Dasha Krit Technologies Pvt. Ltd.

DXDJ – D10X DJANGO BASE FRAMEWORK

Introduction

D10X has worked with multiple projects and own products based on Django web framework. Based on our experiences, we realized that typically medium to large scale Django projects commonly require the following functionality.

1. Simplified settings and a way to read the settings from configuration files.
    a. Separation of Configuration for dev, testing, staging and production environments.
    b. Configuration logging for different environments. (Especially things like email logs for crashes in staging and production environments)
    c. Simplifying Django deployments and generation web server configuration files.
2. A curated list of opensource third-party Django applications for fast UI development (e.g. Django-crispy forms, Django Slippers, Django-tables2 and Django-filters)
3. Custom UI template components developed on top of Django-slippers based on Twitter Bootstrap javascript CSS framework.
4. Dynamic HTML generation/loading with minimal javascript
5. Common security configurations and addressing security concerns.
6. Separation of Users and User Profiles. By default, Django does not have this separation. It adds everything in Django User
7. Roles and permissions
8. File uploads and attachment handling (including encrypted file storage, ability to validate that files are not tampered etc)
9. Email notifications with support for storing email templates in database and modifying email templates at runtime. Ability to add new types of notifications (e.g. whatsapp, SMS, FB messenger etc) as plugins.

10. Scheduled tasks and way to track the status of scheduled tasks.
    a. Integration with cron.
    b. An easy way to add new scheduled tasks without adding cron files.
11. Multi tenancy support
12. Workflow framework (integrated with auth, permissions and easy UI development)

D10X has developed reusable components, utilities, common base classes etc over Django and curated 3rd party Django frameworks. D10X objective was two fold.

- Reduce the bootstrap time for new projects. Our rough estimate is use of DXDJ reduces about 1 person years of efforts in bootstrapping a new project.
- DXDJ also improves the daily productivity of developers by (a) reducing the boiler plate (b) simplifying and improving the reliability of deployments (c) improving the security practices. The end result is number of bugs and security issues are reduced there by increasing the productivity of a typical Django developer.
-

This document describes the submodules available in DXDJ and purpose/functionality available in each submodule.

DXDJ Submodules and Features

DX Core

All low-level core dxdj functionality.

- Improved Django settings with readymade defaults for logging, debugging, 3rd party curated Django packages, google analytics, template caching, recommended security settings, etc.
- Management commands for easy Django deployment and sever configuration generation, migrations cleanup, database backup etc.
- UI utilities for Django Crispy forms, Django Tables2 and others

- Default Django templates preconfigured for HTML5 and typically page structure like navigation bar, header, footers, sidebar, alerts display etc.
- Django templates for generating nginx and apache server configurations with security best practices implemented.
- Template tags for markdown to HTML conversions, google analytics,
- Sub-action views to enable 'function based' views like functionality in Django class based views.

HTMX

HTMX (https://htmx.org/) is web technology to quickly develop dynamic progressive HTML pages with small amount of javascript. DXDJ HTMX module adds Django view mixins, template tags, etc to simplify the development of web applications based on HTMX.

UXCOMP

The UX Components module adds UX components like Cards, Modal popups, various types of buttons, menus etc These are Bootstrap framework based components developed using Django-slippers based syntax. It allows developers to quickly develop the webapps using the standards based theme-able good looking ui .

IAM – DXUser and DXUserProfile

Default Django User adds all user profile information in the Django user itself. This creates potential issues in maintaining audit trails, deleting user identifiable information, deleting users to disable user logins etc. The DXDJ simplifies such issues by separating authentication information (e.g. username and password etc) in DXUser and user profile information (e.g. mobile number, alternative emails etc) in DXUserProfile

IAM module also adds Forms and Views for commonly used functionality like authentication, reset password, forgot password, various rules for valid password and adds additional security checks for multiple brute force login attempts.

Notifications

Email notifications. Templates are stored in database. UI to edit the templates and check the preview.

Attachments - File Uploads and Attachments

Generic way to attach files to any existing Django models. Ability to encrypt files during storage /decrypt files during download. Ability to validate files and detect tampering using the stored SHA1 hash.

Scheduler

Ability to easily schedule hourly, daily, weekly and monthly and yearly tasks. Auto generates the cron files and shell wrappers required. Scheduled task are triggered using Django signal's mechanism. Adding a new daily tasks is as easy as registering a function to review a 'daily scheduled' signal.

Workflows

Django-fsm packages provides Finite State Machine based workflow for Django models. However, default fsm package has few limitations.

1. There is no support for multiple workflow connected to the same model.
2. There is no clean separation of workflow and the Django model.
3. There is no support for 'automatic transitions' , audit trail, notifications, etc
4. The developer must write lot of boilerplate views, forms etc to add user interface to the transitions.
5. The developer must manually link the view permissions with transition permissions. This can result in minor conflicts in permissions and cause bugs/security issues.

DXDJ workflow adds following features on top of Django-fsm

1. Separates workflows and Django models based on concept of 'bounded contexts'. The workflow acts a 'gateway' to modify the state of the bounded context. The bounded context may contain multiple linked model instances of different types.
2. This separation allows developer to create multiple parallel/serial workflows to act on a single bounded context.
3. Adds standard workflow views like State view, Transition View, ListView etc. integrated with workflow transition permissions. These views eliminate boiler

plate of writing views, forms and urls.py files. It also eliminates typical permission related security issues.

Servers

Running a Django application in production requires additional Django server packages like Gunicorn, UVCorn, waitress etc. This requires maintaining multiple server configuration files separate from Django configuration files.

Servers module adds Django management command wrappers over standard python WSGI/ASGI servers such that running the server in production is as simple as running a Django management command. Currently it adds servers based on Hypercorn and Waitress server frameworks. In future more server packages will be added to give choice to developers.