

**server.xml** ..... 3



— 2015/08/03 16:13

# server.xml

Apache Tomcat 8.5.x

```
<?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.
-->
<!-- Note: A "Server" is not itself a "Container", so you may not
define subcomponents such as "Valves" at this level.
Documentation at /docs/config/server.html
-->
<Server address="${tomcat.address.management}"
port="${tomcat.port.management}" shutdown="SHUTDOWN">
  <Listener className="org.apache.catalina.startup.VersionLoggerListener" />
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener"
SSLEngine="on" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener
className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener
className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <Listener
className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

  <!-- Global JNDI resources
Documentation at /docs/jndi-resources-howto.html
-->
  <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"
          factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
          pathname="conf/tomcat-users.xml" />
</GlobalNamingResources>

<!-- A "Service" is a collection of one or more "Connectors" that share
      a single "Container" Note: A "Service" is not itself a "Container",
      so you may not define subcomponents such as "Valves" at this level.
      Documentation at /docs/config/service.html
-->
<Service name="Catalina">

    <!--The connectors can use a shared executor, you can define one or more
    named thread pools-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
              maxThreads="150" minSpareThreads="4"/>
    -->

    <!-- A "Connector" represents an endpoint by which requests are received
    and responses are returned. Documentation at :
    Java HTTP Connector: /docs/config/http.html
    Java AJP Connector: /docs/config/ajp.html
    APR (HTTP/AJP) Connector: /docs/apr.html
    Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector address="${tomcat.address.http}" port="${tomcat.port.http}"
    protocol="HTTP/1.1"
               connectionTimeout="20000"
               maxParameterCount="1000"
               allowTrace="false"
               server="${tomcat.connector.server}"
               maxPostSize="${tomcat.connector.maxPostSize}"
               maxThreads="${tomcat.connector.maxThreads}"
               maxParameterCount="1000"
               redirectPort="${tomcat.port.https}"
               URIEncoding="${tomcat.connector.URIEncoding}"
               useBodyEncodingForURI="true" />

    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

```

-->
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation. The default
SSLImplementation will depend on the presence of the APR/native
library and the useOpenSSL attribute of the
AprLifecycleListener.
Either JSSE or OpenSSL style configuration may be used regardless
of
the SSLImplementation selected. JSSE style configuration is used
below.
-->
<!--
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" SSLEnabled="true">
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
            type="RSA" />
    </SSLHostConfig>
</Connector>
-->
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
This connector uses the APR/native implementation which always uses
OpenSSL for TLS.
Either JSSE or OpenSSL style configuration may be used. OpenSSL
style
configuration is used below.
-->
<!--
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11AprProtocol"
    maxThreads="150" SSLEnabled="true" >
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol"
/>
    <SSLHostConfig>
        <Certificate certificateKeyFile="conf/localhost-rsa-key.pem"
            certificateFile="conf/localhost-rsa-cert.pem"
            certificateChainFile="conf/localhost-rsa-chain.pem"
            type="RSA" />
    </SSLHostConfig>
</Connector>
-->

<!-- Define an AJP 1.3 Connector on port 8009 -->
<!--
<Connector protocol="AJP/1.3"
    address="::1"
    port="8009"
    redirectPort="8443" />
-->
<Connector address="${tomcat.address.ajp}" port="${tomcat.port.ajp}"

```

```
protocol="AJP/1.3"
    maxParameterCount="1000"
    allowTrace="false"
    server="${tomcat.connector.server}"
    maxPostSize="${tomcat.connector.maxPostSize}"
    maxThreads="${tomcat.connector.maxThreads}"
    maxParameterCount="1000"
    redirectPort="${tomcat.port.https}"
    URIEncoding="${tomcat.connector.URIEncoding}"
    secretRequired="false"
    useBodyEncodingForURI="true" />

<!-- An Engine represents the entry point (within Catalina) that
processes
every request. The Engine implementation for Tomcat stand alone
analyzes the HTTP headers included with the request, and passes
them
on to the appropriate Host (virtual host).
Documentation at /docs/config/engine.html -->

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
-->
<Engine name="Catalina" defaultHost="localhost"
jvmRoute="${tomcat.engine.jvmRoute}">

