OSGi-Serviceplatform und SpringDM

Manuel Mauky
Saxonia Systems AG

OSGi Allgemein

- Open Services Gateway initiative
- IBM, Ericsson, Motorola, Sun, Deutsche Telekom, Oracle, Siemens,...
- Klassische Einsatzbereiche: Automobile, Heimautomatisierung, Handy, Embedded
- Aber auch: Eclipse IDE, Enterprise-Anwendungen

Bundles und Services

- Komponentenmodell mittels Bundles und Services
- Programmteile werden in Bundles gekapselt
- Bundle stellt Funktionalität als Service bereit
- Bundles und Services sind versionierbar

Bundles und Services

- Zugriffskontrolle auf Package-Ebene
- Bundle kann eigene Packages exportieren
- Exportierte Packages anderer Bundles können importiert werden
- Import/Export mittles Versionsnummer möglich
- Erweiterung der MANIFEST.MF in Jar-Datei

Life Cycle

- Start, Stop, Installation, Deinstallation und Update von Bundles zur Laufzeit
- Applikation ist (fern)administrierbar
- Abhängigkeiten werden durch Service-Registry automatisch aufgelöst

Implementierungen

- Diverse kommerzielle Frameworks
- Eclipse Equinox
- Apache Felix
- Knopflerfish
- …

Java-EE: Probleme

- Gesamtapplikation (WAR-Archiv) muss bei Änderungen neu deployed werden
- Bei Aktualisierung steht die Anwendung nicht mehr zur Verfügung
- Keine Versionierung von Bibliotheken

Lösung und neue Probleme

- Schlechtere Wiederverwendbarkeit wegen Abhängigkeiten auf OSGi-APIs
- Dynamik der Services muss bei Implementierung beachtet werden
- → OSGi + Spring DM

Spring Kurzgefasst

- Viele OpenSource-Frameworks unter einheitlicher API
- Auflösung von Abhängigkeiten mittels Dependency Injection
- Trennung von unterschiedlichen Belangen mittels AOP – Aspektorientierter Programmierung

Dependency Injection

- Objekte lösen Abhängigkeiten auf andere Objekte nicht selbst auf
- Referenzen werden in XML-Konfiguration zugewiesen (injeziert)
- Objekte sind Unabhängig von Umgebung
- Erleichtert Testen z.B. mittels Mocks

```
<beans
xmlns=„http://www.springframework.org/schema/beans“>
 <bean
   name=„database“
   class=„org.apache.commons.dbcp.BasicDataSource“
   destroy-method=„close“>

    <property name=„url“
       value=„jdbc:mysql://localhost:3306/test_db“/>
    <property name=„username“ value=„testuser“/>
    <property name=„password“ value=„geheim“ />
    <property name=„driverClassName“
       value=„com.mysql.jdbc.Driver“/>

  </bean>
</beans>
```

Spring DM

- Jedes Bundle besitzt eigenen Application-Context
- Klassen haben keine Abhängigkeiten auf OSGi-APIs
- OSGi-Services werden in Bean-Config definiert
- SpringDM puffert Methoden-Aufrufe
- SpringDM-Server

Fragen / Diskussion