INAP
INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

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

Módulo 4. Cooperación entre Apache y Tomcat

Capítulo 4. mod_proxy_ajp (conector HTTP)

Autores

Janine García Morera

Alexandra López de la Oliva Portugués

Julio Villena Román

Octubre de 2014

Índice de contenidos

CAPÍTULO 4 MOD_PROXY_AJP (CONECTOR HTTP)

El proxy AJP es un módulo que se basa en el estándar HTTP proxy que usa AJP en lugar de HTTP. Se apoya en `mod_proxy` para funcionar. Ambos módulos (`mod_proxy` y `mod_proxy_ajp`) deben estar cargados en el servidor, bien de forma estática o bien de forma dinámica (mediante la directiva `LoadModule`). En las últimas distribuciones, viene todo junto en `mod_proxy`.

Esta extensión provee soporte AJP13 al módulo `mod_proxy` para conectar un servidor Apache con uno o varios servidores Tomcat. La forma de usar `mod_proxy_ajp` es similar a la del protocolo HTTP:

```
<Location /servlets/>
  ProxyPass ajp://localhost:8009/servlets/
  ProxyPassReverse ajp://localhost:8009/servlets/
</Location>
```

Esto supone que todas las peticiones que se realicen a la URL `http://localhost/servlets/` serán direccionadas internamente a `http://localhost:8080/servlets` a través del conector ajp13 (puerto 8009). Esto sería equivalente, usando `mod_proxy_http`, a:

```
<Location /servlets/>
  ProxyPass http://localhost:8080/servlets/
  ProxyPassReverse http://localhost:8080/servlets/
</Location>
```

Siempre que se realice una redirección entre servidores, sea usando `mod_jk` o `mod_proxy`, hay que tener cuidado de no mapear URLs que previamente existen en el servidor que actúa de frontal, ya que en este caso la URL queda solapada y ocultada por la URL mapeada. Por ejemplo, si en el ejemplo anterior existiese un directorio `/servlets/` en el `DocumentRoot` de Apache, o un Alias, quedaría solapado por el mapeo realizado, y al acceder a la URL `http://localhost/servlets` se accedería al directorio `/servlets` de Tomcat, y no al de Apache.

4.1. Balanceo de carga con mod_proxy_ajp y mod_proxy_balancer

Ya se describió que uno de los nuevos módulos que se han desarrollado para las últimas versiones de Apache es el `mod_proxy_balancer`, que dota al servidor de capacidad de balancear la carga entre varios servidores HTTP, FTP y AJP apoyándose en el módulo `mod_proxy`.

Apache puede ser configurado como **proxy directo** o como **proxy inverso** (también conocido como modo Gateway o puerta de enlace).

Un **proxy directo** es un servidor intermedio que se encuentra entre el cliente y el servidor web. Con el fin de obtener el contenido desde el servidor web, el cliente envía una petición al proxy indicando como el destino el servidor web, y el proxy entonces solicita el contenido desde el servidor web y lo devuelve al cliente. El cliente debe estar especialmente configurado para utilizar el proxy directo para acceder a otros sitios. Un uso típico de un proxy directo es proporcionar acceso a Internet a los clientes internos de una Intranet que están restringidos por un firewall. El proxy directo también se usa para reducir el uso de la red, utilizando almacenamiento de contenidos en caché (usando `mod_cache`). En Apache, el proxy directo se activa usando la directiva `ProxyRequests`.

Un **proxy inverso** (o puerta de enlace), por el contrario, se presenta al cliente como un servidor web normal, pero que internamente no tiene los contenidos sino que llama a otros servidores para obtenerlos. No se necesita ninguna configuración especial en el cliente. El cliente realiza solicitudes normales de contenido al servidor proxy, que decide entonces dónde enviar las solicitudes, y devuelve el contenido como si fuera en sí mismo el origen.

