

Servlets Bundled with Jetty

Servlets Bundled with Jetty

With the release of Jetty 6 as a light-weight embeddable web server, it doesn't neglect handling of servlets aside from web application's presentation-based java server pages (jsp). In fact, servlet is one of the foundation of Jetty as a component-based infrastructure to hold and run j2ee applications.

Jetty 6 ships with a bundle of servlets that interacts with the key classes. Mostly they can be found in the `org.mortbay.servlet` package. Here is a list of servlets playing a major role in running and maintaining the Jetty server:

[GzipFilter](#)

See [GZIP Compression](#).

[MultiPartFilter](#)

This class implements the `Filter` interface that is called by the web container. It acts as a decoder of the multipart/form-data stream sent by a HTML form that utilizes a file input item. Any files that is sent are stored to a temporary location and there will be a file object added to the request which will act as an attribute. It is made available that other values through the usual `getParameter` API and the `setCharacterEncoding` mechanism is respected when converting bytes to Strings.

[NoJspServlet](#)

This is a simple servlet which extends the abstract class `HttpServlet` that displays the code 500 error message "JSP support not configured."

[ProxyServlet](#)

This is an experimental servlet which implements the `Servlet` interface. It acts as an authorized servlet on behalf on another running servlet. This seems not to be a substitute for the running servlet but provides a mirror for the service.

[QoSFilter](#)

Quality of Service Filter. This filter limits the number of active requests to the number set by the "maxRequests" init parameter (default 10). If more requests are received, they are suspended and placed on priority queues. Priorities are determined by the `getPriority(ServletRequest)` method and are a value between 0 and the value given by the "maxPriority" init parameter (default 10), with higher values having higher priority.

This filter is ideal to prevent wasting threads waiting for slow/limited resources such as a JDBC connection pool. It avoids the situation where all of a containers thread pool may be consumed blocking on such a slow resource. By limiting the number of active threads, a smaller thread pool may be used as the threads are not wasted waiting. Thus more memory may be available for use by the active threads.

Furthermore, this filter uses a priority when resuming waiting requests. So that if a container is under load, and there are many requests waiting for resources, the `getPriority(ServletRequest)` method is used, so that more important requests are serviced first. For example, this filter could be deployed with a maxRequest limit slightly smaller than the containers thread pool and a high priority allocated to admin users. Thus regardless of load, admin users would always be able to access the web application.

The maxRequest limit is policed by a Semaphore and the filter will wait a short while attempting to acquire the

semaphore. This wait is controlled by the "waitMs" init parameter and allows the expense of a suspend to be avoided if the semaphore is shortly available. If the semaphore cannot be obtained, the request will be suspended for the default suspend period of the container or the valued set as the "suspendMs" init parameter.

ThrottlingFilter

This has been replaced by the QoSFilter. Implemented by the Filter interface, the ThrottlingFilter servlet concern is to protect a web application from having to handle an unmanageable load. This accounts when there is a server that holds one application with standardized resource restrictions and with this will be controlled by lowering or limiting the size of the ThreadPool. But when there are several applications in service or a single app having different resource requirements to different URLs, then the ThrottlingFilter comes in and guiding the number of requests being handled by Jetty.

The filter has three configurable values:

maximum - determines the maximum number of requests that may be on the filter chain anytime
block - determines how long (in milliseconds) a request will be queued before it is rejected. Setting this to -1 blocks indefinitely
queue - determines how many requests can be queued simultaneously and any additional requests will be rejected. Setting this to 0 turns off queueing

WelcomeFilter

This is another servlet which also implements the Filter interface. This forwards any servlets to the necessary welcome display of an application when serviced.

DefaultServlet

Found in the org.mortbay.jetty.servlet package, this servlet implements the ResourceFactory interface and extends the HttpServlet abstract class. This is the default servlet of Jetty usually mapped to / provides handling for static content, OPTION and TRACE methods for the context. The MOVE method is allowed if PUT and DELETE are allowed.

The following initParameters are supported:

acceptRanges - if true, range requests and responses are supported
dirAllowed - if true, directory listings are returned if no welcome file is found. otherwise 403 Forbidden is displayed
redirectWelcome - if true, welcome files are redirected rather than forwarded to
gzip - if set to true, then static content will be served as gzip content encoded if a matching resource is found ending with ".gz"
resourceBase - set to replace the context resource base
aliases - if true - aliases of resources are allowed (e.g. symbolic links and caps variations) and may bypass security constraints
maxCacheSize - the maximum total size of the cache or 0 for no cache
maxCachedFileSize - the maximum size of a file to cache
maxCachedFiles - the maximum number of files to cache
useFileMappedBuffer - if set to true, it will use mapped file buffer to serve static content when using NIO connector. Setting this value to false means that a direct buffer will be used instead of a mapped file buffer. By default, this is set to true

*CGI

The CGI servlet class extends the abstract HttpServlet class. When the init parameter is called, the cge bin directory

is set with the `cgibinResourceBase` otherwise, it will default to the resource base of the context. There are three parameters that the `cgi bin` is using:

`commandPrefix` - this is the init parameter that is obtained when there is a prefix set to all commands directed to the method `exec`

`Path` - this is just an init parameter passed to the `exec` environment as a `PATH`. However, this must be run unpacked somewhere in the filesystem

`ENV_` - initParameter that starts with `ENV_` is used to point to an environment variable with the name stripped of the leading `ENV_` and using the init parameter value