Author : Chris Drawater
Date : Jan 2006
Version : 1.1

Apache 2.0, Tomcat 5.5, WARs & PostgreSQL 8.1 JDBC DataSources on Windows XP

Document Status

Introduction

Tomcat is the official Reference Implementation for the SUN Java Servlet and JSP technologies and has support for servlets, JSPs, clustering, load balancing, JNDI, JDBC, SSL, JMX, JAAS. Version 5.5 implements the Servlet 2.4 and JSP 2.0 specifications.

The Tomcat container is open source and can be downloaded, distributed and deployed for free – there are no licence, support or maintenance costs.

This paper documents the setup of an Apache/Tomcat/JDBC development environment on Windows XP and the deployment of an Application WAR file – it is published for R&D/information purposes only.

The principles should be easily transferable to Unix or Linux and for that reason, some of the full file paths have been replaced by paths using the pseudo-environmental variables APACHE2_HOME & TOMCAT HOME.

Technology

This paper is based around the following technologies :

Running on Windows XP Pro SP2
- JDK 1.5.0
- Apache 2.0.55
- Tomcat 5.5.15
- Connector : Apache Tomcat JK 1.2.15 for WIN32 – works with Apache 2.0.55 and later
- PostgreSQL JDBC driver *postgresql-8.1-404.jdbc3.jar*

On Solaris 10
- PostgreSQL 8.1.1

For demonstrative purposes, 'vAuth' is used as the name of the application.

Miscellaneous Concepts and Terminology

A Tomcat **worker** is a Tomcat instance that is waiting to execute servlets or JSPs .

The Tomcat **Context** represents a *web application.*

The Apache Tomcat Database Connection Pool (**DBCP**) uses the Apache Jakarta-Commons Database Connection Pool (see http://tomcat.apache.org/tomcat-5.5-doc/index.html)

The **connector *mod_jk*** is an Apache module that effectively acts as a 'router' to Tomcat, passing servlets/JSP requests. It can be configured to provide load balancing , session 'stickiness' and fault tolerance across Tomcat worker instances if required.

The process architecture is basically as follows :

Apache 2.0 --- *mod_jk* module --- AJP13 protocol --- (1..n) * Tomcat 5.5 instance(s)

Download Apache & Tomcat Binaries

Download the following distributions from http://www.apache.org

apache_2.0.55-win32-x86-no_ssl
apache-tomcat-5.5.15
mod_jk-apache-2.0.55.so

Setup

In this example setup, all Apache software is installed under
C:\Program Files\Apache Group
ie
C:\Program Files\Apache Group\Tomcat 5.5 *# aka TOMCAT_HOME*
C:\Program Files\Apache Group\Apache2 *# aka APACHE2_HOME*

(1) Install Apache http server via GUI
Domain → localhost
Server → localhost
Run as service on port 80 (port can be changed later)
Test by invoking http://localhost

(2) Install Tomcat via GUI
Run as service
Test by invoking http://localhost:8080
backup then edit file *TOMCAT_HOME\conf\tomcat-users.xml*, to include the line :
<user username="manager" password="manager" roles="manager"/>
Test using http://localhost:8080/manager/status

(3) Install *mod_jk* Connector - rename *mod_jk-apache-2.0.55.so* to *mod_jk.so* and move to the Apache2 modules directory
APACHE2_HOME\modules

(4) For safety, backup files
APACHE2_HOME\conf\httpd.conf
TOMCAT_HOME\conf\server.xml

(5) In the http server configuration file *APACHE2_HOME\conf\httpd.conf* , add the following lines

```
# For Tomcat 5.5 with mod_jk
#

# Update this path to match your modules location
LoadModule    jk_module  modules/mod_jk.so

# location of  workers.properties
JkWorkersFile conf\workers.properties

# jk logs
JkLogFile     logs\mod_jk.log

# jk log level [debug/error/info]
JkLogLevel    info

# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "

# JkOptions indicate to send SSL KEY SIZE,
JkOptions     +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat set the request format
JkRequestLogFormat     "%w %V %T"

# Send servlet for context /servlets-examples to worker named worker1
JkMount  /*/servlet/* worker1

# Send JSPs  for context /jsp-examples to worker named worker1
JkMount  /*.jsp worker1
```

