

# Django Authentication System

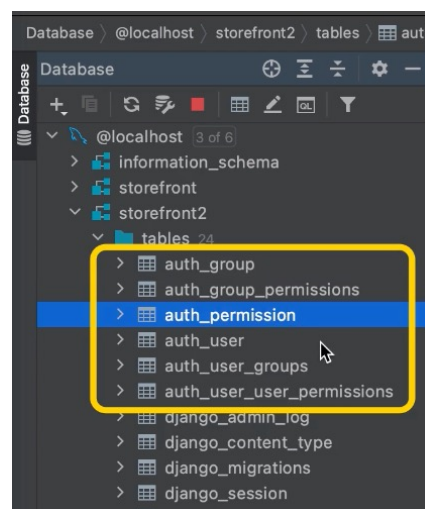
Kavin Asavanant, Ph.D.  
Chulalongkorn Business School

1

---

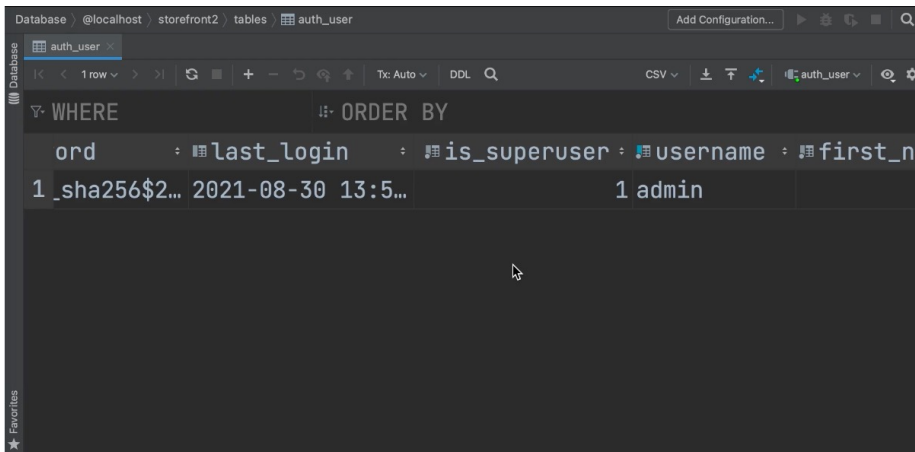
## Authentication module

```
storefront > settings.py > ...  
  
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.sessions',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'django_filters',  
    'rest_framework',  
    'playground',  
    'debug_toolbar',  
    'store',  
]
```



---

# Authentication module



The screenshot shows a database client interface with the following table structure and data:

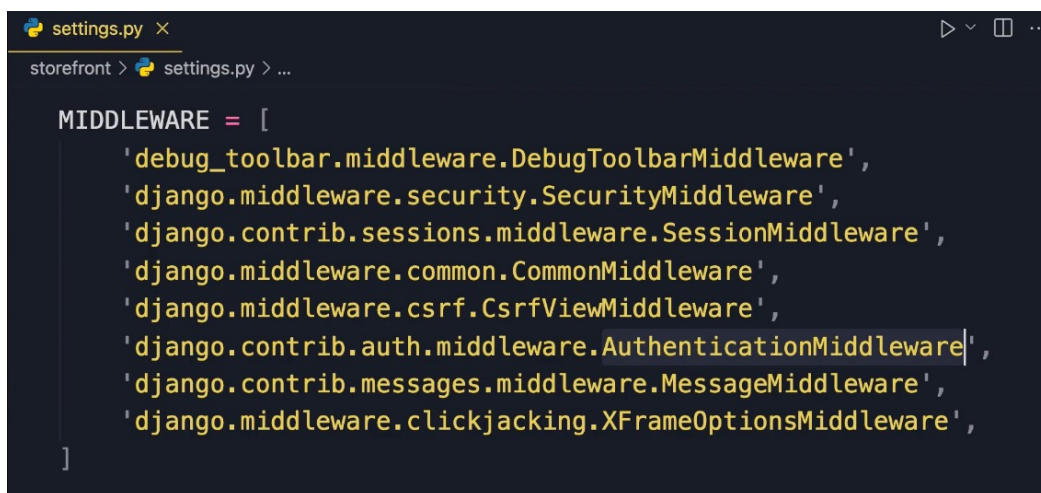
	ord	last_login	is_superuser	username	first_name
1	_sha256\$2...	2021-08-30 13:5...	1	admin	

is\_superuser: has all the permissions  
is\_staff: able to login to the admin module or not  
is\_active: still active or not, not really deleting the users for maintaining foreign key constraints

