Never Write A main() Again!

Harnessing The Power Of
The JBoss MicroContainer

Dimitris Andreadis
Core Developer
JBoss Inc.

March 8, 2005

Agenda

- About the MicroContainer
  - ✓ Motivation
  - ✓ Micro* History
  - ✓ Hijacking JMX
- Extending JBoss
  - ✓ JMX Primer
  - ✓ Service Basics
  - ✓ XMBeans
  - ✓ Service Deployment
  - ✓ Design Guidelines
- The POJO Container

Motivation

"I'm building a server application, but it's not servlets/EJBs"

```
// Here we go again!
public class MySuperServer
{
   public static final void main(String[] args)
   {
      System.out.println("**** Starting ****");

      [ … lots of code omitted … ]

   }
}
```

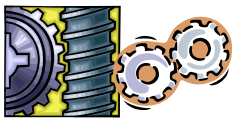

Things to consider

- Componentization
- Packaging
- Classloading
- Configuration
- Naming
- Lifecycle
- Dependencies
- Events
- Management
- Logging
- …

A lot of infrastructure is needed before actually getting to do anything useful

Motivation 2

"I'm already using JBoss (I've seen the light) and I want to add my own cool extensions"



Meet the JBoss MicroContainer

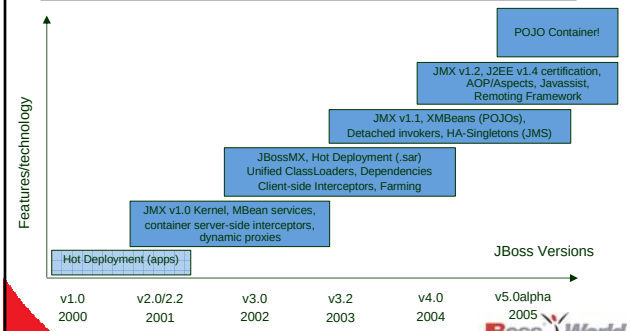
- Has been powering JBoss since v2.x (2001)
- Has been "commoditized" since v3.x (2002)
- Going into orbit with v5.x (2005)

What is the JBoss MicroContainer

- ✓ A lightweight component framework that allows the wiring of a set of Services to create any Server.
- ✓ A Service can be anything, from a complex JMS provider to an EJB container, to a simple user-provided POJO.
- ✓ Services are de-coupled, invocations are routed through an internal bus so runtime recycling is possible.
- ✓ During bootstrap the MicroContainer provides just enough infrastructure (i.e. Services) to start loading other Services.

(a.k.a. The *SuperServer*)

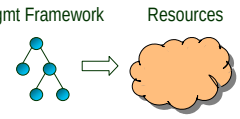
From MicroKernel to POJO Container



Could Management ever be useful?

Management Frameworks/Standards exhibit some interesting properties:

- ✓ A Metadata Model
- ✓ Lifecycle/Configuration
- ✓ Naming/Lookup
- ✓ Invocations/Remoting
- ✓ Notifications

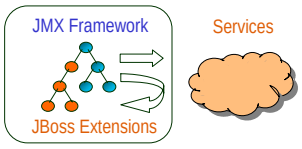


Those are essential ingredients of a component framework!

see
http://www.omg.org/cgi-bin/doc?telecom/00-09-06

Hijacking JMX

- Use JMX as the basis of a mini component framework (reusability, documentation)
- Add the missing pieces to complement JMX as a component framework
- Implement or wrap and access everything as MBeans (manageability, de-coupling)
- Let developers add their own extensions to the server! (do it the OpenSource way)



JBoss MicroContainer!

