## Django学习笔记

虫师：https://github.com/defnngj/

虫师 & 乙醇

课程内容

1.django开发

2.接口开发

接口自动化项目：unittest+requests+HtmlTestRunner+数据初始化+加密

接口安全机制：

认证（接口传一个比较特殊的字段，通过这个字段来验证身份）、

签名（接口的一些字段按照一定规则拼接后作为一个字段来传）、

加密（直接对传参的字符串进行加密）

Web server -->REST 接口设计风格 -->django-REST-Framework

实例：项目，分析接口 只要是协议接口都可以用python来调用

接口分两类：代码接口（代码和代码之间的调用，无法跨语言调用）和协议接口（http、https、soap、tcp/ip）

----------------------

python多个版本问题：

1.python多个版本切换，需要把python下的python.exe改为python3.exe，这样添加环境变量后，在cmd下就可以直接使用python3进入了

2.指定某个版本的pip来安装.whl的包，python3 -m pip install django.whl

3.tar.gz源码安装，进入解压后的文件夹内,python3 setup.py install

pip常用命令

show 看某包安装的信息

python3 -m pip show django

install 包名

不知道包的版本的时候有一个技巧就是包名==1232，随便输一个版本号，执行后就会列出来安装包所有的版本

python3 -m pip install django==12333，安装好的话scripts下有一个django-admin.exe，如果没有的话就是没装好

卸载：python3 -m pip uninstall django==12333

\*\*\*03\*\*\*

项目 --可以看成是网站，外层的guest

项目的子模块 --可以看成是具体的应用，比如内层的guest、sign

manage.py：一个命令行工具，可以让你在使用 Django 项目时以不同的方式进行交

----guest模块

guest/settings.py：Django的web项目的主配置文件，包括 Django 模块应用配置，数据库配置，模板配置等。

guest/urls.py：Django 项目的 URL ，路由文件，路径的配置

guest/wsgi.py：为 WSGI 兼容的 Web 服务器服务项目的切入，部署django的时候用到

----sign模块

admin.py：Django自带的 admin 后台，把models里面数据库的表映射到后台管理。

apps.py：在新（1.9）的 Django 版本中新增，用于应用程序的配置。

models.py：模型，操作数据库的文件，（创建和管理数据库表的文件，django自带的ORM，创建应用程序数据表模型（对应数据库的相关操作））。

views.py：视图，写业务功能的处理逻辑，控制向前端显示哪些数据

比如登录功能：views.py获取用户输入的用户名和密码，校验用户名和密码会调用models.py去数据库查用户和密码数据来进行判断逻辑的处理

tests.py：创建 Django 测试。

migrations/：用于记录 models 中数据的变更。

启动项目

python manage.py runserver django自带容器，运行项目的时候启动一个web容器

------

django 工作流：

0.通过表单或链接发送请求，比如

<html>

<a href="/index/">Index</a>

<h1>user login</h1>

<form>

username:<input type="text" /><br>

password:<input type="password" /><br>

<button>登录</button>

</form>

</html>

1.指定路径

http://127.0.0.1:8000/index/

2.通过setting的配置ROOT\_URLCONF = 'guest.urls'，找到urls路由文件

3.在urls.py通过path('index/', views.index),找到views

4.index 函数返回页面对象

--

1.首先页面有一个链接（不论是表单还是链接来发送请求）

2.链接指定到一个新的路径

http://127.0.0.1:8000/index/

3.通过setting的配置文件去找url的配置路径在哪里，然后找到urls文件在guest项目的urls文件里

4.在urls文件里拿index路径来匹配2中链接的路径，匹配上就会调用views的index函数

path('index/', views.index),

5.index函数拿到请求后，就会通过render方法返回index.html页面，页面默认放在templates目录里。

返回页面就是返回页面的数据，浏览器根据返回的数据把页面渲染出来

--

--------------------------------------------------

\*\*\*04 登录功能1\*\*\*

index.html

<!--登录页面-->

<html>

<a href="/index/">Index</a>

<h1>user login</h1>

<form action="/login\_action/" method="post">

username : <input type="text" name="username" placeholder="请输入用户名" /><br>

password : <input type="password" name="password" placeholder="请输入密码" /><br>

<font color="red">{{ error }}</font><br>

<button type="submit">登录</button>

{% csrf\_token %} <!--输入账号和空密码使用post方法提交会报错，CSRF...，加上这句就不会了-->

</form>

</html>

----

views.py

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

# 首页

def index(request):

return render(request, "index.html")

