让我们构建一个 Web 服务器。第1部分。 (https://ruslanspivak.com/lsbaws-part1/)

日期 2015 年 3 月 9 日，星期一

一天出去散步，一个女人在一个建筑工地上看到三个男人在工作。她问第一个男人：“你在做什么？” 被这个问题惹恼了，第一个男人咆哮道：“你没看到我在砌砖吗？” 她对这个回答不满意，问第二个人在做什么。第二个人回答说：“我在建一堵砖墙。” 然后，他把注意力转向第一个男人，说：“嘿，你刚刚穿过墙的尽头。你需要取下最后一块砖。” 再次对回答不满意，她问第三个人他在做什么。男人抬头仰望天空对她说：“我正在建造这个世界上最大的大教堂。” 当他站在那里抬头仰望天空时，另外两个人开始为那块错误的砖头争论起来。那人转向前两个男人说：“嘿伙计们，不用担心那块砖。这是一堵内墙，它会被抹上灰泥，没有人会看到那块砖。只需移至另一层即可。”

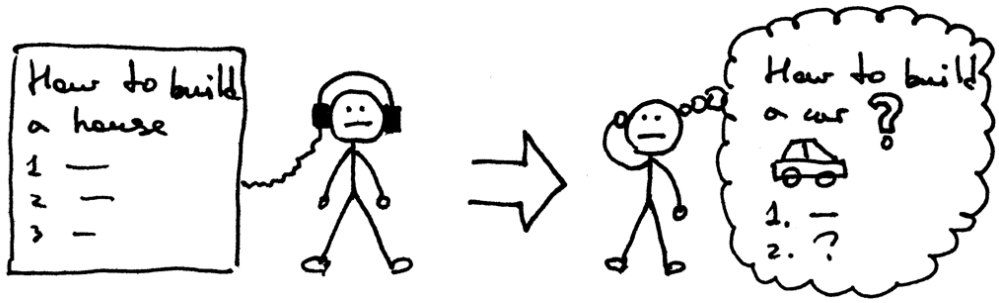
这个故事的寓意是，当您了解整个系统并了解不同部分（砖块、墙壁、大教堂）如何组合在一起时，您可以更快地识别和解决问题（错误的砖块）。

它与从头开始创建您自己的 Web 服务器有什么关系？

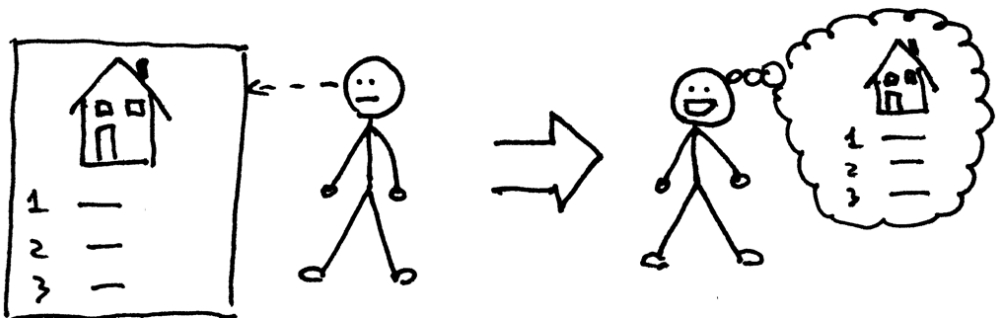
我相信要成为一名更好的开发人员，您必须更好地了解您每天使用的底层软件系统，其中包括编程语言、编译器和解释器、数据库和操作系统、Web 服务器和 Web 框架。而且，为了更好、更深入地了解这些系统，您必须从头开始，一砖一瓦，一墙一墙地重新构建它们。

