Web应用相关知识概览

1. HyperText Transfer Protocol

1. HTTP 1.0 (1996) / 1.1 (1997) / 2.0 (2015)
2. 构建在 TCP 应用层之上的协议
3. 应用领域
4. 认识 URL (统一资源定位符)
    - `http://example.com:80/foo/bar/readme?x=0&y=abc#part1`
    - 组成
        - scheme: http
        - hostname: example.com
        - port: 80
        - path: /foo/bar/readme
        - query: ?x=0&y=abc
        - fragment: #part1

2. Python 系常见 Web 框架

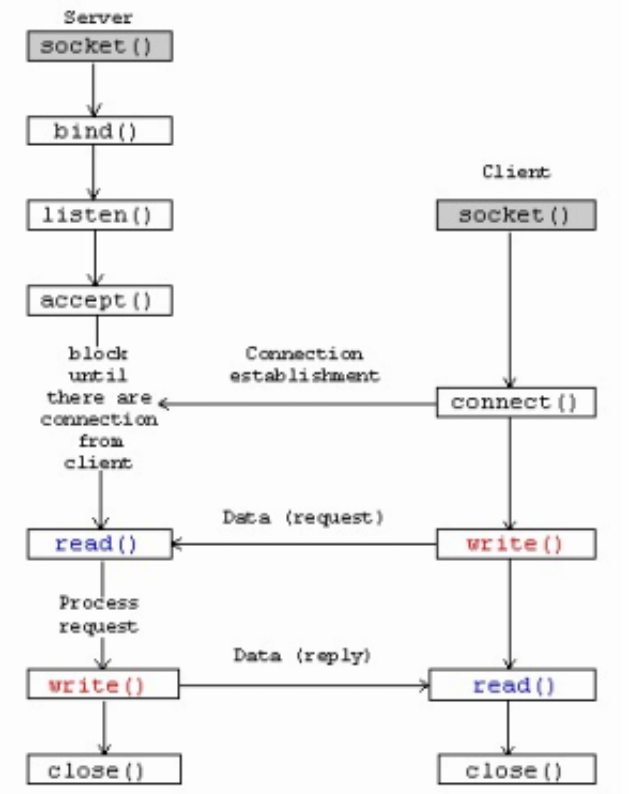
- Django
    - 全能型框架, 大而全, 插件丰富, 文档丰富, 社区活跃, 适合快速开发, 内部耦合比较紧
- Flask
    - 微型框架, 适合新手学习, 极其灵活, 便于二次开发和扩展, 生态环境好, 插件丰富
- Tornado
    - 异步处理, 事件驱动 (epoll), 性能优异
- web.py
    - 代码优秀, 适合学习源码
- bottle
    - 单文件框架
- 其他
    - Falcon
    - web2py
    - Quixote
    - Sanic

3. 点击一个链接后, 都发生了什么

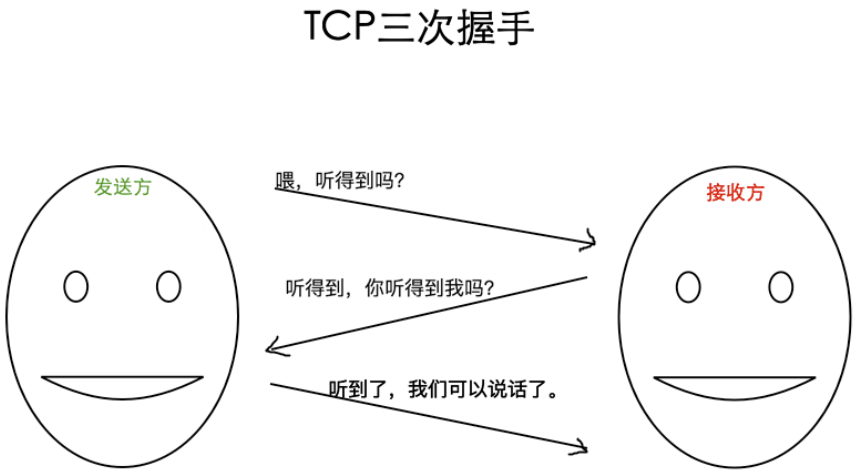
1. DNS解析

- example.com -> 93.184.216.34
- `dig example.com`
- `nslookup example.com`
- `/etc/hosts`

