries in eigenen Bundles
  - Web-UIs in getrennten Bundles
  - Nutzung von OSGi-Services
  - etc.
- **Umdenken im Bau von Anwendungen!**
  - Eine gute Struktur entsteht nicht von alleine

Web-Container und OSGi

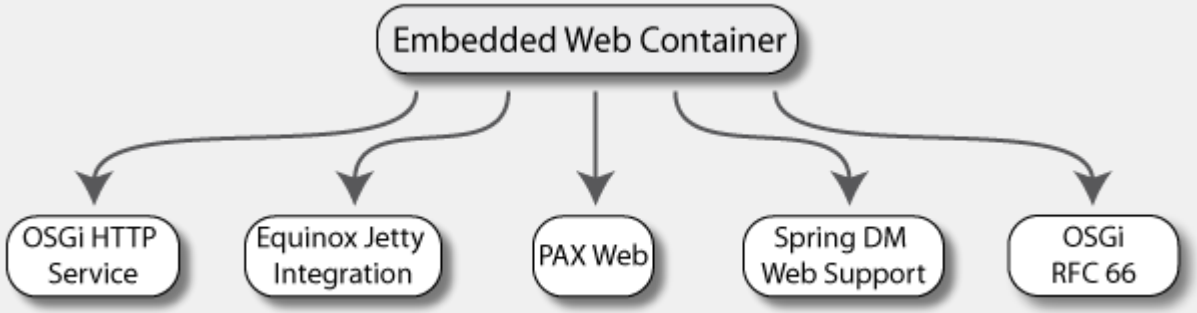
- Die OSGi-Welt besteht aus OSGi-Bundles, die in einer OSGi-Runtime laufen
- **Wo bleibt der klassische Web- und Servlet-Container?**

Architekturmodell 1



- Web Container läuft innerhalb des OSGi Frameworks
- Web Container wird als Bundle innerhalb des OSGi Frameworks installiert und gestartet
- Anwendungs-Bundles können Webapplikationen (oder Teile davon) in den Web Container deployen

Übersicht „Embedded Web Container“



- Fragen:
  - Wie werden Webanwendungen (oder Teile davon) in den eingebetteten Container deployed?
  - Welche Elemente der Servlet-Spezifikation werden unterstützt?

OSGi HTTP Service I

- OSGi Standard Service seit R1
- Zugriff auf den Servlet Container über das Interface „org.osgi.service.http.HttpService“
- Servlets und Ressourcen können dynamisch registriert und deregistriert werden

OSGi HTTP Service II

- Rudimentäre Unterstützung der Servlet-Spezifikation
  - Servlets
  - Resourcen

- Unterstützte Web-Elemente:

Element	Element in web.xml
Servlets	<servlet>
Servlet init params	<servlet> <init-param>
Servlet mappings	<servlet-mapping>
Mime mappings	<mime-mapping>

Equinox-Jetty-Integration

- Alternative Implementierung des OSGi HTTP Service auf Jetty-Basis
  - JSP-Support (Jasper-Engine)
  - Anmelden von Servlets, JSPs und Ressourcen über Extension Points
- Ansonsten gleiche Beschränkungen wie beim OSGi HTTP Service

PAX Web I

- http://wiki.ops4j.org/display/paxweb/Pax+Web
- Basiert auf Jetty
- Erweitert den OSGi HTTP Service
  - Dynamisches An- und Abmeldung von Web Elementen über das Interface „org.ops4j.pax.web.service.WebContainer“
  - JSP-Support

PAX Web II

Element	Element in web.xml
Context params	<context-param>
Session timeout	<session-timeout>
Servlets	<servlet>
Servlet init params	<servlet> <init-param>
Servlet mappings	<servlet-mapping>
Filter	<filter>
Filter init params	<filter><init-param>
Filter mappings	<filter-mapping>
Listeners	<listener>
Error pages	<error-page>
Welcome files	<welcome-file-list>
Mime mappings	<mime-mapping>

PAX Web – Alternative Deployment-Modelle

» Pax Web Extender – Whiteboard

- Implementierung des Whiteboard Patterns
- Servlets, Ressourcen etc. werden selber als OSGi Services an der OSGi Service Registry implementiert

