Django

CC Club MNNIT (Lecture 2)

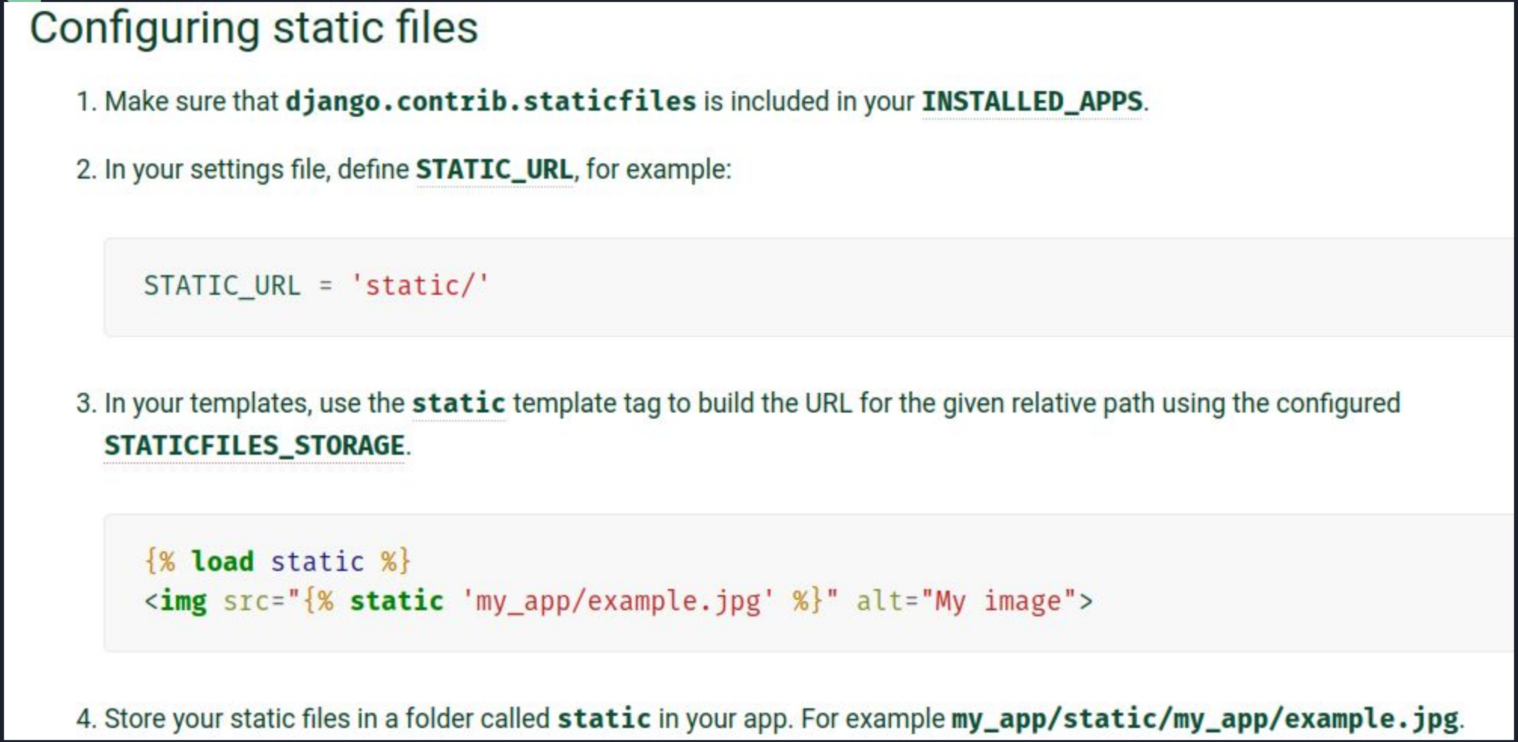
TABLE OF CONTENTS

What is load static tag?Why it is used?

Websites might consists of many css,images,javascript files which remain unchanged we call these files “static files” and to load this static files django have a predefined app called staticfiles.

To load this static files we use ’ {%load static%}’ on top of our HTML files.

How to Load Css/Img Files ?



Configuring static files

1. Make sure that **django.contrib.staticfiles** is included in your **INSTALLED_APPS**.
2. In your settings file, define **STATIC_URL**, for example:

```
STATIC_URL = 'static/'
```

3. In your templates, use the **static** template tag to build the URL for the given relative path using the configured **STATICFILES_STORAGE**.

```
{% load static %}
<img src="{% static 'my_app/example.jpg' %}" alt="My image">
```

4. Store your static files in a folder called **static** in your app. For example **my_app/static/my_app/example.jpg**.

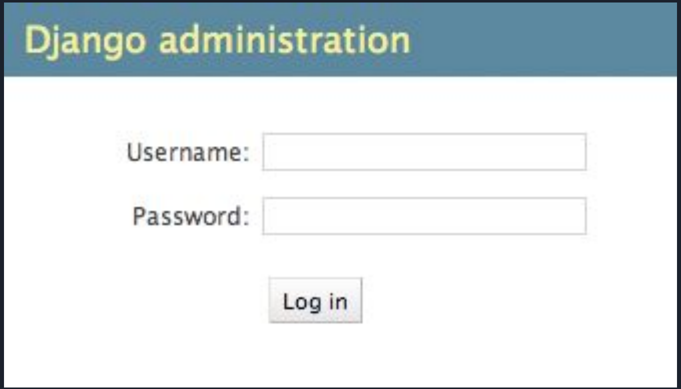
Creating Superuser

“python manage.py createsuperuser” command is used to create the superuser which gives the access to our database (sqlite).

Once entering your username , email id and password your admin user will be created.

