XxuCMS 设计说明

徐潇

2017.12.26

I 系统环境、项目初始化

Ubuntu16.04 Emacs24.5 TexLive2005 Ctex Python3 Django2.0 Sqlite3 Bootstrap3 Jquery1.11

I.1 虚拟环境设置

在 Ubuntu16.04 下设置虚拟环境，在虚拟环境下安装各类软件、开发工具，互相不影响，只对当前虚拟环境产生影响。

1) 安装虚拟环境软件

>sudo apt install virtualenv

2) 创建一个虚拟环境（会自动建设目录，该目录就是一个虚拟环境，以后安装的所有的东西都在该目录中），这里创建了一个虚拟环境目录 (virtualpython3)，并没有启动，我使用了 python3 开发环境:

>virtualenv -p python3 virtualpython3

3) 启动/退出虚拟环境 (先进入环境目录 virtualpython3)：

>source ./bin/activate

>deactivate

I.2 安装 django 创建项目、应用

在虚拟环境下安装 django 并创建一个 django 项目及应用 k。

1) 安装 Django

>sudo apt install django

2) 创建 django 项目与应用。Django 的项目是一个完整的 WEB 项目目录，里面可以包含多个应用，每个应用有其自己的视图、模式、模板，在项目中保存共有的 URL、ADMIN 管理、数据库管理等。每个应用程序可以

看成项目的一个模块，采用多应用方式可以更好地实现迁移、复用。如下，先创建一个项目（自动创建一项目同名目录，生成一系列的目录。再生成一个应用（在项目中自动生成一个应用的同名目录，目录内含此应用的一系列文件）。

>django-admin startproject xxucms

>cd xxucms

>manage.py startapp logreg

I.3 初始化数据库、设置越级用户、测试项目

在创建了项目、应用后，做一点初始工作就可以直接启动应用（WEB）进行测试并进行后台管理，不用写一行代码，如果不用后台管理，连初始化数据库、建立用户都不用。

>manage.py migrate # 初始化数据库

>manage.py createsuperuser # 创建一个越级用户 (输入用户名、密码)

>manage.py runserver # 运行应用，可从 127.0.0.1:8000 访问或加 /admin 后进行后台管理。

II 开发记录

记录一个 CMS 设计过程中的详细经过及出现的各种问题的解决方法。

II.1 目录结构

按照前面的过程，新建虚拟环境、创建一个项目、创建一个应用，创建首页、注册页、公用模板目录、静态目录等。

```
xxucms.....................................................项目根目录
├── db.sqlite3 ..........................................数据库文件
├── doc ..................................................设计文档目录
│   └── design.tex.............................文档目录中的一个文档
├── hiddendanger ..... 在项目中创建的一个应用，下含一些文件
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── models.py
│   └── views.py
├── manage.py.........................创建项目时生成的项目管理工具
├── static ..... 自建的静态文件目录，保存图片、包库等
│   └── bootstrap-3.3.7
│       ├── css
│       ├── fonts
│       └── js
├── templates ..... 公用模板目录，如首页、登录模板等
│   ├── index.html
│   └── userreg.html
└── xxucms.......................项目主目录，下含一些全局配置文件
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    ├── views.py
    └── wsgi.py
```

目录及文件详细说明：1、最顶层的 xxucms 目录是创建项目时的目录也就是项目名称，里面的 xxucms 子目录是创建项目是自动生成的项目主目录，包含全局配置文件 settings.py（项目全局设置文件）、urls.py（URL 访问

转发匹配控制)、views.py(全局项目视图)、wsgi.py(WEB 运行配置)、init.py(python 包标志)。2、doc 目录是自建的文档目录。3、hiddendanger 是在项目中创建的一个应用名称，自动生成目录及里面的文件，admin.py、apps.py 都是要和项目或全局配置沟通的文件，后面会介绍，views.py 是默认的视图文件，应用的中视图一般放在自己的这个文件中，便于隔离及应用程序重用，models.py 为应用中用到的模型（就是数据库结构)。4、static 目录保存静态目录。在 django 中要用到的所有其他文件如 bootstrap 框架、js 文件、图片资源等都不能直接使用，要作为静态文件设置好相关配置才能使用，后面会说如何设置。5、templates 目录是自建的公用模板目录，把一个各个应用程序都要用到的模板如首页、登录、项目关于信息等统一放在这，各个应用一般情况下有专有的模板目录。6、文件。db.sqlite3 为数据库文件，这个是 django 支持的最简单的数据库，只有一个文件，在前面说过如何生成数据库，在每次对模型进行了修改后都要进行数据库更新同步;manage.py 是自动生成的项目管理文件，如启动 WEB、运行 django shell 等。

