INAP
INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

Curso online: **Instalación, Configuración y Administración de Apache + Tomcat**

Módulo 4. Cooperación entre Apache y Tomcat

Capítulo 3. mod_jk (conector JK 1.2) - Actividad 3 - Prácticas

Autores

Janine García Morera

Alexandra López de la Oliva Portugués

Julio Villena Román

Octubre de 2014

PRÁCTICAS

1. Configuración y uso de mod_jk

Este ejercicio sirve de ilustración del proceso de despliegue del módulo mod_jk para conectar Apache y Tomcat, configurando en concreto dos servidores virtuales, uno de ellos con soporte SSL.

El ejemplo está desarrollado para servidores Unix, así que en Windows habría que adaptar las rutas como fuera necesario.

Realiza la secuencia de pasos siguientes:

1. Editar el fichero `workers.properties`:

```
#
workers.tomcat_home=/usr/local/jakarta-tomcat
workers.java_home=/usr/java
#
# En Unix, las barras a la derecha:
ps=/
#
worker.list=ajp12, ajp13
#
# Definición del worker Ajp13
#
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
```

2. Editar el fichero `httpd.conf`

```
# Cargar mod_jk
#
LoadModule jk_module modules/mod_jk.so
# Configurar mod_jk
#
JkWorkersFile                                /usr/local/jakarta-
tomcat/conf/jk/workers.properties
JkLogFile /usr/local/apache/logs/mod_jk.log
JkLogLevel info

# Primer host virtual
#
<VirtualHost 10.0.0.1:80>
      DocumentRoot /web/host1
      ServerName host1.apache.org
      JkMount /*.jsp ajp13
```

```
      JkMount /servlet/* ajp13
</VirtualHost>

# Segundo Host Virtual. Accessible también via HTTPS
#
<VirtualHost 10.0.0.2:80>
      DocumentRoot /web/host2
      ServerName host2.apache.org
      JkMount /*.jsp ajp13
      JkMount /servlet/* ajp13
</VirtualHost>

<VirtualHost 10.0.0.2:443>
      DocumentRoot /web/host2
      ServerName host2.apache.org
      SSLEngine On
      JkMount /*.jsp ajp13
      JkMount /servlet/* ajp13
</VirtualHost>
```

Analiza estos ficheros de configuración y responde a las preguntas siguientes:

- ¿Cuáles son las direcciones y los puertos de los servidores involucrados?
- ¿Cuál sería la URL de acceso a la información (servidor y ruta completa)?

2. Configurar Apache para que sirva los servlets de ejemplo de Tomcat usando mod_jk

Este ejercicio ilustra cómo configurar Apache para que sirva los servlets de ejemplo de la instalación de Tomcat, empleando mod_jk.

Este ejemplo ha sido desarrollado para Windows. Ojo con las rutas, que puede que no coincidan con la instalación que tienes en tu máquina.

Realiza la siguiente secuencia de operaciones:

1. Cerrar Apache y Tomcat
2. Generar el fichero mod_jk.conf con un aspecto similar a este en el directorio `$TOMCAT_HOME/conf`)..

```
<IfModule !mod_jk.c>
      LoadModule jk_module "modules/mod_jk.dll"
</IfModule>

<VirtualHost localhost>
      ServerName localhost
      JkMount /examples ajp13
      JkMount /examples/* ajp13
      JkMount /docs ajp13
      JkMount /docs/* ajp13
      JkMount /manager ajp13
```

```
      JkMount /manager/* ajp13
      JkMount /host-manager ajp13
      JkMount /host-manager/* ajp13
</VirtualHost>
```

Como se ve, se crean todos los puntos de montaje de todas las webapps que están definidas en este momento y las asigna a un worker llamado ajp13. La carga del módulo se realiza usando una dll en lugar de un fichero `.so`. Se trata de un problema de actualización en Tomcat, irrelevante si el módulo se carga correctamente en el fichero `httpd.conf`.

