Django 准备工作

这节课可以说是上一课的升级，是创建一个 django 网页更加正规的流程，所以在这里整理出来，帮你在回顾时翻阅。

为了保证流程的完整性，在这里是从最初的创建流程开始，而不是像课程里那样在上节课的基础上调整。

1. 创建 Django project

① start project

创建一个文件夹，命名为 root（或者你喜欢的名字），打开命令行/终端，用命令行打开文件夹，也就是 cd +文件路径：

```
cd /Users/Hou/Desktop/root
# 这里需要替换成你的文件路径，Mac 可以直接拖拽到终端里面显示文件路径，windows 需要在资源管理器查看文件路径
```

使用 django-admin 的命令创建一个项目，命名为 firstsite （或者你喜欢的名字）

```
django-admin startproject firstsite
```

在最开始创建的 root 文件夹中，你会得到一个这样结构的一堆文件

```
firstsite
├── manage.py
└── firstsite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

在 Atom 里面打开 firstsite 这个文件夹的目录，然后打开manage.py，在第一行代码里面把 python 改成 python3，也就是

```
#!/usr/bin/env python3
```

改好记得保存下。

2. 创建 Django App

你可能会说，靠，django project 和 django app 到底什么关系，可以看下这段来自 django girl 的解释：

每一個 Django project 裡面可以有多個 Django apps，可以想成是類似模組的概念。在實務上，通常會依功能分成不同 app，方便未來的維護和重複使用。

例如，我們要做一個類似 Facebook 這種網站時，依功能可能會有以下 apps：

- 使用者管理 -- accounts
- 好友管理 -- friends
- 塗鴉牆管理 -- timeline
- 動態消息管理 -- news

若未來我們需要寫個購物網站，而需要會員功能時，accounts app（使用者管理）就可以被重複使用。

好了，现在我们开始创建第一个 django app!

① start app

先 cd 到 firstsite 目录下（如果用了atom 的 terminal 插件或者 pycharm 就不用手动 cd 了），然后输入：

```
python3 manage.py startapp firstapp
```

manage.py 是 Django 提供的命令行工具，我们可以利用它执行很多工作，使用方法类似这样：

```
python manage.py <command> [options]
```

如果你想要了解有哪些命令可以使用，输入 help 会列出所有：

```
python manage.py help
```

运行了刚才的命令之后，文件结构又变成了这么一堆东西：

```
firstsite
├── manage.py
├── firstsite
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── firstapp
    ├── __init__.py
    ├── admin.py
    ├── migrations
    ├── models.py
    ├── tests.py
    └── views.py
```

② setting 里增加 app

之前我们用 Atom 建立了 firstapp 这个 app,但为了让 Django 知道要管理哪些 apps，还需要调整设置。

在 Atom 的 setting 里面找到 INSTALLED_APPS,在末尾添加刚才创建的 app 名字：

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'firstapp'  #新增这个，也就是你的 Django app 名字
]
```

3、创建数据库

等创建好数据库就能看到 django 网站的初始界面了。

① 合并、运行数据库

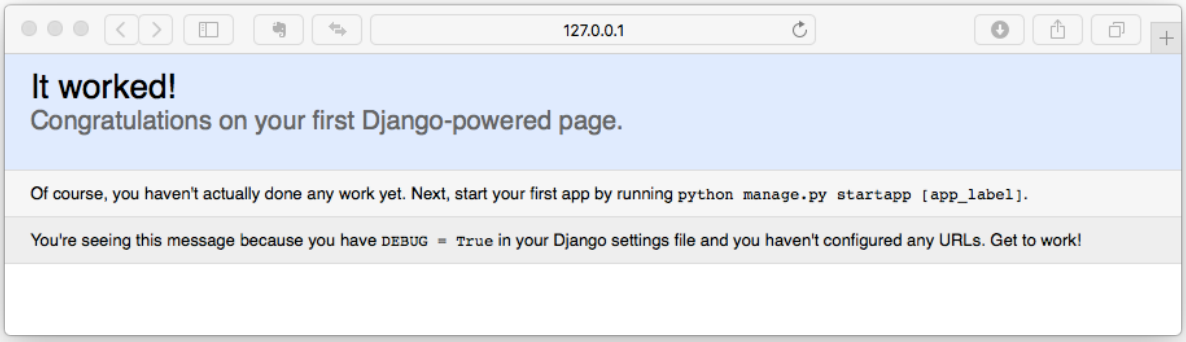
打开terminal,先 cd 到 firstsite 目录下（如果用了atom 的 terminal 插件或者 pycharm 就不用手动 cd 了），然后输入这两行命令合并数据库：

```
python3 manage.py makemigrations
python3 manage.py migrate
```