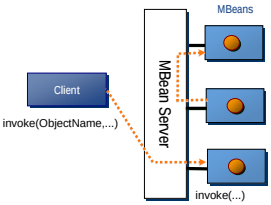
Essential Add-ons

- MicroContainer Extensions
  - ✓ UnifiedClassLoader
  - ✓ Lifecycle control
  - ✓ XML Configuration
  - ✓ Service dependencies
  - ✓ Extensible deployment model
  - ✓ XMBeans
- Related Services
  - ✓ AOP/aspect framework
  - ✓ Clustering/caching framework
  - ✓ Remoting framework
  - ✓ jmx-console/web-console
  - ✓ Protocol adapters
  - ✓ ...

Extending JB o s s

JMX Primer

- MBeanServer
  - ✓ Is always a local entity
  - ✓ Acts as an MBean factory
  - ✓ Acts as an MBean registry
  - ✓ Provides de-typed access to MBeans identified by ObjectName
- MBeans (Managed Beans)
  - ✓ Have CTORs
  - ✓ Have Attributes
  - ✓ Support Operations
  - ✓ Emit Notifications
  - ✓ Expose the above as Metadata



MBeanServer API in a nutshell

createMBean(...)

registerMBean(...)
unregisterMBean(...)

getAttribute(...)
setAttributes(...)

addNotificationListener(...)
removeNotificationListener(...)

public Object invoke(ObjectName name, String operation,
Object params[], String signature[]) throws ... ;
getMBeanInfo(...)

ObjectName Format
<domain>:{<key>=<value>}[,<key>=<value>]
Example
jboss.system:service=Logging,type=Log4jService

Example

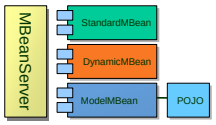
```
// get the MBeanServer
MBeanServer server  =
        (MBeanServer)MBeanServerFactory.findMBeanServer(null).get(0);
// define the target MBean
ObjectName target = new ObjectName("jboss.system:service=MainDeployer");
// set and get an attribute
server.setAttribute(target, new Attribute("CopyFiles", new Boolean(true)));
String state = (String)server.getAttribute(target, "StateString");
// invoke an operation
Boolean result = (Boolean)server.invoke(
        target,
        "isDeployed",
        new Object[] { new URL("./deploy/myapp.ear") },
        new String[] { "java.net.URL" });
```


Example2 (less ugly)

```
// get the MBeanServer
MBeanServer server  =
        (MBeanServer)MBeanServerFactory.findMBeanServer(null).get(0);
// define the target MBean
ObjectName target = new ObjectName("jboss.system:service=MainDeployer");
// Get a type-safe dynamic proxy
MainDeployerMBean deployer =
   (MainDeployerMBean)MBeanServerInvocationHandler.newProxyInstance(
      server, target, MainDeployerMBean.class, false);
// set and get an attribute
deployer.setCopyFiles(true);
String state = deployer.getStateString();
// invoke an operation
boolean result = deployer.isDeployed(new URL("./deploy/myapp.ear"));
```

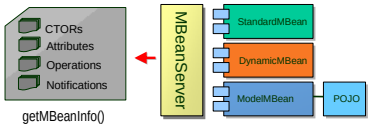

The Basics of Services

- ✓ Services can be implemented as
  - Standard MBeans
  - Dynamic MBeans
  - POJOs wrapped by Model MBeans (XMBeans)
- ✓ Independent of the implementation all MBeans look the same through the MBeanServer
- ✓ Each Mbean provides an Interface to MBeanServer clients or other MBeans



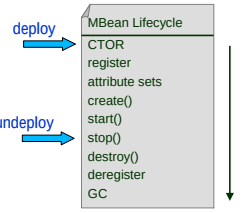
Metadata is the Key

- ✓ Services expose their interface as Metadata
  - Constructors/Attributes/Operations/Notifications
- ✓ Metadata is specified
  - By means of reflection (Standard MBeans)
  - Through an API (Dynamic MBeans)
  - Externally (Model MBeans)



Service Lifecycle

- Services may optionally expose lifecycle operations
  - ✓ create(), start(), stop(), destroy()
  - ✓ no need to extend JBoss classes
- Services may optionally extend JBoss support classes
  - ✓ ServiceMBeanSupport
  - ✓ ServiceDynamicMBeanSupport
  - ✓ ListenerServiceMBeanSupport



