**WEB框架开发**

# 一 Django框架开发

## 1.WEB 应用

Web应用程序是一种可以通过种可以通过Web访问的应用程序，程序的最大好处是用户很容易访问应用程序，用户只需要有浏览器即可，不需要再安装其他软件。应用程序有两种模式C/S、B/S。C/S是客户端/服务器端程序，也就是说这类程序一般独立运行。而B/S就是浏览器端/服务器端应用程序，这类应用程序一般借助谷歌，火狐等浏览器来运行。WEB应用程序一般是B/S模式。Web应用程序首先是“应用程序”，和用标准的程序语言，如java，python等编写出来的程序没有什么本质上的不同。在网络编程的意义下，浏览器是一个socket客户端，服务器是一个socket服务端。

## 2.http协议

### 1.协议简介

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于万维网服务器与本地浏览器之间传输超文本的传送协议。

HTTP是一个属于应用层的面向对象协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。

### 2.http协议特性

1）基于TCP/IP ：HTTP协议是基于TCP/IP协议上的应用层协议。

2）基于请求-响应模式 ：HTTP协议规定，请求从客户端发出，最后服务器端响应该请求并返回。换句话说，肯定是先从客户端开始建立通信的，服务端在没有接收到请求之前不会响应。

3）无状态保存 ： HTTP是一种不保存状态,即无状态(stateless)协议。HTTP协议 自身不对请求和响应之间的通信状态进行保存。也就是说在HTTP这个 级别,协议对于发送过的请求或响应都不做持久化处理。

使用HTTP协议,每当有新的请求发送时,就会有对应的新响应产生。协议本身并不保留之前一切的请求或响应报文的信息。这是为了更快地处理大量事务,确保协议的可伸缩性,而特意把HTTP协议设计成 如此简单的。可是,随着Web的不断发展,因无状态而导致业务处理变得棘手 的情况增多了。比如,用户登录到一家购物网站,即使他跳转到该站的 其他页面后,也需要能继续保持登录状态。针对这个实例,网站为了能 够掌握是谁送出的请求,需要保存用户的状态。HTTP/1.1虽然是无状态协议,但为了实现期望的保持状态功能, 于是引入了Cookie技术。有了Cookie再用HTTP协议通信,就可以管 理状态了。有关Cookie的详细内容稍后讲解。

4）无连接 ：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

### 3.http请求协议与响应协议

http协议包含由浏览器发送数据到服务器需要遵循的请求协议与服务器发送数据到浏览器需要遵循的请求协议。用于HTTP协议交互的信被为HTTP报文。请求端(客户端)的HTTP报文做请求报文,响应端(服务器端)的做响应报文。HTTP报文本身是由多行数据构成的字文本。



1）请求协议

请求格式：



请求方式：get与post请求

* GET提交的数据会放在URL之后，以?分割URL和传输数据，参数之间以&相连，如EditBook?name=test1&id=123456. POST方法是把提交的数据放在HTTP包的请求体中.
* GET提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST方法提交的数据没有限制.
* GET与POST请求在服务端获取请求数据方式不同。

2）响应协议

响应格式：



响应状态码：状态码的值是当客户端向服务端发送请求时，返回的请求结果。借助状态码，用户可以知道服务器端是否正常处理了请求。



实例：

# -\*- coding: utf-8 -\*-  
*import* socket  
sock = socket.socket()  
sock.bind(("127.0.0.1",8801))  
sock.listen(5)  
*while True*:  
 # conn 客户端套接字  
 conn,addr = sock.accept()  
 data = conn.recv(1024)  
 # HTTP/1.1 200 OK 相应首行  
 # 读取html文件  
 # POST 请求才有请求头数据  
 # 按照http请求协议解析数据  
 # 专注于web业务开发  
 # 按照http响应协议封装数据  
 *with* open('login.html','r') *as* f:  
 data = f.read()  
 conn.send(("HTTP/1.1 200 OK\r\n\r\n%s"%data).encode('utf8'))  
 conn.close()

login.html

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**login**</**title**>  
</**head**>  
<**body**>  
<**form** action=**"http://127.0.0.1:8801"** method=**"post"**>  
 **用户名：**<**input** type=**"text"** name=**"user"**>  
 **密码：**<**input** type=**"password"** name=**"pwd"**>  
 <**input** type=**"submit"** value=**"提交"**>  
</**form**>  
</**body**>  
</**html**>

## 3.wsgiref模块

WSGI（Web Server Common Interface）是专门为Python语言制定的web服务器与应用程序之间的网关接口规范，通俗的来说，只要一个服务器拥有一个实现了WSGI标准规范的模块（例如apache的mod\_wsgi模块），那么任意的实现了WSGI规范的应用程序都能与它进行交互。因此，WSGI也主要分为两个程序部分：服务器部分和应用程序部分。

其实 wsgiref 最基本的功能就是封装了socket，不需要自己再去写socket。因为我们不希望接触到TCP连接、HTTP原始请求和响应格式，所以，需要一个统一的接口协议来实现这样的服务器软件，让我们专心用Python编写Web业务。这个接口就是WSGI：Web Server Gateway Interface。而wsgiref模块就是python基于wsgi协议开发的服务模块。

*from* wsgiref.simple\_server *import* make\_server  
  
*def* application(*environ*, *start\_response*):  
 *start\_response*('200 OK', [('Content-Type', 'text/html')])  
 *return* [b'<h1>Hello, web!</h1>']

httpd = make\_server('', 8080, application)  
print('Serving HTTP on port 8000...')  
# 开始监听HTTP请求:  
httpd.serve\_forever()

## 4.DIY一个WEB框架

利用wsgiref模块实现一个简单的WEB框架

目录结构如下：

