**2020** 年 **python** 的面试题整理 **Web** 篇

**Flask**

**140.**对 **Flask** 蓝图**(Blueprint)**的理解？

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。

蓝图的应用场景：

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。

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） 在一个应

用中用不同的 URL 规则多次注册一个蓝图。

通过蓝图提供模板过滤器、静态文件、模板和其他功能。一个蓝图不一定要实

现应用或视图函数。

初始化一个 Flask扩展时，在这些情况中注册一个蓝图。

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。

使用蓝图的三个步骤

1.创建一个蓝图对象

blue = Blueprint("blue",\_\_name\_\_)

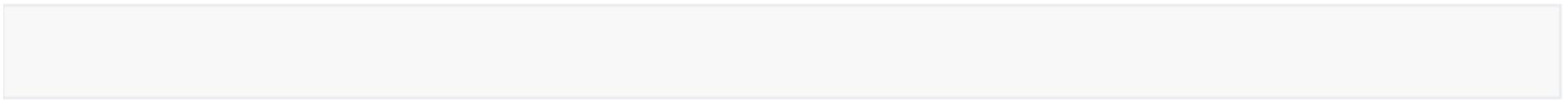
2.在这个蓝图对象上进行操作，例如注册路由、指定静态文件夹、注册模板过

滤器...

@blue.route('/')

return "Welcome to my blueprint"

3.在应用对象上注册这个蓝图对象



app.register\_blueprint(blue,url\_prefix="/blue")

**141.Flask** 和 **Django** 路由映射的区别？

在 django中，路由是浏览器访问服务器时，先访问的项目中的 url，再由项目

中的 url找到应用中 url，这些 url是放在一个列表里，遵从从前往后匹配的规

则。在 flask中，路由是通过装饰器给每个视图函数提供的，而且根据请求方式

的不同可以一个 url用于不同的作用。

**Django**

**142.**什么是 **wsgi,uwsgi,uWSGI?**

WSGI:

web 服务器网关接口，是一套协议。用于接收用户请求并将请求进行初次封

装，然后将请求交给 web 框架。

实现 wsgi协议的模块：wsgiref本, 质上就是编写一 socket服务端，用于接收用

户请求（django)

werkzeug,本质上就是编写一个 socket服务端，用于接收用户请求(flask)

uwsgi:

与 WSGI 一样是一种通信协议，它是 uWSGI 服务器的独占协议，用于定义传

输信息的类型。 uWSGI:

是一个 web 服务器，实现了 WSGI 的协议，uWSGI 协议，http协议

**143.Django**、**Flask**、**Tornado** 的对比？

1、 Django 走的大而全的方向，开发效率高。它的 MTV 框架，自带的

ORM,admin 后台管理,自带的 sqlite数据库和开发测试用的服务器，给开发者

提高了超高的开发效率。 重量级 web 框架，功能齐全，提供一站式解决的思

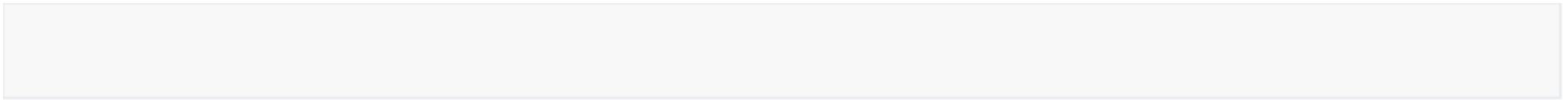
路，能让开发者不用在选择上花费大量时间。

自带 ORM 和模板引擎，支持 jinja等非官方模板引擎。

自带 ORM 使 Django 和关系型数据库耦合度高，如果要使用非关系型数据库，

需要使用第三方库

自带数据库管理 app



成熟，稳定，开发效率高，相对于 Flask，Django 的整体封闭性比较好，适合

做企业级网站的开发。python web 框架的先驱，第三方库丰富

2、 Flask 是轻量级的框架，自由，灵活，可扩展性强，核心基于 Werkzeug

WSGI 工具 和 jinja2模板引擎

适用于做小网站以及 web 服务的 API,开发大型网站无压力，但架构需要自己设

计

与关系型数据库的结合不弱于 Django，而与非关系型数据库的结合远远优于

Django

3、 Tornado 走的是少而精的方向，性能优越，它最出名的异步非阻塞的设计

方式

Tornado 的两大核心模块：

iostraem:对非阻塞的 socket进行简单的封装

ioloop:对 I/O 多路复用的封装,它实现一个单例

**144.CORS** 和 **CSRF** 的区别？

什么是 CORS ？

CORS 是一个 W3C 标准,全称是“跨域资源共享"(Cross-origin resoure sharing).

