# 1.4 客户端POST发送数据

深圳信息职业技术学院
Shenzhen Institute Of Information Technology
教师：黄锐军

# 目 录

COMPANY

# PART ONE

## 客户端POST发送数据

# 客户端POST发送数据

POST方法访问网站时客户端向服务器发送表单数据，表单数据的组织方式与GET方法的参数列表十分相似，结构如下：

"名称1=值1&名称2=值2&名称3=值3……"

多个数据之间用"&"符号隔开，如果参数值包含汉字，那么我们必须使用urllib.parse.quote对参数值进行编码，例如：

```
province= urllib.parse.qoute("广东")
city= urllib.parse.qoute("深圳")
data="province="+province+"&city="+city
data=data.encode()
urllib.request.urlopen("http://127.0.0.1:5000",data=data)
```

# 客户端POST发送数据

这里data=data.encode()是把data字符串按utf-8的编码转为二进制数据。POST方法与GET方法最大的不同是GET的参数放在地址栏的后面，而POST的数据放在urlopen函数的data参数中，而且这个参数值必须是二进制数据。

编程客户端client.py程序如右：

```
import urllib.parse
import urllib.request
url="http://127.0.0.1:5000"
try:
    province= urllib.parse.quote("广东")
    city= urllib.parse.quote("深圳")
    data="province="+province+"&city="+city
    data=data.encode()

html=urllib.request.urlopen("http://127.0.0.1:5000",data=data)
    html = html.read()
    html = html.decode()
    print(html)
except Exception as err:
    print(err)
```

# PART TWO

## 服务器获取POST的数据

# 服务器获取POST的数据

服务器用Flask中的request对象的form来存储GET的参数，用get方法来获取参数，即用flask.request.form.get(参数)来获取参数的值，例如：

**province=flask.request.form.get("province")**

**city=flask.request.form.get("city")**

就可以获取GET传递的参数province与city的值。

编写服务器程序server.py如下：

```
import flask
app=flask.Flask(__name__)

@app.route("/",methods=["POST"])
def index():
    try:
        province=flask.request.form.get("province") if "province" in
flask.request.form else ""
        city = flask.request.form.get("city") if "city" in
flask.request.form else ""
        return province+","+city
    except Exception as err:
        return str(err)

if __name__=="__main__":
    app.run()
```

# 服务器获取POST的数据

值得注意的是在服务器中要指定：

**@app.route("/",methods=["POST"])**

**def index():**

表明函数这个函数接受POST请求。默认时只接受GET请求，如果要接受POST请求就必须明确指明，如果写成：

**@app.route("/",methods=["GET","POST"])**

**def index():**

那么这个函数即可以接受GET请求也可以接受POST请求。

我们先运行server.py建立web网站，再运行client.py，可以看到client.py的结果是：

广东,深圳

# PART Three

## GRT与POST的混合使用

# GET与POST的混合使用

实际上在应用中客户端客户同时使用GET与POST向服务器发送数据，一般GET的数据时放在地址栏后面的，参数简单，数据量少，而POST的数据时表单数据，数据量大。

例：客户端向服务器发送一个城市的简介，服务器接收后返回收到的信息。

我们把省份province与城市city的名称放在url地址栏后面采用GET方式发送给服务器，然后把城市的简介用POST方法发送给服务器，客户端程序如下：

```
import urllib.parse
import urllib.request
url="http://127.0.0.1:5000"
note="深圳依山傍海，气候宜人，实在是适合人类居住的绝佳地。这里四季如春，干净整洁，比邻香港，拥有着丰富的自然景观和人文气息。匆匆过客注意到的也许只有它的时尚繁华，忙碌的暂居者也可能对它有着不识城市真面目之感。只有世代在此生活的老深圳人，才默默的看着它从贫穷走向富饶经历了怎样的艰辛。"
try:
    province= urllib.parse.quote("广东")
    city= urllib.parse.quote("深圳")
    note= "note="+urllib.parse.quote(note)
    param="province="+province+"&city="+city
    html=urllib.request.urlopen("http://127.0.0.1:5000?"+param,data=note.encode())
    html = html.read()
    html = html.decode()
    print(html)
except Exception as err:
    print(err)
```

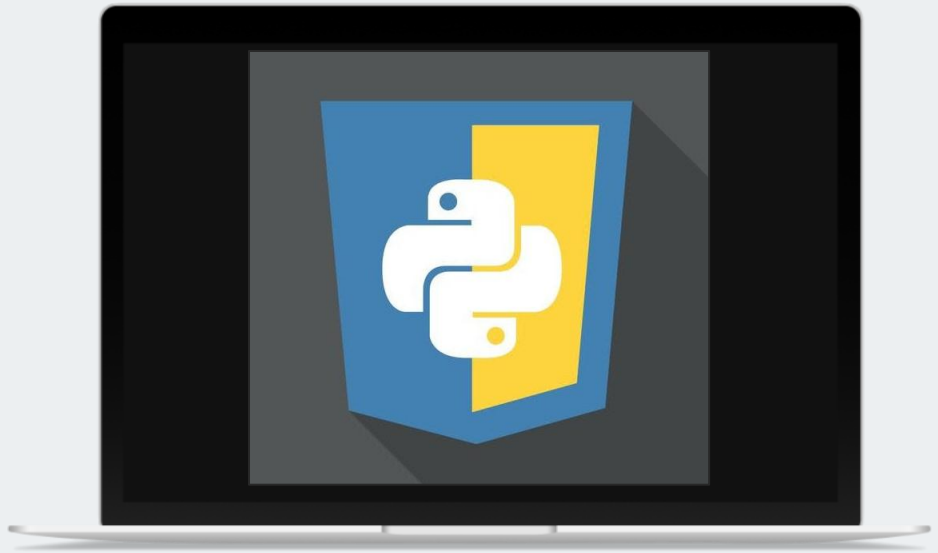
服务器程序如下：

```
import flask
app=flask.Flask(__name__)

@app.route("/",methods=["GET","POST"])
def index():
    try:
        province=flask.request.args.get("province") if "province" in flask.request.args
else ""
        city = flask.request.args.get("city") if "city" in flask.request.args else ""
        note = flask.request.form.get("note") if "note" in flask.request.form else ""
        return province+","+city+"\n"+note
    except Exception as err:
        return str(err)

if __name__=="__main__":
    app.run()
```

我们看到服务器接收GET的参数采用flask.request.args.get(参数)与接收POST参数的方法flask.request.form.get(参数)是不同的，实际上可以把它们进行统一成flask.request.values.get(参数)，采用这个方法可以获取GET的参数也可以获取POST的参数，即服务器程序可以进一步改进如下：



```
import flask
app=flask.Flask(__name__)


@app.route("/",methods=["GET","POST"])
def index():
    try:
        province=flask.request.values.get("province") if "province" in
flask.request.values else ""
        city = flask.request.values.get("city") if "city" in flask.request.values else ""
        note = flask.request.values.get("note") if "note" in flask.request.values else
""
        return province+","+city+"\n"+note
    except Exception as err:
        return str(err)

if __name__=="__main__":
    app.run()
```

# THANK YOU