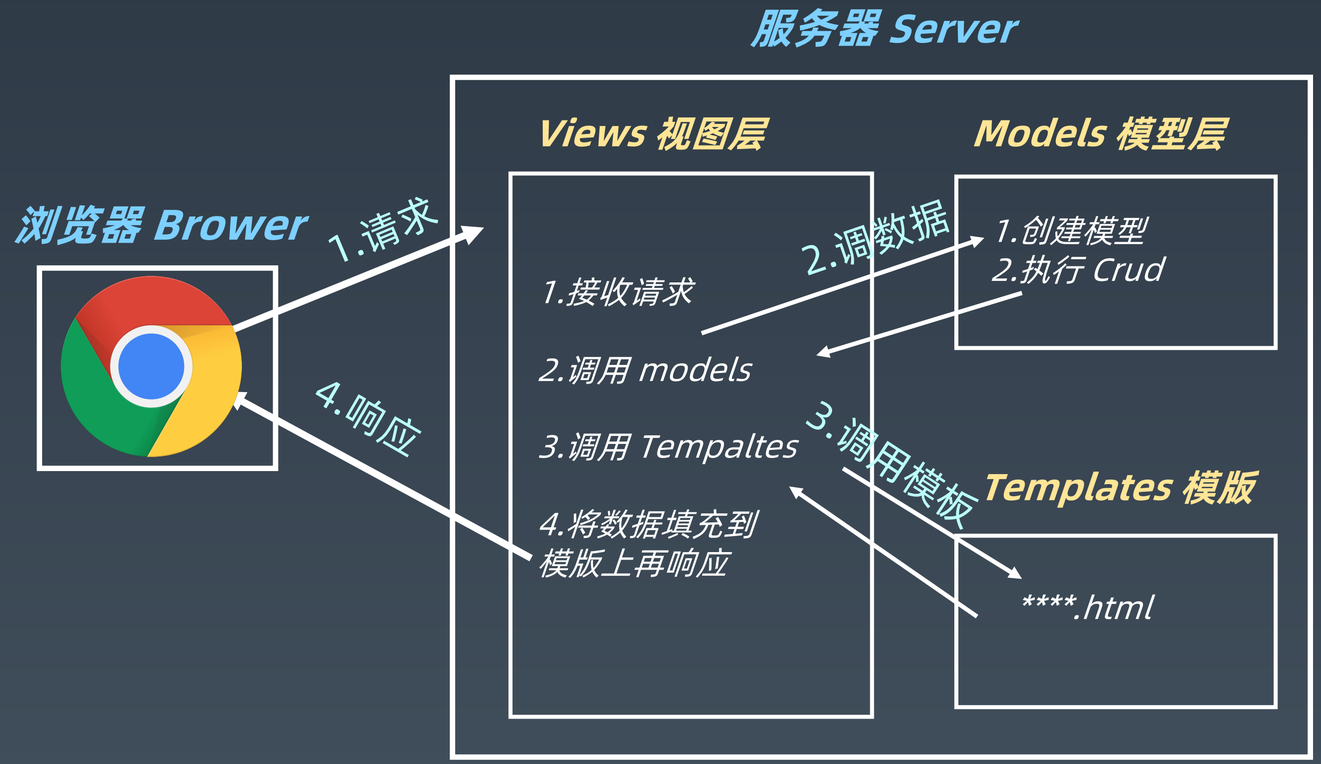
Django 框架简介

- Django 是一个开放源代码的 Web 应用框架
- 最初用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站
- 2005 年 7 月在 BSD 许可证下发布