» Pax Web Extender – War

- Implementierung des Extender Patterns
- Ermöglicht das Deployment von „bundle-fizierten“ WARs
- Spezifikation der Webelemente über „WEB-INF/web.xml“

Spring Dynamic Modules - Web Support I

Spring Dynamic Modules:

- Formerly known as „Spring-OSGi“
- Mitglied der Spring-Familie
- http://www.springframework.org/osgi
- Keine eigene OSGi-Framework-Implementierung, sondern eine Brücke zwischen Spring und OSGi-Framework

Spring DM Web Support:

- Fokus in Spring DM 1.1
- Integration von Spring DM mit Web-Applikationen

Spring Dynamic Modules - Web Support II

- „Natives“ Deployment von Webapplikationen
  - Tomcat-Support
  - Jetty-Support
- Volle Unterstützung der Servlet-Spezifikation 2.5

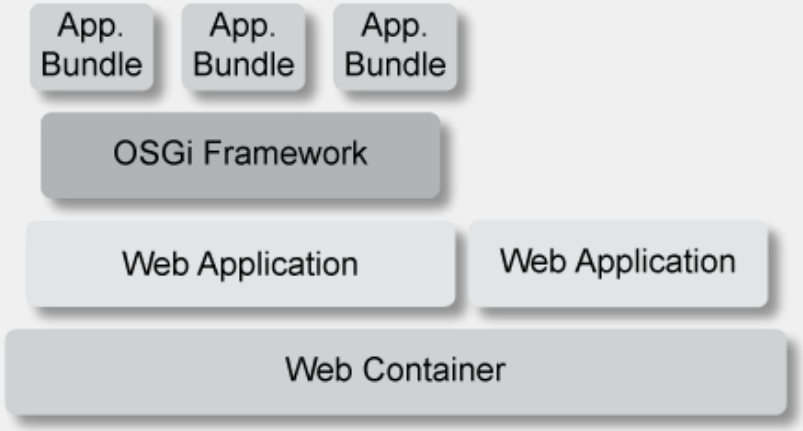
OSGi RFC 66 – OSGi and Web Applications

- Spezifikation
  - Derzeit als Draft verfügbar
- Beschreibt, wie Webanwendungen in OSGi unterstützt werden (Servlet Spec. 2.5, JSP Spec. 2.1)
- Referenzimplementierung:
  - Spring DM / Spring DM Server

Ich habe aber einen App-Server!

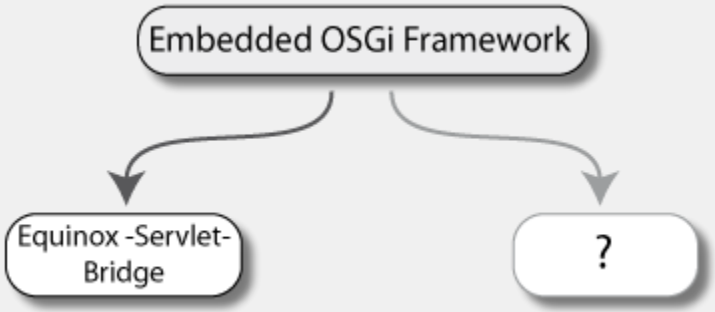
- Und ich darf nur EARs, WARs, etc. deployen
- Was nun?

Architekturmodell 2



- OSGi Framework eingebettet in Webanwendung
  - OSGi Framework wird innerhalb einer Webanwendung ausgeführt
  - Anwendungs-Bundles werden innerhalb des eingebetteten OSGi Frameworks installiert und gestartet

