Java Enterprise Edition 6 & Projekt GlassFish

Daniel Adelhardt
Senior Software Architect
Sun Microsystems GmbH

Agenda

- Java Enterprise Edition Rückblick
  - Java EE 5 – Status Quo
  - Motivation für Java EE 6
- Java Enterprise Edition 6 Ausblick
  - Generelle Änderungen
  - Enterprise Java Beans API 3.1
  - Java Persistence API 2.0
  - Java Server Faces 2.0
  - Misc.
- Projekt GlassFish als Java EE 6 Implementierung

Java Enterprise Edition 5

- Java Enterprise Edition ist seit mehr als 2 Jahren am Markt verfügbar
  - Sehr schnelle und breite Akzeptanz bei Entwickler
  - 9 verschiedene Implementierungen verfügbar
- Weshalb ist die Akzeptanz so gut?
  - Java EE 5 bringt viele Vereinfachungen mit sich
  - “Ease of Development”
  - “Convention over Configuration”
  - Eliminierung vieler althergebrachter Ansätze
  - Vereinfachtes Packaging & Deployment

Java Enterprise Edition 5 Zertifiziert

Apache Geronimo 2.1.1

BEA WebLogic 10

IBM WASCE 2.0

Apusic Application Server v5.0

Oracle Application Server v11.0

SAP NetWeaver v7.1

Sun Java System Application Server 9.x

TmaxSoft JEUS 6

GlassFish v1/v2

http://java.sun.com/javaee/overview/compatibility.jsp

Java EE 5 Inhalte im Detail

- JAX-B (JSR-222)
- JAX-WS (JSR-224)
- StAX (JSR-173)
- Web Services Metadata (JSR-181)

XML/Web Services

- JSP Standard Tag Library (JSR-52)
- Java Server Faces 1.2 (JSR-252)

Web Technologien

- EJB 3.0 und Java Persistence API (JSR-220)
- Common Annotations (JSR-250)

