复习：

见思维导图

|  |
| --- |
| Node.js原生可做Web应用，但麻烦 => 第三方框架Express可以简化 |
| Python原生可做Web应用，但麻烦 => 第三方框架Django可以简化 |

1.Django

Django是一个基于Python的第三方Web应用开发框架，可以简化Web开发。

官网：https://www.djangoproject.com/

主要特点：

①采用MVC模型变种：MTV（Model Template View）

②采用ORM映射技术：访问数据库无需编写任何SQL语句

③采用URL分发技术：把客户端请求地址分派给不同的View来处理

使用Django框架的步骤：

①安装正确版本的Python：Django 2.2 必需 Python3+

②下载并安装Django框架

pip install Django

默认安装在： PYTHON主目录/Lib/site-scripts

③使用Django提供的管理工具创建空白项目

django-admin startproject 项目名

④进入空白项目，运行其自带的开发服务器

cd 项目名

python manage.py runserver

⑤使用客户端浏览器测试访问

http://127.0.0.1:8000

注意：如果服务器启动成功，客户端打开等待半天仍无法访问，可能是Windows防火墙没有关闭，关闭系统自带的防火墙即可。

|  |
| --- |
| 提示：如果使用pip的默认仓库下载第三方模块总出错误，可以切换为国内仓库，例如：  pip install 模块名 -i https://pypi.tuna.tsinghua.edu.cn/simple |

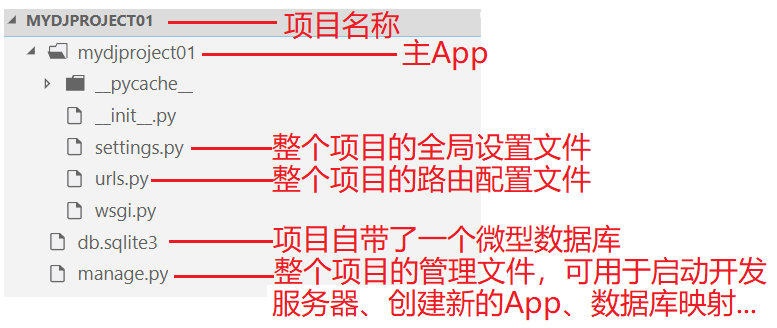
|  |
| --- |
| Django中下面两个概念不同：  ①Project：工程/项目，例如：搜狐门户  ②Application/App：应用，例如：科技频道、旅游频道、房产频道  一个Project可以包含多个App；每个工程都有一个默认App，与工程同名 |

2.配置Django项目中的路由

提示：

**客户端路由**：Vue/React/Angular中“路由词典”指的是客户端浏览器中输入的地址，对应于哪个客户端中运行的组件模板；

**服务器端路由**：Django是服务器端框架，其中的“路由词典”指的是客户端发来的请求地址，该由哪个服务器端的方法来处理，最终返回的应该是HTTP响应消息。



#配置路由词典： 主APP/urls.py文件

urlpatterns = [

path( '路由地址', 处理方法 ),

...

path( '路由地址', 处理方法 ),

]

提示：

①路由处理方法在Django中称为“View”——MTV模式中的V

②路由处理方法必须接收一个req参数，返回一个HttpResponse对象

③路由地址可以在中间或尾部添加/；如login/、user/login；但是不应该在开头添加/，如/login

练习：创建一个路由地址register，服务器端路由向客户端输出：<h2>REGISTER SUCC</h2>；测试：路由地址开头/中间/结尾能否出现 /

注意：如果项目中有很多的路由，路由方法对应的就有很多，全部写在一个模块中不合适，官方推荐将它们划分到不同的App中，例如：App User、App Product、App Cart....

|  |
| --- |
| 创建新的Project： django-admin startproject 项目名称  创建新的App： python manage.py startapp 应用名称 |

练习：在项目中创建App：product，此App中建两个路由处理方法——即View方法：productList()、productDetail；并在路由词典中为它们分配路由地址

