什么是跨域

跨域，指的是从一个域名去请求另外一个域名的资源。即跨域名请求！跨域时，浏览器不能执行其他域名网站的脚本，是由浏览器的同源策略造成的，是浏览器施加的安全限制。
跨域的严格一点来说就是只要协议，域名，端口有任何一个的不同，就被当作是跨域。

同源策略

同源策略是由 Netscape 公司提出的一个著名的安全策略，所有支持 JavaScript 的浏览器都会使用这个策略。所谓同源是指，域名，协议，端口相同。当页面在执行一个脚本时会检查访问的资源是否同源，如果非同源，那么在请求数据时，浏览器会在控制台中报一个异常，提示拒绝访问。
同源策略一般又分为以下两种：
DOM 同源策略：禁止对不同源页面 DOM 进行操作。这里主要场景是 iframe 跨域的情况，不同域名的 iframe 是限制互相访问的。XmlHttpRequest 同源策略：禁止使用 XHR 对象向不同源的服务器地址发起 HTTP 请求。

如何解决跨域问题

在 Django 中间件的第一个位置使用 Django-cors-headers 中间件, 并且在 install-app 配置中添加 cors-headers, 然后设置允许访问的白名单.

简述什么是 FBV 和 CBV

FBV 和 CBV 本质是一样的，基于函数的视图叫做 FBV，基于类的视图叫做 CBV
在 python 中使用 CBV 的优点：
提高了代码的复用性，可以使用面向对象的技术，比如 Mixin（多继承）
可以用不同的函数针对不同的 HTTP 方法处理，而不是通过很多 if 判断，提高代码可读性

什么是 MVVM 模型

MVVM 是 Model-View-ViewModel 的简写。即模型-视图-视图模型。【模型】指的是后端传递的数据。【视图】指的是所看到的页面。【视图模型】mvvm 模式的核心，它是连接 view 和 model 的桥梁。它有两个方向：一是将【模型】转化成【视图】，即将后端传递的数据转化成所看到的页面。实现的方式是：数据绑定。二是将【视图】转化成【模型】，即将所看到的页面转化成后端的数据。实现的方式是：DOM 事件监听。这两个方向都实现的，我们称之为数据的双向绑定。

使用 orm 和原生 sql 的优缺点

1.orm 的开发速度快,操作简单。使开发更加对象化
执行速度慢。处理多表联查等复杂操作时,ORM 的语法会变得复杂
2.sql 开发速度慢,执行速度快。性能强

什么是 nginx

nginx 是一个开源的，支持高性能，高并发的 web 服务器和代理服务软件。
nginx 还可以作为反向代理，负载均衡，以及缓存服务使用。
优点：
支持高并发，能支持几万并发连接, 资源消耗少，
可以做 http 反向代理和负载均衡
支持异步网络 i/o 事件模型 epoll
apache 非常普通的 WEB 服务器 对高并发没有太大的支持

Nginx 负载均衡概述

Web 服务器，直接面向用户，往往要承载大量并发请求，单台服务器难以负荷，我使用多台 WEB 服务器组成集群，前端使用 Nginx 负载均衡，将请求分散的打到我们的后端服务器集群中，实现负载的分发。那么会大大提升系统的吞吐率和请求性能.
Nginx 要实现负载均衡需要用到 proxy_pass 代理模块配置
Nginx 负载均衡与 Nginx 代理不同地方在于:

Nginx 代理仅代理一台服务器，而 Nginx 负载均衡则是将客户端请求代理转发至一组 upstream 虚拟服务池
Nginx 可以配置代理多台服务器，当一台服务器宕机之后，仍能保持系统可用。

nginx 平滑重启

./nginx -s reload　　# 平滑重启 ，修改了 nginx.conf 之后，可以不重启服务，加载新的配置

nginx 正向代理与反向代理

代理服务器一般指局域网内部的机器通过代理服务器发送请求到互联网上的服务器，代理服务器一般作用在客户端.

正向代理

某个网站我访问不了, 但是中间有个代理服务器能够访问到这个网站, 我就可以先连上这个代理服务器, 然后告诉代理服务器我需要那个网站上的信息之类的, 让代理服务器去帮我取数据然后返回给我.

反向代理

反向代理服务器作用在服务器端，它在服务器端接收客户端的请求，然后将请求分发给具体的服务器进行处理，然后再将服务器的相应结果反馈给客户端。Nginx 就是一个反向代理服务器软件。(用户并不知道 nginx 只是一个中间代理服务器的身份, 所以将请求发至 nginx 服务器, 然后 nginx 服务器将请求分发给目标服务器去获取相关数据并返回给客户端)

区别

位置不同

正向代理，架设在客户机和目标主机之间；

反向代理，架设在服务器端；

代理对象不同

正向代理，代理客户端，服务端不知道实际发起请求的客户端；

反向代理，代理服务端，客户端不知道实际提供服务的服务端；

正向代理的作用

1. 访问原来无法访问的资源
2. 用作缓存，加速访问速度
3. 对客户端访问授权，上网进行认证
4. 代理可以记录用户访问记录（上网行为管理），对外隐藏用户信息