孔子是这样说的：

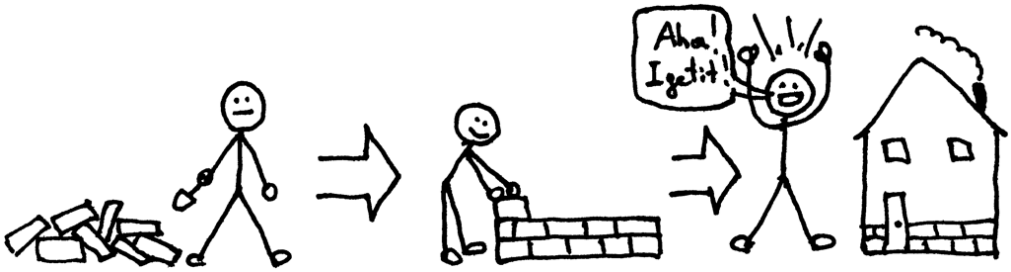
> “我听到了，我忘记了。”



> “我看到了，我记得。”



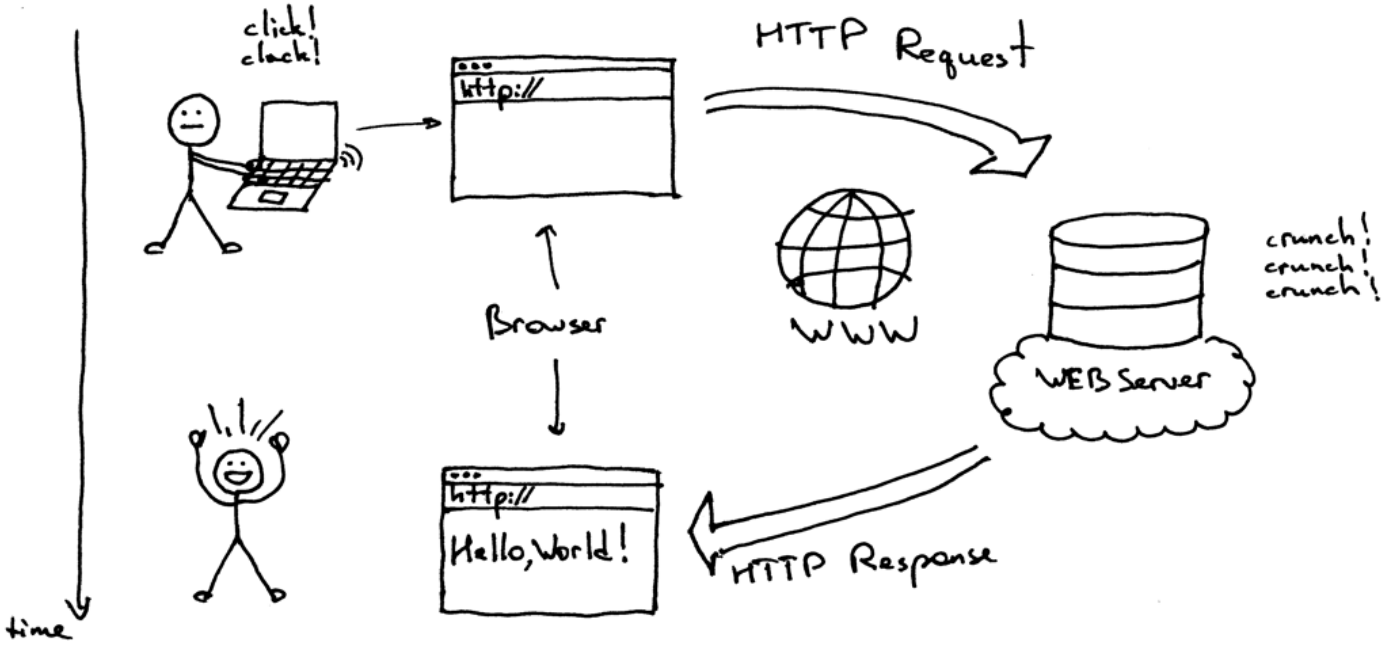
> “我知道，我明白。”



我希望此时您确信开始重新构建不同的软件系统以了解它们的工作方式是个好主意。

在这个由三部分组成的系列中，我将向您展示如何构建您自己的基本 Web 服务器。让我们开始吧。

首先，什么是Web 服务器?



