Infusion of OSGi Technology into a J2EE Application Server

Mikael Desertot, Didier Donsez
Contact: mikael.desertot@imag.fr
Outline

- Context
- J2EE application server overview
- Dynamism in J2EE application servers
- The ObjectWeb JOnAS case
- Motivation for introducing dynamism into J2EE
- New JOnAS OSGi-based architecture
- Improvements
- Conclusion
Context

- IMAG LSR Laboratory, Adele team, Grenoble
  - JOnAS and Oscar contributor
- Bull, an international French company
  - JOnAS contributor and leader
- Members of ObjectWeb
  - 1rst International consortium for Open Source Middleware
  - Co-founded 2002 by Bull, France Telecom R&D and INRIA;
  - Non-profit, open, business neutral
  - Industry-grade projects
- 60 corporate members incl. 10 research labs
- 1700 individual members from 80 countries
- 130,000 visitors/months, 40% EU, 40% N.A
- 100 projects, 470 committers, 5,700 contributors
J2EE Application Server

- A Java-based set of specifications targeting business applications
  - A standard for developing component-based multi-tier enterprise applications
  - Facilitate and standardize the development, deployment, and assembling of application components

Clients → Presentation Layer → Business Logic → Enterprise Information System

Clients → Servlets → JSPs

Business Logic → Session Bean → Entity Bean → Message Driven Bean

Enterprise Information System → DB
What Is And Is Not Addressed

• Two levels inside the server
  – Application level
  – Technical (non-functional) services level
    • Security, Transaction, Persistence, Messaging, …
• Application level is dynamic
  – Hot deployment (install, update and uninstall) of J2EE applications at runtime
  – JSR 88 : J2EE Application Deployment
• Nothing is specified for technical services
Different Ways Of Introducing Dynamism

• The way dynamism is treated for technical services is implementation dependent

• How is it treated in open sources implementations
  – JBoss case
    • Based on JMX
    • Dynamic classloading with JMX classloader
  – Geronimo case
    • Applications and services are components called GBeans
    • Uses Inversion of Control (IoC) design pattern
Motivation: JOnAS On-Demand

- Dynamic server configuration and reconfiguration
  - Start only needed services
  - Start, stop, reconfigure and restart services during runtime

- Incremental and on-demand services delivery (service installation and start up)
  - Deploy new services during runtime
  - Activate them

- Ease services deployment and update
How To

• Idea :
  – Use OSGi, a standard dynamic platform
    • Package services
    • Deploy services
    • Manage service lifecycle
    • Bind services

  – Main interests for OSGi
    • Dynamic Service Oriented Architecture (DSOA)
    • Dynamic class loading
Towards A New JOnAS Architecture

- Current state

JOnAS + Technical J2EE Services

JSR88
Dynamic Deployment
JSR88

Applications

- Application 1
- Application 2
Towards A New JOnAS Architecture V1

- Flat class loader architecture
- Metadata generation at deployment
Towards A New JOnAS Architecture V2

JOnAS On-Demand

Appli. Service

EJB Service

Web Service

Database

Transact.

Mail.

etc

Tomcat

Applications

Application 1 + Metadata

Application 2 + Metadata

JOnAS Core

OSGi framework / Oscar

JOnAS Core

JOnAS On-Demand

Appli. Service

EJB Service

Web Service

Database

Transact.

Mail.

etc

Tomcat

Applications

Application 1 + Metadata

Application 2 + Metadata

JOnAS On-Demand

Appli. Service

EJB Service

Web Service

Database

Transact.

Mail.

etc

Tomcat

Applications

Application 1 + Metadata

Application 2 + Metadata

JOnAS Core

OSGi framework / Oscar
Experimentation

- Architecture implemented with JOnAS and Oscar
- Most of the services already packaged in bundles
- Two visions
  - Deploying JOnAS on an independent OSGi gateway, collocated with other services
  - Embedding OSGi in the JOnAS core
- Works for a single instance of JOnAS On-Demand
- Cluster of multiple instances in progress
  - Open issues: RMI, ...
Particular Improvements Brought By OSGi

• A dynamic service oriented architecture for JOnAS

• Class loading issues
  – Class sharing policy allows an application to use a different class version than the one provided in JOnAS

• Deployment
  – A way of packaging the server and its services
  – Downloading of services delegated to the OSGi framework
  – OSGi deployment tools : OBR, “Service on Demand”

• Specific needs covered by OSGi R4
  – Multiple class version loaded at the same time
  – Package constraints
  – …
Conclusion

• OSGi can be very interesting on the server side:
  – For server configuration/reconfiguration
  – For loading service modules and managing them (lifecycle)

• It can also:
  – Ease contributions to an open source server
    • Services can be developed apart from the server core
  – Provides online incremental server update