Django project base

Django project base

This project removes the boilerplate
associated with project and user handling.

Quick start guide

Removes the boilerplate

This project removes the boilerplate associated with project and user handling.

We start with a project

Everything revolves around it, users, roles, permissions, tags, etc.

Provides foundations

For user profiles, oauth authentication, permissions, projects, tagging, etc.

Introduction

What is django-project-base?

This project removes the boilerplate associated with project / user handling: We start with a project. Everything revolves around it: users, roles, permissions, tags, etc. This project makes it easy to work on that premise: it provides foundations for user profiles, oauth authentication, permissions, projects, tagging, etc.

In order to take advantage of all this, just enable desired middleware and extend the models. This project will take care of the groundwork while you focus on your own project.

This is a django library, based on django-rest-framework with DynamicForms and Django REST Registration integration.

Why django-project-base?

Functionalities provided:

- A base Project definition and editor for it. Extend as you like.
- User profile editor. Manage emails, confirmations, social connections
- Support for REST-based authentication / session creation
- Session / user caching for speed
- Project users editor. Invite users to project. Assign them into roles.
- Roles management & rights assignment.
- Tags editor & manager + support API for marking tagged items with their colours or icons
- Various Vue components for visualising the above in browsers

- various Vue components for visualising the above in browsers

---

Next page
Installation

Installation

Django project base

Install the package:

bash

```
$ pip install django-project-base
```

Extend the BaseProject & BaseProfile model:

myapp/models.py

python

```
from django_project_base import BaseProject

class MyProject(BaseProject):
    # add any fields and methods you like here

class MyProfile(BaseProfile):
    # add any fields and methods you like here
```

Django project base uses Swapper https://pypi.org/project/swapper/, an unofficial API for Django swappable models. You need to override the Project and Profile models before you can use the library: there aren't any migrations available in the library itself. The library only declares properties it itself supports, but you have the option to extend them as you wish to fit your needs too.

Then also make sure your swappable models are loaded instead of django-project-base models:

myproject/settings.py

python

```
DJANGO_PROJECT_BASE_PROJECT_MODEL = 'myapp.MyProject'
DJANGO_PROJECT_BASE_PROFILE_MODEL = 'myapp.MyProfile'

# Add to INSTALLED_APPS
INSTALLED_APPS = [
# ...
    'rest_registration',
    'django_project_base',
    'drf_spectacular',
# ...
]

# Add:
REST_FRAMEWORK = {
# YOUR SETTINGS
    'DEFAULT_SCHEMA_CLASS': 'drf_spectacular.openapi.AutoSchema',
}

# Add:
REST_REGISTRATION = {
    "REGISTER_VERIFICATION_ENABLED": False,
    "REGISTER_EMAIL_VERIFICATION_ENABLED": False,
    "RESET_PASSWORD_VERIFICATION_ENABLED": False,
    "LOGIN_DEFAULT_SESSION_AUTHENTICATION_BACKEND": "django_project_base.base.auth_ba
}
```

Append django project base urls:

myproject/urls.py

python

```
urlpatterns = [
    ...
    path('', include('django_project_base.urls')),
    ...
]
```

Alternatively you can also choose to specify individual URLs. If that's the case, please refer

to django_project_base.urls.py and use only the URLs you need.

> **INFO**
>
> The above general include also includes the Django javascript localisation catalog, so make sure you don't include it again.

There are some additional URLs available for the Django project base, like swagger or documentation. Appending those URLs is described in more details in respective chapters.

Dynamic Forms

Django project base is dependent on Dynamic Forms project https://github.com/velis74/DynamicForms

Read Dynamic Forms documentation for installation steps and more information about project.

You should add at least following code to your project, to enable Dynamic Forms.

myproject/settings.py

python

```
REST_FRAMEWORK = {
...
    'DEFAULT_RENDERER_CLASSES': (
      'rest_framework.renderers.JSONRenderer',
      'rest_framework.renderers.BrowsableAPIRenderer',
      'dynamicforms.renderers.TemplateHTMLRenderer',
      'dynamicforms.renderers.ComponentHTMLRenderer',
      'dynamicforms.renderers.ComponentDefRenderer',
  )
...
}
```

Environment setup

For JS development go to https://nodejs.org/en/ and install latest stable version of nodejs and npm. In {project base directory}/django_project_base/js_app run:

bash

```
$ npm install
```

To run a development server run:

bash

```
$ npm run serve
```

OR:

bash

```
$ vite
```

Go to http://localhost:8080/.

To generate a build run:

bash

```
$ npm run build
```

JS code is present in src subdirectory. For web UI components library vuejs(https://vuejs.org/) is used with single file components.

When developing webpack development server expects that service which provides data runs on host http://127.0.0.1:8000. This can be changed in vue.config.js found in the same directory as package.json. For running example django project prepare python environment and run {project base directory}:

bash

```
$ pip install -r requirements.txt
$ python manage.py runserver
```

Try logging in with user "miha", pass "mihamiha".

Activating features

Requires a settings.py `AUTHENTICATION_BACKENDS` <https://docs.djangoproject.com/en/dev/topics/auth/customizing/> _ setting.

Optionally also a global cache server such as Redis.

Previous page
Introduction

Next page
Settings options

Fields

HEXColorField

Field with validator for color in hex format, currently used for setting background color for Tags.

Previous page
Tags

Next page
Middleware

Settings options - quick overview

Previous page
Installation

Next page
Tags

Django project base

Menu Return to top

Tags

Django project base supports tags usage. See example implementation bellow.

python

```python
class DemoProjectTag(BaseTag):
   content = models.CharField(max_length=20, null=True, blank=True)
   class Meta:
       verbose_name = "Tag"
       verbose_name_plural = "Tags"

class TaggedItemThrough(GenericTaggedItemBase):
   tag = models.ForeignKey(
       DemoProjectTag,
       on_delete=models.CASCADE,
       related_name="%(app_label)s_%(class)s_items",
   )

class Apartment(models.Model):
   number = fields.IntegerField()
   tags = TaggableManager(blank=True, through=TaggedItemThrough,
                          related_name="apartment_tags")

# Example code
from example.demo_django_base.models import DemoProjectTag
dt = DemoProjectTag.objects.create(name='color tag 20', color='#ff0000')

from example.demo_django_base.models import Apartment
a = Apartment.objects.create(number=1)
a.tags.add(dt)
a.tags.all()

<QuerySet [<DemoProjectTag: color tag 20>]>

# Get background svg for tags
```

```
DemoProjectTag.get_background_svg_for_tags(Apartment.objects.all().first().tags.all()
```

Previous page
Settings options

Next page
Fields

Middleware

Previous page
Fields

Next page
Modules

Modules

Previous page
Middleware

Next page
Swagger

Swagger

Installation

To enable swagger gui, add following to urls.py

my_project/urls.py

python

```
urlpatterns = [
    ...
    path('schema/', SpectacularAPIView.as_view(), name='schema'),
    path('schema/swagger-ui/', SpectacularSwaggerView.as_view(url_name='schema', ),
        name='swagger-ui'),
    ...
]
```

Swagger UI is now accessible on /schema/swagger-ui/ url by running example project.

Previous page
Modules

Next page
Open API

Django project base

Menu

Return to top

Open API

Add folloving to settings.py

myapp/settings.py

python

```python
# myapp/settings.py
REST_FRAMEWORK = {
    ...
    'DEFAULT_SCHEMA_CLASS': 'drf_spectacular.openapi.AutoSchema',
    ...
}
```

Previous page
Swagger