3.题外篇：如何安装VSCode插件

**方式1：在线安装**

Ctrl + Shift + X，输入框中搜索即可

**方式2：离线安装**

设法下载插件对应的安装文件 xxx.vsix

Ctrl + Shift + X，点击右上角的...按钮“更多操作” > 从VSIX安装即可

4.Django中的请求数据处理

**①读取客户端提交的QueryString请求数据**

请求消息：GET /user/login*?uname=dang&upwd=123*

读取请求数据：

n = req.**GET**.get('uname', 'guest')

p = req.**GET**.get('upwd', '') #有则使用，无则用默认

**②读取客户端提交的是请求主体中的数据**

请求消息：POST /user/register HTTP/1.1

....

*uname=dang&upwd=123*

读取请求数据：

n = req.**POST**.get('uname', 'guest')

p = req.**POST**.get('upwd', '') #有则使用，无则用默认

**③读取请求URL中的“路由参数”数据**

请求消息：GET /product/detail/*13*

GET /product/detail/*28*

路由配置：path( 'product/detail/<pid>', productDetail )

path( 'product/detail/<int:pid>', productDetail )

路由方法：def productDetail( req, **pid** ):

#pid参数值就是路由参数的值

----------------------------------------------------------------------

**④向客户端返回HTML响应消息**

res = HttpResponse('响应消息主体')

res['响应消息头'] = '值'

return res

**⑤向客户端返回JSON响应消息——前后端分离的项目**

res = JsonResponse( dict或者list ) #此处会自动执行JSON序列化

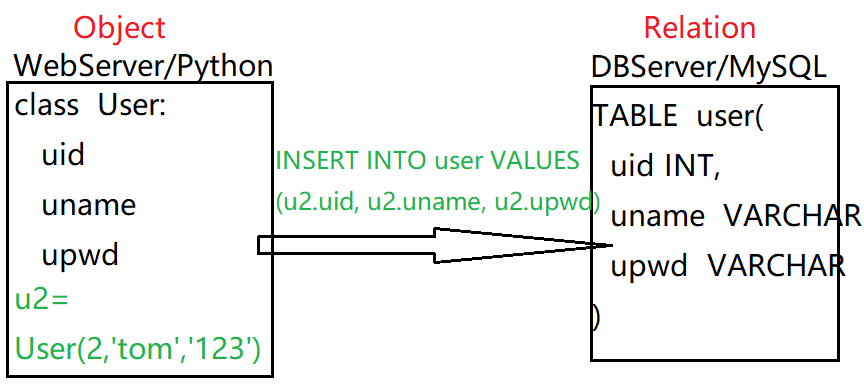
res['响应消息头'] = '值'

return res

5.Django中的ORM映射技术 —— 重点&难点

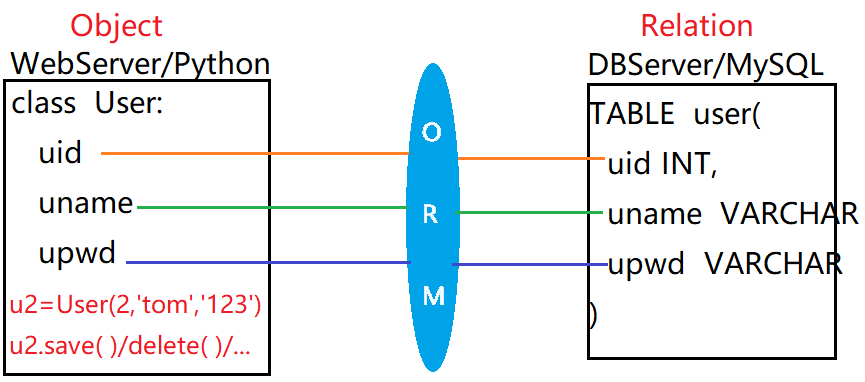
提示：一个框架采用ORM技术，那么开发者就无需编写任何SQL语句即可实现数据库操作。

