

# Introducing Apache Geronimo 1.1

Aaron Mulder  
CTO, Chariot Solutions  
Committer, Apache Geronimo

# Speaker

- Aaron Mulder
- Geronimo Developer
  - Works on deployment, management, console, kernel, plugins, ...
  - Online Geronimo book at <http://chariotsolutions.com/geronimo/>
- CTO of Chariot Solutions
  - Java/Ruby/Open Source consulting firm
  - Partnerships with companies that provide Geronimo support (IBM, Covalent, etc.)

# Agenda

- Overview & Status
- Server Installation and Configuration
- Administration Console
- Deployment & Tools
- Little G
- Geronimo Plugins
- Q&A

# Overview and Status

# Overview

- Complete J2EE 1.4 Application Server
  - Or lightweight, web-focused “Little G”
- Built on best of breed components (Tomcat, Jetty, ActiveMQ, OpenEJB, HOWL, etc.)
- Modular architecture (entirely built w/modules)
- Plugins make it easy to add features
- Web-based admin console for configuration
- Web & JMS clustering support
- In-place upgrades with no restart

# Why is Geronimo Necessary?

- Other open source application servers are encumbered by:
  - \*GPL licensing
  - Not entirely open community, limited options for support, training, etc.
- Tomcat has only web features
  - Even if you don't buy into EJB, the world needs more (messaging, transactions, etc.)
- The app server market is losing innovation (versus, for example, Ruby on Rails)

# Project Status

- 1.0, 1.1 releases so far this year
  - 1.1.1 this week
  - 1.2 this year
- J2EE certified, all J2EE features complete
- Administration console handles common configuration/deployment tasks and keeps growing
- Command-line & IDE tools available
- Initial project documentation, plus a number of books, many articles, etc.

# Community

- 30+ committers from 9+ companies
- Extremely active mailing lists
  - dev list for build/development issues
  - user list for questions & support
- Many companies provide stacks including Geronimo and/or 24/7 support
- Multiple companies building applications on Geronimo, open source and commercial
- Sites as big as Ebay deployed on Geronimo



# Advantages

- Open community & open license
- First app server with plugins
  - Can build and clone a custom distribution
- Web management console
- High-speed JMS messaging server included
- Integration with many other products, open source and commercial
- Many training and support options (both free and commercial support available)

# Management

- Admin console includes:
  - Add/configure network ports
  - Deploy & manage applications
  - Deploy & configure database pools, JMS connection factories, and JMS destinations
  - Deploy & configure security realms
  - Configure keystores & HTTPS
  - Configure Geronimo with Apache HTTP
  - Create or install plugins

# Performance

- Tested with the DayTrader sample application (also open source)
  - Can be configured and run on many app servers for comparison purposes
  - Can test different APIs (JDBC vs CMP, Web, JNDI, EJB, JMS, Web Services, etc.)
- Performance is competitive with commercial J2EE application servers
  - Still occasional weak spots (such as CMP)

# Security

- Definitely a project priority
- Pluggable security realms
  - For J2EE application security
- Initial pluggable JACC support as well
- Password encryption used in config files
- Authentication required for remote deployment and management
- Security bugs are a high priority

# Installation and Configuration

# Installation

- Windows and Mac/UNIX/Linux distributions
- Download and unzip either the Jetty or the Tomcat distribution
- Start it up and go!
- In case of conflict, edit network ports in `var/config/config.xml` (more in a bit)
  - Have to get the web container running before the admin console is available

# Start and Stop

- Easy start and stop commands for developers
- Scripts support various execution and logging options, as well as allowing JVM customization
- Can run Geronimo as a Windows service or UNIX daemon
- Can start/stop in scripts (e.g. for testing)
- Can use admin console or command-line tools to shut down or restart a remote server
  - Can't yet start a remote server