它允许浏览器向跨源服务器，发出 XMLHttpRequest 请求，从而客服了 AJAX

只能同源使用的限制。

什么是 CSRF ？

CSRF 主流防御方式是在后端生成表单的时候生成一串随机 token,内置到表单

里成为一个字段，同时，将此串 token置入 session中。每次表单提交到后端

时都会检查这两个值是否一致，以此来判断此次表单提交是否是可信的，提交

过一次之后，如果这个页面没有生成 CSRF token,那么 token将会被清空,如果

有新的需求，那么 token会被更新。 攻击者可以伪造 POST 表单提交，但是

他没有后端生成的内置于表单的 token，session中没有 token都无济于事。

**145.Session,Cookie,JWT** 的理解

为什么要使用会话管理

众所周知，HTTP 协议是一个无状态的协议，也就是说每个请求都是一个独立

的请求，请求与请求之间并无关系。但在实际的应用场景，这种方式并不能满

足我们的需求。举个大家都喜欢用的例子，把商品加入购物车，单独考虑这个

请求，服务端并不知道这个商品是谁的，应该加入谁的购物车？因此这个请求

的上下文环境实际上应该包含用户的相关信息，在每次用户发出请求时把这一

小部分额外信息，也做为请求的一部分，这样服务端就可以根据上下文中的信

息，针对具体的用户进行操作。所以这几种技术的出现都是对 HTTP 协议的一

个补充，使得我们可以用 HTTP 协议+状态管理构建一个的面向用户的 WEB 应

用。

Session 和 Cookie 的区别

这里我想先谈谈 session与 cookies,因为这两个技术是做为开发最为常见的。

那么 session与 cookies的区别是什么？个人认为 session与 cookies最核心区

别在于额外信息由谁来维护。利用 cookies来实现会话管理时，用户的相关信

息或者其他我们想要保持在每个请求中的信息，都是放在 cookies中,而

cookies是由客户端来保存，每当客户端发出新请求时，就会稍带上 cookies,服

务端会根据其中的信息进行操作。 当利用 session来进行会话管理时，客户端

实际上只存了一个由服务端发送的 session\_id,而由这个 session\_id,可以在服务

端还原出所需要的所有状态信息，从这里可以看出这部分信息是由服务端来维

护的。

除此以外，session与 cookies都有一些自己的缺点： cookies的安全性不

好，攻击者可以通过获取本地 cookies进行欺骗或者利用 cookies进行 CSRF

攻击。使用 cookies时,在多个域名下，会存在跨域问题。 session 在一定的时

间里，需要存放在服务端，因此当拥有大量用户时，也会大幅度降低服务端的

性能，当有多台机器时，如何共享 session也会是一个问题.(redis集群)也就是

说，用户第一个访问的时候是服务器 A，而第二个请求被转发给了服务器 B，

那服务器 B 如何得知其状态。实际上，session与 cookies是有联系的，比如

我们可以把 session\_id存放在 cookies中的。

JWT 是如何工作的

首先用户发出登录请求，服务端根据用户的登录请求进行匹配，如果匹配成

功，将相关的信息放入 payload中，利用算法，加上服务端的密钥生成

token，这里需要注意的是 secret\_key很重要，如果这个泄露的话，客户端就

可以随机篡改发送的额外信息，它是信息完整性的保证。生成 token后服务端

将其返回给客户端，客户端可以在下次请求时，将 token一起交给服务端，一

般是说我们可以将其放在 Authorization首部中，这样也就可以避免跨域问题。

**146.**简述 **Django** 请求生命周期

一般是用户通过浏览器向我们的服务器发起一个请求(request)这,个请求会去访

问视图函数，如果不涉及到数据调用，那么这个时候视图函数返回一个模板也

就是一个网页给用户） 视图函数调用模型毛模型去数据库查找数据，然后逐级

返回，视图函数把返回的数据填充到模板中空格中，最后返回网页给用户。

1.wsgi 请, 求封装后交给 web 框架（Flask，Django)

