DJANGO ADMIN

江琳

USER MODEL

USER MODEL

- **Django has a default user model.**
- **User objects are the core of the authentication system.**
- **Only one class of user exists in Django's authentication framework**

USER OBJECTS

- **User objects have the following fields:**

- **username**
- **first_name**
- **last_name**
- **email**
- **password**
- **groups**
- **user_permissions**
- **is_staff**
- **is_activate**
- **is_superuser**
- **last_login**
- **date_joined**

PLAY WITH DBSHELL

```
(env) $ python manage.py dbshell
SQLite version 3.16.0 2016-11-04 19:09:39Enter ".help"
for usage hints.
sqlite> .table
auth_group                    books_book
auth_group_permissions        books_category
auth_permission               books_publisher
auth_user                     django_admin_log
auth_user_groups              django_content_type
auth_user_user_permissions    django_migrations
books_author                  django_session
```

- **auth_user is the default user model**

PLAY WITH DBSHELL

- The command *python manage.py dbshell* access the database that define in settings.py.
- If the database is PostgreSQL or MySQL, password will be required.
- We can execute custom SQL command under this mode.

CUSTOM USER MODEL

- We can user the User model directly
- But if you're starting a new project, it's highly recommended to set up a custom user model, even if the default User model is sufficient for you.
- You also can add extra fields to the custom user model

CUSTOM USER MODEL

- **Before we custom user model, creating a app named accounts first.**

```
(env) $ python manage.py startapp accounts
```

STEP 1: CUSTOM USER MODEL

```
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    pass
```

- ***AbstractUser*** **is a full User model, compete with fields, as an abstract class so that you can inherit from it and add your own profile fields and methods.**

STEP 1: CUSTOM USER MODEL

```
from django.db import models
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    nickname = models.CharField(max_length=100)
```

STEP2: SETTINGS

- **Open settings.py and add :**

```
AUTH_USER_MODEL = 'accounts.User'
```

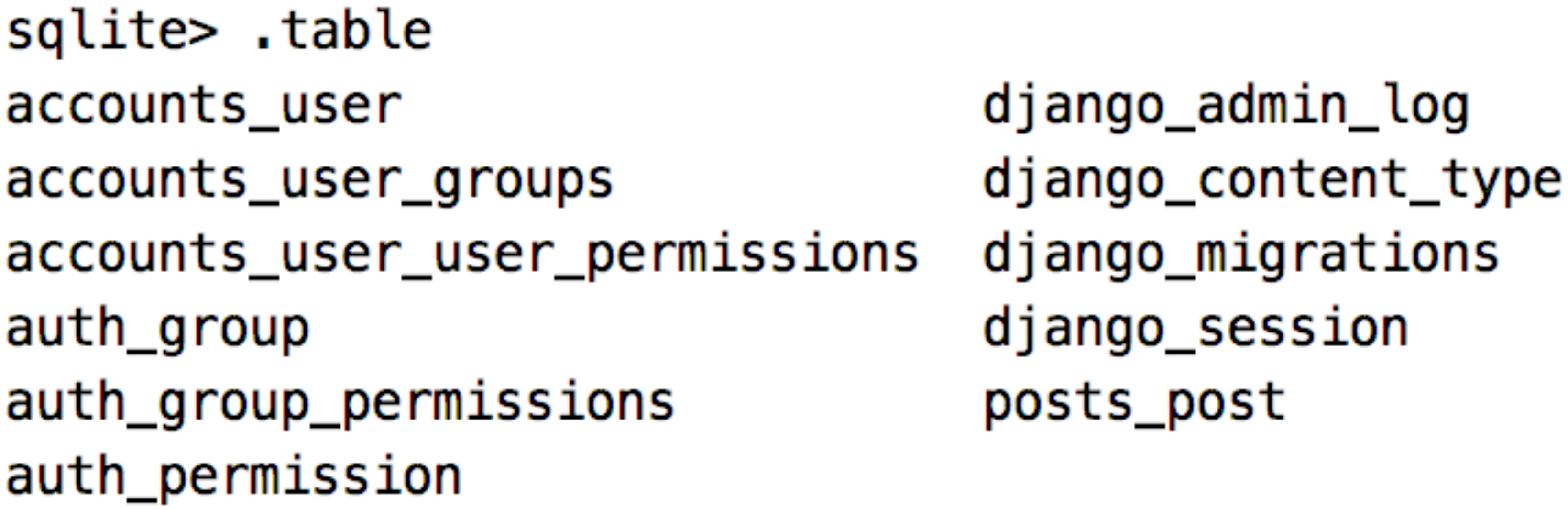
STEP 3: MIGRATIONS

```
$ python manage.py makemigrations accounts
$ python manage.py migrate
```

