


Let's Build A Web Server. Part 1.

(<https://ruslanspivak.com/lbaws-part1/>)

Date  Mon, March 09, 2015

Out for a walk one day, a woman came across a construction site and saw three men working. She asked the first man, "What are you doing?" Annoyed by the question, the first man barked, "Can't you see that I'm laying bricks?" Not satisfied with the answer, she asked the second man what he was doing. The second man answered, "I'm building a brick wall." Then, turning his attention to the first man, he said, "Hey, you just passed the end of the wall. You need to take off that last brick." Again not satisfied with the answer, she asked the third man what he was doing. And the man said to her while looking up in the sky, "I am building the biggest cathedral this world has ever known." While he was standing there and looking up in the sky the other two men started arguing about the errant brick. The man turned to the first two men and said, "Hey guys, don't worry about that brick. It's an inside wall, it will get plastered over and no one will ever see that brick. Just move on to another layer."¹

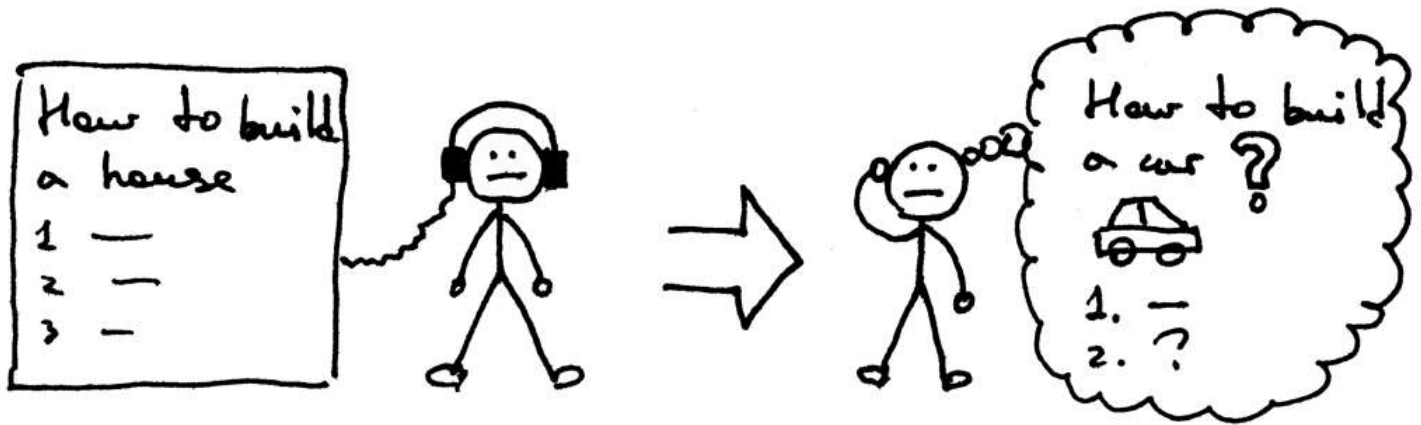
The moral of the story is that when you know the whole system and understand how different pieces fit together (bricks, walls, cathedral), you can identify and fix problems faster (errant brick).

What does it have to do with creating your own Web server from scratch?

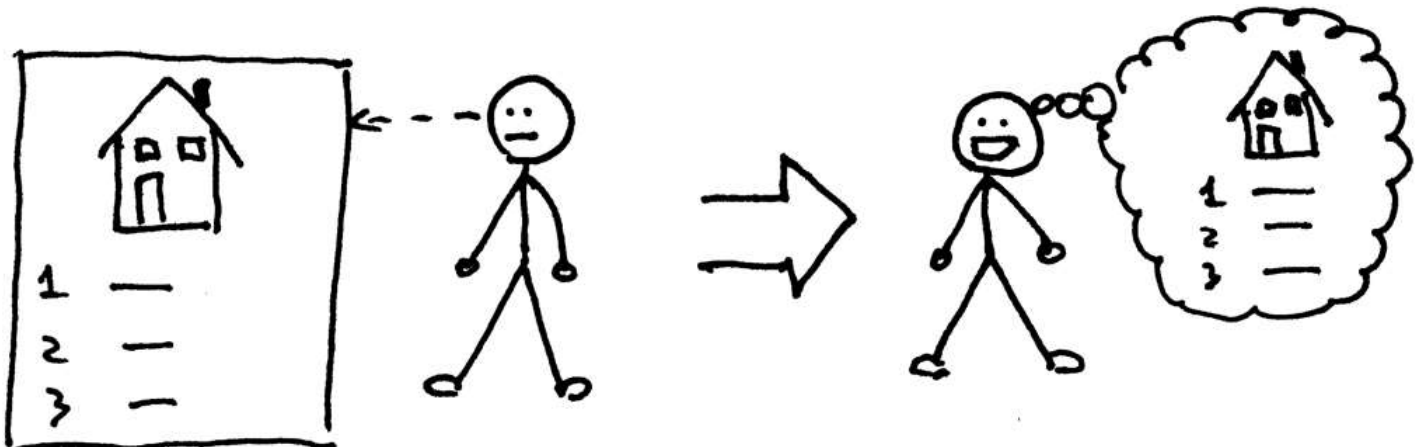
I believe to become a better developer you MUST get a better understanding of the underlying software systems you use on a daily basis and that includes programming languages, compilers and interpreters, databases and operating systems, web servers and web frameworks. And, to get a better and deeper understanding of those systems you MUST re-build them from scratch, brick by brick, wall by wall.