2.中间件，对请求进行校验或在请求对象中添加其他相关数据，例如：

csrf,request.session

3.路由匹配 根据浏览器发送的不同 url去匹配不同的视图函数

4.视图函数，在视图函数中进行业务逻辑的处理，可能涉及到：orm，

templates

5.中间件，对响应的数据进行处理

6.wsgi，将响应的内容发送给浏览器

**147.**用的 **restframework** 完成 **api** 发送时间时区

当前的问题是用 django的 rest framework模块做一个 get请求的发送时间以及

时区信息的 api

local\_time = time.localtime()

time\_zone =settings.TIME\_ZONE

temp = {'localtime':local\_time,'timezone':time\_zone}

return Response(temp)

**148.nginx,tomcat,apach** 到都是什么？

Nginx（engine x)是一个高性能的 HTTP 和反向代理服务器，也是 一个

IMAP/POP3/SMTP

服务器，工作在 OSI 七层，负载的实现方式：轮询，

IP\_HASH,fair,session\_sticky. Apache HTTP Serve是r 一个模块化的服务器，

源于 NCSAhttpd 服务器 Tomcat 服务器是一个免费的开放源代码的 Web 应用

服务器，属于轻量级应用服务器，是开发和调试 JSP 程序的首选。

**149.**请给出你熟悉关系数据库范式有哪些，有什么作

用？

在进行数据库的设计时，所遵循的一些规范，只要按照设计规范进行设计，就

能设计出没有数据冗余和数据维护异常的数据库结构。

数据库的设计的规范有很多，通常来说我们在设是数据库时只要达到其中一些

规范就可以了，这些规范又称之为数据库的三范式，一共有三条，也存在着其

他范式，我们只要做到满足前三个范式的要求，就能设陈出符合我们的数据库

了，我们也不能全部来按照范式的要求来做，还要考虑实际的业务使用情况，

所以有时候也需要做一些违反范式的要求。 1.数据库设计的第一范式(最基

本)，基本上所有数据库的范式都是符合第一范式的，符合第一范式的表具有以

下几个特点：



数据库表中的所有字段都只具有单一属性，单一属性的列是由基本的数据类型

（整型，浮点型，字符型等）所构成的设计出来的表都是简单的二比表

2.数据库设计的第二范式(是在第一范式的基础上设计的)，要求一个表中只具有

一个业务主键，也就是说符合第二范式的表中不能存在非主键列对只对部分主

键的依赖关系

3.数据库设计的第三范式，指每一个非主属性既不部分依赖与也不传递依赖于

业务主键，也就是第二范式的基础上消除了非主属性对主键的传递依赖

**150.**简述 **QQ** 登陆过程

qq 登录，在我们的项目中分为了三个接口，

第一个接口是请求 qq 服务器返回一个 qq 登录的界面;

第二个接口是通过扫码或账号登陆进行验证，qq 服务器返回给浏览器一个

code 和 state利, 用这个 code 通过本地服务器去向 qq 服务器获取

access\_token 覆返回给本地服务器，凭借 access\_token 再向 qq 服务器获取用

户的 openid(openid用户的唯一标识)

第三个接口是判断用户是否是第一次 qq 登录，如果不是的话直接登录返回的

jwt-token给用户，对没有绑定过本网站的用户，对 openid进行加密生成 token

进行绑定

**151.post** 和 **get** 的区别**?**

1.GET 是从服务器上获取数据，POST 是向服务器传送数据

2.在客户端，GET 方式在通过 URL 提交数据，数据在 URL 中可以看到，

POST 方式，数据放置在 HTML ——HEADER 内提交

3.对于 GET 方式，服务器端用 Request.QueryString获取变量的值，对于

POST 方式，服务器端用 Request.Form 获取提交的数据

**152.**项目中日志的作用

一、日志相关概念

1.日志是一种可以追踪某些软件运行时所发生事件的方法

2.软件开发人员可以向他们的代码中调用日志记录相关的方法来表明发生了某

些事情

3.一个事件可以用一个包含可选变量数据的消息来描述

4.此外，事件也有重要性的概念，这个重要性也可以被成为严重性级别(level)