# Startup Sequence

```
Booting Geronimo Kernel (in Java 1.4.2_09)...
```

```
Starting Geronimo Application Server
```

```
[*****] 100% 18s Startup complete
```

```
Listening on Ports:
```

```
1099 0.0.0.0 RMI Naming
```

```
1527 0.0.0.0 Derby Connector
```

```
4201 0.0.0.0 ActiveIO Connector EJB
```

```
4242 0.0.0.0 Remote Login Listener
```

```
8080 0.0.0.0 Jetty Connector HTTP
```

```
8443 0.0.0.0 Jetty Connector HTTPS
```

```
61616 0.0.0.0 ActiveMQ Message Broker Connector
```

```
Started Application Modules:
```

```
EAR: org/apache/geronimo/Console/Jetty
```

```
WAR: org/apache/geronimo/applications/Welcome/Jetty
```

```
Web Applications:
```

```
http://server-hostname:8080/
```

```
http://server-hostname:8080/console
```

```
http://server-hostname:8080/console-standard
```

```
Geronimo Application Server started
```



# Configuration (easy)

- Start server and point browser to `http://localhost:8080/console/`
- Use the screens there to edit network ports, add database connection pools, configure security and JMS resources, etc.
- Can't use if original network ports conflict
  - Use the next option to resolve ports and then go into the console. :)

# Configuration (hard)

- Most configuration is controlled by `config.xml` in `var/config`
  - Controls which modules to load
  - Lets you override settings on any server component (identified by config name + component name + attribute name)
- Note that the server rewrites this file while it's running
  - Edit it only while the server is down!

# config.xml

```
<attributes
xmlns="http://geronimo.apache.org/xml/ns/attributes-1.1">
  <module name="geronimo/rmi-naming/1.0/car">
    <gbean name="RMIRegistry">
      <attribute name="port">1099</attribute>
    </gbean>
    <gbean name="NamingProperties">
      <attribute name="namingProviderUrl">
        rmi://0.0.0.0:1099
      </attribute>
    </gbean>
  </module>
  <module name= ...  />
  ...
</attributes>
```

# Logging

- Uses Log4J
- Config file at `var/log/server-log4j.properties`
- Server log at `var/log/geronimo.log`
- Can easily customize log output, rolling log files, logging to NT/UNIX system logs, etc.
- Can search and view server logs from the admin console

# Security & Login Modules

- A security realm normally uses one JAAS LoginModule, but may include several
- Extra features are added by using multiple LoginModules for the realm
  - auditing, lockout, extra credentials, etc.
- Can also use multiple login modules to access users in separate back-end security repositories (2 LDAP servers, LDAP for users and database for application roles, etc.)

# Realm Example

## SQLSecurityRealm

---

1. **SQL Login Module** → Required
2. **Lockout Login Module** → Required
3. **Auditing Login Module** → Optional

# Included Login Features

- Properties File
- Database
- LDAP
- Active Directory
- Kerberos
- Auditing
- Lockout on repeated failure
- Save credentials to use when invoking a web service or CORBA EJB

# Administration Console



## Database Pools

The screenshot shows the Apache Geronimo Console interface in a Mozilla Firefox browser. The main content area is titled 'Database Pools' and displays the 'Create Database Pool -- Step 2: Select Driver, JAR, Parameters' configuration form. The form includes the following fields and options:

- JDBC Driver Class:**  See the documentation for your JDBC driver.
- Driver JAR:**  The JAR holding the selected JDBC driver. Should be installed under GERONIMO/repository/ (or [Download a Driver](#))
- DB User Name:**  The username used to connect to the database
- DB Password:**  The password used to connect to the database
- Driver Connection Properties:**
  - Typical JDBC URL:**  A property used to connect to PostgreSQL. May be optional (see JDBC driver documentation).
  - Port:**  A property used to connect to PostgreSQL. May be optional (see JDBC driver documentation).
  - Host:**  A property used to connect to PostgreSQL. May be optional (see JDBC driver documentation).
  - Database:**  A property used to connect to PostgreSQL. May be optional (see JDBC driver documentation).