EoD

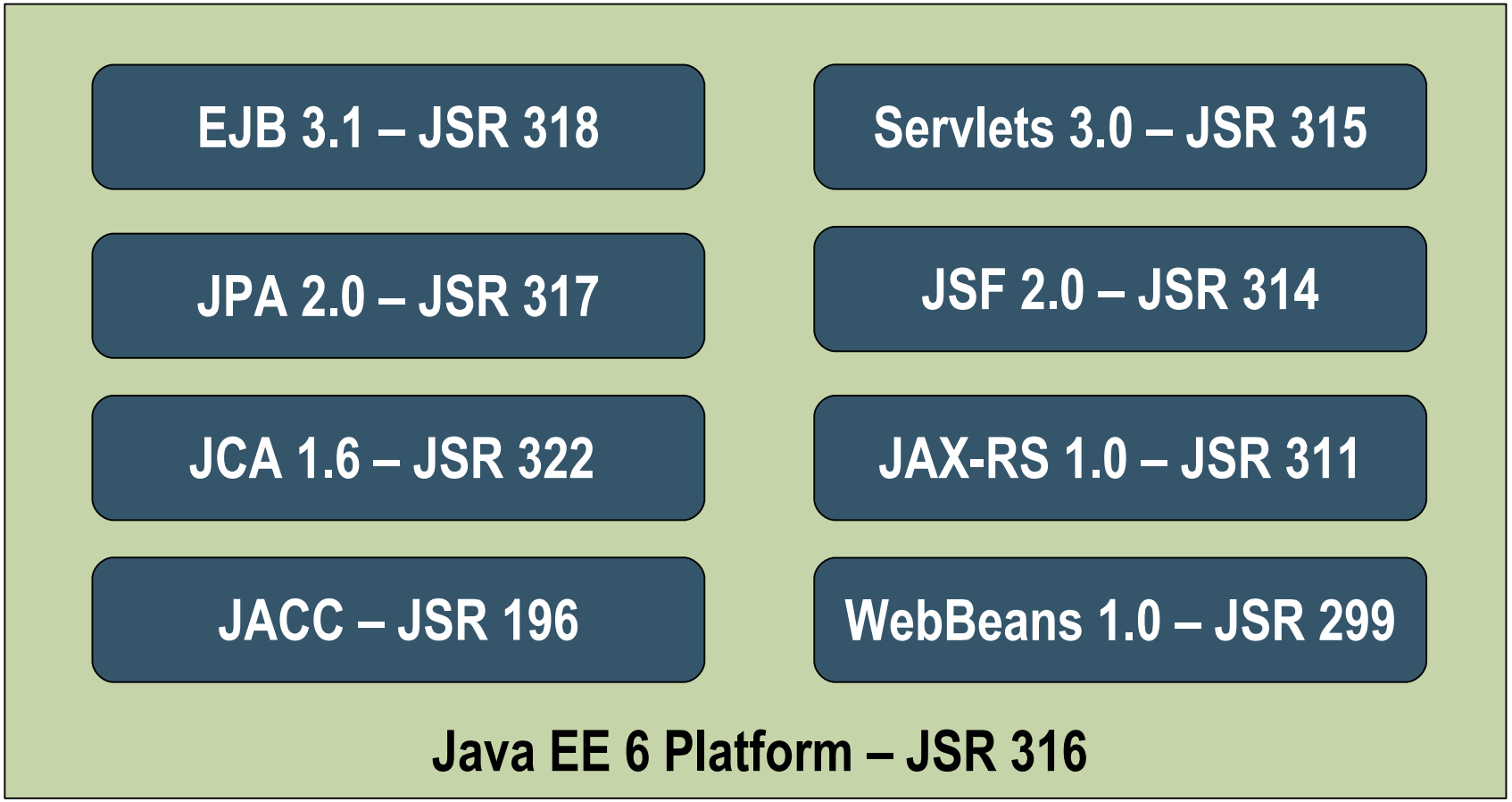
Disclaimer

Die Java EE 6 Spezifikationen sind noch nicht final. Alle Inhalte daher subject to change!

Die Ziele von Java EE 6

- “Rightsizing” der Plattform
  - Vollständige Plattform vs. Modulare Plattform mit Profilen
- Weitere Vereinfachung der Entwicklung
  - Java EE 5 vereinfachte die Entwicklung mit EJBs/JPA und Web Services
  - Java EE 6 vereinfacht die Entwicklung von Web Applikationen
- Erweiterbarkeit – Plugin Schnittstellen für Open Source Frameworks

Der Umfang von Java EE 6(tbc)



Plus Updates für JAX-WS 2.2, JSP 2.2, EL 1.1, Web Services 1.3, Common Annotations, JAX-B 2.2

Generelle Änderungen in Java EE 6

- Einführung von Java EE Profilen
  - Profile als Sub- oder Supermenge von Java EE
  - Web Profile(Profile A/B) als erster Vorschlag
    - Enthält Submenge von Java EE Technologien für Web Applikationen wie Servlet 3.0, JSP 2.2, JSF2.0(*), Web Beans(*), JAX-RS, JAX-WS etc.
- Deployment von EJBs in Web Modulen
  - Betonung des POJO Charakters von EJBs
  - Vereinfachtes Packaging für Entwickler
- Pruning für veraltete Technologien
  - Verschlanken der Plattform durch Entfernen von APIs

Enterprise Java Beans 3.1

- Behutsame Erweiterung von EJB 3.0 um folgende Aspekte
  - Vereinfachung der Entwicklung
    - Vereinfachte Local View auf EJB Session Beans
    - EJB Deployment in Web Archiven (.war Files)
    - “EJB lite” - Subset von EJB für Web Profile
    - Global JNDI Names
    - Container Bootstrapping API für Unit Testing
  - Neue Funktionalität
    - Singleton Session Beans
    - EJB Timer Erweiterungen: Automatische Erzeugung von Timern und Calender Expressions
    - Asynchrone Aufrufe

EJB 3.1 – “No Interface Local View”

- EJB 3.0 sieht für EJB Komponenten ein Business Interface vor
  - > EJB 3.1 vereinfacht die Verwendung von Local EJBs per No Interface Local Views
  - > Nur public Methoden der Bean exponiert

EJB 3.0 Local SB

```
@Local
public interface BarBeanLocal {
    void foo();
}
@Stateless
public class BarBean
    implements BarBeanLocal {
    public void foo(){...}
}
```

EJB 3.1 no-Interface

```
@Stateless
public class BarBean
    public void foo(){...}
}
```

EJB 3.1 – Vereinfachtes Deployment

- Co-Packaging von Beans in Web Archiven vereinfacht Deployment
  - > Single Naming & Classloader Environment

Pre-Java EE 6

MyEar.ear
/META-INF/application.xml

MyEJB.jar
/META-INF/ejb-jar.xml
/com/xyz/MyEJB.class

MyWeb.war
/WEB-INF/web.xml
/WEB-INF/classes/com/abc/MyServlet...

