CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Advanced Geronimo: Custom Components and Distributions

Aaron Mulder

CTO, Chariot Solutions

Agenda

- Configurations
- Configuration Configuration
- GBeans
- Case Study: ActiveMQ
- Deployment & Management
- Case Study: ServiceMix
- Custom Distributions

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Configurations

Configurations

- A collection of services
  - Includes any number of GBean declarations
- The smallest unit of:
  - Start/stop type management
  - Class loaders
- May declare dependencies on
  - Other configurations
  - Third-party JARs

J2EE Configurations

- Every J2EE module is a configuration
- Named for the configId in the Geronimo plan for the module (or the archive name)
- The components in the module are wrapped as services (servlets, EJBs, etc.)
- Various container helper services…
- The normal dependency and service declarations are present in the Geronimo deployment plan for the module

Typical Configuration

```
<configuration configId="MyConfig"
        xmlns="http://geronimo.apache.org/xml/ns/deployment-1.0">
  <import>...</import>
  <dependency>...</dependency>
  <gbean name="MyService"
         class="some.package.MyService">
    <attribute name="foo">...</attribute>
    <reference name="bar">...</reference>
  </gbean>
  <gbean>...</gbean>
  ...
</configuration>
```

Configuration Lifecycle

- Write plan
  - Initial configuration data for each component
  - All available settings may be configured in the deployment plan
- Install required JARs into repository
- Deploy plan
  - Configuration is now stored in processed binary form, no longer tied to original plan
  - Manageable settings may be updated in config.xml
- Redeploy (replace) with revised plan
- Undeploy by name

List Configurations (CLI)

```
> java -jar bin/deployer.jar list-modules

Found 34 modules
  + geronimo/j2ee-deployer/1.0/car
  + geronimo/webconsole-jetty/1.0/car
      `-> foo.war @ http://remus:8080/foo
      `-> bar.war @ http://remus:8080/bar
  + welcome.war @ http://remus:8080/
  + geronimo/jetty-deployer/1.0/car
    geronimo/client-system/1.0/car
    geronimo/ldap-realm/1.0/car
    geronimo/online-deployer/1.0/car
    ...
```

List Configurations (Console)

APACHE GERONIMO

Server Console

About Log Out

Console Navigation

- Welcome
- Server
  - Information
  - JVM
  - Server Logs
  - Shutdown
  - Web Server
  - JMS Server
- Services
  - Common Libraries
  - Database Pools
  - JMS
- Applications
  - Deploy New
  - Application EARs
  - Web App WARs
  - EJB JARs
  - J2EE Connectors
  - App Clients
  - System Modules
- Security
  - Console Realm
  - Security Realms
  - Keystore
- Misc
  - Embedded DB
    - DB Info
    - DB Manager

Installed System Modules [view]

Component Name	State	Commands	
geronimo/activemq-broker/1.0/car	running	Stop	Uninstall
geronimo/client-corba/1.0/car	stopped	Start	Uninstall
geronimo/client-security/1.0/car	stopped	Start	Uninstall
geronimo/client-system/1.0/car	stopped	Start	Uninstall
geronimo/client/1.0/car	stopped	Start	Uninstall
geronimo/directory/1.0/car	stopped	Start	Uninstall
geronimo/geronimo-gbean-deployer/1.0/car	running	Stop	Uninstall
geronimo/hot-deployer/1.0/car	running	Stop	Uninstall
geronimo/j2ee-corba/1.0/car	running	Stop	Uninstall
geronimo/j2ee-deployer/1.0/car	running	Stop	Uninstall
geronimo/j2ee-security/1.0/car	running	Stop	Uninstall
geronimo/j2ee-server/1.0/car	running	Stop	Uninstall
geronimo/j2ee-system/1.0/car	running	Stop	Uninstall
geronimo/javamail/1.0/car	stopped	Start	Uninstall
geronimo/jetty-deployer/1.0/car	running	Stop	Uninstall
geronimo/jetty/1.0/car	running	Stop	Uninstall
geronimo/ldap-realm/1.0/car	stopped	Start	Uninstall
geronimo/online-deployer/1.0/car	stopped	Start	Uninstall
geronimo/rmi-naming/1.0/car	running	Stop	Uninstall
geronimo/shutdown/1.0/car	stopped	Start	Uninstall

CHARIOT SOLUTIONS

What is a CAR Anyway?

