Cross Enterprise Services with OSGi

Thilo Keber
Alex Dereviago
Christian Dedek

Introduction

- b.telligent optile
- Thilo Keber
- Alex Dereviago
- Christian Dedek

Quick Demo

Optile Server
- The idea
- Requirements
- Hi-level architecture
- Implementation details
- Hands On Demo

Conclusion
Java, XML und Open Source seit 1998

- Schulüsselte Realisierung von Java Software
- Individualsoftware
- Pilot- und Migrationsprojekte
- Sanierung von Software
- Software Wartung

- Unterstützung laufender Java Projekte
- Perfect Match
- Rent-a-team
- Coaching on the project
- Inhouse Outsourcing

- Schulungen, Coaching, Weiterbildungsberatung, Train & Solve-Programme
- Methoden, Standards und Tools für die Entwicklung von offenen, unternehmensweiten Systemen

© 2009 Orientation in Objects GmbH

e-Payment Landscape

Merchant

Partner Portfolio
e.g. Payment Providers

Payment Service Provider
Global Collect

Acquirer
Concardis

Clearing

Payment Network
MasterCard

Customer

Billing
Credit Card Statement

Issuer
Deutsche Kreditbank (Lufthansa)

Authorisation

Payment Method
Credit Card

Product
Two-year membership

Payment Service Provider
Global Collect

Acquirer
Concardis

Clearing

Payment Network
MasterCard
The Product

- Optile Server
  - Abstraction layer between merchant and payment processes
  - Abstraction of payment methods and payment operations
  - Extensible service oriented architecture
    - Dynamic deployment of services
  - Dynamic behaviour of payment processes
    - Selection of payment methods
    - Routing
    - Dynamic configuration of workflows
  - Backend defined Data Entry Forms
  - Interface to BI-Data Warehouse
Infrastructural Requirements

- **Initial Effort**
  - Licences
  - Availability(plattforms,os,integration with existing it world)
  - KnowHow, Consulting, Customizing
  - Lifecycle costs(administration,patching,updates etc.)

- **Performance**
  - minimizing resources spend for gateway infrastrucure
    - memory footprint(32bit VM Heapspace)
    - cpu time (32bit single core architecture)
    - network bandwith(10-100 Mbit/s)
      - Gateway call roundtrip (<1ms)

- **Scalability**
  - must not preclude growth in business
  - may produce more operational complexity
  - should have no influence on development speed

Operational Requirements

- **Complexity**
  - Hosted at merchant’s premises
  - sufficient representations for services/components
  - Granularity,Containment,Layering
  - system complexity should not reflect sizing and scaling
    - Functional,transactional

- **Maintainance costs**
  - minimizing runtime administration effort for the gateway
    - self healing ?,(autonomous)agents
  - low infrastructural dependencies
    - cpu,memory,network,filesystem

- **Tools and Monitoring**
  - Limited access from outside
  - support for component/service based maintenance jobs
  - Availability,Licences,Possible upgrading
Development Requirements

- **Expandability**
  - Easy development of new components
  - Optile provides the framework
  - Providers build connectors
  - Extension of existing service components in production

- **Deployment**
  - Repository for components
  - Download and install
  - Avoid server downtimes
  - Runtime activation / deactivation of workflows based on existing service components

- **Version management**
  - Augment dependencies with versions
  - Ability to run different versions of a component within one runtime

- **Know How**
  - Low entry level for component developer
  - No home grown solution

Options

- **Basic architectural Metaphors**
  - **SOA**
    - SOAP, SCA, ESB, JBI...
  - **JEE (5)**
    - Servlet(SIP?), EJB3, JSF
  - **OSGi**
    - Bundle, Lifecycle, Services(Registry)
Service Orientierte Architektur Basics (SOA)

- pure architectural style
  - Not limited to a special technology
  - Nor deeply integrated
- loose coupling of services
  - Independence
  - Flexibility and Interoperability
  - Easy services integration
- Message based Communication
  - RPC supported
  - Scalability, Fault tolerance
- Processes and workflows
  - Combining services
  - (flow) engines

SOA III - Usual SOA suspects not applicable

- SOAP
  - Performance
- SCA
  - Infrastructure
  - Operations
- ESB
  - Infrastructure
  - Operations

- All these concepts are valid and have their respective usage
JEE (5)

- JCP Standard for enterprise architecture
  - [www.jcp.org](http://www.jcp.org)
- Component model
  - Based on special purpose containers
- Elaborated enterprise services API
  - Provided by containers
- Strong architectural compliance
- Application programming model
  - Main criteria: Scalability
    - and Fault tolerance
    - and Performance
  - ..

