**1. 简介**

这份文档中的代码使用 Python 3 运行。  
是的，所以读者需要自己在电脑上安装Python 3 和 pip3。建议安装最新版本，我使用的是Python 3.6.4。  
安装方法，可以自行谷歌或者百度。  
建议在 linux 下实践本教程中命令行操作、执行代码。

**2. 安装**

通过pip3安装Flask即可：

$ sudo pip3 install Flask

进入python交互模式看下Flask的介绍和版本：

$ python3

>>> import flask

>>> print(flask.\_\_doc\_\_)

flask

~~~~~

A microframework based on Werkzeug. It's extensively documented

and follows best practice patterns.

:copyright: © 2010 by the Pallets team.

:license: BSD, see LICENSE for more details.

>>> print(flask.\_\_version\_\_)

1.0.2

**3. 从 Hello World 开始**

本节主要内容：使用Flask写一个显示”Hello World!”的web程序，如何配置、调试Flask。

**3.1 Hello World**

按照以下命令建立Flask项目HelloWorld:

mkdir HelloWorld

mkdir HelloWorld/static

mkdir HelloWorld/templates

touch HelloWorld/server.py

static和templates目录是默认配置，其中static用来存放静态资源，  
例如图片、js、css文件等。templates存放模板文件。

我们的网站逻辑基本在server.py文件中，  
当然，也可以给这个文件起其他的名字。

在server.py中加入以下内容：

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello World!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

运行server.py：

$ python3 server.py

\* Running on http://127.0.0.1:5000/

打开浏览器访问http://127.0.0.1:5000/，  
浏览页面上将出现Hello World!。  
终端里会显示下面的信息：

127.0.0.1 - - [16/May/2014 10:29:08] "GET / HTTP/1.1" 200 -

变量app是一个Flask实例，通过下面的方式：

@app.route('/')

def hello\_world():

return 'Hello World!'

当客户端访问/时，将响应hello\_world()函数返回的内容。注意，这不是返回Hello World!这么简单，Hello World!只是HTTP响应报文的实体部分，状态码等信息既可以由Flask自动处理，也可以通过编程来制定。

**3.2 修改Flask的配置**

app = Flask(\_\_name\_\_)

上面的代码中，python内置变量\_\_name\_\_的值是字符串\_\_main\_\_ 。  
Flask类将这个参数作为程序名称。

当然这个是可以自定义的，比如app = Flask(“my-app”)。

Flask默认使用static目录存放静态资源，templates目录存放模板，这是可以通过设置参数更改的：

app = Flask("my-app", static\_folder="path1", template\_folder="path2")

更多参数请参考\_\_doc\_\_：

from flask import Flask

print(Flask.\_\_doc\_\_)

**3.3 调试模式**

上面的server.py中以app.run()方式运行，  
这种方式下，如果服务器端出现错误是不会在客户端显示的。

但是在开发环境中，显示错误信息是很有必要的，  
要显示错误信息，应该以下面的方式运行Flask：

app.run(debug=True)

将debug设置为True的另一个好处是，程序启动后，会自动检测源码是否发生变化，若有变化则自动重启程序。这可以帮我们省下很多时间。

**3.4 绑定IP和端口**

默认情况下，Flask绑定IP为127.0.0.1，端口为5000。  
我们也可以通过下面的方式自定义：

app.run(host='0.0.0.0', port=80, debug=True)、

0.0.0.0代表电脑所有的IP。80是HTTP网站服务的默认端口。

什么是默认？比如，我们访问网站http://www.example.com，  
其实是访问的http://www.example.com:80，只不过:80可以省略不写。

由于绑定了80端口，需要使用root权限运行server.py。也就是：

$ sudo python3 server.py

**3.5 本节源码**

https://github.com/letiantian/Learn-Flask/tree/master/flask-demo-001

**4. 获取 URL 参数**

URL参数是出现在url中的键值对，例如http://127.0.0.1:5000/?disp=3中的url参数是{‘disp’:3}。

**4.1 建立Flask项目**

按照以下命令建立Flask项目HelloWorld:

mkdir HelloWorld

mkdir HelloWorld/static

mkdir HelloWorld/templates

touch HelloWorld/server.py

**4.2 列出所有的url参数**

在server.py中添加以下内容：

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return request.args.\_\_str\_\_()

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

在浏览器中访问http://127.0.0.1:5000/?user=Flask&time&p=7&p=8，  
将显示：

ImmutableMultiDict([('user', 'Flask'), ('time', ''), ('p', '7'), ('p', '8')])