- Configuration ARchive
- A packaged version of a configuration, with all its metadata, etc.
  - All initial settings were made in the deployment plan
  - In the future, the manageable settings for the CAR may only be changed via config.xml
- It has proven to be deployable in the past (but, no guarantees – you may have changed the environment, etc., etc.)
- May be any configuration (EAR, service…)
- Used as part of the distribution assembly process – more on this later

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Post-Deployment Configuration

var/config/config.xml

- This file controls:
  - Which configurations are loaded, and in what order
- It lets you:
  - Override manageable attributes on the services (compared to the initial values specified in the deployment plan for the configuration)
  - Enable and disable configurations
  - Add or suppress individual GBeans, though this is pretty unusual
- It is rewritten by the server at runtime, when you make changes (e.g. through the console or JMX)

Sample config.xml

```
<attributes
        xmlns="http://geronimo.apache.org/xml/ns/attributes">
  <configuration name="RMINaming">
    <gbean name="RMIRegistry">
      <attribute name="port">1099
                                  </attribute>
    </gbean>
    <gbean name="NamingProperties">
      <attribute name="namingProviderUrl">
            rmi://0.0.0.0:1099</attribute>
    </gbean>
  </configuration>
  <configuration name="ab">...</configuration>
</attributes>
```

Configurations in config.xml

- Are started in the order they appear
  - Must be listed in order to be started
  - Must be present in the server if they're supposed to be started
- Can be disabled with the attribute `load="false"`
- Exception: a configuration will be started, possibly out of order and even if marked as disabled, if dependencies require it
  - e.g. an EJB JAR that uses CORBA may specify the core CORBA features as a dependency, and that should be started even if otherwise disabled

GBeans in config.xml

- Only need to be listed if there is something that needs to be overridden
- May represent an entirely new GBean, which is to be added to the configuration
- Can be disabled with the attribute `load="false"`
- May contain attribute or reference entries for any manageable attributes/references
  - Unlisted ones default to original plan values

Updating config.xml

- Usually only a good idea if the server is hosed
  - Won't start due to listing a configuration that's not available to the server
  - Port number conflict
- Don't bother while the server is running
- Most editing should be done by the console
  - Any runtime changes to configurations/GBean properties result in an updated config.xml
  - Of course, this only works for the GBeans that the console has edit screens for...

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

GBeans

GBeans are…

- Smallest individual components in Geronimo
- Manageable at runtime
  - JMX
  - JSR-77
  - Custom Geronimo APIs
  - Attributes can be inspected and changed
  - Performance/statistics can be exposed
- Normally configured explicitly in a plan
  - Can be created in a configuration as a "side effect" of other things (e.g. J2EE modules)
  - Some plans use custom XML formats, which are decoded to multiple GBeans (e.g. Security Realms)

GBeans have…

- A GBean class
  - Which may or may not be the meaty implementation
- Metadata describing:
  - The constructor to use
  - Attributes
  - Operations
  - Implemented interfaces
- References to other GBeans (single or multiple valued references)

GBean Names

- Every GBean must have a unique name
- Based on JMX ObjectNames
  - domain:name=value,name=value,…
- Specific components are dictated by JSR-77
  - J2EEApplication=, j2eeType=, name=, …
- Many times most of the values can be defaulted
  - <gbean name="foo" class=…> creates a GBean with the full GBean Name geronimo.server:…,name=foo

