Python Simple HTTP Server : A Simple HTTP Web Server With Python

This is a Python Simple HTTP Server Tutorial. Here we will learn how to create HTTP server in python.Creating web server in python is very easy, just a couple lines of code. So let's begin.

What Is Web Server

Overview

- A web server is actually a network application, running on some machine, listening on some port.
- Web server is a computer where web contents are stored.
- A web server serves web pages to clients across the internet or an intranet .
- It hosts the pages, scripts, programs and multimedia files and serve them using HTTP, a protocol designed to send files to web browsers.
- **Apache web server, IIS web server, Nginx web server, Light Speed web server** etc are the common examples of web servers.

Features

- Large data storage support
- Bandwidth controlling to regulate network traffic
- Site analysis

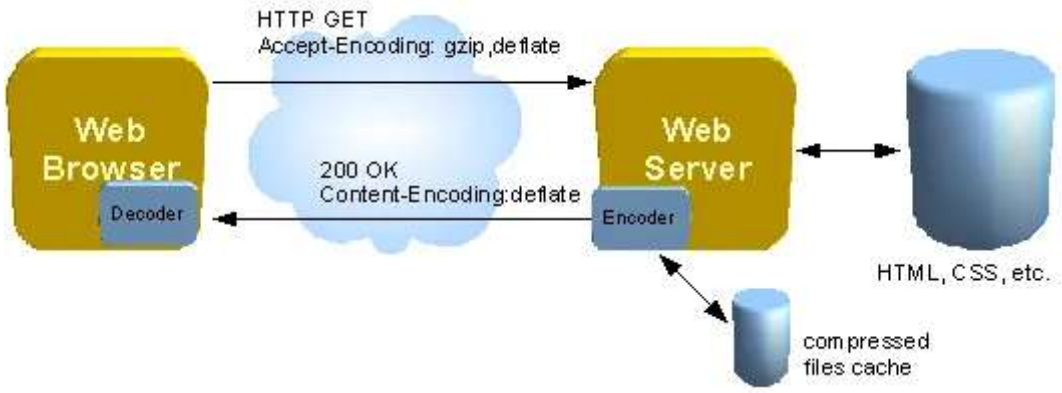
Types

There are two types of web servers –

- **Dedicated web servers :** In this, one web server is dedicated to a single user.This is suitable for websites with more web traffics.
- **Shared web servers :** In this, a web server is assigned to many clients and it shared among all the clients.

How Web Server Works?

This figure clearly depicts the working principle of Web servers.



Let's take a close look on its working –

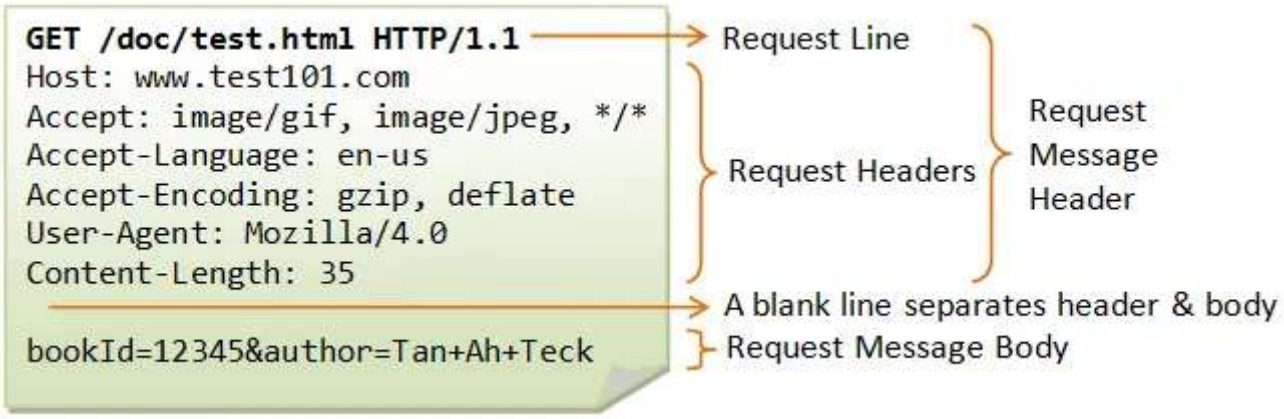
- For eg. a user wants to see a website like (www.google.com), the user type in the url the page using a client program(web browsers) but before this they need to physically connected i.e., the computer of the user and the web server, that's the job of Internet. Using the TCP/IP suite of protocol, it establishes the connection using a combination of cable media and wireless media.
- When the connection is establishes, the client sends a request called an http message and because the HTTP is a connectionless protocol, the clients disconnects from the server waiting for the response.
- The server on the other side process the requests, prepare the response, establishes the connection again and send back to response.
- Again inform a HTTP message to the client when the two computer is completely disconnect, that is the big general.