Typical (old-style) JBoss Service

```
public Class MyService
  extends ServiceMBeanSupport
  implements MyServiceMBean
{
  // Private
  private String anAttribute;

  // One or more public Constructors
  public MyService() { ... }

  // Attributes
  public void setAnAttribute(String value) { ... }
  public String getAnAttribute()  { ... }
  ...
  // Optional Lifecycle hooks
  public void createService() throws Exception  { ... }
  public void startService() throws Exception  { ... }
  public void stopService() { ... }
  public void destroyService() { ... }

  // Operations
  public void doStuff(...) { ... }
  ...
}
```

```
interface MyServiceMBean
  extends ServiceMBean
{
  // exposed JavaBean style attributes
  void setAnAttribute(String value);
  String getAnAttribute();
  ...
  // Exposed operations
  void doStuff(Object input);
  ...
}
```


ServiceMBeanSupport Base class

- Protected Data
  - ✓ log
  - ✓ server
  - ✓ serviceName
- Added MBean Attributes
  - ✓ String Name
  - ✓ int State
  - ✓ String StateString
- NotificationBroadcaster impl.
  - ✓ sendNotification(Notification)
  - ✓ getNextNotificationSequenceNumber()
- Hooks up with ServiceController for enforcing dependencies
  - ✓ createService(), startService()
  - ✓ stopService(), destroyService()

ListenerServiceMBeanSupport Base class

- Added MBean Attributes
  - ✓ org.w3c.dom.Element SubscriptionList

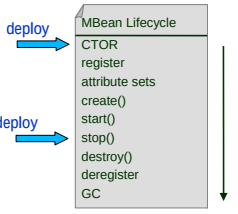
Example

```
<subscription-list>
  <mbean name="jboss.monitor:*"/>
  <mbean name="JMImplementation:type=MBeanServerDelegate">
    <notification type="JMX.mbean.registered"/>
    [ ... ]
  </mbean>
  [ ... ]
</subscription-list>
```

- NotificationListener
  - ✓ Configured declaratively through SubscriptionList
  - ✓ Automates JMX Notification subscription mgmt., use:
    - subscribe(boolean dynamicSubscriptions)
    - unsubscribe()
  - ✓ Override handleNotification2(Notification, ...)

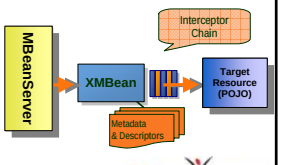
Some interesting points

- Can I generate the MBean interface?
- What goes into my CTOR?
- Why not just start() & stop()?
- Can I stop and restart a service?
- Can I modify attributes while a service is running?
- How redeploy works?



MBeans

- JBoss XML-based implementation of ModelMBean
  - ✓ Wrap as a service any POJO
  - ✓ Combine the simplicity of Standard MBeans and XML with the power of Dynamic MBeans
- Provides for
  - ✓ Rich metadata descriptions
  - ✓ Attribute caching
  - ✓ Attribute persistence
  - ✓ Attribute change notifications
  - ✓ Ability to plug-in custom interceptors



Example JBoss POJO Service

```
public Class MyService
{
  / / One or more public Constructors
  Public MyService() { ... }

  / / Any number of (optionally JavaBean style) attributes
  public void setSomeStringAttribute(String value) { ... }
  public String getSomeStringAttribute() { ... }

  / / Optional Lifecycle operations
  public void create() throws Exception { ... }
  public void start() throws Exception { ... }
  public void stop() { ... }
  public void destroy() { ... }

  / / Optional operations
  public void doStuff(...) { ... }
}
```


