

Django Framework

Merit Ehab

Agenda

- What is a framework
- Introduction to Django
- Django Installation
- Your first Django project
- Create an Application
 - Urls
 - Views
 - Templates
 - Models
- Django Admin
- Forms
- Generic Views

What is a Framework

- A set of components that helps you to develop websites faster and easier.
- You always need a similar set of components: a way to handle user authentication, forms, a way to upload files, etc.
- Frameworks exist to save you from having to reinvent the wheel and to help alleviate some of the overhead when you're building a new site.

What is Django

- A framework written in python developed at World Online by Adrian Holovaty and Simon Willison on 2003.
- Came out as open source on 2005.
- Now it runs by an international team of volunteers.
- Programming tool that helps you build websites .
- MTV framework.

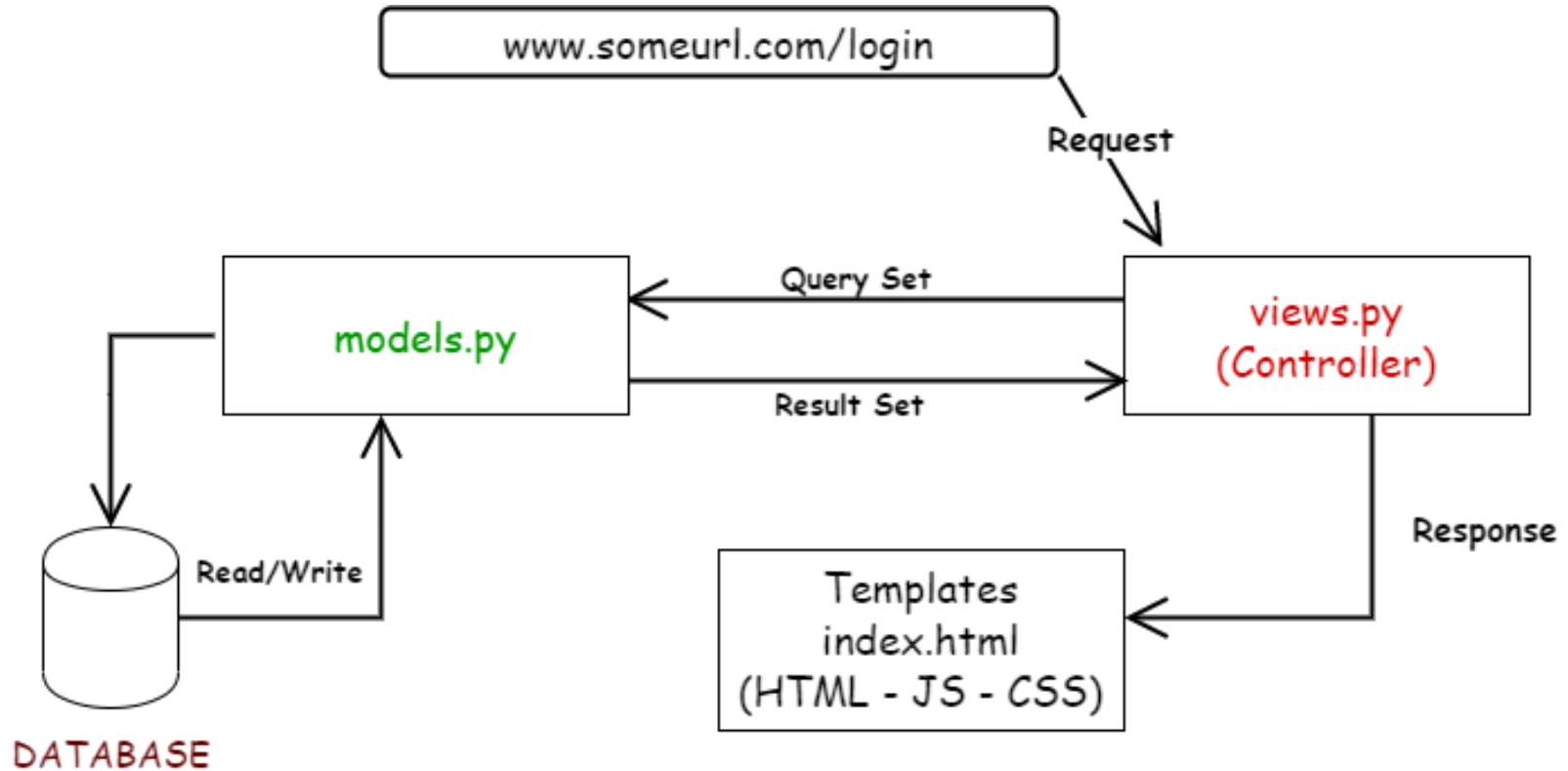
Why Django

- Very Fast.
- Fully loaded.
- Secure.
- Scalable.
- Versatile.

