Django Framework

Merit Ehab

Templates

- A base template is the most basic template that you extend on every page of your website.
- In your ***blog/templates/blog/*** create an new file ***base.html***.
- Edit ***base.html*** and copy everything you have ***index.html*** in it.

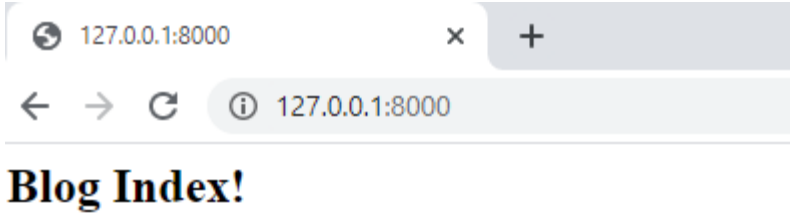
- Replace whatever after the ***page-header div*** in the body with those tags `{% block content %}{% endblock %}` and save the file.

- You used the template tag `{% block %}` to make an area that will have HTML inserted in it. That HTML will come from another template that extends this template (***base.html***).

Templates

- In ***blog/templates/blog/index.html***, remove everything above and under `<h2>Blog Index!</h2>`.

- Include the above code in the content blocks between `{% block content %}` and `{% endblock %}`.




```
blog > templates > blog > <> index.html > ...
1   {% block content %}
2       <h2>Blog Index!</h2>
3   {% endblock %}
```