TIPS

- If faill to migrate:
  - Run: rm –f db.sqlite3 to delete db.sqlite3
  - Then migrate

CUSTOM USER MODEL



```
sqlite> .table
accounts_user                       django_admin_log
accounts_user_groups                django_content_type
accounts_user_user_permissions      django_migrations
auth_group                          django_session
auth_group_permissions              posts_post
auth_permission
```

There is no auth_user table any more, because Django allows one user model only.

CUSTOM USER MODEL

```
(env) $ python manage.py dbshell
SQLite version 3.16.0 2016-11-04 19:09:39
Enter ".help" for usage hints.
sqlite> .table
accounts_user                    books_book
accounts_user_groups             books_category
accounts_user_user_permissions   books_publisher
auth_group                       django_admin_log
auth_group_permissions           django_content_type
auth_permission                  django_migrations
books_author                     django_session
```

There is no auth_user table any more, because Django allows one user model only.

DJANGO ADMIN

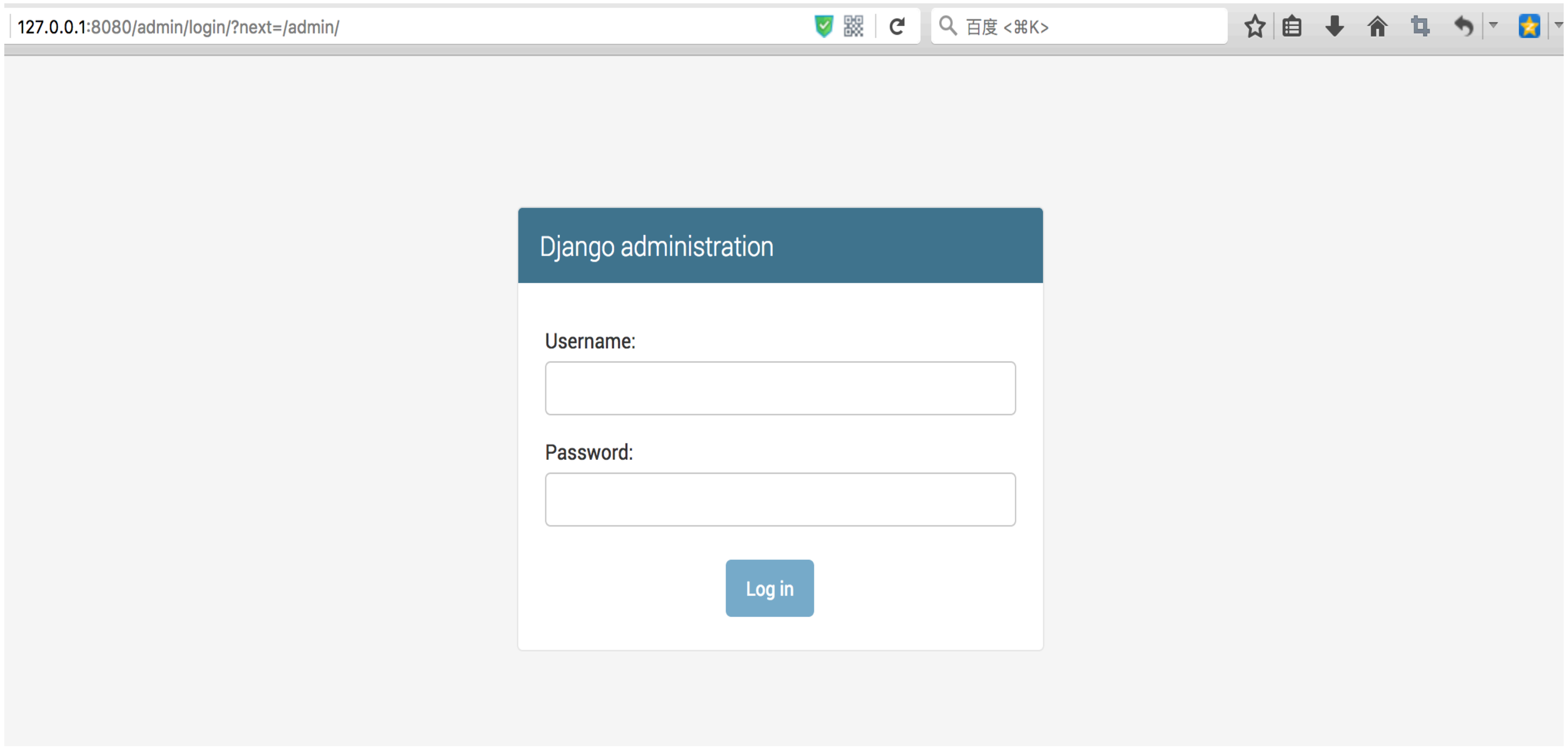