较新的浏览器也支持直接在url中输入中文  
（最新的火狐浏览器内部会帮忙将中文转换成符合URL规范的数据），  
在浏览器中访问http://127.0.0.1:5000/?info=这是爱，  
将显示：

ImmutableMultiDict([('info', '这是爱，')])

浏览器传给我们的Flask服务的数据长什么样子呢？  
可以通过request.full\_path和request.path来看一下：

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

print(request.path)

print(request.full\_path)

return request.args.\_\_str\_\_()

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

浏览器访问http://127.0.0.1:5000/?info=这是爱，  
运行server.py的终端会输出：

/

/?info=%E8%BF%99%E6%98%AF%E7%88%B1%EF%BC%8C

**4.3 获取某个指定的参数**

例如，要获取键info对应的值，如下修改server.py：

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return request.args.get('info')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000)

运行server.py，  
在浏览器中访问http://127.0.0.1:5000/?info=hello，  
浏览器将显示：

hello

不过，当我们访问http://127.0.0.1:5000/时候却出现了500错误，  
浏览器显示：

如果开启了Debug模式，会显示：

**为什么为这样？**

这是因为没有在URL参数中找到info。所以request.args.get(‘info’)返回Python内置的None，而Flask不允许返回None。

解决方法很简单，我们先判断下它是不是None：

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

r = request.args.get('info')

if r==None:

# do something

return ''

return r

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

另外一个方法是，设置默认值，也就是取不到数据时用这个值：

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

r = request.args.get('info', 'hi')

return r

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

函数request.args.get的第二个参数用来设置默认值。  
此时在浏览器访问http://127.0.0.1:5000/，将显示：

hi

**4.4 如何处理多值**

还记得上面有一次请求是这样的吗？  
http://127.0.0.1:5000/?user=Flask&time&p=7&p=8，  
仔细看下，p有两个值。

如果我们的代码是：

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

r = request.args.get('p')

return r

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

在浏览器中请求时，我们只会看到7。  
如果我们需要把p的所有值都获取到，该怎么办？

不用get，用getlist：

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

r = request.args.getlist('p') # 返回一个list

return str(r)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

浏览器输入 http://127.0.0.1:5000/?user=Flask&time&p=7&p=8，我们会看到[‘7’, ‘8’]。

**4.5 本节源码**

https://github.com/letiantian/Learn-Flask/tree/master/flask-demo-002

**5. 获取POST方法传送的数据**

作为一种HTTP请求方法，POST用于向指定的资源提交要被处理的数据。我们在某网站注册用户、写文章等时候，需要将数据传递到网站服务器中。并不适合将数据放到URL参数中，密码放到URL参数中容易被看到，文章数据又太多，浏览器不一定支持太长长度的URL。这时，一般使用POST方法。

本文使用python的requests库模拟浏览器。

**安装方法：**

$ sudo pip3 install requests

**5.1 建立Flask项目**

按照以下命令建立Flask项目HelloWorld:

mkdir HelloWorld

mkdir HelloWorld/static

mkdir HelloWorld/templates

touch HelloWorld/server.py

**5.2 查看POST数据内容**

以用户注册为例子，  
我们需要向服务器/register传送用户名name和密码password。  
如下编写HelloWorld/server.py。

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'hello world'

@app.route('/register', methods=['POST'])

def register():

print(request.headers)

print(request.stream.read())

return 'welcome'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000, debug=True)

@app.route(‘/register’, methods=[‘POST’])  
是指url/register只接受POST方法。  
可以根据需要修改methods`参数，  
例如如果想要让它同时支持GET和POST，  
这样写：