Python | Django-allauth setup and Configuration

Last Updated : 24 May, 2024

User registration is one of the most essential parts of a web application. `django-registration-redux` and `django-alluth` are the most famous registration apps available in Django. This tutorials series deals with setup, configuration, and customization of `django-allauth` and serve as a guide for new users who want to get started quickly with `allauth` and make useful customizations along the way without much pain.

This article covers setup and some basic configurations. Later, we will deal with social login, extending classes and efficient use of `DefaultAccountAdapter` to add custom process.

It can be overwhelming to a `django` novice or a new user of `django-allauth` itself. Although it is well documented, due to time and resource constraints of the developers involved, there has not been many articles and in-depth tutorials on the library. So this series tries to solve that problem and make a comprehensive series of guides to make `django-allauth` easy to use and work with for the django-community.

How to Setup?

below guide you through the setup.

- Create a Django project if you already don't have one.
- Install `django-allauth` using the command `pip install django-allauth`
- Add `'allauth, allauth.account'`, `allauth.socialaccount` and all the necessary social logins to `INSTALLED_APPS`. You can view the entire list of supported API's here. The Social login feature is described in detail in the next article. After you configure your installed apps should be similar as given below.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'allauth.socialaccount.providers.google',
    'allauth.socialaccount.providers.facebook',
    'django.contrib.auth',
    'django.contrib.sites',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

- Configure the `template` context processor settings in `settings.py` and also add URL pattern in the project urls.py

```
TEMPLATES = [
  {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.normpath(os.path.join(BASE_DIR, 'templates')),
        ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'django.template.context_processors.request',
            ],
        },
    },
]
```

- Add the following authentication backend.

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
)
```

- Copy the template files from the django-allauth repository or my custom repository(I have made some modifications and some good structuring) and paste it in the `templates` folder in your project directory.

- Add the allauth urls in `urls.py` of your main project directory. After adding the allauth urls the below should look like,

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import url, include
urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'^accounts/', include('allauth.urls')),
]
```

  - You can also add the custom CSS yourself or my CSS (Well commented and documented) that I have created during my use of the allauth templates. It includes styling for almost all the pages, and even mobile-friendly email templates for confirmation and password reset emails. You can do that by creating a `static` folder in the project directory and placing the CSS in `account` folder.
  - Run `python manage.py makemigrations` and `python manage.py migrate` to run all the necessary migrations and run `python manage.py runserver` to start the django server.
  - Follow the URL patterns to display the registration form.
    Eg: `localhost:8000/accounts/login` to display the login page.

Configuration:
Most django-allauth features are can be configured using the built-in adapters and variables by placing them in `settings.py`. file. Although the documentation has tons of such options with good explanations, highlighted some important ones below.

  - Email confirmation expiry: Sets the number of days within which an account should need to be activated.
    Eg:`ACCOUNT_EMAIL_CONFIRMATION_EXPIRE_DAYS=7`
  - Email required for activation: This option allows you to set whether the email address should be required to register. Set `False` to disable email requirement. Eg: `ACCOUNT_EMAIL_REQUIRED = True`

- Account email verification: This option can be used to set whether an email verification is necessary for a user to login after he registers an account. You can use 'mandatory' to block a user from logging in until the email gets verified. You can set `optional` for sending the email but allowing the user to login without an email. You can also set `none` to not send any verification email. (Not Recommended) Eg: `ACCOUNT_EMAIL_VERIFICATION = "mandatory"`
- Login Attempt Limit: This is an important feature which can be used to prevent brute force attacks on the user login module in allauth. The maximum number of login attempts can be set, and the user gets blocked from logging in until a timeout. This feature makes use of `ACCOUNT_LOGIN_ATTEMPTS_TIMEOUT` setting. Eg: `ACCOUNT_LOGIN_ATTEMPTS_LIMIT = 5`
- Login Attempt Limit timeout: This setting needs to should is used with `ACCOUNT_LOGIN_ATTEMPTS_LIMIT` setting. The value set is in seconds from last unsuccessful login attempt. Please do not that this does not prevent admin login from being brute forced. Eg: `ACCOUNT_LOGIN_ATTEMPTS_TIMEOUT = 86400 # 1 day in seconds`
- Login and Logout URL redirection: When user logs in or logs out, you might want to redirect the user to a particular URL or page and the below settings can be used to set those values. By default allauth redirects login to `/accounts/profile/` URL and logout to the `localhost:8000` or any `localhost` homepage.
  Eg : `ACCOUNT_LOGOUT_REDIRECT_URL ='/accounts/login/'`
  Eg : `LOGIN_REDIRECT_URL = '/accounts/email/'`

Finally, your `allauth` settings should look similar to the below settings.

```
#django-allauth registraion settings
ACCOUNT_EMAIL_CONFIRMATION_EXPIRE_DAYS =1
ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_EMAIL_VERIFICATION = "mandatory"
ACCOUNT_LOGIN_ATTEMPTS_LIMIT = 5

# 1 day
ACCOUNT_LOGIN_ATTEMPTS_TIMEOUT = 86400

#or any other page
ACCOUNT_LOGOUT_REDIRECT_URL ='/accounts/login/'
```

```
# redirects to profile page if not configured.
LOGIN_REDIRECT_URL = '/accounts/email/'
```

gjbht 5

Previous Article Next Article

Generate random numbers and background colour in Django

Similar Reads

Python | Extending and customizing django-allauth

Prerequisite: Django-allauth setup and Configuration Let's deal with customizing django-allauth signup forms, and intervening in registration flow to add custom...

4 min read

How To Setup AWS Xray Tracing Setup Or Django Application ?

AWS X-Ray provides powerful tracing capabilities, allowing you to gain insights into your application's performance, identify bottlenecks in your app, and...

6 min read

Spring Boot – Auto-Configuration vs Manual Configuration

Spring Boot simplifies Java application development by offering features like auto-configuration. While auto-configuration helps developers get started...

3 min read

Wipro Interview Experience for Setup Configuration Specialist

Filled out the form from Cocube in 2023. It conducts a total 3 rounds Group Discussion (topic: "Which is better: Private jobs or Government jobs?")Gate (...

1 min read

How To Docker SetUp Three Architecture Setup ?

This article provides an easily understandable explanation on using Docker to build a three-tier architecture. In this 3tier architecture we use WordPress as...

5 min read

Django Installation and Setup

Installing and setting up Django is a straightforward process. Below are the step-by-step instructions to install Django and set up a new Django project on your...

2 min read

Django Channels - Introduction and Basic Setup

Django is a powerful Python framework for web development. It is fast, secure, and reliable. Channels allow Django projects to handle HTTP along with...

6 min read

Configuration for Django WebSocket Application on Ubuntu Server

This tutorial will walk you through each step in detail on how to configure your django websocket application on a Ubuntu 20.10 server. This article assumes y...

3 min read

Django+React Full Stack Development Setup using Dact

When we work on a project having Django as our Back-end and having a powerful front-end using React, the development setup takes a significant...

2 min read

Adding Tags Using Django-Taggit in Django Project

Django-Taggit is a Django application which is used to add tags to blogs, articles etc. It makes very easy for us to make adding the tags functionality to our djang...

2 min read

Article Tags : Technical Scripter Technical Scripter 2018