2. 建立和断开TCP连接

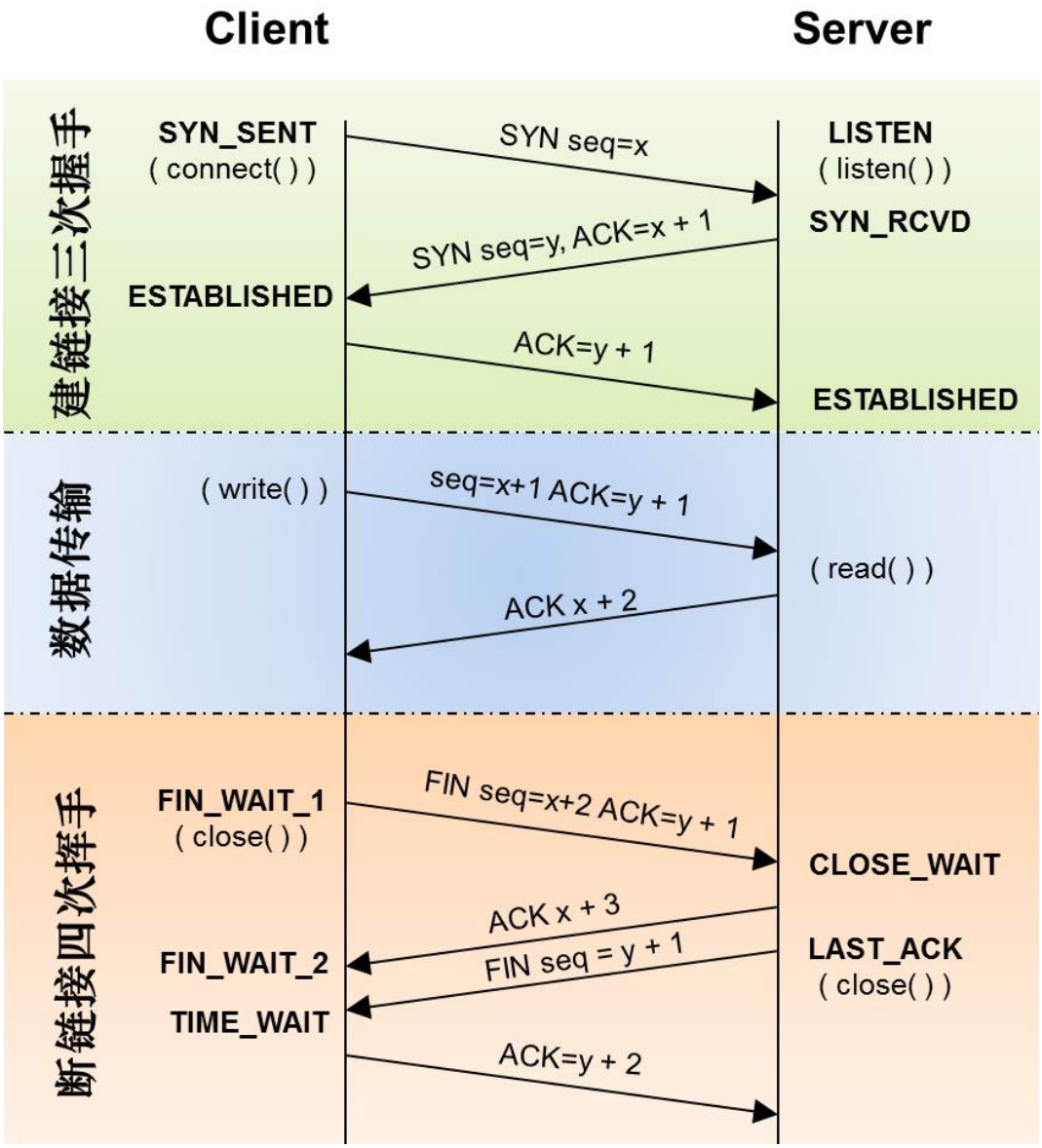


- 建立连接：三次握手
    1. `client -> server`: SYN
    2. `server -> client`: ACK + SYN
    3. `client -> server`: ACK
-



- 断开连接：四次挥手
    1. `client -> server`: FIN
    2. `server -> client`: ACK

3. `server -> client`: FIN
4. `client -> server`: ACK



3. 生成 Request 报文
4. Client 发送 Request 报文
5. Server 接收报文
6. 通过 WSGI 解析报文, 获得 Request 对象
7. Django、Flask 等应用程序进行逻辑处理
    0. 生成 Request 对象
    1. process_request (Middleware)
    2. URL match
    3. process_views (Middleware)
    4. Views --> process_exception
    5. Redner Template
    6. Response
    7. process_response (Middleware)
8. 从 Response 对象生成报文

