



Mit OSGi Webanwendungen entwickeln – Was geht, was nicht?

Peter Roßbach (Systemarchitekt)

Gerd Wütherich (Freier Softwarearchitekt)

Martin Lippert (akquinet it-agile GmbH)



Überblick

- Warum Web-Anwendungen mit OSGi?
- Architektur-Modelle
 - ◆ Web-Container innerhalb von OSGi
 - ◆ OSGi innerhalb eines Web-Containers
- Zukunft der Web-Anwendungen
 - ◆ Dynamik
 - ◆ Module
 - ◆ Bedeutung für Web-Frameworks?
- Ausblick



Modularisierung und Web-Apps

- Modularisierung in Java kommt!
 - ◆ OSGi
 - ◆ Java Module System
 - ◆ Was ganz anderes?
- Web-Apps sind ein wesentliches Einsatzgebiet von Java
- **Wir brauchen eine Antwort, wie das zusammenpasst!**



OSGi – Im Überblick

- „Dynamic Module System for Java“
- Modul-Definitionen, inkl.
 - ◆ Sichtbarkeiten
 - ◆ Abhängigkeiten
 - ◆ Lebenszyklus
 - ◆ Dynamik
- Spezifikation + unterschiedliche Implementierungen



Typische Einsatzgebiete von OSGi

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

- **Und Web-Anwendungen?**



Warum OSGi?

- Aus Entwickler-Sicht:
 - ◆ Modularisierung für Web-Anwendungen
 - ◆ Klares Abhängigkeits-Management
 - ◆ JAR-Hell endlich hinter sich lassen
 - ◆ Dynamik ermöglichen

- Aus Betriebs-Sicht:
 - ◆ 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?

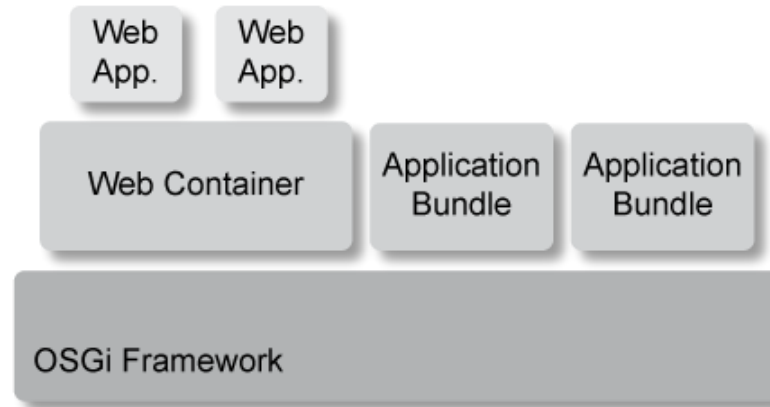
- **Web-Anwendungen bestehen aus OSGi-Bundles**
 - ◆ Keine klassischen WAR-Files mehr
 - ◆ Stattdessen Standard-OSGi-Bundles
 - ◆ Libraries in eigenen Bundles
 - ◆ Web-UIs in getrennten Bundles
 - ◆ Nutzung von OSGi-Services
 - ◆ etc.
- **Umdenken im Bau von Anwendungen!**
 - ◆ Eine gute Struktur entsteht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



Web-Container und OSGi

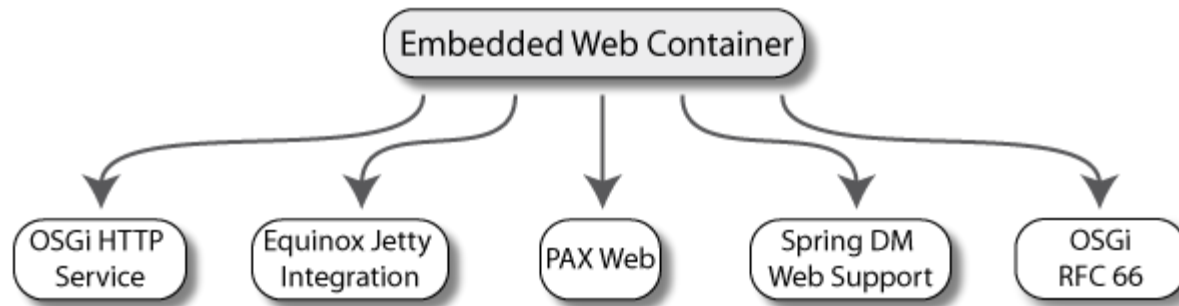
- Die Welt besteht aus OSGi-Bundles,
 - ◆ die auf einer OSGi-Runtime laufen
- **Wo bleibt der klassische Web- und Servlet-Container?**

Architekturmodell 1



- Web Container läuft innerhalb des OSGi Frameworks
 - ◆ Web Container wird als Bundle innerhalb des OSGi Frameworks installiert und gestartet
 - ◆ Anwendungs-Bundles können Web-Applikationen (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

- 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 – Was geht? Was nicht?

