

# 后端开发简介 (一)

胡钰彬

电子科协软件部

2018. 7. 8

# 简介

- 服务器与网站
- 域名与DNS
- 反向代理 —— NGINX
- 信息传达 —— UWSGI
- 环境隔离
- 应用处理 —— Django
- 数据存储 —— Mysql



# 服务器与网站



# 服务器与网站

## ➤什么是服务器？

- Server (computing), a system that responds to requests across a computer network to provide, or help to provide, a network or data service. (wiki)
- 简单说，服务器就是一台常开的电脑，它保持联网状态，随时等待提供服务。我们可以通过其IP地址访问对应端口，完成一些操作，譬如文件的下载、数据的查询等。



# 服务器与网站

➤当我们访问一个网站时，发生了什么？

- 依照输入的网址，找到了一台对应的服务器。
- 点击转到的时候，我们其实是向这个服务器发送了一个请求（request），服务器会根据自己的配置，做出对于这个请求的响应（response）。
- 当我们在一个网站里点击某些按钮的时候，会转到xxxx.com/some\_url，这时服务器就会查找配置文件中有没有针对这个some\_url的响应方式，如果有，就做出响应，没有的话，就返回404 NotFound。
- 我们发送的request有两种形式，GET 和 POST。



# 域名与DNS



# 域名与DNS

➤ 域名是什么？ IP地址是什么？ 端口是什么？ url是什么？

- 域名即我们访问网站的时候输入的地址，如eesast.com，需要购买。子域名是一个域名的下属，如ts19.eesast.com。
- IP地址(Internet Protocol Address)是每一个联网设备在互联网上的地址标识。在互联网上的设备通过IP地址互相查找。
- 端口（指虚拟端口）是一个IP地址对外的不同接口，不同的端口可以响应不同的服务。（web:8000, ssh:22，可以配置）。
- URL（Uniform Resource Locator），统一资源定位符，一般会对应到服务器内部的某个路径。



# 域名与DNS

## ➤DNS服务器是什么？

- Domain Name System, 万维网上作为域名和IP地址相互映射的一个分布式数据库，能够使用户更方便的访问互联网，而不用去记住能够被机器直接读取的IP数串。
- 大家可以给自己的电脑指定对应DNS服务器，方法请自行Google。





# 域名与DNS

➤如何为自己的网站配置一个域名？

- 大致流程：注册域名（到各种域名网站）→ 域名解析（我还没跑过）。
- 可以咨询szy同学。



**反向代理 - - NGINX**



# 反向代理 - - NGINX

## ➤什么是反向代理？

- 代理：用户无法直接访问，需要一个代理服务器，代理服务器替他去拿到响应并返回。
- 反向代理：用户不知情的情况下，服务器自己去别的地方拿数据，譬如服务器上的其他目录。



# 反向代理 - - NGINX

## ➤NGINX可以起到什么作用？

- Nginx(engine x)是一个高性能的HTTP和反向代理服务器。
- 它可以监听服务器的某个端口并做出响应，以达到提供web服务的目的。
- 可以配置监听的端口号，针对何种请求做出何种响应。
- E.g.路径的转移，将请求通过socket发给服务器本身的某个进程，再做后续处理。



# 反向代理 - - NGINX

## ➤ 如何安装并配置NGINX?

- <https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04>
- <http://www.cnblogs.com/jhao/p/6071790.html>



# 反向代理 - - NGINX

## ➤如何启动NGINX?

- ubuntu安装完Nginx后，文件结构大致为：
  - 所有的配置文件都在 /etc/nginx下；
  - 启动程序文件在 /usr/sbin/nginx下；
  - 日志文件在 /var/log/nginx/下，分别是access.log和error.log；
  - 并且在 /etc/init.d下创建了启动脚本nginx。
- Nginx指令：
  - `sudo /etc/init.d/nginx start` # 启动
  - `sudo /etc/init.d/nginx stop` # 停止
  - `sudo /etc/init.d/nginx restart` # 重启

### Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*



**信息传达 - - uWSGI**



# 信息传达 - - uWSGI

## ➤ uWSGI能起什么作用？

- WSGI (Web Server Gateway Interface), 描述web server与web application通信的规范。
  - WSGI协议主要包括server和application两部分。
- uwsgi: 与WSGI一样是一种通信协议, 是uWSGI服务器的独占协议。
- uWSGI: 是一个web服务器, 实现了WSGI协议、uwsgi协议、http协议等。

<https://www.jianshu.com/p/679dee0a4193>





# 信息传达 - - UWSGI

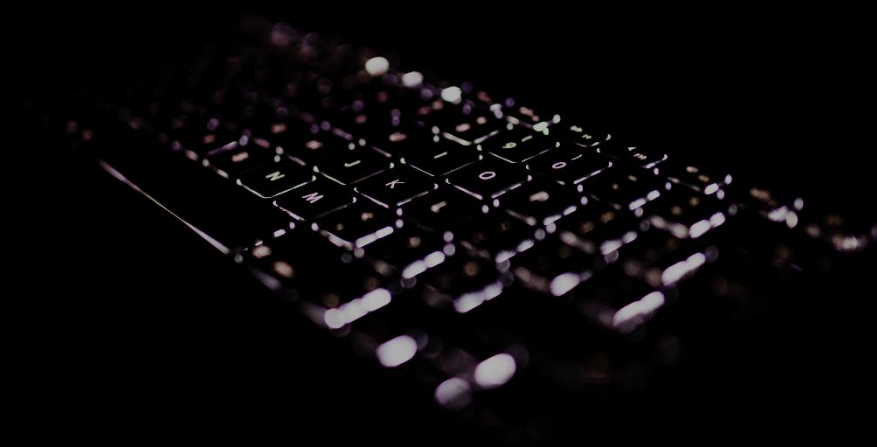
## ➤ 如何安装并配置uWSGI?

- pip install -i <https://pypi.tuna.tsinghua.edu.cn/simple> uwsgi

- ini文件

```
# Django-related settings
socket = :8001
# the base directory (full path)
chdir      = /home/ubuntu/blog
# Django s wsgi file
module     = blog.wsgi
# process-related settings
# master
master     = true
# maximum number of worker processes
processes  = 4
vacuum     = true
```