Confucius put it this way:

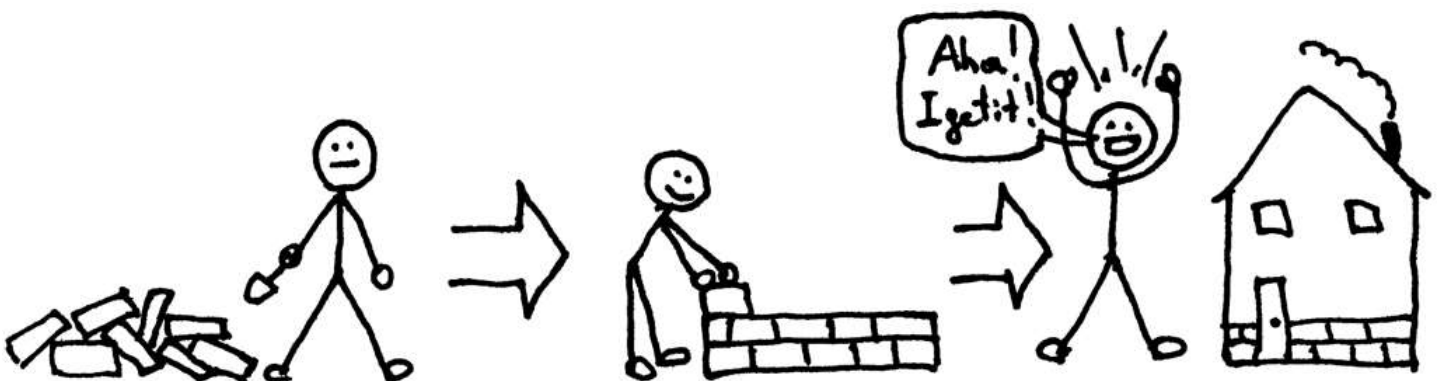
"I hear and I forget."



"I see and I remember."



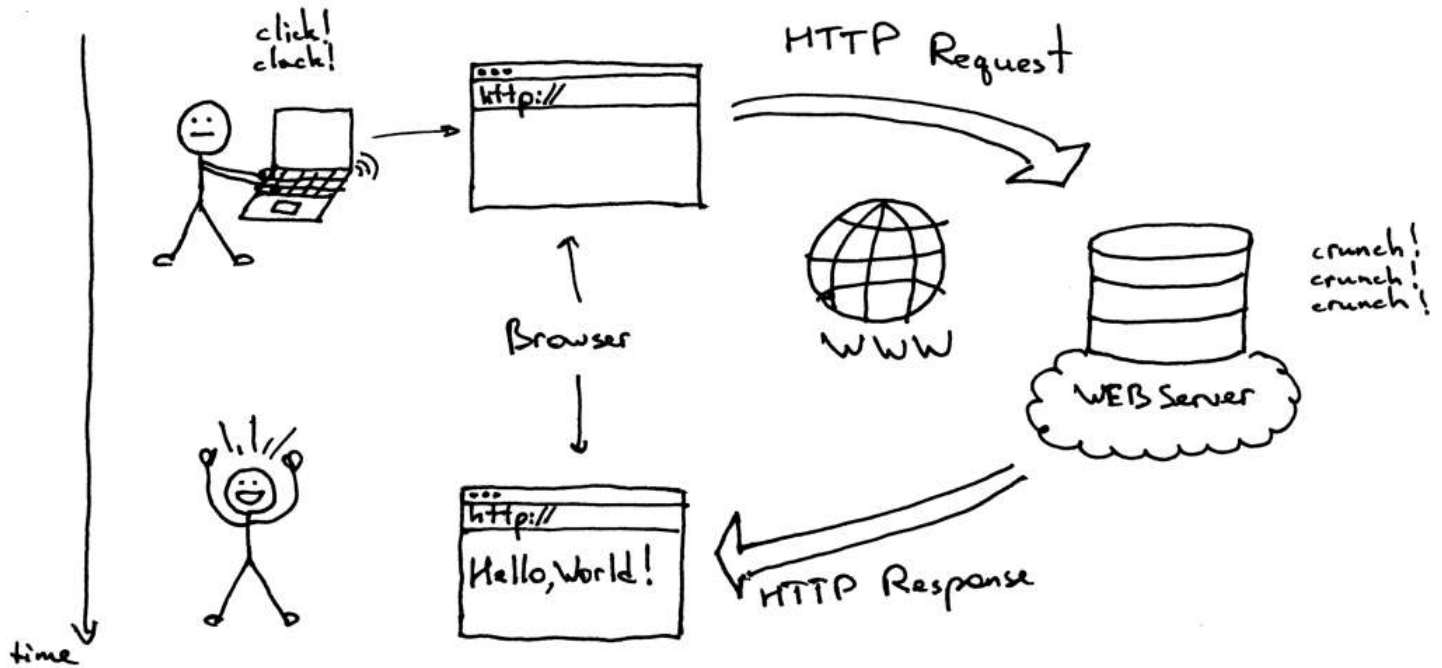
"I do and I understand."



I hope at this point you're convinced that it's a good idea to start re-building different software systems to learn how they work.