II.2 项目设置

一、settings.py。这个文件在创建项目时在项目目录 (本例为 /xxucms/xxucms) 自动生成，里面包含项目中的全局设置信息，下面把重要的一些进行说明。

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__))) BASE_DIR 为项目的根目录，就是你在创建项目的生成的那个目录，项目中的所有文件、应用目录都是基于这个目录。

DEBUG = True 在开发设计时启动 DEBUG 模式，在出错时会显示详细信息，部署到生产环境是一定要关掉。

INSTALLED_APPS =[
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'hiddendanger',# 这个是新建的应用程序
]

INSTALL_APPS 配置设置了项目中哪些应用被安装启动了，所有的应用都必须被安装启用后才能被访问到。以 django 开头的为系统自带的应用：后台管理、用户认证系统、会话消息管理、静态文件支持等，最后一个

是在项目内建的一个应用，每次新建的应用都要加到这个设置列表中。

MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
#'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

设置了中间件，如在 POST 数据时进行站外交叉访问保护的 csrf，一般不用修改。

TEMPLATES = [
{'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [os.path.join(BASE_DIR,'templates')],
'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',
#'django.core.context_processors.csrf',
],},},]

模板参数设置，DIRS 列表设置了模板目录列表，系统在此列表中寻找匹配的模板文件;APP_DIRS 为 True 则说明可以自动从应用程序的 template 目录中寻找模板;context_processors 列表一般不用修改。

DATABASES = {'default':
{'ENGINE': 'django.db.backends.sqlite3',
'NAME': os.path.join(BASE_DIR, 'db.sqlite3'), }}

数据库设置，目前官方支持数据库有 ORACLE、MYSQL、POSTGRESQL、SQLITE，其它数据库有第三方人员在做。此处设置了使用默认的 sqlite3 数据库，并设置了数据库文件路径及名称。

LANGUAGE_CODE = 'zh-hans'

设置使用的语言，默认为'en-us'，此处改为中文简体，后面自动转为中文界面。

STATIC_URL = '/static/'

设置静态文件访问的 URL, 在使用静态文件时使用% static 来访问.

STATICFILES_DIRS = (os.path.join(BASE_DIR,'static'),)

设置静态文件实际路径，为一个元组，可以设置多个路径。

项目应用中用到的静态文件（bootstrap、图片、声音、CSS、JS 等文件）分类统一放在项目根目录下的 static 目录中。在模板中要用到相应的文件时这样用 (如使用 bootstrap)：

<link href="{% static 'bootstrap-3.3.7/css/bootstrap.min.css' %}" rel="stylesheet">

二、urls.py。创建项目时自动生成在项目主目录中，django 项目启动后所有的 WEB 访问（url）都将由这里的设置转到相应的视图函数中，再由视图函数来决定怎么处理、调用哪个模板。一般小视图不多、单应用程序下用默认这个 URLS 文件就行了，如果应用较多，视图较多，可以用多个 URLS 文件，可以把应用专有的视图放在应用程序的目录下（同样取名 urls.py)。如下是其中一些设置：

```
urlpatterns = [
path('',xxucms.views.index),
path('home/',xxucms.views.index),
path('admin/', admin.site.urls),
path('login/',xxucms.views.xlogin),
path('logout/',xxucms.views.xlogout),
path('userreg/',xxucms.views.userreg),
path('regsave/',xxucms.views.regsave),]
```

简要解释下。第一个空路径 "" 是指在浏览器中访问网站根目录 (如本地 http://127.0.0.1:8088/) 时调用视图中的 index 函数（函数中具体做什么看设计功能，一般最后都要返回一个模板），'home/' 指访问了 http://rootweb/home/ 时也调用 index 视图，其它是一样的意思。所有的 WEB 访问、FORM 动作、点击事件等行为如果要转到某个页面都要在此进行登记，和一个视图函数配对。xxucms.views.index, 为项目目录 (同时也是一个包)xxucms 下 views.py 文件下的视图函数 index。