反向代理的作用

1. 保护内网安全
2. 负载均衡
3. 缓存，减少服务器的压力

负载均衡策略

1.轮询

每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器宕机，能自动剔除

2.ip_hash

每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器，可以解决 session 问题

3.最少连接(least_conn)

下一个请求将被分派到活动链接数量较少的服务器

4.fair

按后端服务器的响应时间来分配请求，响应时间短的优先分配

5.url_hash

按访问 url 的 hash 结果来分配请求，使每个 url 定向到同一个后端服务器，后端服务器为缓存时比较有效

6.加权轮询(weight)

在配置的 server 后面加个 weight=number, weight 值越大，分配到的访问几率越高

异步非阻塞机制

每个工作进程使用异步非阻塞方式 ，可以处理多个客户端请求 。

当某个工作进程接收到客户端的请求以后，调用 IO 进行处理，如果不能立即得到结果，就去处理其他请求 （即为非阻塞 ）；而客户端在此期间也无需等待响应 ，可以去处理其他事情（即为 异步 ）。

当 IO 返回时，就会通知此工作进程 ；该进程得到通知，暂时挂起当前处理的事务去响应客户端请求 。

nginx、WSGI、uwsgi、uWSGI、django

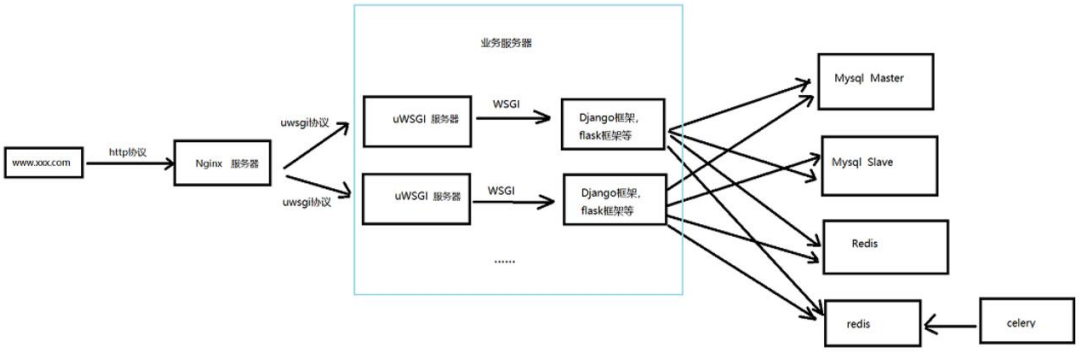
WSGI 一种协议，描述 web 应用程序(Django, flask 等框架写的程序)和 web 服务器(nginx,uWWSGI)之间通信的规则, 让应用程序和后端服务器顺利通信.

运行在 wsgi 上的 web 框架有 bottle，flask，django

uwsgi uwsgi 是一种线路协议而不是通信协议，在此常用于在 uWSGI 服务器与其他网络服务器(比如 nginx)之间的数据通信。uwsgi 协议是一个 uWSGI 服务器自有的协议，它用于定义传输信息的类型。
uWSGI 是一个 Web 服务器，它实现了 WSGI 协议、uwsgi、http 等协议。用于接收前端服务器转发的动态请求并处理后发给 web 应用程序。
nginx web 服务器，更加安全，更好的处理处理静态资源，缓存功能，负载均衡，因此 nginx 的强劲性能，配合 uWSGI 服务器会更加安全，性能有保障。
django 高级的 python web 框架，用于快速开发，解决 web 开发的大部分麻烦，程序员可以更专注业务逻辑，无须重新造轮子

流程

Nginx 接受来自客户端的 Http 请求发送给 uWSGI，uWSGI 处理请求并将关键信息传递给 web 应用(django，flask 等)，应用返回 Response 经由 uWSGI 发送给 Nginx，Nginx 再发送给客户端。
1 首先 nginx 是对外的服务接口，外部浏览器通过 url 访问 nginx,
2 nginx 接收到浏览器发送过来的 http 请求，将包进行解析，分析 url，如果是静态文件请求就直接访问用户给 nginx 配置的静态文件目录，直接返回用户请求的静态文件，如果不是静态文件，而是一个动态的请求，那么 nginx 就将请求转发给 uwsgi, uwsgi 接收到请求之后将包进行处理，处理成 wsgi 可以接受的格式，并发给 wsgi, wsgi 根据请求调用应用程序的某个文件的某个函数，最后处理完将返回值再次交给 uwsgi,uwsgi 将返回值进行打包，打包成 uwsgi 能够接收的格式，uwsgi 接收 wsgi 发送的请求，并转发给 nginx, nginx 最终将返回值返回给浏览器。
3 要知道第一级的 nginx 并不是必须的，uwsgi 完全可以完成整个的和浏览器交互的流程，但是要考虑到某些情况
(1) 安全问题，程序不能直接被浏览器访问到，而是通过 nginx,nginx 只开放某个接口，uwsgi 本身是内网接口，这样运维人员在 nginx 上加上安全性的限制，可以达到保护程序的作用。
(2)负载均衡问题，一个 uwsgi 很可能不够用，即使开了多个 work 也是不行，毕竟一台机器的 cpu 和内存都是有限的，有了 nginx 做代理，一个 nginx 可以代理多台 uwsgi 完成 uwsgi 的负载均衡。
(3)静态文件问题，用 django 或是 uwsgi 这种东西来负责静态文件的处理是很浪费的行为，而且他们本身对文件的处理也不如 nginx 好，所以整个静态文件的处理都直接由 nginx 完成，静态文件的访问完全不去经过 uwsgi 以及其后面的东西。



