# Tomcat clustering

## Setting Up Tomcat to Test Wicket Clustering

**NOTE:** This is only one of the many ways to set up clustering. This approach turns the entire Tomcat into a clustered one. It is also possible to set up multiple virtual hosts, with one host holding non-clustered applications and one having the clustered apps. See the descriptions in the Tomcat conf/web.xml for more details. The description below is how to set up the SIMPLEST, but not the BEST!

### Get Tomcat

Get a copy of Tomcat. I would recommend 5.0.30 as this is the version that I have tested all of the config files on. You could also try 5.5.x, but the config may be different. The Tomcat clustering implementation is almost identical between the two versions, so if you get it working on one then you should also be able to get it working on the other.

**IMPORTANT: The current releases of Tomcat have a clustering bug that causes Wicket to fail.** There is a fix for this in the Tomcat CVS but this has not made it into the current production builds. Your options are therefore:

1. Build a tomcat from CVS HEAD
2. Download the tomcat src and patch the `DeltaRequest.java` file so that the `execute(DeltaSession)` method ends with:

   ```
   reset();
   session.endAccess();
   ```

   as opposed to just:

   ```
   session.endAccess();
   ```

3. Download a modified catalina-cluster.jar from my website (http://www.templemore.co.uk/catalina-cluster.jar). This contains the patch plus some useful debug output so that you can see what is going on!

### Install Tomcat

Install tomcat either via the installer, or unpack into a particular directory.
DELETE the folowing unecessary stuff (for faster startup and to stop error messages):

```
conf/Catalina/localhost/balancer.xml
webapps/balancer''
webapps/jsp-examples''
webapps/servlets-examples''
webapps/tomcat-docs''
webapps/webdav''
```

### Configure Tomcat

Edit the conf/server.xml file so that it contains the following content (minimum required for a clustered Tomcat server). This creates a basic clustered server using synchronous replication of session attributes (e.g. request does not complete until any session attribute changes have synchronised to all servers). This is not the most performant solution, but it is the best for testing and debugging.

NOTE: All of the settings surrounded by # characters need to be changed for EACH SERVER IN THE CLUSTER (remove the # characters!). Each server should have a different value. Settings in surrounded by @ characters may need to be changed, but they must be identicalon every server (remove the @ characters!).

```
<Server port="#11005#" shutdown="SHUTDOWN" debug="0">

        <!-- Global JNDI resources -->
        <GlobalNamingResources>
                <!--
                        Editable user database that can also be used by
                             UserDatabaseRealm to authenticate users
                -->
                <Resource name="UserDatabase" auth="Container"
                          type="org.apache.catalina.UserDatabase"
                          description="User database that can be updated and saved">
                </Resource>
```

```
                <ResourceParams name="UserDatabase">
                        <parameter>
                                <name>factory</name>
                                <value>org.apache.catalina.users.MemoryUserDatabaseFactory</value>
                        </parameter>
                        <parameter>
                                <name>pathname</name>
                                <value>conf/tomcat-users.xml</value>
                        </parameter>
                </ResourceParams>
        </GlobalNamingResources>


        <!-- The Tomcat Service -->
        <Service name="Catalina">
                <!-- HTTP/1.1 Connector on port 11080 -->
                <Connector port="#11080#"
                           maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
                           enableLookups="false" redirectPort="#11443#" acceptCount="100"
                           debug="0" connectionTimeout="20000"
{panel}
                           disableUploadTimeout="true" />
{panel}

                <!-- Define a Coyote/JK2 AJP 1.3 Connector on port 11009 -->
                <Connector port="#11009#"
                           enableLookups="false" redirectPort="#11443#" debug="0"
                           protocol="AJP/1.3" />

                <!-- Define the top level container in our container hierarchy -->
                <Engine name="Catalina" defaultHost="localhost" debug="0" jvmRoute="#serverA#">
                        <!-- Global logger unless overridden at lower levels -->
                        <Logger className="org.apache.catalina.logger.FileLogger"
{panel}
                                     prefix="catalina_log." suffix=".txt"
                                     timestamp="true"/>
{panel}

                        <!-- Real used for security purposes -->
                        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
                               debug="0" resourceName="UserDatabase"/>

{panel}
                        <!-- Define a clustered host for the Web Tools application --
>
{panel}
                        <Host name="localhost" debug="0" appBase="webapps"
                              unpackWARs="true" autoDeploy="true"
                              xmlValidation="false" xmlNamespaceAware="false">
                                <!-- Define the cluster configuration -->
                                <Cluster className="org.apache.catalina.cluster.tcp.SimpleTcpCluster"
                                         name="wicket-examples"
                                         managerClassName="org.apache.catalina.cluster.session.DeltaManager"
                                         expireSessionsOnShutdown="false"
                                         useDirtyFlag="true">

                                        <Membership
                                                className="org.apache.catalina.cluster.mcast.McastService"
                                                mcastAddr="@228.0.0.4@"
                                                mcastPort="<at:var at:name="45564" />"
                                                mcastFrequency="500"
                                                mcastDropTime="3000"/>

                                        <Receiver
                                                className="org.apache.catalina.cluster.tcp.ReplicationListener"
                                                tcpListenAddress="auto"
                                                tcpListenPort="#11901#"
                                                tcpSelectorTimeout="100"
                                                tcpThreadCount="2"/>

                                        <Sender
```

```
                                        className="org.apache.catalina.cluster.tcp.
ReplicationTransmitter"
                                        replicationMode="synchronous"/>

                                <Valve
                                        className="org.apache.catalina.cluster.tcp.ReplicationValve"
                                        filter=".*\.gif;.*\.js;.*\.jpg;.*\.htm;.*\.html;.*\.txt;.*\.css"
/>
                            </Cluster>

                            <!-- Define a logger for this host -->
                            <Logger className="org.apache.catalina.logger.FileLogger"
                                    directory="logs"  prefix="localhost_log." suffix=".txt"
                                    timestamp="true"/>

                    </Host>
            </Engine>
        </Service>
</Server>
```