Übersicht „Embedded OSGi Framework“

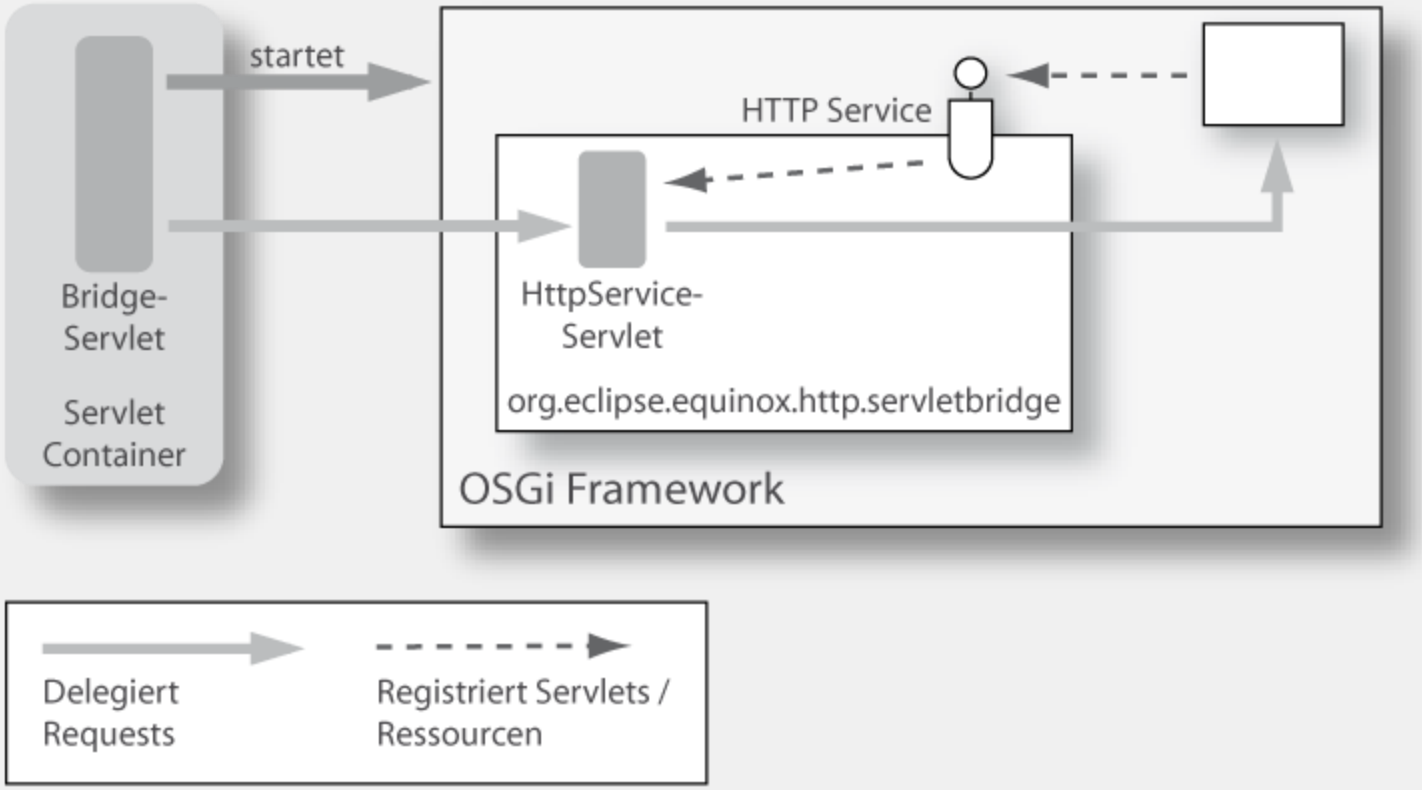


- Fragen:
  - Wie werden Webanwendungen (oder Teile davon) in den umgebenden Container deployed?
  - Welche Elemente der Servlet-Spezifikation werden unterstützt?

Eclipse Equinox Servlet-Bridge

- Bestandteil der Equinox-Distribution
- Ermöglicht das Ausführen eines OSGi Frameworks innerhalb einer Webanwendung

Eclipse Equinox Servlet-Bridge



Servlet-Bridge – Was geht? Was nicht?

- Alles über OSGi-Http-Service
- Vorteile:
  - Funktioniert überall
  - WAR-File wird deployed
  - Wenn gewünscht, Management über normalen Management-Agent von OSGi
- Nachteile:
  - Deployment erfolgt über das Interface „org.osgi.service.http.HttpService“ ⇨ Gleiche Einschränkungen wie beim OSGi HTTP Service

Wo stehen wir gerade?

