01-Django快速入门

Django官网：

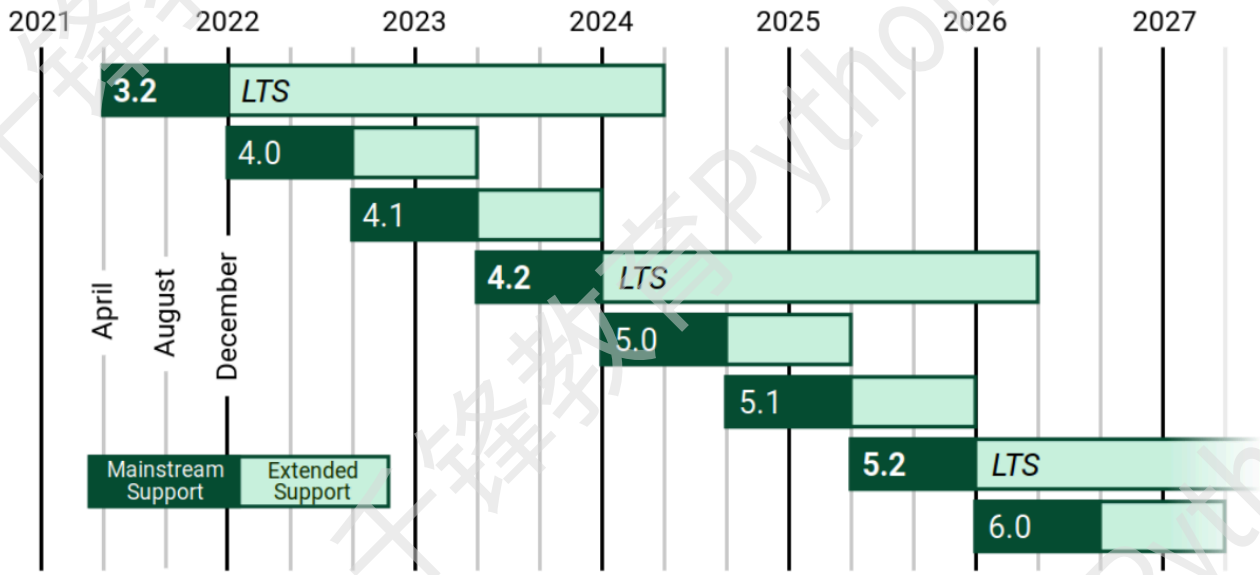
https://docs.djangoproject.com/en/4.2/

Django是一个开放源代码的Web应用框架，由Python写成。采用了MTV的框架模式，即模型M，视图V和模版T。它最初是被开发来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的，即是CMS（内容管理系统）软件。并于2005年7月在BSD许可证下发布。这套框架是以比利时的吉普赛爵士吉他手Django Reinhardt来命名的。2020年，Django 3. 0发布。

2022年，Django 4. 0发布。Django 4.0 将支持 Python 3.8、3.9 与 3.10

2023年，Django4.2LTS发布，长期支持版本，会维护3年，在企业中我们建议使用LTS版本。

Django每2年会推出全新的大版本。



1. 创建虚拟环境(virtualenv 和virtualenvwrapper)

创建虚拟环境(复习)

```
# 先打开cmd

# 安装virtualenv （windows操作系统）
pip install virtualenv virtualenvwrapper-win

# workon 查看虚拟环境
workon

# mkvirtualenv 创建新的虚拟环境
```

```
mkvirtualenv env

# rmvirtualenv 删除虚拟环境
# rmvirtualenv env

# 进入虚拟环境
workon env
```

