

2.3 用 Apache HTTP Server 和 Flask 开设 WWW 服务

2.3.1 实验环境

操作系统：各大主流操作系统。本实验暂时使用 Windows 7(32bit)完成。

所需软件：

- (1) Apache HTTP Server v2.4
- (2) Python 3.8
- (3) Flask

2.3.2 实验步骤

- (1) 使用 Python 在示例项目的基础上编写代码，完成网页计数器和显示图片功能。
- (2) Apache 服务器的运行和配置，及 Python WSGI 的配置，使得可以通过 Apache 服务器在 80 端口上对外提供 WWW 服务。

2.3.3 实验思考

- (1) 在 conf/httpd.conf 中，了解其他选项，比如，如何修改服务器 IP 地址和端口，默认首页文件名，如何设置虚拟目录等。
- (2) Nginx 是近年来另一款非常流行的 HTTP 服务器。请比较 Apache HTTP Server 与 Nginx 的区别和优劣。
- (3) 在访问量统计功能中，用户的访问数据是被存储在数据库里的，这里我们使用的是基于文件的数据库 SQLite。数据存储在哪个文件中？阅读代码，visitor 数据表中包含了哪些信息？
- (4) 除了 Flask 外，还有哪些 Web 编程框架？请举出至少三个，并分别简述其使用的编程语言和主要特点。

2.3.4 附录：实验参考资料

1. Python 的环境配置与运行

通常，你可以在 Python 官网(<https://www.python.org/downloads/>)下载到 Python。对于 Windows，你可以下载到两种类型：installer 和 embeddable package。installer 指的是安装程序，.exe 格式，需要按照引导安装后才能运行。而 embeddable package 下载到的则是.zip 文件，解压后直接执行其中的 python.exe 文件即可直接运行。为了给同学们带来方便，本实验中我们选择的是第二种。

Python 程序往往要用到大量的依赖，Python 有一套官方维护的依赖索引，称作 PyPI(Python Package Index)。凡是发布在 PyPI 上的包，均可以方便地通过 pip install

命令下载并安装到本地的 Python 中,例如: `pip install Flask`。(注意官网下载的 installer 是自带 pip 的,但 embeddable package 不自带 pip,需要用 get-pip 等方法手动安装)。

在本实验中,为了方便同学们,我们发布给大家的压缩包不但安装好了 pip,也安装好了 Flask 等依赖,因此同学们直接解压该压缩包即可。

在解压好的压缩包中,直接双击 `python.exe`,可以启动一个终端窗口,这是一个交互式的 Python 命令行,如图 1 所示。输入 `exit()` 或 `ctrl+Z` 可以退出 Python。

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May 3 2021, 11:34:34) [MSC v.1928 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.platform
'win32'
>>>
```

图 1 交互式 Python 命令行

这种方法虽然方便,但难以给程序传入启动参数,因此更常用的方式是通过命令行启动,接下来将重点介绍通过命令行启动 Python 的方式。

(1) 按快捷键 Win+R (或点击开始-运行),输入 `cmd` 后回车,打开终端。(注:下文提到“打开终端”或“在终端中执行”时,均指这一步骤;这里使用的是命令提示符,你也可以自由选用其他终端如 powershell)

(2) 使用 `cd` 命令切换到 `python3.8` 所在的目录,如图 2 第一条命令所示。(注:实验环境实际的目录可能与图 2 中有所区别)

(3) 输入命令 `python` 可启动交互式命令行,输入 `python xxx.py` 可执行特定的 python 脚本文件,输入 `python -m xxx` 可执行特定的软件包。

```
C:\Users\70919>cd C:\Users\70919\Desktop\HTTP实验\python-3.8.10-embed-win32

C:\Users\70919\Desktop\HTTP实验\python-3.8.10-embed-win32>python ..\test.py
Hello World!

C:\Users\70919\Desktop\HTTP实验\python-3.8.10-embed-win32>python -m pip

Usage:
  C:\Users\70919\Desktop\HTTP实验\python-3.8.10-embed-win32\python.exe -m pip <command> [options]