3. Crear en el directorio $APACHE_HOME/conf el fichero workers.properties con la definición del worker ajp13:

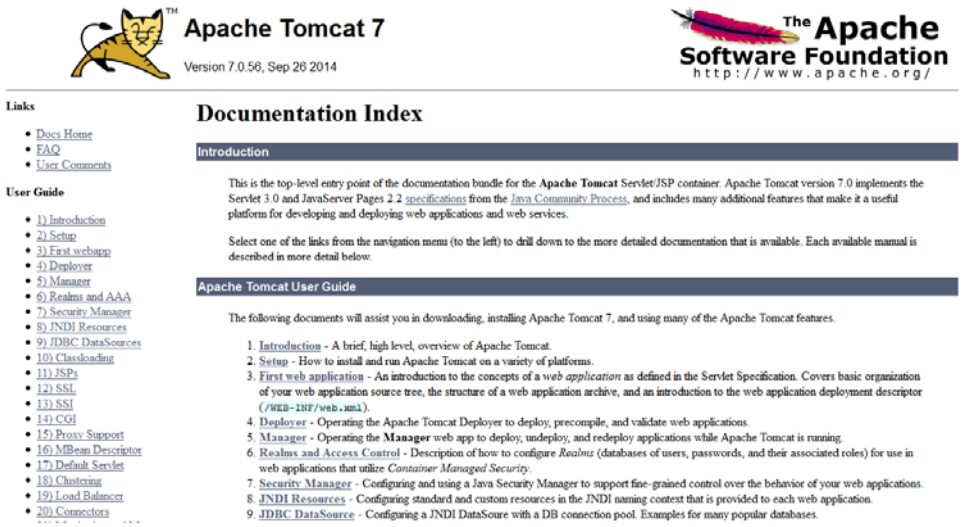
```
#
workers.tomcat_home=C:\\Curso\\Tomcat
workers.java_home=C:\\Archivos de programa\\Java\\jdk
# En Windows, las barras a la izquierda:
ps=\\

#
worker.list=ajp13
# Definicion para el worker Ajp13
#
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
```

4. Hacer una copia del fichero `httpd.conf` en el directorio `$APACHE_HOME/conf`. Modificar el fichero `httpd.conf` de apache para que cargue el módulo `mod_jk` y el fichero de configuración `mod_jk.conf`, y para definir las directivas de `mod_jk`.

```
LoadModule jk_module modules/mod_jk.so
#
# Conector mod_jk
Include "C:\\Curso\\Tomcat\\conf\\mod_jk.conf"
JkWorkersFile conf\workers.properties
JkLogFile    logs\mod_jk.log
JkLogLevel    info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
```

5. Levantar Apache con el monitor de Apache.

6. Para comprobar si la redirección funciona correctamente, abrir un navegador y acceder a cualquier página de Tomcat a través del puerto 80, por ejemplo `http://localhost/docs`, que debería presentar la página de documentación local de Tomcat.



7. De igual manera, se puede acceder a la página de ejemplos usando la URL `http://localhost/examples/`. También podemos ejecutar los distintos ejemplos comprobando que los accesos funcionan correctamente y se realizan a través del puerto 80.

Otra consideración es que esta misma página se podría alcanzar accediendo mediante el puerto por defecto Tomcat, el 8080, puesto que Tomcat sigue escuchando por dicho puerto. Se puede modificar el fichero `server.xml` para que Tomcat sólo pueda ser accesible a través de Apache, evitando que Tomcat escuche en el puerto 8080, comentando el conector HTTP y dejando activo el conector AJP1.3. Hazlo y comprueba que es así.

Haz un post en el foro de actividades con tus impresiones. Comenta cualquier aspecto que consideres conveniente relacionado con el ejercicio. ¿Qué dificultades has encontrado? ¿Te parece cómoda la forma de Tomcat y Apache para definir la interacción empleando `mod_jk`?

3. Ejemplo de balanceo de carga usando mod_jk.

Este ejercicio ilustra cómo balancear con Apache entre dos instancias Tomcat, en cada una de las cuales se ejecutará una página JSP diferente.

Si se dispone de dos servidores conectados en red, el ejercicio se puede realizar instalando una instancia en cada servidor. Si sólo se dispone de un servidor, se pueden instalar dos instancias escuchando por puertos distintos.

En anteriores ejercicios ya habías pensado cómo desplegar más de una instancia de Tomcat. Aquí se explica concretamente la forma de realizarlo en un sistema Windows. Se asume que la instancia principal está instalada en `C:\Curso\Tomcat1` y la copia en `C:\Curso\Tomcat2`.

1. Crear una copia de la carpeta `C:\Curso\Tomcat1` en el directorio `C:\Curso\Tomcat2`.

2. Puesto que el servicio está asociado a la primera instancia, esta segunda habrá que levantarla manualmente. Previamente habrá que modificar los demás puertos para que no colisione con la instancia principal. Para ello, editar el fichero `server.xml` de la nueva instancia.