然后运行服务器，输入：

```
python3 manage.py runserver
```

现在打开浏览器，输入 http://127.0.0.1:8000/ 或是 http://localhost:8000/
会看到你的 Django 网站已经在 web server 上成功运行了！



4. 把 html,CSS,图片放到模板里

① 创建 templates 和 static 文件夹

在你创建的 Django app 文件夹（课程中是 firstapp）下面创建两个文件夹，分别是：templates,static。
然后把 html 文件放到 templates 文件夹中， CSS,images 等所有静态文件放到 static 文件夹中。
你会得到这样的文件结构：

```
firstsite
├── manage.py
├── firstsite
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── firstapp
    ├── __init__.py
    ├── admin.py
    ├── migrations
    ├── models.py
    ├── tests.py
    ├── views.py
    ├── templates
    │   └── first_web_2.html
    └── static
        ├── css
        └── images
```

② 在 setting 里修改模板路径

为了让 django 知道我们的模板放在哪，需要回到 settings.py 中，修改 TEMPLATES 的 DIRS 如下：

```
# TEMPLATES 部分

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
```

```
        'DIRS': [os.path.join(BASE_DIR, 'templates').replace('\\',
'/')], # 新增这一行，修改模板的路径
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

③ 在 html 里增加模板标签

然后回到 templates 文件夹下的 html 文件中，增加必需的模板标签，为所有图片、css 替换路径：

```
# 在 html 首行增加
{% load staticfiles %}

# 把 css 路径替换为这个
href="{% static 'css/semantic.css' %}"

# 把所有图片路径替换为类似这个的格式
{% static 'images/star_banner.jpg' %}
```

5. 创建后台和超级管理员

Django 诞生于新闻网站的环境中，所以很重视内容管理，提供了管理后台，让使用者方便新增或修改网站内容。

这个管理后台，在 Django 中以内置 app 的形式存在，叫做：Django Admin。现在我们来进行设置：

① 建立管理员账号

先 cd 到 firstsite 目录下（如果用了atom 的 terminal 插件或者 pycharm 就不用手动 cd 了），然后输入：

