目录

第三阶段

1、用 Python 匹配 HTML tag 的时候，<.*>和<.*?>有什么区别？

答：术语叫贪婪匹配(<.*>)和非贪婪匹配(<.*?>)
例如:
test
<.*> :
test
<.*?> :

2、三次握手

客户端通过向服务器端发送一个 SYN 来创建一个主动打开，作为三路握手的一部分。客户端把这段连接的序号设定为随机数?A。
服务器端应当为一个合法的 SYN 回送一个 SYN/ACK。ACK?的确认码应为?A+1，SYN/ACK?包本身又有一个随机序号?B。
最后，客户端再发送一个 ACK。当服务端受到这个 ACK 的时候，就完成了三路握手，并进入了连接创建状态。此时包序号被设定为收到的确认号?A+1，而响应则为 B+1。

3、四次挥手

由于 TCP 连接是全双工的，因此每个方向都必须单独进行关闭。这原则是当一方完成它的数据发送任务后就能发送一个 FIN 来终止这个方向的连接。收到一个 FIN 只意味着这一方向上没有数据流动，一个 TCP 连接在收到一个 FIN 后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。

a.TCP 客户端发送一个 FIN，用来关闭客户到服务器的数据传送。

b.服务器收到这个 FIN，它发回一个 ACK，确认序号为收到的序号加 1。和 SYN 一样，一个 FIN 将占用一个序号。

c.服务器关闭客户端的连接，发送一个 FIN 给客户端。

d.客户端发回 ACK 报文确认，并将确认序号设置为收到序号加 1。

4、ARP 协议

地址解析协议(Address Resolution Protocol):根据 IP 地址获取物理地址的一个 TCP/IP 协议

5、urllib 和 urllib2 的区别

urllib 提供 urlencode 方法用来 GET 查询字符串的产生，而 urllib2 没有。这是为何 urllib 常和 urllib2 一起使用的原因。
urllib2 可以接受一个 Request 类的实例来设置 URL 请求的 headers，urllib 仅

可以接受 URL。这意味着，你不可以伪装你的 User?Agent 字符串等。

6、Post 和 Get

下面的表格比较了两种 HTTP 方法：GET 和 POST。

	GET	POST
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
书签	可收藏为书签	不可收藏为书签
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。
历史	参数保留在浏览器历史中。	参数不会保存在浏览器历史中。
对数据长度的限制	是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。	无限制。
对数据类型的限制	只允许 ASCII 字符。	没有限制。也允许二进制数据。
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。 在发送密码或其他敏感信息时绝不要使用 GET！	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。
可见性	数据在 URL 中对所有人都是可见的。	数据不会显示在 URL 中。

7、Cookie 和 Session

Cookie Session 储存位置 客户端 服务器端
目的 跟踪会话，也可以保存用户偏好设置或者保存用户名密码等 跟踪会话
安全性 不安全 安全
session 技术是要使用到 cookie 的，之所以出现 session 技术，主要是为了安全。

8、apache 和 nginx 的区别

Nginx 相对 apache?的优点：
轻量级，同样起 web 服务，比 apache 占用更少的内存及资源
抗并发，nginx 处理请求是异步非阻塞的，支持更多的并发连接，而 apache 则是阻塞型的，在高并发下 nginx 能保持低资源低消耗高性能，配置简洁，高度模块化的设计，编写模块相对简单
Apache 相对 nginx 的优点：
Rewrite ，比 nginx 的 rewrite 强大
模块超多，基本想到的都可以找到
少 bug ，nginx 的 bug 相对较多
超稳定

9、网站用户密码保存

明文保存
明文 hash 后保存,如 md5
MD5+Salt 方式,这个 salt 可以随机
知乎使用了 Bcrypy(好像)加密

10、HTTP 和 HTTPS

状态码 定义
1xx 报告 接收到请求，继续进程
2xx 成功 步骤成功接收，被理解，并被接受
3xx 重定向 为了完成请求,必须采取进一步措施
4xx 客户端出错 请求包括错的顺序或不能完成
5xx 服务器出错 服务器无法完成显然有效的请求
403: Forbidden
404: Not?Found
HTTPS 握手,对称加密,非对称加密,TLS/SSL,RSA

11、XSRF 和 XSS

CSRF(Cross-site?request?forgery)跨站请求伪造
XSS(Cross?Site?Scripting)跨站脚本攻击
CSRF 重点在请求,XSS 重点在脚本