Login in that admin user by going to

“ http://127.0.0.1:8000/admin/”


Django administration

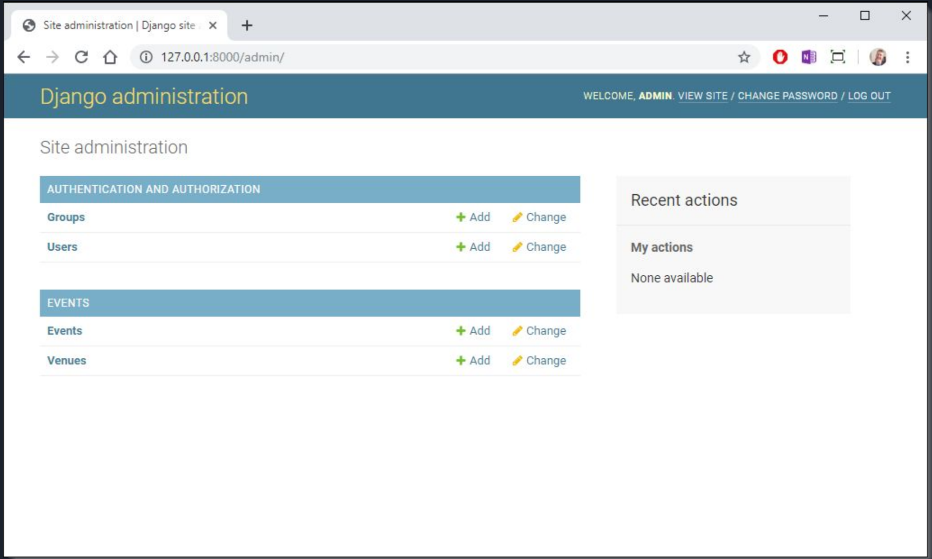
Username:

Password:

Log in

1. Can create multiple superusers for a single project

2. Direct access of the database from here


Site administration | Django site

127.0.0.1:8000/admin/

Django administration

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

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

EVENTS		
Events	+ Add	Change
Venues	+ Add	Change

Recent actions

My actions

None available

Authentication and User Object

1. Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions.
2. User Object : Default object provided by django to use Django Auth
3. The primary attributes of the default user are:

- `username`
- `password`
- `email`
- `first_name`
- `last_name`

Modules Required to use Django Auth

```
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.models import User
```

Registering a new user

```
new_user = User.objects.create_user(username=”rohan”,email =
“rohan@gmail.com”,password=”abcde”)
new_user.first_name = fname
new_user.last_name = lname
new_user.save()
```

Authenticating Users

Use authenticate() to verify a set of credentials. It takes credentials as keyword arguments, username and password for the default case, checks them against each authentication backend, and returns a User object if the credentials are valid for a backend. If the credentials aren’t valid for any backend or if a backend raises PermissionDenied, it returns None.

```
from django.contrib.auth import authenticate

user = authenticate(username='john', password='secret')

if user is not None:

    # A backend authenticated the credentials

    login(request,user)

else:

     # No backend authenticated the credentials

    return redirect (‘home’)
```

Logging Out

```
from django.contrib.auth import logout

def logout_view(request):

    logout(request)

      # Redirect to a success page.
```

Note : To access user at point of time we can :

```
def func(request):

    user = request.user

    If user.is_authenticated:

        # checks whether this user is valid or not
```

Homework:
How to change password (Forgot Password)

Models in Django

Models define your database layout (as in what are the different data fields you are going to have).

```
from django.db import models
# Create your models here.
class studentProfile(models.Model):
    student_id=models.AutoField
    firstname=models.CharField(max_length=100,default="",blank=True)
    lastname=models.CharField(max_length=100,default="",blank=True)
    email=models.EmailField(max_length=100)
    password=models.CharField(max_length=25)
    address = models.CharField(max_length=100,default="",blank=True)

    def __str__(self):
        return self.firstname+" "+self.lastname
```

Migrations

```
“python manage.py makemigrations”
```

```
“python manage.py migrate”
```

By running makemigrations, you’re telling Django that you’ve made some changes to your models (in this case, you’ve made new ones) and that you’d like the changes to be stored as a *migration*.

The migrate command takes all the migrations that haven’t been applied (Django tracks which ones are applied using a special table in your database called django_migrations) and runs them against your database - essentially, synchronizing the changes you made to your models with the schema in the database.

admin.py

You need to register your model in admin.py

The register function is used to add models to the Django admin so that data for those models can be created, deleted, updated and queried through the user interface.

```
from django.contrib import admin

# Register your models here.

from .models import studentProfile

admin.site.register(studentProfile)
```

Operations performed on data

- Create
- Read
- Update
- Delete

Creating objects in Database

- For Creating new object of a class :
  new_student = StudentProfile(first_name = “rohan” , last_name = lname , mobile = mobile_no)
- StudentProfile.save(new_student)
- Or we can also do this
- new_student = StudentProfile()
- new_student.first_name = “rohan”
- StudentProfile.save(new_student)

Read / Fetch Data

Using previous example , we will fetch the data of student from StudentProfile.

student = StudentProfile.objects.get(mobile=mobile_no)

student will store the details of the student whose mobile number is mobile_no

mobile_no-> mobile number of the student

mobile-> name of attribute

What if we want to get the list of students with some particular name

Use of filter : gives a list of objects

students = StudentProfile.objects.filter(student = student_name)

students : list of students whose name are student_name

student_name : name of the student

student : name of the attribute

Update and Delete is similar to previous ones

- new_student.first_name = “rohan”
- StudentProfile.save(new_student) or new_student.save()
- Similarly for deleting an object
- new_student.delete()