传统的数据库操作存在的问题：



Python中数据是封装好的对象，但是关系型数据库中存储的却是离散的数据；保存数据就需要把封装好的对象“拆散”为离散的数据存储到关系型数据库；读取数据就是把离散的几个数据重新组装为对象。

解决方案：ORM（Object Relation Mapping）



ORM框架原理：ORM框架本身记录对象和关系表间的映射关系，以及属性和列之间的映射关系，用户只需要操作对象，ORM框架会自动生成需要SQL语句，并发送给数据库执行。即：

u2.save( ) #ORM会自动提交INSERT INTO....

u2.delete( ) #ORM会自动提交DELETE FROM....

......

项目中ORM可能呈现的方式有三种：

①用户创建Object Class，用户创建关系数据库中的表，用户再创建映射文件，指定class和table间的映射关系

**②正向工程**：用户创建Object Class，使用ORM检索这些类，自动生成创建表的SQL并发送给数据库执行

**③逆向工程**：用户创建数据库表结构，使用ORM检索这些表，自动生成每个表对应的Object Class

咱们的项目目前都已经写好了.sql文件并在数据库中执行了，直接使用方式3“逆向工程”即可生成每个表对应的Class

6.使用Django ORM技术生成类

①使用pip下载并安装MySQL数据库驱动模块(支持ORM技术)

pip install pymysql

pip install pymysql -i https://pypi.tuna.tsinghua.edu.cn/simple

②在当前项目主APP初始化时注册该驱动程序

#主APP/\_\_init\_\_.py

import pymysql

pymysql.install\_as\_MySQLdb( )

③在项目配置文件中声明MySQL服务器连接参数

#主APP/settings.py

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'xz',

'HOST': '127.0.0.1',

'PORT': '3306',

'USER': 'root',

'PASSWORD': '',

}

}

④启动MySQL数据库服务器，确保其中已有需要的关系表

⑤使用Django ORM技术检索数据库中的表，自动生成对应的Class

**python manage.py inspectdb** #默认生成的class会输出到命令行

**python manage.py inspectdb > ./user/models.py**  #把生成的class输出到指定的Model文件

|  |
| --- |
| 使用ORM检视MySQL数据库时可能报如下错误1：  *django.core.exceptions.ImproperlyConfigured: mysqlclient 1.3.13 or newer is required; you have 0.9.3.*  解决方案：取消数据库驱动版本检查；编辑下面文件  C:\Users\web\AppData\Local\Programs\Python\Python38-32\lib\site-packages\django\db\backends\mysql\base.py, 注释掉其中的第35/36行 |
| 使用ORM检视MySQL数据库时可能报如下错误2：  *AttributeError: 'str' object has no attribute 'decode'*  解决方案：把Python3已经删除的方法decode改为encode  C:\Users\web\AppData\Local\Programs\Python\Python38-32\lib\site-packages\django\db\backends\mysql\operations.py，把第146中的decode改为encode |

课后任务：

(1)使用Django ORM技术，根据mfresh(净美仕)项目的关系型数据库表结构，自动创建对应的模型类——净美仕必需的.sql文件在“mfresh项目资料.zip”

(2)“净美仕官网”项目实现步骤

**旧有系统：**www.codeboy.com/mfresh/mfresh.zip 静态页面+PHP整合项目

**系统重构：**前后端分离：前端子系统(Vue.js)、后端子系统(Python/Django)

**<<<<<<<<<<<后端子系统实现步骤：>>>>>>>>>>>>>**

一、创建一个空白Django项目，并保证可以运行

1.1 下载django模块

pip install django

pip install django -i http://pypi.douban.com/simple

1.2 运行django工具，创建空白项目

django-admin startproject mfresh\_api

1.3 进入空白项目，运行起来

cd mfresh\_api

python manage.py runserver

可以使用浏览器访问该项目 http://127.0.0.1:8000

二、创建子App以及路由系统

2.1 创建必需的子App

python manage.py startapp user

python manage.py startapp news

python manage.py startapp product

python manage.py startapp cart

2.2 在主App中注册所有的子App

#编辑主App/settings.py/INSTALLED\_APPS

#添加上述四个子App

INSTALLED\_APPS = [

'user',

'news',

'product',

'cart',

.....系统原有的.....

