JOnAS: an Open Source Unbreakable Java™ EE Server Platform

F. Exertier, F. Fornaciari
Agenda

> Introduction
> A flexible and modular platform
> A distributed and scalable architecture
> Self healing capabilities
> Demonstration
> Summary
Evolution of application server usage

> Mission critical applications

> Ambient computing
  • Small computers, resource-constrained electronic devices

> Examples:
  • Health care, mobile, home, public services
  • M2M, Home Automation and RFID-based applications
  • Telco and Internet mixed applications: convergence between IMS, Web and Media service technologies
Evolution of application server scope

Application Servers

Enterprise Information System

Internet

M2M gateways

Mobile Devices

Probes, sensors

Monitoring, self-management tools
New requirements

> Dynamic service-oriented platform which is able to adapt itself
> Embedded system domain
> Bridge the gap between the different parts of new generation applications: edge, premise, server
> Inter domain interoperability
> Support adaptable distributed applications
> Service continuity
Unbreakable Java EE Server Platform

> Non-stop self-healing application server for mission critical applications, suitable for ambient computing

> Fully OSGi-based distributed and scalable architecture
  • Result from advanced research in component and services models
  • OSGi provides key features to application servers
    • Dynamicity, flexibility, modularity, plug ability, reduced footprint

> Full service continuity thanks to
  • Automated reconfiguration and management
  • Smooth and transparent migration solutions
  • High availability clustering solutions
Solution: Java EE Server Platform

> JOnAS application server based on OSGi
  • Highly dynamic and adaptable platform
  • Bridge toward OSGi world (M2M, Home Automation and RFID-based applications)

> Clustering solution
  • High Availability, failover

> JASMINe Management Platform
  • Self-healing features
  • Migration solutions
JOnAS: Java EE Server Platform

> Java EE 5 certified
> Provides technical services
  > Webcontainer with Tomcat/Jetty
  > EJB3/JPA Persistence: EasyBeans (EclipseLink, Hibernate, OpenJPA)
  > Transaction with JOTM
  > Clustering with CMI
  > Security
  > WebService with CXF/Axis2
  > Messaging with JORAM
> Administration: web console and JASMINe project
JOnAS: Open Source

> Developed as open source (LGPL) in OW2

• OW2: International consortium dedicated to open source middleware

• Main contributors: Bull, FranceTelecom, Peking University, INRIA, UJF, UNIFOR, SERLI

• Related OW2 projects: EasyBeans, JASMINe, JORAM, JOTM, CMI, Shelbie

• Other open source communities: Apache, CodeHaus, JBoss…
OSGi native architecture
Dynamic platform

> Modularity / Maintainability
> Better lifecycle management
> Services composed of OSGi bundles
> Incremental service delivery
> Dynamic re-configuration
> Automatic adaptation to runtime constraints
Agenda

> Introduction
> A flexible and modular platform
> A distributed and scalable architecture
> Self healing capabilities
> Demonstration
> Summary
OSGi native architecture
Flexible platform (1/2)

> Communications handled by the OSGi™ service layer
  • Loose-coupling between modules
  • Not dependent on a specific implementation

> Allows dynamic OSGi™ service replacement
  • Update Java EE technical services to the latest version
OSGi native architecture
Flexible platform (2/2)

> On Demand Services

![Diagram of JOnAS](image)

Java EE technical services

Applications

USB Thermometer
OSGi native architecture
Advanced deployment mechanisms

> Deployment Plan:
  - XML file describing resources to be deployed in a given order
  - Resources are stored in repositories. The server maintains a configurable list of repositories
  - Resources types : URL, Maven, OBR
  - « reloadable » option for automatic reload of modified resources
  - Clustering deployment eased thanks to shared deployment plans
OSGi native architecture
Java EE -> OSGi

> Java EE applications can communicate with the OSGi™ world transparently

> EJB™ 3.0 can use the BundleContext
  • Listen to OSGi™ events (framework, bundle, service)
  • Access to OSGi™ services

@OSGiResource
BundleContext bundleContext = null;

@OSGiResource
PrintService printService = null;
OSGi native architecture
OSGi -> Java EE

> Provides Java EE functionalities to pure OSGi applications
  • Expose Session Beans as OSGi™ services
    • Stateless Bean offer entry points to the Java EE™ world
  • Advantage:
    • Benefit of persistence, transaction, …

```java
BundleContext bundleContext = …
ServiceReference serviceRef = bundleContext.getServiceReference(StatelessLocal.class.getName());
StatelessLocal stateless = bundleContext.getService(serviceRef);
```
OSGi native architecture

Summary

- Adaptation to user’s needs
- Ease of exploitation
- Modularity and lightness: facilitates maintenance
- Optimized resources consumption, system footprint
- Dynamic Adaptation, Configuration
Agenda

- Introduction
- A flexible and modular platform
- A distributed and scalable architecture
- Self healing capabilities
- Demonstration
- Summary
A distributed and scalable architecture
Clustering

