# Django Project Base

## version 0.1.0

April 14, 2021

# Welcome to Django Project Base's documentation!

## What is django-project-base?

We start with a project. Everything revolves around it: users, roles, permissions, tags, etc. Everything belongs to a project first, then to database. This project makes it easy to work on that premise. At the same time it integrates a few basic operations that you need in every project so that you don't have to do them over and over again.

This is a django library, based on django-rest-framework with django-allauth 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

### *Installation*

Install the package:

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

Extend the BaseProject & BaseProfile model:

```
# myapp/models.py
from django_project_base import BaseProject

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

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

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

```
# myproject/settings.py

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

# urls.py add
from django_project_base.router import django_project_base_urlpatterns
urlpatterns = [ ... ] + django_project_base_urlpatterns

Add to INSTALLED_APPS
 'rest_registration',
 'django_project_base',
 'drf_spectacular',

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

### *Warning*

This is important!!! You need to do the overriding before you create migrations. Migrating after default models had been created and used is a really hard and painful process. So make triple sure you don't deploy your application without first making sure the model you want to use is either your own or you are satisfied with our default implementation.

## *Settings*

```
DJANGO_PROJECT_BASE_BASE_REQUEST_URL_VARIABLES: {
    'project': {'value_name': 'current_project_slug', 'url_part': 'project-'},
    'language': {'value_name': 'current_language', 'url_part': 'language-'}
}
```

This setting defines dictionary of attribute names on request object. For e.g. project info is set on request object under propery current_project_slug. Language information is set on request objects under property current language. Is language or project is given in request path like: language-EN, then url_part settings is found and EN string is taken as language value.

```
DJANGO_PROJECT_BASE_SLUG_FIELD_NAME: 'slug'
```

When creating models with slug field they should be named with this setting value. This enables that we can use object slug instead of object pk when making api requests.

## *Javascript Client*

### *Usage*

Look at django_project_base/templates/index.html for examples.

**API Documentation**

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

**Translations**:

If you want to use your Django translations in your app include <script src="{% url 'javascript-catalog' %}"></script> in your html document header.

### *For developers*

For code formatting use .jshintrc file present in repository. Set tab size, ident, continuation ident in your editor to 2 places.

For JS development go to https://nodejs.org/en/ and install latest stable version of nodejs and npm. In project base directory run npm install. To run a development server run npm run dev (go to http://0.0.0.0:8080/). To generate a build run npm run build.

JS code is present in src directory. For web UI components library vuejs(https://vuejs.org/) is used. Components are built as Vue global components(https://vuejs.org/v2/guide/components.html) with x-templates. Templates are present in templates directory.

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 webpack.config.js file. For running example django project prepare python environment and run (run in repository root):

- pip install -r requirements.txt (run in content root)
- python manage.py runserver

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

## *Tags*

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

```
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_ta
# 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())
```

## Middleware

***Project Middleware***

ProjectMiddleware: If you wan't to set current project which is selected to request object you can use ProjectMiddleware which should be placed to start of MIDDLEWARE list in settings.py. Middleware sets DJANGO_PROJECT_BASE_BASE_REQUEST_URL_VARIABLES setting dict values to request object. Default value for DJANGO_PROJECT_BASE_BASE_REQUEST_URL_VARIABLES setting is {'project': 'current_project_slug', 'language': 'current_language'}. This means request will have current_project_slug attribute which will have value set to current project slug and request will have current_language attribute which will have value set to current language set. If project or language cannnot be determined its value is set to None.

To set current project to ajax requests 'Current-Project' header should be used: 'Current-Project': 'current project slug'. Current slug can also be determined from request path. See DJANGO_PROJECT_BASE_PROJECT_DEFINED_URL_PART setting description in setting section.

```
# myproject/settings.py

MIDDLEWARE = [
 'django_project_base.base.middleware.UrlsVarsMiddleware',
 ...
 ]
```

## Modules

The page contains all information about Django Project Base modules:

### Authentication

***Impersonate user***

Sometimes is useful if we can login into app as another user for debugging or help purposes. User change is supported via REST api calls or you can use userProfile component (django_project_base/templates/user-profile/bootstrap/template.html) which already integrates api functionality. Functionality is based on django-hijack package.

For determining which user can impersonate which user you can set your own logic. Example below:

```
# settings.py
 HIJACK_AUTHORIZATION_CHECK = 'app.utils.authorization_check'

# app.utils.py
 def authorization_check(hijacker, hijacked):
     """
     Checks if a user is authorized to hijack another user
     """
     if my_condition:
         return True
     else:
         return False
```

**To increase AUTH performance you can set backend which caches users:**

- django_project_base.base.auth_backends.UsersCachingBackend
- django_project_base.base.auth_backends.CachedTokenAuthentication

Example (add in settings.py):

```
# myproject/settings.py

AUTHENTICATION_BACKENDS = (
    'django_project_base.base.auth_backends.UsersCachingBackend',  # cache users for auth to
    'django.contrib.auth.backends.ModelBackend',
)
```