Buttons for 'Next' and 'Cancel' are visible at the bottom of the form. The left sidebar shows the 'Console Navigation' menu with various options like 'Server', 'Services', 'Applications', and 'Security'.

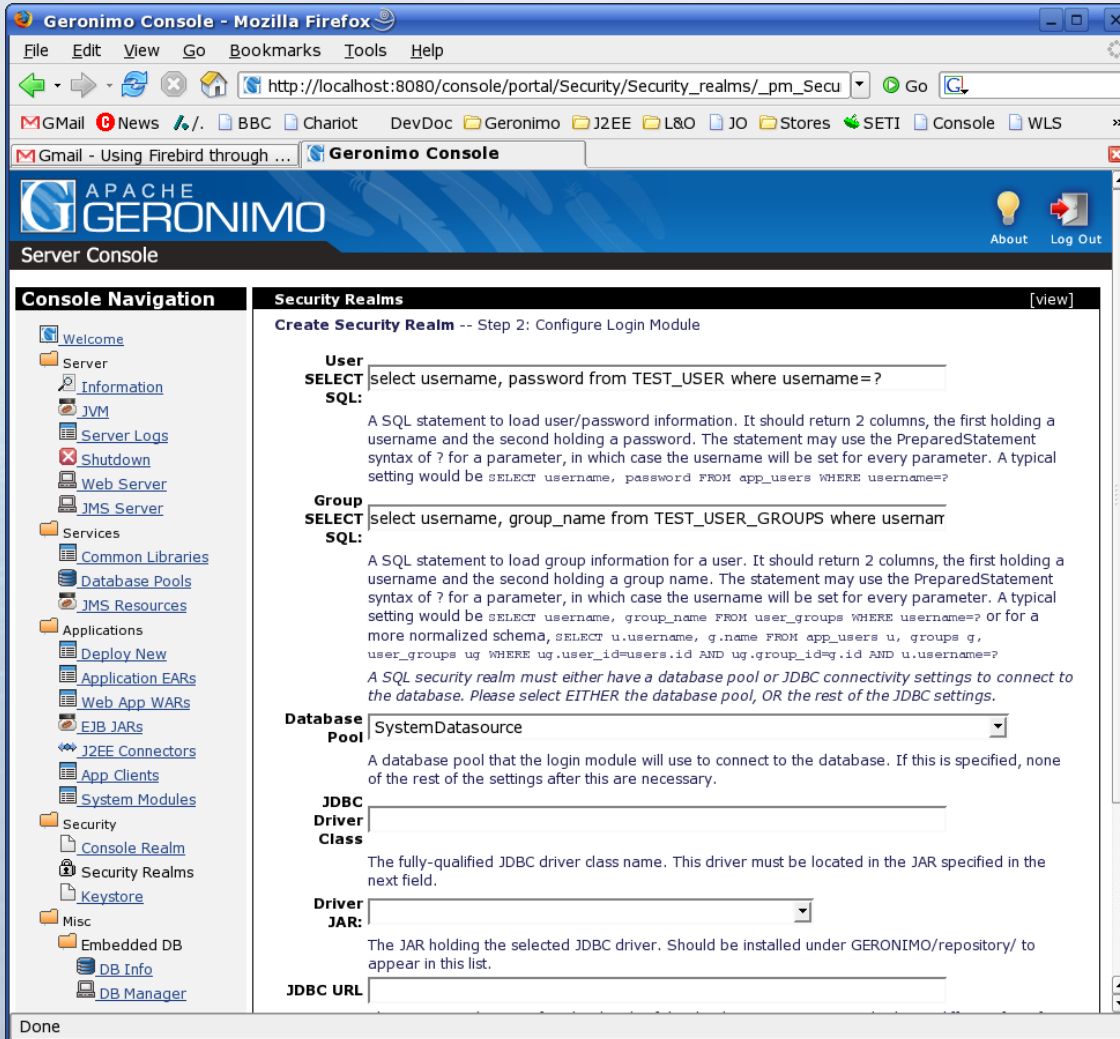
- Can deploy by hand
- Can deploy as part of an application
- Options include pool size, exception handler, etc.