### JEE - Container

![Diagram of J2EE Container](image-url)
JavaEE - Services

JEE 5

- RMI-IOP
- HTTP/s
- JDBC
- JCA
- Java IDL
- Deployment JSR 88
- Management JSR 77
- JAAS /JACC
- Java Mail
- JMS
- JNDI
- JAXP /StAX
- SAAJ
- JAXP/R
- JAX-RPC/ JAX-WS
- JAXB
- JAF

Enterprise Application

JEE Deployment architecture

Browser

Applet Container

Web Container

EJB Container

App. Client Container

HTTP SSL

HTTP SSL

HTTP SSL

DB
OSGi - Open Services Gateway initiative

- no JavaEE standard
  - defined by OSGi Alliance [http://www.osgi.org](http://www.osgi.org)
    - Members e.g. IBM, Nokia, Oracle, Prosyst
- Componenten model
  - Structure build on „bundles“
  - Management and lifecycle based on Bundles
    - install/start/stop/uninstall/update
- Visibility based on packages
  - Definition of Im- and Exports
    - Entry in META-INF/MANIFEST
- Dependency management between Bundles
  - resolution at runtime
  - Enrichable with version information
- Slogan: SOA in a JVM“
  - Service Registry manages Services

OSGi I - Architecture Overview

OS

JVM

Application Modules

Classloader

Lifecycle

Service Registry

Services

Security
OSGi II - Service Platform

- Ausführungsplattform für OSGi Bundles
  - Lightweight Container
  - Containerdienste
  - Lifecycle für Bundles
  - Standalone oder Embedded einsetzbar

- OSS Implementierungen
  - Equinox (http://www.eclipse.org/equinox/)
  - Apache Felix (http://felix.apache.org/)
  - Knopflerfish (http://knopflerfish.org/)

Conclusion

- Combination of two architectural metaphors
  - Complementary usage
  - Weaknesses compensated by complementary approach

- SOA
  - Independence
  - Flexibility and Interoperability
  - Easy services integration
  - Fault tolerance
  - Scalability

- OSGi
  - Infrastructure
  - Deployment
  - Versionmanagement
  - Operations complexity
  - Skills availability
Optile Server technologies stack

- Java SE 6
- SpringSource dm Server™ 1.0.2 RELEASE
  - Eclipse Equinox OSGi
  - Spring Framework
  - Spring Dynamic Modules for OSGi™ Service Platforms
- JBoss jBPM
- JBoss Drools
- Spring Integration
- XForms 1.0

XForms

- W3C recommendation
  - Specification for Data Entry Forms
  - XML, XHTML document
  - Data Model
  - Presentation
  - Dynamic behaviour (validations) mit XPath
- Poor native browser support
- Server side rendering
  - Open source frameworks
XForms Model

```xml
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:xf="http://www.w3.org/2002/xforms"
     xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <head>
    <title>VISA</title>
    <xf:model>
      <xf:instance>
        <Transaction>
          <network>Visa</network>
          <Account>
            <expiryMonth/>
            <expiryYear/>
            <holderName/>
            <number/>
            <verificationCode/>
          </Account>
        </Transaction>
      </xf:instance>
    </xf:model>
  </head>
</html>
```