In this three-part series I will show you how to build your own basic Web server. Let's get started.

First things first, what is a Web server?



In a nutshell it's a networking server that sits on a physical server (oops, a server on a server) and waits for a client to send a request. When it receives a request, it generates a response and sends it back to the client. The communication between a client and a server happens using HTTP protocol. A client can be your browser or any other software that speaks HTTP.

What would a very simple implementation of a Web server look like? Here is my take on it. The example is in Python but even if you don't know Python (it's a very easy language to pick up, try it!) you still should be able to understand concepts from the code and explanations below:

```
import socket

HOST, PORT = '', 8888

listen_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
listen_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
listen_socket.bind((HOST, PORT))
listen_socket.listen(1)
print 'Serving HTTP on port %s ...' % PORT
while True:
    client_connection, client_address = listen_socket.accept()
    request = client_connection.recv(1024)
    print request

    http_response = """\
HTTP/1.1 200 OK

Hello, World!
"""

    client_connection.sendall(http_response)
    client_connection.close()
```

Save the above code as `webserver1.py` or download it directly from [GitHub](https://github.com/rspivak/lbaws/blob/master/part1/webserver1.py) (<https://github.com/rspivak/lbaws/blob/master/part1/webserver1.py>) and run it on the command line like this

```
$ python webserver1.py
Serving HTTP on port 8888 ...
```

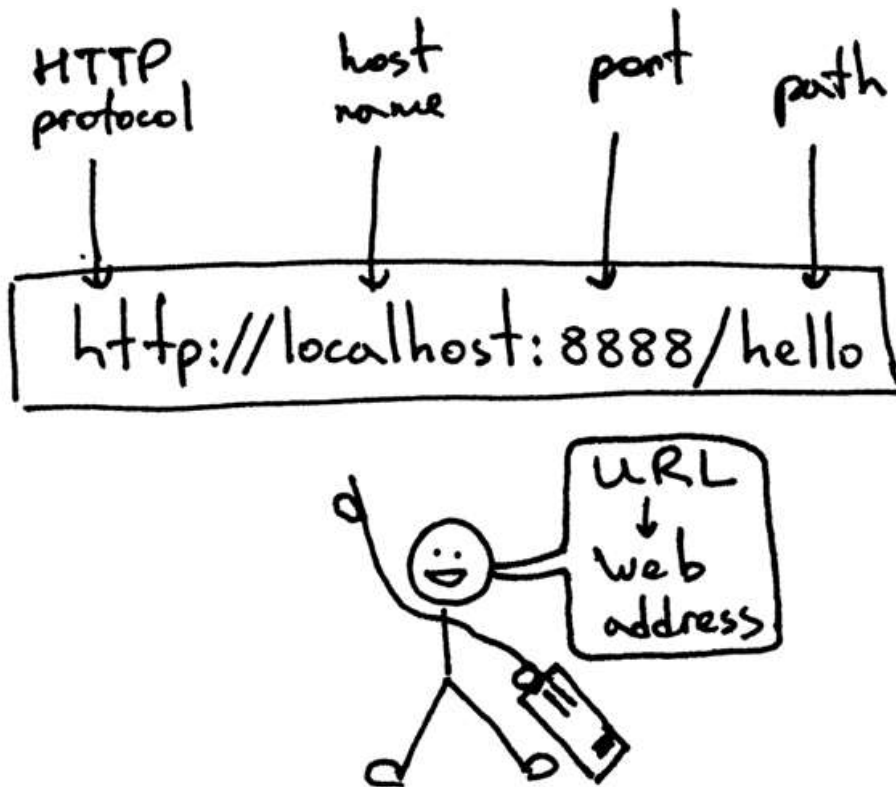
Now type in the following URL in your Web browser's address bar <http://localhost:8888/hello> (<http://localhost:8888/hello>), hit Enter, and see magic in action. You should see *"Hello, World!"* displayed in your browser like this:



Just do it, seriously. I will wait for you while you're testing it.

Done? Great. Now let's discuss how it all actually works.

First let's start with the Web address you've entered. It's called an URL (http://en.wikipedia.org/wiki/Uniform_resource_locator) and here is its basic structure:



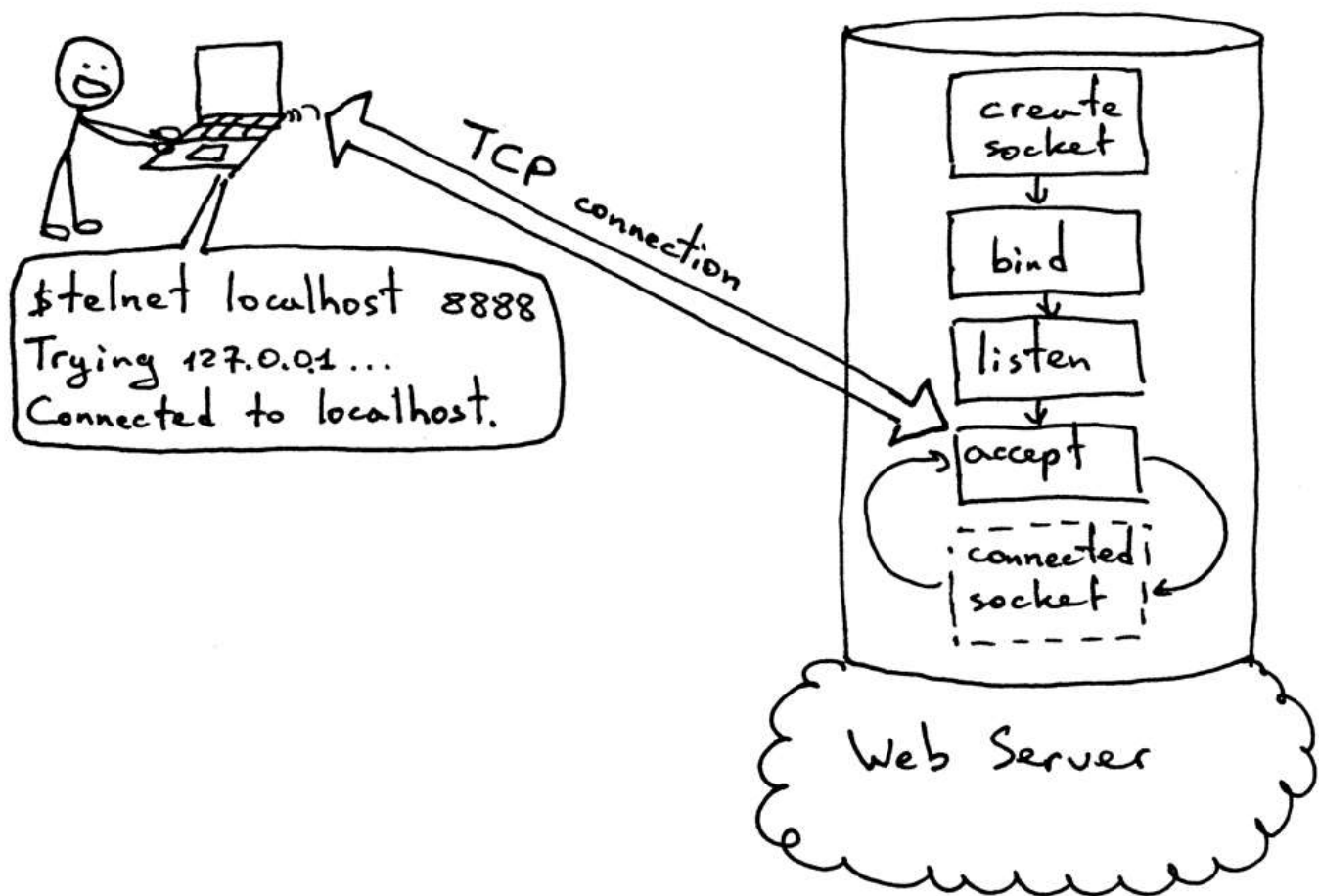
This is how you tell your browser the address of the Web server it needs to find and connect to and the page (path) on the server to fetch for you. Before your browser can send a HTTP request though, it first needs to establish a TCP connection with the Web server. Then it sends an HTTP request over the TCP connection to the server and waits for the server to send an HTTP response back. And when your browser receives the response it displays it, in this case it displays *"Hello, World!"*

Let's explore in more detail how the client and the server establish a TCP connection before sending HTTP requests and responses. To do that they both use so-called *sockets*. Instead of using a browser directly you are going to simulate your browser manually by using *telnet* on the command line.

On the same computer you're running the Web server fire up a telnet session on the command line specifying a host to connect to *localhost* and the port to connect to *8888* and then press Enter:

```
$ telnet localhost 8888
Trying 127.0.0.1 ...
Connected to localhost.
```

At this point you've established a TCP connection with the server running on your local host and ready to send and receive HTTP messages. In the picture below you can see a standard procedure a server has to go through to be able to accept new TCP connections.