## JMS Messaging Resources

The screenshot shows the Apache Geronimo Console in a Mozilla Firefox browser. The console is titled "Geronimo Console - Mozilla Firefox" and the address bar shows "http://localhost:8080/console/portal/services/services\_jms/\_rp\_services". The console interface includes a navigation pane on the left with categories like "Server", "Services", "Applications", "Security", and "Misc". The main content area is titled "JMS Resources" and shows a configuration form for a "JMS Resource Group". The form includes fields for "Connection Factory Name" (set to "AnotherConFactory"), "Transaction Support" (set to "XA"), "Pool Min Size" (set to "2"), "Pool Max Size" (set to "10"), "Blocking Timeout" (set to "5000"), and "Idle Timeout" (set to "10"). Below the form, there is a "Current Status for JMS Resource Group AaronRA" section with a list of resources: 2 Connection Factories (CustomCF, In Process) and 2 Destinations (TestQueue, TestTopic). A "Next" button is visible at the bottom of the form.

- Geronimo starts an ActiveMQ broker by default
- Can deploy JMS resources by hand or as part of an application

# Security Realms



**Console Navigation**

- Welcome
- Server
  - Information
  - JVM
  - Server Logs
  - Shutdown
  - Web Server
  - JMS Server
- Services
  - Common Libraries
  - Database Pools
  - JMS Resources
- 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

**Security Realms** [view]

Create Security Realm -- Step 2: Configure Login Module

**User**

SELECT  select username, password from TEST\_USER where username=?

**SQL:**

A SQL statement to load user/password information. It should return 2 columns, the first holding a username and the second holding a password. The statement may use the PreparedStatement syntax of ? for a parameter, in which case the username will be set for every parameter. A typical setting would be `SELECT username, password FROM app_users WHERE username=?`

**Group**

SELECT  select username, group\_name from TEST\_USER\_GROUPS where usernam

**SQL:**

A SQL statement to load group information for a user. It should return 2 columns, the first holding a username and the second holding a group name. The statement may use the PreparedStatement syntax of ? for a parameter, in which case the username will be set for every parameter. A typical setting would be `SELECT username, group_name FROM user_groups WHERE username=?` or for a more normalized schema, `SELECT u.username, g.name FROM app_users u, groups g, user_groups ug WHERE ug.user_id=users.id AND ug.group_id=g.id AND u.username=?`

A SQL security realm must either have a database pool or JDBC connectivity settings to connect to the database. Please select EITHER the database pool, OR the rest of the JDBC settings.

**Database Pool:** SystemDataSource

A database pool that the login module will use to connect to the database. If this is specified, none of the rest of the settings after this are necessary.

**JDBC Driver Class:**

The fully-qualified JDBC driver class name. This driver must be located in the JAR specified in the next field.

**Driver JAR:**

The JAR holding the selected JDBC driver. Should be installed under GERONIMO/repository/ to appear in this list.

**JDBC URL:**

- Based on JAAS LoginModules
- Can deploy by hand or as part of an application
- Default security settings in `var/security` properties files

# Keystore Manager

- Create and unlock a keystore with server's private key, trusted CA certificates, etc.

**Keystore Configuration** [\[view\]](#)

This screen lists the contents of a keystore.