简而言之，它是一个位于物理服务器上的网络服务器（哎呀，服务器上的服务器）并等待客户端发送请求。当它收到请求时，它会生成一个响应并将其发送回客户端。客户端和服务器之间的通信使用HTTP协议进行。客户端可以是您的浏览器或任何其他支持HTTP 的软件。

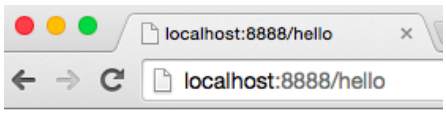
一个非常简单的 Web 服务器实现会是什么样的？这是我的看法。这个例子是用 Python 编写的（在 Python3.7+ 上测试过），但即使你不知道 Python（它是一种非常容易上手的语言，试试吧！）你仍然应该能够从下面的代码和解释中理解概念：

```
# Python3.7+
导入 套接字

主机, 端口 =  '' ,  8888

listen_socket  = 套接字。插槽（插座。AF_INET ，  插座。SOCK_STREAM ）
listen_socket 。setsockopt的（插座。SOL_SOCKET ，  插座。SO_REUSEADDR ，  1 ）
listen_socket 。绑定（（主机，  端口））
listen_socket 。监听( 1 )
打印( f 'Serving HTTP on port {PORT} ...' )
而 真:
    client_connection ，  client_address  =  listen_socket 。接受（）
    request_data  =  client_connection 。recv ( 1024 )
    打印( request_data . decode ( 'utf-8' ))

    http_response  =  b """ \
HTTP/1.1 200 OK

你好，世界！
"""
    client_connection 。sendall （HTTP_RESPONSE ）
    client_connection 。关闭（）
```

将上面的代码另存为webserver1.py或者直接从GitHub (https://github.com/rspivak/lsbaws/blob/master/part1/webserver1.py)下载，然后像这样在命令行上 (https://github.com/rspivak/lsbaws/blob/master/part1/webserver1.py)运行

```
$ python webserver1.py
在端口8888上提供 HTTP 服务……
```

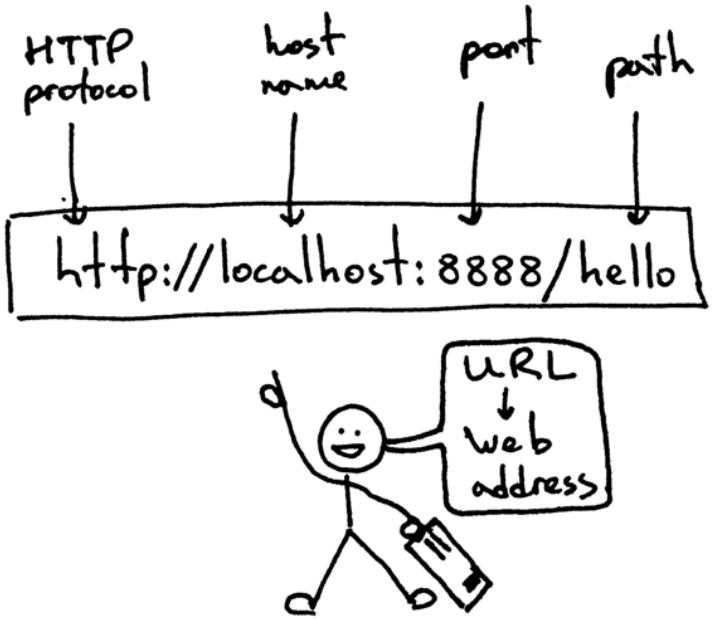
现在在您的 Web 浏览器的地址栏中输入以下URL http://localhost:8888/hello (http://localhost:8888/hello)，按 Enter 键，然后看看魔法在起作用。你应该看到“Hello, World!” 在浏览器中显示如下：