12、幂等 Idempotence

HTTP 方法的幂等性是指一次和多次请求某一个资源应该具有同样的副作用。(注意是副作用)
GET http://www.bank.com/account/123456，不会改变资源的状态，不论调用一次还是 N 次都没有副作用。请注意,这里强调的是一次和 N 次具有相同的副作用，而不是每次 GET 的结果相同。GET http://www.news.com/latest-news 这个 HTTP 请求可能会每次得到不同的结果，但它本身并没有产生任何副作用，因而是满足幂等性的。
DELETE 方法用于删除资源，有副作用，但它应该满足幂等性。比如：DELETE http://www.forum.com/article/4231，调用一次和 N 次对系统产生的副作用是相同的，即删掉 id 为 4231 的帖子；因此，调用者可以多次调用或刷新页面而不必担心引起错误。
POST 所对应的 URI 并非创建的资源本身，而是资源的接收者。比如：POST http://www.forum.com/articles 的语义是在 http://www.forum.com/articles 下创建一篇帖子，HTTP 响应中应包含帖子的创建状态以及帖子的 URI。两次相同的 POST 请求会在服务器端创建两份资源，它们具有不同的 URI；所以，POST 方法不具备幂等性。
PUT 所对应的 URI 是要创建或更新的资源本身。比如：PUT

http://www.forum/articles/4231 的语义是创建或更新 ID 为 4231 的帖子。对同一 URI 进行多次 PUT 的副作用和一次 PUT 是相同的；因此，PUT 方法具有幂等性。

13、SOAP

SOAP（原为 Simple Object Access Protocol 的首字母缩写，即简单对象访问协议）是交换数据的一种协议规范，使用在计算机网络 Web 服务（web service）中，交换带结构信息。SOAP 为了简化网页服务器（Web Server）从 XML 数据库中提取数据时，节省去格式化页面时间，以及不同应用程序之间按照 HTTP 通信协议，遵从 XML 格式执行资料互换，使其抽象于语言实现、平台和硬件。

14、RPC

RPC（Remote Procedure Call Protocol）——远程过程调用协议，它是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。RPC 协议假定某些传输协议的存在，如 TCP 或 UDP，为通信程序之间携带信息数据。在 OSI 网络通信模型中，RPC 跨越了传输层和应用层。RPC 使得开发包括网络分布式多程序在内的应用程序更加容易。
总结:服务提供的两大流派.传统意义以方法调用为导向通称 RPC。为了企业 SOA,若干厂商联合推出 webservice,制定了 wsdl 接口定义,传输 soap.当互联网时代,臃肿 SOA 被简化为 http+xml/json.但是简化出现各种混乱。以资源为导向,任何操作无非是对资源的增删改查，于是统一的 REST 出现了.
进化的顺序: RPC -> SOAP -> RESTful

15、CGI 和 WSGI

CGI 是通用网关接口，是连接 web 服务器和应用程序的接口，用户通过 CGI 来获取动态数据或文件等。
CGI 程序是一个独立的程序，它可以用几乎所有语言来写，包括 perl，c，lua，python 等等。
WSGI, Web Server Gateway?Interface，是 Python 应用程序或框架和 Web 服务器之间的一种接口，WSGI 的其中一个目的就是让用户可以用统一的语言(Python)编写前后端。

16、中间人攻击

中间人攻击（Man-in-the-middle attack，通常缩写为 MITM）是指攻击者与通讯的两端分别创建独立的联系，并交换其所收到的数据，使通讯的两端认为他们正在通过一个私密的连接与对方直接对话,但事实上整个会话都被攻击者完全控制。

17、c10k 问题

所谓 c10k 问题，指的是服务器同时支持成千上万个客户端的问题，也就是 concurrent 10 000 connection（这也是 c10k 这个名字的由来）。

18、socket

Socket=Ip address+ TCP/UDP + port

19、浏览器缓存

简单来说，浏览器缓存就是把一个已经请求过的 Web 资源（如 html 页面，图片，js，数据等）拷贝一份副本储存在浏览器中。缓存会根据进来的请求保存输出内容的副本。当下一个请求来到的时候，如果是相同的 URL，缓存会根据缓存机制决定是直接使用副本响应访问请求，还是向源服务器再次发送请求。比较常见的就是浏览器会缓存访问过网站的网页，当再次访问这个 URL 地址的时候，如果网页没有更新，就不会再次下载网页，而是直接使用本地缓存的网页。只有当网站明确标识资源已经更新，浏览器才会再次下载网页