Alias	Type	Certificate Fingerprint
geronimo	Private Key	A6:77:EB:E0:92:3F:33:40:82:96:00:88:CF:71:D6:63

[Add Trust Certificate](#) [Create Private Key](#) [Return to keystore list](#)

- Choose it for HTTPS web connectors

**SSL Settings**

Key Store:

The keystore to use for accessing the server's private key

Trust Store:

The keystore to use for accessing the server's private key

# Apache HTTP Configuration

- Select web applications to expose through Apache, and console generates the config files

**Apache mod\_jk Configuration** [view]

**Apache mod\_jk -- Web App Selection**

For each web application *currently running* in Geronimo, select:

**Through Apache**  
Whether the web application should be exposed through Apache

**Static Content**  
Whether Apache should serve static content for the web application (instead of all content being handled by Geronimo)

**Dynamic Paths**  
If Apache is serving static content, which URL paths should be passed to Geronimo (e.g. `/servlet/*` or `*.jsp`)

Web Application	Through Apache	Static Content	Dynamic Paths
geronimo/remote-deploy-jetty/1.1-SNAPSHOT/car	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
framework.war	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text" value="/console/portal/*"/>
standard.war	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
geronimo/welcome-jetty/1.1-SNAPSHOT/car	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

[Cancel](#)

# Deployment & Tools

# Deployment Overview

- For applications: need an archive or directory with a J2EE deployment descriptor, and typically a Geronimo deployment plan
- For services (custom configurations): need a Geronimo deployment plan (with optional JAR)
- Use the deploy tool, maven plugin, console, or hot deploy directory to deploy the module
  - Deploy tool and Maven plugin return errors and a success code to the caller; better for scripting

# Deployment Plan

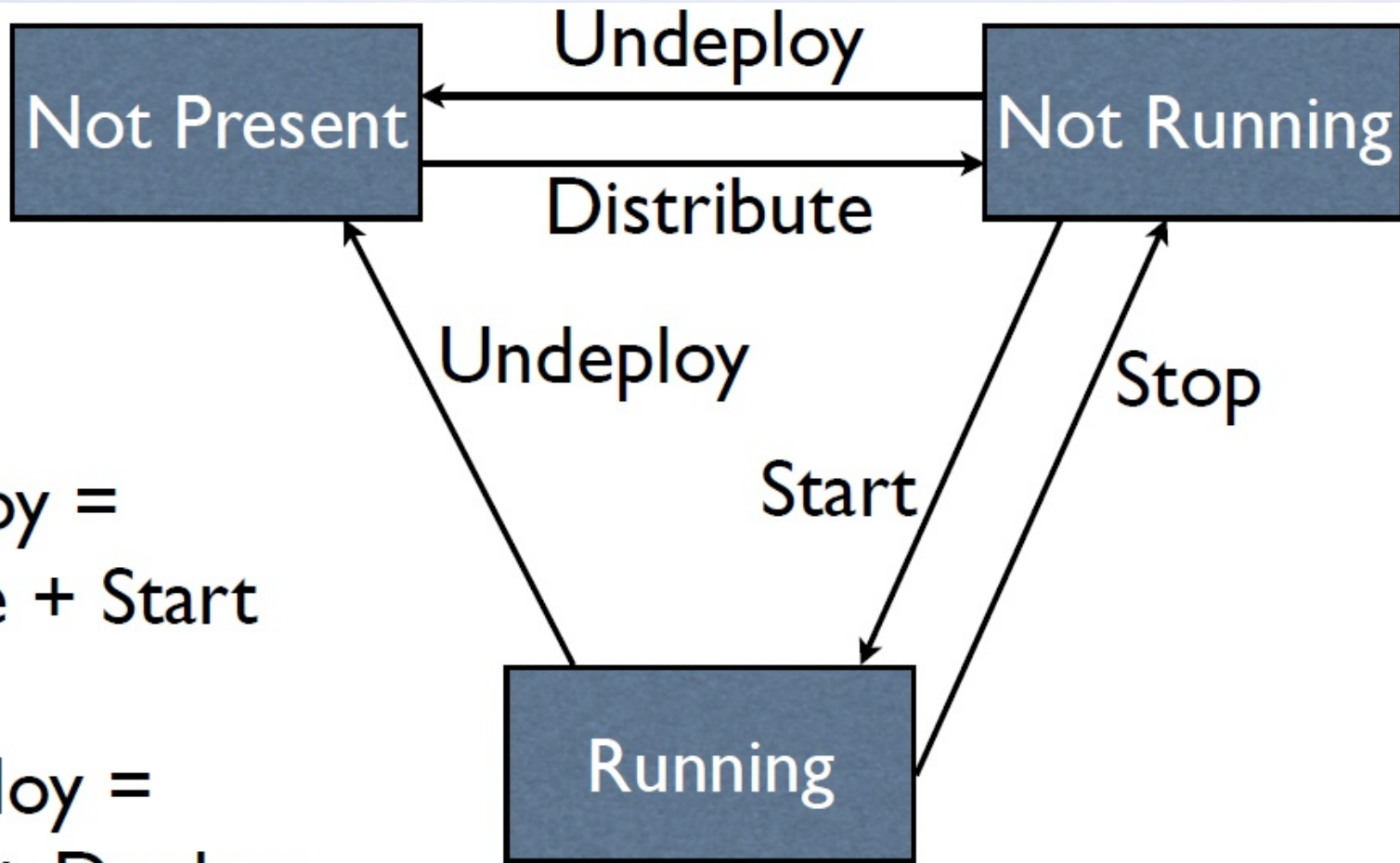
- aka “server-specific deployment descriptor”
- Geronimo plans are based on XML Schemas (normally one per module type)
- Schemas can be found in schemas/
- All plans can have a `moduleId` (a unique ID for the module) and optional `dependency` elements as well as a couple others
  - used to set up class loaders and force dependencies to start first



