

REFERÊNCIA RÁPIDA DJANGO

Models, views, templates, ORM, formulários, admin, autenticação

Configuração do Projeto

Criar Projeto e App

```
pip install django
django-admin startproject mysite
cd mysite
```

```
python manage.py startapp blog
```

Comandos Comuns

runserver

Iniciar servidor de desenvolvimento na porta 8000

makemigrations Gerar arquivos de migração a partir de alterações no modelo

migrate Aplicar migrações ao banco de dados

createsuperuser Criar superusuário admin

shell Shell Python interativo com Django

test Executar suite de testes

Estrutura do Projeto

manage.py Ponto de entrada CLI
settings.py Configuração do projeto
urls.py Configuração de URL raiz
wsgi.py / asgi.py Pontos de entrada do servidor
apps/models.py Modelos de banco de dados
apps/views.py Handlers de requisição

Models

Definir um Model

```
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)
```

Tipos de Campo

CharField(max_length=N) Texto curto (max_length obrigatório)
TextField() Texto longo (sem limite)
IntegerField() Valor inteiro
FloatField() Número de ponto flutuante
BooleanField() True / False
DateTimeField() Data e hora
EmailField() Email com validação
FileField(upload_to='') Upload de arquivo

Relacionamentos

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

Meta e Métodos

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

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

Views

View Baseada em Função

```
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})
```

Views Baseadas em Classe

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

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

CBVs Comuns

ListView Exibir lista de objetos
DetailView Exibir objeto único
CreateView Formulário para criar objeto
UpdateView Formulário para editar objeto
DeleteView Confirmar e excluir objeto
TemplateView Renderizar template (sem model)

Resposta JSON

```
from django.http import JsonResponse
```

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

Templates

Sintaxe de Template

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

Laços e Condicionais

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

Herança de Template

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

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

Filtros Comuns

|date:"Y-m-d" Formatar data
|default:"N/A" Fallback para valores vazios
|length Contar itens na lista
|truncatewords:N Limitar a N palavras
|safe Marcar como HTML seguro (sem escape)
|slugify String em letras minúsculas segura para URL

URLs

Padrões de URL

```
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')),
]
```

Conversores de Caminho

<int:pk> Inteiro (ex.:42)
<str:slug> String sem barras
<slug:slug> Slug (letras, números, hífens)
<uuid:id> Formato UUID
<path:rest> Caminho completo incluindo barras

URLs Reversas

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

Formulários

Model Form

```
from django import forms

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

Processar Formulário na View

```
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})
```

Formulário no Template

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

Validação

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

Admin

Registrar Model

```
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']
```

Opções do Admin

list_display Colunas na visualização em lista
list_filter Opções de filtro na barra lateral
search_fields Campos pesquisáveis
prepopulated_fields Auto-preencher (ex: slug a partir do título)
readonly_fields Não editável no admin
ordering Ordenação padrão

Queries ORM

Queries Básicas

```
Post.objects.all() # todos os registros
Post.objects.get(pk=1) # único (lança exceção se ausente)
Post.objects.filter(published=True) # queryset
Post.objects.exclude(draft=True) # excluir correspondências
Post.objects.count() # total
```

Lookups de Campo

field_exact Correspondência exata (padrão)
field_icontains Contém (case-insensitive)
field_gt / --lt Maior que / menor que
field_gte / --lte Maior/menor ou igual
field_in=[1,2,3] Valor na lista
field_isnull=True É NULL
field_startswith Começa com string
field__range=(a,b) Entre a e b inclusive

Encadeamento e Agregação

```
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'))
```

Criar, Atualizar, Excluir

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

Autenticação

Login / Logout

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

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

Proteger Views

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

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

URLs de Auth

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

Auth no Template

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

Configurações

Configurações Principais

DEBUG `True` para dev, `False` para produção
ALLOWED_HOSTS Lista de hostnames válidos
SECRET_KEY Chave de assinatura criptográfica (manter secreta)
DATABASES Engine, nome, host e credenciais do banco
INSTALLED_APPS Lista de apps registrados
STATIC_URL Prefixo de URL para arquivos estáticos
MEDIA_URL / MEDIA_ROOT Caminhos para arquivos enviados por usuários

Configuração de Banco de Dados

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

Arquivos Estáticos

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