WSGI 1.0 Cheat-sheet

Definitions

- **Server**: HTTP server that has Python embedded (or it's itself a Python application), and calls the WSGI application callable directly.
- **Gateway**: Server-independent Python-powered application that calls the WSGI application callable directly and "connects" it to the Web server.
- **Application**: The WSGI application callable.
- **Middleware**: Callable that wraps one or more WSGI applications, to filter their requests and/or responses.

WSGI environ variables

There are many variables, but the following are possibly the most commonly used ones when you want to do something simple yet low-level without a framework:

- **REQUEST_METHOD**: The HTTP request method (e.g., "POST", "GET", "HEAD", "PUT").
- **SCRIPT_NAME**: The portion of the URL path that is not consumed by the application. For example, if you have Trac running on http://example.org/trac/, the script name for Trac will always be "/trac". If Trac is running on http://example.org/ then the script name is an empty string.
- **PATH_INFO**: The portion of the URL path that is consumed by the application. For example, if you have Trac running on http://example.org/trac/, the path info for Trac will always be **everything after** "/trac". It is the path info is the part of the URL path not use by the script name.
- **QUERY_STRING**: The URL-encoded string that contains the so-called "GET arguments". If set, it comes after the question mark in URLs.
- **REMOTE_USER**: If the user that made the request has been authenticated successfully (in this request or in a previous one), this variable represents his unique user identifier (e.g., a name).
- **HTTP_*** variables: Those present in the HTTP request, in upper case and with hyphens replaced with underscores. For example, *User-Agent* becomes *HTTP_USER_AGENT*.
- **wsgi.input**: The file-like object that contains the **body of the request**.
- **wsgi.url_scheme**: The scheme portion of the URL ("http" or "https").

API

WSGI application

It can be any callable. It takes two positional arguments, the WSGI environ and the server-provided start_response() callable. To send a response, start_response() must be called to send the headers first and then an iterable representing the body must be returned.

WSGI middleware

WSGI middleware must be a callable with the same API as a WSGI application. Servers or gateways must not try to distinguish an application from a piece of middleware.

WSGI environ

A *dict* instance. Not a dictionary-like object.

start_response()

A callable provided by the server or gateway on every request, used by the WSGI application to send its response headers. It takes two positional arguments: The HTTP status string (e.g., "200 OK") and a list made up of tuples whose first element is the header name and the second is the header value.

It returns a callable which can be used by legacy applications to send the body (aka write() callable), if they cannot return an iterable. It can be called multiple times and its only argument is a string to be sent.

File wrapper

If the server or gateway supports a high performance method to send files, it'd be available in the WSGI environ as "wsgi.file_wrapper".

It's a callable that takes one mandatory argument (the file-like object whose contents should be sent) and an optional size hint (the amount of bytes that should be sent at the same time).

wsgi.errors

A file-like object in the WSGI environ where non-fatal errors should be written. The messages are usually sent to the server's main error log.

Links

- **Web-SIG**: The best place to ask WSGI-related questions:
  http://mail.python.org/mailman/listinfo/web-sig
  http://wsgi.org/
- **PEP-333**: http://www.python.org/dev/peps/pep-0333/

WSGI "shops"

- **Paste**: http://www.pythonpaste.org
- **Repoze**: http://www.repoze.org/