<!--For clustering, please take a look at documentation at:
/docs/cluster-howto.html (simple how to)
/docs/config/cluster.html (reference documentation) -->
<!--
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
-->

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"
channelSendOptions="6" channelStartOptions="3">

<!-- Delta Manager -->
<Manager className="org.apache.catalina.ha.session.DeltaManager"
expireSessionsOnShutdown="false"
notifyListenersOnReplication="true"/>

<!-- Backup Manager
<Manager className="org.apache.catalina.ha.session.BackupManager"
expireSessionsOnShutdown="false"
notifyListenersOnReplication="true"
mapSendOptions="6"/>
-->

<Channel
className="org.apache.catalina.tribes.group.GroupChannel">
```

```
        <!-- Multicast Member -->
        <!--
        <Membership
className="org.apache.catalina.tribes.membership.McastService"
        address="${tomcat.cluster.member.address}"
        port="${tomcat.cluster.member.port}"
        frequency="500"
        dropTime="3000"/>
    -->

    <Receiver
className="org.apache.catalina.tribes.transport.nio.NioReceiver"
        address="${tomcat.cluster.receiver.address}"
        port="${tomcat.cluster.receiver.port}"
        selectorTimeout="100"
        maxThreads="6"/>

    <Sender
className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
        <Transport
className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>
    </Sender>
    <Interceptor
className="org.apache.catalina.tribes.group.interceptors.TcpPingInterceptor"
/>
    <Interceptor
className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"
/>
    <Interceptor
className="org.apache.catalina.tribes.group.interceptors.MessageDispatchInte
rceptor"/>
    <Interceptor
className="org.apache.catalina.tribes.group.interceptors.ThroughputIntercept
or"/>

    <Interceptor
className="org.apache.catalina.tribes.group.interceptors.StaticMembershipInt
erceptor">
        <LocalMember
className="org.apache.catalina.tribes.membership.StaticMember"
            domain="staging-cluster"
uniqueId="{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,1}"/>

        <Member
className="org.apache.catalina.tribes.membership.StaticMember"
            port="5001"
            securePort="-1"
            host="tomcat2"
            domain="staging-cluster"
            uniqueId="{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,2}"/>
    </Interceptor>
```

```
</Channel>

    <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"
filter=".*\.gif|.*\js|.*\jpeg|.*\jpg|.*\png|.*\htm|.*\html|.*\css|.*\
.txt"/>
    <Valve
className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>

    <!--
    <Deployer
className="org.apache.catalina.ha.deploy.FarmWarDeployer"
        tempDir="/tmp/war-temp/"
        deployDir="/tmp/war-deploy/"
        watchDir="/tmp/war-listen/"
        watchEnabled="false"/>
    -->

    <ClusterListener
className="org.apache.catalina.ha.session.ClusterSessionListener"/>
</Cluster>

<!-- Use the LockOutRealm to prevent attempts to guess user passwords
via a brute-force attack -->
<Realm className="org.apache.catalina.realm.LockOutRealm">
    <!-- This Realm uses the UserDatabase configured in the global JNDI
resources under the key "UserDatabase". Any edits
that are performed against this UserDatabase are immediately
available for use by the Realm. -->
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>
</Realm>

<Host name="localhost" unpackWARs="true" autoDeploy="true"
createDirs="true"
    appBase="${tomcat.engine.localhost.appBase}"
    workDir="${tomcat.engine.localhost.workDir}">

    <!-- SingleSignOn valve, share authentication between web
applications
Documentation at: /docs/config/valve.html -->
    <!--
    <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
    -->

    <!-- Access log processes all example.
Documentation at: /docs/config/valve.html
Note: The pattern used is equivalent to using pattern="common"
-->
    <Valve className="org.apache.catalina.valves.AccessLogValve"
directory="logs"
        prefix="localhost_access_log" suffix=".txt"
```



```
pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

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

From:

<https://atl.kr/dokuwiki/> - AllThatLinux!

Permanent link:

[https://atl.kr/dokuwiki/doku.php/conf\\_server.xml.cluster](https://atl.kr/dokuwiki/doku.php/conf_server.xml.cluster)

Last update: **2024/01/12 02:09**

