Subject: 3 Easy Command for Proxy Application in Apache
Posted by admin on Thu, 30 Jul 2026 10:53:13 GMT
View Forum Message <> Reply to Message

Introduction

The Apache web server is much of the time utilized as a server as a part of front of a servlet holder. While there are no genuine specialized motivations to front Jetty with apache, at times this is required for programming load adjusting, or to fit with a corporate foundation, or essentially to stay with a known arrangement structure.

Approaches to interface Apache to Jetty include:

1. Utilizing Apache mod_proxy and a typical Jetty HTTP connector

2. Utilizing Apache mod_proxy_ajp and the Jetty AJP connector

3. Utilizing Apache mod_jk and the Jetty AJP connector

We suggest utilizing the HTTP connectors for the accompanying reasons: Pier performs fundamentally better with HTTP.

The AJP convention is inadequately archived and has numerous adaptation anomalies.

In the event that you should utilize AJP, the mod_proxy_ajp module is superior to anything mod_jk. Already, the heap adjusting abilities of mod_jk implied that you needed to utilize (endure) it, however with Apache 2.2, mod_proxy_balancer is accessible and works over HTTP and AJP connectors.
MOD_PROXY

A mod proxy module is accessible for all variants of Apache. Be that as it may, before Apache 2.2, just invert intermediary components were accessible and mod_proxy_balancer was not accessible for burden adjusting. Designing mod proxy as a Reverse Proxy. The setup document format for Apache differs extraordinarily with adaptation and appropriation, yet to arrange mod proxy as a converse intermediary, the accompanying is vital: Design Jetty with an ordinary HTTP connector, on port 8080 or comparative.
Load the intermediary module (and whatever other intermediary augmentation utilized)

Load Module proxy module modules/mod_proxy.so ordinarily packages mod proxy, mod_proxy_ajp, and mod_proxy_balancer, so frequently you don't have to introduce them independently. In the event that they are packaged independently in your working framework, for instance, as RPMs or Debians, make certain to introduce them. Turn off forward proxy.

```
proxy request

ProxyRequests Off

<Proxy >
```

Order deny, allow

Allow from all

</Proxy>

Configure reverse proxy paths with the URL of the Jetty server

Proxy Pass test http://localhost:8080/test

Often Apache documentation teaches that you utilize ProxyPassReverse design with the goal that Apache can modify any URLs in headers. Be that as it may, on the off chance that you utilize the ProxyPreserveHost setup, Jetty can produce the right URLs, and reworking is redundant:
ProxyPreserveHost On

Alternatively, since Jetty 6.1.10, instead of preserving the host and to repossess the client remote address in the webapp (ServletRequest#getRemoteAddr()), you can use the forwarded property on AbstractConnector, which reads the mod_proxy_http "X-Forwarded-"* headers in its place:

CodeAdress

<Configure id="Server" class="org.eclipse.jetty.Server">

...

<Call name="addConnector">

<Arg>

<New class="org.eclipse.jetty.nio.SelectChannelConnector">

<Set name="port">8080</Set>

<Set name="forwarded">true</Set>

</New>

</Arg>

</Call>

...

</Configure>

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

```
<Configure id="Server" class="org.eclipse.jetty.Server">

...

<Call name="addConnector">

<Arg>

<New class="org.eclipse.jetty.nio.SelectChannelConnector">

<Set name="port">8080</Set>

<Set name="forwarded">true</Set>

<Set name="hostHeader">example.com:81</Set>

</New>

</Arg>

</Call>

...

</Configure>
```

It is also very useful to turn on proxy status monitoring (see management below):
Proxy Status On

Proxying SSL on Apache to HTTP on Jetty

The situation here is:

```
https http

> Apache > Jetty
```

In the event that you need to offload the SSL onto Apache, and afterward utilize plain http solicitations to your Jetty backend, you have to arrange Jetty to utilize https://in all diverted solicitations.

You can do that by developing your preferred Connector class, e.g. the SelectChannelConnector, and actualize the customize (Endpoint, Request) strategy to compel the plan of the Request to be https like so (remember to call super. Customize (endpoint, request)!):

```
Endpoint Request
```

public void customize(org.eclipse.io.EndPoint endpoint, Request request) throws IOException

{

request.setScheme("https");

super.customize(endpoint, request);

}

Read this article to achieve the same thing without coding.

An even easier way to achieve this with Jetty 9 + Apache 2.2 is this:

add

RequestHeader set X-Forwarded-Proto "https" env=HTTPS

to your Apache preparation (adapted from this environment post and ServerFault.com).

On the off chance that you need access on Jetty to a helping of the SSL data open on Apache, then you have to do some strategy traps on Apache to embed the SSL info as headers on active solicitations. Take after the Apache arrangement suggestions on this instructional exercise, which demonstrates to you proper methodologies to utilize mod_headers to embed the suitable solicitation headers.

Obviously you will likewise need to code your application to search for the comparing custom solicitation headers bearing the ssl data.mod_proxy_balancer Through Apache 2.2 mod proxy is bright to use the postponement mod_proxy_balancer.
Configuring mod_proxy_balancer

The configuration of mod_proxy_balancer is comparable to pure mod_proxy, excluding that balancer:// URLs may be used as a protocol in its place of http:// when specifying destinations (workers) in Proxy Pass elements.

mode proxy configuration
map to cluster with session affinity (sticky sessions)

Proxy Pass balancer!

ProxyPass / balancer:</nowiki>//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:// describes the nodes (workers) in the bunch. Each member can be a http:// or ajp:// URL or another balancer:// URL for cascaded load balancing formation.