9. Server 返回报文给 Client
10. 关闭连接
11. 解析、渲染 HTML 页面

4. 最简单的 Server

- HTTP Server
    - 创建、关闭连接
    - 发送、接收数据
- WSGI: 沟通 HTTPServer 和 Web 应用程序 之间的接口
- Web Application
    - 网络程序的功能和逻辑
    - 解析用户请求, 生成 HTML 页面

5. 常见 Header 字段

Header Names	**-**
Accept	text/plain
Accept-Charset	utf-8
Accept-Encoding	gzip (Content-Encoding)
Accept-Language	zh-CN en-US
Cache-Control	true,max-age
Connection	keep-alive
Content-Length	请求体/响应体的长度
Content-Type	用于指定响应的HTTP内容类型. 如果未指定 默认为 text/html
User-Agent	浏览器标识
Cookie	'bid=YgKWanWEzlI;utmz=30149280.152516787;...'

6. HTTP 状态码

Code	Description	-
200	OK	成功
301	Moved Permanently	重定向 (永久迁移)
302	Moved Temporarily	重定向 (临时迁移)
303	See Other	重定向 (非 GET 请求的重定向)
400	Bad Request	客户端请求错误
403	Forbidden	拒绝访问
404	Not Found	找不到页面
500	Internal Server Error	服务器内部错误
502	Bad Gateway	网关错误
503	Service Unavailable	服务器维护或者过载
504	Gateway Timeout	请求超时

7. GET 和 POST

- 表象
    - GET 在浏览器可以回退, 而 POST 则会再次提交请求
    - GET 的 URL 可以被 Bookmark, 而 POST 不可以.
    - GET 请求会被浏览器主动缓存, 而 POST 不会, 除非手动设置.
    - GET 请求参数会被完整保留在浏览器历史记录里, 而 POST 中的参数不会被保留.
    - GET 请求的数据只能进行 URL 编码, 而 POST 支持多种编码方式.
    - GET 请求在 URL 中传送的参数是有长度限制的 (URL 的最大长度是 2048 个字符), 而 POST 没有.
    - 对参数的数据类型, GET 只接受 ASCII 字符, 而 POST 没有限制.
    - GET 比 POST 更不安全, 因为参数直接暴露在URL上, 所以不能用来传递敏感信息.
    - GET 参数通过 URL 传递, POST 放在 Request body 中.
- 深层
    - GET 产生一个TCP数据包；POST产生两个TCP数据包.
    - GET: 浏览器会把 http 的 header和data一并发送出去, 服务器响应 200（返回数据）；
    - POST: 浏览器先发送 header, 服务器响应 100 continue, 浏览器再发送

data, 服务器响应200 ok（返回数据）.

8. cookie 和 session

- 无状态协议的无奈之举
  - 通信如同一次无法看到脸的握手, 如何识别用户
- 异同
  1. session 在服务器端, cookie 在客户端（浏览器）
  2. session 默认被存在在服务器的一个文件里（不是内存）
  3. session 的运行依赖 session id, 而 session id 是存在 cookie 中的, 也就是说, 如果浏览器禁用了 cookie , 同时 session 也会失效（但是可以通过其它方式实现, 比如在 url 中传递 session_id）
  4. session 可以放在 文件、数据库、或内存中都可以.
  5. 用户验证这种场合一般会用 session
- 产生过程
  1. 客户端请求
  2. 服务器产生 session_id, 并传回浏览器
  3. 浏览器将 session_id 写入 cookie
  4. 后续请求会写入 Header
- 使用

9. RESTful

- 一种网络软件架构风格, 而非标准
- 用 URL 定位一个网络资源
- 用 HTTP 描述对资源的操作
- 四个动词
  - GET: 用来获取资源
  - POST: 用来新建资源
  - PUT: 用来更新资源
  - DELETE: 用来删除资源
- 误区
  - URL 中使用动词
  - URL 中出现版本号

10. HTTPS

- 优点
  - 防窃听: 建立一个信息安全通道，来保证数据传输的安全
  - 防篡改: 防止内容被第三方修改
  - 防冒充: 确认网站的真实性
- 缺点
  - 加密、解密消耗 CPU
  - 握手过程繁琐

- SSL / TLS (安全套接字层)
- 加密算法
    - 对称加密: TEA, AES, 3DES

```
text: abcdefg
        |  ^
        v  |
key:    1234
        |  ^
        v  |
new:  hasjdkfhasdf
```

    - 非对称加密: RSA, ED25519

```
text:   abcdefghijklmn
          |       ^
          v       |
pub_key: 123      |
pri_key:  |   12345678901234567890546789
          |       ^
          v       |
new: ajsgdpfqibwfmbsdlkfjbq;ejkwbf;qkbfd
```

- Let's Encrypt: https://letsencrypt.org/
- 传输过程