Who uses Django

A long list of websites uses django (Instagram, Spotify, Pinterest, Bitbucket, Prezi, Washington Post,...).

How Django works



Virtual Environment

- A tool to create an isolated environment for Python projects.
- Each project can have its own dependencies, regardless of what dependencies every other project has.

```
pip install virtualenv OR apt install python3-venv
```

```
mkdir envs && cd envs
```

```
python3 -m venv env (Linux and OS) python -m venv env (Windows).
```

OR `virtualenv env` (By default, this will not include any of your existing site packages).

```
source env/bin/activate (Linux and OS) source env/Scripts/activate  
(Windows).
```

```
deactivate (To get out of your env).
```

Django Installation

- To Install Django write this command.

```
pip install Django
```

- To install a specific version of Django.

```
pip install Django==2.2.4
```

- To know the version you have installed use this command.

```
python -m django --version
```


Your first Django project

- To create a django project use this command

```
django-admin startproject mysite . (Linux and OS).
```

```
django-admin.exe startproject mysite . (Windows).
```

- The startproject command will create a file structure in your directory that look like this.

```
mysite
├── manage.py
├── mysite
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── asgi.py
│   └── wsgi.py
```

Your first Django project

- Starting the web server.

```
python manage.py runserver
```

- If you navigate to <http://127.0.0.1:8000/> in your browser you would see something like this.

django

[View release notes for Django 3.0](#)

The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



Django Documentation
Topics, references, & how-to's



Tutorial: A Polling App
Get started with Django



Django Community
Connect, get help, or contribute

Your first Django project

- Save every package you install in the **requirements.txt** file to make it easier for other developers to install the correct version of the required packages.
- Use this command

```
pip freeze > requirements.txt
```

```
≡ requirements.txt
1  asgiref==3.2.7
2  Django==3.0.6
3  pytz==2020.1
4  sqlparse==0.3.1
```

```
mysite
├── ...
├── ...
└── requirements.txt
```

Creating an Application

- To create an application use this command

```
python manage.py startapp blog
```

(blog is your app name)


- After creating the project we should tell Django that it should use it go to **mysite/settings.py** and look for **INSTALLED_APPS**
- Add **blog.apps.BlogConfig** to the list.

- Run this command

```
python manage.py migrate
```

This will apply all the unapplied migrations on the SQLite database which comes along with the Django installation.


Project Structure



```
mysite
├── blog
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   └── views.py
├── db.sqlite3
├── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    ├── asgi.py
    └── wsgi.py
```

Urls

- Create a file called **urls.py** in your blog directory .
- Your blog directory will look like this .
- Write the following code in your urls file.

```
blog >  urls.py > ...  
1  from django.urls import path  
2  from . import views  
3  
4  
5  urlpatterns = [  
6  |      path('', views.index, name='index'),  
7  |  ]  
8
```

Url Pattern


View

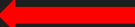
Url Name

```
|— blog  
|   |— admin.py  
|   |— apps.py  
|   |— __init__.py  
|   |— migrations  
|   |   |— __init__.py  
|   |— models.py  
|   |— tests.py  
|   |— urls.py ←  
|   |— views.py
```