Example (complex) XMBean descriptor

```
<mbean>
  <description>XMBean Attribute Persistence Service</description>
  <descriptors>
    <persistence persistPolicy="OnUpdate" persistLocation="${jboss.server.data.dir}/xmbean-attrs"
                 persistName="AttributePersistenceService.ser" />
    <persistence-manager value="org.jboss.mx.persistence.ObjectStreamPersistenceManager" />
    <injection id="MBeanServerType" setMethod="setMBeanServer" />
    <injection id="ObjectNameType" setMethod="setObjectName" />
  </descriptors>
  <class>org.jboss.system.pm.AttributePersistenceService</class>
  ...
  <attribute access="read-write" getMethod="getVersionTag" setMethod="setVersionTag">
    <description>The version tag to use for stored/loaded Attribute data</description>
    <name>VersionTag</name>
    <type>java.lang.String</type>
    <descriptors>
      <descriptor name="persistPolicy" value="OnUpdate" />
    </descriptors>
  </attribute>
...
  <operation>
    <description>Delegated to the active APM it returns a single string
                 with the ids of all persisted attribute images</description>
    <name>apmListAllAsString</name>
    <return-type>java.lang.String</return-type>
  </operation>
...
```

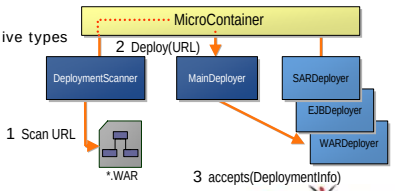
If you can read this, you don't need glasses...

Configuring and Deploying Services

- Services are loaded/configured using XML by specifying
  - ✓ Classloading scoping
  - ✓ Codebase specification
  - ✓ Attribute overrides (property injection)
  - ✓ Dependencies (create/create, start/start)
    - By name
    - By type (IoC style)
- Services can be hot-deployed
  - ✓ Using stand-alone service descriptors (-service.xml)
  - ✓ Or, packaged with classes into Service Archives (.sar)

Deployment Architecture Overview

- DeploymentScanner
  - ✓ Scans local filesystem or remote web server for modules
- MainDeployer
  - ✓ Orchestrates deployment, manages SubDeployers
- SubDeployers
  - ✓ Handle specific archive types e.g. jar, war, ear, har, aop, bsh, etc.
- Support for
  - ✓ remote loading
  - ✓ "Russian Doll" packaging
  - ✓ Unpacked archives



Example Service Descriptor (1)

```
<mbean code="org.jboss.naming.NamingService"
       name="jboss:service=Naming"
    xmbean-dd="resource:xmdesc/NamingService-xmbean.xml">

    <attribute name="CallByValue">false</attribute>
    <attribute name="Port">1099</attribute>
    <attribute name="BindAddress">${jboss.bind.address}</attribute>
    <attribute name="RmiPort">1098</attribute>
    <attribute name="RmiBindAddress">${jboss.bind.address}</attribute>

    <depends optional-attribute-name="LookupPool"
             proxy-type="org.jboss.util.threadpool.BasicThreadPoolMBean">
        jboss.system:service=ThreadPool
    </depends>
</mbean>
```


Example Service Descriptor (2)

```
<mbean code="org.jboss.invocation.jrmp.server.JRMPInvoker"
       name="jboss:service=invoker,wantsClientAuth=true">

  <attribute name="RMIObjectPort">0</attribute>
  <attribute name="RMIClientSocketFactory">org.jboss.ssl.RMISSLCSF</attribute>
  <attribute name="RMIServerSocketFactoryBean"
             attributeClass="org.jboss.security.ssl.RMISSLServerSocketFactory"
             serialDataType="javaBean">
    <property name="bindAddress">${jboss.bind.address}</property>
    <property name="securityDomain">java:/jaas/rmi-ssl</property>
    <property name="wantsClientAuth">true</property>
    <property name="needsClientAuth">true</property>
  </attribute>
</mbean>
```