20、Ajax

AJAX,Asynchronous JavaScript and XML（异步的 JavaScript 和 XML），是与在不重新加载整个页面的情况下，与服务器交换数据并更新部分网页的技术。

21、unix 进程间通信方式(IPC)

管道（Pipe）：管道可用于具有亲缘关系进程间的通信，允许一个进程和另一个与它有共同祖先的进程之间进行通信。
命名管道（named?pipe）：命名管道克服了管道没有名字的限制，因此，除具有管道所具有的功能外，它还允许无亲缘关系进程间的通信。命名管道在文件系统中有对应的文件名。命名管道通过命令 mkfifo 或系统调用 mkfifo 来创建。
信号（Signal）：信号是比较复杂的通信方式，用于通知接受进程有某种事件发生，除了用于进程间通信外，进程还可以发送信号给进程本身；linux 除了支持 Unix 早期信号语义函数 sigal 外，还支持语义符合 Posix.1 标准的信号函数 sigaction（实际上，该函数是基于 BSD 的，BSD 为了实现可靠信号机制，又能够统一对外接口，用 sigaction 函数重新实现了 signal 函数）。
消息（Message）队列：消息队列是消息的链接表，包括 Posix 消息队列 system?V 消息队列。有足够权限的进程可以向队列中添加消息，被赋予读权限的进程则可以读走队列中的消息。消息队列克服了信号承载信息量少，管道只能承载无格式字节流以及缓冲区大小受限等缺
共享内存：使得多个进程可以访问同一块内存空间，是最快的可用 IPC 形式。是针对其他通信机制运行效率较低而设计的。往往与其它通信机制，如信号量结合使用，来达到进程间的同步及互斥。
内存映射（mapped?memory）：内存映射允许任何多个进程间通信，每一个使用该机制的进程通过把一个共享的文件映射到自己的进程地址空间来实现它。
信号量（semaphore）：主要作为进程间以及同一进程不同线程之间的同步手段。

套接口（Socket）：更为一般的进程间通信机制，可用于不同机器之间的进程间通信。起初是由 Unix 系统的 BSD 分支开发出来的，但现在一般可以移植到其它类 Unix 系统上：Linux 和 System?V 的变种都支持套接字。

22.cookie 和 session 的区别？

1、 cookie 数据存放在客户的浏览器上，session 数据放在服务器上。

2、 cookie 不是很安全，别人可以分析存放在本地的 cookie 并进行 cookie 欺骗考虑到安全应当使用 session。

3、 session 会在一定时间内保存在服务器上。当访问增多，会比较占用服务器的性能考虑到减轻服务器性能方面，应当使用 cookie。

4、 单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie。

5、 建议：将登陆信息等重要信息存放为 SESSION 其他信息如果需要保留，可以放在 cookie 中

23.HTTP 协议状态码有什么用，列出你知道的 HTTP 协议的状态码，然后讲出他们都表示什么意思？

通过状态码告诉客户端服务器的执行状态，以判断下一步该执行什么操作。
常见的状态机器码有：
100-199：表示服务器成功接收部分请求，要求客户端继续提交其余请求才能完成整个处理过程。
200-299：表示服务器成功接收请求并已完成处理过程，常用 200（OK 请求成功）。
300-399：为完成请求，客户需要进一步细化请求。302（所有请求页面已经临时转移到新的 url）。
304、307（使用缓存资源）。
400-499：客户端请求有错误，常用 404（服务器无法找到被请求页面），403（服务器拒绝访问，权限不够）。
500-599：服务器端出现错误，常用 500（请求未完成，服务器遇到不可预知的情况）。

24.说一下什么是 tcp 的 2MSL？

主动发送 fin 关闭的一方，在 4 次挥手最后一次要等待一段时间我们称这段时间为 2MSL
TIME_WAIT 状态的存在有两个理由：
1.让 4 次挥手关闭流程更加可靠
2.防止丢包后对后续新建的正常连接的传输造成破坏

25.说说 HTTP 和 HTTPS 区别？

HTTP 协议传输的数据都是未加密的，也就是明文的，因此使用 HTTP 协议传输隐私信息非常不安
全，为了保证这些隐私数据能加密传输，于是网景公司设计了 SSL（Secure Sockets Layer）协议用于对 HTTP 协议传输的数据进行加密，从而就诞生了 HTTPS。简单来说，HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，要比 http 协议安全。
HTTPS 和 HTTP 的区别主要如下：