<http://www.cnblogs.com/jhao/p/6071790.html>



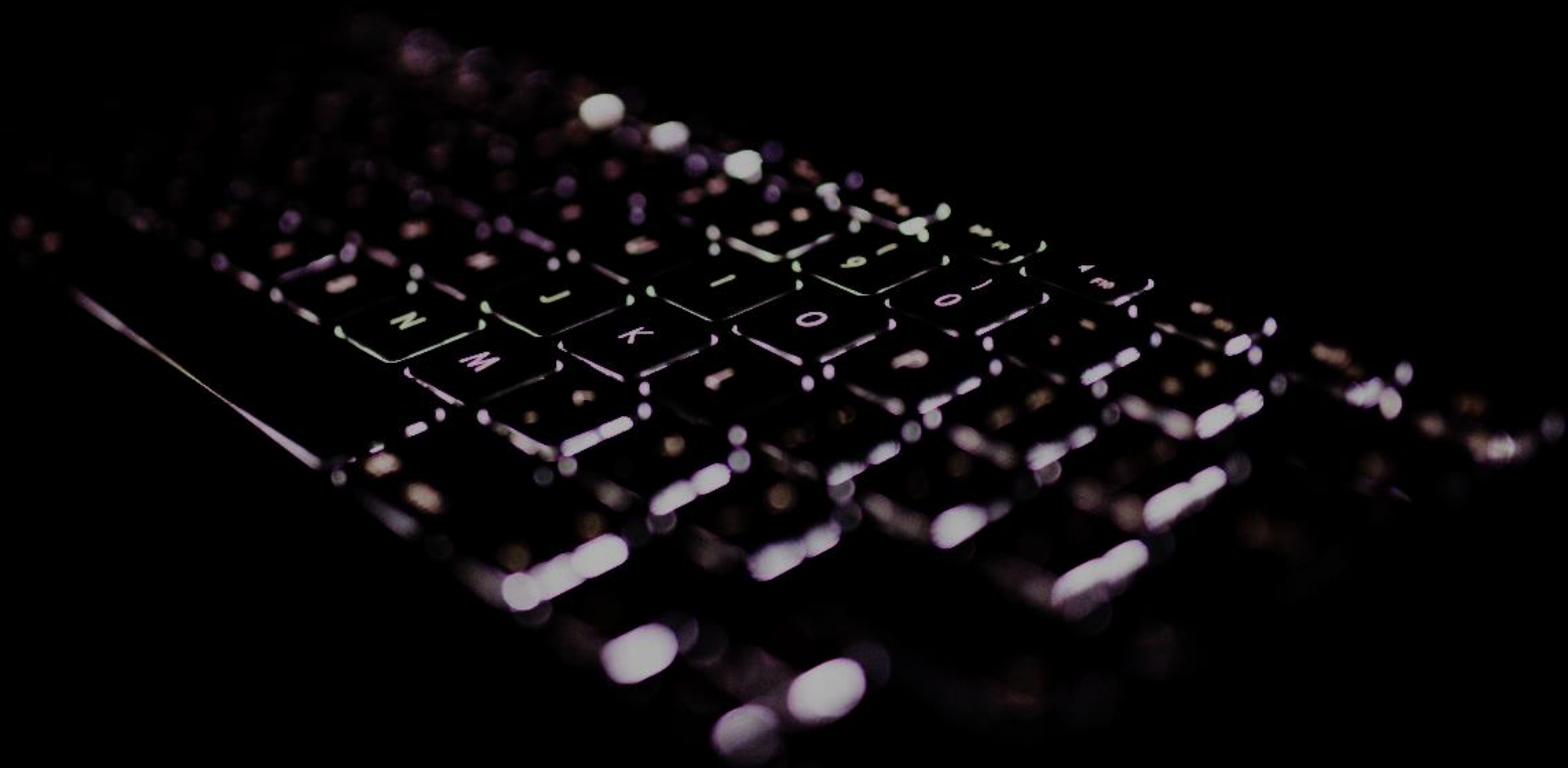
# 信息传达 - - UWSGI

➤ 如何启动uWSGI?

- `sudo uwsgi --ini uwsgi.ini`



# 环境隔离



# 环境隔离

## ➤什么是环境隔离?为什么要进行环境隔离?

- (非官方) 环境隔离就是为每一个应用设置其单独的运行环境, 使得各应用之间的依赖项不互相干扰。
- 往往在同一台设备上有很多应用, 他们可能用到某一框架/编译器的不同版本, 如不同的python版本。为了能够不加修改地进行应用迁移, 就有必要为每个应用建立一个环境, 做好对应的配置。
- 可以理解为作用域, 装好的python对应全局作用域, 各自虚拟环境对应子作用域。



# 环境隔离

➤如何创建一个虚拟环境？

- `pip3 install virtualenv` 或 `pip install virtualenv` 或 `apt install python3-virtualenv`
- `virtualenv testenv`    #运行virtualenv建立一个名为testenv的环境



# 环境隔离

➤如何进入这个虚拟环境？怎么退出？

- 在建立虚拟环境的目录下可以看到文件夹testenv，可以cd进去看一看目录内容
- 开启虚拟环境：`source testenv/bin/activate`，在开头可以看到一个括号表明当前状态
- 关闭虚拟环境：`deactivate`



# 应用处理 - - Django



# 应用处理 - Django

## ➤ Django起到的作用是什么？

- 可以更为细致地处理来自客户端的请求，针对不同url做出不同相应，可以用python编写函数处理来自客户端的数据。

```
urlpatterns = [  
    path('teams/add/', views.TeamAdd),  
    path('teams/join/', views.TeamJoin),  
    path('teams/exit/', views.TeamExit),  
    path('teams/oneteam/', views.MyTeam),  
    path('teams/allteam/', views.AllTeam),  
]
```

```
@csrf_exempt  
def TeamAdd(request):  
    if request.method == 'POST':  
        form = TeamAddForm(request.POST)  
        if form.is_valid():#创建队伍  
            teams = TeamInfo.objects.all()  
            success = True  
            message = ""  
            response = {}  
            the_leader = form.cleaned_data['userid']  
            invite_code = form.cleaned_data['invitecode']
```

<https://www.djangoproject.com/>



# 应用处理 - - Django

➤ 我们为什么选用Django? 和Django并肩的还有哪些框架?