```
python3 manage.py createsuperuser
```

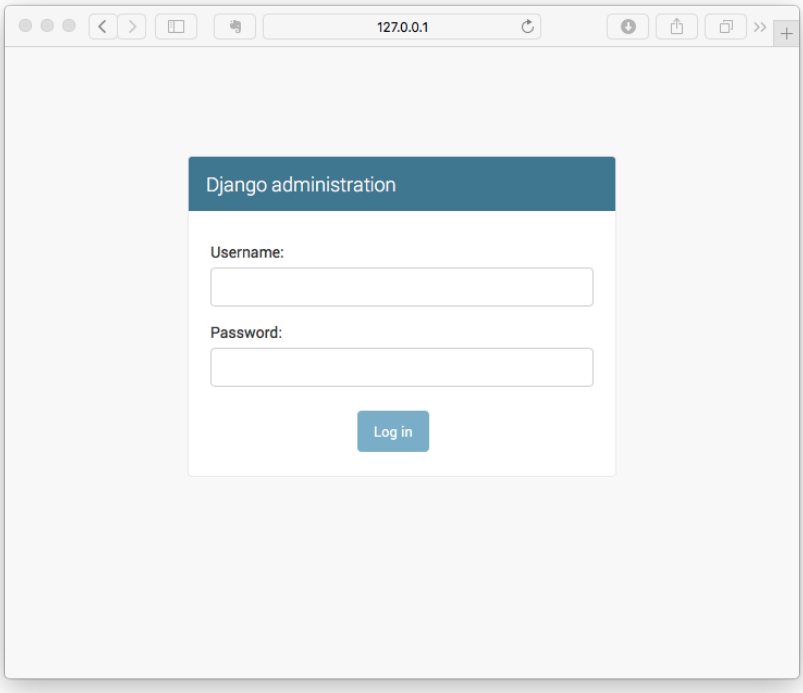
会弹出提示让你分别输入用户名、邮件、密码，要记住哦。然后就完成 superuser （超级管理员）的账号设置了。在终端里面输入名字密码的时候会不显示任何东西，就是这样的交互设计，不是你的键盘坏掉了，放心大胆的输完敲回车就行。

