**django的组成？为什么选择MVC设计风格**

1)使用mvc的思想来设计一个软件，最根本的原因是为了实现模型的复用:

            a,模型不用关心处理结果如何展现。

            比如，模型返回一些数据，然后交给不同的视图来展现(表格的方式、图形的方式等等)。

            b,可以使用不同的视图来访问同一个模型。

 2)代码的维护性更好

            修改模型不会影响到视图，反过来，修改视图，也不会影响到模型。

 3)方便测试

**三个web框架的区别**

Flask是一个微框架，主要面向需求简单的小应用。

Flask 自由、灵活，可扩展性强，第三方库的选择面广，

Django是面向大的应用，Django 的自带 ORM 非常优秀，

Django的目标是提供web应用开发的一站式解决方案，所以相应的模块也就比较多。自带 ORM 和模板引擎，灵活和自由度不够

**单独实现用户权限，需要设计几张表**

一个用户拥有若干角色，每一个角色拥有若干权限。这样，就构造成“用户-角色-权限“模式

用户，角色，权限，权限关联的开放区间

**web后端的核心逻辑写在哪，是否可拆**

**拆分views**新建view文件夹，在urls.py文件里引入import preheat.view.order\_view，其实就是引入你自己的view路径

**对http的了解**

1. 客户端与服务器建立TCP连接，一般使用80端口。

2. 客户端向服务器发送请求。

3. 服务器向客户端发送响应。

4. 重复上述步骤。

5. 通讯完毕，TCP连接断开。

**1.HTTP协议是无状态的**

就是说每次HTTP请求都是独立的，任何两个请求之间没有什么必然的联系。但是在实际应用当中并不是完全这样的，引入了Cookie和Session机制来关联请求。

**2.多次HTTP请求**

在客户端请求网页时多数情况下并不是一次请求就能成功的，服务端首先是响应HTML页面，然后浏览器收到响应之后发现HTML页面还引用了其他的资源，例如，CSS，JS文件，图片等等，还会自动发送HTTP请求这些需要的资源。现在的HTTP版本支持管道机制，可以同时请求和响应多个请求，大大提高了效率。

**3.基于TCP协议**

HTTP协议目的是规定客户端和服务端数据传输的格式和数据交互行为，并不负责数据传输的细节。底层是基于TCP实现的。现在使用的版本当中是默认持久连接的，也就是多次HTTP请求使用一个TCP连接。

django页面访问量

pip install django-tracking

首先控制器接收用户的请求，并决定应该调用哪个模型来进行处理，然后模型用业务逻辑来处理用户的请求并返回数据，最后控制器用相应的视图格式化模型的数据，并通过视图层展示给用户。

**跨域问题**

**mongodb和mysql的索引怎么建立**

mongodb的索引建立：

db.collection.ensureIndex({field:1/-1}) # 1是正序,-1是倒序

db.collection.ensureIndex({field1:1/-1, field2:1/-1})

**mysql的索引：**

create index index\_name on students (sname(5));

1.添加PRIMARY KEY（主键索引）

mysql>ALTER TABLE `table\_name` ADD PRIMARY KEY ( `column` )

2.添加UNIQUE(唯一索引)

mysql>ALTER TABLE `table\_name` ADD UNIQUE ( `column` )

3.添加INDEX(普通索引)

mysql>ALTER TABLE `table\_name` ADD INDEX index\_name ( `column` )

4.添加FULLTEXT(全文索引)

mysql>ALTER TABLE `table\_name` ADD FULLTEXT ( `column`)

5.添加多列索引

mysql>ALTER TABLE `table\_name` ADD INDEX index\_name ( `column1`, `column2`, `column3` )

varchar(n)

长度为 n 个字节的可变长度且非 Unicode 的字符数据。n 必须是一个介于 1 和 8,000 之间的数值。存储大小为输入数据的字节的实际长度，而不是 n 个字节。

nvarchar(n)

包含 n 个字符的可变长度 Unicode 字符数据。n 的值必须介于 1 与 4,000 之间。字节的存储大小是所输入字符个数的两倍。

varchar(4) 可以输入4个字母，也可以输入两个汉字

nvarchar(4) 可以输四个汉字，也可以输4个字母，但最多四个

两字段分别有字段值：我和coffee

那么varchar字段占2×2+6=10个字节的存储空间，而nvarchar字段占8×2=16个字节的存储空间。

**sql的left join 、right join 、inner join之间的区别**

left join(左联接) 返回包括左表中的所有记录和右表中联结字段相等的记录

right join(右联接) 返回包括右表中的所有记录和左表中联结字段相等的记录

inner join(等值连接) 只返回两个表中联结字段相等的行

**SQL UNION 语法：**

sql脚本代码如下:

1 SELECT column\_name FROM table1

2 UNION

3 SELECT column\_name FROM table2

注释：默认地，UNION 操作符选取不同的值。如果允许重复的值，请使用 UNION ALL。

当 ALL 随 UNION 一起使用时（即 UNION ALL），不消除重复行

SQL UNION ALL 语法

sql脚本代码如下:

1 SELECT column\_name FROM table1

2 UNION ALL

3 SELECT column\_name FROM table2

**mongodb 数据库去重**

**优化爬虫**

**多进程，多线程，分布式**

**元类的理解**

函数type实际上是一个元类，元类就是用来创建类的"模板"。我们可以通过类"模板"创建实例对象，同样，也可以使用元类"模板"来创建类对象；也就是说，元类就是类的类。

**1、进程与线程比较**

**2、协程多与线程进行比**较

**分布式**：不同的业务模块部署在不同的服务器上或者同一个业务模块分拆多个子业务，部署在不同的服务器上，解决高并发的问题

**集群**：同一个业务部署在多台机器上，提高系统可用性

**写出单例**

**为什么使用scrapy-redis而不是scrapy**

**网络安全知识**

**装饰器默认的使用规则**

**全局变量的使用**

**django的ORM底层**

**算法的时间复杂度和空间复杂度稳定性**

**怎么解决跨域问题**

**Scrap-redis中redis起到的作用**

**Linux如何同多进程id查找进程名**

**Cookie和session的作用**

**Web中Redis如何实现缓存**

**Django如何实现前后端分离**

**如何确定爬虫数据的完整性**

**如何用shell启动爬虫**

**用shell遍历目录找出.py的文件**

**a,b=b,a实现的原理**

**django如何设计用户只能单设备登录**

**如何检测代码的质量**

**Linux如何配置git无密码**