- **RIDICULOUSLY FAST.** Django was designed to help developers take applications from concept to completion as quickly as possible.
- **FULLY LOADED.** Django includes dozens of extras you can use to handle common Web development tasks.
- **REASSURINGLY SECURE.** Django takes security seriously and helps developers avoid many common security mistakes.
- **EXCEEDINGLY SCALABLE.** Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.
- **INCREDIBLY VERSATILE.** Companies, organizations and governments have used Django to build all sorts of things

Flask

Tornado

Ruby On Rails

Nodejs

# 应用处理 - - Django

## ➤ 如何安装Django并新建一个项目？

- `pip3 install django`
- `pip3 show django`
- `python` 然后 `import django`
- Attention: 建议启动virtualenv之后再安装django，这样相当于只在虚拟幻境里安装了django
- 然后 `django-admin startproject mysite` , `python manage.py startapp polls`
- <https://docs.djangoproject.com/en/2.0/intro/tutorial01/>



# 应用处理 - Django

➤ 这么多文件都是干什么的？

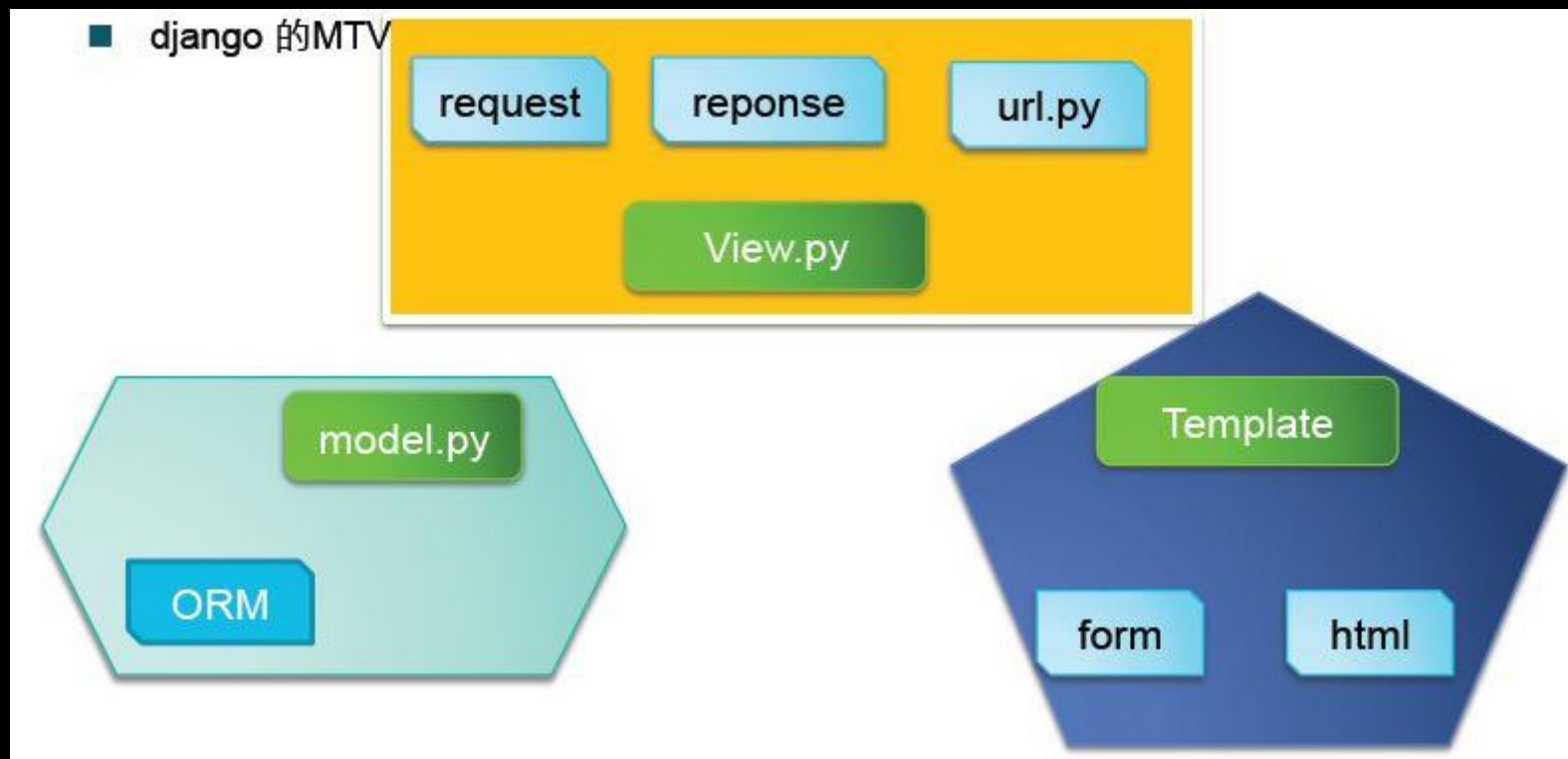
```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py
```

```
polls/  
  __init__.py  
  admin.py  
  apps.py  
  migrations/  
    __init__.py  
  models.py  
  tests.py  
  views.py
```



