

How to gracefully shutdown

Graceful shutdown

Graceful shutdown of a server, context or connector is when existing request/connections are allowed to gracefully complete while no new requests and/or connections are accepted. It is configured on the Server instance with the `setGracefulShutdown(long)` method. Here's an example of setting this via the `jetty.xml` file, where we specify a "grace" period of 1000 milliseconds:

```
<Set name="gracefulShutdown">1000</Set>
```

The "grace" period is the time the container will wait for requests currently inside the container to finish processing before shutting down.

As soon as the shutdown command is given, the container will close the connectors so that they do not accept any more inbound connections. This will inform most load balancers that the server is no longer part of the cluster. The contexts are closed so that they do not accept any more requests, but the requests currently inside the container will drain out and the Server instance will shutdown after the grace period expires.

You must also call the `setStopAtShutdown(boolean)` method with a value of `true` for the grace period to take effect.

```
<Set name="stopAtShutdown">true</Set>
```

APIs For Shutdown Implementation

Even though Jetty will automatically handle a controlled shutdown if you use the `setGracefulShutdown(long)` method on the Server instance, sometimes you may want to implement your own shutdown handling, for example, shutting down just a single context or a connector. Here's how.

Graceful context shutdown

The `ContextHandler` and all the classes derived from it (`Context`, `WebAppContext`) has a `setShutdown(boolean)` method, which if passed `true` will prevent the context from accepting new requests. Requests that are currently being handled by the context are not affected.

Requests that are rejected by a shutdown context are passed to the next configured context that matches the context path. Note that several contexts can be registered for the same context path, so a new context (or a notification of maintenance context) may be revealed by a shutdown context.

A `setShutdown(false)` can be called to allow the context to continue handing new requests.

Once a context is shutdown and the number of requests has reduced to zero (or near zero), then `stop()` should be called to actually stop the components of the context.

The `ContextHandler.setShutdown(boolean)` method is exposed via an MBean and can be called from a management agent.

Graceful connector shutdown

The `Connector.close()` method can be called to close the server socket on which a connector is listening. This will prevent new connections from being accepted and inform most load balancers that the server is no longer part of a cluster. Existing client connections can continue to run until they timeout or `stop()` is called on the connector.

A closed connector should be stopped before being restarted. `Connector.open()` cannot be called on a started connector.