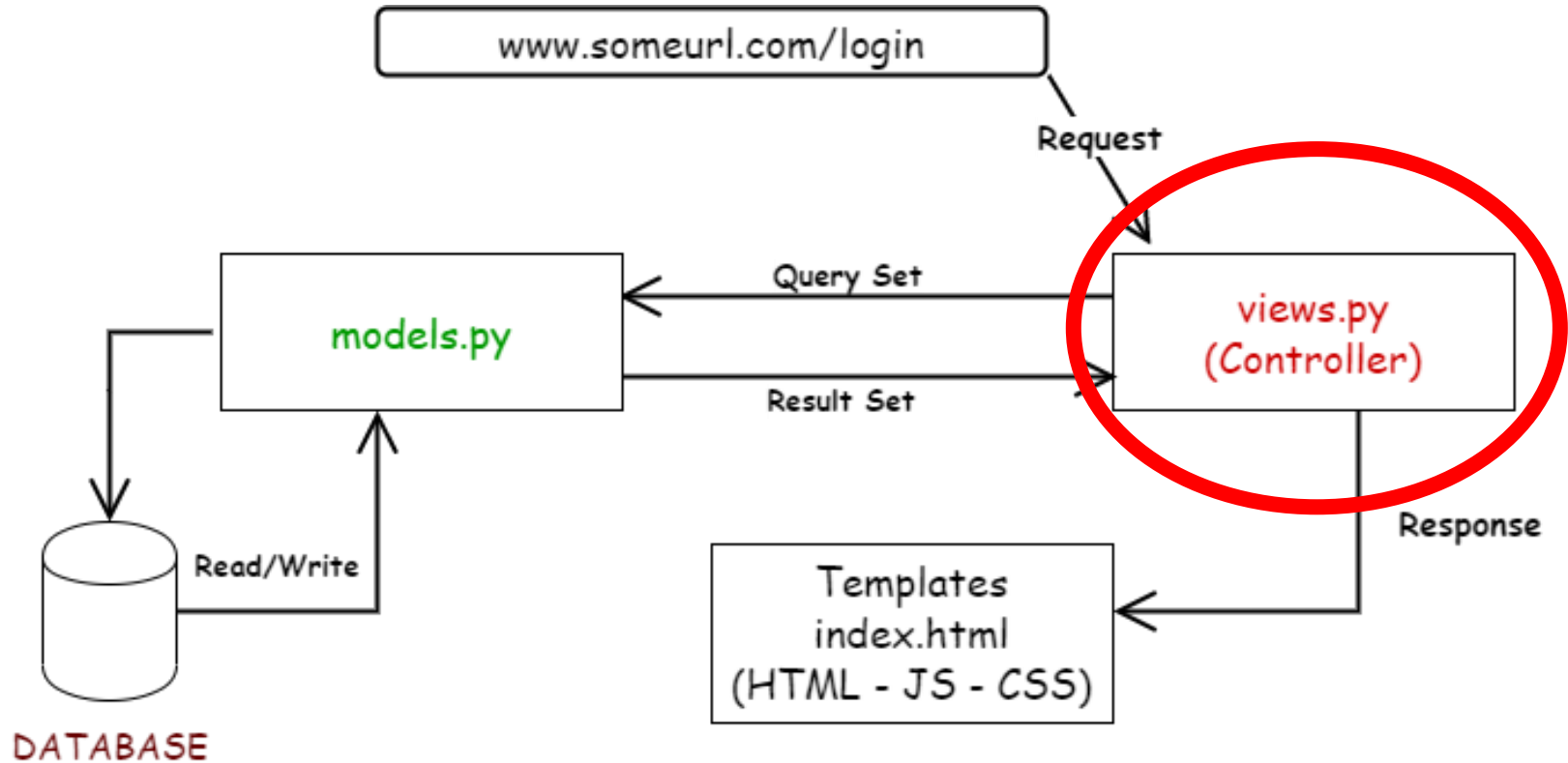
Urls

- Next you will need to make the URLconf point to the `blog.urls` module .
- Go to `mysite/urls.py`, add to your `django.urls` import **include** .
- And insert a new url pattern for your blog app using the include in the `urlpatterns`.

```
mysite >  urls.py > ...
1  """mysite URL Configuration
2
3  > The `urlpatterns` list routes URLs to views. For more information please see: ...
4
5  Examples:
6  > Function views ...
7
8  > Class-based views ...
9
10 > Including another URLconf ...
11
12 >
13 """
14
15
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('', include('blog.urls')),
22 ]
23
```



Views

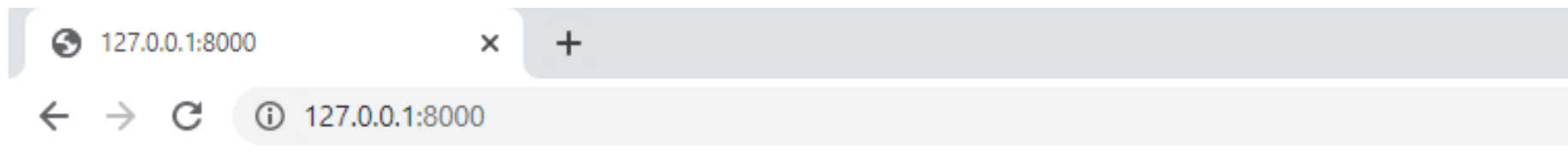


Views

blog/views.py file.

