Administering Apache Geronimo With Custom Server Assemblies and Maven

David Jencks

What is Geronimo?

- JavaEE 5 certified application server from Apache
- Modular construction
- Wires together other projects
  - –OpenEJb
  - –ActiveMQ
  - –Jetty, Tomcat
  - –Cxf, Axis2
  - –MyFaces

Geronimo Philosophy

- Package configuration and code together into reusable units (plugins)
- Expose limited customization options
- Expose dependency information
- Use the exposed information to assemble a server containing the desired functionality (your apps) together with the infrastructure needed to run it (EE containers)

ApacheCon

What is Maven?

- Build and project management system that explicitly encodes the relationship of your project modules to all other software.
- Provides a naming and distribution system for software artifacts

Geronimo with Maven

- Both use the same repository structure
- Maven includes more/different metadata about multiple versions and build instructions
- Geronimo includes more geronimo specific dependency information and customization info
- Possible to do without maven but
  - –harder to script
  - –harder to get in scm

Traditional Server Configuration

- App server is regarded as monolithic black box
- Configuration scripts are applied to server
- Applications are deployed into server
- Unclear how to determine the configuration state of a server instance

Geronimo configuration with Maven

- Your application is the center
- Maven packages configuration with the application to form a plugin
- Dependency info packaged with plugin, derived from maven dependency info
- Plugin deployed to maven repository providing distribution mechanism
- Custom Server can be assembled around app and deployed to maven repo
- Everything is in scm... it is part of your build

What's a Geronimo Plugin?

- Deployed artifact with additional metadata
- Examples:
  - bunch of files to install, no classloader (rare)
  - a classloader with parents and jars
  - classloader with geronimo services (gbeans)
  - deployed javaee app with classloader and gbeans

Plugin metadata

- Metadata includes
  - cataloging information (name, description, category)
  - environment dependencies (jdk)
  - installation dependencies (other plugins and jars)
  - exposed configuration
  - replacement information (artifact-aliases)
  - instructions on what to unpack on installation
    - spring xml files
    - data files
    - log4j configuration

Geronimo plans

- All plugins need a plan
  - –src/main/plan/plan.xml in maven project
- type depends on kind of plugin
  - –system module (gbeans only)
  - –javaee app (plan similar to ee dd)
- admin console has wizards
- copy and modify examples
  - –geronimo is assembled from plugins

Access to Geronimo plugins

- Geronimo plugin repo (looks like a maven repo)
- Geronimo server (if it has a web server installed)
- Local maven repo - local builds install plugins
- Remote maven repo such as a repository manager (e.g. nexus)
- Plugin catalog is helpful for manual assembly.

Server Customization

- Install different plugins
  - plugins can indicate they substitute for another using “artifact aliases”
- Configure plugins
  - plugins can expose properties for customization in “config substitutions”. Initial values are supplied in the plugin
  - When assembling a server, property customizations can be installed

Example of plugin substitution

- Developers need independent clean environment
  - –derby db and properties security realm
  - –default plugins
- QA needs realistic non-production environment
  - –production db and ldap copies
  - –qa plugins
- Production needs isolated environment
  - –production plugins

project structure

- my-app (ear, perhaps)
- my-app-jetty-cxf (pluginized-ear)
  - dependencies on dev plugins
- db-dev (embedded derby)
- security-dev (local property files)
- db-qa (isolated db2)
  - aliased to replace dev plugin
- security-qa (isolated apacheds)
- db-production (production db2)
- security-production (production ldap)
- server-dev
- server-qa
- server-production

Service Customization

- Services (gbeans) configured in geronimo plan
- Overrides in config.xml
- Overrides can use ${property}
- Property values from config-substitutions.properties

```
plan:
    <gbean name="JettyWebConnector" class="org.apache.geronimo.jetty6.connector.HTTPSelectChannelConnector">
        <attribute name="host">localhost</attribute>
        <attribute name="port">8080</attribute>
        <attribute name="headerBufferSizeBytes">8192</attribute>
        <reference name="JettyContainer">
            <name>JettyWebContainer</name>
        </reference>
        <reference name="ThreadPool">
            <name>DefaultThreadPool</name>
        </reference>
        <attribute name="maxThreads">50</attribute>
    </gbean>

config.xml:
            <gbean name="JettyWebConnector">
                <attribute name="host">${ServerHostname}</attribute>
                <attribute name="port">${HTTPPort + PortOffset}</attribute>
                <attribute name="redirectPort">${HTTPSPortPrimary + PortOffset}</attribute>
            </gbean>

config-substitutions.xml:
ServerHostname=localhost
HTTPPort=8080
HTTPSPortPrimary=8443
PortOffset=0
```