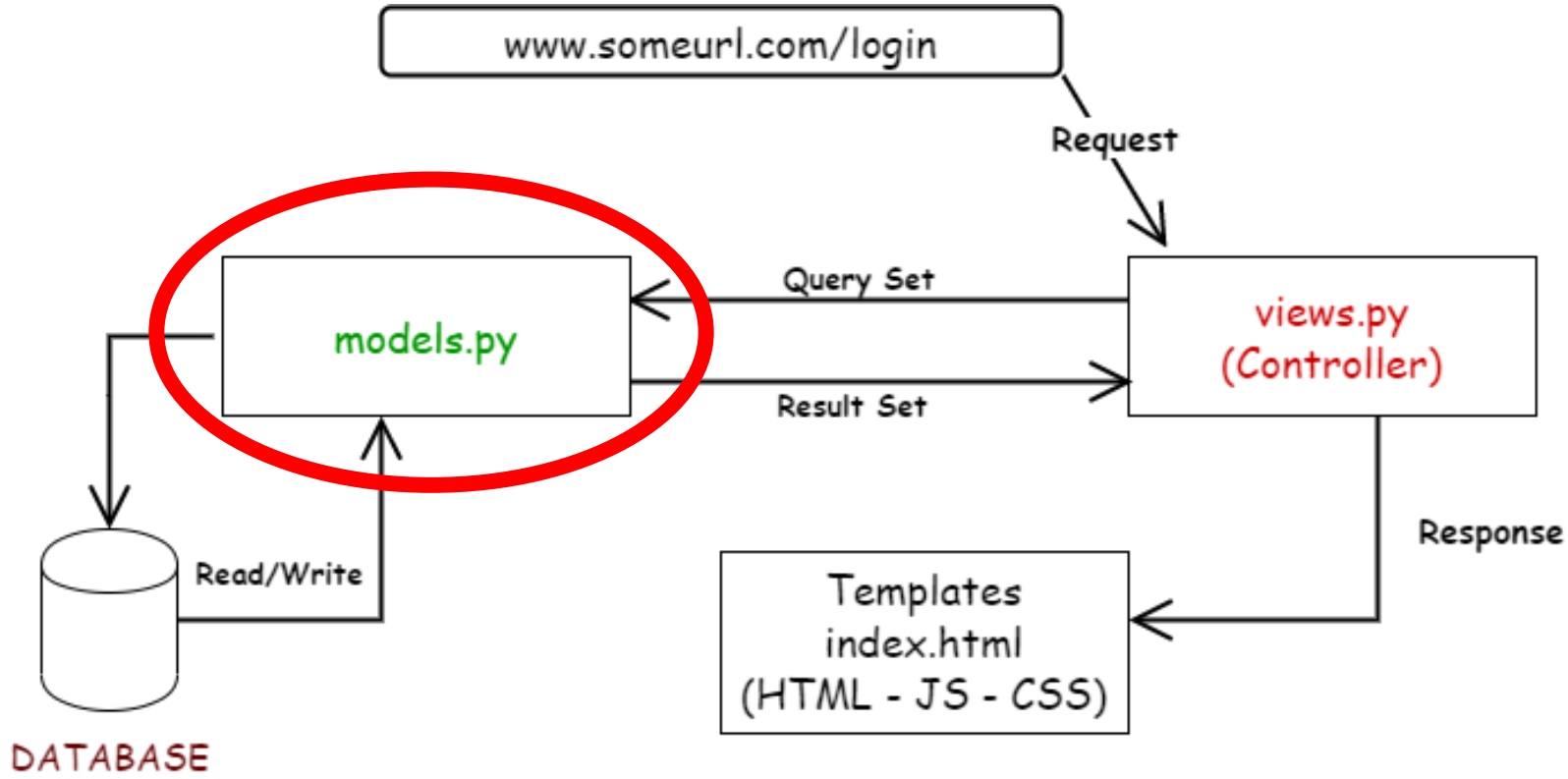
Templates

- To connect the base template and the index template together, add an extends tag to the beginning of ***blog/templates/blog/*** file.

```
blog > templates > blog > <> index.html > ...
1   {% extends 'blog/base.html' %}
2
3   {% block content %}
4       <h2>Blog Index!</h2>
5   {% endblock %}
```

- If you visit your webpage once more you will find that everything works as expected.

Models


www.someurl.com/login
Request
models.py
Query Set
Result Set
views.py
(Controller)
Read/Write
Response
Templates
index.html
(HTML - JS - CSS)
DATABASE

Models

- A model in Django is a special kind of object that is saved in the database.
- We store our models in <app_dir>/models.py => blog/models.py.
- Edit ***blog/models.py*** and write the following code.

Models

blog > models.py > ...

```python
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User



class Post(models.Model):
    author = models.ForeignKey(User, on_delete= models.CASCADE,related_name='blog_posts')
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(default=timezone.now)
    published_date = models.DateTimeField(blank=True, null=True)

    def publish(self):
        self.published_date = timezone.now()
        self.save()

    def __str__(self):
        return self.title
```

Models (Field Options)

null	Django will store empty values as **NULL** in the database. Default is **False**.
blank	If **True** the field is allowed to be blank, Default is **False**.
db_column	The name of the database column to use for this field.
default	The default value for the field.
peimary_key	If **True**, this field is the primary key for the model (null=False and unique=True).
unique	If **True**, this field must be unique throughout the table.
verbose_name	A human-readable name for the field. If it isn't given, Django will automatically create it using the field's attribute name, converting underscores to spaces

```
title = models.CharField(null=False,blank=False,default="Untitled",unique=True)
```

```
os = models.CharField(.....,verbose_name=_('Operating System'))
```

Models (Field Options)

choices	A Tuple of choices that Field value can be (The first element in each tuple is the actual value to be set on the model, and the second element is the human-readable name)

```python
from django.db import models

class Student(models.Model):
    FRESHMAN = 'FR'
    SOPHOMORE = 'SO'
    JUNIOR = 'JR'
    SENIOR = 'SR'
    YEAR_IN_SCHOOL_CHOICES = [
        (FRESHMAN, 'Freshman'),
        (SOPHOMORE, 'Sophomore'),
        (JUNIOR, 'Junior'),
        (SENIOR, 'Senior'),
    ]
    year_in_school = models.CharField(
        max_length=2,
        choices=YEAR_IN_SCHOOL_CHOICES,
        default=FRESHMAN,
    )

    def is_upperclass(self):
        return self.year_in_school in (self.JUNIOR, self.SENIOR)
```

Models (Char & Text)

CharField	A string field, for small- to large-sized strings.
EmailField	A **CharField** that checks that the value is a valid email address.
URLField	A **CharField** accepts valid urls only.
max_length	The maximum length (in characters) of the field.
TextField	A large text field

Models (Numeric & Boolean)

IntegerField	An integer. Values from -2147483648 to 2147483647.
AutoField	An **IntegerField** that automatically increments according to available IDs.
DecimalField	A fixed-precision decimal number.
max_digits	The maximum number of digits allowed in the number.
decimal_places	The number of decimal places to store with the number.
BooleanField	A true/false field.

```
models.DecimalField(..., max_digits=5, decimal_places=2)
```

Models (Date & Time)

DateField	A date, represented in Python by a **datetime.date** instance.
TimeField	A time, represented in Python by a **datetime.time** instance.
DateTimeField	A date and time, represented in Python by a **datetime.datetime** instance.
auto_now	Automatically set the field to now every time the object is saved.
auto_now_add	Automatically set the field to now when the object is first created.