```
Username (leave blank to use 'YOUR_NAME'):
Email address: your_name@yourmail.com
Password:
Password (again):
Superuser created successfully.
```

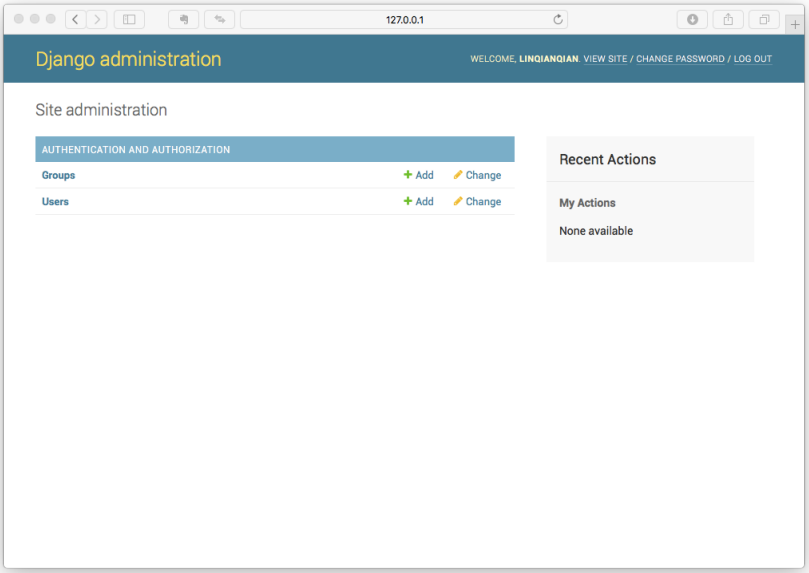
看到最后一行 successfully 就是创建成功了。

② 使用管理后台

执行 runserver 指令,然后进入 http://127.0.0.1:8000/admin ，可以看到管理后台的登录页面，还记得你刚才创建的 superuser的账号密码吧？填进来，进入管理后台。



会看到这样的后台~

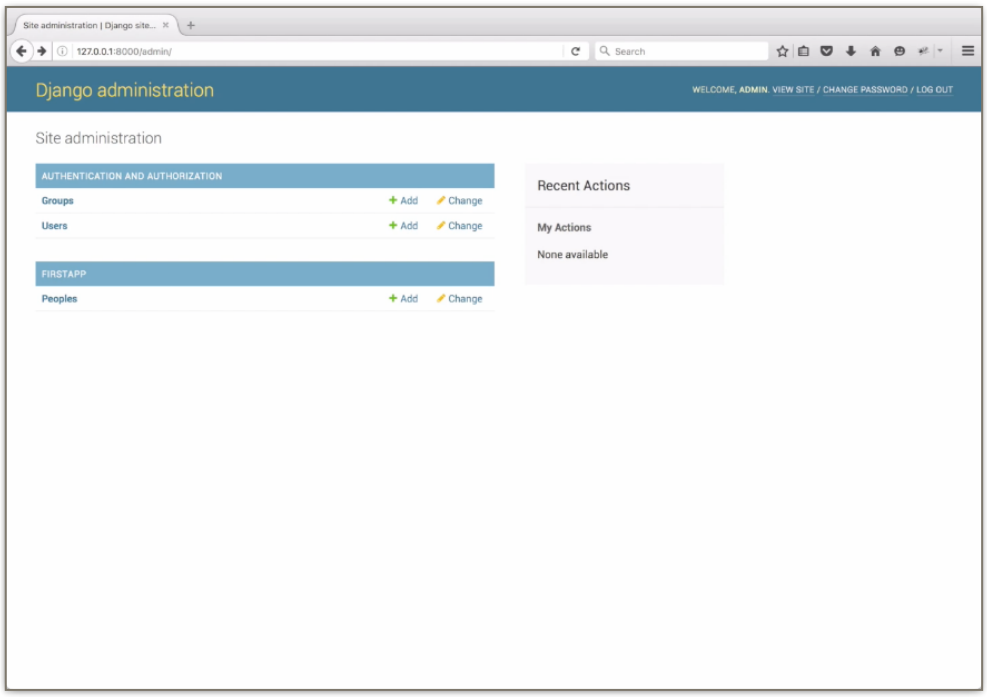


③ 在 admin 里增加想要的管理后台数据项

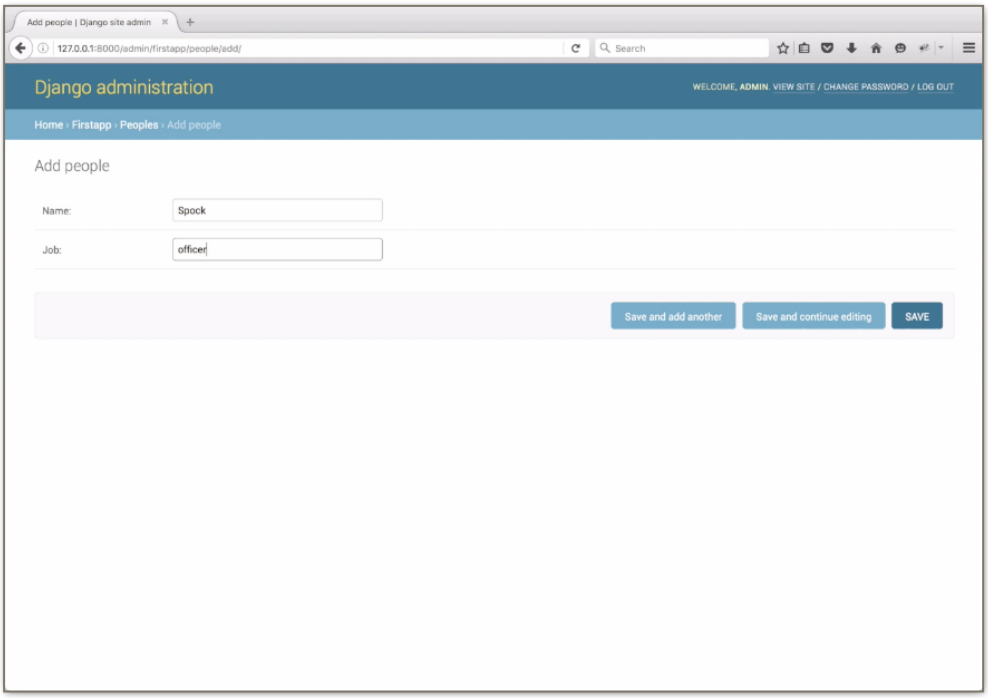
在 admin.py 里面增加想要的数据项：