# 登录处理

def login\_action(request):

if request.method == "POST":

print("-----------------------")

# 接收用户输入的用户名、密码（方法要与页面提交的方法一致）

login\_username = request.POST.get("username")

login\_password = request.POST.get("password")

print("login\_username: "+login\_username)

print("login\_password: "+login\_password)

if login\_username == "" or login\_password == "":

#"username or password null"

return render(request, "index.html", {"error":"username or password null."}) #key error相当于发送给客户端的变量，变量对应的值就是后面的value

elif login\_username != "admin" or login\_password != "123":

#"username or password error"

return render(request, "index.html", {"error":"username or password error."}) #key error相当于发送给客户端的变量，变量对应的值就是后面的value

else:

return render(request, "event\_manage.html")

else:

return render(request, "index.html")

def logout\_action(request):

return HttpResponse("logout...")

--------------------------------------------------

### \*\*\*05 登录功能2\*\*\*

cookie 存折： 存钱、取钱记录 存折理解为客户端，对应的是浏览器

session 银行卡：卡号（12121一串数字，对用户而言没有用，对应浏览器session的sessionid） -->银行卡的敏感信息都是存在银行（服务端）里

用户 --user/pwd--->网站：登录 --通过user/pwd来确认用户是合法的

用户 网站返回给客户端success和sessid，浏览器把sessid作为cookie存在浏览器，这种cookie是存在浏览器进程里

用户登录后，发送所有的请求都要带上sessid，sessid保存在网站的数据库里，用户发请求的时候需要带上sessid一块发送给网站，

网站拿req带来的sessid和网站自己数据库的sessid作对比是不是合法用户，如果是我网站给的sessid（即在数据库找到了），就返回给你要的操作；

如果请求带来的sessid在数据库中不存在，网站返回一个新的页面。

sessid expire=30m，超过30m，会自动在浏览器里删除掉sessionid（一种会话cookie），另一种是存在硬盘里的cookie

---

其实Cookie有两种:

一种是存在于浏览器的进程中;

一种是存在于硬盘上

而session的Cookie是存在于浏览器的进程中，那么这种Cookie我们称为会话Cookie，当我们重新打开浏览器窗口时，之前的Cookie中存放的Sessionid已经不存在了，此时

服务器从HttpServletRequest对象中没有检查到sessionid，服务器会再发送一个新的存有Sessionid的Cookie到客户端的浏览器中，此时对应的是一个新的会话，而服务器上

原先的session等到它的默认时间到之后，便会自动销毁。

---

执行数据库同步

第一步

λ python manage.py migrate

cookie版views实现代码

from django.shortcuts import render

from django.http import HttpResponse,HttpResponseRedirect

# Create your views here.

# 首页

def index(request):

return render(request, "index.html")

# 登录处理

def login\_action(request):

if request.method == "POST":

print("-----------------------")

# 接收用户输入的用户名、密码（方法要与页面提交的方法一致）

login\_username = request.POST.get("username")

login\_password = request.POST.get("password")

print("login\_username: "+login\_username)

print("login\_password: "+login\_password)

if login\_username == "" or login\_password == "":

# key error相当于发送给客户端的变量，变量对应的值就是后面的value

return render(request, "index.html", {"error":"username or password null."})

elif login\_username == "admin" and login\_password == "123":

response = HttpResponseRedirect('/event\_manage/')

# 添加浏览器 cookie，10分钟过期

response.set\_cookie('user',login\_username,600)

return response

else:

return render(request, "index.html", {"error":"Incorrect username or password."})

else:

return render(request, "index.html")