Commands:
  install          Install packages.
  download         Download packages.
  uninstall        Uninstall packages.
  freeze           Output installed packages in requirements format.
  inspect          Inspect the python environment.
  list             List installed packages.
  show             Show information about installed packages.
  check            Verify installed packages have compatible dependencies.
  config           Manage local and global configuration.
```

图 2 使用命令行,执行特定的.py 脚本文件或软件包

2. Flask 与模板渲染简介

Flask 是一种基于 Python 编程语言的 Web 服务端框架,可以让用户实现方便、灵活的服务端编写。根据 2021 年 Python 开发者调查,Flask 是使用最广泛的 Python Web 编程框架,占比达 41%。Flask 可以被用于各种形式的 Web 服务,对网站所用的技术类型限制较小,静态网站、纯 API 服务、模板渲染等都可以在 Flask 中进行使用。

在继续进行前,你需要将我们准备的示例项目(`simple_server.zip`)解压到一个文件夹中,建议文件夹名为 `simple_server`。

使用 VSCode 打开项目

我们为大家准备了 VSCode，以方便大家编写代码。VSCode 位于实验软件目录下的 VSCode-win32-ia32-1.70.2 目录中，双击 Code.exe 即可打开。打开后，点击 File-Open Folder，打开项目文件夹（上面所说的 simple_server 文件夹即可）。

项目结构

一种经典的 Flask 项目的结构如图 3 所示，项目中包括静态文件、渲染模板、Python 脚本等，其中最核心的是 Python 脚本和渲染模板。

Python 脚本的入口点是 app.py。当在开发模式下启动一个 Flask 应用时，本质上就是在执行这个 app.py 文件。在这个文件中，定义了一个 Flask 对象（名为 app），代表着整个网站。再通过注解声明请求处理函数，例如@app.route('/')表示到该网站的/路径下的请求由该函数处理。

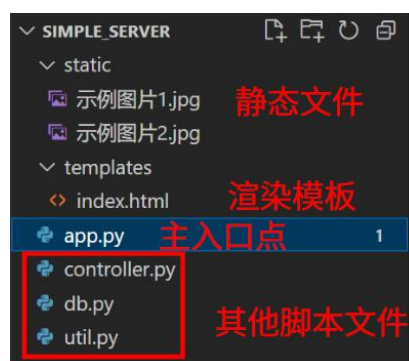


图 3 Flask 项目结构示例

运行服务

在开发环境下，我们可以通过 flask run 命令，启动 flask 内置的开发服务器，方便地进行调试。

(1) 打开终端，切换到项目目录（本实验中为 simple_server 目录）。

(2) 使用 flask run 运行服务器，注意由于我们的实验环境中 python 并未全局安装到系统中，因此需要通过 python -m 运行，并需要完整指定出 python 的路径。实际执行的命令应为..\python-3.8.10-embed-win32\python -m flask run。

(3) 服务开启后，会在终端中打印出 Running on http://127.0.0.1:5000 的字样。在浏览器中访问 <http://127.0.0.1:5000> 即可。

(4) 按 ctrl+C 可以关闭服务器。

然而，同学们可以注意到，使用 flask run 运行时会打印出如下的警告：WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.这是因为，flask run 所提供的仅仅是一个内置的调试用服务器，这种调试用服务器无论是在性能还是在安全性上都有较大的问题。因此，在生产环境中，不应该使用 flask run 所得到的调试服务器对外提供 Web 服务，而应该使用 Apache HTTP Server 等专业的 HTTP 服务器，通过 WSGI 调用我们的 Flask 应用程序，这便是下一节所要讲述的内容。

请求处理函数

在请求处理函数中，可以通过 request 对象拿到请求方法、请求路径、请求头、请求体等 HTTP 协议中规定的各种内容，和来源 IP 等信息，同时提供了多种函数供开发者方便的

使用这些信息。具体获得这些内容的方法，感兴趣的同学可以查看 Flask 的官方文档 (<https://flask.palletsprojects.com/en/2.2.x/api/#incoming-request-data>)。

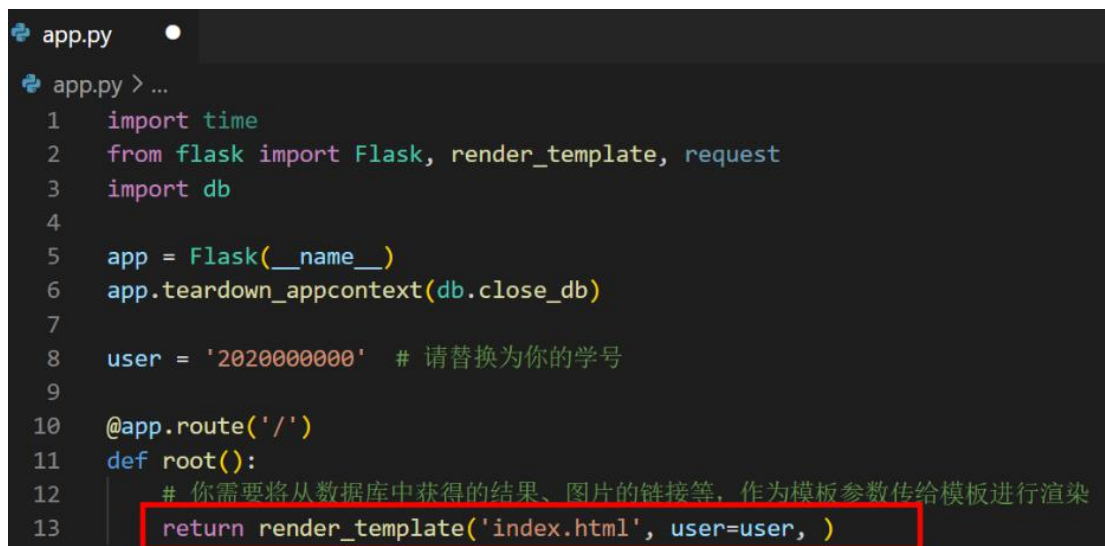
请求处理函数需要返回适当的内容，作为给用户的响应。这种内容可以是一个字符串（此时状态码为 200），可以是（字符串，状态码），还可以是一个返回对象。返回对象可以被看做一个对返回内容（状态码、响应头、响应体）的封装，含有生成 HTTP 响应所需的全部信息。但事实上，我们很少会手动实例化一个返回对象，而往往是通过调用 `render_template`（模板渲染）、`url_for`（重定向）等函数得到一个返回对象。感兴趣的同学可以查看文档：(<https://flask.palletsprojects.com/en/2.2.x/api/#response-objects>)。