# 应用处理 - - Django

## ➤ Django的MTV模式



# 应用处理 - - Django

➤如何写Django的urls.py?

polls/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

```
path('teams/battletime/', views.GetBattleTime),
path('teams/inquire/<str:battleid>', views.Inquire),
path('team_id=<int:pk>', views.GroupDetail.as_view(), name='GroupDetail'),
path('user_id=<int:pk>', views.StudentDetail.as_view(), name='StudentDetail'),
path('RuleFile_id=<int:pk>', views.RuleFileView.as_view(), name='RuleFile'),
```

# 应用处理 - - Django

## ➤如何写Django的views.py?

polls/views.py

```
from django.http import HttpResponseRedirect

def index(request):
    return HttpResponseRedirect("Hello, world. You're at the polls index.")
```

```
@csrf_exempt
def Inquire(request, battleid):
    if request.method == 'POST':
        return JsonResponse({'success': False})
    elif request.method == 'GET':
        #team1 = TeamInfo.objects.get(id = id1)
        #team2 = TeamInfo.objects.get(id = id2)
```



# 应用处理 - - Django

## ➤ 如何写Django的views.py?

```
@csrf_exempt
def StudentLeader(request):
    success = False
    message = ""
    if request.method == 'POST':
        the_id = request.POST['userid']
        the_student = StudentInfo.objects.get(id = the_id)
        isleader = the_student.is_leader
        the_name = the_student.student_nickname
        success = True
        return JsonResponse({'success':success, 'message':message, 'name':the_name, 'isleader':isleader})
    else :
        return JsonResponse({'success':str(request.body), 'POST':str(request.POST), 'GET':str(request.GET)})
```

# 应用处理 - Django

## ➤如何写Django的表单与model?

polls/models.py

```
from django.db import models

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
```

```
>>> from polls.models import Choice, Question

# No questions are in the system yet.
>>> Question.objects.all()
```

mysite/settings.py

```
INSTALLED_APPS = [
    'polls.apps.PollsConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```



# 应用处理 - - Django

## ➤ Django的template?

- 我们放弃了django的template及其渲染，采用前后端分离的方式，后端只需要返回数据，渲染全部交给前端进行。

```
def index(request):  
    latest_question_list = Question.objects.order_by('-pub_date')[:5]  
    template = loader.get_template('polls/index.html')  
    context = {  
        'latest_question_list': latest_question_list,  
    }  
    return HttpResponse(template.render(context, request))
```



# 应用处理 - - Django

## ➤如何运行我的Django App?

- `python manage.py makemigrations`
- `python manage.py migrate`
- `python manage.py runserver`

```
(testenv) huyb@DESKTOP-E3E01CI: /trainexample/mysite$ python manage.py runserver
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
July 05, 2018 - 16:19:30
```


```
Django version 2.0.7, using settings 'mysite.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CONTROL-C.
```



# 数据存储 - - Mysql



# 数据存储 - - Mysql

➤ 为什么需要数据库？

- 原因很简单，我就不BB了。



# 数据存储 - - Mysql

## ➤如何安装并设置Mysql?

- `sudo apt update`
- `sudo apt install mysql-server`
- 然后进入到mysql安装目录`cd /etc/init.d`
- `sudo service mysql stop`
- `sudo service mysql start`



# 数据存储 - - Mysql

## ➤如何安装并设置Mysql?