Mit Java EE 6

MyWeb.war

/WEB-INF/web.xml
/WEB-INF/ejb-jar.xml

/WEB-INF/classes/com/xyz/MyEJB.class
/WEB-INF/classes/com/abc/MyServlet.class

EJB 3.1 Singleton Beans

- Singleton Beans werden pro JVM instanziiert
  - > Erhalten Zustand zwischen Calls
    - – Zustand muss nicht persistiert werden
  - > Container vs. Bean managed Concurreny
    - – @ReadOnly, @ReadWrite Lock Annotations

```
@Startup  //Startup Annotation für Inititalisierug bei Start
@Singleton //Marker Annotation für Singletons
@DependsOn("MySecondSingleton")  //Abhängigkeit zu 2.ter Bean
public class MySingletonEJB {

    private String sharedData="...";
    @PostConstruct
    void init () {...}

}
```

EJB 3.1 – Asynchrone Aufrufe

- EJB 3.1 führt optionale Asynchronität für Session Beans ein
  - Return Typen: `void` oder `java.lang.concurrent.Future<v>`
  - Verwendung: `@Asynchronous` Annotation für Klassen/Methoden und `javax.ejb.AsyncResult` als Helper Klasse
  - Client kann Abbruch der Operation per `Future<v>.cancel()` erreichen und Bean kann per `SessionContext.isCancelled()` Abbruch prüfen
  - Transaktionalität und Persistenz der Operation unterstützt

EJB 3.1 – Timer Service Neuerungen

- Der Timer Service wird von jedem EJB Container seit EJB 2.1 bereitgestellt
  - Verwendung in EJB 3.0 per TimedObject Interface oder @Timeout Annotation
- Neu in EJB 3.1: @Schedule Annotation für Unix cron like Timer Events
  - Beispiele:

```
@Schedule(dayOfWeek="Mon") //Jeden Montag 0 Uhr
@Schedule(minute="*",hour="*") // Jede Minute
```

  - @Schedule Annotation wird auf Methoden angewand
    - Methode kann optional Timer Objekt in Signatur nutzen

EJB 3.1 “lite” - Under Discussion

- Es gibt Überlegungen eine Submenge der EJB Spec für das Web Profile aufzunehmen
- Vorgeschlagene Submenge
  - Local Session Beans, Annotations / ejb-jar.xml
  - CMT / BMT, Deklarative Security, Interceptoren
  - Java Persistence API 2.0
- OHNE:
  - Message Driven Beans, EJB Web Service Endpoints
  - RMI-IIOP, 2.x / 3.x Remote view, 2.x Local view
  - Timer Service, CMP / BMP

Java Persistence API 2.0

- Design Ziele
  - Mehr Modellierungsmöglichkeiten
  - Erweitertes OR Mapping
  - Erweiterungen der Query Sprache
  - Erweitertes Handling von detached Instances
  - Validation
- Referenzimplementierung: EclipseLink Projekt

Java Persistence API 2.0

- Bessere Modellierung und Mapping
  - Collections von Basis Typen und Embeddable Typen

```
@ID protected String user;
@ElementCollection
@CollectionTable(name="Aliase")
@Column(name="Alias")
protected Set<String> aliase;
```

- Geordnete Listen
  - Spezifikation per @OrderBy Annotation für transiente Sortierung

```
@OneToMany()
@OrderBy("lastName")
List<Customer> kunden;
```

  - Nutzung von @OrderColumn für persistente Sortierung
    - Per Join Table

Java Persistence API 2.0

- Unterstützung für pessimistisches Locking
  - EntityManager Erweiterungen lock(), find(), refresh()
  - Query Erweiterungen setLockMode(), setHint()
  - @NamedQuery lockMode Attribut
- Caching per Java Persistence API 2.0
  - Interface für optionale Caching Infrastruktur eines JPA Providers
  - Simple API javax.persistence.Cache:
    `contains(entityClass, pk), evict(entityClass,pk), evictAll()`
- Integration mit JSR-303 (Beans Validation) für Validierung auf Entity Klassen

JAX-RS 1.0 – Restful Web Services

- Rest als Architekturstil für Web Services wird zunehmend populär - JSR 311 stellt eine Standard API für Java bereit
- Elementares über Rest
  - Jedes Objekt hat eine ID http://jfs2008.de/besucher/id4711
  - Verlinkung von Objekten
  - Multipe Repräsentation von Dingen in XML, JSON
  - Verwendung von Standard (HTTP) Methoden wie GET/POST/PUT/DELETE
  - Stateless Design
- Rest Services sind einfach zu skalieren und zu nutzen!