HTTP Protocol

The most important part of a web server is HTTP protocol.So now we will see what exactly this is –

- It stands for Hyper Text Transfer Protocol.
- It is an application layer protocol that allows web based applications to communicate and exchange data.
- The HTTP is the messenger of web.
- The computer that communicate via the HTTP must speak the HTTP protocol.
- It is a TCP/IP based protocol.
- It is used to deliver contents, for eg. images, audios, videos, documents etc.
- Using HTTP is the most convenient way to quickly and reliably move data on the web.

Example of an HTTP Message

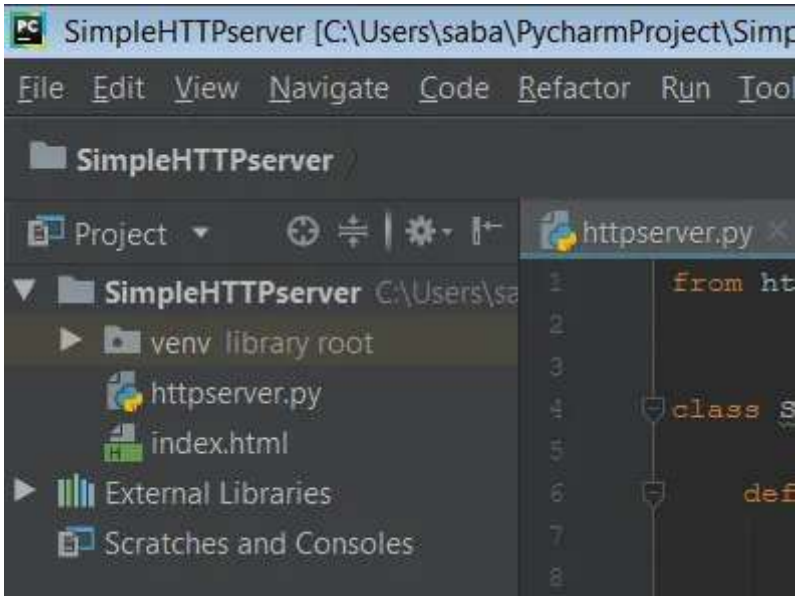


Python Simple HTTP Server Tutorial

Python actually comes with an built-in library just for creating a web server. Creating web server in python is very-very simple with just a couple lines of code. So let's start –

Creating a New Project

In your python IDE create a new project and inside this project create a python file like that –



Creating HTML file

First of all create an HTML file –

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Hello world!</title>
    </head>
    <body>
        <h1>Index page!</h1>
        <p>What Is Web Server

A web server is actually a network application, running on some machine, listening
on some port.
Web server is a computer where web contents are stored.
A web server serves web pages to clients across the internet or an intranet .
It hosts the pages, scripts, programs and multimedia files and serve them using
HTTP, a protocol designed to send files to web browsers.
            </p>
    </body>
</html>
```

Importing Modules

Python provides built-in http.server module so we need not to install any module using pip install.

http.server Module

- http.server is a python module which allow us to create web server.
- By using http.server, we can make any directory that you choose as your web server directory.

Importing Class

We have to import two class **HTTPServer** and **BaseHTTPRequestHandler**. So write the following codes.