Plugin Metadata

- geronimo-plugin.xml file
- Normally constructed by car-maven-plugin
  - –dependencies
  - –catalog info (description, category)
  - –version info
  - –customization options (config.xml content)
  - –artifact aliases
  - –substitution values
  - –unpacking instructions

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<geronimo-plugin xmlns="http://geronimo.apache.org/xml/ns/plugins-1.3" xmlns:ns2="http://geronimo.apache.org/
xml/ns/attributes-1.2">
    <name>Geronimo Plugins, Jetty :: Jetty 6</name>
    <pluginGroup>false</pluginGroup>
    <description>Geronimo Jetty Web Server integration.</description>
    <url>http://geronimo.apache.org/</url>
    <author>The Apache Geronimo development community</author>
    <license osi-approved="true">The Apache Software License, Version 2.0</license>
    <plugin-artifact>
        <module-id>
            <groupId>org.apache.geronimo.configs</groupId>
            <artifactId>jetty6</artifactId>
            <version>2.2-SNAPSHOT</version>
            <type>car</type>
        </module-id>
        <geronimo-version>2.2-SNAPSHOT</geronimo-version>
        <jvm-version>1.5</jvm-version>
        <jvm-version>1.6</jvm-version>
        <dependency>
            <groupId>org.apache.geronimo.configs</groupId>
            <artifactId>j2ee-server</artifactId>
            <version>2.2-SNAPSHOT</version>
            <type>car</type>
        </dependency>
----lots more dependencies, omitted ---
        <source-repository>~/.m2/repository/</source-repository>
        <source-repository>http://repo1.maven.org/maven2/</source-repository>
        <source-repository>http://people.apache.org/repo/m2-snapshot-repository/</source-repository>
```

----continued

```
    <config-xml-content>
      <ns2:gbean name="JettyWebConnector">
        <ns2:attribute name="host">${ServerHostname}</ns2:attribute>
        <ns2:attribute name="port">${HTTPPort + PortOffset}</ns2:attribute>
        <ns2:attribute name="redirectPort">${HTTPSPortPrimary + PortOffset}</ns2:attribute>
      </ns2:gbean>
      <ns2:gbean name="JettyAJP13Connector">
        <ns2:attribute name="host">${ServerHostname}</ns2:attribute>
        <ns2:attribute name="port">${AJPPort + PortOffset}</ns2:attribute>
        <ns2:attribute name="redirectPort">${HTTPSPortPrimary + PortOffset}</ns2:attribute>
      </ns2:gbean>
      <ns2:gbean name="JettySSLConnector">
        <ns2:attribute name="host">${ServerHostname}</ns2:attribute>
        <ns2:attribute name="port">${HTTPSPort + PortOffset}</ns2:attribute>
      </ns2:gbean>
    </config-xml-content>
    <artifact-alias key="org.apache.geronimo.configs/jetty6//car">
      org.apache.geronimo.configs/jetty6/2.2-SNAPSHOT/car</artifact-alias>
    <artifact-alias key="org.apache.geronimo.configs/jetty6/2.1.4/car">
      org.apache.geronimo.configs/jetty6/2.2-SNAPSHOT/car</artifact-alias>
    <config-substitution key="HTTPPort">8080</config-substitution>
    <config-substitution key="AJPPort">8009</config-substitution>
    <config-substitution key="HTTPSPort">8443</config-substitution>
    <config-substitution key="webcontainer">JettyWebContainer</config-substitution>
    <config-substitution key="webcontainerName">jetty6</config-substitution>
  </plugin-artifact>
</geronimo-plugin>
```

Unpacking resources

- pack resources into plugin
- specify where to unpack

```
<copy-file relative-to="server" dest-dir="var">security</copy-file>
```

- examples:
  - –tomcat base configuration files
  - –property login module files
  - –activemq or apacheds spring xml files
  - –base server directory structure

Building plugins with car-maven-plugin

- From maven pom:
  - dependencies (can customize scope)
    - also inserted into geronimo plan
  - name
  - description
- One level of overriding from parent pom
  - e.g. category
- config.xml contents, artifact aliases, and property values from maven plugin configuration
- local plugin catalog maintained in local maven repo.

Plugin maven project