# Typical Deployment Plan

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1">
  <environment>
    <moduleId>
      <groupId>BigCo</groupId>
      <artifactId>TestApp</artifactId>
    </moduleId>
    <dependency>
      <artifactId>commons-collections</artifactId>
      <version>3.1</version>
    </dependency>
  </environment>
  <context-root>/debug-tool</context-root>
</web-app>
```

# Module Lifecycle



Deploy =  
Distribute + Start

Redeploy =  
Undeploy + Deploy

# Command-Line Deploy Tool

- Communicates with a running server
- Run with `java -jar bin/deployer.jar [options] command [command-options]`
- Commands include `login`, `help`, `deploy`, `undeploy`, `redeploy`, `list-modules`, `search-plugins`, `install-plugin`, etc.
- Normally prompted for a username and password (“system” and “manager”)
  - see `var/security/*.properties`

# Remote Deployment

- Deploy tool can manage and deploy to a remote server
- Need to be able to access the RMI port (1099) and an HTTP(S) port (8080)
- Must have the remote-deploy web application deployed on the server
  - It is deployed by default
- use `--host` and `--port` (or perhaps `--uri`)

# Sample Commands

- `java -jar bin/deployer.jar ...`
  - `login`
  - `distribute [archive] [plan]`
  - `deploy [archive] [plan]`
  - `undeploy configId`
  - `redeploy [archive] [plan] [configId]`
  - `stop configId`
  - `start configId`
  - `list-modules`
  - `search-plugins url`

# Hot Deploy Directory

- `geronimo/deploy/`
- Copy files to this directory to deploy
  - update file to redeploy
  - delete file to undeploy
- On startup, recognizes new and updated deployments (but will not undeploy)
- Don't try to deploy with the command-line tool and then copy a newer version into the hot deploy dir

# Maven Plugin

- Deployment plugin for Maven 1.x can start & stop server, deploy/undeploy/redeploy applications, start server and wait until it runs, etc.
- Can be included in build scripts and won't return until application is running (for subsequent testing, etc.)
- Maven 2 plugins running in the 1.2 tree, can be used today (as snapshot plugins)

# Eclipse Plugin

- Works with Eclipse WTP
- Can create Geronimo apps, including XDoclet-based EJBs, etc.
- Can run an embedded Geronimo server
- Can deploy to Geronimo
- Can debug the embedded Geronimo
  - Can debug into JSPs, etc.
- Versions available for Geronimo 1.0 and 1.1



# Debugging

- In IDEA, create a new debugging configuration and select “Remote”
- IDEA gives you a bunch of command-line parameters; start Geronimo with those  