就去做吧，认真的。当你测试它时，我会等你。

完毕？伟大的。现在让我们讨论一下这一切是如何运作的。

首先让我们从您输入的 Web 地址开始。它被称为URL (http://en.wikipedia.org/wiki/Uniform_resource_locator)，这是它的基本结构：



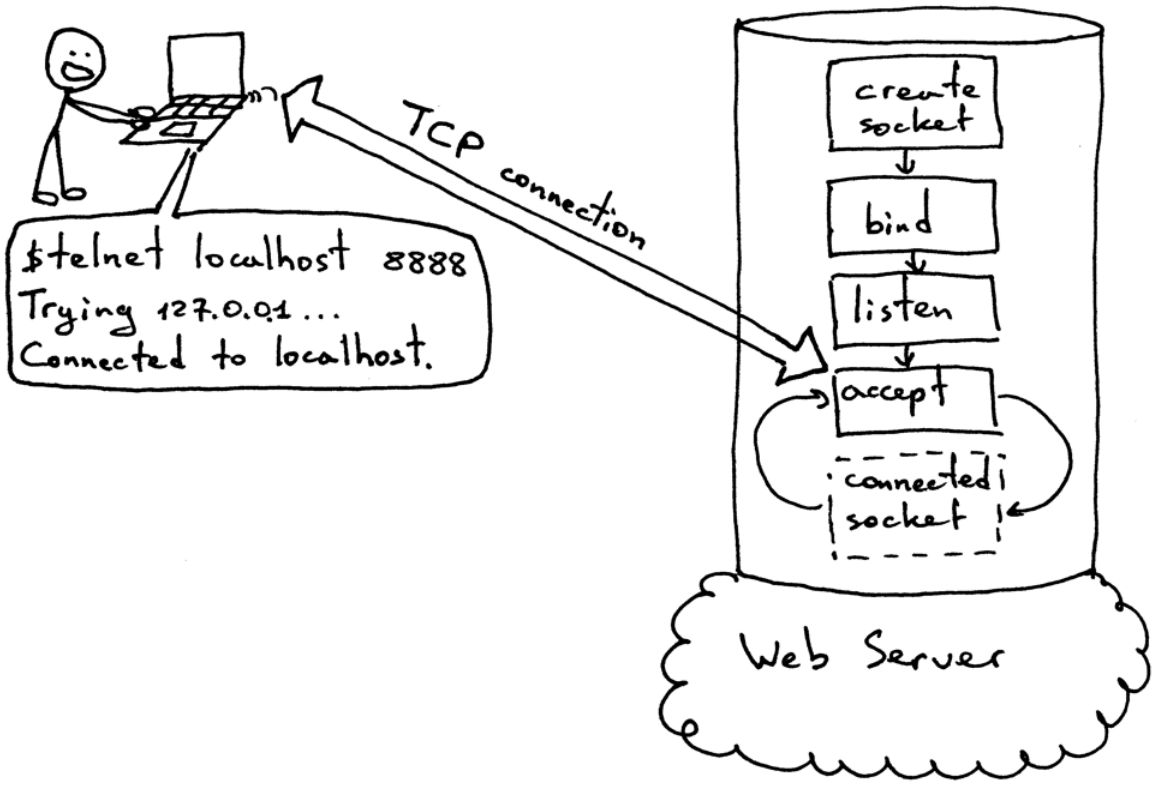
这就是您告诉浏览器它需要查找和连接的 Web 服务器的地址以及要为您获取的服务器上的页面（路径）的方式。不过，在您的浏览器可以发送HTTP请求之前，它首先需要与 Web 服务器建立TCP连接。然后它通过TCP连接向服务器发送HTTP请求，并等待服务器发回HTTP响应。当您的浏览器收到响应时，它会显示它，在这种情况下，它会显示“Hello, World！”

让我们更详细地探讨客户端和服务器如何在发送HTTP请求和响应之前建立TCP连接。为此，他们都使用所谓的sockets。您将通过在命令行上使用telnet手动模拟浏览器，而不是直接使用浏览器。

