Django Multi App Project Structure

Multiple Django apps with separated concerns. Standard approach for medium-sized projects.

#django #python #web #backend #modular

PNG PDF Copy Prompt

Project Directory

```
myproject/
    manage.py
  config/ Project configu…
    __init__.py
  settings/ Split by enviro…
      __init__.py
      base.py Shared settings
      local.py Dev overrides
      production.py Prod overrides
      test.py Test overrides
    urls.py Root URL config
    wsgi.py
    asgi.py
  apps/ All Django apps…
  users/ Custom user mod…
      __init__.py
      admin.py
      apps.py
      models.py
      views.py
      urls.py
      forms.py
      managers.py Custom user man…
    migrations/
      __init__.py
    tests/
      __init__.py
      test_models.py
      test_views.py
  blog/ Example feature…
      __init__.py
      admin.py
      apps.py
      models.py
      views.py
      urls.py
      forms.py
    migrations/
      __init__.py
    tests/
      __init__.py
      test_models.py
      test_views.py
    __init__.py
  templates/ Project-wide te…
    base.html Master template
    404.html
    500.html
  users/
      login.html
      profile.html
  blog/
      post_list.html
      post_detail.html
  static/ Project-wide st…
    css/
    js/
    images/
  media/ User uploads (g…
  requirements/ Split requireme…
      base.txt
      local.txt
      production.txt
    .env.example
    .gitignore
    pytest.ini
```

Why This Structure?

This structure separates apps into their own directory, uses split settings for different environments, and centralizes templates at the project level. It scales well for teams of 2-5 developers working on distinct features.

Key Directories

`config/settings/` - Environment-specific settings (`base` → `local` / `prod` / `test`)

`apps/` - All Django apps grouped together, not scattered at root

`templates/` - Project-level templates, organized by app name

`requirements/` - Split dependencies matching settings structure

Settings Strategy

`base.py` - `INSTALLED_APPS` , middleware, shared config

`local.py` - `DEBUG=True` , SQLite, django-debug-toolbar

`production.py` - `DEBUG=False` , PostgreSQL, security headers

`test.py` - Fast password hasher, in-memory cache

Getting Started

1. Create virtualenv and install `requirements/local.txt`
2. `cp .env.example .env` and set `DJANGO_SETTINGS_MODULE=config.settings.local`
3. `python manage.py migrate`
4. `python manage.py createsuperuser`
5. `python manage.py runserver`

Creating New Apps

- `cd apps && django-admin startapp newapp`
- Add `"apps.newapp"` to `INSTALLED_APPS` in `base.py`
- Create `apps/newapp/urls.py` with `urlpatterns`
- Include in `config/urls.py` : `path("newapp/", include("apps.newapp.urls"))`

When To Use This

- Projects with 3-10 distinct feature areas
- Small teams (2-5 developers)
- Multiple deployment environments needed
- Projects expecting 6+ months of development

Trade-offs

More files - Split settings add complexity vs single `settings.py`

Import paths - `"apps.blog.models"` instead of `"blog.models"`

Initial setup - Takes longer than single-app but pays off quickly