《Django Web框架教学笔记》

- 讲师: 魏明择
- 时间: 2019

目录

Django中的用户认证 (使用Django认证系统)

- Django带有一个用户认证系统。 它处理用户账号、组、权限以及基于cookie的用户会话。
- 作用:
  1. 添加普通用户和超级用户
  2. 修改密码
- 文档参见
  - https://docs.djangoproject.com/en/1.11/topics/auth/
- User模型类
  - 位置: `from django.contrib.auth.models import User`
- 默认user的基本属性有:

属性名	类型	是否必选
username	用户名	是
password	密码	是
email	邮箱	可选
first_name	名	
last_name	姓	
is_superuser	是否是管理员帐号(/admin)	
is_staff	是否可以访问admin管理界面	
is_active	是否是活跃用户,默认True。一般不删除用户，而是将用户的is_active设为False。	
last_login	上一次的登录时间	
date_joined	用户创建的时间	

- 数据库表现形式

```
mysql> use myauth;
mysql> desc auth_user;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | int(11)      | NO   | PRI | NULL    | auto_increment |
| password     | varchar(128) | NO   |     | NULL    |                |
| last_login   | datetime(6)  | YES  |     | NULL    |                |
| is_superuser | tinyint(1)   | NO   |     | NULL    |                |
| username     | varchar(150) | NO   | UNI | NULL    |                |
| first_name   | varchar(30)  | NO   |     | NULL    |                |
| last_name    | varchar(30)  | NO   |     | NULL    |                |
| email        | varchar(254) | NO   |     | NULL    |                |
| is_staff     | tinyint(1)   | NO   |     | NULL    |                |
```

```
| is_active    | tinyint(1)   | NO   |     | NULL    |                |
| date_joined  | datetime(6)  | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

auth基本模型操作:

- 创建用户
  - 创建普通用户create_user

```
from django.contrib.auth import models
user = models.User.objects.create_user(username='用户名', password='密码', email='邮箱',...)
...
user.save()
```

  - 创建超级用户create_superuser

```
from django.contrib.auth import models
user = models.User.objects.create_superuser(username='用户名', password='密码', email='邮箱',...)
...
user.save()
```

- 删除用户

```
from django.contrib.auth import models
try:
    user = models.User.objects.get(username='用户名')
    user.is_active = False  # 记当前用户无效
    user.save()
    print("删除普通用户成功！")
except:
    print("删除普通用户失败")
return HttpResponseRedirect('/user/info')
```

- 修改密码set_password

```
from django.contrib.auth import models
try:
    user = models.User.objects.get(username='laowei')
    user.set_password('654321')
    user.save()
    return HttpResponse("修改密码成功！")
except:
    return HttpResponse("修改密码失败！")
```

- 检查密码是否正确check_password

```
from django.contrib.auth import models
try:
    user = models.User.objects.get(username='laowei')
    if user.check_password('654321'):  # 成功返回True,失败返回False
        return HttpResponse("密码正确")
    else:
        return HttpResponse("密码错误")
except:
    return HttpResponse("没有此用户！")
```

项目部署

- 项目部署是指在软件开发完毕后，将开发机器上运行的开发板软件实际安装到服务器上进行长期运行
- 部署要分以下几个步骤进行
  1. 在安装机器上安装和配置同版本的数据库
  2. django 项目迁移(在安装机器上配置与开发环境相同的python版本及依懒的包)
  3. 用 uwsgi 替代`python3 manage.py runserver`方法启动服务器
  4. 配置 nginx 反向代理服务器
  5. 用nginx 配置静态文件路径,解决静态路径问题

1. 安装同版本的数据库

    - 安装步骤略
2. django 项目迁移
    1. 安装python
        - `$ sudo apt install python3`
    2. 安装相同版本的包
        - 导出当前模块数据包的信息:
            - `$ pip3 freeze > package_list.txt`
        - 导入到另一台新主机
            - `$ pip3 install -r package_list.txt`
    3. 将当前项目源代码复制到运程主机上(scp 命令)
        - $ sudo scp -a 当前项目源代码 远程主机地址和文件夹

WSGI Django工作环境部署

- WSGI (Web Server Gateway Interface)Web服务器网关接口，是Python应用程序或框架和Web服务器之间的一种接口，被广泛使用
- 它实现了WSGI协议、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。

uWSGI 网关接口配置 (ubuntu 18.04 配置)

- 使用 `python manage.py runserver` 通常只在开发和测试环境中使用。
- 当开发结束后，完善的项目代码需要在一个高效稳定的环境中运行，这时可以使用uWSGI
- uWSGI是WSGI的一种,它可以让Django、Flask等开发的web站点运行其中.
- 安装uWSGI
    - 在线安装 uwsgi

      ```
      $ sudo pip3 install uwsgi
      ```

    - 离线安装
        1. 在线下载安装包:

           ```
           $ pip3 download uwsgi
           ```

            - 下载后的文件为 `uwsgi-2.0.18.tar.gz`
        2. 离线安装

           ```
           $ tar -xzvf uwsgi-2.0.18.tar.gz
           $ cd uwsgi-2.0.18
           $ sudo python3 setup.py install
           ```

- 配置uWSGI
    - 添加配置文件 `项目文件夹/uwsgi.ini`
        - 如: mysite1/uwsgi.ini

      ```
      [uwsgi]
      # 套接字方式的 IP地址:端口号
      # socket=127.0.0.1:8000
      # Http通信方式的 IP地址:端口号
      http=127.0.0.1:8000
      # 项目当前工作目录
      chdir=/home/weimz/.../my_project 这里需要换为项目文件夹的绝对路径
      # 项目中wsgi.py文件的目录，相对于当前工作目录
      wsgi-file=my_project/wsgi.py
      # 进程个数
      process=4
      # 每个进程的线程个数
      threads=2
      # 服务的pid记录文件
      pidfile=uwsgi.pid
      # 服务的目志文件位置
      daemonize=uwsgi.log
      ```

- uWSGI的运行管理
    - 启动 uwsgi