- Es gibt eine OSGi Plattform, aber nur wenige nutzen diese Software zur Implementierung ihrer Webanwendungen
  - Möglichkeiten der Modularisierung und Versionierung nutzen
  - Verbesserung des Schnitts unserer Anwendungen
    - Statt einem WAR entstehen mehrere Module
  - Betrieb verschiedener Versionen
  - Laufzeitdynamik für Zero Down Time nutzen

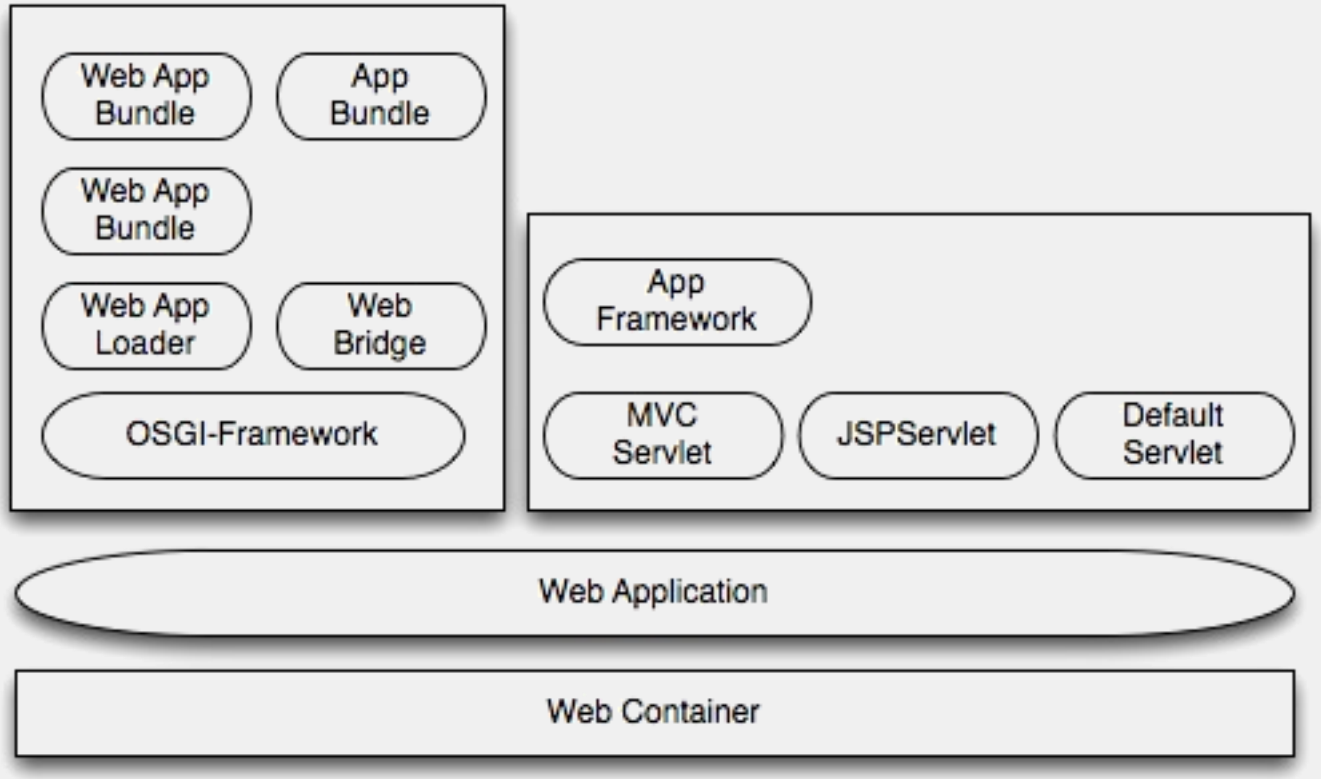
Kandidaten

- OSGi Runtime in einer WebApp
  - Nutzung einer ServletBridge
  - Wiederverwendung der bestehenden Frameworks
  - Schrittweise Integration
  - Reduktion der Möglichkeiten des API‘s
- WebContainer im OSGi Runtime
  - Laden der War‘s in den Container
  - RFC 66

Die Aufgabe

- Umbau der bestehenden Webframeworks
  - Integration der OSGi Services in die „Factories“
  - Aktion, Controller, Bean ,....
  - Mapping von Request zu Abarbeiter
  - Delegation der Verarbeitung
    - Content Aufbereitung (JSP, Template, ...)
- Nutzen bestehende Service API‘s
  - Datenbankzugriff
  - Reporting
  - Backend Zugriff

