HOPE foundation

PYTHON PROGRAMMING (Basics)

* Web Servers

Web Servers

Web servers respond to Hypertext Transfer Protocol (HTTP) requests from clients and send back a response containing a status code and often content such as HTML, XML or JSON as well.

Why are web servers necessary?

Web servers are the ying to the web client's yang. The server and client speak the standardized language of the World Wide Web. This standard language is why an old Mozilla Netscape browser can still talk to a modern Apache or Nginx web server, even if it cannot properly render the page design like a modern web browser can.

The basic language of the Web with the request and response cycle from client to server then server back to client remains the same as it was when the Web was invented by Tim Berners-Lee at CERN in 1989.

Web server implementations

The conceptual web server idea can be implemented in various ways. The following web server implementations each have varying features,
extensions and configurations.

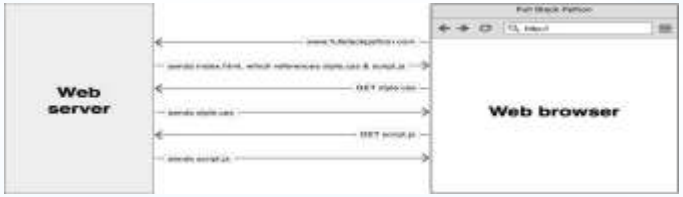
- The Apache HTTP Server has been the most commonly deployed web server on the Internet for 20+ years.
- Nginx is the second most commonly used server for the top 100,000 websites and often serves as a reverse proxy for Python WSGI servers.
- Caddy is a newcomer to the web server scene and is focused on serving the HTTP/2 protocol with HTTPS.
- rwasa is a newer web server written in Assembly with no external dependencies that tuned to be faster than Nginx.

Client requests

A client that sends a request to a web server is usually a browser such as Internet Explorer, Firefox, or Chrome, but it can also be a

- headless browser, commonly use for testing, such as phantomjs commandline utility, for example wget and cURL
- text-based web browser such as Lynx web crawler.

Web servers process requests from the above clients. The result of the web server's processing is a response code and commonly a content response. Some status codes, such as 204 (No content) and 403 (Forbidden), do not have content responses.



A web server sends files to a web browser based on the web browser's request. In the first request, the browser accessed the "www.fullstackpython.com" address and the server responded with the index.html HTML-formatted file. That HTML file contained references to other files, such as style.css and script.js that the browser then requested from the server.

Building web servers

Building a basic HTTP Server from scratch in Python (source code builds a very simple but insecure web server to show you how HTTP works.

I. MySQLdb Installation

II. MySQLdb Interface

CLIENT SERVER NETWORK:

Python – Database Access:

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

Working with Database using Python:

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

I. Connecting Python and MySQL Server

II. Accessing MySQLdb with Python

III. Sample Programs

Connecting Python and MySQL:

We can establish a connection between Python and MySQL using pip.

- pip is pre-installed in python 3.
- set the path for pip3 in environment variables.
- go to command prompt
- pip install mysql-connector-python

Accessing MySQL database with Python:

- Open any Python IDLE
- Import MySQL module import mysql.connector
- Connect to SQL server localhost root tiger world
- Create a cursor using cursor() function mycursor
- Execute the code using execute() function mycursor.execute("select *
- Terminate cursor and connection