MTV 框架模式

- 模型（Model）
- 模板（Template）
- 视图（Views）



Django 的特点

- 采用了 MTV 的框架
- 强调快速开发和代码复用 DRY (Do Not Repeat Yourself)
- 组件丰富：

  ORM (对象关系映射) 映射类来构建数据模型

  URL 支持正则表达式

  模板可继承

  内置用户认证，提供用户认证和权限功能

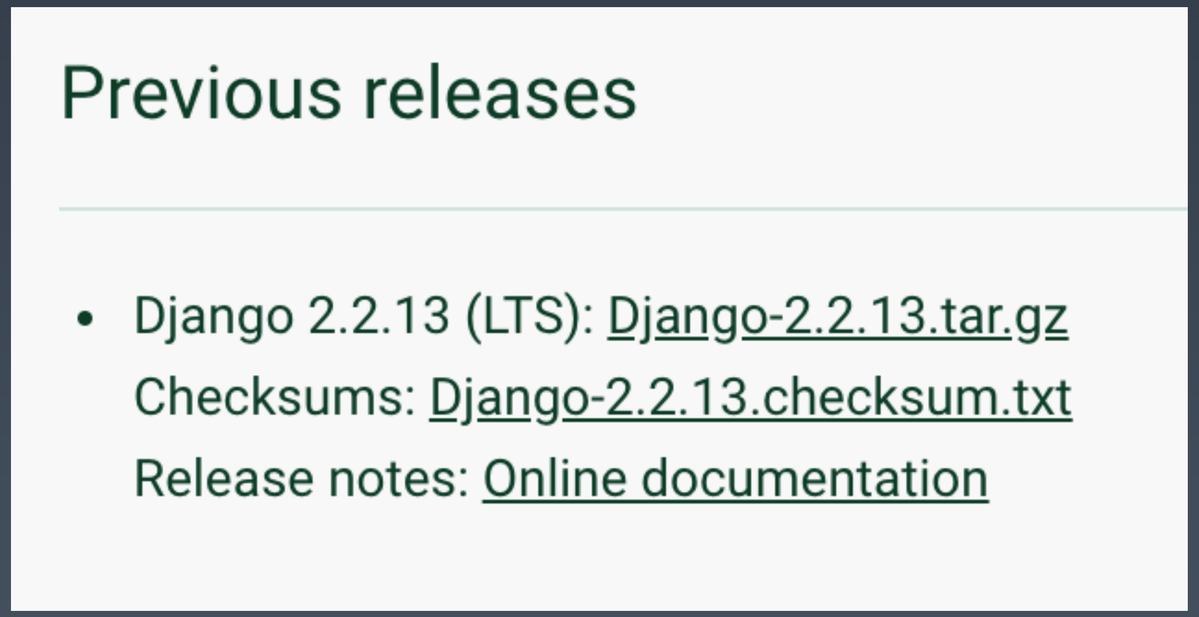
  admin 管理系统

  内置表单模型、Cache 缓存系统、国际化系统等

Django 的版本

Django 最新 3.0 版本，目前比较多的是 2.2.13（LTS）

```
$ pip install --upgrade django==2.2.13
>>> import django
>>> django.__version__
'2.2.13'
```



创建 Django 项目

```
$ django-admin startproject MyDjango
```

目录结构如下：

```
$ find MyDjango/
MyDjango/
MyDjango/manage.py                    命令行工具
MyDjango/MyDjango
MyDjango/MyDjango/__init__.py
MyDjango/MyDjango/settings.py         项目的配置文件
MyDjango/MyDjango/urls.py
MyDjango/MyDjango/wsgi.py
```

创建 Django 应用程序

$ python manage.py help 查看该工具的具体功能

$ python manage.py startapp index

index/migrations	数据库迁移文件夹
index/models.py	模型
index/apps.py	当前 app 配置文件
index/admin.py	管理后台
index/tests.py	自动化测试
index/views.py	视图

启动和停止 Django 应用程序

$ python manage.py runserver

默认是127.0.0.1:8000

$ python manage.py runserver 0.0.0.0:80

Quit the server with CONTROL-C

$ CONTROL-C

Django 的配置文件

配置文件包括：

- 项目路径
- 密钥
- 域名访问权限
- App 列表
- 静态资源，包括 CSS、JavaScript 图片等
- 模板文件
- 数据库配置
- 缓存
- 中间件

URL 调度器

MyDjango/urls.py 文件中的 urlpatterns 列表，实现了：

从 URL 路由到视图 (views) 的映射功能

过程中使用了一个 Python 模块，**URLconf**(URL configuration)，通常这个功能

也被称作 URLconf

Django 如何处理一个请求

当一个用户请求 Django 站点的一个页面：

1. 如果传入 HttpRequest 对象拥有 urlconf 属性（通过中间件设置），它的值将被用来代替 ROOT_URLCONF 设置。
2. Django 加载 URLconf 模块并寻找可用的 urlpatterns，Django 依次匹配每个 URL 模式，在与请求的 URL 匹配的第一个模式停下来。
3. 一旦有 URL 匹配成功，Djagno 导入并调用相关的视图，视图会获得如下参数：
    - 一个 HttpRequest 实例
    - 一个或多个位置参数提供
4. 如果没有 URL 被匹配，或者匹配过程中出现了异常，Django 会调用一个适当的错误处理视图。

