REFERENSI CEPAT DJANGO

Model, view, template, ORM, form, admin, autentikasi

Setup Proyek

Buat Proyek & App

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

Perintah Umum

`runserver`	Jalankan dev server di port 8000
`makemigrations`	Buat file migrasi dari perubahan model
`migrate`	Terapkan migrasi ke database
`createsuperuser`	Buat admin superuser
`shell`	Shell Python interaktif dengan Django
`test`	Jalankan test suite

Struktur Proyek

`manage.py`	Titik masuk CLI
`settings.py`	Konfigurasi proyek
`urls.py`	Konfigurasi URL root
`wsgi.py / asgi.py`	Titik masuk server
`apps/models.py`	Model database
`apps/views.py`	Handler request

Model

Definisi 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)
```

Tipe Field

`CharField(max_length=N)`	Teks pendek (wajib max_length)
`TextField()`	Teks panjang (tanpa batas)
`IntegerField()`	Nilai integer
`FloatField()`	Angka floating point
`BooleanField()`	True / False
`DateTimeField()`	Tanggal dan waktu
`EmailField()`	Email dengan validasi
`FileField(upload_to='')`	Upload file

Relasi

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

Meta & Method

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

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

View

Function-Based View

```
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})
```

Class-Based View

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

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

CBV Umum

`ListView`	Tampilkan daftar objek
`DetailView`	Tampilkan satu objek
`CreateView`	Form untuk membuat objek
`UpdateView`	Form untuk mengedit objek
`DeleteView`	Konfirmasi dan hapus objek
`TemplateView`	Render template (tanpa model)

JSON Response

```
from django.http import JsonResponse

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

Template

Sintaks Template

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

Loop & Kondisi

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

Pewarisan Template

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

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

Filter Umum

`|date:"Y-m-d"`	Format tanggal
`|default:"N/A"`	Fallback untuk nilai kosong
`|length`	Hitung item dalam list
`|truncatewords:N`	Batasi ke N kata
`|safe`	Tandai sebagai HTML aman (tanpa escaping)
`|slugify`	String huruf kecil yang aman untuk URL

URL

Pola 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')),
]
```

Converter Path

`<int:pk>`	Integer (mis. 42)
`<str:slug>`	String tanpa slash
`<slug:slug>`	Slug (huruf, angka, tanda hubung)
`<uuid:id>`	Format UUID
`<path:rest>`	Path lengkap termasuk slash

URL Terbalik

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

Form

Model Form

```
from django import forms

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

Proses Form di 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})
```

Form di Template

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

Validasi

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

Admin

Daftarkan 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']
```

Opsi Admin

`list_display`	Kolom dalam tampilan list
`list_filter`	Opsi filter sidebar
`search_fields`	Field yang bisa dicari
`prepopulated_fields`	Isi otomatis (mis. slug dari title)
`readonly_fields`	Tidak bisa diedit di admin
`ordering`	Urutan sort default

Query ORM

Query Dasar

```
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
```

Field Lookup

`field__exact`	Kecocokan tepat (default)
`field__icontains`	Mengandung (case-insensitive)
`field__gt / __lt`	Lebih besar / lebih kecil dari
`field__gte / __lte`	Lebih besar/kecil atau sama dengan
`field__in=[1,2,3]`	Nilai ada dalam daftar
`field__isnull=True`	Bernilai NULL
`field__startswith`	Dimulai dengan string
`field__range=(a,b)`	Antara a dan b (inklusif)

Chaining & Agregasi

```
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'))
```

Buat, Perbarui, Hapus

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

Autentikasi

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)
```

Lindungi View

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

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

URL Auth

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

Auth di Template

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

Pengaturan

Pengaturan Utama

`DEBUG`	`True` untuk dev, `False` untuk produksi
`ALLOWED_HOSTS`	Daftar hostname yang valid
`SECRET_KEY`	Kunci signing kriptografis (jaga kerahasiaannya)
`DATABASES`	Engine DB, nama, host, kredensial
`INSTALLED_APPS`	Daftar app yang terdaftar
`STATIC_URL`	Prefix URL untuk file statis
`MEDIA_URL / MEDIA_ROOT`	Path file yang diunggah pengguna

Konfigurasi Database

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

File Statis

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