- Rudimentäre Unterstützung der Servlet-Spezifikation
 - ◆ Servlets
 - ◆ Ressourcen
- 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

- <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 – Was geht? Was nicht?

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

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 DM Web Support – Was geht? Was nicht?

- „Natives“ Deployment von Web-Applikationen
- Volle Unterstützung der Servlet-Spezifikation 2.5



OSGi RFC 66 – OSGi and Web Applications

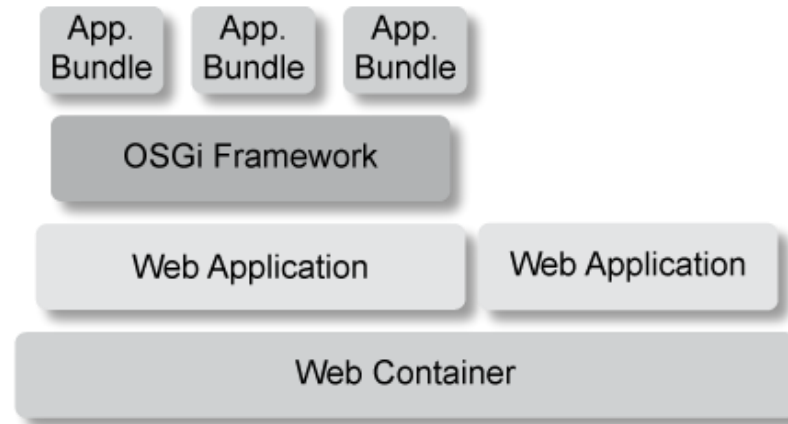
- Spezifikation
 - ◆ Derzeit als Draft verfügbar
 - ◆ Bestandteil der OSGi Spec. 4.2
- Beschreibt, wie Web-Anwendungen in OSGi unterstützt werden (Servlet Spec. 2.5, JSP Spec. 2.1)
- Referenzimplementierung:
 - ◆ Spring DM / Spring DM Server



Ich habe aber einen App-Server!

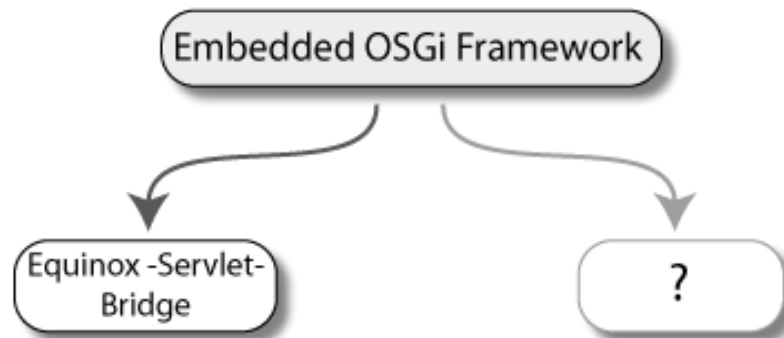
- Und ich darf nur EARs, WARs, etc. deployen
- Was nun?

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?

