Multiple Repositories

{scrollbar}

This article applies to Geronimo 2.1.

This article describes running multiple server instances from one installation instance of Geronimo, where the server instances share as much of the base Geronimo artifacts and file system space as possible. These are the geronimo directories, all sub-directories of <geronimo_home> by default:

- bin
- lib
- schema
- var
- repository
- deploy (hot deployment)

<geronimo_home> is the directory where you installed Geronimo. bin, lib and schema are read-only, and thus may be shared between instances. The disk space taken up by a typical vanilla freshly installed Geronimo instance is equally split between var and repository at almost 50 MB each, with the rest being noise. Most of var is 40 MB of ActiveMQ journal files.This motivates the idea of a second repository, in particular, one for each server instance.

In addition to a simple read/write partition of capabilities, security may well come into play as we access permissions on a set of directories. Each server instance should be able to read or write only its var directory, for instance.

Configuration Substitution Properties

A feature that helps with running multiple servers, whether or not they are sharing a repository, is the configuration substitution property facility. config.xml can have expressions of the form

${prop1 + prop2 + prop3}

etc. These expressions are evaluated with jexl using values set from:

- a properties file, by default var/config/config-substitutions.properties. The name of this file may be changed with the command line property: -Dorg.apache.geronimo.config.substitutions.file=var/config/some-other-file.properties
- system environment properties.
- command line properties.
  These override each other in the order above: command line properties override everything else. System environment properties and command line properties must be preceded by a prefix, by default "org.apache.geronimo.config.substitution." (note the final ".") You may change the prefix with the command line property
  -Dorg.apache.geronimo.config.substitution.prefix=myPrefix.

Multiple Server Instances

A server instance is easy to create in Geronimo:

1. Set the `org.apache.geronimo.server.name` system property to the instance name before you start the server.
   - Use the syntax `-Dorg.apache.geronimo.server.name=foo` to name your instance `foo`.
   - There are two ways to do this:
     a. Add this to your `GERONIMO_OPTS` environment variable, or
     b. Pass it on the java command-line invocation of the server.
   - This server's var and deploy directory will then be under `<geronimo_home>/foo`.
   - `org.apache.geronimo.server.name` may be any pathname relative to (descending from) <geronimo_home>. For example, `servers/bar` would put the server's var directory under `<geronimo_home>/servers/bar`.
   - The `org.apache.geronimo.server.dir` system property may also be used, and it overrides `org.apache.geronimo.server.name`.
   - Use `org.apache.geronimo.server.dir` to specify an absolute path, which need not be relative to <geronimo_home>. For example, `/ag20/servers/bar` would put the server's var directory under `/ag20/servers/bar`. Otherwise, the two system properties behave the same.
2. **`mkdir foo`**
3. Copy var/* to foo/var/
4. Edit `foo/var/config/config-substitutions.properties`, change `PortOffset` to a value like `1,2,10,11,12,20,21,22,...` so the ports in the new server instance will not conflict with existing server instances you already have defined and/or started. (Alternatively start the server with the property -Dorg.apache.geronimo.config.substitution.PortOffset=3 in the command line)
5. Start the server.

To deploy applications to the new server instance, you need to specify the NamingPort+PortOffset used, such as for PortOffset=1:

- deploy -port 1100 list-modules

Multiple Repositories

First, we consider the single server instance case, and just add a second repository. Say we want to leave Geronimo in its repository, but add a second repository to deploy our applications. Adding a second repository is pretty easy.

The detailed steps to install a server specific repository as specified next can be bypassed by utilizing an easy to install Geronimo Plugin for Geronimo 2.1 +. In fact, the plan displayed below was generated by building that plugin. Reference the specific steps to gain a deeper understanding or if you need to follow the manual steps to customize the result. However, if this general plan is sufficient for you just simply install the Geronimo Server Specific Repository Plugin (org.apache.geronimo.plugins/server-repo/1.0/car) and pick up the discussion on using the new repository.

