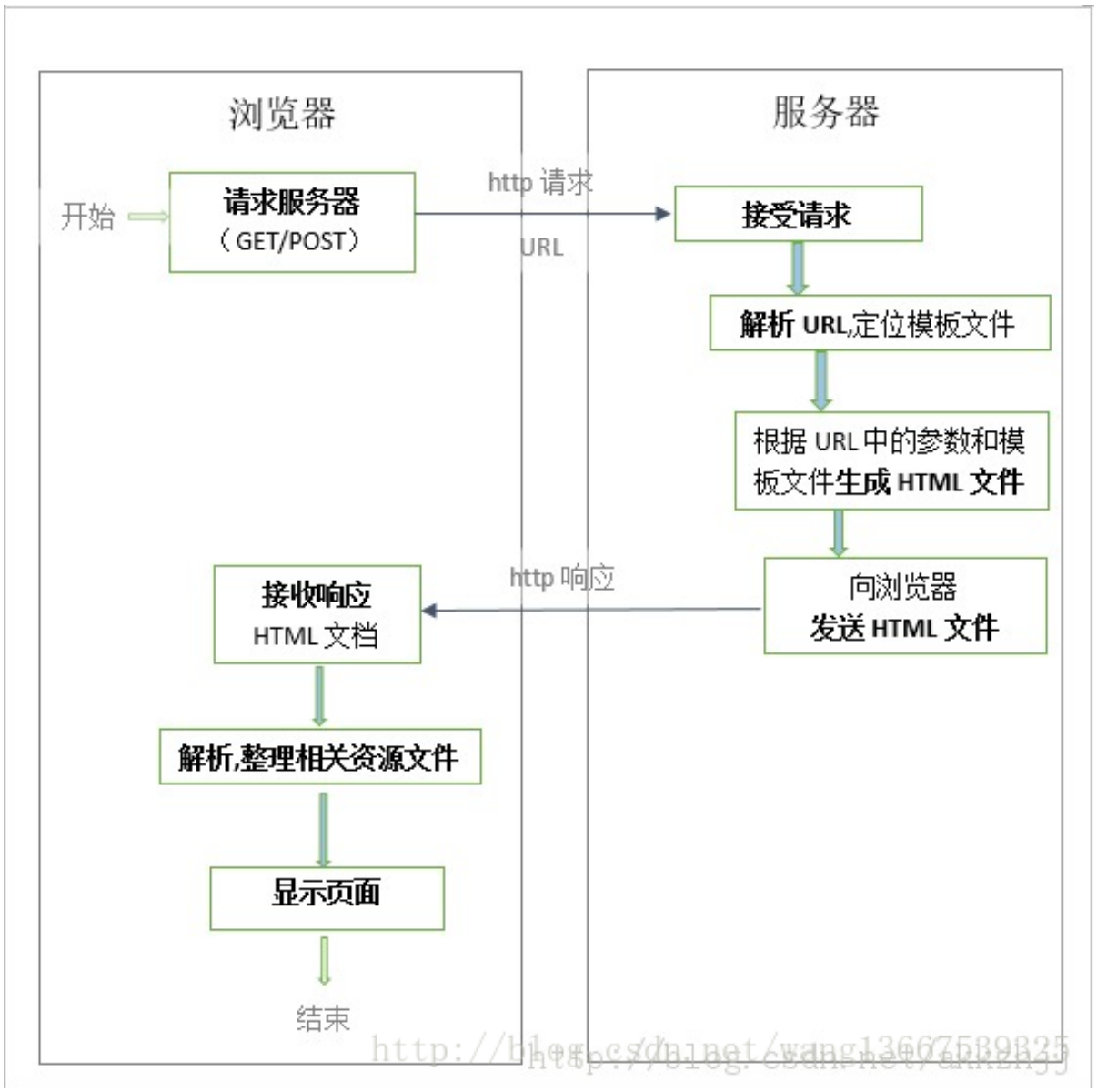
web框架

B/S架构

B/S（Brower/Server,浏览器/服务器）模式又称B/S结构，是Web兴起后的一种网络结构模式。Web浏览器是客户端最主要的应用软件。

这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用；

客户机上只需要安装一个浏览器，服务器上安装SQL Server, Oracle, MySql等数据库；浏览器通过Web Server同数据库进行数据交互。



- 优点：
    - B/S最大的优点就是可以在任何地方进行操作而不用安装任何专门的软件，只要有一台能上网的电脑就能使用，客户端零安装、零维护。系统的扩展非常容易。
    - 由需求推动了AJAX技术的发展，它的程序也能在客户端电脑上进行部分处理，从而大大的减轻了服务器的负担；并增加了交互性，能进行局部实时刷新。
    - B/S结构主要利用了不断成熟的Web浏览器技术：结合浏览器的多种脚本语言和ActiveX技术，用通用浏览器实现原来需要复杂专用软件才能实现的强大功能，节约了开发成本。

一、http协议

HTTP是基于客户端/服务端（C/S）的架构模型，通过一个可靠的链接来交换信息，是一个无状态的请求/响应协议。

一个HTTP"客户端"是一个应用程序（Web浏览器或其他任何客户端），通过连接到服务器达到向服务器发送一个或多个HTTP的请求的目的。