> End to End solution
  • Web and EJB / Load Balancing and HA
> Clustering for EJB2 and EJB3, performance oriented
> Dynamic re-configuration of the load balancing logic (via the console)
> Dynamic and transparent update of the API
> Simplified deployment
Clustering

> Load-balancing and fail-over
  • Web: with mod_jk/mod_proxy_balancer
  • EJB2/EJB3: with CMI v2

> Replication
  • Web session replication with Tomcat
  • EJB2/EJB3 replication with HA Service
    • JGroups/Terracotta/P2P
JOnAS Clustering
Management of Load Balancing policy

Master node

Cluster logic V2

Update Cluster logic

Ctrl flow

Cluster logic V1

JOnAS Administration

Reset

Domain (sampleCluster2Domain)

Server JOnAS (node1)

Properties JMX, CMIS, CMII, cluster

Global
Name org.ow2.easybeans.examples.democluster.StatelessBean_Ow2
Cluster name test_cluster
Interface org.ow2.easybeans.examples.democluster.StatelessBean_REMOTE
Deployer servers
http://127.0.0.1:24000

Load balancing info
Policy class FirstAvailablePolicy
Policy strategy RoundRobinPolicy
Pool Info
Min pool size 1
Max pool size 5

org.ow2.easybeans.examples.democluster Stateless
Clustering
Summary

> Ease of use
  • Zeroconf at the client side
  • Control from the server side

> Flexibility

> Dynamic configuration
Agenda

> Introduction
> A flexible and modular platform
> A distributed and scalable architecture
> Self healing capabilities
> Demonstration
> Summary
JASMINe

- Graphical tool for configuration, deployment and supervision of middleware
- Help for error detection
- Performance monitoring
- Autonomous behavior
JASMINE

> JASMINE Design
  - Graphic tool for building a middleware configuration

> JASMINE Deploy
  - Framework for deploying a middleware configuration

> JASMINE Monitoring
  - Tools for performance tracking and error detection

> JASMINE Self-management
  - Control loop for Self-optimization & Self-healing

> JASMINE Repository
  - Static and dynamic data regarding managed system
JASMINe Design Principles

> Eclipse EMF/GMF for configurations design (RCP GUI)
> Jade/OSGi for deploying configurations on the infrastructure

1. Describe the middleware configuration
2. Deploy the middleware configuration
JASMINe Design
JASMINe Monitoring

- Probes (JMX, OS)
- Distributed mediation layer
- Error detection with rules engine
- EoS console (flex)

Benefits
- Governance
- Reactivity
- Reduced admin costs
JASMINE Monitoring

> Error Detection
  • Based on Drools rules engine
  • Rules Examples :
    • Error logs or counter aggregation
    • Cpu overload detection over a significant period (ignore peak load)
    • Memory saturation is close
    • Datasource bottleneck
    • ...
  • Actions : alarms, mail, … extensible
JASMINEe Monitoring

> Quick visualisation environment
JASMINE Monitoring

- Visualisation environment resulting from an advanced configuration of the tool
JASMINe Self-management

- Self-healing
- Self-optimization
JASMINe
Advanced Management: Versioning (1/2)

> Context
  • Mission Critical Applications
  • Version Upgrade
    • Without service interruption
    • Without loosing user sessions
    • Without over sizing the system

> Principle
  • Several versions of a same application co-running in a same JOnAS instance
  • Configurable request routing policy toward versions

> Available for WebApp Tomcat, EJB2/EJB3
JASMINe
Advanced Management: Versioning (2/2)

> A client uses the same version until session expiration
> New clients access to the version defined by the deployment policy
JASMINe
Summary

> Improved Quality of Service
> Automatic Deployment
> Better reactivity
> Decreased Risks
> Self-behaviors: self repair, self optimization, …
JOnAS Server Adapters
Eclipse/NetBeans

> Hot deployment
  • Update only modified parts of Java EE applications
  • Keep active sessions

> EoD
Agenda

- Introduction
- A flexible and modular platform
- A distributed and scalable architecture
- Self healing capabilities
- Demonstration
- Summary
Agenda

> Introduction
> A flexible and modular platform
> A distributed and scalable architecture
> Self healing capabilities
> Demonstration
> Summary
Summary

> Non-stop self-healing application server for mission critical applications, suitable for ambient computing

> Fully OSGi-based distributed and scalable architecture
  • Dynamicity, flexibility, modularity, plug ability, reduced footprint

> Full service continuity thanks to
  • Automated reconfiguration and management
  • Smooth and transparent migration solutions
  • High availability clustering solutions
Enterprise Class

- Java EE Certified (J2EE 1.4, Java EE 5)
- Incremental solution, adaptable to functional environment
- Exploitation features: administration, cluster
- Plugins: Eclipse WTP, Cargo
- Code maturity (project started in 1998)
- JOnAS 4 deployed in many sectors
  - Public, aerospace, automotive, bank, industry, telecommunication, defence, health, ministries…
Thank You

F. Exertier, F. Fornaciari
Francois.Exertier@bull.net
Francois.Fornaciari@bull.net