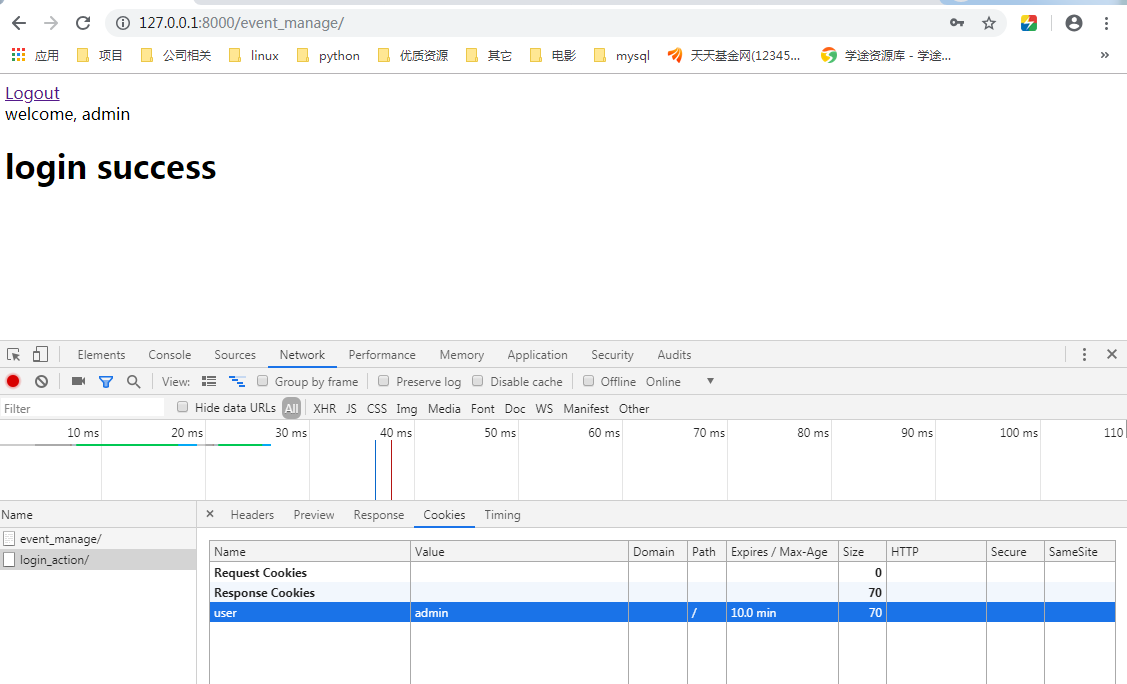
# 发布会管理

def event\_manage(request):

# 读取浏览器 cookie

username = request.COOKIES.get('user', '')

return render(request, "event\_manage.html", {"user":username})



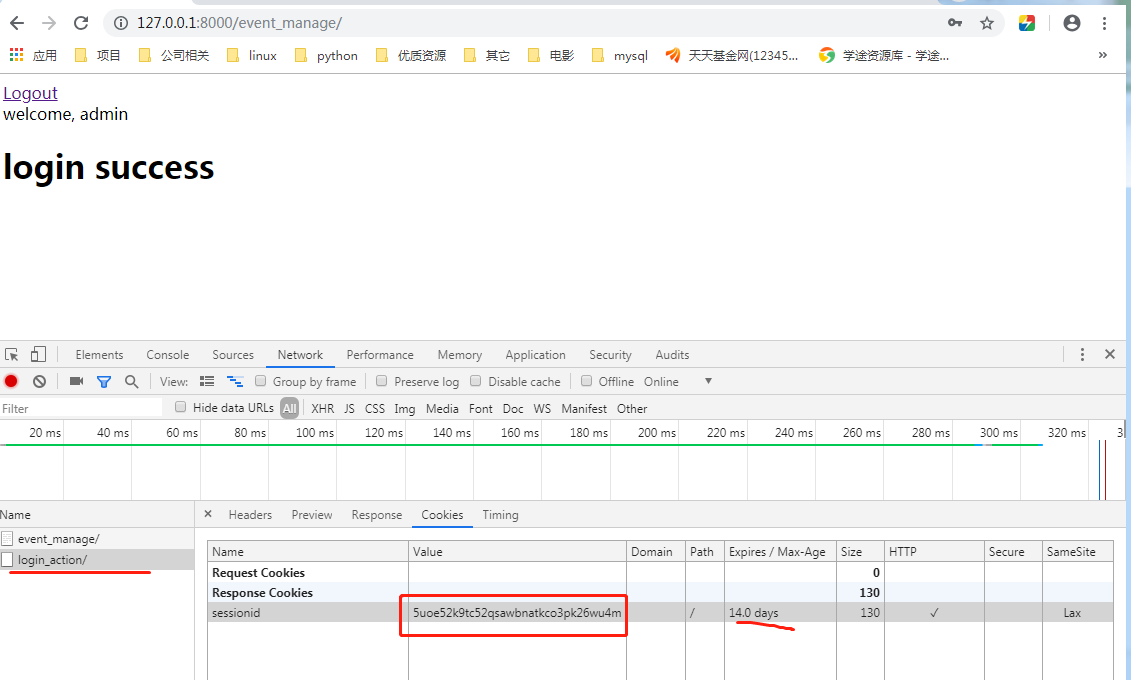
**这种cookie是存在进程里的，一般浏览器刷新页面、或者关闭浏览器就没了**