```
huyb@DESKTOP-E3E01CI:/etc/init.d$ sudo mysql_secure_installation
```

```
Securing the MySQL server deployment.
```

```
Connecting to MySQL using a blank password.
```

```
VALIDATE PASSWORD PLUGIN can be used to test passwords  
and improve security. It checks the strength of password  
and allows the users to set only those passwords which are  
secure enough. Would you like to setup VALIDATE PASSWORD plugin?
```

```
Press y|Y for Yes, any other key for No: y
```

```
There are three levels of password validation policy:
```

```
LOW    Length >= 8
```

```
MEDIUM Length >= 8, numeric, mixed case, and special characters
```

```
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
```

```
Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 0
```

```
Please set the password for root here.
```

执行完上一页的操作之后，打开链接执行2、3两步。

2中后面的配置自行阅读说明进行选择即可，不懂来问我

file

<https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-18-04>

# 数据存储 - - Mysql

## ➤如何安装并设置Mysql?

- 进行到这一步时，注意这里已经把密码设为了password，之后用mysql -u root -p 进入mysql时，要用password做密码。
- 如果密码设置出现错误，一般是密码太简单，如果就想用简单密码，请：
- set global validate\_password\_policy=0;

4 rows in set (0.00 sec)

In this example, you can see that the root user does in fact authenticate using the `auth_socket` plugin. To configure the root account to authenticate with a password, run the following `ALTER USER` command. Be sure to change `password` to a strong password of your choosing:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
```

# 数据存储 - - Mysql

➤如何在mysql中进行基本操作?

- 通过mysql -u root -p进入mysql之后
- show databases; 查看所有数据库
- create database mytest; 创建一个数据库
- use mytest; 转入改数据库
- show tables; 查看改数据库中的表



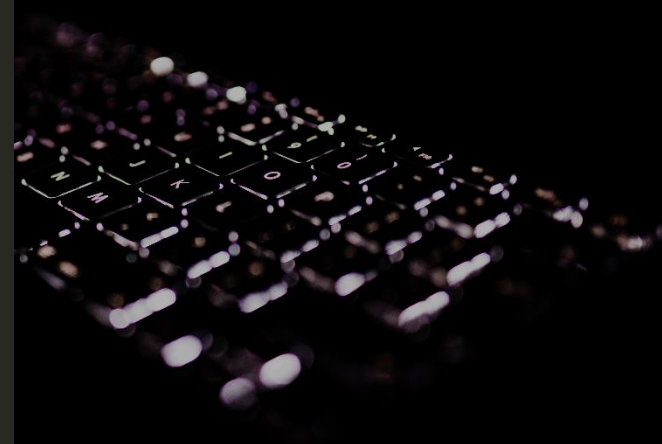


# 数据存储 - - Mysql

➤如何让Django使用mysql?

- 进入mysite/mysite/ 目录，在settings.py中做出如下修改：

```
DATABASES = {  
    'default': {  
#         'ENGINE': 'django.db.backends.sqlite3',  
#         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'mytest',  
        'USER': 'root',  
        'PASSWORD': 'password',  
        'HOST': 'localhost',  
    }  
}
```



# 数据存储 - - Mysql

➤如何在Django中操作数据库？ - 查、改、增、删

- Django会根据现有的model，在对应数据库中建立数据表
- 在Django中操作model就是在操作数据库
- `q = Question.objects.filter(id=1)`
- `q = Question.objects.get(id=2)`
- `q.title = 'qustionA'`
- `q.save()`
- `user = User.objects.create(email='a@b.com')`
- `user.delete()`



# 数据存储 - - Mysql

➤ Mysql 原生语法? (即Mysql黑框里写的语法)

- `SELECT * from mytest_table;`
- `UPDATE mytest_table SET name = 'rls' where id=2;`
- .....
- 记得加分号



总结



# 总结

前端



用户信息注册、查看

发送轮询查看对战结果

UWSGI + NGINX + django

接收、处理HTTP请求，分配url路径，匹配静态文件

存储用户信息

进行在线对战

后端