3. Buscar el puerto de shutdown (8005) y cambiarlo por ejemplo, por 8006:

```
<Server port="8005" shutdown="SHUTDOWN">
```

se sustituirá por:

```
<Server port="8006" shutdown="SHUTDOWN">
```

4. Cambiar el puerto del conector HTTP (8080) por ejemplo, por 8081:

```
<Connector port="8080" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true" />
```

pasa a ser:

```
<Connector port="8081" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8444" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true" />
```

5. Cambiar también el puerto del conector AJP para que no colisione con la otra instancia:

```
<Connector port="8009"
enableLookups="false" redirectPort="8443"
protocol="AJP/1.3"/>
```

se sustituirá por:

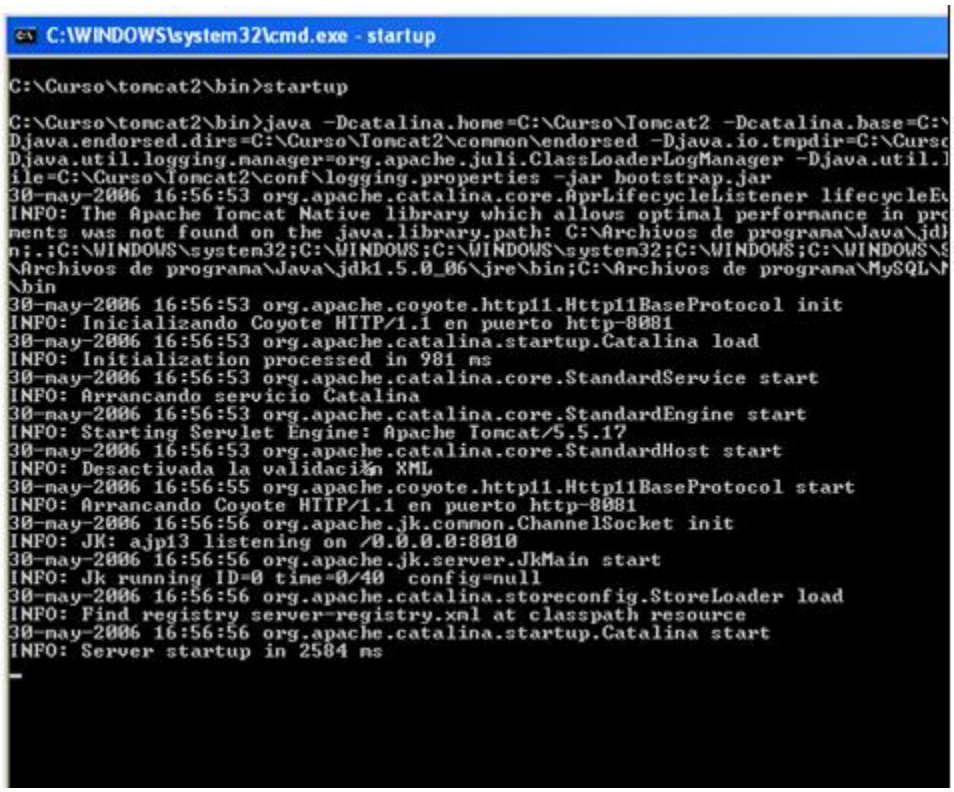
```
<Connector port="8010"
enableLookups="false" redirectPort="8444"
protocol="AJP/1.3"/>
```

6. Crear en el directorio `/bin` de la segunda instancia (Tomcat2) un fichero llamado `startup.bat` con el siguiente contenido (en una sola línea):

```
java
   -Dcatalina.home=C:\\Curso\\Tomcat2
   -Dcatalina.base=C:\\Curso\\Tomcat2
   -Djava.endorsed.dirs=C:\\Curso\\Tomcat2\\common\\endorsed
   -Djava.io.tmpdir=C:\\Curso\\Tomcat2\\Temp
   -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogMana
ger
```

```
   -Djava.util.logging.config.file=C:\\Curso\\Tomcat2\\conf\\
logging.properties
   -jar bootstrap.jar
```

7. Para levantar la instancia, abrir una ventana de comandos DOS y situarse sobre el directorio `/bin` de la nueva instancia. Ejecutar startup. Para cerrar la instancia, pinchar en la ventana DOS y teclear CTRL+C.



8. Comprobar que la instancia funciona correctamente abriendo una instancia de navegador y accediendo a la URL `http://localhost:8081/`.

Una vez que se dispone de los dos servidores Tomcat, tenemos que proceder a configurar el balanceo de carga con Apache. Para ello hay que seguir los siguientes pasos:

1. En lo que sigue, llamaremos a los servidores tomcat1 y tomcat2 (y a los directorios donde se instalan: `C:\Curso\Tomcat1` y `C:\Curso\Tomcat2`).