二、日志的作用

1.通过 log的分析，可以方便用户了解系统或软件、应用的运行情况;

2.如果你的应用 log足够丰富，可以分析以往用户的操作行为、类型喜好，地

域分布或其他更多信息;

3.如果一个应用的 log同时也分了多个级别，那么可以很轻易地分析得到该应

用的健康状况，及时发现问题并快速定位、解决问题，补救损失。

4.简单来讲就是我们通过记录和分析日志可以了解一个系统或软件程序运行情

况是否正常，也可以在应用程序出现故障时快速定位问题。不仅在开发中，在

运维中日志也很重要，日志的作用也可以简单。总结为以下几点：

1.程序调试

2.了解软件程序运行情况，是否正常

3,软件程序运行故障分析与问题定位

4,如果应用的日志信息足够详细和丰富，还可以用来做用户行为分析

153.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

def \_\_init\_\_():

pass

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

def process\_request(request):

pass

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

def process\_view(request,view\_func,view\_args,view\_kwargs):

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。



def process\_template\_response(request,response):

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。

def process\_response(request,response):

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。

def process\_exception(request,exception):

pass

**154.**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。

nginx 是一个开源的高性能的 HTTP 服务器和反向代理：

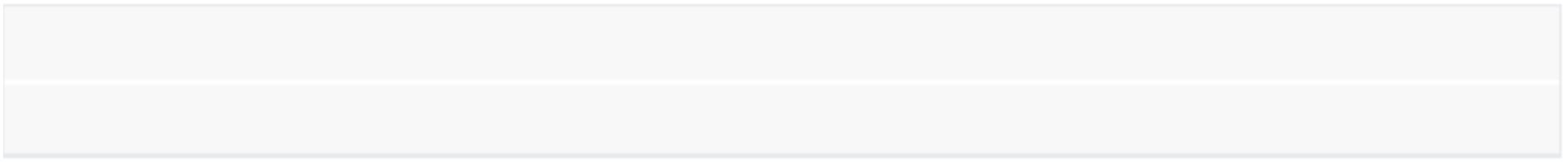
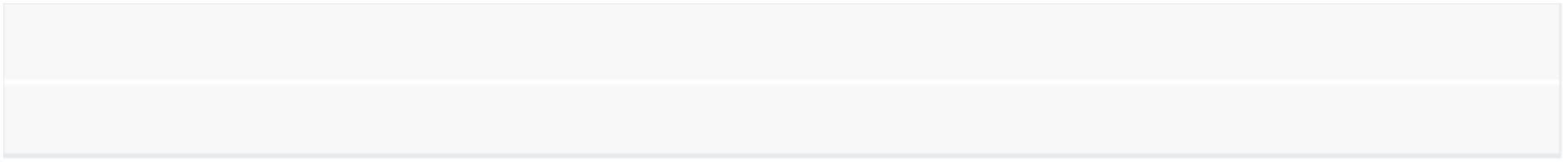
1.作为 web 服务器，它处理静态文件和索引文件效果非常高

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



**155.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，,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,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，提考开发效率，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 、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。

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

**156.Django** 中哪里用到了线程？哪里用到了协程？

哪里用到了进程？

1.Django中耗时的任务用一个进程或者线程来执行，比如发邮件，使用 celery.