一个HTTP"服务器"同样也是一个应用程序（通常是一个Web服务，如Apache Web服务器或IIS服务器等），通过接收客户端的请求并向客户端发送HTTP响应数据。

HTTP使用统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。

1.1主要特点

- 简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST、PUT、DELETE等。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。
- 灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。
- 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。
- 无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能

导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

- 支持B/S及C/S模式。

1.2 客户端请求消息

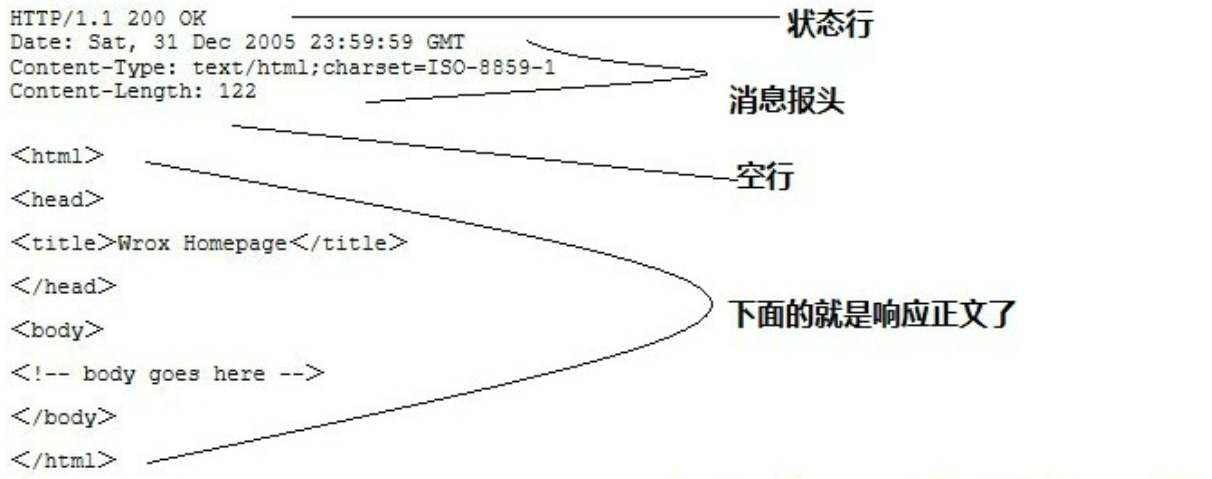
客户端发送一个HTTP请求到服务器的请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据四个部分组成，下图给出了请求报文的一般格式。