1、https 协议需要到 ca 申请证书，一般免费证书较少，因而需要一定费用。

2、http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议。

3、http 和 https 使用的是完全不同的连接方式，用的端口也不一样，前者是 80，后者是 443。

4、http 的连接很简单，是无状态的；HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全。

26.谈一下 HTTP 协议以及协议头部中表示数据类型的字段？

HTTP 协议是 Hyper Text Transfer Protocol（超文本传输协议）的缩写，是用于从万维网
（WWW:World Wide Web）服务器传输超文本到本地浏览器的传送协议。

HTTP 是一个基于 TCP/IP 通信协议来传递数据（HTML 文件， 图片文件， 查询结果等）。
HTTP 是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体
信息系统。它于 1990 年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在 WWW 中使用的是 HTTP/1.0 的第六版，HTTP/1.1 的规范化工作正在进行之中，而且 HTTP-NG(Next
Generation of HTTP)的建议已经提出。
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。
表示数据类型字段： Content-Type

27.HTTP 常见请求头？

1.Host (主机和端口号)
2.Connection (链接类型)
3.Upgrade-Insecure-Requests (升级为 HTTPS 请求)
4.User-Agent (浏览器名称)
5.Accept (传输文件类型)
6.Referer (页面跳转处)
7.Accept-Encoding（文件编解码格式）
8.Cookie （Cookie）
9.x-requested-with :XMLHttpRequest （是 Ajax 异步请求)

28.简述浮动的特征和清除浮动的方法？

浮动的特征：
浮动元素有左浮动(float:left)和右浮动(float:right)两种。
浮动的元素会向左或向右浮动，碰到父元素边界、其他元素才停下来。相邻浮动的块元素可以并在一行，超出父级宽度就换行。浮动让行内元素或块元素转化为有浮动特性的行内块元素(此时不会有行内块元素间隙问题)。
父元素如果没有设置尺寸(一般是高度不设置)，父元素内整体浮动的子元素无法撑开父元素，父元素需要清除浮动。清除浮动的方法：
父级上增加属性 overflow：hidden。
在最后一个子元素的后面加一个空的 div，给它样式属性 clear:both。
使用成熟的清浮动样式类，clearfix。

1..clearfix:after,.clearfix:before{ content: "";display: table;}
2..clearfix:after{ clear:both;} 3. .clearfix{zoom:1;}

29.AJAX 试什么？如何使用 AJAX？

ajax(异步的 javascript 和 xml) 能够刷新局部网页数据而不是重新加载整个网页。
第一步，创建 xmlhttprequest 对象，var xmlhttp =new XMLHttpRequest ();XMLHttpRequest
对象用来和服务器交换数据。
第二步,使用 xmlhttprequest 对象的 open()和 send()方法发送资源请求给服务器。第三步,使用 xmlhttprequest 对象的 responseText 或 responseXML 属性获得服务器的响应。
第四步，onreadystatechange 函数，当发送请求到服务器，我们想要服务器响应执行一些功能就
需要使用 onreadystatechange 函数，每次 xmlhttprequest 对象的
readyState 发生改变都会触发
onreadystatechange 函数。

30. Flask 中正则 URL 的实现？

@app.route('<URL>')中 URL 显式支持 string、int、float、path 4 种类型，隐式支持正则。
第一步：写正则类，继承 BaseConverter，将匹配到的值设置为 regex 的值。

```
1. class RegexUrl(BaseConverter):
2. def __init__(self, url_map, *args):
3. super(RegexUrl, self).__init__(url_map)
4. self.regex = args[0]
```

第二步：把正则类赋值给我们定义的正则规则。

```
5. app.url_map.converters['re'] = RegexUrl
```

第三步：在 URL 中使用正则。

```
6. @app.route('/regex/<re("[a-z]{3}"):id>') 7. def
regex111(id):
8. return 'id:%s'%id
```

31.Flask 中请求上下文和应用上下文的区别和作用？

current_app、g 是应用上下文。 request、session 是请求上下文。手动创建上下文的两种方法：

```
1.with app.app_context()
```

2.app = current_app._get_current_object()