]

2.3 在每个子App下创建路由方法(Django称为Views)

#创建user App下的路由方法——编辑user/views.py，创建路由方法：

def userLogin( req ):

def userRegister( req ):

def userCheckUname( req ):

def userCheckPhone( req ):

#类似的，创建news App下的路由方法——编辑news/views.py，创建方法：

#类似的，创建product App下的路由方法——编辑product/views.py，创建方法：

#类似的，创建cart App下的路由方法——编辑cart/views.py，创建方法：

2.4 为每个路由方法分配路由地址

#编辑主App/urls.py，添加更多的路由设置

from user import views as uv

urlpatterns = [

path('user/login', uv.userLogin),

path('user/register', uv.userRegister),

path('user/check/uname', uv.userCheckUname),

path('user/check/phone', uv.userCheckPhone),

]

#类似的，再添加news/product/cart三个App下的路由设置

三、使用ORM实现每个路由方法中的数据库操作

3.1 启动MySQL数据库，在其中创建好mfresh库及所有表

3.2 下载并安装MySQL数据库连接驱动：pymysql

pip uninstall pymysql #如果已经安装了老版本的，先卸载

pip install pymysql #如果报错，使用豆瓣网的仓库

pip install pymysql -i http://pypi.douban.com/simple

3.3 在主App中注册使用pymysql提供的数据库驱动

#编辑主App/\_\_init\_\_.py文件

import pymysql

pymysql.install\_as\_MySQLdb( )

3.4 在主App中指定MySQL服务器的连接参数

#编辑主App/settings.py文件：DATABASES

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'mfresh',

'HOST': '127.0.0.1',

'PORT': 3306,

'USER': 'root',

'PASSWORD': '',

}

}

3.5 使用ORM反向生成工具，自动根据数据库表结构创建对应的Object

#查验数据库中的表结构，生成每个表对应的Object对象

python manage.py inspectdb #生成的class会输出到控制台

python manage.py inspectdb > user/models.py #生成的class输出到控指定的Model文件

|  |
| --- |
| 此步骤会遇到Python3和Django2.2不兼容的问题，可以把最后的错误提示粘贴到百度，查找解决方案：  ①修改下述文件：C:\Users\web\AppData\Local\Programs\Python\Python36-32\lib\site-packages\django\db\backends\mysql\base.py", **注释掉其中的第35/36两行代码**  ②继续修改下述文件：C:\Users\web\AppData\Local\Programs\Python\Python36-32\lib\site-packages\django\db\backends\mysql\operations.py，**将其中第146中的decode函数改为encode函数** |

3.6 使用Django QuerySet提供的API操作模型对象的增删改查，实现底层数据库的CRUD：

查询所有：MfUser.objects.**values('uname',...)**

查询一个：MfUser.objects.**filter(uid=3)**.values('uname',...)

添加新的：MfUser.objects.**create(uname='..', upwd='..')**

删除数据：MfUser.objects.filter(uid=4)**.delete( )**

更新数据：MfUser.objects.filter(uid=4)**.update(uname='..', upwd='..')**

**<<<<<<<<<<<前端子系统实现步骤：>>>>>>>>>>>>>**

一、使用Vue-CLI脚手架创建空白项目并启动

1.1 下载Vue-CLI全局脚手架工具

npm i -g @vue/cli

1.2 运行脚手架工具，创建空白项目

vue create mfresh\_fe

1.3 启动空白项目

cd mfresh\_fe

npm i axios

npm run serve

测试访问：http://127.0.0.1:8080

二、创建路由组件，并配置路由词典

2.1 创建路由组件

2.2 为每个路由组件分配路由地址

三、详细编写每个路由组件及其子组件，获取数据，并实现数据绑定