在您运行 Web 服务器的同一台计算机上，在命令行上启动一个 telnet 会话，指定连接到localhost 的主机和连接到8888的端口，然后按 Enter：

```
$ telnet localhost 8888
尝试127 .0.0.1 ...
连接到本地主机。
```

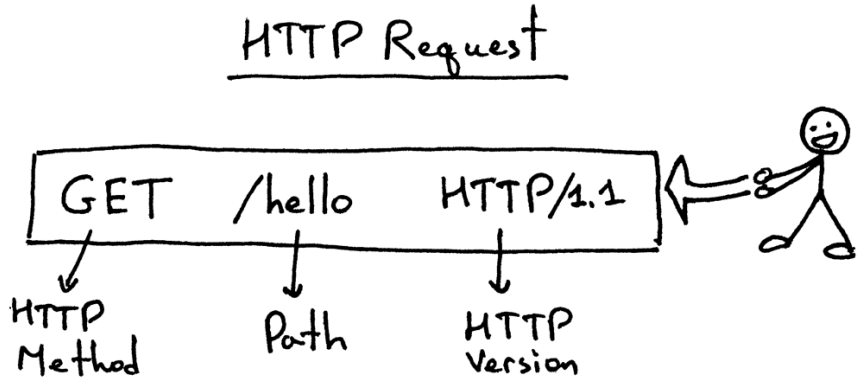
此时，您已经与在本地主机上运行的服务器建立了TCP连接，并准备好发送和接收HTTP消息。在下图中，您可以看到服务器必须经过的标准程序才能接受新的TCP连接。



在同一个 telnet 会话中键入**GET /hello HTTP /1.1**并按 Enter：

```
$ telnet localhost 8888
尝试127 .0.0.1 ...
连接到本地主机。
获取 /hello HTTP/1.1

HTTP/1.1 200正常
你好，世界！
```

您刚刚手动模拟了您的浏览器！您发送了一个HTTP请求并得到了一个HTTP响应。这是HTTP 请求的基本结构：

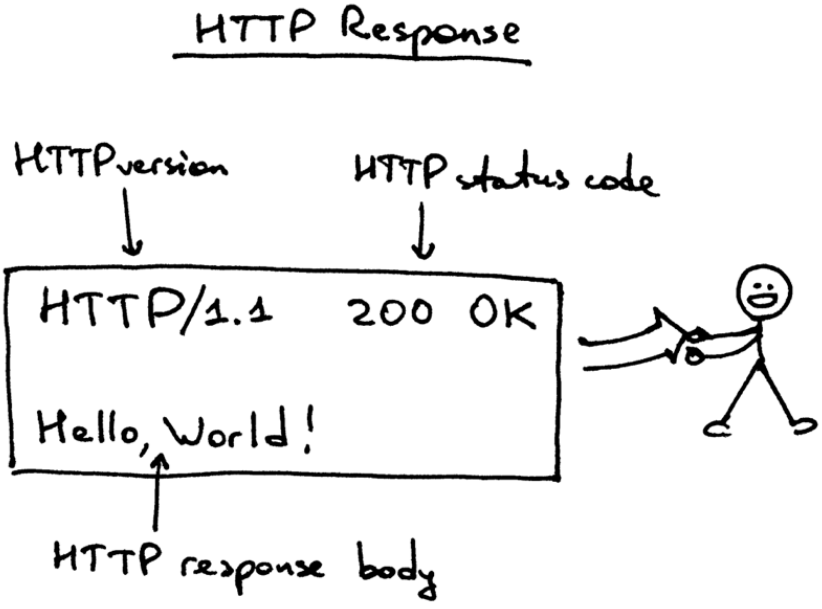


在HTTP请求由指示该行的HTTP方法（**GET**，因为我们要求我们的服务器返回我们的东西），路径/你好，指示的“页面”，我们希望在服务器和协议版本上。

为简单起见，此时我们的 Web 服务器完全忽略了上述请求行。你也可以输入任何垃圾而不是“ GET /hello HTTP /1.1”，你仍然会得到一个“Hello, World！” 回复。