两者区别：

请求上下文：保存了客户端和服务器交互的数据。

应用上下文：flask 应用程序运行过程中，保存的一些配置信息，比如程序名、数据库连接、应用

信息等。

两者作用：

请求上下文(request context)：

Flask 从客户端收到请求时，要让视图函数能访问一些对象，这样才能处理请求。请求对象是一个很好的例子，它封装了客户端发送的 HTTP 请求。

要想让视图函数能够访问请求对象，一个显而易见的方式是将其作为参数传入视图函数，不过

这会导致程序中的每个视图函数都增加一个参数，除了访问请求对象，如果视图函数在处理请求时还要访问其他对象，情况会变得更糟。为了避免大量可有可无的参数把视图函数弄得一团糟，Flask 使用上下文临时把某些对象变为全局可访问。

应用上下文(application context)：

它的字面意思是 应用上下文，但它不是一直存在的，它只是 request context 中的一个对 app 的代理(人)，所谓 local proxy。它的作用主要是帮助 request 获取当前的应用，它是伴 request 而生，随 request 而灭的。

32.Flask 中数据库设置？

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql://root:mysql@127.0.0.1:3306/test'
```

33.常用的 SQLAlchemy 查询过滤器？

过滤器	说明
filter()	把过滤器添加到原查询上，返回一个新查询
filter_by()	把等值过滤器添加到原查询上，返回一个新查询
limit	使用指定的值限定原查询返回的结果
offset()	偏移原查询返回的结果，返回一个新查询
order_by()	根据指定条件对原查询结果进行排序，返回一个新查询
group_by()	根据指定条件对原查询结果进行分组，返回一个新查询

34.对 Flask 蓝图(Blueprint)的理解？

1) 蓝图的定义

蓝图 /Blueprint 是 Flask 应用程序组件化的方法，可以在一个应用内或跨越多个项目共用蓝图。使用蓝图可以极大地简化大型应用的开发难度，也为 Flask 扩展 提供了一种在应用中注册服务的集中式机制。

2) 蓝图的应用场景

1. 把一个应用分解为一个蓝图的集合。这对大型应用是理想的。一个项目可以实例化一个应用对象，初始化几个扩展，并注册一集合的蓝图。
2. 以 URL 前缀和/或子域名，在应用上注册一个蓝图。 URL 前缀/子域名中的参数即成为这个蓝图下的所有视图函数的共同的视图参数（默认情况下）。
3. 在一个应用中用不同的 URL 规则多次注册一个蓝图。
4. 通过蓝图提供模板过滤器、静态文件、模板和其它功能。一个蓝图不一定要实现应用或者视图函数。
5. 初始化一个 Flask 扩展时，在这些情况中注册一个蓝图。

3) 蓝图的缺点

不能在应用创建后撤销注册一个蓝图而不销毁整个应用对象。

4) 使用蓝图的三个步骤

1.创建 一个蓝图对象
2. blue = Blueprint("blue", __name__)
2.在这个蓝图对象上进行操作 ，例如注册路由、指定静态文件夹、注册模板过滤器...

```
1. @blue.route('/') 2. def blue_index():
3.  return 'Welcome to my blueprint'
```

3.在应用对象上注册这个蓝图对象

1. app.register_blueprint(blue, url_prefix='/blue')

35.Flask 中 WTF 表单数据验证？

在 Flask 中，为了处理 web 表单，我们一般使用 Flask-WTF 扩展，它封装了 WTForms，并且它有验证表单数据的功能。
WTForms 支持的 HTML 标准字段：

字段对象	说明
StringField	文本字段
TextAreaField	多行文本字段

字段对象	说明
PasswordField	密码文本字段
HiddenField	隐藏文件字段
DateField	文本字段，值为 datetime.date 文本格式
DateTimeField	文本字段，值为 datetime.datetime 文本格式
IntegerField	文本字段，值为整数
DecimalField	文本字段，值为 decimal.Decimal
FloatField	文本字段，值为浮点数
BooleanField	复选框，值为 True 和 False

RadioField	一组单选框
SelectField	下拉列表
SelectMutipleField	下拉列表，可选择多个值
FileField	文件上传字段
SubmitField	表单提交按钮
FormField	把表单作为字段嵌入另一个表单
字段对象	说明
FieldList	一组指定类型的字段

WTForms 常用验证函数

验证函数	说明
InputRequired	确保字段中有数据
DataRequired	确保字段中有数据并且数据为真
EqualTo	比较两个字段的值，常用于比较两次密码输入
Length	验证输入的字符串长度
NumberRange	验证输入的值在数字范围内
URL	验证 URL

AnyOf	验证输入值在可选列表中
NoneOf	验证输入值不在可选列表中

使用 Flask-WTF 需要配置参数 SECRET_KEY。
CSRF_ENABLED 是为了 CSRF（跨站请求伪造）保护。 SECRET_KEY 用来生成加密令牌，当 CSRF 激活的时候，该设置会根据设置的密匙生成加密令牌。

36.项目接口实现后路由访问不到怎么办？

可以通过 postman 测试工具测试,或者看 log 日志信息找到错误信息的大概位置。

37.Flask 中 url_for 函数？

1.URL 反转：根据视图函数名称得到当前所指向的 url。
2.url_for() 函数最简单的用法是以视图函数名作为参数，返回对应的 url，还可以用作加载静态文件。
1. <link rel="stylesheet"
href="{{url_for('static',filename='css/index.css')}}">
该条语句就是在模版中加载 css 静态文件。
3.url_for 和 redirect 区别
url_for 是用来拼接 URL 的，可以使用程序 URL 映射中保存的信息生成 URL。
url_for() 函数最简单的用法是以视图函数名作为参数， 返回对应的 URL。例如，在示例程序中 hello.py 中调用
url_for('index') 得到的结果是 /。
redirect 是重定向函数，输入一个 URL 后，自动跳转到另一个 URL 所在的地址，例如，你在函数
中写 return redirect('https://www.baidu.com') 页面就会跳转向百度页面。

38. django 中间件的使用？

Django 在中间件中预置了六个方法，这六个方法的区别在于不同的阶段执行，对输入或输出进行干预，方法如下：

1.初始化：无需任何参数，服务器响应第一个请求的时候调用一次，用于确定是否启用当前中间件。