--------------------------------------------

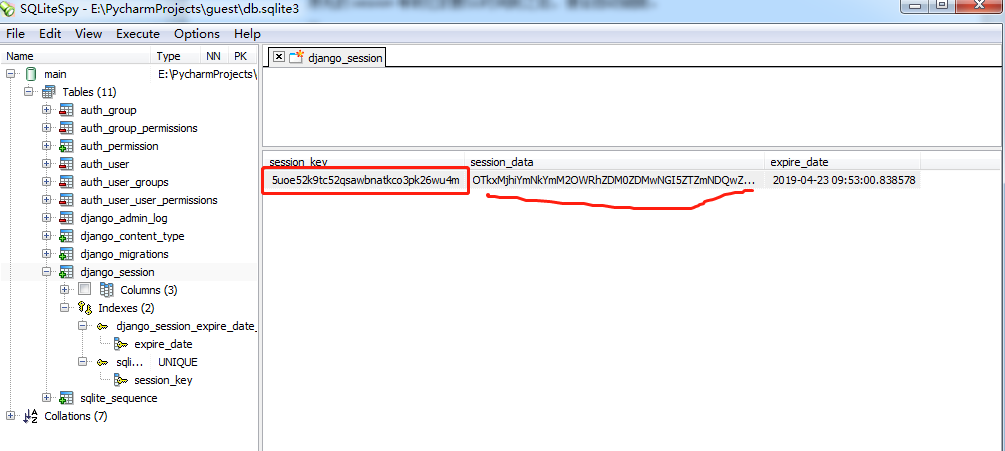
### \*\*\*06 登录功能3\*\*\*

**另一种会话cookie 是存在磁盘的一个文件**

浏览器的sessionid的value是服务端数据库session表里的session\_key,而session表里的session\_data（真正的用户信息）则是django处理后的用户名admin



浏览器的cookie可以清除，但是服务器数据库的会一直在，直到过期销毁



Session完成了，下面开始搞后台admin

首先创建admin用户

E:\PycharmProjects\guest

λ python manage.py createsuperuser

Username (leave blank to use 'administrator'): admin

Email address: admin@mail.com

Password:admin1234

Password (again):

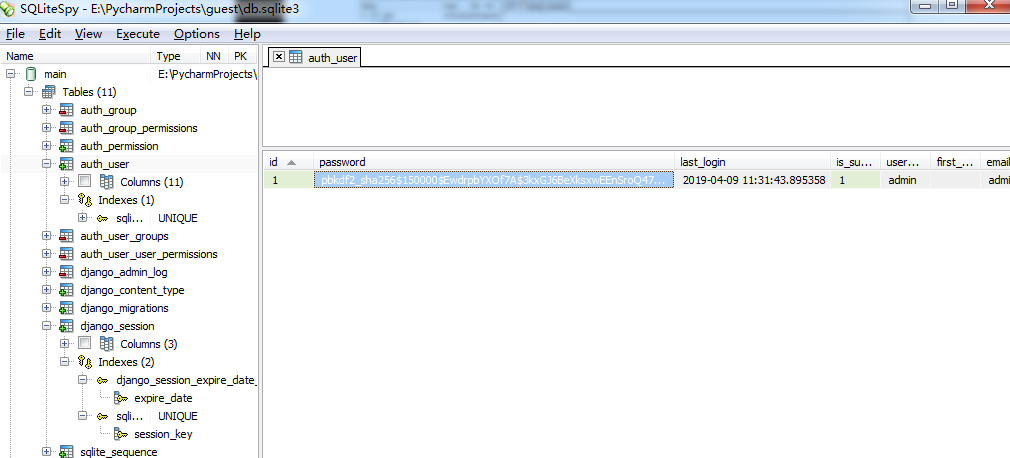
The password is too similar to the username.