```
+-src
  +-main
    +-plan
      +-plan.xml  //gbeans, g. jee plan
    +-resources
      +-myappdata //data to unpack
|-pom.xml
  //maven id becomes plugin id
  //dependencies specify classloader
  //car-maven-plugin configuration
    //specifies where to unpack, and
    //content of configuration files.
```

```
<plugin>
  <groupId>org.apache.geronimo.buildsupport</groupId>
  <artifactId>car-maven-plugin</artifactId>
  <configuration>
    <instance>
      <plugin-artifact>
        <config-xml-content>
          <gbean name="JettyWebConnector">
            <attribute name="host">#{ServerHostname}</attribute>
            <attribute name="port">#{HTTPPort + PortOffset}</attribute>
            <attribute name="redirectPort">#{HTTPSPortPrimary + PortOffset}</attribute>
          </gbean>
          <gbean name="JettyAJP13Connector">
            <attribute name="host">#{ServerHostname}</attribute>
            <attribute name="port">#{AJPPort + PortOffset}</attribute>
            <attribute name="redirectPort">#{HTTPSPortPrimary + PortOffset}</attribute>
          </gbean>
          <gbean name="JettySSLConnector">
            <attribute name="host">#{ServerHostname}</attribute>
            <attribute name="port">#{HTTPSPort + PortOffset}</attribute>
          </gbean>
        </config-xml-content>
        <artifact-alias key="org.apache.geronimo.configs/jetty6/2.1.4/car">
          org.apache.geronimo.configs/jetty6/2.2-SNAPSHOT/car</artifact-alias>
        <config-substitution key="HTTPPort">8080</config-substitution>
        <config-substitution key="AJPPort">8009</config-substitution>
        <config-substitution key="HTTPSPort">8443</config-substitution>
        <config-substitution key="webcontainer">JettyWebContainer</config-substitution>
        <config-substitution key="webcontainerName">jetty6</config-substitution>
      </plugin-artifact>
    </instance>
  </configuration>
</plugin>
```

Assembling a custom server

- car-maven-plugin with server-assembly packaging
- list all the plugins you need as dependencies
- transitive dependencies (through geronimo-plugin.xml, not maven) will be pulled in
- requires rather unpleasant server model configuration.

Server assembly maven project

```
+-src
 +-main
  +-resources
   +-var
    +-config
     +-overrides
      +server-overrides.xml
|-pom.xml
  //maven id becomes server id
  //dependencies specify server
    //contents, following transitive
    //dependencies
```

Leading the Wave
of Open Source

Server models

- Several “servers” can be run from one geronimo installation (e.g. main ee server, app client container)
- Each “server” has a set of configuration files e.g.
  - var/config/config.xml
  - var/config/config-substitutions.properties
  - var/config/artifact-aliases.properties
- Metadata about each server is contained in a “model” indicating where the configuration files are
- Plugin metadata indicates which model the customization goes into.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <parent>
    <artifactId>activemq</artifactId>
    <groupId>org.apache.geronimo.plugins</groupId>
    <version>2.2-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.apache.geronimo.plugins</groupId>
  <artifactId>activemq-server</artifactId>
  <packaging>server-assembly</packaging>
  <name>Geronimo ActiveMQ v5 Micro-Server</name>
  <description>Small server for testing activemq v5 integration</description>

  <dependencies>
    <dependency>
      <groupId>org.apache.geronimo.framework.plugingroups</groupId>
      <artifactId>framework</artifactId>
      <version>${geronimoVersion}</version>
      <type>car</type>
    </dependency>
    <dependency>
      <groupId>org.apache.geronimo.configs</groupId>
      <artifactId>activemq-broker</artifactId>
      <version>${geronimoVersion}</version>
      <type>car</type>
    </dependency>
    <dependency>
      <groupId>org.apache.geronimo.configs</groupId>
      <artifactId>activemq-ra</artifactId>
      <version>${geronimoVersion}</version>
      <type>car</type>
    <plugin>
        <groupId>org.apache.geronimo.buildsupport</groupId>
        <artifactId>car-maven-plugin</artifactId>
        <configuration>
<!-- put this stuff in a shared parent pom configuration if at all possible -->
            <servers>
                <serverInstance>
                    <name>default</name>
                    <configFile>var/config/config.xml</configFile>
                    <configSubstitutionsFile>var/config/config-substitutions.properties</configSubstitutionsFile>
                    <configSubstitutionsPrefix>org.apache.geronimo.config.substitution.</configSubstitutionsPrefix>
                    <artifactAliasesFile>var/config/artifact_aliases.properties</artifactAliasesFile>
                </serverInstance>
                <serverInstance>
                    <name>client</name>
                    <attributeManagerFrom>default</attributeManagerFrom>
                    <artifactAliasesFile>var/config/client_artifact_aliases.properties</artifactAliasesFile>
                </serverInstance>
                <serverInstance>
                    <name>offline</name>
                    <configFile>var/config/offline-deployer-config.xml</configFile>
                    <configSubstitutionsFile>var/config/config-substitutions.properties</configSubstitutionsFile>
                    <configSubstitutionsPrefix>org.apache.geronimo.config.substitution.</configSubstitutionsPrefix>
                    <artifactAliasesFile>var/config/artifact_aliases.properties</artifactAliasesFile>
                </serverInstance>
                <serverInstance>
                    <name>jsr88</name>
                    <configFile>var/config/jsr88-configurer-config.xml</configFile>
                    <configSubstitutionsFile>var/config/config-substitutions.properties</configSubstitutionsFile>
                    <configSubstitutionsPrefix>org.apache.geronimo.config.substitution.</configSubstitutionsPrefix>
                    <artifactAliasesFile>var/config/artifact_aliases.properties</artifactAliasesFile>
                </serverInstance>
            </servers>
```