```
1. def __init__():
```

```
2.      pass
```

2.处理请求前：在每个请求上调用，返回 None 或 HttpResponse 对象。

```
1. def process_request(request):
2.      pass
```

3.处理视图前：在每个请求上调用，返回 None 或 HttpResponse 对象。

```
1. def process_view(request, view_func, view_args, view_kwargs):
2.      pass
```

4.处理模板响应前：在每个请求上调用，返回实现了 render 方法的响应对象。

```
1. def process_template_response(request, response):
2.      pass
```

5.处理响应后：所有响应返回浏览器之前被调用，在每个请求上调用，返回 HttpResponse 对象。

```
1. def process_response(request, response):
2.      pass
```

6.异常处理：当视图抛出异常时调用，在每个请求上调用，返回一个 HttpResponse 对象。

```
1. def process_exception(request,exception):
2.      pass
```

39.谈一下你对 uWSGI 和 nginx 的理解？

1.uWSGI 是一个 Web 服务器，它实现了 WSGI 协议、uwsgi、http 等协议。Nginx 中 HttpUwsgiModule 的作用是与 uWSGI 服务器进行交换。WSGI 是一种 Web 服务器网关接口。它是一个 Web 服务器（如 nginx，uWSGI 等服务器）与 web 应用（如用 Flask 框架写的程序）通信的一种

规范。要注意 WSGI / uwsgi / uWSGI 这

三个概念的区分。

WSGI 是一种通信协议。
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的数据通信。
uWSGI 是实现了 uwsgi 和 WSGI 两种协议的 Web 服务器。

2. nginx 是一个开源的高性能的 HTTP 服务器和反向代理：

1.作为 web 服务器，它处理静态文件和索引文件效果非常高；
2.它的设计非常注重效率，最大支持 5 万个并发连接，但只占用很少的内存空间；
3.稳定性高，配置简洁；
4.强大的反向代理和负载均衡功能，平衡集群中各个服务器的负载压力应用。

40.说说 nginx 和 uWISG 服务器之间如何配合工作的？

首先浏览器发起 http 请求到 nginx 服务器，Nginx 根据接收到请求包，进行 url 分析，判断访问的资源类型，如果是静态资源，直接读取静态资源返回给浏览器，如果请求的是动态资源就转交给 uwsgi 服务器，uwsgi 服务器根据自身的 uwsgi 和 WSGI 协议，找到对应的 Django 框架，Django 框架下的应用进行逻辑处理后，将返回值发送到 uwsgi 服务器，然后 uwsgi 服务器再返回给 nginx，最后 nginx 将返回值返回给浏览器进行渲染显示给用户。

41.django 开发中数据库做过什么优化?

1.设计表时，尽量少使用外键，因为外键约束会影响插入和删除性能；
2.使用缓存，减少对数据库的访问；
3.在 orm 框架下设置表时，能用 varchar 确定字段长度时，就别用 text；
4.可以给搜索频率高的字段属性，在定义时创建索引；
5.Django orm 框架下的 Querysets 本来就有缓存的；
6.如果一个页面需要多次连接数据库，最好一次性取出所有需要的数据，减少对数据库的查询次数；
7.若页面只需要数据库里某一个两个字段时，可以用 QuerySet.values()；
8.在模板标签里使用 with 标签可以缓存 Qset 的查询结果。

42.验证码过期时间怎么设置？

将验证码保存到数据库或 session，设置过期时间为 1 分钟，然后页面设置一个倒计时(一般是前端 js 实现 这个计时)的展示，一分钟过后再次点击获取新的信息。

43.Python 中三大框架各自的应用场景？

django：主要是用来搞快速开发的，他的亮点就是快速开发，节约成本，正常的并发量

不过 10000，
如果要实现高并发的话，就要对 django 进行二次开发，比如把整个笨重的框架给拆掉，自己写 socket 实现 http 的通信，底层用纯 c，c++写提升效率，ORM 框架给干掉，自己编写封装与数据库交互的框架，因为啥呢，ORM 虽然面向对象来操作数据库，但是它的效率很低，使用外键来联系表与表之间的查询；

flask：轻量级，主要是用来写接口的一个框架，实现前后端分离，提升开发效率，Flask 本身相当于一个内核，其他几乎所有的功能都要用到扩展（邮件扩展 Flask-Mail，用户认证 Flask-Login），都需要用第三方的扩展来实现。比如可以用 Flask-extension 加入 ORM、窗体验证工具，文件上传、身份验证等。Flask 没有默认使用的数据库，你可以选择 MySQL，也可以用 NoSQL。