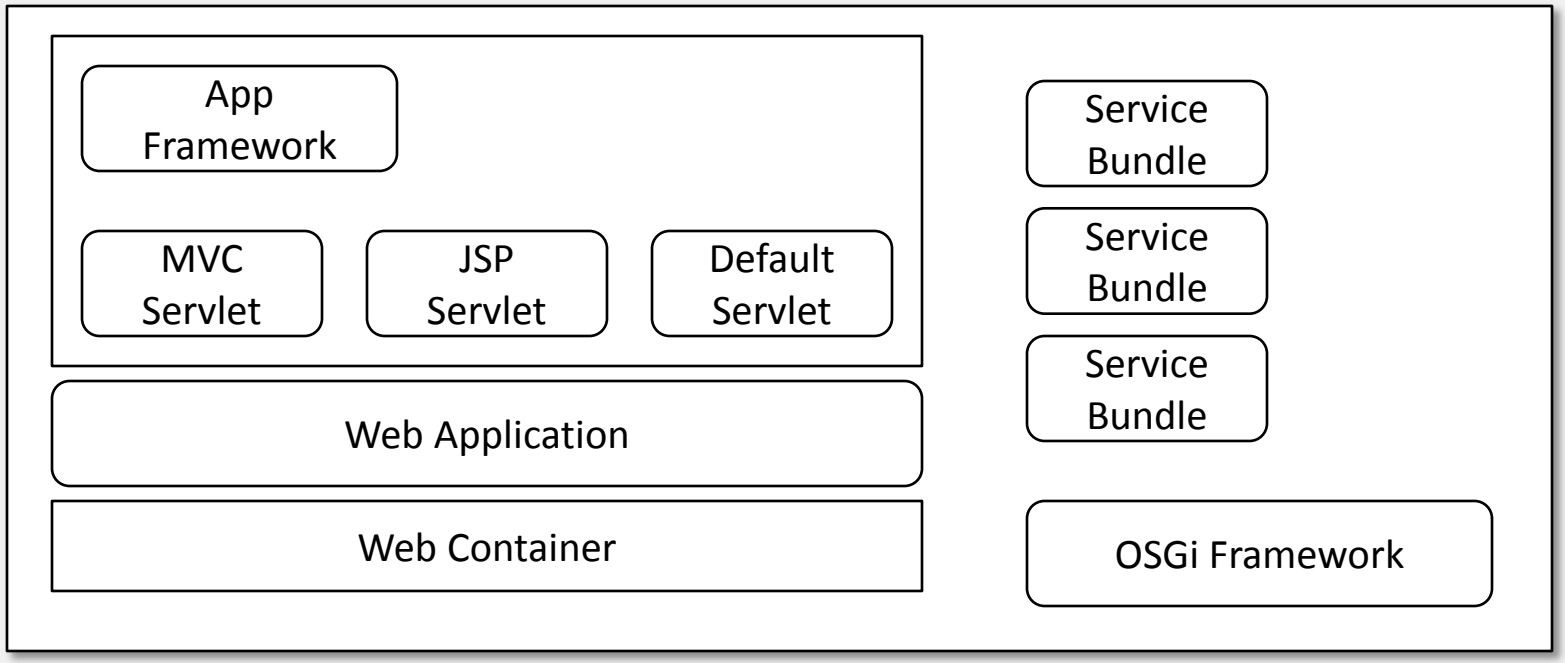
OSGi Runtime als Webanwendung



Bewertung

- Nebeneinander ist möglich
- Keine Änderung der Betriebsinfrastruktur
- Man muss einen Webcontainer nachbauen
  - Integration aller Servlet Objekte
    - Servlet
    - Listener
    - Filter

Webcontainer im OSGi Runtime - RFC 66



Webcontainer im OSGi Runtime

- Aus technologischer Sicht ist das die sinnvollere Variante
- Vorteile von Webcontainer und OSGI können voll ausgespielt werden
- Die Webcontainer haben schon dynamische API‘s und es gelten nicht die Einschränkungen der Servlet API
- Aber:
  - Die bestehende Infrastruktur muss sich ändern
  - Anpassung für jeden Container erforderlich

Fazit

- Ideen sind im Fluss
- Viele Fragen
- Zurückhaltende Verwendung

- Wir sehen ein großes Potential

Thank you!