```
from firstapp.models import People
admin.site.register(People)
```

就能看到管理后台多了 people 这一项，是不是很方便呢



你可以点进来新增内容：

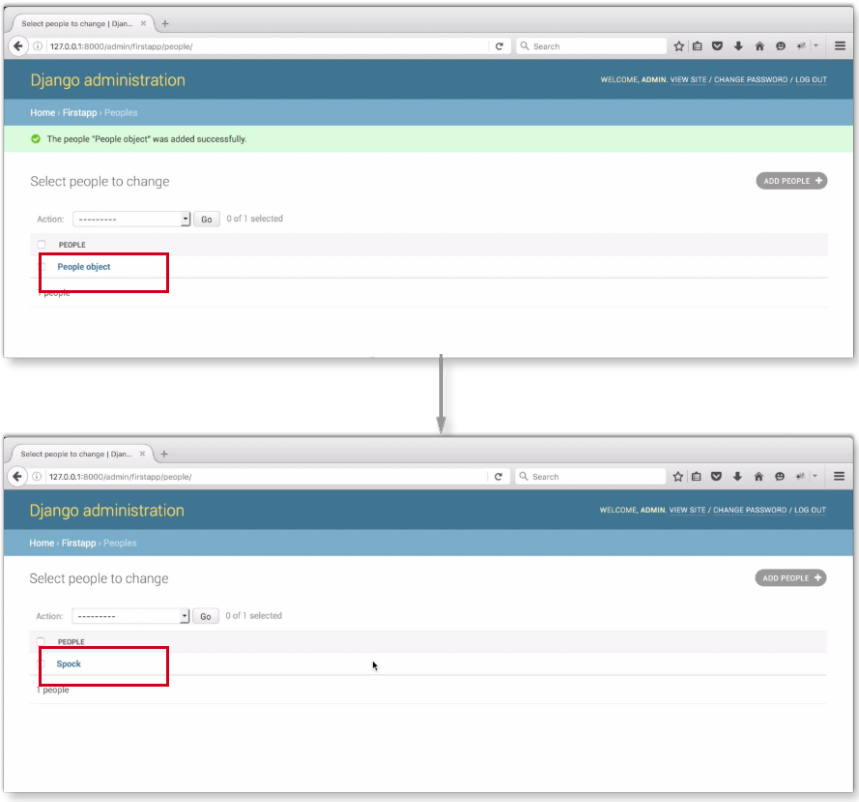


④ 在 models 里面设置内容列表的标题

为了让内容列表里能直接显示标题，需要到 models.py 里面，增加一个直接显示名字的函数

```
class People(models.Model):
    name = models.CharField(null=True, blank=True, max_length=200)
    job = models.CharField(null=True, blank=True, max_length=200)
    def __str__(self):          #需要添加这一行
        return self.name        #还需要添加这一行
```

然后就能看到内容列表的变化。



⑤ 在 admin 里继续增加文章管理

按照上面的方式，再来把文章这一项添到管理后台里面。

在 admin.py 里面继续增加想要的数据项：