This password is too common.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

查看数据库



登录功能完整代码在磁盘上有备份

---------------------------------------------

### \*\*\*07 Django 模型 1\*\*\*

**MVT**

m：models.py 模型，数据库操作

v：views.py 视图，业务处理的逻辑

承上启下，从页面接收请求传来的数据，然后调用模型去操作数据库，最后把结果响应给浏览器（浏览器进行渲染）

t：templates 模板，前端页面

### 增删改查

插入数据

from sign.models import Event, Guest

from datetime import datetime

1.两步走

e1 = Event(id=3,name='nubia X Pro 发布会',limit=2000,status=True,address='厦门',start\_time=datetime(2019,8,10,14,0,0))

e1.save()

2.一步搞定

Event.objects.create(id=2,name='nubia 红魔2 发布会',limit=4000,status=False,address='福州',start\_time=datetime(2019,1,10,14,0,0))

Guest.objects.create(realname='张三',phone=12345001101,email='zhangs@mail.com',sign=False,event\_id=2)

Guest.objects.create(realname='李四',phone=13900000000,email='lis@mail.com',sign=True,event\_id=4)

Guest.objects.create(realname='康平汆',phone=13600805241,email='kpc@mail.com',sign=True,event\_id=5)

Guest.objects.create(realname='andy',phone=13611001101,email='andy@mail.com',sign=False,event\_id=3)

需要说明的是，表的 id 字段已经设置了自增，所以，该字段为空可以添加数据，但在创建嘉宾时数据时

需要指定关联的发布会 id。Event 表指定了 id=3，Guest 表指定 event\_id=3，所以嘉宾 andy 对应的是红米 MAX发布会

查询数据

table.objects.get()方法用于从数据库表中取得一条匹配的结果，返回一个对象，如果记录不存在的话，那

么它会报 DoesNotExist 类型错误

--get完全匹配

>>> e3=Event.objects.get(id=3)

>>> e3.name

'nubia X Pro 发布会'

>>> Event.objects.get(id=3).address

'厦门'

>>> Guest.objects.get(phone='13600805241')

<Guest: 康平汆>

>>> Guest.objects.get(phone='13600805241').event

<Event: 红米 MAX 发布会>

>>> g.realname

'康平汆'

>>> Guest.objects.get(phone='13600805241',sign=1) 多个条件

<Guest: 康平汆>

table.objects.filter()方法是从数据库的取得匹配的结果，返回一个对象列表，如果记录不存在的话，它会

返回[]。

>>> Event.objects.filter(id=1)

<QuerySet [<Event: 努比亚X发布会>]>

>>> Event.objects.filter(name='nubia X Pro 发布会').address

Traceback (most recent call last):

File "<console>", line 1, in <module>

AttributeError: 'QuerySet' object has no attribute 'address'

--filter模糊匹配

在 name 和 contains 之间用双下划线。这里，contains 部分会被 Django 翻译成 LIKE 语句。

>>> e=Event.objects.filter(name\_\_contains='nubia')

>>> e

<QuerySet [<Event: nubia X Pro 发布会>, <Event: nubia 红魔 发布会>]>

删除数据

查询 phone='13611001101' 的嘉宾，通过 delete()方法删除

>>> g1=Guest.objects.get(email='lis@mail.com')

>>> g1.delete()

(1, {'sign.Guest': 1})

>>> g1

<Guest: 李四>

>>> g1=Guest.objects.get(email='lis@mail.com')

Traceback (most recent call last):

File "<console>", line 1, in <module>

File "D:\Python37\lib\site-packages\django\db\models\manager.py", line 82, in manager\_method

return getattr(self.get\_queryset(), name)(\*args, \*\*kwargs)

File "D:\Python37\lib\site-packages\django\db\models\query.py", line 408, in get

self.model.\_meta.object\_name

sign.models.Guest.DoesNotExist: Guest matching query does not exist.

更新数据

查询 phone='13611001101' 的嘉宾，更新 realname='andy2'

法1

>>> g2=Guest.objects.get(phone='13611001101')

>>> g2

<Guest: andy>

>>> g2.realname='andy2'

>>> g2

<Guest: andy2>

法2

>>> g2.save() 没有保存的话，不会刷新到磁盘（不会落库）

Guest.objects.select\_for\_update().filter(phone='13611001101').update(realname='andy')

查询条件可以为多个

Guest.objects.select\_for\_update().filter(phone='13611001101',sign=0).update(realname='andy2')