GBean Attributes

- May be of any type, whether Serializable or not
  - Simple, Serializable types are recommended for the benefit of remote management
- May be denoted as persistent and/or manageable
- Some "special" attributes are available
  - The system can provide these to a GBean, but they are never configurable (the GBean's ObjectName, the current Kernel, ClassLoader, etc.)
  - The GBean declares the special attributes in its metadata, as neither persistent nor manageable
  - Normally passed to the constructor (with no setter or getter defined)

More GBean Attributes

- Attributes are set via injection
  - Constructor injection if there's a constructor available that takes that attribute
  - Setter injection otherwise
- Persistent attributes are saved and will be re-injected into the GBean when it is instantiated
- Manageable attributes may be edited in config.xml
- If an attribute is not set in the plan, it will be set to null (for constructor injection) or not set at all (for setter injection)

GBean References

- Single-valued references
  - An attribute with a type fitting the type defined for the reference
  - Configured with a full GBeanName, or a pattern like geronimo.server:name=OtherBean,*
    - But the pattern must resolve to a unique GBean!
- Multiple-valued reference
  - An attribute of type java.util.Collection, with values of the type defined for the reference
  - Configured with one or more GBeanName patterns like geronimo.server:type=BeanType,*
  - (Eventually we want to add interface-based references)
- All matches must have the correct class or interface (as defined on the reference)

GBean Operations

- A method other than a straight accessor
- May have any arguments or return types
  - Generally speaking any operation may be invoked from a remote client, but if you know this shouldn't be the case you can have non-Serializable types in the signature
- Should not change the state of the GBean
  - Such a change would not normally be noticed and saved in the persistent state, because the kernel can only observe setter calls that go through it
  - *Workaround: get a kernel via the kernel special attribute, and then call setter method on yourself through the kernel*
  - May still create new GBeans, change the state of other GBeans via references, etc.

GBean Interfaces

- A GBean does not strictly require any interface
  - Generally must have either an empty constructor or implement an interface
- It's most convenient to have a management interface including key management attributes and operations
  - Any client can request a proxy to a GBean that implements one or all GBean interfaces
  - Much easier to code to an interface than e.g. JSR-77 `setAttribute("gbean","name","value")`
- Often want to implement `GBeanLifecycle` in order to take action during startup/shutdown

GBean Metadata

- Stored in an object of type `GBeanInfo`
- This must be provided by a static `getGBeanInfo` method on the GBean class
- There are helper routines to construct it
- This is really the only requirement of the GBean class
  - The actual implementation with all the operations and attributes and things may be a separate class
    - This makes it easy to wrap a non Geronimo-specific service with GBeans for startup, configuration, and management

Sample GBean Metadata

```
public static final GBeanInfo GBEAN_INFO;
static {
  GBeanInfoBuilder factory =
    GBeanInfoBuilder.createStatic(
      GBeanClass.class, ImplClass.class);
  factory.addAttribute("att1name", Class);
  factory.addOperation("op1name");
  factory.addInterface(Class);
  factory.addReference("ref1name", Class);
  factory.setConstructor(new
        String[]{"att1name", "att2name"});
  GBEAN_INFO = factory.getBeanInfo();
}
```

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Case Study: ActiveMQ

ActiveMQ Overview

- The ActiveMQ server needs a Broker plus one or more Connectors (transports)
- Then various connection factories and destinations can be created as needed
- Geronimo wants an embedded broker, with at a minimum the in-VM transport (but usually TCP/IP too)
  - This means a Broker GBean, one or more Connector GBeans, and a configuration for them

ActiveMQ GBeans

- For ActiveMQ 3.x
  - See activemq-gbean module in the ActiveMQ source
  - GBeans for
    - Container (broker)
    - Various persistent stores
    - Connector
    - Manager (for management purposes, more later)
- For ActiveMQ 4.x
  - Haven't been updated yet, in the sandbox