```
from http.server import HTTPServer, BaseHTTPRequestHandler
```

- `HTTPServer` is a `socketserver.TCPServer` subclass. It creates and listens at the HTTP socket, dispatching the requests to a handler.
- **BaseHTTPRequestHandler** is used to handle the HTTP requests that arrive at the server. By itself, it cannot respond to any actual HTTP requests; it must be subclassed to handle each request method (e.g. GET or POST).`BaseHTTPRequestHandler` provides a number of class and instance variables, and methods for use by subclasses.

Creating a Server Holder Class

Now we need to create a class that holds the server. Give the class name web_server or anything you like. We pass **BaseHTTPRequestHandler** as a argument to this class that inherits the message from BaseHTTPRequestHandler.

Inside this class write the following code –

```
class web_server(BaseHTTPRequestHandler):

    def do_GET(self):
        if self.path == '/':
            self.path = '/index.html'
        try:
            #Reading the file
            file_to_open = open(self.path[1:]).read()
            self.send_response(200)
        except:
            file_to_open = "File not found"
            self.send_response(404)
```

```
self.end_headers()
 self.wfile.write(bytes(file_to_open, 'utf-8'))
```

What We Did?

- First of all we defined a method **do_GET().** This method runs when you send GET request. And anytime you go to a webpage in web-browser and type the address of that webpage, you sending a **GET** request to the server.
- **self.path == '/'** check the path of the request. We check if its a '/' that means they are on the index page.
- And if we run the index page, **self.path = '/index.html'** is the path for our index.html page.
- The next thing we have done is that we try to read the file that user trying to access.
- If the reading of file is successful then the try block ends successfully.
- So if the requested file is found then we send a **200** response. 200 response is response every webpage sends whenever you access a page successfully.
- Now inside the **except** block, we just print a message that file not found and this will execute whenever a user requests an invalid webpage.
- When file not find we just send a response **404** that means the file is not found.
- Then we have to send **self.end_headers()** message that is required by the BaseHTTPRequestHandler class. It sends a blank line, indicating the end of the HTTP headers in the response.

- And finally we write the context of a file on the screen. So to write on screen we have to converted the bytes so all over file we coded using UTF-8 so we converted the text into byte using the byte() method.

Creating HTTP Variable

Now finally we create a HTTP variable that instance of HTTP damon which is just call a program that runs on backend because that typically how web service run. Write the following code –

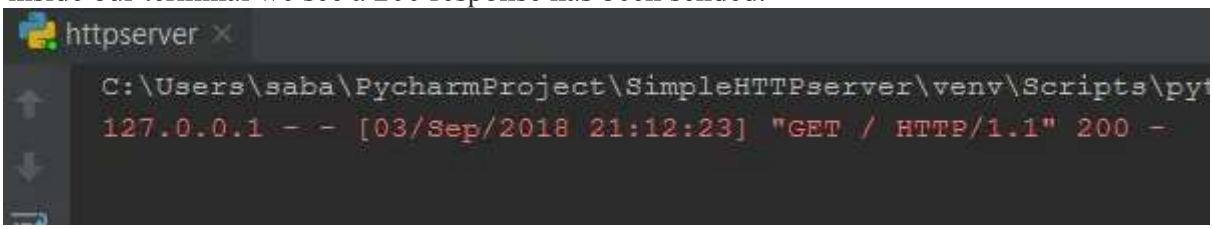
```
httpd = HTTPServer(('localhost', 8080), web_server)
httpd.serve_forever()
```

- Call HTTPServer class that python already implemented it for us.
- We pass localhost i.e., the address of our computer and the port on which we listen. And finally pass the **web_server** class which is we have created.
- Then finally run the serve_forever() method. It Handle requests until an explicit `shutdown()` request. Poll for shutdown every **poll_interval** seconds. Ignores the **`timeout`** attribute. If you need to do periodic tasks, do them in another thread.

Finally we have completed coding part successfully and now we will see its execution.

Now go to your browser and type **http://localhost:8080/** in the URL.And this way we get the following output –

And now inside our terminal we see a 200 response has been sended.



Congratulations we have successfully created our own simple HTTP web server and it is working perfectly.