JAX-RS 1.0 - Überblick

- Verwendung der @Path Annotation um POJOs zu annotieren
  - > ID wird relativ zum Deployment Context vergeben
  - > Einbettbare Parameter per Annotation @PathParm

```
@Path("besucher/{visitor_id}")
public class JFSBesucher {
    @GET
    @ProduceMime({"application/xml", "application/json"})
    public JFSBesucherConverter get(@PathParam("visitor_id")
        String id
}
```

Servlet 3.0 API Neuerungen

- Pluggability
  - Verwendung von Addon Frameworks ohne extra Config
  - Modularisierung der web.xml – Include Mechanismus für web.xml Fragmente in 3rd Party Jar Files per **`<WebFragment>`** Element
  - Erweiterung der ServletContext API um beim Start Servlets, Filter, URL Mappings und Listener zu einer Web App hinzuzufügen
- Ease of Development
  - `@Servlet` Annotation mit url-Mapping
  - `@GET, @PUT, @POST, @DELETE, @HEAD, @HttpMethod` Annotationen für Methoden

Servlet 3.0 API Neuerungen

- Ease of Development Beispiel

```
@Servlet(urlMapping={"/myServlet"},name="MyServlet")
public class MySampleServlet {
    @GET
    public void myGet(HttpServletRequest req,
                      HttpServletResponse res)
    {...}
}
```

- Servlet Filter per Annotations

```
@ServletFilter
@FilterMapping(urlPattern="/myServlet/*")
public class MyFilter {
 public void doFilter(req,res)
```

Servlet 3.0 - Asynchronität

- Die berücksichtigten Use Cases für Asynchronität sind
  - Server Side AJAX Push per Comet
  - Asynchrone Web Services und Web Proxies
  - Ermöglicht non-blocking Verhalten des Containers bei langen Requests
- Erweiterung von ServletRequest um
  - void suspend(long timeOutMs); void resume();
  - void complete(); boolean isSuspended(); boolean isResumed(); boolean isTimeout();

Java Server Faces 2.0

- JSF 2.0 ist eine massive Überarbeitung der Spec mit folgenden Zielen
  - Einfachere Entwicklung eigener Komponenten
  - AJAX Support
  - Page Description Language
  - Vereinfachte Konfiguration
  - Kompatibilität zwischen JSF Implementierungen
  - Bookmarkfähigkeit
  - Neues State Management
  - Vereinfachtes Deployment
  - Standard Facelets Integration

Java Server Faces 2.0

- JSF Project Stages
  - Ähnlich wie (J)Ruby on Rails RAILS_ENV
  - Project Stage Varianten: Production, Development, UnitTest, SystemTest, Extension
  - Definiert als context-param im web.xml oder per JNDI unter java:comp/env/jsf/ProjectStage.
  - Zur Laufzeit per Application.getProjectStage() abrufbar

Web Beans – JSR 299

- Die Web Beans Spezifikation adressiert
  - Schaffung eines einheitlichen Komponentenmodells für Web-/Business Logik
  - State & Konversationsmanagement
  - Konfiguration von Komponenten
  - Enge Zusammenarbeit mit JSF und EJB3 – jedoch keine starre Abhängigkeit dazu!
- Bestandteile
  - Komponententyp+API + Annotations + Name + Scope
  - Komponententypen: Stateless+Statefull SB, Entity, PoJo

Web Beans – JSR 299

- Eigenschaften einer “WebBean”
  - > Applikationskomponente mit Business Logik
  - > Abhängigkeiten untereinander Container gemanaged
  - > Lifecycle Kontrolle durch Container
  - > Meist Statefull & Kontext bezogen

```
@Component
@SessionScoped
@Named(“FooFoo”)
public class Foo{
    public String foo(){
    ...
    }
}
```

```
@Component
public class Bar{
@In Foo myFoo;
}
```

```
<h:outputText
 value="#{FooFoo.foo}" >
```