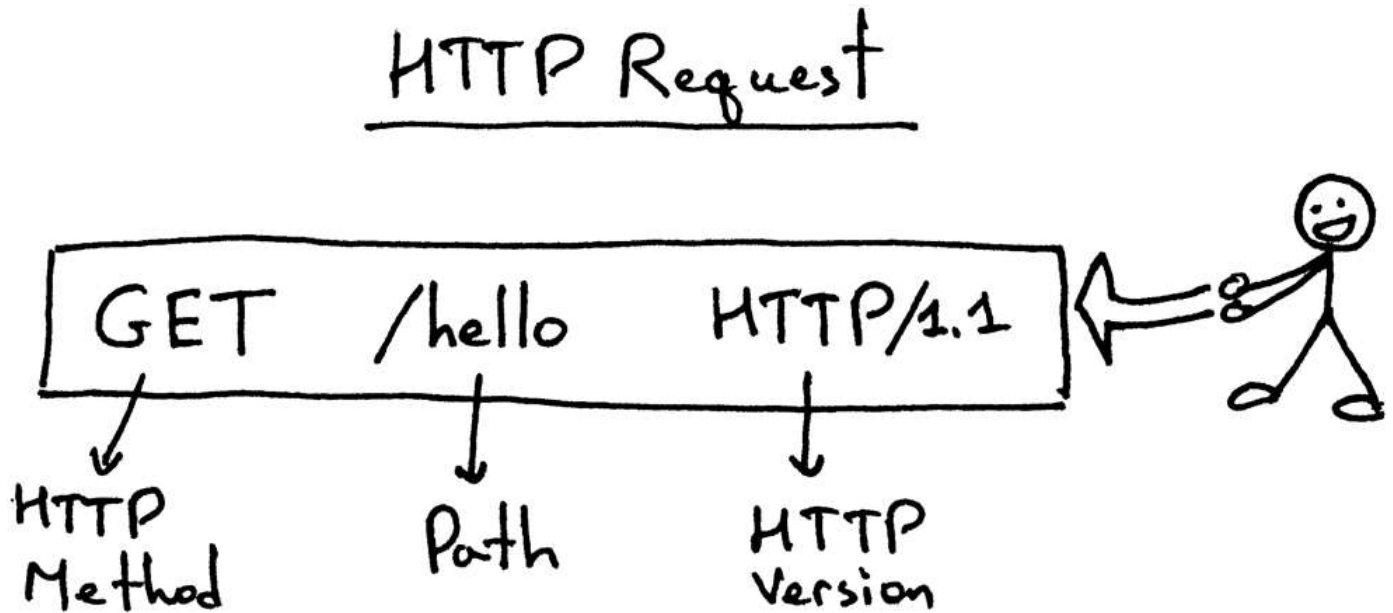


In the same telnet session type ***GET /hello HTTP/1.1*** and hit Enter:

```
$ telnet localhost 8888
Trying 127.0.0.1 ...
Connected to localhost.
GET /hello HTTP/1.1

HTTP/1.1 200 OK
Hello, World!
```

You've just manually simulated your browser! You sent an HTTP request and got an HTTP response back. This is the basic structure of an HTTP request:

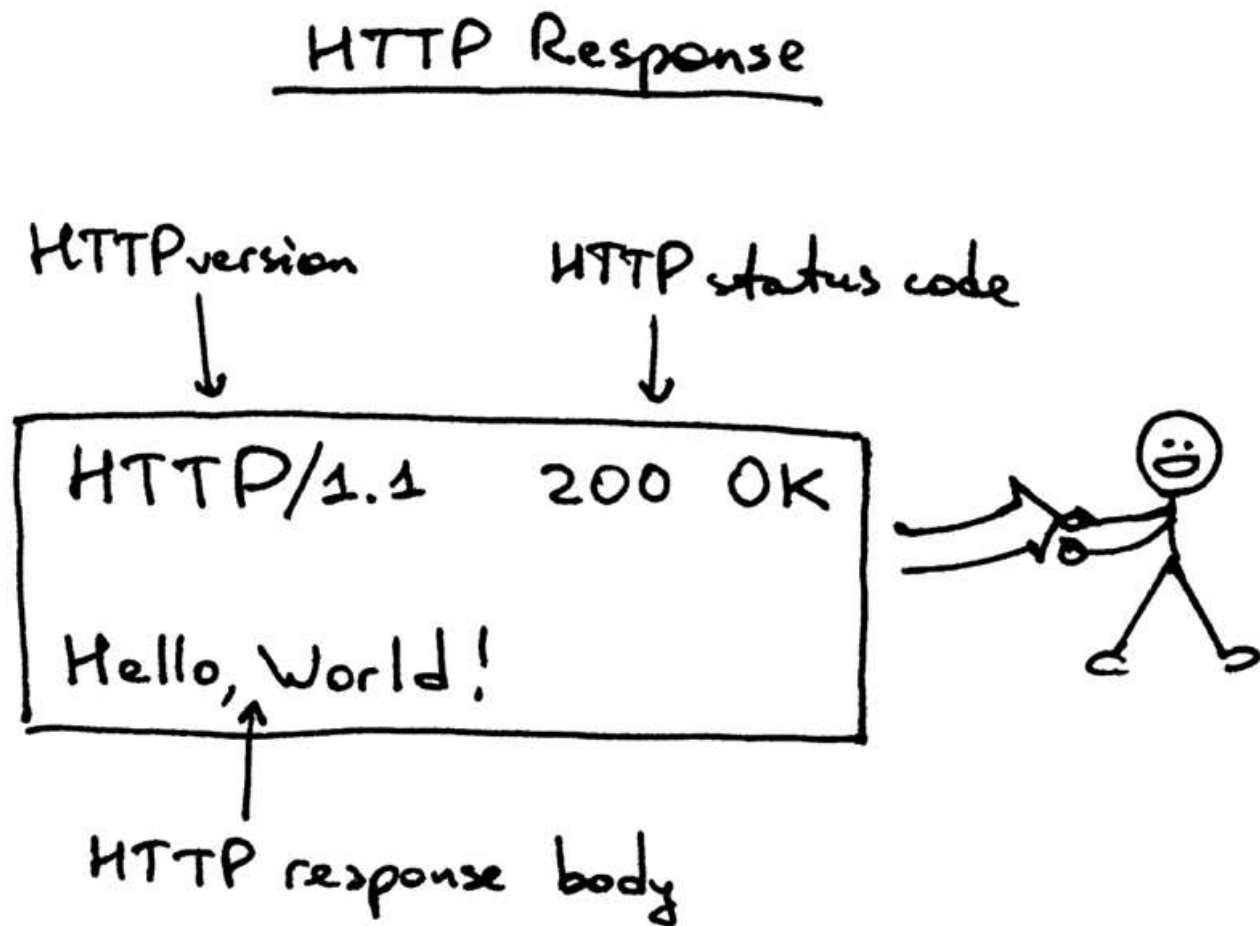


The HTTP request consists of the line indicating the HTTP method (**GET**, because we are asking our server to return us something), the path */hello* that indicates a “page” on the server we want and the protocol version.