Optionally change the http listener port by changing line

```
Listen 80
```

to, for example

```
Listen 9999
```

(6)To let the Connector module know where to pass on servlet/JSP requests, create file *APACHE2_HOME\conf\worker.properties* containing the following entries :

```
# Define 1 Tomcat instance
worker.list=worker1

# Set properties for worker1 instance
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=50
worker.worker1.cachesize=10
worker.worker1.cache_timeout=600
worker.worker1.socket_keepalive=1
worker.worker1.recycle_timeout=300
```

(7) Shutdown Apache
Restart Tomcat
Startup Apache

Tomcat Configuration - PostgreSQL JDBC DataSources

To configure a PostgreSQL DataSource specific to an application (ie not defined globally) create a *context.xml* file containing :

```
<Resource
        auth="Container"
        description="vAuth Postgresql DB Connection"
        name="jdbc/vAuthDS"
        type="javax.sql.DataSource"

        username="xyz"
        password="xyz"
        driverClassName="org.postgresql.Driver"
        url="jdbc:postgresql://10.248.42.122:5432/db9"

        initialSize="3"
        maxActive="10"
        maxIdle="5"
        minIdle="3"
        maxWait="5000"

        validationQuery=""
        poolPreparedStatements="false"
/>
```

JDBC DataSource Usage Example

A very simple example of application code acquiring a pooled database *Connection* object via a *DataSource* using a JNDI lookup is shown below :

```
String dsString = "java:/comp/env/jdbc/vAuthDS";

Context ic = new InitialContext();
DataSource ds = (DataSource) ic.lookup(dsString);

Connection con = ds.getConnection();
```

Application WAR deployment

(1) Copy all required JDBC driver jar files → *TOMCAT_HOME\common\lib*

For PostgreSQL 8.1,
postgresql-8.1-404.jdbc3.jar

Copy any shared application libraries (not visible to Tomcat internal code) →
TOMCAT_HOME\shared\lib

(2) To enable Apache to pass servlet/jsp requests onto Tomcat but have Apache serve up static content such as html, gifs etc, insert Connector directives similar to below

```
# Send servlet & JSP requests to Tomcat instance worker1
JkMount  /vAuth/servlet/* worker1
JkMount  /vAuth/*.jsp worker1

# Let apache serve up static content (html etc)
JkAutoAlias "C:\Program Files\Apache Group\Tomcat 5.5\webapps"
```

into the Apache server configuration file *APACHE2_HOME\conf\httpd.conf*
Restart Apache.

(3) To define any JNDI referenced JDBC DataSources & other resources used by the application, create an application specific file *context.xml* (as per above) under META-INF (alongside WEB-INF) in the WAR .
See http://tomcat.apache.org/tomcat-5.5-doc/config/context.html for further information.

The hierarchical application WAR directory tree should look something like this :

<app root>	
<app root>/.jsp*	*files*
<app root>/.html*	*files*
<app root>/.gif*	*files*
<app root>/.jsp*	*files*
<app root>/WEB-INF	*dir*
*<app root>/WEB-INF/**web.xml***	*file*
<app root>/WEB-INF/classes	*dir*
<app root>/WEB-INF/lib	*dir*
<app root>/WEB-INF/.jar*	*files*
<app root>/META-INF	*dir*
*<app root>/META-INF/**context.xml***	*file*

To enable the application to reference the DataSource, a resource XML entry (matching the DataSource defined in *context.xml*) must be placed in the application *web.xml* file – for example :

```
<resource-ref>
        <description>vAuth Datasource</description>
        <res-ref-name>jdbc/vAuthDS</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
</resource-ref>
```

(4) Create & copy the application WAR file into directory *TOMCAT_HOME\webapps* and tomcat will automatically deploy it! It is important to note that the WAR name should **match** the (application = context name).

Deployment Diagnostics

To log the contents (headers, footers, parameters, cookie contents etc) of requests passed to Tomcat, uncomment the following lines

<Valve className="org.apache.catalina.valves.RequestDumperValve"/>

in file

TOMCAT_HOME\conf\server.xml

which will log the info into file

TOMCAT_HOME\logs\ catalina.<DATE>.txt

To check that the application context is correct and to log access to the application, add the following XML

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
 prefix="vAuth_access_log." suffix=".txt"
 pattern="common"/>
```

into the application *context.xml.*

Concluding Remarks

This paper demonstrates, for R&D purposes, the basics for deploying an application WAR into an Apache/Tomcat development environment.

Chris Drawater has been working with RDBMSs since 1987 and the JDBC API since late 1996.