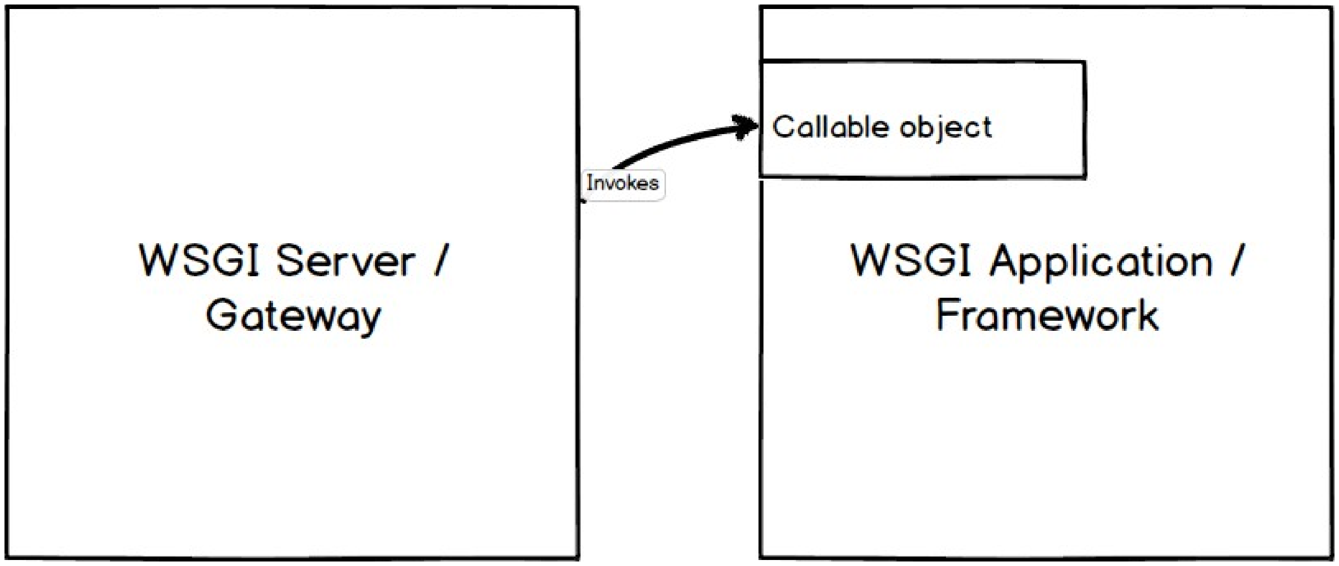
WSGI(Web Server Gateway Interface)

A WSGI server implements the web server side of the WSGI interface for running Python web applications.

Why is WSGI necessary?

Therefore the Python community came up with WSGI as a standard interface that modules and containers could implement. WSGI is now the accepted approach for running Python web applications.



As shown in the above diagram, a WSGI server simply invokes a callable object on the WSGI application as defined by the PEP 3333 standard.

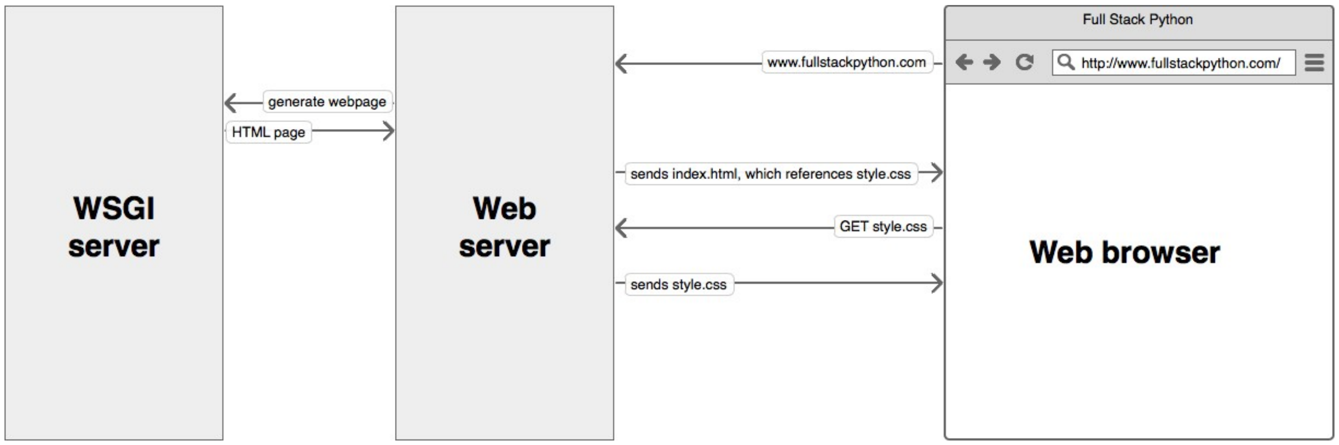
WSGI's Purpose

Why use WSGI and not just point a web server directly at an application?

- **WSGI gives you flexibility**. Application developers can swap out web stack components for others. For example, a developer can switch from Green Unicorn to uWSGI without modifying the application or framework that implements WSGI. From PEP 3333:

  The availability and widespread use of such an API in web servers for Python [...] would separate choice of framework from choice of web server, freeing users to choose a pairing that suits them, while freeing framework and server developers to focus on their preferred area of specialization.

- **WSGI servers promote scaling**. Serving thousands of requests for dynamic content at once is the domain of WSGI servers, not frameworks. WSGI servers handle processing requests from the web server and deciding how to communicate those requests to an application framework's process. The segregation of responsibilities is important for efficiently scaling web traffic.



WSGI is by design a simple standard interface for running Python code. As a web developer you won't need to know much more than

- what WSGI stands for (Web Server Gateway Inteface)
- that a WSGI container is a separate running process that runs on a different port than your web server
- your web server is configured to pass requests to the WSGI container which runs your web application, then pass the response (in the form of HTML) back to the requester

If you're using a standard web framework such as Django, Flask, or Bottle, or almost any other current Python framework, you don't need to worry about how frameworks implement the application side of the WSGI standard. Likewise, if you're using a standard WSGI container such as Green Unicorn, uWSGI, mod_wsgi, or gevent, you can get them running without worrying about how they implement the WSGI standard.

However, knowing the WSGI standard and how these frameworks and containers implement WSGI should be on your learning checklist though as you become a more experienced Python web developer.

Examples:
http://fosshelp.blogspot.com/2014/03/getting-started-with-python-wsgi-and.html
http://wsgi.tutorial.codepoint.net/parsing-the-request-get