<https://docs.djangoproject.com/en/4.1/ref/contrib/auth/>

---

# Middleware



```
settings.py ×
storefront > settings.py > ...

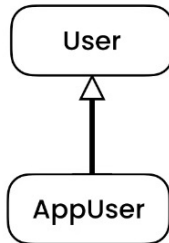
MIDDLEWARE = [
    'debug_toolbar.middleware.DebugToolbarMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

# Customizing the User Model

---

## Option 1 Inheritance approach

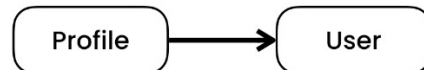
**Extend User**



For storing attributes related to authentication

## Option 2 Composition approach

**Create Profile**



For storing non-auth related attributes

# Customizing the User Model

---

**sales**

Customer

**hr**

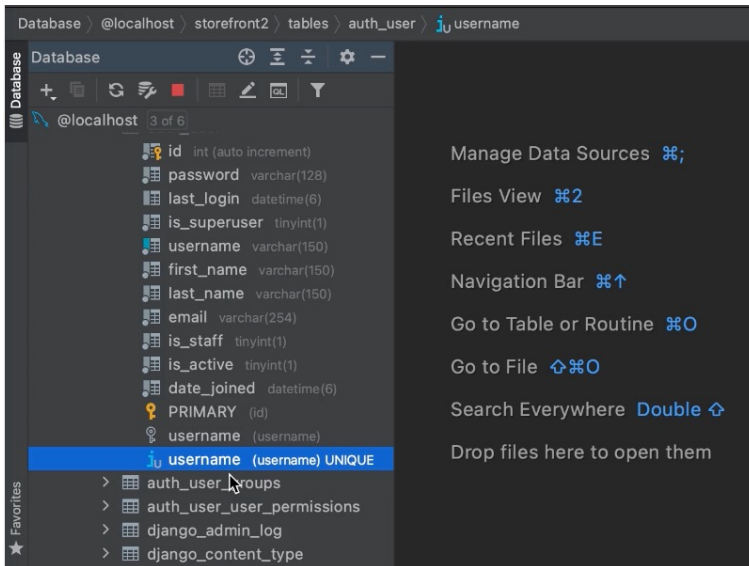
Employee

**training**

Student

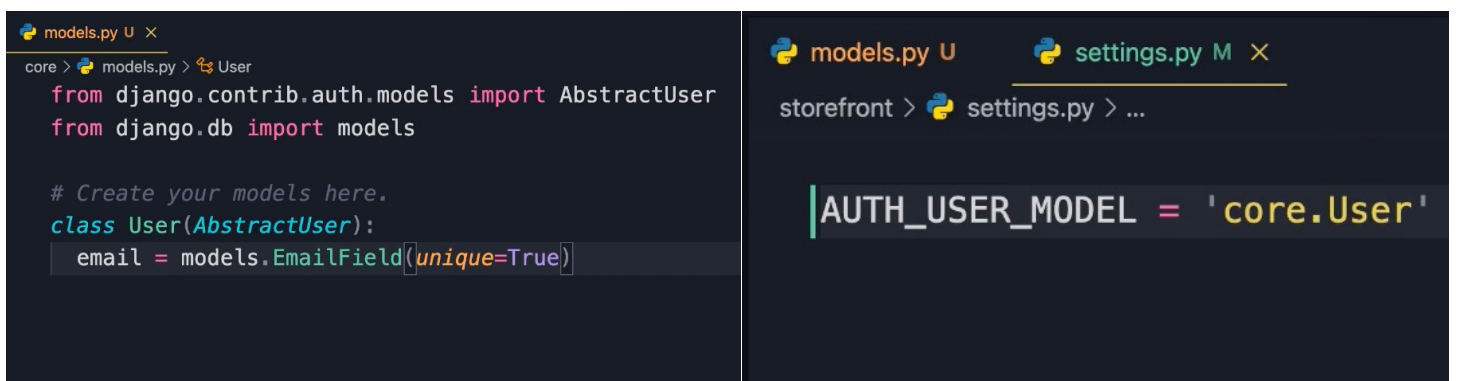
Different apps have different concepts of profile

# Option 1: Extend User



Use case: Making the email unique so that users can login with their emails

# Option 1: Extend User



# Option 1: Extend User

---

```
admin.py U x
core > admin.py > ...
from store.models import Product
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
from django.contrib.contenttypes.admin import GenericTabularInline
from store.admin import ProductAdmin
from tags.models import TaggedItem
from .models import User

@admin.register(User)
class UserAdmin(BaseUserAdmin):
    pass
```

# Option 1: Extend User

---

```
admin.py core 2, U x admin.py ~/.../auth
core > admin.py > UserAdmin
@admin.register(User)
class UserAdmin(BaseUserAdmin):
    add_fieldsets = (
        (None, {
            'classes': ('wide',),
            'fields': ('username', 'password1', 'password2'),
        }),
    )
```

# Option 2: Creating User Profiles

---

```
models.py 2, M X
store > models.py > ...
from django.conf import settings
from django.core.validators import MinValueValidator
from django.db import models
from uuid import uuid4
```

```
models.py M X
store > models.py > Customer
(MEMBERSHIP_GOLD, 'Gold'),
]
first_name = models.CharField(max_length=255)
last_name = models.CharField(max_length=255)
email = models.EmailField(unique=True)
phone = models.CharField(max_length=255)
birth_date = models.DateField(null=True, blank=True)
membership = models.CharField(
    max_length=1, choices=MEMBERSHIP_CHOICES, default=MEMBERSHIP_GOLD
)
user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
```