增加项目 urls

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include('index.urls')),
]
```

增加 index 的 urls

```
# index/urls.py
from django.urls import path
from . import views

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

# index/views.py
from django.shortcuts import render
from django.http import HttpResponse
def index(request):
    return HttpResponse("Hello Django!")
```

模块和包

模块： .py 结尾的 Python 程序

包： 存放多个模块的目录

__init__.py 包运行的初始化文件，可以是空文件

常见以下几种方式导入：

import

from ... import ...

from ... import ... as ...

带变量的 URL

Django 支持对 URL 设置变量, URL 变量类型包括：

- str
- int
- slug
- uuid
- path

```
path('<int:year>', views.myyear),
```

URL 支持正则表达式

urls.py

```
re_path('(?P<year>[0-9]{4}).html', views.myyear, name='urlyear’),
```

views.py

```
def myyear(request, year):
    return render(request, 'yearview.html')
```

Templates 文件夹增加 yearview.html

```
<a href="{%  url 'urlyear' 2020 %}">2020 booklist</a></div>
```

view 视图

响应类型	说明
HttpResponse(‘Hello world’)	HTTP 状态码 200，请求已成功被服务器接收
HttpResponseRedirect(‘/admin/’)	HTTP 状态码 302，重定向 Admin 站点的 URL
HttpResponsePermanentRedirect(‘/admin/’)	HTTP 状态码 301，永久重定向 Admin 站点URL
HttpResponseBadRequest(‘BadRequest’)	HTTP 状态码 400，访问的页面不存在或者请求错误
HttpResponseNotFound(‘NotFound’)	HTTP 状态码 404，页面不存在或者网页的 URL 失效
HttpResponseForbidden(‘NotFound’)	HTTP 状态码 403，没有访问权限
HttpResponseNotAllowed(‘NotAllowedGet’)	HTTP 状态码 405，不允许使用该请求方式
HttpResponseSeverError(‘SeverError’)	HTTP 状态码 500，服务器内容错误

Django 快捷函数

render()

将给定的模板与给定的上下文字典组合在一起，并以渲染的文本返回一个 HttpResponse 对象。

redirect()

将一个 HttpResponseRedirect 返回到传递的参数的适当URL。

get_object_or_404()

在给定的模型管理器(model manager) 上调用 get() ，但它会引发 Http404 而不是模型的 DoesNotExist 异常。

基于类的视图

因此在视图函数里处理 HTTP GET 的代码应该像下面这样：

```
from django.http import HttpResponse

def my_view(request):
    if request.method == 'GET':
        # <view logic>
        return HttpResponse('result')
```

而在基于类的视图里，会变成：

```
from django.http import HttpResponse
from django.views import View

class MyView(View):
    def get(self, request):
        # <view logic>
        return HttpResponse('result')
```

模型与数据库

- 每个模型都是一个 Python 的类，这些类继承 django.db.models.Model
- 模型类的每个属性都相当于一个数据库的字段
- 利用这些，Django 提供了一个自动生成访问数据库的 API

```
from django.db import models
class Person(models.Model):
    id = models.IntegerField(primary_key=True)
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

对应 SQL

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

```
$ python manage.py makemigrations
$ python manage.py migrate
```

字段

字段类型

模型中每一个字段都应该是某个 Field 类的实例

字段选项

每一种字段都需要指定一些特定的参数

参考：https://docs.djangoproject.com/zh-hans/2.2/topics/db/models/

关联关系

- 多对一
- 多对多
- 一对一

查询

执行查询常见方法：

- save()
- add()
- all()
- filter()
- get()

事务

装饰器

```
from django.db import transaction
@transaction.atomic
@transaction.non_atomic_requests
```