PHILOSOPHY

- Generating admin sites for your staff or clients to add, change, and delete content is tedious work that doesn't require much creativity. For that reason, Django entirely automates creation of admin interfaces for models.
- The admin isn't intended to be used by site visitors. It's for site managers.

ADMIN SITE

- python manage.py runserver
- Open the link: http://127.0.0.1/admin

ADMIN SITE


127.0.0.1:8080/admin/login/?next=/admin/

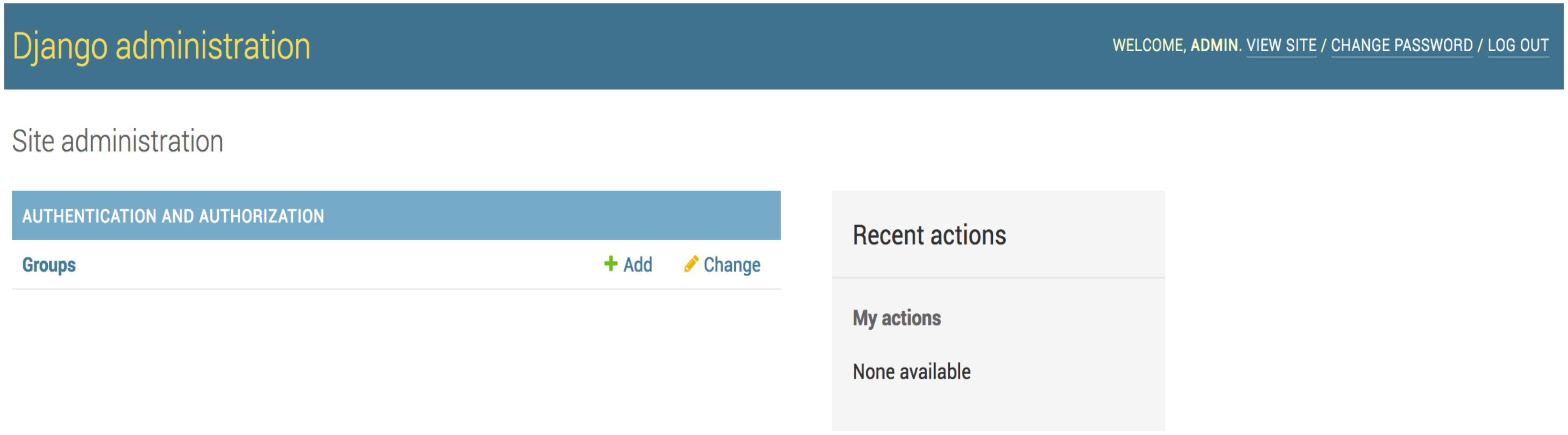
CREATING SUPERUSER

- **First of all, we need to create a user who can login to the admin site.**

```
(env) $ python manage.py createsuperuser
Username: admin
Email address: admin@admin.com
Password:
Password (again):
Superuser created successfully.
```

ADMIN SITE

- **Now we can login to the admin site**


Django administration

WELCOME, **ADMIN**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups + Add ✎ Change

Recent actions

My actions

None available

ADMIN SITE

- Nothing except group. Where is post model and user model?
- We need tell tell the admin that who have an admin interface.