```
<? index.html X
templates > <? index.html > html
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="UTF-8">
5   <title>实验二 - HTTP服务</title>
6 </head>
7 <body>
8
9 <h1>欢迎 {{ user }}!</h1>
10
11 </body>
12 </html>
```

← ↻ ⓘ 127.0.0.1:5000

欢迎 2020000000!



```
app.py
app.py > ...
1 import time
2 from flask import Flask, render_template, request
3 import db
4
5 app = Flask(__name__)
6 app.teardown_appcontext(db.close_db)
7
8 user = '2020000000' # 请替换为你的学号
9
10 @app.route('/')
11 def root():
12   # 你需要将从数据库中获得的结果、图片的链接等，作为模板参数传给模板进行渲染
13   return render_template('index.html', user=user, )
```

图 4 模板渲染的例子：左上角为模板，其中声明了名为 `user` 的模板替换标记；下图为 Python 代码，在运行时，`render_template` 函数会将模板中的 `{{user}}` 替换为所传入的 `user` 变量的值。右上角为运行结果。

模板渲染

模板渲染是一种动态生成网页内容的方法。模板文件是模板渲染的核心，它整体上看与一般的 HTML 十分相似，不同之处仅仅在于文件中包含了很多形如 `{{ xxx }}` 的标记，称为模板替换标记。

为了使用模板渲染，我们需要调用 `render_template` 函数，调用的方法例如 `render_template('index.html', user=user)`，第一个参数表示模板的文件名（相对于 `templates` 目录），而剩下的关键字参数表示模板参数的值。此时，Flask 内置的模板渲

渲染引擎会读取模板文件，然后将模板文件中的所有形如`{{ xxx }}`的模板标记用名为 `xxx` 的模板参数替换。例如，在图 4 给出的例子中，`index.html` 中的`{{ user }}`会被我们传给 `render_template` 函数的 `user` 变量（值为 2020000000）替换，所以我们会得到图 4 最右所示的网页。

这里有一点请同学们注意：`user=user` 中，第一个 `user` 表示模板变量的名字，模板变量是存在于模板中的，与 `python` 中的变量无任何关联；而第二个参数是我们为模板变量赋的值，可以是数字字面量（例如 100）、字符串字面量（例如 `'hello'`），当然更可以是已定义的 `python` 变量。

3. Apache HTTP Server 与 WSGI

Apache HTTP Server 的简介 —— 详见旧版指导书相应段落

Apache HTTP Server 的启动、关闭与配置

在我们本次实验中，为大家准备的是免安装版的 Apache Longue，无需安装，解压即可使用，文件名为 `Apache24.zip`。这里，建议大家解压到 `C:\Apache24` 下，否则需要自行更改 `httpd.conf` 中的 `SRVROOT` 到正确的路径。

解压完毕后，打开安装目录下的 `bin/httpd.exe` 就可以打开服务器了。这时会弹出一个终端窗口，但窗口中不会有任何输出，这是正常的。

此时打开浏览器，访问 <http://localhost/>，如果看到 `It works!` 的字样，即说明 Apache 服务器启动成功。

点击终端窗口右上角的叉即可关闭服务器，注意这里的关闭过程可能会卡几秒是正常的。

Apache HTTP Server 有很多配置文件，均位于 `conf` 文件夹中，其中，最重要的是 `httpd.conf`。下文所说的所有配置修改均是在 `httpd.conf` 中进行的。

DocumentRoot 与静态文件分发

在 `httpd.conf` 中有一个名为 `DocumentRoot` 的配置，默认值为 `${SRVROOT}/htdocs`，对应了 Apache 安装目录下的 `htdocs` 子目录。点击进这个目录不难发现，刚才我们看到的 `It works!` 实际上就来源于这里的 `index.html`。因此，分发静态文件的最简单的方式是要分发的静态文件放到这里即可。例如，在 `htdocs` 目录中新建一个记事本文件 `1.txt`，输入任意内容，再访问 <http://localhost/1.txt>，即可获得文件中的内容。放入图片、视频等也同理。

基于 Alias 的静态文件分发

上一节直接将文件放进 `DocumentRoot` 的做法，对于较大的项目或多个项目共用服务器的情况有很大的缺陷：把所有文件全部集中放在一起，非常不便于管理。因此，更好的办法是使用 `Alias`，为不同的请求路径指定不同的存放文件的位置。

例如，我们希望在 HTTP 的 `/static/` 路径下分发 `simple_server` 中的 `static` 文件夹中提供的示例图片，这需要用到名为 `alias_module` 的模块提供的 `Alias` 指令。

我们需要找到 `<IfModule alias_module>` 标签（即 `<IfModule alias_module>` 和 `</IfModule>` 之间的部分。下文再提到 `XXX` 标签，均指 `<XXX>` 和 `</XXX>` 之间的部分），在其中添加一行：