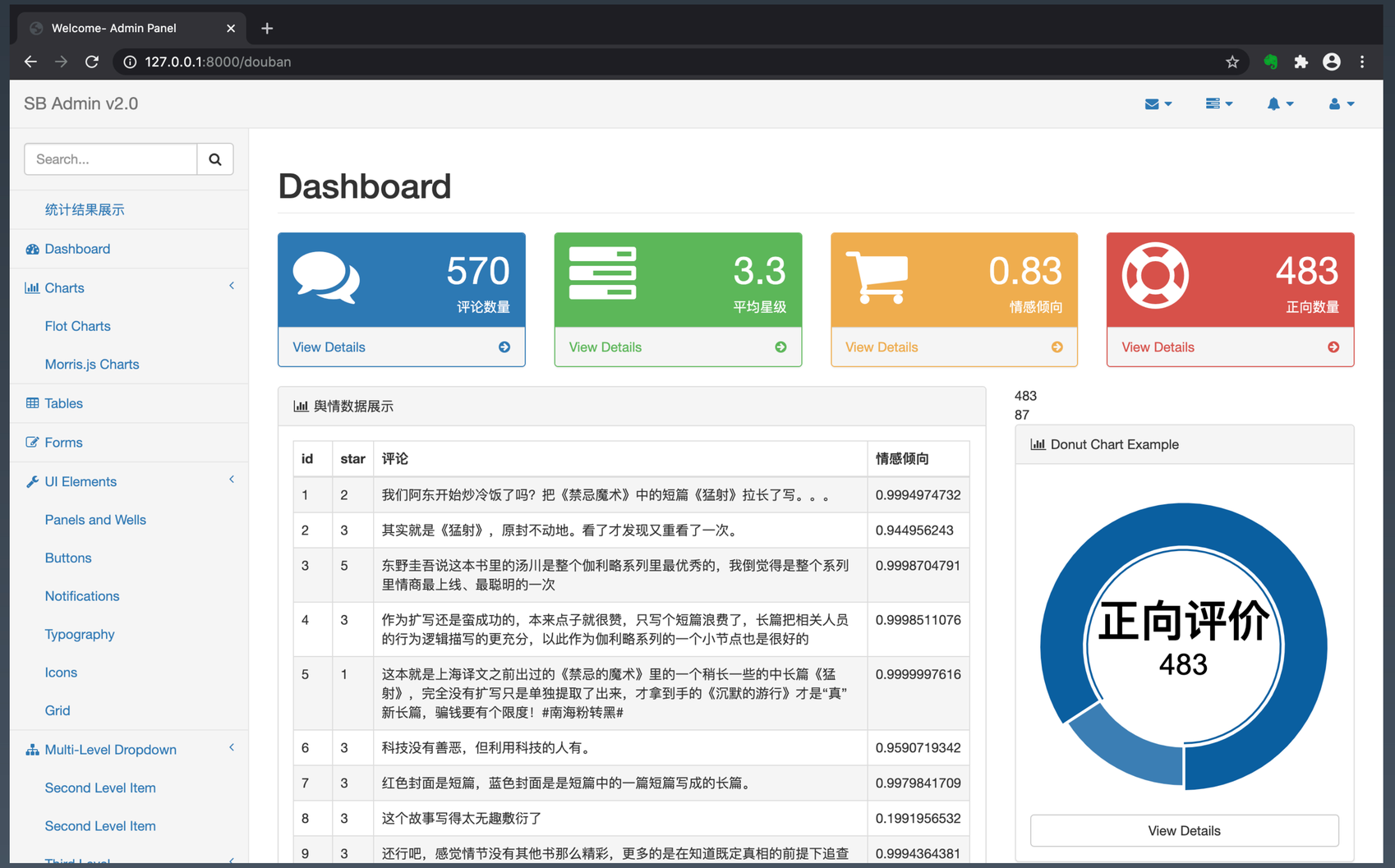
保存点

```
with transaction.atomic():
    transaction.on_commit(foo)
```

模板

- jieba 分词与提取关键词
- jieba 调整词典
- jieba 词性标注

豆瓣书评展示


Welcome- Admin Panel
127.0.0.1:8000/douban
SB Admin v2.0
Search...
统计结果展示
Dashboard
Charts
Flot Charts
Morris.js Charts
Tables
Forms
UI Elements
Panels and Wells
Buttons
Notifications
Typography
Icons
Grid
Multi-Level Dropdown
Second Level Item
Second Level Item
Dashboard
570
评论数量
View Details
3.3
平均星级
View Details
0.83
情感倾向
View Details
483
正向数量
View Details
舆情数据展示
id
star
评论
情感倾向
1
2
我们阿东开始炒冷饭了吗？把《禁忌魔术》中的短篇《猛射》拉长了写。。。
0.9994974732
2
3
其实就是《猛射》，原封不动地。看了才发现又重看了一次。
0.944956243
3
5
东野圭吾说这本书里的汤川是整个伽利略系列里最优秀的，我倒觉得是整个系列里情商最上线、最聪明的一次
0.9998704791
4
3
作为扩写还是蛮成功的，本来点子就很赞，只写个短篇浪费了，长篇把相关人员的行为逻辑描写的更充分，以此作为伽利略系列的一个小节点也是很好的
0.9998511076
5
1
这本就是上海译文之前出过的《禁忌的魔术》里的一个稍长一些的中长篇《猛射》，完全没有扩写只是单独提取了出来，才拿到手的《沉默的游行》才是“真”新长篇，骗钱要有个限度！#南海粉转黑#
0.9999997616
6
3
科技没有善恶，但利用科技的人有。
0.9590719342
7
3
红色封面是短篇，蓝色封面是是短篇中的一篇短篇写成的长篇。
0.9979841709
8
3
这个故事写得太无趣敷衍了
0.1991956532
9
3
还行吧，感觉情节没有其他书那么精彩，更多的是在知道既定真相的前提下追查
0.9994364381
483
87
Donut Chart Example
正向评价
483
View Details