创建虚拟环境

```
# 创建虚拟环境
mkvirtualenv django4env

# 进入虚拟环境
workon django4env
```

2. 安装django

pip install django==4.2 -i https://pypi.douban.com/simple

（4.2是LTS长期支持版本, 推荐使用， 如不写则会安装最新版本）

```
测试Django是否安装成功
pip show django
```

3. 创建一个Django项目

- 方式一: 进入到指定要存放项目的目录，执行 **django-admin startproject HelloDjango** 来创建一个名字为HelloDjango 的工程
- 方式二: 使用Pycharm专业版创建Django项目.

```
创建项目后，默认的目录结构：
manage.py：
    · 是Django用于管理本项目的命令行工具，之后进行站点运行，数据库自动生成等都是通过本文件完成。

HelloDjango/__init__.py：
  · 告诉python该目录是一个python包，暂无内容，后期一些工具的初始化可能会用到

HelloDjango/settings.py：
    · Django项目的配置文件，默认状态其中定义了本项目引用的组件，项目名，数据库，静态资源等。

HelloDjango/urls.py：
    · 维护项目的URL路由映射，即定义当客户端访问时由哪个模块进行响应。

HelloDjango/wsgi.py：
    · 全称为Python Web Server Gateway Interface，即Python服务器网关接口，是Python应用与Web服务器之间的接口，用于Django项目在服务器上的部署和上线，一般不需要修改。
```

```
HelloDjango/asgi.py:
    · 定义ASGI的接口信息，和WSGI类似，在3.0以后新增ASGI，相比WSGI，ASGI实现了异步处理，用于启动异步通信服务，比如:实现在线聊天等异步通信功能。(类似Tornado异步框架)
```

4. 测试服务器的启动

python manage.py runserver [ip:port]

```
可以直接进行服务运行 默认执行起来的端口是8000

也可以自己指定ip和端口:
1. 监听机器所有可用 ip （电脑可能有多个内网ip或多个外网ip）：
        python manage.py runserver 0.0.0.0:8000
2. 同时在settings.py中将
        ALLOWED_HOSTS=['*']

3. 在其他局域网电脑上可以通过在浏览器输入 Django项目所在电脑的 IP:8000 来访问
```

5.数据迁移

迁移的概念: 就是将模型映射到数据库的过程

生成迁移文件: python manage.py makemigrations

执行迁移: python manage.py migrate

> 不需要初始化迁移文件夹，每个应用默认有迁移文件夹migrations

6.创建应用

python manage.py startapp App

创建名称为App的应用

使用应用前需要将应用配置到项目中，在settings.py中将应用加入到INSTALLED_APPS选项中

```
应用目录介绍:
__init__.py:
    其中暂无内容，使得app成为一个包
admin.py:
    管理站点模型的声明文件，默认为空
apps.py:
    应用信息定义文件，在其中生成了AppConfig，该类用于定义应用名等数据
models.py:
    添加模型层数据类文件
views.py:
    定义URL相应函数,视图函数
migrations包:
    自动生成，生成迁移文件的
```

```
tests.py:
    测试代码文件
```

7.基本视图

```
# 首先我们在views.py中建立一个路由响应函数
from django.http import HttpResponse
def welcome(request):
  return HttpResponse('HelloDjango');


# 接着我们在urls中进行注册
# 1. 直接访问视图
path(r'hello/', hello, name='hello'),

# 2. 导入App中的子路由urls.py文件
path('app/', include('App.urls'))

# 3.使用命名空间（后面讲解）
path('app/', include(('App.urls', 'App'), namespace='App'))


# 子路由写法如下:
urlpatterns = [
    # django1.8, 2.0正则表达式写法：不再使用,不推荐
    # url(r'^index/$', index),

    # Djangov2.0, v3.0, v4.0写法：常用
    path(r'hello/', hello, name='hello'),
]
```

8.基本模板

```
模板实际上就是我们用HTML写好的页面
创建模板文件夹templates，在模板文件夹中创建模板文件
在views中去加载渲染模板，使用render函数：return render(request,'index.html')
```

9.定义模型

在models.py 中引入models
from django.db import models

创建自己的模型类，但切记要继承自 models.Model

案例驱动: 使用模型定义班级，并在模板上显示班级列表

```
# 班级table : grade
#   columns:
班级名称 - name
成立时间 - date
女生个数 - girlnum
男生个数 - boynum
是否删除 - is_delete
```

10.Admin 后台管理

在admin.py中将model加入后台管理：

```
admin.site.register(Grade)
```

创建超级用户：**python manage.py createsuperuser**

访问admin后台：http://127.0.0.1:8000/admin/

11.展示班级列表

在views.py文件中编写班级的视图函数：

```
def grade_list(request):
    # 获取班级所有数据
    g_list = Grade.objects.all()
    return render(request, 'grade/grade_list.html', {'g_list': g_list})
```

模板文件(html文件)：

```
{% for grade in g_list %}
  {{ grade.sname }}
{% endfor %}
```

12.配置url

在grade App中新建urls.py文件，输入如下代码：

```
from django.conf.urls import url
from .views import grade_list

urlpatterns = [
  path(r'grade/', grade_list),
]
```

在工程的urls.py文件中添加如下代码：

```
path(r'/', include('grade.urls')),
```