Zeitplan für Java EE 6

- Viele der Spezifikationen sind bereits als Early Access Drafts verfügbar
  - EJB 3.1, JPA 2.0, Servlet 3.0, JSF 2.0,...
  - Erste Implementierungen bereits verfügbar
- Zeitplan sieht April/Mai 2009 für Java EE 6 Release vor
- Sun's Implementierung – GlassFish v3 wird zeitgleich veröffentlicht werden

Projekt GlassFish & Java EE 6

- GlassFish ist die Open Source Community für Java EE Technologien
  - Neue Technologien und Specs werden im Rahmen des GlassFish Projektes entwickelt
- GlassFish ist ein vollständiger Java EE Application Server ausgelegt für produktiven Einsatz
  - Hohe Performance & Skalierungsfähigkeit, Load Balancing & Clustering
- Mehr als 6 Millionen Downloads seit 2006

GlassFish - Ökosystem & Community

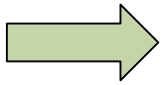

Open ESB
The *Open* Enterprise Service Bus

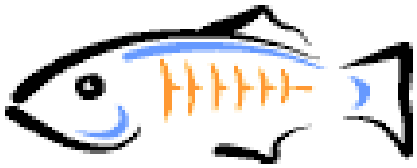

OpenDS
Open Source. Open Standards. Open Directory Se

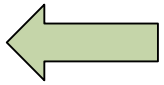
OpenSSO
Open Access . Open Federation

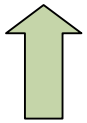

OpenPortal
Open Source. Open Standards. Open Portal Server.

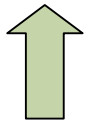
Erweiterungen bzw. Nutzung von GlassFish


24x7 Support




Tooling für GlassFish


Contributions durch Oracle, Ericsson, Community


Technologiebasis

Java Community Process

OpenJDK

GlassFish für Entwickler

GlassFish ist die offizielle Sun Java EE 5 Referenz

- 100% Java EE 5
- Support für Java 5 und 6 als Runtime
- Unterstützung für Java Business Integration (JSR-208)
- Top Web Services Stack (Projekt Metro) mit JAX-WS 2.x, JAX-B
- Unterstützung für WS-RM, WS-AtomicTX, MTOM,...
  - JMX Instrumentierung