在项目主目录中自动创建了这个文件，如果有多个项目，所有的 URL 转发都由项目 URLS 来处理会显得很混乱，并且不利于后结应用迁移，这时可以在每个应用中创建自己 urls.py 文件来处理应用的 URL 视图转发。要在应用中使用 URLS.PY 转发，还要在项目的 URLS.PY 文件中配置应用的 urls.py，以通知不同的应用能正确找到相应的视图。如前面新建的应用 hid-

dendange, 为了使所有 rootweb/hiddendanger/* 的 URL 能正确的转发到这个应用的视图中, 只要做两步就行。一、在项目的 urls.py 文件中 urlpatterns = [ ] 中加入一条 path('hiddendanger/',include(hiddendanger.urls)), 指明只要是/hiddendanger/的所有访问都转发到 hiddendange.urls 中转发规则中。二、在应用的 urls.py 中把/hiddendange/看成根 ' ' 就行。具体应用看以后的实例说明。

三、views.py。创建项目时自动生成在项目主目录中, 包含应用中要用到所有实际操作、业务逻辑、程序功能模块，主要是函数。一般在每个应用中也会有这个文件来处理应用本身的业务，文件名称可以任意，只要在 urls.py 中配对好就行。如下例子:

```
def index(request):
    return render_to_response('index.html',{'isok':False})

def xlogin(request):
   '''登录视图'''
   uid = request.POST.get('uid','')
   pwd = request.POST.get('pwd','')
   user = auth.authenticate(username = uid,password = pwd)
   isok = True
   if user is not None and user.is_active:
       auth.login(request,user)
       return render_to_response('index.html',locals())
   else:
       isok = False
       return render_to_response('index.html',locals())

def xlogout(request):
   auth.logout(request)
   return render_to_response('index.html',locals())

def userreg(request):
   '''点击了用户注册后的视图'''
   return render_to_response('userreg.html',{'regsave':False,'regok':False}
                                            )
```

当在浏览器中输入了前面 URLS.PY 中的访问地址或页面中发生了相应的事件就会调用上面的相应视图函数，在视图函数中进行业务逻辑处理，最后再返回一个模板，在模板中对传入的变量进行处理，显示相应的界面。

II.3 设计首页、登录界面

在

II.4 电子书籍管理工具

首页面使用主页模板。上面一行为软件功能区（上传、查询。。。），左边为书籍分类列表。中间开始时为更新列表（显示最近上传的 20 本图书资源）。

模型设计：出版到、作者、书籍类别、图书列表几个模型，其中出版社、类别与书籍是一对多关系，作者与图书是多对多关系。在设计好模型、或对模型进行了更改后，要对模型进行更新，如下。

```
>./manage.py makemigrations  #对所有的更改进行更新
>./manage.py migrate
```

下面创建了实际的模型代码，用上述命令后生成实际的数据库。

```
from django.db import models

"""电子图书模型"""

class BookType(models.Model):
    '''书籍模型'''
    name = models.CharField(max_length = 20,verbose_name = '分类名称')

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']  #按书籍名称升序排列

class Publisher(models.Model):
    '''出版社'''
    name = models.CharField(max_length = 80,verbose_name = '出版社名称')

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']

class Author(models.Model):
    '''作者'''
    name = models.CharField(max_length = 50,verbose_name = '作者')

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']

class Books(models.Model):
    '''书籍模型'''
```

```
    name = models.CharField(max_length = 60,verbose_name = '图书名称')
    booktype = models.ForeignKey(BookType,on_delete = models.CASCADE)  #2.0
                                        开始要加 on_delete
    publisher = models.ForeignKey(Publisher,on_delete = models.CASCADE)
    authors = models.ManyToManyField('Author')
    detial = models.TextField(verbose_name = '内容简介')   #书籍描述
    uploadtime = models.DateTimeField(auto_now_add = True,verbose_name = '上
                                        传时间')
    uploaduser = models.CharField(max_length = 30,verbose_name = '上传用户')
    accessnums = models.IntegerField(verbose_name = '访问次数')
    thumbupnums = models.IntegerField(verbose_name = '点赞')

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']
```