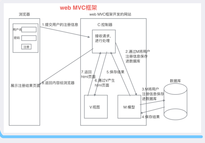
Django概览

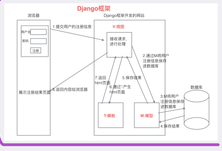
MVC框架

- M(模型)：和数据库交互
- V：视图：产生html页面
- C：控制器：接收请求、进行处理、返回应答



MTV

- M(模型):和数据库交互
- T(模板):产生html页面
- V(视图):接收请求，进行处理，返回应答



虚拟环境搭建和使用

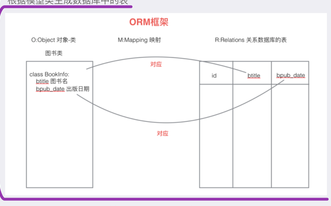
- virtualenvwrapper
  - 进入虚拟环境工作　workon 虚拟环境名
  - 查看机器上有所少个虚拟环境　workon 空格+两个tab键
- virtualenv
  - 创建虚拟环境命令　mkvirtualenv 虚拟环境名
  - 创建python3虚拟环境名　mkvirtualenv -p python3 虚拟环境名
  - 退出虚拟环境　deactivate
  - 删除虚拟环境　rmvirtualenv 虚拟环境名

Django项目创建

- 创建项目
  - 方法
    - django-admin startproject 项目名
    - 或者在pycharm中直接新建Django项目
  - 生成项目目录
    - manage.py 项目的管理文件
    - 项目文件夹
      - __init__.py python包文件夹说明
      - settings.py 项目的配置文件
      - urls.py 进行url路由的配置
      - wsgi.py web服务器和django交互的入口
- 创建应用
  - 方法　进入项目目录后：python manage.py startapp 应用名
  - 生成应用目录
    - __init__.py 说明该目录是一个python模块
    - models.py 写和数据库有关的内容，**设计模型类**
    - views.py 接收请求、进行处理，与M、T进行交互，返回应答，**视图函数或者视图类**
    - tests.py 写测试代码的文件
    - admin.py 网站后台管理相关文件
    - 一般一个项目会有多个应用，所以建议每个应用都手动创建一个自己的urls.py以便管理
    - migrations文件夹
- 应用注册
  - 建立应用和项目之间的联系，需要对应用进行注册。
  - 修改 settings.py 中的 INSTALLED_APPS 配置项
- 修改迁移执行
  - 迁移记录　python manage.py makemigrations 应用名
  - 相应的模型修改记录会记录到migrations里
  - 迁移执行　python manage.py migrate 应用名
- 启动项目　python manage.py runserver

模型M

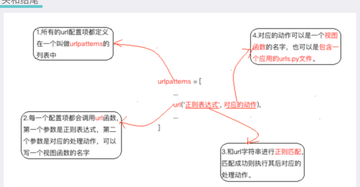
- 模型类设计
  - ORM
    - 建立类和表之间的关系
    - 根据模型类生成数据库中的表



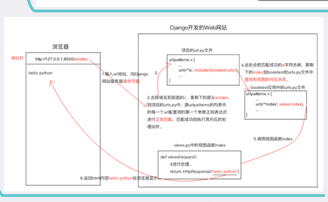
  - models.py
    - 设计模型类
      - 继承models.Model类
      - 每一个模型类实例代表该模型类对应的数据表的一行
      - 外键需要设置在多那一方
- 根据模型类生成表
  - 配置数据库
    - settting.py中的DATABASES
    - Django默认配置数据库为sqlite
      - 'ENGINE': 'django.db.backends.sqlite3',
      - 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    - 若配置数据库为mysql
      - 'ENGINE': 'django.db.backends.mysql',
      - 'NAME': 'test02', # 使用的数据库名称　需要手动在mysql中创建同名的数据库，注意数据类型集为utf8
      - 'USER',
      - 'PASSWORD'
      - 'HOST'
      - 'PORT'
  - 生成迁移历史文件　python manage.py makemigrations 应用名
  - 执行迁移生成表　python mange.py migrate 应用名　默认生成的数据表名为：应用名_模型类名小写
- 通过模型类操作数据表
  - python manage.py shell
  - save()
  - get()
  - delete()
  - .objects.all()
  - 关联操作
    - 多方查询一方　h.hbook
    - 一方查询多方　b.heroinfo_set.all()

视图V

- views.py
- 定义视图函数　视图函数必须有一个参数 request
- URL配置
  - 目的　建立 url 和视图函数的对应关系
  - 严格匹配开头和结尾



- url匹配的过程



模板T

- 模板文件夹的创建以及配置
  - setings.py 文件配置　'DIRS': [os.path.join(BASE_DIR, 'templates')],
  - 除了应用的模板文件，还应建立同级下的静态static文件夹等
    - 图片　images
    - 样式　style
    - js
  - 模板文件夹templates　建议放在项目根目录，然后每个应用均在其下放置一个独自的文件夹
- 模板文件的使用
  - 1.加载模板文件
    - from django.template import loader
    - template = loader.get_template('booktest/index.html')
  - 2.定义模板上下文（参数）
    - 字典类型
    - context = {'title': '图书列表', 'list': range(10)}
  - 3.模板渲染
    - 返回结果
    - from django.http import HttpResponse,
    - return HttpResponse(template.render(context, request))
  - 更简便的方式
    - 使用快捷方式　from django.shortcuts import render
    - context = {'title': '图书列表', 'list': range(10)}
    - return render(request, 'booktest/index.html', context)
  - 重定向　return HttpResponseRedirect('/index')
- 模板参数的使用
  - 模板变量　{{模板变量名}}
    - 模板变量名对应为视图函数中context中的键值
  - 模板代码段　{% 代码段 %}

Django后台管理

- 本地化设置　修改settings.py
  - 修改时区　TIME_ZONE = 'Asia/Shanghai'
  - 修改语言　LANGUAGE_CODE = 'zh-hans'
- 创建管理员　python manage.py createsuperuser
- 注册模型类　修改admin.py　admin.site.register(模型类名)
- 自定义管理页面