XForms View

```html
<html>
  <head>
    <title>VISA</title>
    <link rel="stylesheet" href="http://localhost/optile/css/forms01.css" type="text/css" />
    <body>
      <div id="logo">
        <a href="http://www.visa.com/"><img alt="Visa logo" title="Visa" src="logo_180x58.gif" /></a>
      </div>
      <xf:group>
        <xf:input ref="Account/number">
          <xf:label>Card Number</xf:label>
          <xf:alert>invalid card number</xf:alert>
        </xf:input>
        <xf:select1 ref="Account/expiryMonth">
          <xf:label>Card Expiration Month</xf:label>
          <xf:field list="Jan, Feb, Mar, Apr, May, June, July, Aug, Sep, Oct, Nov, Dec">
            <xf:value>01</xf:value>
          </xf:field>
        </xf:select1>
      </xf:group>
      <xf:submit submission="submit">
        <xf:label>Submit</xf:label>
      </xf:submit>
    </body>
</html>
```
XForms Controller

```
...<head>
 <xf:model>
  ...
  <xf:bind nodeset="Account/expiryMonth" type="xs:integer"
    constraint=". > 0 and 13 > "/>
  <xf:bind nodeset="Account/expiryYear" type="xs:qYear"
    constraint="number(substring(now(),1,4)) or
    number(substring(now(),1,4)) or
    ( . = number(substring(now(),1,4)) and
    expiryMonth >= number(substring(now(),6,2)))"/>
  <xf:bind nodeset="Account/holderName" type="xs:string"
    constraint="string-length(.) > 4"/>
  <xf:bind nodeset="Account/number" type="xs:string"
    constraint="string-length(.) = 16 and starts-with(.,'4') and is-card-number(.)"/>
  <xf:bind nodeset="Account/verificationCode" type="xs:string"
    constraint="string-length(.) = 4"/>
  <xf:submission id="submit" method="post" action="http://localhost/cgi-bin/echo.sh"/>
 </xf:model>
...```

Development setup

- Java SE 6
- Apache Maven
- Maven plugins:
  - org.apache.maven.plugins:maven-jar-plugin
  - org.apache.felix:maven-bundle-plugin
- SpringSource Tool Suite
Project setup: pom.xml

- **Multiple Module Project**
  - **parent pom**
    - Defines project modules
    - Defines modules versions
    - Defines used repositories
    - Defines common settings
    - Defines common properties
  - **module’s pom**
    - References to the parent pom and uses specified settings
    - Defines properties for OSGi imports
    - Specifies instructions for MANIFEST.MF generation
    - Specifies extra instructions for resource filtering

Project setup: spring configuration

- Spring configuration files are by default in /src/main/resources/META-INF/spring directory
- spring-context.xml defines bundle’s spring beans
- osgi-context.xml imports OSGi services and exports module’s beans as OSGi services
Follow-me-development demo

Project setup: module testing

- TestNG
- Mockito
- XStream

Define testng.xml with testing instructions
Define mock-osgi-context.xml with mock beans of all imported services
  - Stubs are possible
  - Iterations verification is possible

Use ClassPathXmlApplicationContext for Spring initialization

```java
ApplicationContext context = new ClassPathXmlApplicationContext(new String[] {
  "META-INF/spring/osgi-context.xml", "META-INF/spring/module-context.xml" });
```

Tests

```java
@BeforeClass
public void beforeClass() {
  target = (AbstractAdapter) context.getBean("providerAdaptersBean", AbstractAdapter.class);
}
```
Project setup: integration tests

- Why integration tests
  - MANIFEST.MF validation
  - Services import and export validation
  - Environment validation
    - Server’s environment
    - Bundle’s dependencies
  - Bundles correlation validation
  - Validation of system integrity

How to do integration tests

- Spring Dynamic Modules supports JUnit 3, all tests are lay out as OSGi bundle!
- Extend AbstractConfigurableBundleCreatorTests by each test class
- Overwrite
  - protected String[] getTestFrameworkBundleNames() to specify system bundles
  - protected String[] getTestBundleNames() to specify application bundles
    - formal <groupId><manifestFileName><version>
  - protected String getPlatformName() to specify target platform (Equinox is by default)
  - protected String[] getConfLocations() for Spring configuration files
- Import OSGi services in Spring configuration file
- Inject the target service what should be tested into the JUnit Test class
- Configure pom.xml with OSGi server as dependency, other dependencies could be specified as well
Best practices

- Use standard project layout
- Use conventions for classes names, configuration files and their contents
- Avoid cross references between bundles
- Think about services, not classes
  - embedded jar wrapping
- Use maven-bundle-plugin for MANIFEST.MF generation
- Use archetypes for project’s skeleton generation
- Use SpringSource Enterprise Bundle Repository for OSGi-adopted artifacts

Conclusion

- OSGi advantages / disadvantages
  - The count of bundles in the system
  - Bundles deployment order
  - Development view
- SpringSource OSGi products
  - Spring dm Server™
  - Spring Dynamic Modules for OSGi™ Service Platforms
- XForms
Mehr von OIO zum Thema...

- Schulung: Einführung in die OSGi Service Platform  
  http://www.oio.de/seminar/java/seminar osgi-schulung-equinox-training.htm

- Schulung: Java EE Design Patterns  
  http://www.oio.de/seminar/java/training-java-kurs-enterprise-patterns-schulung.htm

- Schulung: Java Enterprise Code Camp  
  http://www.oio.de/seminar/java/jee-patterns-workshop-java-ee-schulung.htm

- Beratung/Consulting Open Source Plattformen wie OSGi, Spring, RCP…  

- Veröffentlichung: Flexible Plattformen - Open Source Portale im Vergleich  

- Veröffentlichung: Vergleich von Java SSO Lösungen  
  http://www.oio.de/public/java/sso/single-sign-on-vergleich.htm

b.telligent optile GmbH
Uptown München
Georg-Brauchle-Ring 60
80992 München
thilo.keber@btelligent.com
alex.gerenago@btelligent.com