其 WSGI 工具箱采用 Werkzeug（路由模块），模板引擎则使用 Jinja2。这两个也是 Flask 框架的核心。Python 最出名的框架要数 Django，此外还有 Flask、Tornado 等框架。虽然 Flask 不是最出名的框架，但是 Flask 应该算是最灵活的框架之一，这也是 Flask 受到广大开发者喜爱的原因。

Tornado： Tornado 是一种 Web 服务器软件的开源版本。Tornado 和现在的主流 Web 服务器框架（包括大多数 Python 的框架）有着明显的区别：它是非阻塞式服务器，而且速度相当快。

得利于其非阻塞的方式和对 epoll 的运用，Tornado 每秒可以处理数以千计的连接，因此 Tornado 是实时 Web 服务的一个 理想框架。

44.django 如何提升性能（高并发）？

对一个后端开发程序员来说，提升性能指标主要有两个一个是并发数，另一个是响应时间网站性能的优化一般包括 web 前端性能优化，应用服务器性能优化，存储服务器优化。

对前端的优化主要有：

1.减少 http 请求，减少数据库的访问量，比如使用雪碧图。

2.使用浏览器缓存，将一些常用的 css，js，logo 图标，这些静态资源缓存到本地浏览器，通过设置 http 头中的 cache-control 和 expires 的属性，可设定浏览器缓存，缓存时间可以自定义。

3 对 html，css，javascript 文件进行压缩，减少网络的通信量。

对我个人而言，我做的优化主要是以下三个方面：

1.合理的使用缓存技术，对一些常用到的动态数据，比如首页做一个缓存，或者某些常用的数据做个缓存，设置一定得过期时间，这样减少了对数据库的压力，提升网站性能。

2.使用 celery 消息队列，将耗时的操作扔到队列里，让 worker 去监听队列里的任务，实现异步操作，比如发邮件，发短信。

3.就是代码上的一些优化，补充：nginx 部署项目也是项目优化，可以配置合适的配置参数，提升效率，增加并发量。

4.如果太多考虑安全因素，服务器磁盘用固态硬盘读写，远远大于机械硬盘，这个技术现在没有普及，主要是固态硬盘技术上还不是完全成熟， 相信以后会大量普及。

5.另外还可以搭建服务器集群，将并发访问请求，分散到多台服务器上处理。

6.最后就是运维工作人员的一些性能优化技术了。

45.什么是 restful api，谈谈你的理解?

REST:Representational State Transfer 的缩写，翻译："具象状态传输"。一般解释为"表现层状态转换"。

REST 是设计风格而不是标准。是指客户端和服务器的交互形式。我们需要关注的重点是如何设计 REST 风格的网络接口。

REST 的特点：

1.具象的。一般指表现层，要表现的对象就是资源。比如，客户端访问服务器，获取的数据就是资源。比如文字、图片、音视频等。

2.表现：资源的表现形式。txt 格式、html 格式、json 格式、jpg 格式等。浏览器通过 URL 确定资源的位置，但是需要在 HTTP 请求头中，用 Accept 和 Content-Type 字段指定，这两个字段是对资源表现的描述。

3.状态转换：客户端和服务器交互的过程。在这个过程中，一定会有数据和状态的转化，这种转化叫做状态转换。其中，GET 表示获取资源，POST 表示新建资源，PUT 表示更新资源，DELETE 表示删除资源。HTTP 协议中最常用的就是这四种操作方式。

RESTful 架构：

1.每个 URL 代表一种资源；

2.客户端和服务器之间，传递这种资源的某种表现层；

3.客户端通过四个 http 动词，对服务器资源进行操作，实现表现层状态转换。

46 如何设计符合 RESTful 风格的 API

一、域名：将 api 部署在专用域名下：

二、版本：将 API 的版本号放在 url 中。

三、路径：路径表示 API 的具体网址。每个网址代表一种资源。

资源作为网址，网址中不能有动词只能有名词，一般名词要与数据库的表名对应。而且名词要使用复数。

四、使用标准的 HTTP 方法：对于资源的具体操作类型，由 HTTP 动词表示。 常用的 HTTP 动词有四个。

GET	SELECT ： 从服务器获取资源。
POST	CREATE ： 在服务器新建资源。
PUT	UPDATE ： 在服务器更新资源。
DELETE	DELETE ： 从服务器删除资源。

五、过滤信息：如果资源数据较多，服务器不能将所有数据一次全部返回给客户端。API 应该提供参数，过滤返回结果。

六、状态码：服务器向用户返回的状态码和提示信息，常用的有：

200 OK ： 服务器成功返回用户请求的数据

201 CREATED ： 用户新建或修改数据成功。

202 Accepted：表示请求已进入后台排队。

400 INVALID REQUEST ： 用户发出的请求有错误。

401 Unauthorized ： 用户没有权限。

403 Forbidden ： 访问被禁止。

404 NOT FOUND ： 请求针对的是不存在的记录。

406 Not Acceptable ：用户请求的的格式不正确。
500 INTERNAL SERVER ERROR ：服务器发生错误。

七、错误信息：一般来说，服务器返回的错误信息，以键值对的形式返回。

