

Django クイックリファレンス

モデル、ビュー、テンプレート、ORM、フォーム、管理画面、認証

プロジェクトのセットアップ

プロジェクトとアプリの作成

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

よく使うコマンド

runserver	ポート 8000 で開発サーバーを起動
makemigrations	モデル変更からマイグレーションファイルを生成
migrate	データベースにマイグレーションを適用
createsuperuser	管理者スーパーユーザーを作成
shell	Django 付きの対話型 Python シェル
test	テストスイートを実行

プロジェクト構造

manage.py	CLI エントリーポイント
settings.py	プロジェクト設定
urls.py	ルート URL 設定
wsgi.py / asgi.py	サーバーエントリーポイント
apps/models.py	データベースモデル
apps/views.py	リクエストハンドラ

モデル

モデルの定義

```
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)
```

フィールドの種類

CharField(max_length=N)	短いテキスト (max_length 必須)
TextField()	長いテキスト (制限なし)
IntegerField()	整数値
FloatField()	浮動小数点数
BooleanField()	True / False
DateTimeField()	日付と時刻
EmailField()	バリデーション付きメールアドレス
FileField(upload_to='')	ファイルアップロード

リレーションシップ

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

Meta とメソッド

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

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

ビュー

関数ベースビュー

```
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})
```

クラスベースビュー

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

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

よく使う CBV

ListView	オブジェクトのリストを表示
DetailView	単一オブジェクトを表示
CreateView	オブジェクト作成フォーム
UpdateView	オブジェクト編集フォーム
DeleteView	確認してオブジェクトを削除
TemplateView	テンプレートをレンダリング (モデルなし)

JSON レスポンス

```
from django.http import JsonResponse

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

テンプレート

テンプレート構文

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

ループと条件分岐

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

テンプレート継承

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

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

よく使うフィルター

 date:"Y-m-d"	日付のフォーマット
 default:"N/A"	空の値のフォールバック
 length	リストの件数をカウント
 truncatewords:N	N 単語に制限
 safe	安全な HTML としてマーク (エスケープなし)
 slugify	URL 安全な小文字文字列

URL

URL パターン

```
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')),
]
```

パスコンバーター

<int:pk>	整数 (例: 42)
<str:slug>	スラッシュを含まない文字列
<slug:slug>	スラッグ (文字、数字、ハイフン)
<uuid:id>	UUID 形式
<path:rest>	スラッシュを含む完全なパス

URL の逆引き

```
from django.urls import reverse
url = reverse('detail', kwargs={'pk': 1})
# テンプレートで: {% url 'detail' pk=post.pk %}
```

フォーム

モデルフォーム

```
from django import forms

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

ビューでフォームを処理

```
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})
```

テンプレートでのフォーム

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

バリデーション

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

Django クイックリファレンス

管理画面

モデルの登録

```
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']
```

管理画面オプション

list_display	一覧ビューのカラム
list_filter	サイドバーのフィルターオプション
search_fields	検索可能なフィールド
prepopulated_fields	自動入力 (例: タイトルからスラッグ)
readonly_fields	管理画面で編集不可
ordering	デフォルトのソート順

ORM クエリ

基本クエリ

```
Post.objects.all() # すべてのレコード
Post.objects.get(pk=1) # 1件 (存在しない場合は例外)
Post.objects.filter(published=True) # クエリセット
Post.objects.exclude(draft=True) # 一致を除外
Post.objects.count() # 合計件数
```

フィールドルックアップ

field__exact	完全一致 (デフォルト)
field__icontains	大文字小文字を区別しない部分一致
field__gt / __lt	より大きい / より小さい
field__gte / __lte	以上 / 以下
field__in=[1,2,3]	リスト内の値
field__isnull=True	NULL である
field__startswith	文字列で始まる
field__range=(a,b)	a 以上 b 以下

チェーンと集計

```
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'))
```

作成、更新、削除

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

認証

ログイン / ログアウト

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

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

ビューの保護

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

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

認証 URL

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

テンプレートでの認証

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

設定

主要な設定

DEBUG	開発時は True 、本番は False
ALLOWED_HOSTS	有効なホスト名のリスト
SECRET_KEY	暗号署名キー (秘密に保つ)
DATABASES	DB エンジン、名前、ホスト、認証情報
INSTALLED_APPS	登録済みアプリのリスト
STATIC_URL	静的ファイルの URL プレフィックス
MEDIA_URL / MEDIA_ROOT	ユーザーアップロードファイルのパス

データベース設定

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

静的ファイル

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / 'static']
# テンプレートで: {% load static %}
# <link href="{% static 'css/style.css' %}">
```