```
$ cd 项目文件夹
$ sudo uwsgi --ini 项目文件夹/uwsgi.ini
```

- 停止 uwsgi

```
$ cd 项目文件夹
$ sudo uwsgi --stop uwsgi.pid
```

- 说明:
  - 当uwsgi 启动后,当前django项目的程序已变成后台守护进程,在关闭当前终端时此进程也不会停止。

- 测试:
  - 在浏览器端输入http://127.0.0.1:8000 进行测试
  - 注意，此时端口号为8000

nginx 反向代理配置

- Nginx是轻量级的高性能Web服务器，提供了诸如HTTP代理和反向代理、负载均衡、缓存等一系列重要特性，在实践之中使用广泛。
- C语言编写，执行效率高
- nginx 作用
  - 负载均衡， 多台服务器轮流处理请求
  - 反向代理
- 原理:
- 客户端请求nginx,再由nginx 请求 uwsgi, 运行django下的python代码
- ubuntu 下 nginx 安装 $ sudo apt install nginx
- nginx 配置
  - 修改nginx 的配置文件 /etc/nginx/sites-available/default

```
# 在server节点下添加新的location项，指向uwsgi的ip与端口。
server {
    ...
    location / {
        uwsgi_pass 127.0.0.1:8000;  # 重定向到127.0.0.1的8000端口
        include /etc/nginx/uwsgi_params; # 将所有的参数转到uwsgi下
    }
    ...
}
```

- nginx服务控制

```
$ sudo /etc/init.d/nginx start|stop|restart|status
# 或
$ sudo service nginx start|stop|restart|status
```

> 通过 start,stop,restart,status 可能实现nginx服务的启动、停止、重启、查扑克状态等操作

- 修改uWSGI配置
  - 修改`项目文件夹/uwsgi.ini`下的Http通信方式改为socket通信方式,如:

```
[uwsgi]
# 去掉如下
# http=127.0.0.1:8000
# 改为
socket=127.0.0.1:8000
```

  - 重启uWSGI服务

```
$ sudo uwsgi --stop uwsgi.pid
$ sudo uwsgi --ini 项目文件夹/uwsgi.ini
```

```
- 测试:
    - 在浏览器端输入<http://127.0.0.1> 进行测试
    - 注意，此时端口号为80(nginx默认值)

### nginx 配置静态文件路径
```

- 解决静态路径问题
  ```ini
  # file : /etc/nginx/sites-available/default
  # 新添加location /static 路由配置，重定向到指定的绝对路径
  server {
      ...
      location /static {
          # root static文件夹所在的绝对路径,如:
          root /home/weimz/my_django_project; # 重定向/static请求的路径，这里改为你项目的文件夹
      }
      ...
  }
  ```

- 修改配置文件后需要重新启动 nginx 服务

404 界面

- 在模板文件夹内添加 404.html 模版，当视图触发Http404 异常时将会被显示
- 404.html 仅在发布版中(即setting.py 中的 DEBUG=False时) 才起作用
- 当向应处理函数触发Http404异常时就会跳转到404界面

```python
from django.http import Http404
def xxx_view(request):
    raise Http404  # 直接返回404
```