2.部署 django项目是时候，配置文件中设置了进程和协程的相关配置。

**157.**有用过 **Django REST framework** 吗？

Django REST framework 是一个强大而灵活的 Web API 工具。使用

RESTframework 的理由有：

Web browsable API 对开发者有极大的好处

包括 OAuth1a 和 OAuth2 的认证策略

支持 ORM 和非 ORM 数据资源的序列化

全程自定义开发--如果不想使用更加强大的功能，可仅仅使用常规的 function-

based views额外的文档和强大的社区支持

**158.**对 **cookies** 与 **session** 的了解？他们能单独用

吗？

Session 采用的是在服务器端保持状态的方案，而 Cookie 采用的是在客户端保

持状态的方案。但是禁用 Cookie 就不能得到 Session。因为 Session 是用

Session ID来确定当前对话所对应的服务器 Session，而 Session ID是通过

Cookie 来传递的，禁用 Cookie 相当于 SessionID,也就得不到 Session。

爬虫

**159.**试列出至少三种目前流行的大型数据库

**160.**列举您使用过的 **Python** 网络爬虫所用到的网络

数据包**?**

requests, urllib,urllib2, httplib2

**161.**爬取数据后使用哪个数据库存储数据的，为什么？

**162.**你用过的爬虫框架或者模块有哪些？优缺点？

Python 自带：urllib,urllib2

第三方：requests

框架： Scrapy

urllib和 urllib模2 块都做与请求 URL 相关的操作，但他们提供不同的功能。

urllib2: urllib2.url可op以en接受一个 Request 对象或者 url,在(接受 Request 对

象时，并以此可以来设置一个 URL 的 headers),urllib.urlop只en接收一个 url。

urllib有 urlencode,urllib没2有，因此总是 urllib, urll常ib会2一起使用的原因

scrapy是封装起来的框架，他包含了下载器，解析器，日志及异常处理，基于

多线程，twisted的方式处理，对于固定单个网站的爬取开发，有优势，但是对

于多网站爬取 100 个网站，并发及分布式处理不够灵活，不便调整与扩展

requests是一个 HTTP 库，它只是用来请求，它是一个强大的库，下载，解析

全部自己处理，灵活性高



Scrapy 优点：异步，xpath，强大的统计和 log系统，支持不同 url。shell方便

独立调试。写 middleware 方便过滤。通过管道存入数据库

**163.**写爬虫是用多进程好？还是多线程好？

**164.**常见的反爬虫和应对方法？

**165.**解析网页的解析器使用最多的是哪几个**?**

**166.**需 要 登 录 的 网 页 ， 如 何 解 决 同 时 限 制 **ip** ，

**cookie,session**

**167.**验证码的解决**?**

**168.**使用最多的数据库，对他们的理解？

**169.**编写过哪些爬虫中间件？

**170.“**极验**”**滑动验证码如何破解？

**171.**爬虫多久爬一次，爬下来的数据是怎么存储？

**172.cookie** 过期的处理问题？

**173.**动态加载又对及时性要求很高怎么处理？

**174.HTTPS** 有什么优点和缺点？

**175.HTTPS** 是如何实现安全传输数据的？

**176.TTL**，**MSL**，**RTT** 各是什么？

**177.**谈一谈你对 **Selenium** 和 **PhantomJS** 了解

**178.**平常怎么使用代理的 ？

**179.**存放在数据库**(redis**、**mysql** 等**)**。

**180.**怎么监控爬虫的状态**?**

**181.**描述下 **scrapy** 框架运行的机制？

**182.**谈谈你对 **Scrapy** 的理解？

**183.**怎么样让 **scrapy** 框架发送一个 **post** 请求（具

体写出来）

**184.**怎么监控爬虫的状态 ？

**185.**怎么判断网站是否更新？

**186.**图片、视频爬取怎么绕过防盗连接

**187.**你爬出来的数据量大概有多大？大概多长时间爬

一次？

**188.**用什么数据库存爬下来的数据？部署是你做的吗？

怎么部署？

**189.**增量爬取

**190.**爬取下来的数据如何去重，说一下 **scrapy** 的具

体的算法依据。

191.Scrapy 的优缺点?

192.怎么设置爬取深度？

193.scrapy 和 scrapyredis 有什么区别？为什么选

择 redis 数据库？

194.分布式爬虫主要解决什么问题？

195.什么是分布式存储？

196.你所知道的分布式爬虫方案有哪些？

197.scrapyredis ，有做过其他的分布式爬虫吗？