Django 快速参考

模型、视图、模板、ORM、表单、管理后台、认证

项目初始化

创建项目与应用

```
pip install django
django-admin startproject mysite
cd mysite
python manage.py startapp blog
```

常用命令

runserver	在 8000 端口启动开发服务器
makemigrations	根据模型变更生成迁移文件
migrate	将迁移应用到数据库
createsuperuser	创建管理员超级用户
shell	带 Django 上下文的交互式 Python shell
test	运行测试套件

项目结构

manage.py	CLI 入口
settings.py	项目配置
urls.py	根 URL 配置
wsgi.py / asgi.py	服务器入口
apps/models.py	数据库模型
apps/views.py	请求处理器

模型

定义模型

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=200)
    body = models.TextField()
    created = models.DateTimeField(auto_now_add=True)
    published = models.BooleanField(default=False)
```

字段类型

CharField(max_length=N)	短文本（必须指定 max_length）
TextField()	长文本（无长度限制）
IntegerField()	整数
FloatField()	浮点数
BooleanField()	True / False
DateTimeField()	日期与时间
EmailField()	带验证的邮箱
FileField(upload_to='')	文件上传

关联关系

```
author = models.ForeignKey(
    User, on_delete=models.CASCADE
)
tags = models.ManyToManyField(Tag, blank=True)
profile = models.OneToOneField(User, on_delete=models.CASCADE)
```

Meta 与方法

```
class Meta:
    ordering = ['-created']
    verbose_name_plural = 'posts'

def __str__(self):
    return self.title
```

视图

函数视图

```
from django.shortcuts import render, get_object_or_404

def post_list(request):
    posts = Post.objects.filter(published=True)
    return render(request, 'blog/list.html', {'posts': posts})
```

类视图

```
from django.views.generic import ListView, DetailView

class PostListView(ListView):
    model = Post
    template_name = 'blog/list.html'
    context_object_name = 'posts'
    paginate_by = 10
```

常用类视图

ListView	显示对象列表
DetailView	显示单个对象
CreateView	创建对象的表单
UpdateView	编辑对象的表单
DeleteView	确认并删除对象
TemplateView	渲染模板（无模型）

JSON 响应

```
from django.http import JsonResponse

def api_posts(request):
    data = list(Post.objects.values('id', 'title'))
    return JsonResponse(data, safe=False)
```

模板

模板语法

```
{{ variable }}
{{ post.title|truncatewords:30 }}
{% if user.is_authenticated %}
  <p>Welcome, {{ user.username }}!</p>
{% endif %}
```

循环与条件

```
{% for post in posts %}
  <h2>{{ post.title }}</h2>
  {% if forloop.last %}<hr>{% endif %}
{% empty %}
  <p>No posts yet.</p>
{% endfor %}
```

模板继承

```
{# base.html #}
<html>
<body>{% block content %}{% endblock %}</body>
</html>

{# child.html #}
{% extends "base.html" %}
{% block content %}<h1>Hello</h1>{% endblock %}
```

常用过滤器

|date:"Y-m-d"	格式化日期
|default:"N/A"	空值时的回退
|length	统计列表元素数量
|truncatewords:N	限制为 N 个词
|safe	标记为安全 HTML（不转义）
|slugify	URL 安全的小写字符串

URL

URL 配置

```
from django.urls import path, include

urlpatterns = [
    path('', views.index, name='index'),
    path('post/<int:pk>/', views.detail, name='detail'),
    path('blog/', include('blog.urls')),
]
```

路径转换器

<int:pk>	整数（如 42）
<str:slug>	不含斜杠的字符串
<slug:slug>	slug（字母、数字、连字符）
<uuid:id>	UUID 格式
<path:rest>	含斜杠的完整路径

反向解析 URL

```
from django.urls import reverse
url = reverse('detail', kwargs={'pk': 1})
# In templates: {% url 'detail' pk=post.pk %}
```

表单

ModelForm

```
from django import forms

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ['title', 'body', 'published']
```

在视图中处理表单

```
def create_post(request):
    form = PostForm(request.POST or None)
    if form.is_valid():
        post = form.save(commit=False)
        post.author = request.user
        post.save()
        return redirect('detail', pk=post.pk)
    return render(request, 'blog/form.html', {'form': form})
```

模板中的表单

```
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</button>
</form>
```

验证

```
def clean_title(self):
    title = self.cleaned_data['title']
    if len(title) < 5:
        raise forms.ValidationError("Title too short.")
    return title
```

Django 快速参考

管理后台

注册模型

```
from django.contrib import admin
from .models import Post

@admin.register(Post)
class PostAdmin(admin.ModelAdmin):
    list_display = ['title', 'author', 'created', 'published']
    list_filter = ['published', 'created']
    search_fields = ['title', 'body']
```

Admin 选项

list_display	列表视图的列
list_filter	侧边栏过滤选项
search_fields	可搜索的字段
prepopulated_fields	自动填充（如从标题生成 slug）
readonly_fields	后台不可编辑的字段
ordering	默认排序

ORM 查询

基础查询

```
Post.objects.all()                # all records
Post.objects.get(pk=1)            # single (raises if missing)
Post.objects.filter(published=True) # queryset
Post.objects.exclude(draft=True)  # exclude matches
Post.objects.count()              # total count
```

字段查找

field__exact	精确匹配（默认）
field__icontains	不区分大小写包含
field__gt / __lt	大于 / 小于
field__gte / __lte	大于等于 / 小于等于
field__in=[1,2,3]	值在列表中
field__isnull=True	为 NULL
field__startswith	以字符串开头
field__range=(a,b)	在 a 和 b 之间（含）

链式与聚合

```
from django.db.models import Q, Count, Avg

Post.objects.filter(
    Q(title__icontains='django') | Q(body__icontains='django')
).order_by('-created')[:10]

Post.objects.aggregate(avg_views=Avg('views'))
```

增删改

```
post = Post.objects.create(title='New', body='...')
post.title = 'Updated'
post.save()
Post.objects.filter(draft=True).update(published=False)
post.delete()
```

认证

登录 / 登出

```
from django.contrib.auth import authenticate, login, logout

user = authenticate(request, username='admin', password='pw')
if user is not None:
    login(request, user)
```

保护视图

```
from django.contrib.auth.decorators import login_required

@login_required
def dashboard(request):
    return render(request, 'dashboard.html')
```

认证 URL

```
# urls.py
path('accounts/', include('django.contrib.auth.urls'))
# Provides: login, logout, password_change, password_reset
```

模板中的认证

```
{% if user.is_authenticated %}
  <p>Hi, {{ user.username }}</p>
  <a href="{% url 'logout' %}">Logout</a>
{% else %}
  <a href="{% url 'login' %}">Login</a>
{% endif %}
```

配置

关键配置项

DEBUG	开发用 **True**，生产用 **False**
ALLOWED_HOSTS	允许的主机名列表
SECRET_KEY	加密签名密钥（保密）
DATABASES	数据库引擎、名称、主机、凭证
INSTALLED_APPS	已注册的应用列表
STATIC_URL	静态文件 URL 前缀
MEDIA_URL / MEDIA_ROOT	用户上传文件路径

数据库配置

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'mydb',
        'USER': 'dbuser',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

静态文件

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / 'static']
# In templates: {% load static %}
# <link href="{% static 'css/style.css' %}">
```