For simplicity's sake our Web server at this point completely ignores the above request line. You could just as well type in any garbage instead of “GET /hello HTTP/1.1” and you would still get back a “Hello, World!” response.

Once you've typed the request line and hit Enter the client sends the request to the server, the server reads the request line, prints it and returns the proper HTTP response.

Here is the HTTP response that the server sends back to your client (*telnet* in this case):



Let's dissect it. The response consists of a status line *HTTP/1.1 200 OK*, followed by a required empty line, and then the HTTP response body.

The response status line *HTTP/1.1 200 OK* consists of the *HTTP Version*, the *HTTP status code* and the *HTTP status code reason* phrase *OK*. When the browser gets the response, it displays the body of the response and that's why you see "*Hello, World!*" in your browser.

And that's the basic model of how a Web server works. To sum it up: The Web server creates a listening socket and starts accepting new connections in a loop. The client initiates a TCP connection and, after successfully establishing it, the client sends an HTTP request to the server and the server responds with an HTTP response that gets displayed to the user. To establish a TCP connection both clients and servers use *sockets*.

Now you have a very basic working Web server that you can test with your browser or some other HTTP client. As you've seen and hopefully tried, you can also be a human HTTP client too, by using *telnet* and typing HTTP requests manually.

Here's a question for you: "How do you run a Django application, Flask application, and Pyramid application under your freshly minted Web server without making a single change to the server to accommodate all those different Web frameworks?"

I will show you exactly how in Part 2 of the series. Stay tuned.

BTW, I'm writing a book ***“Let's Build A Web Server: First Steps”*** that explains how to write a basic web server from scratch and goes into more detail on topics I just covered. Subscribe to the mailing list to get the latest updates about the book and the release date.

Enter Your First Name *

Enter Your Best Email *

Get Updates!

All articles in this series:

- Let's Build A Web Server. Part 1. (<http://ruslanspivak.com/lbaws-part1/>)
- Let's Build A Web Server. Part 2. (<http://ruslanspivak.com/lbaws-part2/>)
- Let's Build A Web Server. Part 3. (<http://ruslanspivak.com/lbaws-part3/>)

1. Inspired by [Lead with a Story: A Guide to Crafting Business Narratives That Captivate, Convince, and Inspire](http://www.amazon.com/gp/product/0814420303/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0814420303&linkCode=as2&tag=russblo0b-20&linkId=HY2LNXTSGPPFZ2EV) (http://www.amazon.com/gp/product/0814420303/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0814420303&linkCode=as2&tag=russblo0b-20&linkId=HY2LNXTSGPPFZ2EV)

Comments

78 Comments

Ruslan's Blog

 Rahul Vats ▾

♥ Recommend 37

🐦 Tweet

f Share

Sort by Best ▾



Join the discussion...



ChinaMoe • 4 years ago

how a great post!

80 ^ | ▾ • Reply • Share ›



LynnRice1 • 4 years ago

Last night I created an account here and left a message asking for help because the first exercise failed. Only difference I could see was that I was using python 3.4. (yes I Changed the print statements). Anyway, My message that was here last night is not showing up. So, maybe this one will. help

3 ^ | ▾ • Reply • Share ›



Charles MacKay → LynnRice1 • 2 years ago

is your error "TypeError: a bytes-like object is required, not 'str'?"

if so you have to change your http request to be a byte,
instead of string. can do this by `my_string.encode()`

change the send all request as follows:

```
client_connection.sendall(http_response.encode())
```

tested to be working on python 3.5

4 ^ | v • Reply • Share ›



Martin Breuss → LynnRice1 • 7 months ago

Another option is to prefix the response string with a "b", like
so:

```
http_response = b"""\n\nHTTP/1.1 200 OK
```

```
Hei there everyone!\n\n\""""
```

This converts it to a bytes object, which used to be the
standard for Python 2.x

^ | v • Reply • Share ›



Ankit Panwar • 2 years ago

hey ruslan nice post, when i am trying to run
`http://localhost:8888/hello` the browser doesn't show anything, i am
using linux machine and python 2.7.....

1 ^ | v • Reply • Share ›



TusharShivan • 2 years ago

Hi there,
Yesterday I wandered that how could I connect to my this newly
created server with my iPad or something so I tried to connect to
the server under same wifi network in which I created this server on
my laptop. I tried `http://localhost:8888/hello` and it didn't worked.

Any ideas on how to connect to this server?

1 ^ | v • Reply • Share ›



Daniel Hilpoltsteiner → TusharShivan • 2 years ago

using localhost here does not work. because localhost
refers to 127.0.0.1 which is your local address.
go find your ip adress in the network. (look in your router)
maybe it looks like 192.168.178.52 for example
consider using the `:8888/hello` at the end and you'll go fine

^ | v • Reply • Share ›



LynnRice1 • 4 years ago

I like this approach, thank you. However, could you get me past a
problem? I'm running Python 3.4 and

this line in `webserver1.py` fails:

```
client_connection.sendall(http_response)
```

The error is: `TypeError: 'str' does not support the buffer interface.`

Thanks in advance.

