Web 服务器与 Tornado 入门

> **Organization**: 千锋教育 Python 教学部
>
> **Date** : 2019-09-08
>
> **Author**: 张旭

一、HTTP 服务器的真相

HTTP 协议是建立在 TCP 协议之上的短连接协议。

它利用了 TCP 协议的可靠性，用来传输超文本 (HTML)，通信一次连接一次，通信完成后 TCP 连接关闭。

所以如果想创建一个 HTTP Server 需要通过 Socket 搭建一个 服务端程序。

1. 最简单的 HTTP Server

```
import socket

ADDR = ('0.0.0.0', 80)

RESPONSE = b'''
HTTP/1.1 200 OK
<!DOCTYPE html>
<html>
    <head>
        <title>Hello</title>
    </head>
    <body>
        <h1 style="text-align: center;">Hello World</h1>
    </body>
</html>
'''

listen_socket = socket.socket()  # 建立 SOCK 连接
listen_socket.bind(ADDR)         # 绑定 IP:端口
listen_socket.listen(100)        # 开始监听

print('Server is running: %s:%s ...' % ADDR)

while True:
    client_socket, client_address = listen_socket.accept()  # 接受客户端的连接
    print('client request from %s:%s' % client_address)
    request = client_socket.recv(1024)  # 接收客户端数据
```

```
    http_response = RESPONSE

    client_socket.sendall(http_response)
    client_socket.close()
```

2. 对 SimpleServer 进行扩展

1. 根据不同 URL 显示不同页面
2. 页面整体样式不变，根据不同参数，从数据库中取出不同学生信息，并填充到页面中

二、Web 框架概述

随着技术的发展，我们每天的要处理的信息量都在爆炸新的增加。传统的静态页面技术早已跟不上时代需求，因而催生了动态页面技术。

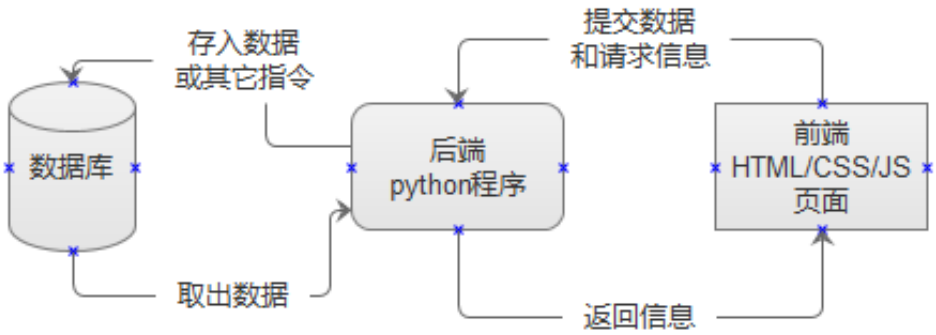
所谓动态页面，即所有的页面用程序来生成，以细节实现上的不同，又可分为“前端动态页面”和“后端动态页面”。

我们在 Web 前端阶段所学 Ajax、VUE 等技术，就是前端动态页面。而今后我们所学的主要是后端动态页面技术，甚至是两者结合使用。

1. Web服务器原理

第一小节的 SimpleServer 虽然代码仅仅 30 多行，但已经包含了一个小型 Web 系统的核心。

一个完备的 Web 系统如下图所示:



2. 常见的 Web 框架

如果想完成更复杂的功能，还需要深入开发很多东西，比如模版系统、ORM系统、路由系统、会话机制……

好在这些基础且通用的东西已经被很多前辈开发完成，我们没有必要再造轮子。他们按照自己的用途和想法，将各种系统开发、组合，最终为我们提供了各种各样的 Web 框架。

Web Framework	Description
Django	全能型框架, 大而全, 插件丰富, 文档丰富, 社区活跃, 适合快速开发, 内部耦合比较紧
Flask	微型框架, 适合新手学习, 极其灵活, 便于二次开发和扩展, 生态环境好, 插件丰富
Tornado	异步处理, 事件驱动 (epoll), 性能优异
Bottle	单文件框架, 结构紧凑，适合初学者阅读源码，了解 Web 原理
web.py	代码优美, jie适合学习源码
Falcon	性能优异适合写 API 接口
Quixote	一个爷爷级别的框架，著名的豆瓣网用的便是这个
Sanic	后起之秀，性能秒杀以上所有前辈，但没有前辈们稳定。

三、Tornado 入门

Tornado 是有 FriendFeed 公司开发的 Web 框架，该公司已经于 2009 年被 Facebook 收购，原本的 FriendFeed 已经成为了 Facebook 的一部分。

Tornado 最大的特点就是他实现了一个 “异步非阻塞” 的 HTTP Server，性能非常优异。

1. 安装

```
pip install tornado
```

2. Hello World

```
import tornado.ioloop
import tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")

def make_app():
    return tornado.web.Application([
        (r"/", MainHandler),
    ])

if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
```

```
    tornado.ioloop.IOLoop.current().start()
```

3. 启动参数

```
from tornado.options import parse_command_line, define, options

define("host", default='0.0.0.0', help="主机地址", type=str)
define("port", default=8888, help="主机端口", type=int)

parse_command_line()
print('你传入的 host: %s' % options.host)
print('你传入的 port: %s' % options.port)
```

4. 路由处理

```
import tornado.ioloop
from tornado.web import RequestHandler, Application

class HomeHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("欢迎进入主页")

class BookHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("你想看的书应有尽有")

app = Application([
    ('/', HomeHandler),
    ('/book/', BookHandler),
])

app.listen(8000)
tornado.ioloop.IOLoop.current().start()
```

5. 处理 GET 和 POST 请求

```
class StoryHandler(tornado.web.RequestHandler):
    stories = {1: '小红帽', 2: '皮诺曹', 3: '阿拉丁神灯'}

    def get(self):
        story_id = self.get_argument('story_id')
        story = self.stories[story_id]
        self.write("你想看 %s 的故事" % story)

    def post(self):
        pass
```

HTTP 的请求方法:

Method	Description
POST	向指定资源提交数据进行处理请求, 数据被包含在请求体中。
GET	请求指定的页面信息，并返回实体主体。
PUT	从客户端向服务器传送的数据取代指定的文档的内容。
DELETE	请求服务器删除指定的页面。
HEAD	类似于 GET 请求，只不过返回的响应中没有具体的内容，用于获取报头
PATCH	是对 PUT 方法的补充，用来对已知资源进行局部更新 。
OPTIONS	允许客户端查看服务器的性能。

6. 练习

1. 在 MySQL 中建立 user 表, 包含字段: id, name, age, sex, city 等, 并插入若干数据
2. 开发接口，并根据用户传入的 id 显示对应的用户信息
3. 开发接口，并根据用户传入的数值修改用户数据