## Install Wicket Examples

Modify the wicket-examples web.xml file to add a `<distributable/>` entry directly after the `<display-name>` entry.

Copy the wicket-examples.war file into the webapps directory.

## Repeat For All Servers

Repeat the above stages for all servers, making sure that you change the port numbers and jvmroute so that each server has a different value.

## Start The Servers

Start each server in turn. You should see each attempt to establish cluster membership for the wicket-examples application and to detect other servers as they come up.

You can test the wicket examples directly by accessing any server in the cluster. The URL will be http://localhost:XXXXX/wicket-examples where XXXXX is the port you set for the HTTP connector in the web.xml.

If you used my modified catalina-cluster.jar file you should see session attributes synchronising across the cluster as to navigate through the pages.

## Round-Robin Clustering Using Apache

Setup an apache 2.0.x server (I'll assume you already know how to do this!). Download and add the mod_jk2 apache module into the modules directory.

Add the following entries to conf\httpd.conf file:

```
# Add support for Tomcat integration
<IfModule mod_jk2.c>
{panel}
    #-----------------------------------------------
    # Where to find the workers2.properties file
    #-----------------------------------------------
    #
    JkSet config.file conf/workers2.properties
{panel}
</IfModule>
```

Create a workers2.properties file in the conf directory, replacing all of the ports and jvm route values with the ones you configured in the server.xml files.

```
[shm]
info=Scoreboard. Requried for reconfiguration and status with multiprocess servers.
file=anon

# Defines a load balancer named lb. Use even if you only have one machine.
[lb:lb]
stickySession=0

# Channel to cluster tomcat: serverA
[channel.socket:chturner-1:11009]
tomcatId=serverA

# Channel to cluster tomcat: serverB
[channel.socket:chturner-1:12009]
tomcatId=serverB

# Worker for cluster tomcat: serverA
[ajp13:chturner-1:11009]
channel=channel.socket:chturner-1:11009
group=lb

# Worker for cluster tomcat: serverB
[ajp13:chturner-1:12009]
channel=channel.socket:chturner-1:12009
group=lb

# Map the wocket-examples webapp to the Web server uri space
[uri:/wicket-examples/*]
group=lb

[status:]
info=Status worker, displays runtime information

[uri:/jkstatus/*]
info=The Tomcat /jkstatus handler
group=status:
```

The above configures Apache to loadbalance using a round-robin approach.