- HTTP请求方法

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	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
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

1.3 服务器端响应消息

HTTP响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。



1.4 HTTP状态码

当浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。

HTTP状态码由三个十进制数字组成，第一个十进制数字定义了状态码的类型，后两个数字没有分类的作用。HTTP状态码共分为5种类型：

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

- 常见状态码

状态码	状态码英文名称	中文描述

100	Continue	继续。客户端应继续其请求
101	Switching Protocols	切换协议。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到HTTP的新版本协议
200	OK	请求成功。一般用于GET与POST请求
201	Created	已创建。成功请求并创建了新的资源
202	Accepted	已接受。已经接受请求，但未处理完成
203	Non-Authoritative Information	非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本
204	No Content	无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档
300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择
301	Moved Permanently	永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替
302	Found	临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI
400	Bad Request	客户端请求的语法错误，服务器无法理解
401	Unauthorized	请求要求用户的身份认证
403	Forbidden	服务器理解客户端的请求，但是拒绝执行此请求
404	Not Found	服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源

		无法找到"的个性页面
500	Internal Server Error	服务器内部错误，无法完成请求
501	Not Implemented	服务器不支持请求的功能，无法完成请求
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求
503	Service Unavailable	由于超载或系统维护，服务器暂时的无法处理客户端的请求。

