

# 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

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

### 项目结构

<b>manage.py</b>	CLI 入口
<b>settings.py</b>	项目配置
<b>urls.py</b>	根 URL 配置
<b>wsgi.py / asgi.py</b>	服务器入口
<b>apps/models.py</b>	数据模型
<b>apps/views.py</b>	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

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

<b>ListView</b>	显示对象列表
<b>DetailView</b>	显示单个对象
<b>CreateView</b>	创建对象的表单
<b>UpdateView</b>	编辑对象的表单
<b>DeleteView</b>	确认并删除对象
<b>TemplateView</b>	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

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

## 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

<b>&lt;int:pk&gt;</b>	整数 (如 42)
<b>&lt;str:slug&gt;</b>	不含斜杠的字符串
<b>&lt;slug:slug&gt;</b>	slug (字母、数字、连字符)
<b>&lt;uuid:id&gt;</b>	UUID 格式
<b>&lt;path:rest&gt;</b>	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
```

## 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

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

<b>field__exact</b>	精确匹配 (默认)
<b>field__icontains</b>	不区分大小写包含
<b>field__gt / __lt</b>	大于 / 小于
<b>field__gte / __lte</b>	大于等于 / 小于等于
<b>field__in=[1,2,3]</b>	值在列表中
<b>field__isnull=True</b>	为 NULL
<b>field__startswith</b>	以字符串开头
<b>field__range=(a,b)</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')[1:10]

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

<b>DEBUG</b>	开发用 `True`，生产用 `False`
<b>ALLOWED_HOSTS</b>	允许的主机名列表
<b>SECRET_KEY</b>	加密签名密钥 (保密)
<b>DATABASES</b>	数据库引擎、名称、主机、凭证
<b>INSTALLED_APPS</b>	已注册的应用列表
<b>STATIC_URL</b>	静态文件 URL 前缀
<b>MEDIA_URL / MEDIA_ROOT</b>	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' %}">
```