```
Alias /static/ "xxx/simple_server/static/"
```

其中 xxx/simple_server 需要是 simple_server 实际所在的目录。**注意：/不能打成\, static/最后的一个/不能省略。**

此外，我们还需要给 Apache 服务器提供访问上述目录的权限，否则访问时会报 403 Forbidden 错误。给目录添加权限的方法可参照<Directory "\${SRVROOT}/cgi-bin"> 标签。为了让我们的静态文件能够访问，我们应该至少给 xxx\simple_server\static 目录添加上权限。

```
<Directory "xxx\simple_server">
    AllowOverride None
    Options None
    Require all granted
</Directory>
```

实际上，因为下一小节我们将把 xxx\simple_server\app.py 配置为 WSGI 的脚本入口点，因此这里此时可以直接对整个 xxx\simple_server 给予授权。直接添加上述配置即可；完成以上配置之后，访问<http://localhost/static/示例图片1.jpg>应该就可以看到图片了。

WSGI 的配置

WSGI (Web Server Gateway Interface) 是为 Python 语言定义的 Web 服务器和 Web 应用程序或框架之间的一种简单而通用的接口。利用 WSGI，我们可以实现从 Apache 中调用我们用 Python 语言编写的 Web 服务器。

首先，我们需要在 python 环境中安装 WSGI 的依赖。我们已为大家下载好该依赖的 wheel 文件：mod_wsgi-4.9.2-cp38-cp38-win32.whl。打开终端，切换到 python 所在的目录，然后执行：

```
python -m pip install ..\mod_wsgi-4.9.2-cp38-cp38-win32.whl
```

接下来，我们还需要在 Apache 的 httpd.conf 中完成配置。配置分为两部分：第一部分是声明 mod_wsgi 和 python 安装的位置，让 Apache 能够调用 mod_wsgi；第二部分则是声明我们的项目的位置，让 WSGI 知道应该运行哪个 python 文件、使用其中的哪个对象作为接口对象。

对于第一部分，mod_wsgi 已经为我们提供了一个工具来自动化的产生所需的配置，该工具位于 python 安装路径下的 Script\mod_wsgi-express.exe。打开命令行，切换到 python 的安装目录后，执行 .\Scripts\mod_wsgi-express.exe module-config，会产生 3 行输出，将这三行输出粘贴到 httpd.conf 中即可。

对于第二部分，我们需要声明 WSGIScriptAlias、WSGICallableObject、WSGI PythonPath 三个变量。WSGIScriptAlias 和 Alias 有点类似，仍然代表一种 URL 路径和物理路径之间的映射关系，不同的是 WSGIScriptAlias 中需要指定的是一个 Python 脚本文件，而非一个目录。WSGICallableObject 指的是 WSGI 应该调用 WSGIScriptAlias 中指定的 Python 文件里面的哪一个对象。在 app.py 中，我们的 Flask 对象名为 app。最后，WSGI PythonPath 是用于解析 import 的路径，应该设置为我们的项目路径。综上所述，第二部分的配置应为：（注意要把 xxx\simple_server 替换为 simple_server 实际所在的路径）

```
WSGIScriptAlias / "xxx\simple_server\app.py"  
WSGICallableObject app  
WSGIPythonPath "xxx\simple_server"
```