```
java -Xdebug -Xnoagent... -jar bin/server.jar
```
- Then remote connection works perfectly
- Eclipse can run and debug Geronimo locally
- Should be able to debug both the server (if you have the source) and applications

# Little G

# Lightweight Geronimo

- 1.0 release only offers a full J2EE server configuration
- Lots of installation overhead if you just want to use it for lightweight applications
- “Little G” is a web-oriented version of Geronimo – about a 20 MB download
- Can be scaled up to JMS (or full J2EE) using additional plugins

# Little G Considerations

- Can use transactions, database pools, security, tools, etc. out of the box with Little G
- Console does not run without upgrading to a fatter stack
  - Currently depends on things like JMS
  - In a future release should be flexible enough to run with only the installed features
- Can always upgrade later (no restart!)

# Geronimo Plugins

# Geronimo Plugins

- A plugin is a packaged Geronimo module, either an application or a service module
- Can be installed by pointing the console at a plugin repository and selecting from a list, or downloading the plugin and installing from the command line
- No configuration or XML required – plugins “just work”

# Plugin Repository

- Plugins live on a web site (using the Maven 2 repository format)
- Dependencies (other plugins or JARs) are downloaded from the same or sites
- Default plugin hosting site supports plugins with any license (OSS/proprietary)
- [ibiblio.org](http://ibiblio.org) (used for many common open source projects) is the default site hosting dependency JARs

# Installing Plugins

- Geronimo checks prerequisites to ensure that the plugin can be installed
- Any obsolete modules are stopped
- Each dependency is downloaded if it's not already available to the server
  - And their dependencies, and so on...
- The plugin is installed and started, and is immediately available



# Creating Plugins

- Any application or module running in Geronimo can be exported as a plugin
- Console prompts for the necessary metadata (friendly name, license, etc.), then gives you a “save as” dialog
- Just a few things to keep in mind:
  - Declare dependencies explicitly, and don't pack them into the application if you want to share them with other apps/plugins/etc.

# Plugin Inspirations

- Geronimo uses plugin infrastructure to clone features from server to server
  - First developer gets everything set up, next developer just pulls it all down
  - Can migrate apps from dev to test, etc.
- Plugins help integrate other products
  - LDAP server, scheduler, portal, etc.
- Plugins can be used to simplify application installation

# Some Current Plugins

- JPA (from EE 5) for web applications
- Liferay portal
- ServiceMix JBI ESB
- Quartz scheduler
- File archiving
- Jasper reports
- Oracle XA driver
- Apache Directory LDAP server
- ...

# Summary

# Next Steps

- Still more powerful plugins
  - Include admin console screens/features
- Admin console for Little G
- Mini G
- More statistics & console monitors
- EJB clustering
- Improved Spring integration
- More Java EE 5 support
  - Web, Transactions, Web Service features

# Closing Thoughts

- A complete J2EE server
  - Open license and community
  - Extremely customizable (modular, w/ plugins)
- Can pack resources & services in an EAR
- Deployment, configuration, and monitoring through the web admin console
- Many additional deployment tools and plugins
- Integration available with projects like Apache Directory, ServiceMix, Liferay, Quartz, etc.

# Q&A

## Slides

- <http://chariotsolutions.com/presentations.html>

## E-Mail Lists

- [user@geronimo.apache.org](mailto:user@geronimo.apache.org)
  - [user-subscribe@geronimo.apache.org](mailto:user-subscribe@geronimo.apache.org)
- [dev@geronimo.apache.org](mailto:dev@geronimo.apache.org)
  - [dev-subscribe@geronimo.apache.org](mailto:dev-subscribe@geronimo.apache.org)

## IRC

- [#geronimo](irc://irc.freenode.net/#geronimo) on [irc.freenode.net](irc://irc.freenode.net)