Models (Relationships Field)

ForeignKey	A many-to-one relationship.
ManyToManyField	A many-to-many relationship.
OneToOneField	A one-to-one relationship.
on_delete	
to_field	

Models (Relationships Field)

```python
class artist(models.Model):
    name = models.CharField(max_length=200)

class movie(models.Model):
    title = models.CharField(max_length=100)
    artists = models.ManyToManyField(artist, related_name = 'actor', through='roles')

class role(models.Model):
    role_name = models.CharField(max_length=100)
    artist = models.ForeignKey(artist, on_delete=models.CASCADE)
    movie = models.ForeignKey(movie, on_delete=models.CASCADE)
```

Models

- To add the new model to our database, we have to let django know about our change.

  `python manage.py makemigrations blog`

```
$ python manage.py makemigrations blog
Migrations for 'blog':
  blog\migrations\0001_initial.py
    - Create model Post
```

- To apply the migration file you have just made to your database

  `python manage.py migrate blog`

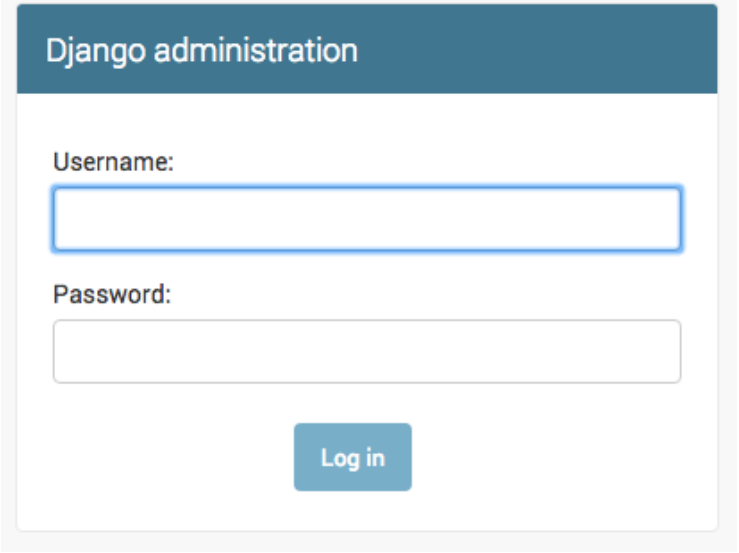
```
$ python manage.py migrate blog
Operations to perform:
  Apply all migrations: blog
Running migrations:
  Applying blog.0001_initial... OK
```

Django Admin

- Register your **Post** model in the **blog/admin.py** to be able to add, edit, delete posts through the django admin.

- Run your server and go to http://127.0.0.1:8000/admin, to login you need to create a superuser account to have control over everything on the site

```
blog > admin.py
1  from django.contrib import admin
2  from .models import Post
3
4  admin.site.register(Post)
```