Un uso típico de un proxy inverso es proporcionar a los usuarios de Internet el acceso a un servidor que está detrás de un firewall. Los proxies inversos también pueden utilizarse para equilibrar la carga entre varios servidores de back-end, o para proporcionar almacenamiento en caché a servidores más lentos. O simplemente para unificar varios servidores en el mismo espacio URL. Un proxy inverso se activa usando la directiva `ProxyPass`.

Esta característica de proxy inverso es especialmente útil si la aplicamos a la configuración de un cluster de servidores Tomcat: un servidor Apache actúa de frontal y reparte las peticiones de contenido dinámico (servlets o JSPs) a una granja de servidores Tomcat mediante un algoritmo de balanceo, con mantenimiento de sesión o no, según las necesidades del entorno.

La conexión entre el servidor Apache que actúa como balanceador y los servidores Tomcat se puede basar en el protocolo HTTP (mediante el módulo `mod_proxy_http`, usando el puerto 8080) o en el protocolo AJP13 (mediante el módulo `mod_proxy_ajp`, usando el puerto 8009). Esta última opción como ya se sabe es más eficiente, al tratarse AJP13 de un protocolo optimizado para la conexión con Tomcat. Este procedimiento sustituiría al implementado a través del módulo `mod_jk`.

La documentación completa del módulo de Apache está en <httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html>

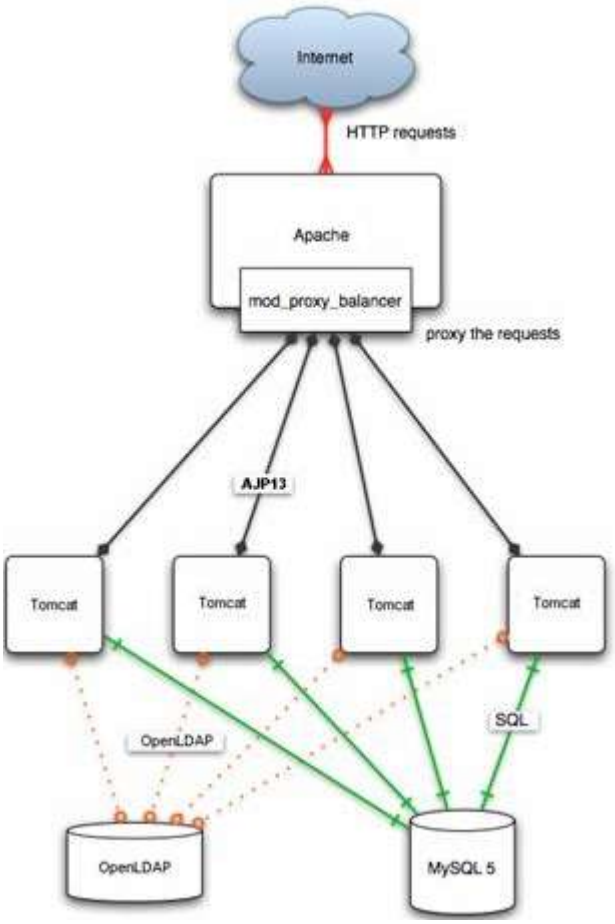

Figura 4.4.1: Conexión entre Apache y Tomcat

Para activar el soporte de balanceador AJP es necesario que el servidor Apache cargue, bien de forma estática o bien de forma dinámica, los módulos `mod_proxy`, `mod_proxy_ajp` y `mod_proxy_balancer`. Además, si se quiere realizar el control dinámico del balanceador, es necesario cargar también el módulo `mod_status`.

Para configurar el balanceo es necesario usar la directiva de Apache Proxy con el parámetro `balancer://nombre_cluster`. De esta manera se configura un cluster, de nombre nombre_cluster, que balanceará entre los miembros (workers) que se definirán a continuación (`BalancerMember`).