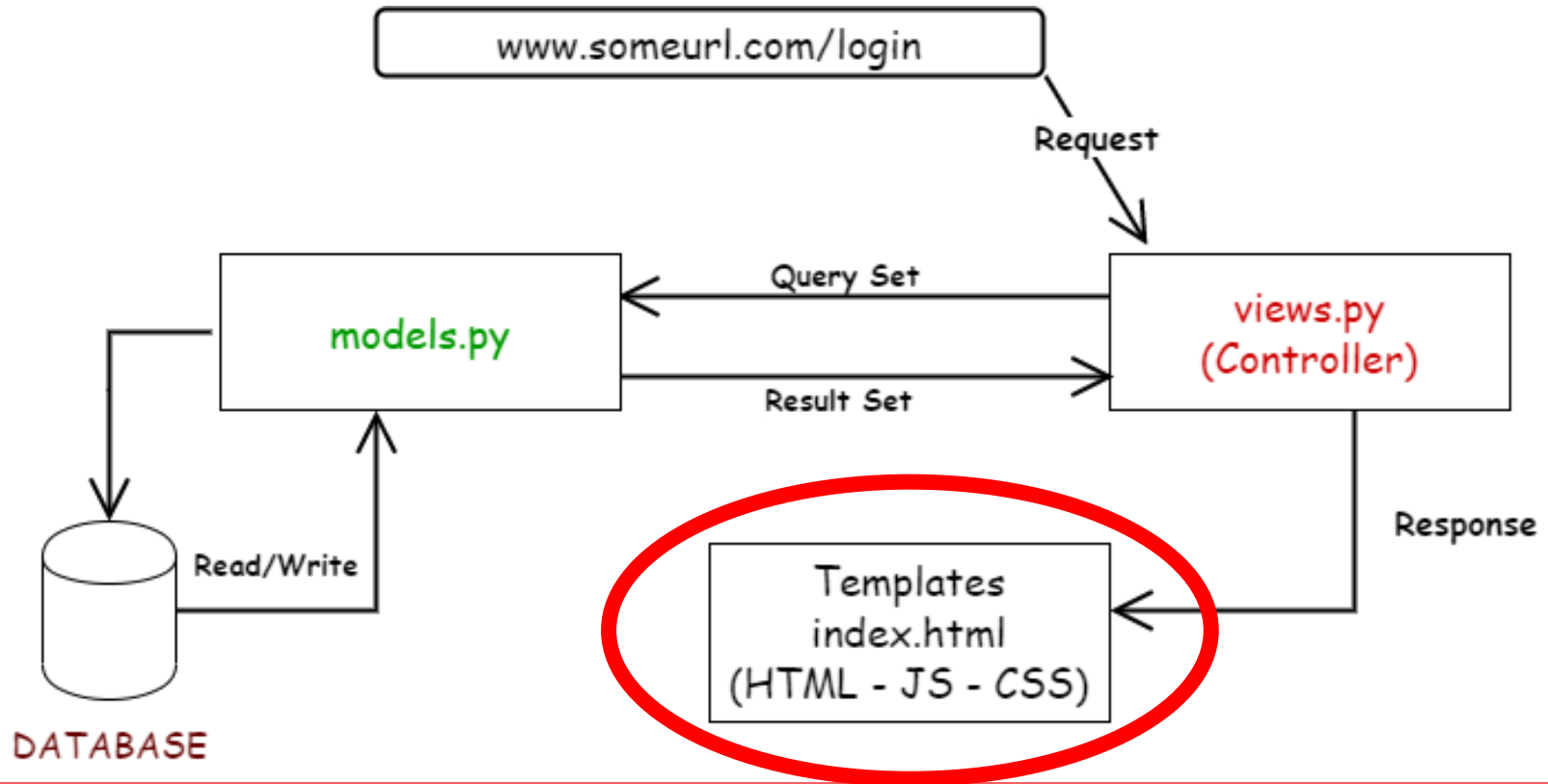
```
views.py X
blog > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4
5  def index(request):
6      return HttpResponse("Hello, Welcome to your blog index.")
7
8
```

Views



Hello, Welcome to your blog index.