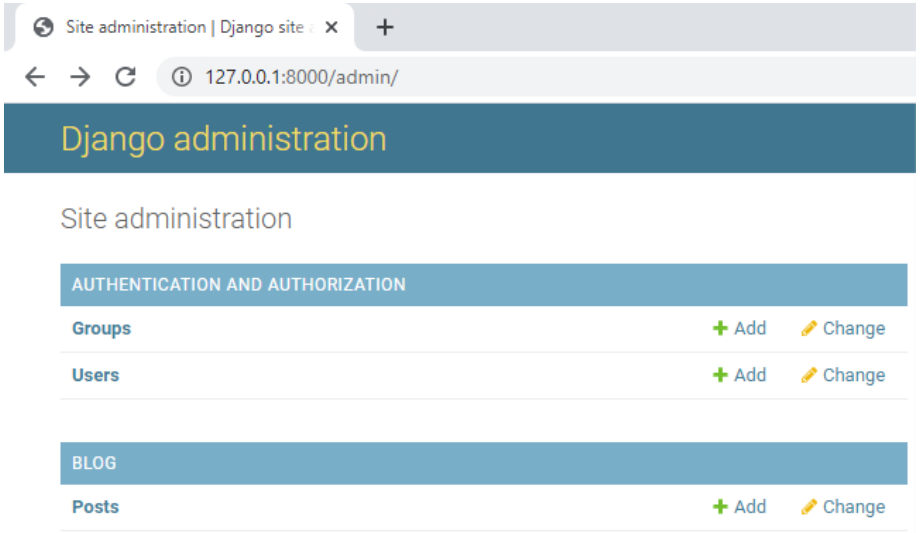


Django Admin

- Create Superuser.
  `python manage.py createsuperuser`

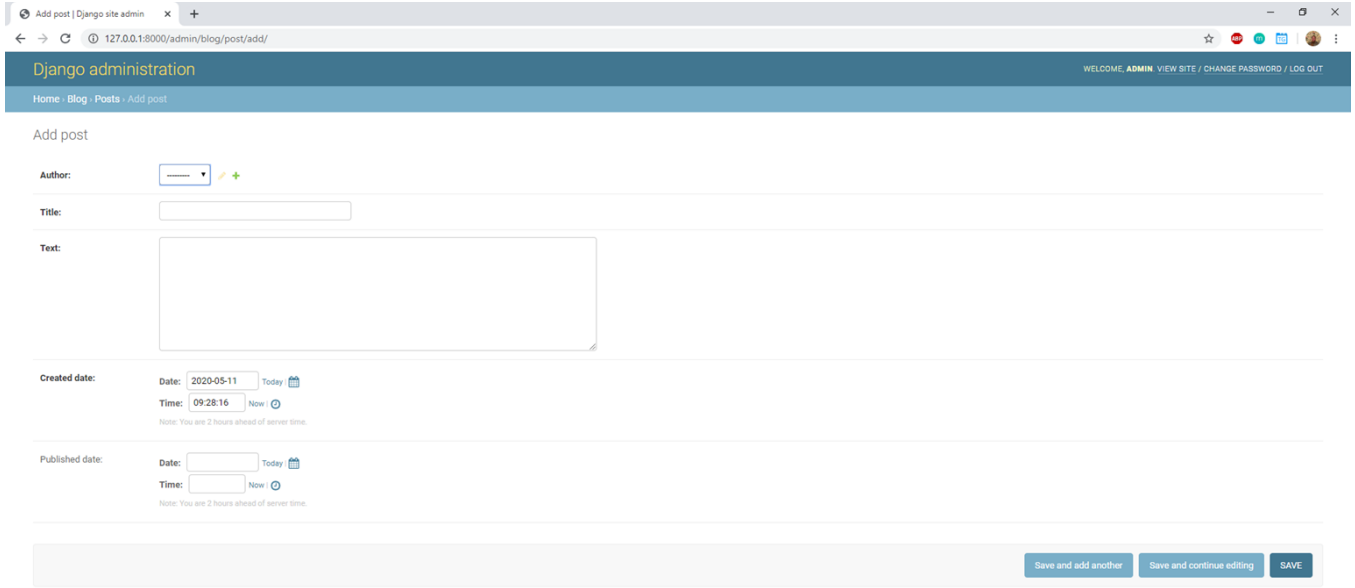
- Return to your browser and login with superuser's credentials to access django admin board.

```
$ python manage.py createsuperuser
Username (leave blank to use 'merit.ehab'): admin
Email address: admin@example.com
Password:
Password (again):
Superuser created successfully.
```



Django Admin

- Now you can go to the posts table and add some blog posts to work with them and make sure some of them has a publish date.



Django ORM & QuerySets

- QuerySets is a list of objects of a given models.
- It allows you to read data from database, filter it and order it.
- Open your console and type

  `python manage.py shell`
- Try the following queries to get to know the model operations

Django ORM & QuerySets (Insert)

```
In [1]: from blog.models import Post

In [2]: from django.contrib.auth.models import User

In [3]: user = User.objects.get(username="admin")

In [4]: Post.objects.create(author=user, title="Demo", text="Some text")
Out[4]: <Post: Demo>
```

OR

```
In [5]: post = Post(author=user, title="Demo2")

In [6]: post.text = "This is demo2 post text"

In [7]: post.save()
```

Django ORM & QuerySets (Select...Where)

```
In [8]: Post.objects.create(author=user, title="Demo", text="Hello World!")
Out[8]: <Post: Demo>

In [9]: Post.objects.all()
Out[9]: <QuerySet [<Post: First post>, <Post: The post>, <Post: Django>, <Post: Random>, <Post: The Web>, <Post: Demo>, <Post: Demo2>, <Post: Demo>]>

In [10]: Post.objects.filter(title="Demo")
Out[10]: <QuerySet [<Post: Demo>, <Post: Demo>]>

In [11]: Post.objects.get(title="Demo2")
Out[11]: <Post: Demo2>
```