Por ejemplo, definimos el cluster `micluster` que consta de tres miembros: `tomcat1`, `tomcat2` y `tomcat3`, de la siguiente manera:

```
<Proxy balancer://micluster>
  BalancerMember ajp://tomcat1:8009/
  BalancerMember ajp://tomcat2:8009/
  BalancerMember ajp://tomcat3:8009/
</Proxy>
```

Ahora podemos asignar el balanceador a todas las peticiones que nos llegan a una determinada URL (por ejemplo, `http://miservidor/tomcat/`)

```
<Location /tomcat/>
  ProxyPass balancer://micluster/
</Location>
```

Si no se usa la sección <`Location`>, se asignaría de la siguiente forma:

```
ProxyPass /tomcat/ balancer://micluster/
```

4.1.1. Parámetros de las directivas

En la definición de los workers (`BalancerMember`) se pueden usar un gran número los parámetros. Toda la información está en <httpd.apache.org/docs/2.4/mod/mod_proxy.html>.

La siguiente tabla presenta un resumen de los parámetros más importantes y frecuentemente utilizados.

Parámetro	Defecto	Descripción
min	0	Mínimo número de conexiones que se abrirán contra el servidor de backend
max	1…N	Máximo número de conexiones contra el backend. El valor por defecto depende del tipo de MPM que se utilice.
smax	max	Límite máximo de conexiones que se crearán bajo demanda. Cualquier conexión por encima de `smax` tendrá un tiempo máximo de inactividad fijado por el parámetro ttl
ttl	-	Tiempo de vida de las conexiones inactivas que se hayan creado
timeout	-	Timeout de conexión en segundos. Si no se ha configurado, Apache esperará hasta que se libere una conexión.
acquire	-	Tiempo de espera hasta obtener una conexión libre del pool de conexiones. Si se supera este tiempo sin obtener conexión, se devuelve al cliente un estado `SERVER_BUSY`.
Keepalive	on	Indica al sistema operativo que mantenga la conexión empleando señales de keepalive para que no se corten por inactividad. Necesario en el caso de existir un firewall.

retry	60	Tiempo de reintentos para conseguir una conexión.
loadfactor	1	Factor de carga del worker, que se utiliza con `LoadBalancer`. Es un número entre 1 y 100 que indica la carga proporcional que se asigna al worker.
route	-	Ruta que se asigna al worker en un balanceador de carga. Es un valor que se concatena al `session_id`.
redirect	-	Ruta de redirección del worker cuando se usa en un balanceador de carga. Este valor normalmente se asigna de forma dinámica para hacer más sencilla la retirada de un node del cluster.

Cuando la directiva <Proxy> comienza con `balancer://`, (definición de un worker virtual, llamado `balancer://`) hay un conjunto especial de parámetros que pueden ser utilizados para controlar el comportamiento de ese worker virtual:

Parámetro	Defecto	Descripción
lbmethod	Byterequests	Método de balanceo de carga: según el algoritmo `byterequests` o `bytraffic`
stickysession	-	Nombre de la sticky session del balanceador. Este valor normalmente se fija a `JSESSIONID` o `PHPSESSIONID` (según se balancee entre servidores Java o PHP) y depende de la aplicación de backend que soporte las sesiones.
nofailover	Off	Si se fija a On, la sesión abortará si el worker está en estado de error o desactivado.
timeout	0	Timeout del balanceador en segundos. Es el tiempo máximo de espera para un worker libre. El defecto es que no se espera a ningún worker.
maxattempts	1	Máximo número de intentos de failover

Por ejemplo, si en el cluster `micluster` uno de los servidores de backend es más pequeño que los otros dos y queremos que sirva menos peticiones y que tenga menos carga proporcional de trabajo, y además queremos que el balanceador trabaje con persistencia de sesión y que una sesión falle si el worker con el que trabaja pasa a error, la configuración será la siguiente:

```
<Proxy balancer://micluster>
  BalancerMember ajp://tomcat1:8009/ smax=10 loadfactor=5
  BalancerMember ajp://tomcat2:8009/ smax=10 loadfactor=5
  BalancerMember ajp://tomcat3:8009/ smax=1 loadfactor=1
</Proxy>
<Location /tomcat/>
  ProxyPass    balancer://micluster/    stickysession=jsessionid
nofailover=On
</Location>
```