

Configuring mod_proxy

Configuring Apache mod_proxy with Jetty

The apache web server is frequently used as a server in front of a servlet container. While there are no real technical reasons to front Jetty with apache, sometimes this is needed for software load balancing, or to fit with a corporate infrastructure, or simply to stick with a known deployment structure.

There are 3 main alternative for connection Apache to Jetty:

1. Using apache [mod_proxy](#) and an normal Jetty HTTP connector.
2. Using apache [mod_proxy_ajp](#) and the Jetty AJP connector.
3. Using apache [mod_jk](#) and the Jetty AJP connector.

Using the HTTP Connectors is greatly preferred, as Jetty performs significantly better with HTTP and the AJP protocol is poorly documented and there are many version irregularities. If AJP is to be used, the then [mod_proxy_ajp](#) module is preferred over [mod_jk](#). Previously, the load balancing capabilities of [mod_jk](#) meant that it had to be used (tolerated), but with apache 2.2, [mod_proxy_balancer](#) is available and load balance over HTTP and AJP connectors.

mod_proxy

Apache has a [mod_proxy](#) module available for almost all versions of apache. However, prior to apache 2.2, only reverse proxy features were available and [mod_proxy_balancer](#) was not available for load balancing.

Documentation for [mod_proxy](#) is available for:

- [apache 1.3](#)
- [apache 2.0](#)
- [apache 2.1](#)
- [apache 2.2](#)

Configuration as a [Reverse Proxy](#)

The configuration file layout for apache varies greatly with version and distribution, but to configure [mod_proxy](#) as a reverse proxy, the follow configuration is key:

1. Jetty needs to be configured with a normal HTTP connector, probably on port 8080 or similar.
2. The proxy module (and other proxy extension used) must be loaded:

```
LoadModule proxy_module modules/mod_proxy.so
```

Apache 2.2 normally bundles [mod_proxy](#), [mod_proxy_ajp](#) and [mod_proxy_balancer](#), so they often do not need to be installed separately. If they are separately bundled by your operation system (eg as RPMs or debians) ensure that they are installed.

3. Forward proxy needs to be turned off:

```
ProxyRequests Off

<Proxy *>
Order deny,allow
Allow from all
</Proxy>
```

4. Reverse proxy paths must be configured with URL of the jetty server:

```
ProxyPass /test http://localhost:8080/test
```

5. Frequently apache documentation will instruct that `ProxyPassReverse` configuration be used so that apache can rewrite any URLs in headers etc. However, if you use the `ProxyPreserveHost` configuration, Jetty can generate the correct URLs and they do not need to be rewritten:

```
ProxyPreserveHost On
```

Alternatively, since Jetty 6.1.10, instead of preserving the host and to retrieve the client remote address in the webapp (`ServletRequest#getRemoteAddr()`) you can use the [forwarded](#) property on `AbstractConnector` which interprets the [mod_proxy http "x-forwarded-" headers](#) instead:

```
<Configure id="Server" class="org.mortbay.jetty.Server">
...
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.nio.SelectChannelConnector">
      <Set name="port">8080</Set>
      <Set name="forwarded">true</Set>
    </New>
  </Arg>
</Call>
...
</Configure>
```

Or, to force the result of `ServletRequest#getServerName()` and `ServletRequest#getServerPort()` (if headers are not available):

```

<Configure id="Server" class="org.mortbay.jetty.Server">
  ...
  <Call name="addConnector">
    <Arg>
      <New class="org.mortbay.jetty.nio.SelectChannelConnector">
        <Set name="port">8080</Set>
        <Set name="forwarded">true</Set>
        <Set name="hostHeader">example.com:81</Set>
      </New>
    </Arg>
  </Call>
  ...
</Configure>

```

6. It is also very useful to turn on proxy status monitoring (see management below):

```
ProxyStatus On
```

Proxying SSL on Apache to HTTP on Jetty

The situation here is:

```

https           http
----->  Apache  ----->  Jetty

```

If you want to offload the SSL onto Apache, and then use plain http requests to your Jetty backend, you need to configure Jetty to use https:// in all redirected requests.

You can do that by extending the Connector class of your choice, eg the SelectChannelConnector, and implement the customize(EndPoint, Request) method to force the scheme of the Request to be https like so (**don't forget to call super.customize(endpoint,request)!**):

```
public void
customize(org.mortbay.io.EndPoint endpoint,
Request request) throws IOException
{
    request.setScheme("https");
    super.customize(endpoint, request);
}
```

If you need access on Jetty to some of the SSL information accessible on Apache, then you need to some configuration tricks on Apache to insert the SSL info as headers on outgoing requests. Follow the Apache configuration suggestions on this [tutorial](#) which shows you how to use `mod_headers` to insert the appropriate request headers. Of course you will also need to code your application to look for the corresponding custom request headers bearing the ssl information.

mod_proxy_balancer

With apache 2.2 [mod_proxy](#) is able to use the extension [mod_proxy_balancer](#)

Configuration

The configuration of `mod_proxy_balancer` is similar to pure `mod_proxy`, except that `balancer://` URLs may be used as a protocol instead of `http://` when specifying destinations (workers) in `ProxyPass` elements.

```
# map to cluster with session affinity
(sticky sessions)
ProxyPass /balancer !
ProxyPass / balancer://my_cluster/
stickysession=jsessionid nofailover=On

<Proxy balancer://my_cluster>
    BalancerMember http://yourjetty1:8080
route=jetty1
    BalancerMember http://yourjetty2:8080
route=jetty2
</Proxy>
```

Proxy balancer:// - defines the nodes (workers) in the cluster. Each member may be a `http://` or `ajp://` URL or another `balancer://` URL for cascaded load balancing configuration.

If the worker name is not set for the Jetty servers, then session affinity (sticky sessions) will not work. The `JSESSIONID` cookie must have the format `<sessionID>.<worker name>`, in which `worker name` has the same value as the `route` specified in the `BalancerMember` above (in this case "jetty1" and "jetty2"). See [this article](#) for details. The following can be added to the `jetty-web.xml` in the `WEB-INF` directory to set the worker name.

```
<Configure
class="org.mortbay.jetty.webapp.WebAppContext">
  <Get name="sessionHandler">
    <Get name="sessionManager">
      <Call name="setIdManager">
        <Arg>
          <New
class="org.mortbay.jetty.servlet.HashSession
IdManager">
            <Set
name="WorkerName">jetty1</Set>
          </New>
        </Arg>
      </Call>
    </Get>
  </Get>
</Configure>
```

Management

Apache provide [mod_status](#) and [Balancer Manager Support](#) so that the status of the proxy and balancer can be viewed on a web page. The following configuration enables these UIs at /balancer and /status URLs:

```
<Location /balancer>  
SetHandler balancer-manager
```

```
Order Deny,Allow  
Deny from all  
Allow from all  
</Location>
```

```
ProxyStatus On  
<Location /status>  
SetHandler server-status
```

```
Order Deny,Allow  
Deny from all  
Allow from all  
</Location>
```

These UIs should be protected from external access.