ActiveMQ 3.x GBeans

- The Container GBean has a reference to the first persistence GBean
- Each persistence GBean has a reference to the next (cache sits on journal on DB…)
  - Some have additional stuff, such as a directory or database settings
- Each connector GBean has a reference to a container GBean (the connector channels I/O to that container…)
- The manager is not directly connected to the rest

ActiveMQ 3.x GBean Style

- Generally pretty thin layer on top of existing ActiveMQ objects, used to:
  - Gather configuration data for the underlying objects
  - Wire the objects together
  - Allow them to be managed by the console
- Amount of configuration means they are non-trivial, but you don't see any message-handling code there

ActiveMQ: The Gory Details

- Not going to show in full here
- For the GBeans, poke around at:
  - svn+ssh://svn.activemq.org/scm/activemq/trunk/activemq/modules/gbean/src/java/org/activemq/
- For the configuration plan, look at:
  - https://svn.apache.org/repos/asf/geronimo/branches/1.0/configs/activemq-broker/src/plan/plan.xml

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Deployment and Management

Deployment

- Geronimo has a master Deployer GBean (in the *geronimo-gbean-deployer* configuration)
- The Deployer has a multi-value reference to ConfigBuilder implementations
- During deployment, each ConfigBuilder gets to look at a module and decide whether it handles it
- To add modules of a new type, simply create and deploy a new ConfigBuilder

Config Builders

- Normally looks for a specific deployment descriptor (`WEB-INF/web.xml`, etc.)
- Responsible for constructing all the GBeans required to represent the application module being deployed
- Gets an archive and maybe a plan as input, and produces a Configuration full of application GBeans as output
- Itself a GBean, of course

Config Builder Implemenation

- Normally heinously complex
- All the GBean classes must already exist
- The builder prepares the instance-level metadata for each GBean
  - Values for each attribute or reference specified in the GBeanInfo for that GBean
- Often driven by processing XML files (such as J2EE deployment descriptors)
- Assembles a configuration as a set of all the GBeans for the input module

Why Do You Care?

- To add new module types, of course
  - A Spring archive format
  - A Hibernate archive format
  - A ServiceMix archive format
  - A whatever-you-want archive format
- Many services don't require this, but think it over
  - A scheduler may typically not need this…
  - But what if you define a "job" archive format? Then you could hot deploy and start and stop jobs at runtime using all the existing Geronimo plumbing…

Management

- Any GBean may be invoked via JMX and JSR-77
  - Looks a lot like reflection (ugh)
  - Keep signatures simple for this!
- Providing an interface lets a client build and interact with a proxy
  - e.g. A generated class that implements the requested interface and makes the nasty kernel calls entirely under the covers
- Integrating the service interface into the JSR-77 component tree can make it easy to locate the component and/or a proxy

JSR-77 Component Tree

- JSR-77 starts with a Domain; the domain has Servers; servers have JVMs and applications and…
- Can navigate around (Domain.getServers(), etc.), though the navigation methods return ObjectNames
  - Some helper classes are around to automatically decode these to proxies
- This is fine for a custom distribution or core Geronimo functionality
  - No clean process for registering or looking up extensions that may or may not be there

Customizing JSR-77 Components

- **The JSR-77 interfaces defined by the spec are in *org.apache.geronimo.management***
- **Geronimo extensions are in *org.apache.geronimo.management.geronimo***
  - **Add Geronimo-specific methods, e.g. to get the ServerInfo or WebContainer for a J2EEServer**
- **Non-product-specific interfaces can be added here (e.g. RulesEngine, but not DroolsEngine)**

Manager Classes

- For services that have ancillary connectors or other configurable services
  - e.g. web container has ports/protocols
  - CORBA ORB has TSS/CSS configurations
- A manager class has methods to navigate to, add, and remove those children
  - Otherwise, can be hard for a client to resolve references, apply default values, etc.
- Examples: JMS manager, Web container manager

