

## E.9 OSGi — Open Services Gateway Initiative

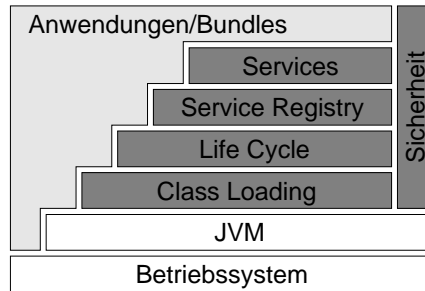
### 1 Überblick

- OSGi Spezifikationen:  
standardisierte, komponenten-orientierte Ausführungsumgebung für vernetzte Dienste
- Basis: Java
- Ziel-Anwendungsbereiche
  - Mobiltelefone
  - Automobil (Infotainment-Bereich)
  - Telematik
  - Eingebettete Systeme
  - Haustechnik
  - PC- und Mainframe-Anwendungen
- Literatur: <http://www.osgi.org/>

## 2 Einsatzbeispiele

- NG Smart Phones
  - einheitliche Plattform für Mobiltelefon-Anwendungen
  - Anwendungen zentral administrierbar
- Shell HomeGenie
  - Softwareplattform zur Integration von Kameras, Sensoren, Thermostaten, Stromschaltern, etc. im Hausautomatisierungsbereich
- Eclipse
  - Java-Entwicklungsumgebung
  - ab Version 3.0: Installation und Update von Plug-ins zur Laufzeit
- BMW iDrive (5er und 7er, neuer 3er)
  - Basis: Betriebssystem VxWorks und JVM Jeode von Windriver
  - OSGi-Implementierung von Siemens VDO
  - Integration von Navigation, Mobiltelefon (Bluetooth), Internet-Anbindung und anderen Infotainment-Anwendungen

### 3 Architektur



- mehrere unabhängige Anwendungen in einer JVM
- Anwendung = Bundle
  - JAR-Datei mit zusätzlichen Header-Informationen (Manifest)
  - Schnittstellenkonventionen -> Komponentenmodell
  - Installation, Update, Start, Stop und Deinstallation über Netzwerk

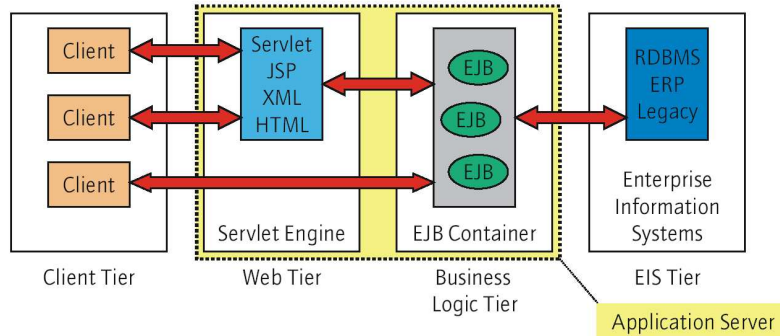
### 4 Services

- Class Sharing
  - Bundles können Pakete exportieren und von anderen Bundles importieren
  - vermeidet Redundanz von Code
  - ermöglicht Bereitstellung von Diensten in einer OSGi-Umgebung
  - Regeln zum Umgang mit unterschiedlichen Versionen
- Life Cycle Management
  - Dienste zur Installation von Bundles
- Service Registry
  - Nameservice
  - registrierte Objekte werden *Services* genannt: Schnittstelle und Menge von Eigenschaften (*Properties*)
  - automatische Abmeldung bei Bundle-Deinstallation + Information anderer Bundles

## E.10 Web-Services

### 1 EJB-Architektur (Details später in Kapitel EJB)

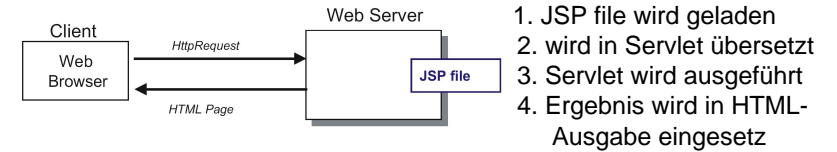
- Server-seitiges Komponentenmodell für *Multi-Tier* Anwendungen



- nicht mit JavaBeans verwechseln!

## 2 Java Server Pages

- HTML-Seiten mit integriertem Java-Code
- Schnittstelle zwischen Web-Seiten und verteilten objektorientierten Anwendungen



- Können eingesetzt werden
  - als Ersatz für traditionelle CGI-Skripten
  - als HTML-Schnittstelle zu objektorientierten Anwendungen (RMI, CORBA)

### 3 JSP Beispiel

```
<html>
<body>
<h2>Welcome to the Weather Site</h2> <hr>
Today's date is <%= new java.util.Date() %>
</body>
</html>
```

### 4 JSP Ziele

- Wiederverwendbarer Code wird in Komponenten oder JSP-Beans zur Verfügung gestellt
- Trennung von Web-Seiten-Design und Komponenten-Entwicklung

### 5 Servlets

- Grundproblem: Dynamik in Web-Seiten

- Lösungen:

- Client-seitig: JavaScript, Applets
- Server-seitig: CGI, ASP, LiveWire, PHP ...

- ➔ Servlets:

- “A servlet is a Java technology-based **web component**, managed by a container, that generates dynamic content. Like other Java-based components, servlets are platform-independent Java classes that are compiled to platform-neutral bytecode that can be loaded dynamically into and run by a Java-enabled web server. Containers, sometimes called servlet engines, are web server extensions that provide servlet functionality. Servlets interact with web clients via a **request/response** paradigm implemented by the servlet container.”

- Standard-API
- Plattformunabhängig Java
- Threads

- Ausführungsumgebung für Servlets: Servlet-Engine / Container

- Netzwerk-Service
  - (empfangen und zustellen von) Requests
  - (versenden von) Responds
  - Verwendung von HTTP (evtl. zusätzlich HTTPS, etc.)
- Steuerung des Lebenszyklus von Servlets / JSPs
  - laden, initialisieren, aufrufen, löschen
- Deployment
  - einrichten ("installieren") der Applikation