```
from firstapp.models import People, Article    # 增加 Article 的引入

admin.site.register(People)
admin.site.register(Article)                   # 增加这行
```

⑥ 在 model 里继续增加文章数据字段

在 models.py 里面定义文章的数据字段：

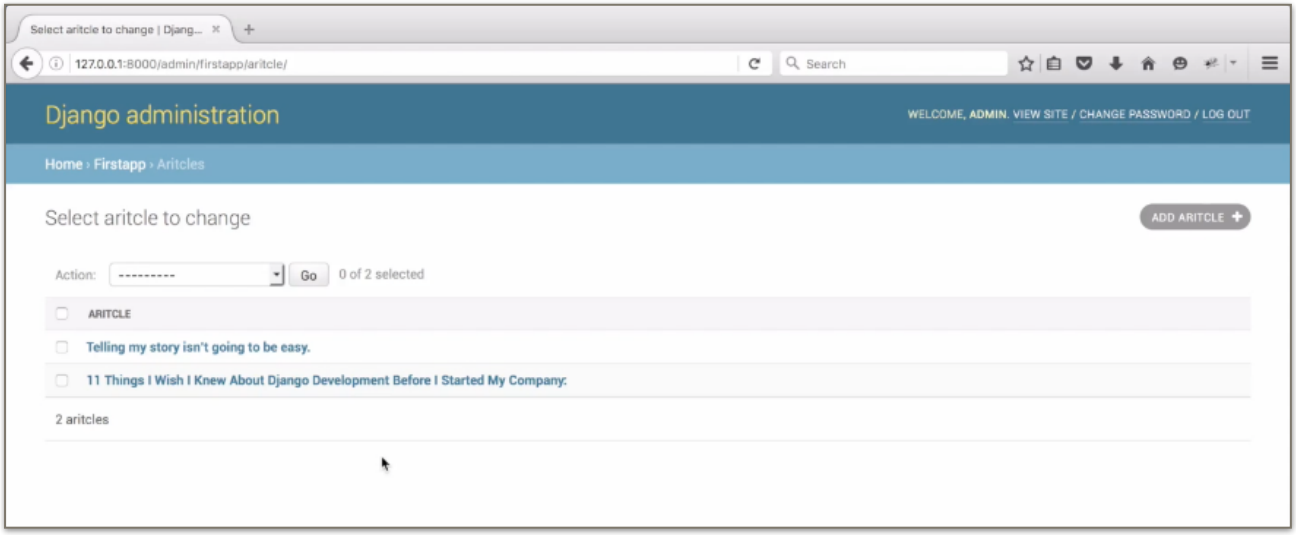
```
class Article(models.Model):
    headline = models.CharField(null=True, blank=True,
max_length=500)
    content = models.TextField(null=True, blank=True)
    def __str__(self):
        return self.headline
```

⑦ 合并数据库

打开terminal,先 cd 到 firstsite 目录下（如果用了atom 的 terminal 插件或者 pycharm 就不用手动 cd 了），然后输入这两行命令合并数据库：

```
python3 manage.py makemigrations
python3 manage.py migrate
```

每次 models 里面有数据改动，都要输入这两行命令，切记切记！
现在就可以到管理后台去添加文章内容了。



6. 在 View 中获取 Model 中的数据

引用 model 里面写好的文章列表，然后去渲染文章列表。

```
from firstapp.models import People, Article   # 增加 Article 的引入

def index(request):
    context = {}
    article_list = Article.objects.all()
    context['article_list'] = article_list
    index_page = render(request, 'first_web_2.html', context)
    return index_page
```

7. 在 Template 中增加动态内容

回到 templates 文件夹下的 html 文件中，增加必需的模板标签，让文章内容可以取管理后台的内容：

```
<div class="ui  vertical segment">
    {% for article in article_list %}
    # 增加模板的 for 循环，从而支持无数篇文章都能用管理后台的数据
        <div class="ui container vertical segment">
            <a href="#">
                <h1 class="ui header">
                    {{ article.headline }}  # 为文章标题增加模板标签
                </h1>
            </a>
            <i class="icon grey small unhide">10,000</i>
            <p>
                {{ article.content|truncatewords:100 }}
                # 为文章内容增加模板标签
                <a href="#">
                    <i class="angle tiny double grey right
icon">READMORE</i>
                </a>
            </p>
            <div class="ui mini  tag label">
                life
            </div>
        </div>
    {% endfor %}  # 增加 for 循环的结束标签
</div>
```

8. 在 URL 中分配网址

在 urls.py 中添加如下代码，作用是让链接可以被访问：

```
from django.conf.urls import url
from django.contrib import admin
from firstapp.views import first_try, index      #添加index

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^first_try/', first_try),
    url(r'^index', index, name='index'),          #添加的代码
]
```

现在打开浏览器，输入：http://127.0.0.1:8000/index

就能看到这个真正的网页啦，嘿，还挺像那么回事的，你要是想自己搭一个博客，就直接把这个模板拿去用好了。