输入请求行并按 Enter 键后，客户端将请求发送到服务器，服务器读取请求行，打印它并返回正确的HTTP 响应。

这是服务器发送回客户端的HTTP响应（在本例中为telnet）：



我们来剖析一下。响应包含一个状态行HTTP /1.1 200 OK，后跟一个必需的空行，然后是HTTP响应正文。

响应状态行HTTP /1.1 200 OK由HTTP版本、HTTP状态代码和HTTP状态代码原因短语OK 组成。当浏览器收到响应时，它会显示响应的正文，这就是您看到“Hello, World!”的原因。在您的浏览器中。

这就是 Web 服务器如何工作的基本模型。总结一下：Web 服务器创建一个侦听套接字并开始在循环中接受新连接。客户端发起TCP连接，成功建立连接后，客户端向服务器发送HTTP请求，服务器响应显示给用户的HTTP响应。要建立TCP连接，客户端和服务器都使用套接字。

现在您有一个非常基本的 Web 服务器，您可以使用浏览器或其他一些HTTP客户端对其进行测试。正如您已经看到并希望尝试过的，您也可以通过使用telnet并手动键入HTTP请求来成为人类HTTP客户端。

这里有一个问题要问你：“你如何在新创建的 Web 服务器下运行 Django 应用程序、Flask 应用程序和 Pyramid 应用程序，而无需对服务器进行任何更改以适应所有这些不同的 Web 框架？”

我将在本系列的第 2 部分中 (https://ruslanspivak.com/lsbaws-part2/)向您展示具体方法。敬请关注。

用于准备本文的资源（链接是附属链接）：

1. Unix 网络编程，第 1 卷：套接字网络API（第 3 版） (http://www.amazon.com/gp/product/0131411551/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0131411551&linkCode=as2&tag=russblo0b-20&linkId=2F4NYRBND566JJQL)
2. UNIX环境中的高级编程，第 3 版 (http://www.amazon.com/gp/product/0321637739/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0321637739&linkCode=as2&tag=russblo0b-20&linkId=3ZYAKB537G6TM22J)
3. Linux 编程接口：Linux 和UNIX系统编程手册 (http://www.amazon.com/gp/product/1593272200/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1593272200&linkCode=as2&tag=russblo0b-20&linkId=CHFOMNYXN35I2MON)
4. 用故事引导 (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)

> **更新：2019 年 7 月 13 日星期六**
>
> - 更新了服务器代码以在 Python 3.7+ 下运行
> - 添加了用于准备文章的资源

本系列所有文章：

- 让我们构建一个 Web 服务器。第1部分。 (https://ruslanspivak.com/lsbaws-part1/)
- 让我们构建一个 Web 服务器。第2部分。 (https://ruslanspivak.com/lsbaws-part2/)
- 让我们构建一个 Web 服务器。第 3 部分。 (https://ruslanspivak.com/lsbaws-part3/)

注释

Disqus 似乎比平时花费的时间更长。重新加载?

由Disqus提供支持的评论 (http://disqus.com)

社会的

- github (https://github.com/rspivak/)
- 推特 (https://twitter.com/rspivak)
- 链接 (https://linkedin.com/in/ruslanspivak/)

热门帖子

- 让我们构建一个 Web 服务器。第1部分。 (https://ruslanspivak.com/lsbaws-part1/)
- 让我们构建一个简单的解释器。第1部分。 (https://ruslanspivak.com/lsbasi-part1/)
- 让我们构建一个 Web 服务器。第2部分。 (https://ruslanspivak.com/lsbaws-part2/)
- 让我们构建一个 Web 服务器。第 3 部分。 (https://ruslanspivak.com/lsbaws-part3/)
- 让我们构建一个简单的解释器。第2部分。 (https://ruslanspivak.com/lsbasi-part2/)

免责声明

这个网站上的一些链接有我的亚马逊推荐 ID，它为我提供了每次销售的小额佣金。感谢您的支持。

回到顶部