1 ^ | v • Reply • Share ›



rspivak Mod → LynnRice1 • 4 years ago

This should do the trick:

```
import socket

HOST, PORT = '', 8888

listen_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
listen_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
listen_socket.bind((HOST, PORT))
listen_socket.listen(1)
print("Serving HTTP on port %s ..." % PORT)
while True:
    client_connection, client_address = listen_socket.accept()
    request = client_connection.recv(1024)
    print(request.decode('utf-8'))

    http_response = """\
HTTP/1.1 200 OK

Hello, World!
"""
    client_connection.sendall(bytes(http_response, 'utf-8'))
    client_connection.close()
```

2 ^ | v • Reply • Share ›



LynnRice1 → rspivak • 4 years ago

Yes, perfect, Thank you very much and thanks for this series, was just what I was needing, it didn't feel right that things just magically worked, taking a look at things one level down is important to me. While I'm at it, could you give me a quick opinion about the tentative choices I've made for my tools. Flask, Python 3.4, Pycharm, Sublime text 2, VirtualEnv and I'm running from windows 7 and 8. Should I perhaps use VMware and Linux or can I get by? Two last things, 1) I'd like everything I do, from Hello World on to be "Responsive" by way of bootstrap and 2) Am I biting off too much in thinking about setting up another Machine acting as my Webserver here in my house so that I can do my learning and testing with everything under my full control. I have a domain name already and my neighbor is a professional server man.

In conclusion, I am concerned, as are many others, about starting out right. Any advice you can give me (and others listening) will be greatly appreciated. I do have 40 years experience programming and a lot of time and determination on my hands but have been learning Python for only about two years. Sorry for being long-winded, guess I'm suffering from 'no one to talk to about all this'.

If anyone has any suggestions for me please comment. thanks

1 ^ | v • Reply • Share ›



Jonathan Hartley → LynnRice1 • 4 years ago

Hey Lynn,

FWIW, No, I don't think you're biting off too much in running a separate webserver in your house for learning purposes. I used to do the same, and ran my personal web site off it for years. I recall at one point being very proud of having had 4 years continuous

uptime, even with me messing about on the server, installing and updating things all the time.

Of course, it means the computer has to be on 24/7 if you want your website to be up, but if it's quiet or tucked out of the way then that's easy enough.

It also means your website has to be served by unloading over your home internet

[see more](#)

1 ^ | v • Reply • Share ›



Balazs → Jonathan Hartley • 3 years ago

If you can't get a static IP, most ISP won't really like it (or you serving a webpage from home, but screw them right?). You can get a DynDNS service. nice free one:

<http://www.duckdns.org/>

^ | v • Reply • Share ›



Dylan → rspivak • 2 years ago

```
<client_connection.sendall(bytes(http_response, 'utf-8'))="">
```

Helped me with the error...

```
" TypeError: a bytes-like object is required, not 'str' "
```

^ | v • Reply • Share ›



Gercino Júnior → Dylan • 2 years ago

...

```
http_response = b"""\n\nHTTP/1.1 200 OK
```

```
Hello, World!
```

```
"""
```

...

Note the 'b' before <"""">

^ | v • Reply • Share ›



Milan → Dylan • 2 years ago

You may also use:

```
client_connection.sendall(http_response.encode('utf-8'))
```

^ | v • Reply • Share ›



eurico → rspivak • 4 years ago

Tks.

^ | v • Reply • Share ›



Migo • 4 years ago

This is great, checking back regularly for part 2.

1 ^ | v • Reply • Share ›



techstonia • 4 years ago

Waiting for part II & III. :-)

1 ^ | v • Reply • Share ›

**Kris** • 4 years ago

Good article. I've written one in C, so after just glancing over the code, I think you should show other's how to fork on accepting a request. Keep up the good articles!

1 ^ | v • Reply • Share ›

**rspivak** Mod ➔ Kris • 4 years ago

Hi Kris,

Thanks a lot for the suggestion! Do you happen to have your C code publicly available somewhere where I could take a look at it?

1 ^ | v • Reply • Share ›

**Srinivas Pithani** • 5 months ago

Just loved this article , one of the best . Thanks

^ | v • Reply • Share ›

**Chris Shyi** • 8 months ago

Excellent post Ruslan! I learned so much.

^ | v • Reply • Share ›

**TusharShivan** • 9 months ago

Hi,

What if anyone wants to connect this server, anywhere in the world? How can we do it?

Thanks

^ | v • Reply • Share ›

**gui** • a year ago

I really like those comics! It's so vivid!

^ | v • Reply • Share ›

**이민구** • a year ago

Oh my god this posting is just purely amazing.

Thank you very much for all your efforts.

May I translate it to Korean? It would be a great help to Korean students interested in this field.

^ | v • Reply • Share ›

**rspivak** Mod ➔ 이민구 • a year ago

You're welcome. Yes, you may translate it. If you don't mind, leave a comment with links to your translations, when they're ready. Thanks!

^ | v • Reply • Share ›

**이민구** ➔ rspivak • a year ago

No worries, thank you.

I'll leave my comment with the links as soon as I finish translating it!