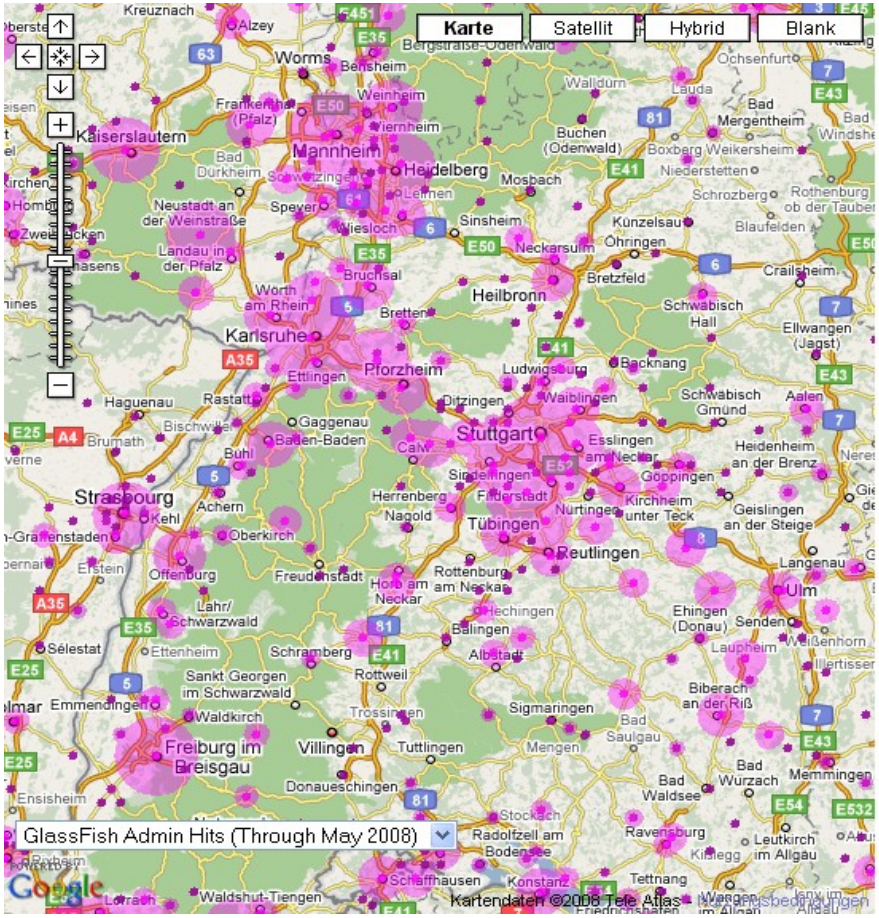
Vereinfachung der Entwicklung

Project Glassfish

- Developer Profil mit vereinfachter Architektur
- Tool Integration mit NetBeans und Eclipse WTP
- Scripting Fähigkeiten
- Schneller Start
- Web Start Integration
- Profiling / Debugging Tools integriert
- Umfangreiche Dokumentation, Tutorials, Blogs

Java™

GlassFish Nutzung - Stuttgart



Link To: Default View | This View

Current Zoom: 8 *(details at 14+)*

What is This Map?

Stats For Visible Area

GlassFish: 46788

solaris » Get it Now

NetBeans » Get it Now

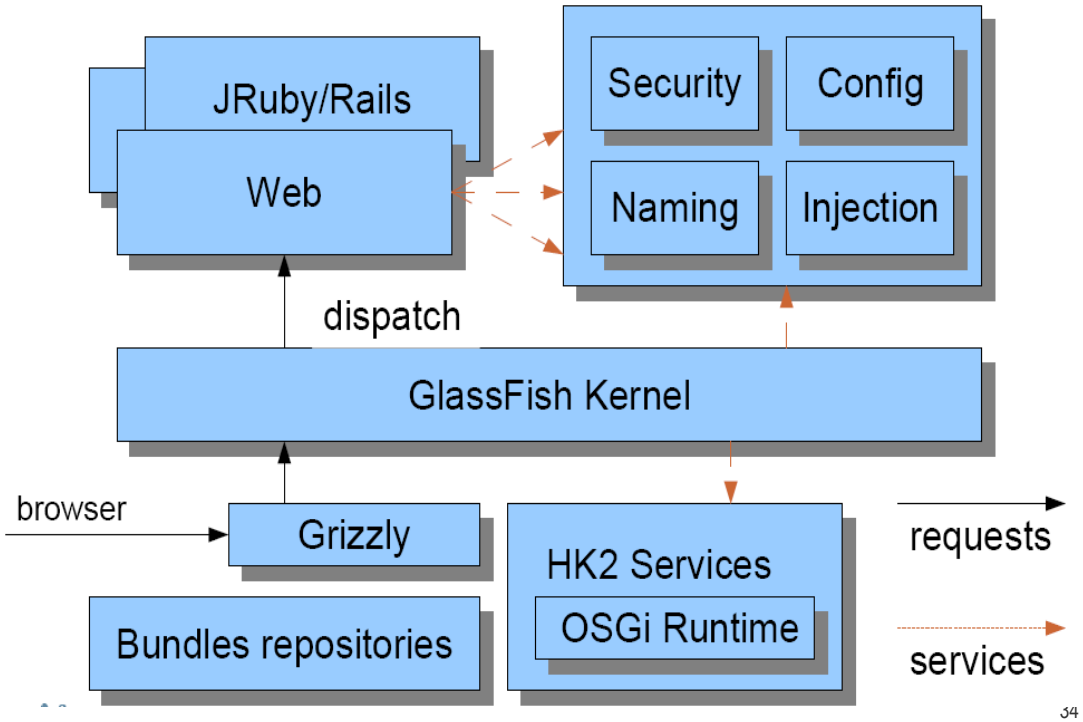
Project GlassFish

Sun microsystems

http://beta.glassfish.java.net:81/maps/

GlassFish v3

- Ziel von von v3 ist Modularität
  - Rightsizing der Container Architektur
  - Customizable & Embeddable Container
- GlassFish v3 wird auf OSGi Basis entwickelt
- GlassFish für Scripting: JRuby/Rails, Groovy/Grails, Php



Java Enterprise Edition 6 & Projekt GlassFish

daniel.adelhardt@sun.com