Templates



Templates

- Edit **blog/views.py** change **index function**.

```
6 return render(request, 'blog/index.html', {})
```

- Now, the index function will return the render of the given template **“blog/index.html”**.

TemplateDoesNotExist at /

index.html

Request Method: GET
Request URL: http://127.0.0.1:8000/
Django Version: 3.0.6
Exception Type: TemplateDoesNotExist
Exception Value: index.html
Exception Location: C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\lib\site-packages\django\template\loader.py in get_template, line 19
Python Executable: C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\python.exe
Python Version: 3.7.0
Python Path: ['D:\\djangoCourse\\mysite',
C:\\Users\\merit.ehab.FLAIRSTECH\\AppData\\Local\\Programs\\Python\\Python37\\python37.zip',
C:\\Users\\merit.ehab.FLAIRSTECH\\AppData\\Local\\Programs\\Python\\Python37\\DLLs',
C:\\Users\\merit.ehab.FLAIRSTECH\\AppData\\Local\\Programs\\Python\\Python37\\lib',
C:\\Users\\merit.ehab.FLAIRSTECH\\AppData\\Local\\Programs\\Python\\Python37',
C:\\Users\\merit.ehab.FLAIRSTECH\\AppData\\Local\\Programs\\Python\\Python37\\lib\\site-packages']
Server time: Sat, 9 May 2020 14:19:03 +0000

Template-loader postmortem

Django tried loading these templates, in this order:

Using engine django:

- django.template.loaders.app_directories.Loader: C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\lib\site-packages\django\contrib\admin\templates\index.html (Source does not exist)
- django.template.loaders.app_directories.Loader: C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\lib\site-packages\django\contrib\auth\templates\index.html (Source does not exist)

Traceback [Switch to copy-and-paste view](#)

C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\lib\site-packages\django\core\handlers\exception.py in inner

34. response = get_response(request)

► Local vars

C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\lib\site-packages\django\core\handlers\base.py in _get_response

115. response = self.process_exception_by_middleware(e, request)

► Local vars

C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\lib\site-packages\django\core\handlers\base.py in _get_response

113. response = wrapped_callback(request, *callback_args, **callback_kwargs)

► Local vars

D:\djangoCourse\mysite\blog\views.py in index

6. return render(request, 'index.html', {})

► Local vars

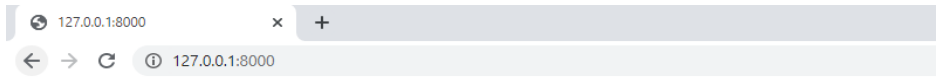
C:\Users\merit.ehab.FLAIRSTECH\AppData\Local\Programs\Python\Python37\lib\site-packages\django\shortcuts.py in render

19. content = loader.render_to_string(template_name, context, request, using=using)

► Local vars

Templates

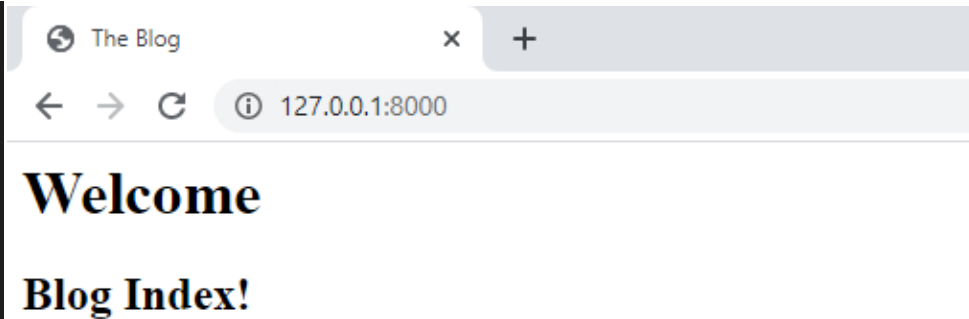
- Templates are saved in `<app_dir>/templates/<app_name> => (blog/templates/blog)`.
- Create an empty file **blog/templates/blog/index.html**.