2. Creamos un fichero llamado `workers.properties` en `$APACHE_HOME/conf`

```
# workers.properties
# En Windows, las barras a la izquierda:
ps=\\
# lista de workers por sus nombres
worker.list=tomcat1, tomcat2, loadbalancer
```

```
# ------------------------
# Primer servidor tomcat
# ------------------------
worker.tomcat1.tomcat_home=”C:\\Curso\\Tomcat1\\”
worker.tomcat1.port=8009
worker.tomcat1.host=localhost
worker.tomcat1.type=ajp13
# Factor de balanceo de carga (1 a 100), más bajo->menos
trabajo
worker.tomcat1.lbfactor=1

# ------------------------
# Segundo servidor tomcat
# ------------------------
worker.tomcat2.tomcat_home=C:\\Curso\\Tomcat2\\
worker.tomcat2.port=8010
worker.tomcat2.host=localhost
worker.tomcat2.type=ajp13
# Factor de balanceo de carga (1 a 100), más bajo->menos
trabajo
worker.tomcat2.lbfactor=1

# ------------------------
# Load Balancer worker
# ------------------------
#
# El worker loadbalancer es de tipo lb, y ejecuta balanceo de
carga
# con ponderación round-robin mediante sticky sessions.
# Nota: si un worker se cae, el balanceador comprobará su
estado
# de tanto en tanto. Hasta que se levante de nuevo, todo el
trabajo
# será redirigido al otro worker
worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=tomcat1, tomcat2
```

3. Hacemos una copia del fichero `httpd.conf` en el directorio `$APACHE_HOME/conf`. Modificamos el fichero `httpd.conf` de apache para que cargue el módulo `mod_jk` y el fichero de configuración `mod_jk.conf`, y para definir las directivas de `mod_jk`.

```
LoadModule jk_module modules/mod_jk.so
#
# Conector mod_jk
Include "C:\\Curso\\Tomcat1\\conf\\auto\\mod_jk.conf"
JkWorkersFile conf\workers.properties
JkLogFilelogs\mod_jk.log
JkLogLevel    info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
```

4. Cerrar Apache con el monitor de Apache.

5. Creamos un fichero, al que llamamos `indice.jsp`, y lo ponemos en `C:\Curso\Tomcat1. \webapps\ROOT`, con el siguiente contenido:

```
<html>
  <body bgcolor="red">
    <center>
      <%= request.getSession().getId() %>
      <h1>Tomcat 1&lt/h1>
    </center>
  </body>
</html>
```

6. Creamos un fichero, al que llamamos `indice.jsp`, y lo ponemos en `C:\Curso\Tomcat2\webapps\ROOT`, con el siguiente contenido:

```
<html>
  <body bgcolor="blue">
    <center>
      <%= request.getSession().getId() %>
        <h1>Tomcat 2&lt/h1
    </center>
  </body>
</html>
```

7. Arrancamos ambos servidores Tomcat, y Apache y probamos la instalación.

   Primero, vamos a verificar que Apache sirve el contenido estático:

   - En `http://localhost/`, debemos ver la página de bienvenida de Apache.
   - En `http://localhost/indice.jsp`, debemos ver cuál de los servidores Tomcat sirve nuestra página web: Si aparece una página roja, la ha servido Tomcat 1, y si es azul, Tomcat 2. Si abrimos instancias distintas de navegador y accedemos a `http://localhost/indice.jsp`, como el factor de balanceo es el mismo para los dos workers deberían aparecer alternativamente una página roja y una azul.
   - Ahora probamos las sticky session (o afinidad de sesión): pinchamos el botón de recarga del navegador varias veces: no debe cambiar el color de la página recibida.

Si completas el ejercicio, haz un post en el foro de actividades con tus impresiones. Comenta cualquier aspecto que consideres conveniente.

4. Configurar el conector de cooperación entre Apache y Tomcat mod_jk para que se tengan dos instancias de Tomcat totalmente separadas pero accesibles desde un mismo servidor Apache.

Este ejercicio ilustra otro escenario: no necesitamos balanceo de carga, pero queremos configurar la pareja Apache/Tomcat para que utilicen instancias privadas de Tomcat (por ejemplo, en un entorno de ISP que vende hosting compartido con máquinas virtuales Tomcat).

Para ello, la configuración sería como sigue:

1. Modificar el fichero `conf/auto/mod_jk.conf` de la primera instancia de Tomcat para incluir las siguientes entradas:

```
NameVirtualHost *
<VirtualHost *>
  ServerName localhost1
  JkMount /*.jsp tomcat1
  JkMount /servlet/* tomcat1
</VirtualHost>

<VirtualHost *>
  ServerName localhost2
  JkMount /*.jsp tomcat2
  JkMount /servlet/* tomcat2
</VirtualHost>
```

2. Se puede probar introduciendo localhost1 y localhost2 en el fichero /etc/hosts con la dirección 127.0.0.1 (la de localhost). Reiniciar Apache y abrir una instancia de navegador. Si se teclea la URL `http://localhost1/indice.jsp` aparecerá la página roja y si se teclea la URL `http://localhost2/indice.jsp` aparecerá la pantalla azul, ya que en el primer caso contesta la primera instancia de Tomcat y en el segundo caso contesta la segunda.

Haz un nuevo post en el foro de actividades con tus impresiones, comentando lo que consideres conveniente del ejercicio.