```
{
    error: 'Invalid API KEY'
}
```

八、响应结果：针对不同结果，服务器向客户端返回的结果应符合以下规范。

九、使用链接关联相关的资源：在返回响应结果时提供链接其他 API 的方法，使客户端很方便的获取相关联的信息。

十、其他：

服务器返回的数据格式，应该尽量使用 JSON，避免使用 XML。

47.什么 csrf 攻击原理？如何解决？

简单来说就是: 你访问了信任网站 A，然后 A 会用保存你的个人信息并返回给你的浏览器一个
cookie，然后呢，在 cookie 的过期时间之内，你去访问了恶意网站 B，它给你返回一些恶意请求代码，要求你去访问网站 A，而你的浏览器在收到这个恶意请求之后，在你不知情的情况下，会带上保存在本地浏览器的 cookie 信息去访问网站 A，然后网站 A 误以为是用户本身的操作，导致来自恶意网站 C 的攻击代码会被执：发邮件，发消息，修改你的密码，购物，转账，偷窥你的个人信息，导致私人信息泄漏和账户财产安全收到威胁

48.启动 Django 服务的方法？

runserver 方法是调试 Django 时经常用到的运行方式，它使用 Django 自带的 WSGI Server 运行，主要在测试和开发中使用，并且 runserver 开启的方式也是单进程 。

49.怎样测试 django 框架中的代码？

在单元测试方面，Django 继承 python 的 unittest.TestCase 实现了自己的 django.test.TestCase，编写测试用例通常从这里开始。测试代码通常位于 app 的 tests.py 文件中(也可以在 models.py 中编写，一般不建议)。在 Django 生成的 depotapp 中，已经包含了这个文件，并且其中包含了一个测试用例的样例：

1. python manage.py test：执行所有的测试用例
2. python manage.py test app_name， 执行该 app 的所有测试用例
3. python manage.py test app_name.case_name: 执行指定的测试用例

一些测试工具：unittest 或者 pytest

50.有过部署经验？用的什么技术？可以满足多少压力？

1.有部署经验，在阿里云服务器上部署的
2.技术有：nginx + uwsgi 的方式来部署 Django 项目
3.无标准答案（例：压力测试一两千）