Templates

Edit `mysite/templates/blog/index.html`.

```
1 <html>
2   <head>
3     <title>The Blog</title>
4   </head>
5   <body>
6     <h1>Welcome</h1>
7     <h2>Blog Index!</h2>
8   </body>
9 </html>
```



Templates

- At the end we want to have a blog and to see in our index page some posts with a title and link to read the full post, something to tell us when it was published and so on, at minimum we want to see something like this in our browser.



Templates

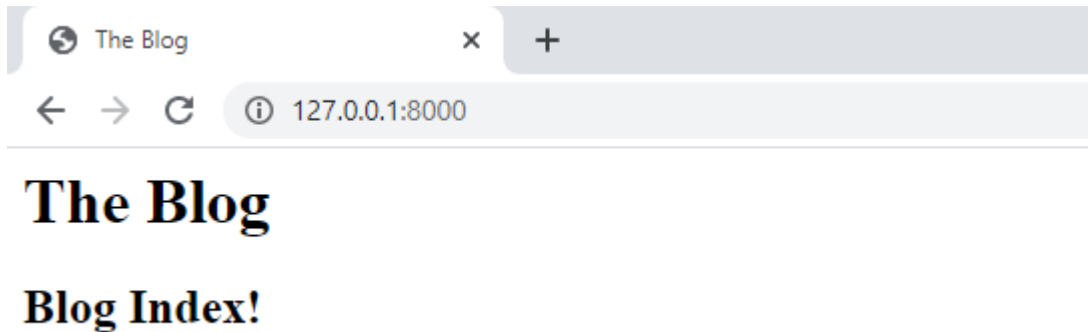
- *Django template tags* allow us to transfer Python-like things into HTML, so you can build dynamic websites faster.
- Edit **blog/views.py**.

```
5  ✓ def index(request):  
6      text = "The Blog"  
7      return render(request, 'blog/index.html', {"text":text})  
8
```

Templates

- In `templates/blog/index.html`
- Noticed `{{text}}` this is what is called a *Template Tag*.

```
1 <html>
2   <head>
3     <title>The Blog</title>
4   </head>
5   <body>
6     <h1>{{text}}</h1>
```

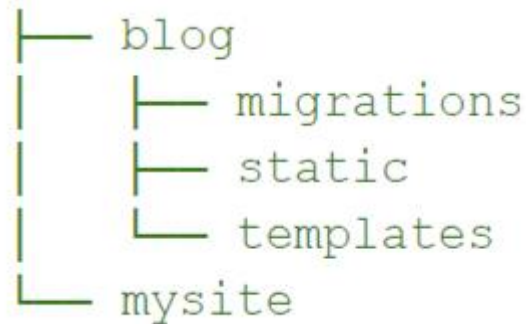


Templates

- To install Bootstrap, in your `.html` file add this to the `<head>` section.

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
```

- **Static files:** are all your CSS and images.
- Create a **static** directory in your blog app.
- Django will automatically find any folders called "static" inside any of your apps' folders, Then it will be able to use their contents as static files.



Templates

- Create a directory called `css` ins your `static` directory. Then a file called `blog.css` .

- In `blog/static/css/blog.css`.

```
blog > static > css > # blog.css > ...  
1  h1 , h2 {  
2  |    color: ■rgb(7, 134, 50);  
3  }
```

```
└─ blog  
   └─ static  
      └─ css  
         └─ blog.css
```

- In `blog/templates/blog/index.html` add this line at the very beginning of it.
`{% load static %}`

Templates

- Then import your css file in the head section after the bootstrap css file

```
<link rel="stylesheet" href="{% static 'css/blog.css' %}">
```

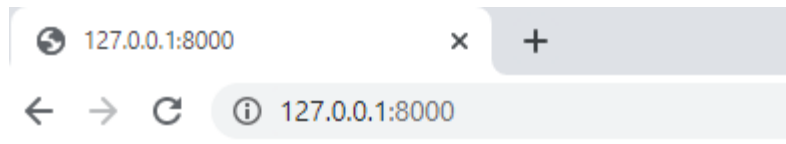


Templates

- 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.