URLconf 的处理

http://ip/xxx

http://ip/yyy

http://ip/douban/xxx

http://ip/douban/yyy

将 URL 中的 douban/ 注册成一个独立的 APP

settings.py 的处理

- 注册 APP

```
INSTALLED_APPS=[Douban]
```

- 数据库

```
DATABASES = {
    ... ...
}
```

模型的处理

- 从 Django 到 SQL
  - python manage.py migrate
- 从 SQL 到 Django
  - python manage.py inspectdb

视图的处理

- objects.all() #选取所有数据
- objects.all().count() #计数
- objects.values('n_star') #某一列数据
- queryset.filter(**condtions).count() #符合条件计数

URL 规则的处理

MyDjango/MyDjango/urls.py

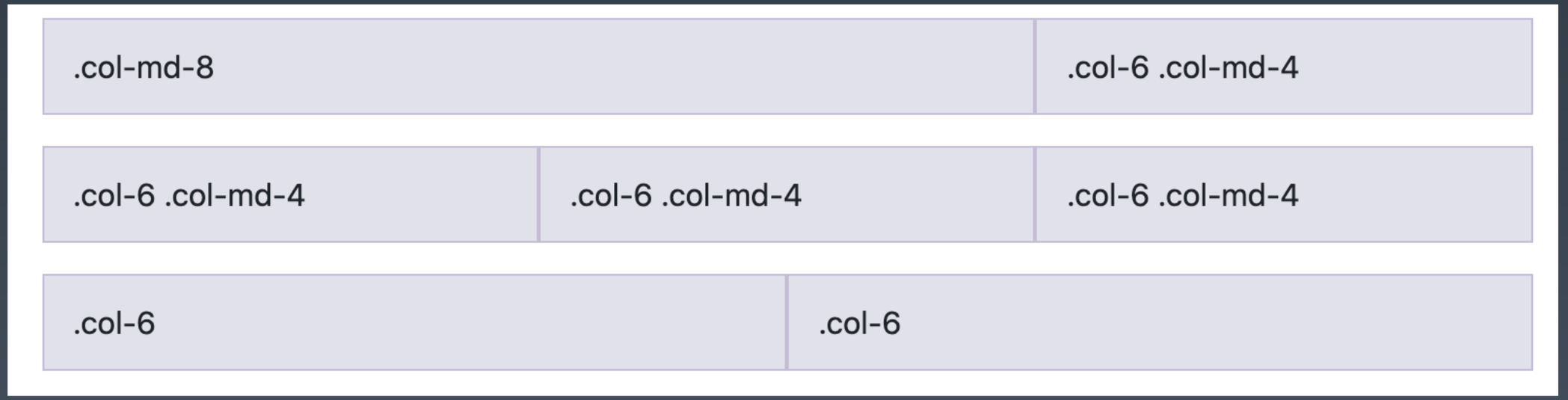
```
urlpatterns = [
    path('douban/',include('Douban.urls'))
]
```

MyDjango/Douban/urls.py

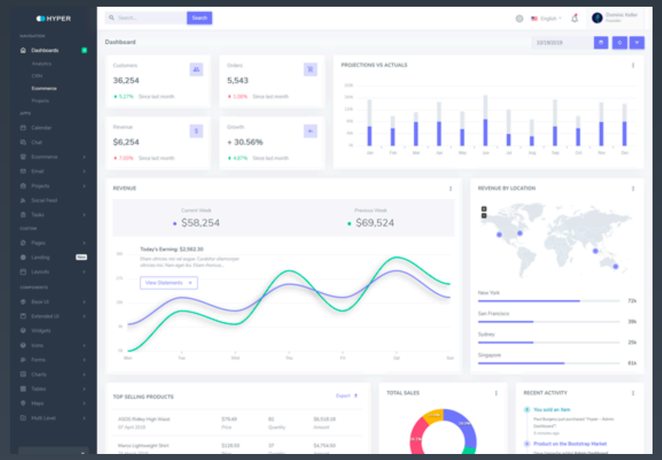
```
urlpatterns = [
    path('index', views.books_short)
]
```