Leading the Wave of Open Source

```
<configuration>
    <overrides>
        <override>
            <server>default</server>
            <overrides>server-overrides.xml</overrides>
        </override>
    </overrides>
</configuration>
```

Doing without maven

- don’t, the process is hard to get in scm
- admin console has incomplete metadata editing
  - –no config.xml, artifact-aliases, properties
- extract plugin from admin console or gshell
- extract server from admin console or gshell
  - –server is based on plugins, without any manual customizations that may have been done.

Framework Server with Plugins

- Framework server contains just enough functionality to install plugins
- With the Cluster node plugin the server can join a multicast cluster
- Scripts or cluster controller can instruct servers to install the plugins of interest
- Plugins typically on maven repo such as sonatype nexus
- All required plugins installed as dependencies

Maven review

- projects are multi-module
- maven archetypes help create new modules
- one artifact generated per module
- project-wide configuration in parent project, inherited by sub-modules.
- geronimo plugin archetype
- geronimo assembly archetype
- Easy to run from m2eclipse and (theoretically) idea

geronimo-plugin-archetype

- get archetypes into catalog
  - get archetypes into repo
  - run mvn archetype:crawl
- mvn archetype-generate
  - supply info requested
- Move appropriate configuration into parent pom
  - add dependencies
  - add plan
  - add deployers
  - for ee, add ee application
  - add configuration

g-p-a gotchas

- You have to list all the deployers needed as dependencies with scope provided
- You have to configure all the deployers needed in the car-maven-plugin (properties are suggested to make this easier)
- You have to include as maven dependencies all the dependencies the deployers add automatically
- This can all be fixed with a better archetype

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <parent>
    <artifactId>activemq</artifactId>
    <groupId>org.apache.geronimo.plugins</groupId>
    <version>2.2-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <artifactId>activemq-webconsole-jetty</artifactId>
  <name>ActiveMQ web console on Jetty</name>
  <packaging>car</packaging>
<!-- deployers -->
  <dependencies>
    <dependency>
      <groupId>org.apache.geronimo.framework</groupId>
      <artifactId>geronimo-gbean-deployer</artifactId>
      <version>${geronimoVersion}</version>
      <type>car</type>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.apache.geronimo.configs</groupId>
      <artifactId>j2ee-deployer</artifactId>
      <version>${geronimoVersion}</version>
      <type>car</type>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.apache.geronimo.configs</groupId>
      <artifactId>${jetty}-deployer</artifactId>
      <version>${geronimoVersion}</version>
      <type>car</type>
      <scope>provided</scope>
```

