# fastcgi

# 1 Overview

The fastcgi library is a small library providing bigloo with partial support for the fastcgi protocol. Although the fastcgi protocol supports a number of application types (responder, authorizer, filter), the bigloo fastcgi library only supports the most common application type, responder. The responder application type provides a similar, but more efficient, application model to CGI/1.1. Support for the authorizer and filter application types may be added later.

The rest of this document will describes the bigloo fastcgi application programming interface.

# 2 API Reference

.

## 2.1 fastcgi-for-each

The most important procedure provided by the bigloo fastcgi library is `fast-for-each`. It provides the the means by which a handler for fastcgi requests are established.

`fastcgi-for-each` *proc* [Function]
: `fastcgi-for-each` accepts a single argument, the procedure to handle incoming fastcgi requests. In normal operation this method will not return. It waits in a loop for fastcgi requests and upon receiving a request dispatches them to *proc*. *proc* must be a procedure that accepts a single fastcgi-request argument and produces a fastcgi response.

## 2.2 fastcgi-request

The `fastcgi-request` class reifies a fastcgi request. It allows access to the request information as well as the means to form a fastcgi response.

`input` [Instance Variable of `fastcgi-request`]
: The `input` field is an input port providing access to the input data, if any, provided with the fastcgi request.

`output` [Instance Variable of `fastcgi-request`]
: The `output` field is an output port. Data pushed to `output` is used to form the response to a fastcgi request.

`error` [Instance Variable of `fastcgi-request`]
: The `error` field is an output port. Data pushed to `output` is used to report errors to the application (usually a web server) calling the fastcgi application.

`getenv` *request name* [Method on `fastcgi-request`]
: The `getenv` method retrieves the environment variable, `name`, from the `request` object. If `name` is not found, `getenv` returns #f.

`get-param` *request name* [Method on `fastcgi-request`]
: The `get-param` method retrieves the GET or POST parameter, `name`. If `name` is not found, `get-param` returns #f.

`get-part` *request name* [Method on `fastcgi-request`]
: The `get-part` method retrieves the part, `name`. If `name` is not found, `get-part` returns #f.

`finish` *request* [Method on `fastcgi-request`]
: The `finish` method finalizes the response to a fastcgi-request.

The `form-part` class represents a single part in a multi-part document.

`name` *part* [Method on `form-part`]
: `name` returns the name of the part.

`get-header` *part name* [Method on `form-part`]

`get-header` returns the header, `name`. If the `name` does not exist, `get-header` returns #f.

The following is a simple example of using the fastcgi library. It returns a simple hello world page when invoked.

```
(module example
   (library fastcgi)
   (main main))

(define (handle-request request)
   (with-access::fastcgi-request request (output)
      (fprint output "Content-type: text/html")
      (newline output)
      (fprint output "<html><head><title>Example</title>")
      (fprint output "</head><body><h1>Example</h1>")
      (fprint output "<p>Hello World</p></body</html>")
      (fastcgi-finish request)))


(define (main args)
   (fastcgi-for-each handle-request))
```

# Index

## F



## G



## N

# Table of Contents