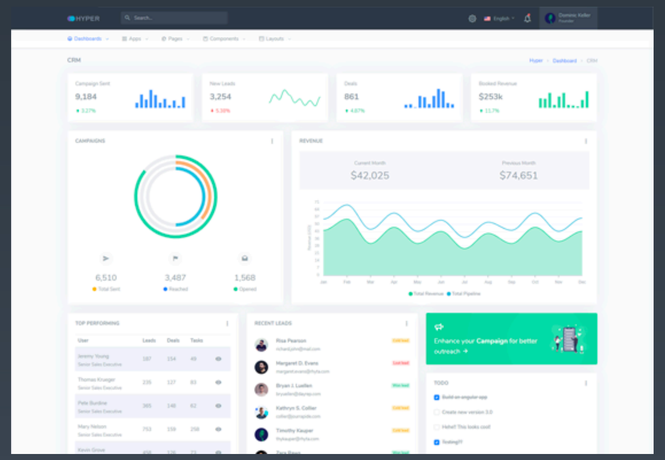
BootStrap 模版

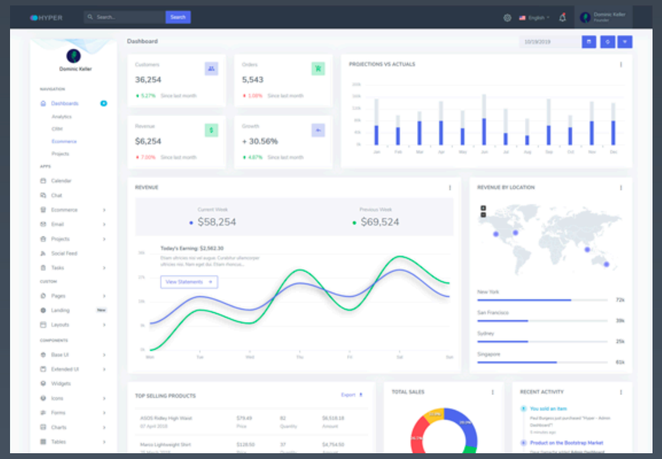
- 栅格系统

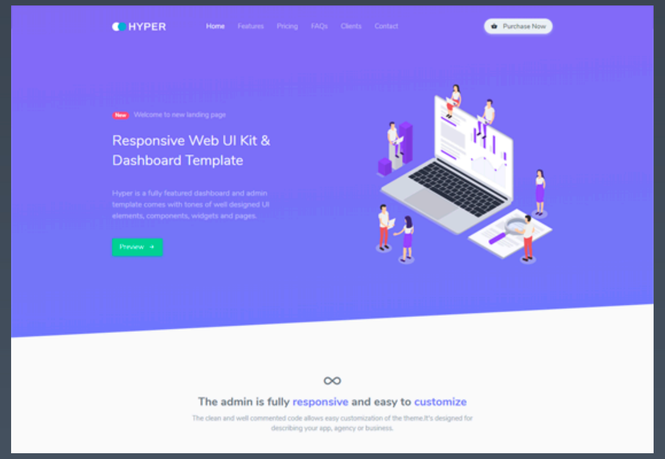


BootStrap 模版









为什么要阅读源代码

- 解决文档里没有描述的特定问题
- 二次开发
- 学习语言 -- 代码风格、规范、高级语法
- 学习设计模式 -- 接口、框架、架构
- 学习算法
- 阅读源代码不是唯一的学习手段，也不是最高效的那个

从哪里开始阅读源码

- 从 manage.py 开始
  - 一个非常典型的 Python 入口 `if __name__ == '__main__':`

最终目的

- 接收用户的 HTTP 请求

manage.py 做了些什么

1. 解析 manage.py 的 runserver 和 IP 端口参数
2. 找到 command 目录加载 runserver.py
3. 检查 INSTALL_APP、IP 地址、端口、ORM 对象
4. 实例化 wsgiserver
5. 动态创建类并接收用户的请求

THANKS!