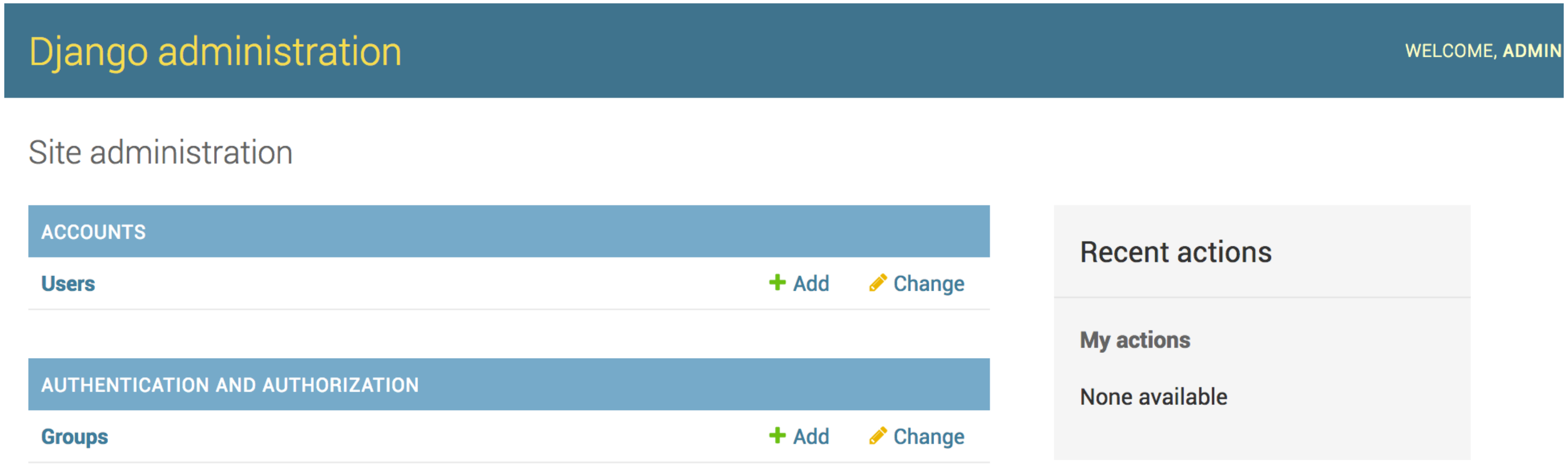
ADMIN SITE

- **open the accounts/admin.py and code below:**

```
from django.contrib import admin
from .models import User


admin.site.register(User)
```

ADMIN SITE


Django administration

WELCOME, ADMIN

Site administration

ACCOUNTS		
Users	+ Add	Change

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change

Recent actions

My actions

None available

Now, we can add new user and change existing user in admin site

ADMIN SITE

- **Open books/admin.py then code below:**

```
from django.contrib import admin
from .models import Publisher, Author, Category, Book


admin.site.register(Publisher)
admin.site.register(Author)
admin.site.register(Category)
admin.site.register(Book)
```

MODELADMIN

- **The ModelAdmin class is the representation of a model in the admin interface.**

```
from django.contrib import admin
from .models import User


class UserAdmin(admin.ModelAdmin):
    pass

admin.site.register(User, UserAdmin)
```

MODELADMIN

```
from django.contrib import admin
from .models import User


class UserAdmin(admin.ModelAdmin):
    fields = ('username', 'is_superuser', 'password')
    list_display = ('username', 'is_superuser')


admin.site.register(User, UserAdmin)
```

MODELADMIN

- *fields* option to make simple layout changes in the forms on the "add" and "change" pages
- *list_display* to control which fields are displayed on the change list page of the admin.

THE REGISTER DECORATOR

```
from django.contrib import admin
from .models import User


@admin.register(User)
class UserAdmin(admin.ModelAdmin):
    fields = ('username', 'is_superuser', 'password')
    list_display = ('username', 'is_superuser')
```

- **The register decorator can instead of admin.site.reigster**

Questions?