Management Console

- Built from portlets
- Can add a portlet for any service
- Has lots of hooks for accessing management proxies, etc.
- Not as nice as it could be
  - Must manually configure the contents of each portal page
  - One class loader for the whole console, so the custom service's interfaces must be on the console class path
  - Scheduled for enhancement

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Case Study: ServiceMix

ServiceMix Overview

- Java Business Integration (JBI) container
  - Need a container GBean to initialize ServiceMix and any container-level services
  - Uses Geronimo thread pool and Geronimo transaction manager
- Individual Service Assemblies (packaged as zip file) can be deployed to ServiceMix
  - Need to add a deployer service to handle runtime deployment of service assemblies
- Potential for still more integration
  - Binding components to expose other services in Geronimo to the JBI bus?
  - Security spanning J2EE application code and JBI calls?

ServiceMix GBean Detail

- Container GBean gets the basic configuration items, and starts and stops the ServiceMix container
  - Thread pool, configuration directory, etc.
- Deployer GBean knows how to pass service assemblies to the container (which then uses the normal logic)
  - Reference to the container
  - Default parent for the service assembly Configuration

ServiceMix Container GBean

```
<gbean name="ServiceMix"
class="org.apache.servicemix.gbean.Service
MixGBean">
  <attribute name="name">ServiceMix
  </attribute>
  <attribute name="directory">servicemix/
  </attribute>
  <reference
    name="transactionContextManager">...
  </reference>
  <reference name="workManager">...
  </reference>
</gbean>
```

ServiceMix Deployer GBean

```
<gbean name="ServiceMixDeployer"
class="org.apache.servicemix.gbean.Service
MixConfigBuilder">
  <attribute name="defaultParentId">
    ServiceMixConfigID
  </attribute>
  <reference name="servicemix">
    <name>ServiceMix</name>
  </reference>
</gbean>
```

Room For Improvement

- The container configuration is still largely controlled by `servicemix.xml`
- Would be nice to have GBeans wrapping JBI components, so statically configured components could be configured that way
  - Would then also use those for components deployed at runtime for additional management, statistics, etc.
- Would be nice to add some more GBeans with normalized message router & bus statistics, etc.

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Custom Distributions

A Custom Distribution

- A version of Geronimo including only the services (configurations) you want
- Maybe more lightweight
- Maybe more heavyweight
- Maybe just different components
- Could just start with the default distribution and deploy and undeploy things, though that's not as clean and repeatable

Elements of a Distribution

- Each configuration is packaged into a portable format known as a CAR file
- This includes the processed configuration, metadata, etc.
- A distribution (assembly) is built by starting empty and applying CAR files until the server looks like what you want
- Then zipped for distribution

Distribution Prerequisites

- Anything you want to add must be either:
  - An application (normally with a Geronimo deployment plan)
  - A service configuration (itself a Geronimo deployment plan)
- Anything you want to remove must be in a separate plan from the things you want to keep
  - In 1.0, for example, the EJB container was in the same plan as the core J2EE infrastructure, meaning you couldn't keep the transaction manager but drop EJBs

Creating a Distribution

- The current Geronimo assemblies are built using a series of Maven plugins
- The packaging plugin creates a CAR file
  - See examples under geronimo/configs/*/
- The assembly plugin applies CAR files
  - See examples at geronimo/assemblies/j2ee-(webcontainer)-server/

Room for Improvement

- Would be nice to have a wide selection of CAR files online somewhere, and an apt-like tool to list, download, and install them
- Would be nice to have command-line tools to import/export CAR files from a Geronimo installation
- Currently difficult to handle certain config builders that things depend on
  - Web Services rely on the Web Container, but the Web and EJB deployers also rely on a Web Services builder being defined...
  - Can replace one of these with a "null" builder, but cannot leave it out entirely without having unresolved reference errors

CHARIOT SOLUTIONS

OPTIMIZING YOUR INVESTMENT IN JAVA TECHNOLOGY

Q&A

http://chariotsolutions.com/geronimo/