^ | v • Reply • Share ›

**Gaurav Bhandari** • a year ago

import socket

I tried below, But response is not coming back on webpage. What I

am dng wring here?:

HOST, PORT = " , 8888

```
listen_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
listen_socket.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
listen_socket.bind((HOST, PORT))
listen_socket.listen(1)
print ('Host PORT: %s ...' % PORT)
while True:
client_connection, client_address = listen_socket.accept()
request = client_connection.recv(1024)
print (request.decode('utf-8'))
```

```
http_response = """\
HTTP/1.1 200 OK
Hello, World!
"""\
client_connection.sendall(bytes(http_response, 'utf-8'))
client_connection.close()
```

^ | v • Reply • Share ›



washolive → Gaurav Bhandari • 8 months ago

In http_response, it's required you put a blank line between
http header (HTTP/1.1 200 OK) and response printable
(Hello World).

^ | v • Reply • Share ›



Yunsoo Jung • a year ago

I love your Post

^ | v • Reply • Share ›



Halil İbrahim Oymacı • a year ago

In python 3, use must use this prin syntax:

```
print('Serving HTTP on port %s ...',PORT)
..
print(request)
```

And you must use byte for sendall method:

```
my_str_as_bytes = str.encode(http_response)
client_connection.sendall(my_str_as_bytes)
```

^ | v • Reply • Share ›



Rohit Naidu • a year ago

Par excellence

^ | v • Reply • Share ›



lemontea • a year ago

Really nice tutorial! And using Python for this is a good match. In
fact I have been inspired by your posts to write a kind of sequel:
Micro Web Framework in Python :)

^ | v • Reply • Share ›



Jerzy Drożdż • a year ago

Great job! This is what I looking for! Thanks a lot!



Great job. This is what I needed for. Thanks a lot.

^ | v • Reply • Share ›



vergil • a year ago

Hi Ruslan.

Just wanted to ask if you could provide an explanation for the source code. I am new to networking stuff and understanding all of this using the docs seems overwhelming. Thanks.

^ | v • Reply • Share ›



Jesús Alberto González Vences • 2 years ago

Hi there, I hope you read this, I want to translate this awesome work to Spanish, can I do it, I mean, do I have your permission? Can I borrow your illustrations? I'm intended to do it in my spare time, so it can take me quite some time.

^ | v • Reply • Share ›



rspivak Mod ➔ **Jesús Alberto González Vences** • 2 years ago

Sure. Good luck with your translation!

^ | v • Reply • Share ›



Koushtav Chakrabarty • 2 years ago

Nicely done article! :)

It would be better if you'd include a description of the python code and/or link to external resources about socket programming just in case someone very unfamiliar with sockets won't get a cold feet!

^ | v • Reply • Share ›



Vladimir Donets • 2 years ago

Отличный подход к статьям для начинающих. За Let's Build A Simple Interpreter отдельное спасибо! Освежает память.

^ | v • Reply • Share ›



Esteban • 2 years ago

Thanks for the tutorial!! it has been really helpful. I only have one inconvenient. I have tried to run the telnet command "telnet localhost 8888" and it freezes until I touch any other key. Then it prints the message but right after that it says "conection to host lost". Anyone knows how to fix this? I need to keep conection open so that I can send the GET requests.

Thanks for any help provided!!

^ | v • Reply • Share ›



Avatar

This comment was deleted.



Milan ➔ **Guest** • 2 years ago

You may use only GET as well.

^ | v • Reply • Share ›



Rajguru S J • 2 years ago

Hello sir, Can u plz suggest me how to implement web cache server?

^ | v • Reply • Share ›

**Adrian Vrabie** • 2 years ago

I enjoyed reading it! Regards from Moldova!

^ | v • Reply • Share ›

**rspivak** Mod ➔ Adrian Vrabie • 2 years ago

Thanks! :)

^ | v • Reply • Share ›

**Guillee191** • 2 years ago

how could I do this on w10?

^ | v • Reply • Share ›

**Joao Gabriel** • 3 years ago

Hi guys,

Is there a way to do this?

```
value = 100
http_response = ""
```

```
HTTP/1.1 200 OK
```

```
{"key": value}
```

```
""
```

I want to do this, because i'm passing a "json" and printing in the page, and then retrieving this json in a iOS app.

^ | v • Reply • Share ›

**Peter Raeth** • 3 years ago

Here is an update for Python v3.3

🏠 Social

 [github \(https://github.com/rspivak/\)](https://github.com/rspivak/) [twitter \(https://twitter.com/alienoid\)](https://twitter.com/alienoid) [linkedin \(https://linkedin.com/in/ruslanspivak/\)](https://linkedin.com/in/ruslanspivak/)

🏠 Popular posts

[Let's Build A Web Server. Part 1. \(https://ruslanspivak.com/lbaws-part1/\)](https://ruslanspivak.com/lbaws-part1/)[Let's Build A Simple Interpreter. Part 1. \(https://ruslanspivak.com/lbasi-part1/\)](https://ruslanspivak.com/lbasi-part1/)[Let's Build A Web Server. Part 2. \(https://ruslanspivak.com/lbaws-part2/\)](https://ruslanspivak.com/lbaws-part2/)[Let's Build A Web Server. Part 3. \(https://ruslanspivak.com/lbaws-part3/\)](https://ruslanspivak.com/lbaws-part3/)[Let's Build A Simple Interpreter. Part 2. \(https://ruslanspivak.com/lbasi-part2/\)](https://ruslanspivak.com/lbasi-part2/)

Disclaimer

Some of the links on this site have my Amazon referral id, which provides me with a small commission for each sale. Thank you for your support.

© 2017 Ruslan Spivak

[↑ Back to top](#)