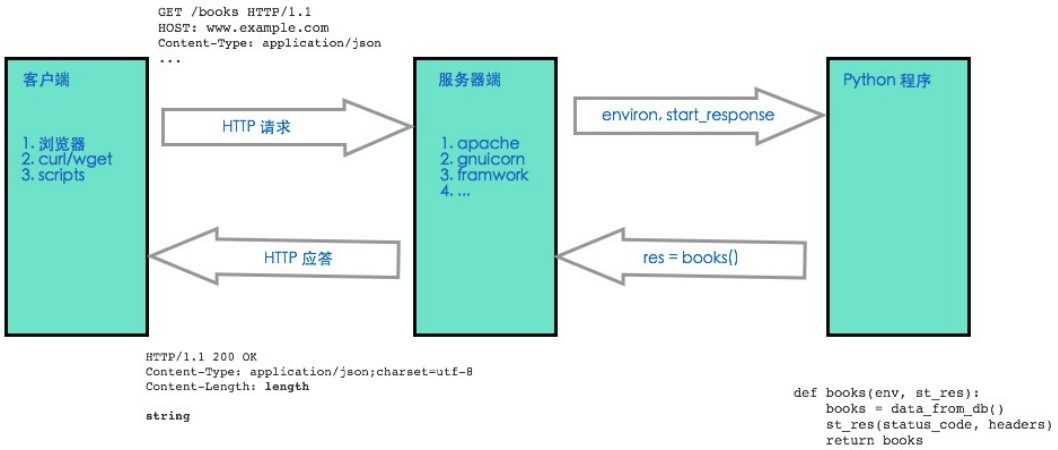
1.5 HTTP content-type

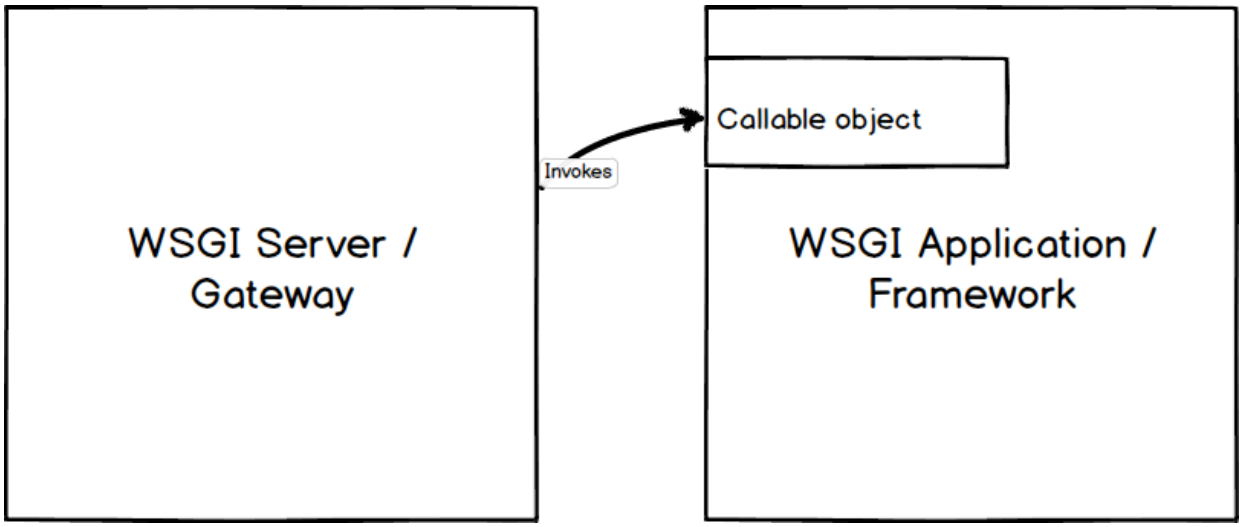
Content-Type，内容类型，一般是指网页中存在的Content-Type，用于定义网络文件的类型和网页的编码，决定浏览器将以什么形式、什么编码读取这个文件，这就是经常看到一些Asp网页点击的结果却是下载到的一个文件或一张图片的原因。

文件扩展名	**Content-Type(Mime-Type)**
.html	text/html
.js	application/x-javascript
.css	text/css
.asp	text/asp
.txt	text/plain

二、wsgi

Web服务器网关接口（WSGI）是Web服务器软件和用Python编写的Web应用程序之间的标准接口。拥有这样的标准接口是便于应用程序支持有WSGI的大量的不同的web服务器。





WSGI 规定每个 python 程序（Application）必须是一个可调用的对象（函数或者是实现了`__call__`方法的类），接受两个参数`environ`（WSGI 的环境信息） 和`start_response`（开始响应请求的函数），并且返回 iterable。几点说明：

1. `environ` 和 `start_response` 由 http server 提供并实现
2. `environ` 变量是包含了环境信息的***字典***
3. `Application` 内部在返回前调用 `start_response`
4. `start_response`也是一个 callable，接受两个必须的参数，`status`（HTTP状态）和 `response_headers`（响应消息的头）
5. 可调用对象要返回一个值，这个值是可迭代的。

environ中重要的键值对：

('HTTP_REFERER', '--------', 'http://192.168.8.190:8000/') 来源页面

'wsgi.input': <socket._fileobject object at 0x7fd4d2d36950>, post请求参数获取

('QUERY_STRING', '--------', 'name=tom&page=1') 查询参数
('REQUEST_METHOD', '--------', 'GET') 请求方法 ('REMOTE_ADDR', '--------', '127.0.0.1') 客户端地址 ('wsgi.url_scheme', '--------', 'http') 协议 ('SERVER_PORT', '--------', '8000') 端口 ('REMOTE_HOST', '--------', 'localhost') 服务器地址

'HTTP_HOST': '192.168.71.23:9000',

('PATH_INFO', '--------', '/index.html') 文件路径

- 应用程序

```
#!/usr/bin/env python
# coding:utf-8

def application(environ,start_response):
    start_response("200 ok",[('ContentType','text/html')])
    return [b"<h1>hello world</h1>"]
```

- 服务器

```
from wsgiref.simple_server import make_server
# 导入我们自己编写的application函数:
from app import application

# 创建一个服务器，IP地址为空，端口是8000，处理函数是application:
httpd = make_server('', 8000, application)
print('Serving HTTP on port 8000...')
# 开始监听HTTP请求:
httpd.serve_forever()
```