uwsgi 热加载启动

uwsgi --http :9000 --module 项目名.wsgi --py-autoreload=1

集群和分布式

集群: 一组运行各自服务的独立服务器所组成的一个较大的计算机服务系统, 每一台服务器都可以向用户提供数据服务(一群机器做相同的事情)
集群作用: 主要用来分担请求的压力, 在不同的服务器上部署相同的服务来分担客户端的请求; 高可用(某台服务器宕机不影响继续提供服务)
分布式: 将一套系统拆分成多个不同的子系统在不同的服务器上,各个服务器之间必然存在着通信协调.
分布式优势: 服务可以按照业务类型拆分成多个应用,可以很方便地在任何一个环节拓展应用而且不会影响到其他应用, 解决了集中式不便拓展的弊端.但是相应的开发测试运维成本也提高了.

哨兵(redis 高可用)

Redis-Sentinel 是 redis 官方推荐的高可用性解决方案.
当用 redis 作 master-slave 的高可用时，如果 master 本身宕机，redis 本身或者客户端都没有实现主从切换的功能。而 redis-sentinel 就是一个独立运行的进程，用于监控多个 master-slave 集群，自动发现 master 宕机，进行自动切换 slave > master。

redis 集群

redis 集群完全去中心化，将 redis 所有的 key 分成了 16384 个槽位，每个 redis 实例负责一部分 slot，集群中的所有信息通过节点数据交换而更新。

redis 实例集群主要思想是将 redis 数据的 key 进行散列，通过 hash 函数特定的 key 会映射到指定的 redis 节点上(将数据分片存储在多个 redis 实例中(分布式 db)，每一片就是一个 redis 实例)

docker

docker 是 linux 容器的一种封装，提供简单易用的容器使用接口。docker 将应用程序与程序的依赖，打包在一个文件里面, 运行这个文件就会生成一个虚拟容器。程序运行在虚拟容器里，就不用担心环境问题了。

优点:

（1）启动快

容器里面的应用，直接就是底层系统的一个进程，而不是虚拟机内部的进程。所以，启动容器相当于启动本机的一个进程，而不是启动一个操作系统，速度就快很多。

（2）资源占用少

容器只占用需要的资源，不占用那些没有用到的资源；虚拟机由于是完整的操作系统，不可避免要占用所有资源。另外，多个容器可以共享资源，虚拟机都是独享资源。

（3）体积小

容器只要包含用到的组件即可，而虚拟机是整个操作系统的打包，所以容器文件比虚拟机文件要小很多。

(4) 迁移更加方便

Docker 可以在 很多平台上运行，无论是物理机、虚拟机、公有云、私有云，甚至是笔记本，其运行结果是一致的。就可以很轻易的将在一个平台上运行的应用，迁移到另一个平台上.

部署项目

nginx + uwsgi + Django + supervisor

部署路飞：

1.安装 python3 环境
2.安装 mysql，redis，nginx
3.部署前端
    1.安装 node.js 的环境
    2.安装依赖包
    3.修改 axios 的发送的端口接口
    4.生成 dist 静态项目文件
4.部署后端
    1.安装 virtualenv
    2.创建虚拟环境
    3.安装 django 和 uwsgi，以及项目的依赖包
    4.修改 uwsgi 的配置文件
    5.通过 uwsgi -ini 配置文件启动 django 项目
5.配置 nginx
    1.创建两个虚拟主机，分别监听 80 和 8000 端口
    2.访问 80 端口是访问 vue
    3.访问 8000 端口是 vue 发起的 8000 端口请求，反向代理到 9000 的 uwsgi
6.启动 nginx,mysql,redis
7.通过 supervisor 来管理

mysql 主从同步

1. 主数据库写入数据之后， 会有 data changes(数据变化)记录
2. 有变化记录之后，将增删改的一些 sql 语句记录到本地的 Binary log(二进制日志)中
3. 从库会一直开启着一个线程
4. 通过线程去读取这个二进制日志的内容
5. 从库会将数据写入到自己的 Relay log(中继日志)中
6. 从库会将中继日志中的操作转化为 SQL thread(SQL 语句)
7. 通过转化的 SQL 语句写入到自己的数据库, 两边的数据就一致了