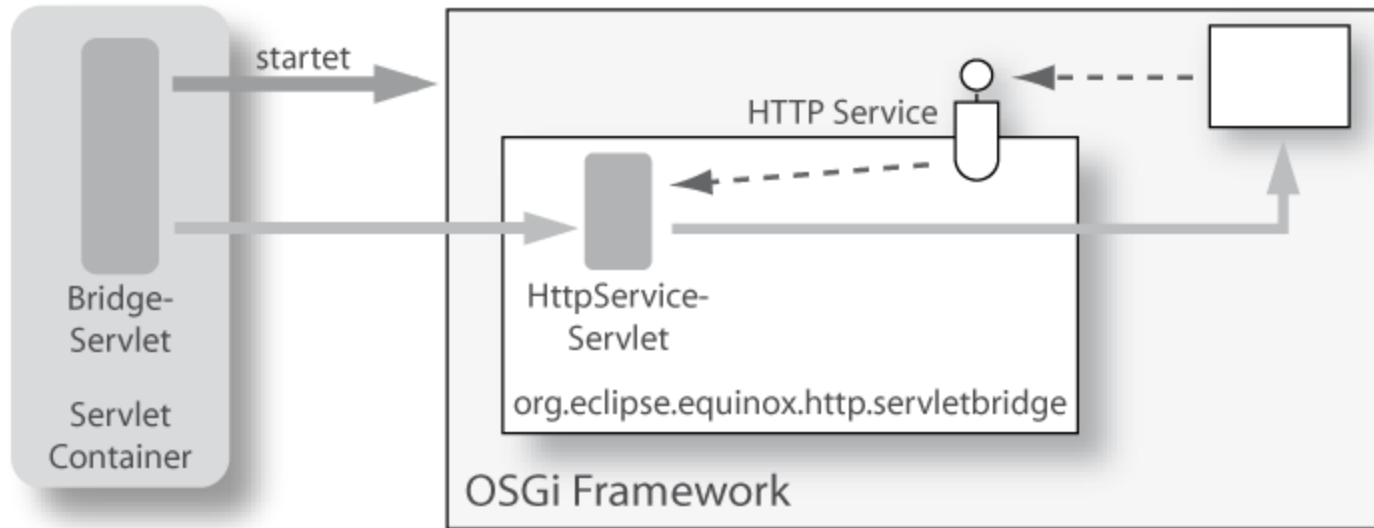


Eclipse Equinox Servlet-Bridge

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



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 Module
 - ◆ Betrieb verschiedener Versionen
 - ◆ Laufzeitdynamik für Zero Down Time nutzen
- Servlet API 3.0 Early Draft liegt vor



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

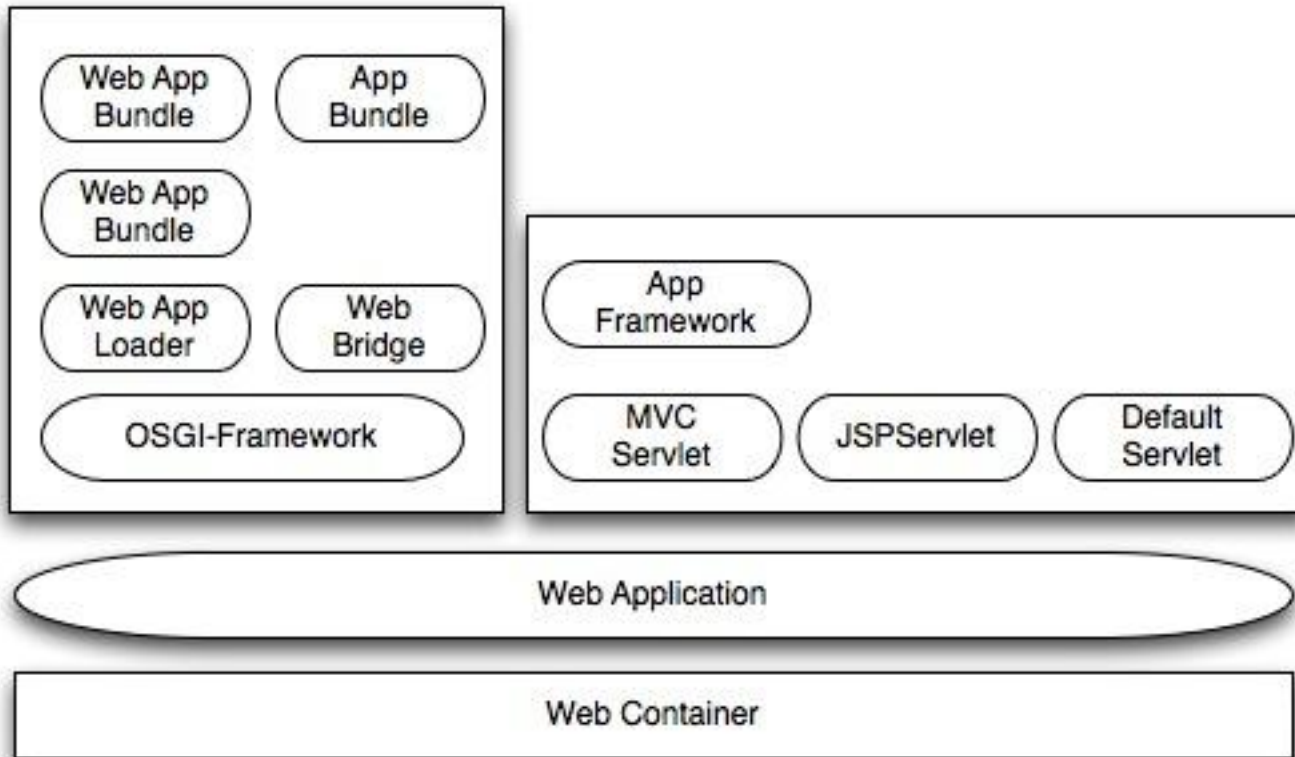


Die Aufgabe

- 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

- Aus technologischer Sicht ist das die sinnvollere Variante
- Vorteile von Webcontainer und OSGi können voll ausgespielt werden
- Die Webcontainer habe 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.



Vielen Dank für die Aufmerksamkeit

- Fragen und Feedback jederzeit willkommen!

Peter Roßbach: pr@objektpark.de

Gerd Wütherich: gerd@gerd-wuetherich.de

Martin Lippert: lippert@acm.org