Django HTMX Hybrid Project Structure

Server-rendered Django with HTMX for dynamic interactions. No JavaScript framework required.

#django #python #htmx #web #backend

PNG PDF Copy Prompt

Project Directory

```
myproject/
    manage.py
  config/
    __init__.py
    settings.py
    urls.py
    wsgi.py
    asgi.py
  apps/
  core/ Shared app
    __init__.py
    apps.py
    views.py Home, dashboard
    urls.py
  templatetags/ Custom tags
      __init__.py
      htmx_tags.py HTMX helper tags
  tasks/ Example feature
    __init__.py
    admin.py
    apps.py
    models.py
    views.py Full page + par…
    urls.py
    forms.py
  migrations/
      __init__.py
  tests/
      __init__.py
      test_views.py
      test_htmx.py HTMX response t…
    __init__.py
templates/
  base.html HTMX script inc…
  404.html
  components/ Reusable UI pie…
    button.html
    modal.html
    toast.html OOB toast notif…
    loading.html Loading spinner
  partials/ HTMX swap targe…
    _messages.html Flash messages
  tasks/
    task_list.html Full page
    task_detail.html
  partials/ HTMX fragments
    _task_row.html Single task row
    _task_list.html List without la…
    _task_form.html Inline edit form
    _task_count.html OOB counter upd…
static/
  css/
    main.css
    htmx.css HTMX transitions
  js/
    htmx.min.js Or use CDN
    app.js Minimal custom …
requirements.txt
.env.example
.gitignore
```

Why This Structure?

This structure embraces HTMX's "HTML over the wire" approach. Each feature has full-page templates plus `partials/` for HTMX fragments. The `components/` folder holds reusable UI pieces. No build step, no JavaScript framework—just Django templates with superpowers.

Key Directories

templates/components/ - Reusable UI (buttons, modals, toasts)

templates/partials/ - Global HTMX fragments

templates/{app}/partials/ - App-specific HTMX fragments

static/css/htmx.css - HTMX transition styles

Template Naming

task_list.html - Full page with `base.html` extends

_task_list.html - Partial (underscore = no extends)

_task_row.html - Single item for `hx-swap-oob`

HTMX Patterns Used

hx-get + hx-target - Load partials into page sections

hx-post + hx-swap - Inline form submission

hx-swap-oob - Update multiple page areas at once

hx-trigger="revealed" - Infinite scroll / lazy load

Getting Started

1. `pip install django django-htmx`
2. Add `"django_htmx"` to `INSTALLED_APPS`
3. Add `HtmxMiddleware` to `MIDDLEWARE`
4. Include `htmx.min.js` in `base.html`
5. Use `request.htmx` to detect HTMX requests

View Pattern

Views check `request.htmx` to return either a full page or just a partial. Example: `if request.htmx: return render(request, "tasks/partials/_task_list.html", ctx) else: return render(request, "tasks/task_list.html", ctx)`

When To Use This

- Traditional web apps needing modern UX
- Teams without frontend specialists
- Rapid prototyping with real interactivity
- SEO-important sites that need SPA-like feel
- Avoiding React/Vue complexity

Trade-offs

More templates - Partials add to template count

Testing complexity - Need to test both full and partial responses

Limited offline - Requires server for every interaction