MBean Services in JBoss

- Core/Boot System (jboss.system)
- UnifiedClassLoaders (jmx.loading)
- Service Providers (jca, jms, jndi, security, etc.)
- Subsystems (aop, ejb, web, ws, iiop, etc.)
- Deployers (.sar, .jar, .ear, .war, .har, etc.)
- Invokers (http, socket, iiop, etc.)
- ...
- Management (JSR-77, jboss.managament.local)
- Monitoring (standard JMX monitors)

MBean Service Design Guidelines

- When to MBean?
  - ✓ When integrating code or external systems
  - ✓ When dynamic loading is desired
  - ✓ When precise lifecycle control is needed
  - ✓ When a management aspect is exposed
- Traps to avoid?
  - ✓ Watch out for fine grained components
  - ✓ Watch out for parameter class leakage
  - ✓ Watch out for concurrency issues

<Your-Service-Here>

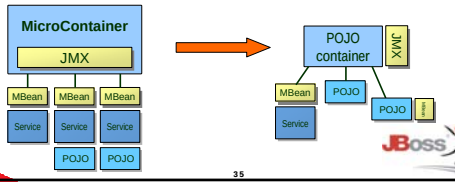
- Protocol listeners (http, socket, etc.)
- Remote object wrapping (RMI, CORBA, etc.)
- Lightweight adapters to external systems
- Lightweight event processing
- Dynamic configuration settings
- JNDI object registrations
- Deployer extensions
- Scheduling

Limitations

- JMX is Ok for most server applications, but it's not available in applets or J2ME
- We want to run any JBoss service in other containers where we don't control the MBeanServer

The Road Ahead – The POJO container

- A standalone lightweight product
  - ✓ Run JBoss services without the Application Server
  - ✓ Use JBoss services in somebody else's Application Server
- It fits JBoss's POJO strategy
  - ✓ Code in plain java (no need to import framework classes)
  - ✓ JBoss manages your POJO, adding middleware features
  - ✓ JMX is not disappearing, POJOs can be "JMX manageable"



Recap

- JBoss MicroContainer is a well tested and proven technology
- You can use the MicroContainer as a framework for building different types of servers
- You can easily extend JBoss with your own POJO or MBean services
- The new POJO Container builds on past experience to target the most diverse runtime environments

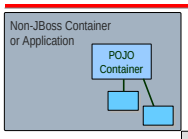
Resources

- Forums
  - ✓ Management, JMX/Jboss
  - ✓ Design of JMX on Jboss
  - ✓ Design of Management Features on Jboss
  - ✓ Design the new POJO MicroContainer
  - ✓ Design of POJO Server
- Docs
  - ✓ JBoss Admin&Dev Guide – Chapter 2 - The JBoss JMX Microkernel http://docs.jboss.org/jbossas/jboss4guide/r2/html/
- Wiki
  - ✓ http://www.jboss.org/wiki/Wiki.jsp?page=JBossMicrokernel
  - ✓ http://www.jboss.org/wiki/Wiki.jsp?page=FAQJBossJMX
  - ✓ http://www.jboss.org/wiki/Wiki.jsp?page=JBossKernel
- POJO Kernel - Milestone Releases
  - ✓ http://sourceforge.net/project/showfiles.php?group_id=22866&package_id=137237

JBoss World 2005

37

What about main()?



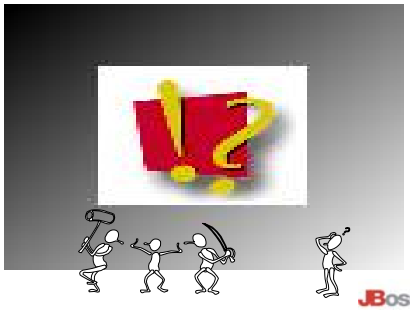
Never Say Never Again ☺

```
// Here we go again!
public class MySuperServer
{
   public static final void main(String[] args)
   {
      System.out.println("*** Starting ***");

      Kernel kernel = new JBossKernel();

      [ … lots of code omitted … ]
   }
}
```

JBoss World 2005

38

dimitris@jboss.com



JBoss World 2005

39