```
<!-- server components to run app -->
      <dependency>
          <groupId>org.apache.geronimo.configs</groupId>
          <artifactId>${jetty}</artifactId>
          <version>${geronimoVersion}</version>
          <type>car</type>
       </dependency>
       <dependency>
          <groupId>org.apache.geronimo.configs</groupId>
          <artifactId>jasper</artifactId>
          <version>${geronimoVersion}</version>
          <type>car</type>
       </dependency>
       <dependency>
          <groupId>org.apache.geronimo.configs</groupId>
          <artifactId>activemq-broker</artifactId>
          <version>${geronimoVersion}</version>
          <type>car</type>
       </dependency>
<!-- the actual javaee app we're deploying -->
       <dependency>
          <groupId>org.apache.geronimo.plugins</groupId>
          <artifactId>activemq-webconsole</artifactId>
          <version>${geronimoVersion}</version>
          <type>war</type>
          <scope>provided</scope>
       </dependency>
```

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.geronimo.buildsupport</groupId>
                <artifactId>car-maven-plugin</artifactId>
                <configuration>
                    <deploymentConfigs>
<!-- properties defined in parent pom: refer to deployer plugins -->
                        <deploymentConfig>${gbeanDeployer}</deploymentConfig>
                        <deploymentConfig>${j2eeDeployer}</deploymentConfig>
                        <deploymentConfig>${jettyDeployer}</deploymentConfig>
                        <deploymentConfig>${jasperDeployer}</deploymentConfig>
                    </deploymentConfigs>
<!-- when deploying an ee app, list it as a module: -->
                    <module>
                        <groupId>org.apache.geronimo.plugins</groupId>
                        <artifactId>activemq-webconsole</artifactId>
                        <type>war</type>
                    </module>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

geronimo-assembly-archetype

- mvn archetype:generate....
- if you are assembling more than one server, move configuration into parent
- add framework plugin to dependencies
- add your app plugins to dependencies
- Once you install an artifact-alias, the replaced artifact will not be installed

Server Customization files

- Plugins can only install customizations for themselves in config.xml
- Occasionally that is not enough.
- Include additional customization files in src/main/resources/var/config/overrides/<server-name>-overrides.xml
- Configure car-maven-plugin for server assembly

```
<configuration>
  <overrides>
    <override>
      <server>default</server>
      <overrides>server-overrides.xml</overrides>
    </override>
  </overrides>
</configuration>
```

Sample server customization

```
<attributes xmlns="http://geronimo.apache.org/xml/ns/attributes-1.2">
  <module name="org.apache.geronimo.framework/j2ee-security/${geronimoVersion}/car">
<!-- Turn off a single gbean in a plugin -->
    <gbean name="SecurityService" load="false"/>
  </module>
<!-- don't load a plugin at all -->
  <module name="org.apache.geronimo.framework/server-security-config/${geronimoVersion}/car" load="false"/>
  <module name="org.apache.geronimo.configs/activemq-ra/${geronimoVersion}/car" load="false"/>
<!-- override some settings not exposed as config-substitutions -->
  <module name="org.apache.geronimo.configs/j2ee-corba-yoko/${geronimoVersion}/car">
    <gbean name="CORBASSLConfig">
      <attribute name="keyStore">clientcert.jks</attribute>
      <attribute name="keyAlias">cts</attribute>
      <attribute name="trustStore">ssl-truststore</attribute>
      <attribute name="protocol">SSL</attribute>
    </gbean>
    <gbean name="NameServer">
      <attribute name="port">${orbDefaultPort}</attribute>
    </gbean>
    <gbean name="Server">
      <attribute name="port">6684</attribute>
      <attribute name="host">localhost</attribute>
    </gbean>
  </module>
...
```

Plugin based clustering (trunk)

- Cluster nodes are framework + node plugin
- Cluster admin node tracks cluster members, plugins, plugin lists.
- Multicast discovery
- When a new node is detected it is instructed to install the appropriate plugins.
- All plugins and dependencies downloaded from a plugin repo
- Does not solve provisioning problem of distributing and starting the node servers.
- Cute but just distributing custom servers may be simpler

Summary

- Plugins include maven-style dependency information
- Installing a plugin pulls in all its dependencies
- Plugins have a customization facility
- Assembling a server starting with your application plugins pulls in all the server parts needed to run your apps
- All the configuration is in scm