JBoss Certified Technology Partner

Israel JBoss User Group
Session 08 / 16.4.2008

JBoss 5: IoC for the Deployables

By : Zvika Markfeld, Tikal Knowledge

Agenda

- Introduction
- New Features
- JBoss Microcontainer
- JBoss and OSGi
- Business Analysis

Introduction

- What's happening in the JEE world recently
  - POJO, POJO, POJO
  - Spring, Hibernate, JSF, Web Beans, OSGi
  - Server-side changing...
- JBoss want to win back the runtime stack

JBoss 5 New Features

- ***Microcontainer***
- ***JBoss AOP***
- Virtual Deployment Framework
- Unified Invokers
- Classloading
- Messaging
- Transactions
- Clustering
- JSF1.2(CDDL)
- Web Stack

JBoss MC

- JBoss Microcontainer 2.0: The new kernel
  - » Refactoring of the JMX Microkernel
- Supports direct ***POJO deployment*** and standalone use outside the JBoss application server
- Bootstrapped by ***Profile service***
  - Responsible for managing the POJOs now binding services together
- Running on top of IoC container:
  - » JMXKernel & MainDeployer, rebranded as POJO
- jboss-service.xml -> ***jboss-beans.xml***
- conf/jboss-service.xml -> ***conf/bootstrap-beans.xml***

JBoss MC

- Emphasizes the concept of a ***state machine***
  - It manages fine grained state transition for beans
  - Bean States: *Not Installed, Described, Instantiated, Configured, Create, Start, Installed and Error*
  - Will support hot-redeployment of any bean
  - Not a general purpose IoC container!
  - » Missing many useful features found in Spring/Pico/Nuts
    - Auto-wiring, inner bean/local bean, prototype bean, module, abstract bean, ad-hoc bean combination
- Provides a set of APIs that can be used directly to manage beans without having to use xml.

JBoss MC

- Lightweight, Unit Testable, Mavenized
- POJO based, Service Oriented
- Inerface/Class-based
  - *"A Java Bean is a reusable software component that can be manipulated visually in a builder tool."*

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:jboss:bean-deployer:2.0 bean-deployer_2_0.xsd"
        xmlns="urn:jboss:bean-deployer:2.0">

  <bean name="HRService" class="org.jboss.example.service.HRManager">
      <property name="salaryStrategy"><inject bean="AgeBasedSalary"/></property>
      <property name="hiringFreeze">false</property>
  </bean>
...
</deployment>
```

JBoss MC

- PropertyEditor Support
- No prototypes
  - *"trivial logic which is very unlikely to break"*
- Service Jar:
  - ***META-INF/jboss-beans.xml:***
    service descriptor
  - ***META-INF/maven/.../pom.xml:***
    generated by Maven, used for dependency resolution
  - class files & resources

Demo

- Coding humanResourcesService
- Running the client
- Observing client bootstrap

Microcontainer Architecture

- Services are POJOs, behavior is added through AOP
- Effectively reproducing all JMX microkernel features
- Improved DI and classloading
- Composed of:
  - ***Kernel:***
    Provides the bus and registry, and an event manager
  - ***Container:***
    Wrapper for POJOs, AOP jointpoints and reflection on the actual service implementation POJO
  - **Dependency:**
    Basically an abstract state machine that manages service dependencies.
- Some optional packages on top of the core: OSGI integration, Guice integration.

JBoss MC Client Invocation Styles

- Direct, typed, cached

```
bootstrap = new EmbeddedBootstrap();
bootstrap.run();
bootstrap.deploy(url);
kernel = bootstrap.getKernel();
controller = kernel.getController()
 context = controller.getInstalledContext(HRSERVICE);
manager = (HRManager) context.getTarget();
manager.addEmployee(newEmployee);
```

JBoss MC Client Invocation Styles

- JMX-style, untyped, dynamic

```
bootstrap = new EmbeddedBootstrap();
bootstrap.run();
bootstrap.deploy(url);
kernel = bootstrap.getKernel();
bus = kernel.getBus();
bus.invoke(
    "HRService",
    "addEmployee",
    new Object[] {emp},
    new String[] {“org.jboss.example.service.Employee”});
```

Demo

- Client invocation styles

Classloading Customization

- What if a service should be loaded from a jar not included in the launcher classpath?

```
<bean name="URL" class="java.net.URL">
    <constructor>
       <parameter>file:/Users/zvika/.../service.jar</parameter>
    </constructor>
 </bean>

 <bean name="customCL" class="java.net.URLClassLoader">
    <constructor> ...
             <inject bean="URL"/>
     ... </constructor>
 </bean>

 <bean name="HRService" class="org.jboss.example.service.HRManager">
     <classloader><inject bean="customCL"/></classloader>
 </bean>
```

Demo

- Custom classloading service jar

JBoss AOP 2.0

- Integrated with the microcontainer
- Custom syntax & implementation, using javassist
  - Wheels are not meant to be shared!
- Add behavior to a POJO using AOP
  - POJOs won't be deployed before the aspect is available
  - Aspects are not deployed if the POJOs they depend on are not deployed
- Aspects can be bound to POJO lifecycle
  - e.g an aspect that binds a proxy into JNDI when the POJO enters the deployed state.
- It also has some new plain AOP features: Before, After, Throwing, Finally flows for interception.

Demo

- Adding a simple Aspect

JBoss MC – Lifecycle Callbacks

- NOT_INSTALLED – deployment descriptor parsed
- DESCRIBED – aop dependencies added to the bean
- INSTANTIATED – an instance has been created
- CONFIGURED – properties have been injected
- CREATE – the create method, if defined, was called
- START – the start method, if defined, was called
- INSTALLED – custom install actions executed, bean is ready to access

JBoss MC – Lifecycle Callbacks

```
<aop:lifecycle-install
   xmlns:aop="urn:jboss:aop-beans:1.0"
   name="InstallAdvice"
   class="org.jboss.test.microcontainer.support.LifecycleCallback"
   classes="@org.jboss.test.microcontainer.support.Install">
</aop:lifecycle-install>
```

- Read:
  LifecycleCallback should be applied to any beans annotated with *@Install* before and after the *INSTALLED* state

JBoss MC – Additional Features

- Annotation Support
- Spring <beans> Support
- Other xml / custom formats
- Factories
- Bean Aliases
- more...

Profile Service

- Management system for POJOs
- Replaces the JMX based administration
- Centralized maintenance as profiles, e.g. all, minimal, default, ...
- Persistence of changes made to a profile across server restarts
- Propagation of profile changes across a cluster
- Profile Versioning
- Loaded from *bootstrap-beans.xml*

JBoss and OSGi

- Goal #1: OSGi based classloader
  - First for JBoss runtime, later for application developers via OEEG(OSGi Enterprise Expert Group)
  - Restricting class visibility
  - Better control over the exposure of implementation details
  - Integrate OSGi Bundle Repository (OBR)
  - Fits nicely with ProfileService, VFS
- Goal #2: Full OSGi core spec v4.1 implementation
  - With added features: AOP, JMX support, fine grain DI, scoped metadata, generic deployers
- Current State: Work In Progress

Business Analysis

Tale Of 3 Frogs...

Business Analysis

- So runtime stack is important after all...
  - » Will the effort pay off? Will JBoss replace Spring?
- What happened to (the wonderful friendship with) Guice?
  - » Not much
- What does JBoss 5 prove?
  - » IoC paradigm still has a way ahead...
- Pitchfork: Eat their dust!
  - » or not?

Q & A

Thank You!