Geronimo Tomcat Context Level Clustering - Sample Application

{scrollbar}

This web tier clustering sample applies to a Geronimo with Tomcat distribution.

Part 1 - Installing/Running the Clustering Example

Example Overview
This example demonstrates how to use context level clustering with the Tomcat web container in Geronimo. It contains the necessary attachments for setting up cluster members on two separate physical machines. The cluster configuration will allow the two cluster members to replicate httpsession data via memory to memory multicast communication. Also, a load balancer can be used to spray the incoming requests to the available cluster members. In this example, we recommend Apache HTTP server and Apache mod_jk.

"Context level" clustering has a known bug in Tomcat as reported in https://issues.apache.org/jira/browse/GERONIMO-2577 and http://issues.apache.org/bugzilla/show_bug.cgi?id=41620. Hence we recommend that you use "Host level" clustering as documented in http://cwiki.apache.org/GMOxDOC11/geronimo-tomcat-host-level-clustering-sample-application.html.

Installing the Example
This example contains 4 attachments:

- servlets-examples-cluster-node1.war - web application for Cluster Member 1
- servlets-examples-cluster-node2.war - web application for Cluster Member 2
- servlets-examples-tomcat-cluster-plan-5.5.9.xml - Geronimo deployment plan for Tomcat 5.5.9 (this is the plan to use for Geronimo v1.0)
- servlets-examples-tomcat-cluster-plan-5.5.15.xml - Geronimo deployment plan for Tomcat 5.5.15 (this is the plan to use for Geronimo V1.1)

Each geronimo cluster member must have a unique jvmRoute designation. The jvmRoute attribute allows the mod-jk load balancer to provide "sticky session" (sending all requests for the same httpsession to the same cluster member). This is possible since the load balancer places the jvmRoute value in the session cookie (or encoded url) that is returned to the web browser.

The jvmRoute attribute can be set by updating the **var/config/config.xml** file by adding the lines indicated in bold below.

<configuration name="geronimo/tomcat/1.0/car">
<gbean name="TomcatResources">
</gbean>
<gbean name="TomcatEngine">
<attribute name="initParams">
name=Geronimo
jvmRoute=nodeXYZ
</attribute>
</gbean>
<gbean name="TomcatWebConnector">
<attribute name="host">0.0.0.0</attribute>
<attribute name="port">8080</attribute>
<attribute name="redirectPort">8443</attribute>
</gbean>

Remember that the jvmRoute value must be unique for each cluster member and the server must be stopped when updating config.xml.

Now start the geronimo server (e.g. bin/geronimo.bat|sh run) and deploy the example on each of the cluster members. For this example, the applications are slightly different for each cluster member. The difference is merely to indicate the current Server number (e.g. Server 1, Server 2) in the output of the application. This will be useful when trying to determine which cluster member is servicing the http request from the browser.

Install the attached applications to the appropriate cluster member assuring that you use the correct deployment plan for each member. Note that the deployment plan must be updated with the hostname (or IP address) for each machine. The appropriate spots are identified in the plans with xx.yy.zz.aa.

Once you get the applications installed on each cluster member, you can test httpsession replication by hitting the application with your favorite browser. Probably something like: http://localhost:8080/servlets-examples-cluster/servlet/SessionExample . Note that the output page contains the ID of the server that is servicing the request. In your browser window, fill in the appropriate input fields and hit the submit button. The console dialogue (the prompt where you started geronimo) should show that that httpsession data is being transmitted and received between the cluster members. Note that the transmit /receive confirmation messages in the log are only present for Tomcat 5.5.12.

Load Balancing and failover
Now you are ready to setup the Load Balancer. We recommend using Apache HTTP server and mod_jk for this example.

Install Apache HTTP server - instructions and downloads available at http://httpd.apache.org/
Install Apache mod_jk - See http://tomcat.apache.org/tomcat-5.5-doc/balancer-howto.html

Configuration tips for mod_jk:
worker.list=loadbalancer,status
worker.node1.port=8009
worker.node1.host=*your.first.cluster.member.host.name*
worker.node1.type=ajp13
worker.node1.lbfactor=1

```
worker.node2.port=8009
worker.node2.host=your.second.cluster.member.host.name
worker.node2.type=ajp13
worker.node2.lbfactor=1

worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1
worker.status.type=status

/servlets-examples-cluster=loadbalancer
/servlets-examples-cluster/*=loadbalancer
```

Note that worker.nodeXYZ values should agree with the jvmRoute values that were used in the config.xml for each node.

Testing Load Balancing and Failover

Once you get Apache HTPP Server and mod_jk setup correctly.. You can test load balancing and failover by requesting the following URLs on port 80 (Apache HTTP Server default port).

http://Yourhost/servlets-examples-cluster - HttpSession is not used here, hence no sticky session
http://Yourhost/servlets-examples-cluster/servlet/SessionExample - HttpSession is used here, hence sticky session should be in effect

You can test failover by stopping the geronimo server that owns the sticky session and seeing that the next http request will failover into the remaining cluster member. The httpsession data from the previous request should be recovered and displayed in the refreshed browser window.

Tips

- When testing using a web browser, make sure that you erase cookies and cached pages between test cases. Browser caching can cause confusion when testing.
- Make sure your application has the distributable attribute defined in web.xml
- Memory to memory replication currently requires that **all cluster members must reside on the same physical subnet** since multicast broadcast is used. Make sure all cluster members are on the same physical subnet and that multicast broadcast is supported on the subnet.

Also, see http://tomcat.apache.org/tomcat-5.0-doc/cluster-howto.html for more information on tomcat clustering.

Special Acknowledgement to Jeff Genender for developing and supporting the Tomcat clustering GBeans for Geronimo!!

Back Button Sample applications