

DJANGO QUICK REFERENCE

Models, views, templates, ORM, forms, admin, auth

Project Setup

Create Project & App

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

Common Commands

runserver	Start dev server on port 8000
makemigrations	Generate migration files from model changes
migrate	Apply migrations to database
createsuperuser	Create admin superuser
shell	Interactive Python shell with Django
test	Run test suite

Project Structure

manage.py	CLI entry point
settings.py	Project configuration
urls.py	Root URL configuration
wsgi.py / asgi.py	Server entry points
apps/models.py	Database models
apps/views.py	Request handlers

Models

Define a 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)
```

Field Types

CharField(max_length=N)	Short text (required max_length)
TextField()	Long text (no limit)
IntegerField()	Integer value
FloatField()	Floating point number
BooleanField()	True / False
DateTimeField()	Date and time
EmailField()	Email with validation
FileField(upload_to='')	File upload

Relationships

```
author = models.ForeignKey(
    User, on_delete=models.CASCADE
)

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

Meta & Methods

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

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

Views

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 Views

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

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

Common CBVs

ListView	Display list of objects
DetailView	Display single object
CreateView	Form to create object
UpdateView	Form to edit object
DeleteView	Confirm and delete object
TemplateView	Render a template (no model)

JSON Response

```
from django.http import JsonResponse

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

Templates

Template Syntax

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

Loops & Conditions

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

Template Inheritance

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

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

Common Filters

 date:"Y-m-d"	Format date
 default:"N/A"	Fallback for empty values
 length	Count items in list
 truncatewords:N	Limit to N words
 safe	Mark as safe HTML (no escaping)
 slugify	URL-safe lowercase string

URLs

URL Patterns

```
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')),
]
```

Path Converters

<int:pk>	Integer (e.g., 42)
<str:slug>	String without slashes
<slug:slug>	Slug (letters, numbers, hyphens)
<uuid:uid>	UUID format
<path:rest>	Full path including slashes

Reverse URLs

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

Forms

Model Form

```
from django import forms

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

Process Form in 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 in Template

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

Validation

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

Admin

Register 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']
```

Admin Options

list_display	Columns in list view
list_filter	Sidebar filter options
search_fields	Searchable fields
prepopulated_fields	Auto-fill (e.g., slug from title)
readonly_fields	Non-editable in admin
ordering	Default sort order

ORM Queries

Basic Queries

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 Lookups

field__exact	Exact match (default)
field__icontains	Case-insensitive contains
field__gt / __lt	Greater / less than
field__gte / __lte	Greater/less than or equal
field__in=[1,2,3]	Value in list
field__isnull=True	is NULL
field__startswith	Starts with string
field__range=(a,b)	Between a and b inclusive

Chaining & Aggregation

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

Post.objects.filter(
    Q(title__icontains='django') | Q(body__icontains='django')
).order_by('-created')[0:10]

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

Create, Update, Delete

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

Authentication

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)
```

Protect Views

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

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

Auth URLs

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

Template Auth

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

Settings

Key Settings

DEBUG	True for dev, False for production
ALLOWED_HOSTS	List of valid hostnames
SECRET_KEY	Cryptographic signing key (keep secret)
DATABASES	DB engine, name, host, credentials
INSTALLED_APPS	Registered apps list
STATIC_URL	URL prefix for static files
MEDIA_URL / MEDIA_ROOT	User-uploaded file paths

Database Config

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

Static Files

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