Introduction

WSGI[1] is not a server, a python module, a framework, an API or any kind of software. It is just an interface specification by which server and application communicate. Both server and application interface sides are specified in the **PEP 3333**. If an application (or framework or toolkit) is written to the WSGI spec then it will run on any server written to that spec.

WSGI applications (meaning WSGI compliant) can be stacked. Those in the middle of the stack are called middleware and must implement both sides of the WSGI interface, application and server. For the application in top of it it will behave as a server and for the application (or server) bellow as an application.

A WSGI server (meaning WSGI compliant) only receives the request from the client, pass it to the application and then send the response returned by the application to the client. It does nothing else. All the gory details must be supplied by the application or middleware.

It is not necessary to learn the WSGI spec to build applications on top of frameworks or toolkits. To use middleware one must have a minimum understanding of how to stack them with the application or framework unless it is already integrated in the framework or the framework provides some kind of wrapper to integrate those that are not.

Python 2.5 and later comes with a WSGI server which will be used in this tutorial. In 2.4 and earlier it can be installed. For production code employ an industry proven standard such as Apache with mod_wsgi.

All the code in this tutorial is low level and has the sole purpose to demonstrate the WSGI specification at work. It is not meant for real use. For production code stick to toolkits, frameworks and middleware.

[1] Web Server Gateway Interface