1. Create a plan (say server-repo.xml) for your repository module. xmlsolidserver-repo.xml <?xml version="1.0" encoding="UTF-8"?> <!--Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.--> <!--$Rev: 665870 $ $Date: 2008-06-09 16:10:16 -0400 (Mon, 09 Jun 2008) $--> <module xmlns="http://geronimo.apache.org/xml/ns/deployment-1.2"> <!--Repository--> <environment> <moduleId> <groupId>org.apache.geronimo.plugins</groupId> <artifactId>server-repo</artifactId> <version>1.0</version> <type>car</type> </moduleId> <dependencies> <dependency> <groupId>org.apache.geronimo.framework</groupId> <artifactId>j2ee-system</artifactId> <type>car</type> </dependency> </dependencies> <hidden-classes/> <non-overridable-classes/> </environment> <gbean name="ServerRepo" class="org.apache.geronimo.system.repository.Maven2Repository"> <attribute name="root">var/repository/</attribute> <attribute name="resolveToServer">true</attribute> <reference name="ServerInfo"> <name>ServerInfo</name> </reference> </gbean> <!--Configuration Store service--> <gbean name="Local2" class="org.apache.geronimo.system.configuration.RepositoryConfigurationStore"> <reference name="Repository"> <name>ServerRepo</name> </reference> </gbean> </module>
2. Create the repository's root directory via **`mkdir <geronimo_home>/var/repository`**#* The directory is specified by the `root` attribute of the Maven2Repository GBean, `repository/` in the above example. It is a path relative to the base server directory <geronimo_home>.
    - The `resolveToServer` attribute specifies the repository's location.
        - `true` means this path is relative to baseServer, which is useful with multiple server instances.
        - `false` means this path is relative to the base directory <geronimo_home>.
3. Deploy the repository module by deploying `server-repo.xml` via **`deploy deploy server-repo.xml`**.
   The **`deploy`** command is the script <geronimo_home>/bin/deploy.{bat,sh}, depending on your operating system. It is invoked by typing simply `deploy` either from the <geronimo_home>/bin directory, or by having that directory in your path.

Using the new repository

Using the new repository is a little tricky, and is only supported from the command line currently. The essence is to use the `--targets` option of the **`deploy`** command to target your module to deploy in your new repository. First, use the deployer list-targets command to see the repositories. The target names are long and cumbersome:

`deploy list-targets`

```
Available Targets:
org.apache.geronimo.plugins/server-repo/1.0/car?ServiceModule=org.apache.geronimo.plugins/server-repo/1.0/car,j2eeType=ConfigurationStore,name=Local2
org.apache.geronimo.framework/j2ee-system/2.1.1/car?ServiceModule=org.apache.geronimo.framework/j2ee-system/2.1.1/car,j2eeType=ConfigurationStore,name=Local
```

The use of environment variables is recommended for command-line use. For example, **`set REPO2= org.apache.geronimo.plugins/server-repo/1.0/car?ServiceModule=org.apache.geronimo.plugins/server-repo/1.0/car,j2eeType=ConfigurationStore,name=Local2`**.

To deploy to the new repository, use: **`deploy deploy --targets %REPO2% sample.war`**

`deploy list-modules` also gives those long target names for each repository, followed by the usual short names of each module deployed in the repo. **`deploy list-modules %REPO2%`** gives the accustomed short output for the target repository.

The syntax to undeploy from a repository is: **`deploy undeploy "%REPO2%|geronimo/jsp-examples/1.1.1/war"`**.

Note the | character separates the repository name from the module name. The " quotes are used around the entire parameter to escape this special character from command shell interpretation.
If no `--targets` are given on the deploy command, the module is deployed to all repositories. This certainly seems undesirable. If you only want to put the module in one repository, be sure to use the `--targets` option!

Multiple Server Instances, each with its own repository(ies)

In the case of multiple server instances, a separate path may be given for the Maven2Repository root attribute for each server. This might be done with `config.xml` overrides for each server instance. A better solution is to add a `resolveToServer` attribute to this GBean, so that the `root` attribute may be set to `repository`, and it will thus be unique for each server instance with no changes required for each instance. Each server instance would then have a repository located at `<geronimo_home>/<instance_name>/repository`.

What's lacking is console support for multiple repositories, and support for hot deployment and plugins.
An interesting wrinkle here is in what repository is the new repository deployed? It's clear that the second repository (say `server-repo.xml` above) **will** be deployed in the first repository. Thus, making this first repository read-only will not work for dynamically adding repositories. Perhaps a second repository is added to contain these server-unique repositories. The second repository must be shared and read-write, so it does not fit nicely in the use case. 😒