

Introduction

Installation

Settings Module

Requests and Response

Running development server

Django admin site introduction

Installation

- ▶ Python Installation
- ▶ Django Installation

```
pip install django
```

Settings Module

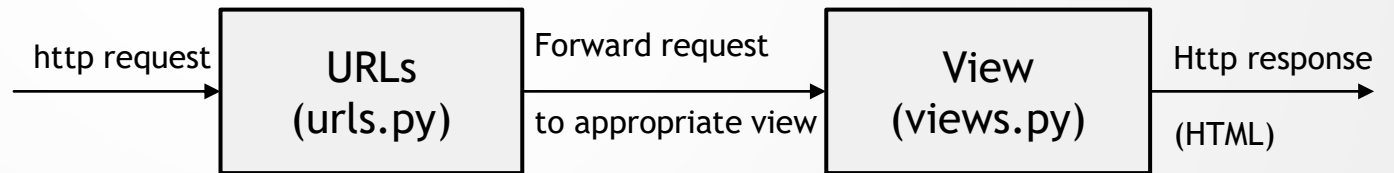
- ▶ settings.py - why these variables and values are used

Separating dev, prod, test environments

- ▶ Create a folder called settings or config
- ▶ Move the settings.py to that dir
- ▶ Create a __init__.py
- ▶ Create a dev.py, import settings to it, overwrite the values in dev.py
- ▶ Repeat the same for prod.py
- ▶ In settings.py, point the base dir one step up
- ▶ In wsgi.py and manage.py, point the settings to dev
- ▶ Run the server

Requests & Response

- ▶ When a url is striked in the web browser, *urls.py* is involved to map the path to a view (views.py)
- ▶ *views.py* is responsible to send a response HTML



Requests & Response

urls.py

```
from .views import index

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index),
]
```

views.py

```
from django.http import HttpResponse

def index(request):
    return HttpResponse('<b>Hello world<b>')
```

Running development server

- ▶ Make a new project directory
- ▶ Start the project

```
django-admin startproject mysite
```

- ▶ Run the server

```
python manage.py runserver
```

Django admin site introduction

- ▶ Apply / enable the default app(s) by migrate

```
python manage.py migrate
```

- ▶ Create a super user

```
python manage.py createsuperuser
```

- ▶ Run the server and play around admin site

```
python manage.py runserver
```

Template Layer

Overview of template language

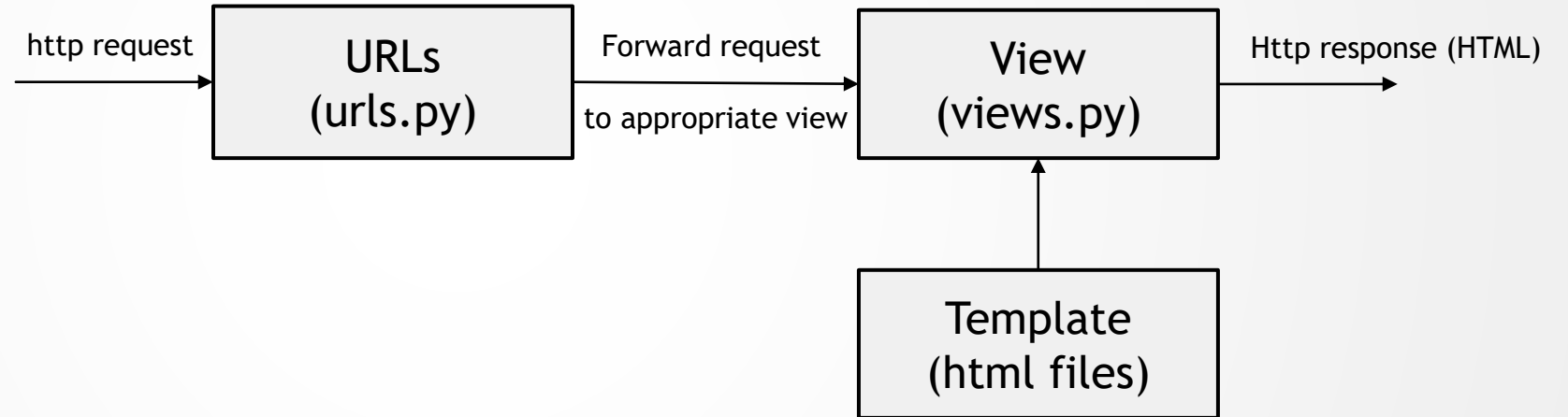
Built-in tags and filters

Humanization

custom tags and filters

csrf token

Overview of template language



Overview of template language

- ▶ Start an app

```
>python manage.py startapp members
```

- ▶ Add app to Installed Apps
- ▶ In project directory, in settings.py, add the newly created members

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'members'  
]
```

Overview of template language

- ▶ In app directory, create a folder called **templates** → backend → index.html
- ▶ Edit index.html

index.html

```
<html >

<style>
body {
    background-color: lightblue;
}
</style>
<body>
    hey
</body>
```

Overview of template language

- ▶ Edit members/views.py

views.py

```
def url1(request):  
    return render(request,  
        'backends/blog.html')
```

- ▶ Add path to to project's urls.py

urls.py

```
from members.views import page  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('members/', page)  
]
```

Overview of template language

► Makemigrations

```
>python manage.py makemigrations
```

► Migrate

```
>python manage.py migrate
```

► Run server

```
>python manage.py runserver
```

Built-in tags and filters

- ▶ For

- ▶ If

- ▶ Block

```
{% block content %}
```

```
{% endblock %}
```

- ▶ Extends

```
{% extends 'backend/base.html' %}
```

Built-in tags and filters

► Passing context from views.py

views.py

```
from django.shortcuts import render

# Create your views here.

context = {'data': [

    {'name': 'jessie', 'department': 'IT', 'count': 1},
    {'name': 'malini', 'department': 'Chem'}

]}

def page(request):
    return render(request, 'backend/index.html',
context)
```

Built-in tags and filters

► Using for and if

index.html

```
<h1> my app </h1>
<b> data is </b>

{{ data }}

<b> elements in data </b>

{% for student in data %}
{% if student.name == 'jessie' %}
{{ student.count }}
{% endif %}
{% endfor %}
```


Templates

Built-in tags and filters

- ▶ Add
- ▶ Capfirst

Additional Reference - [official doc](#)

Templates Built-in tags and filters

► Using add

index.html

```
<b> elements in data </b>
```

```
{% for student in data %}
```

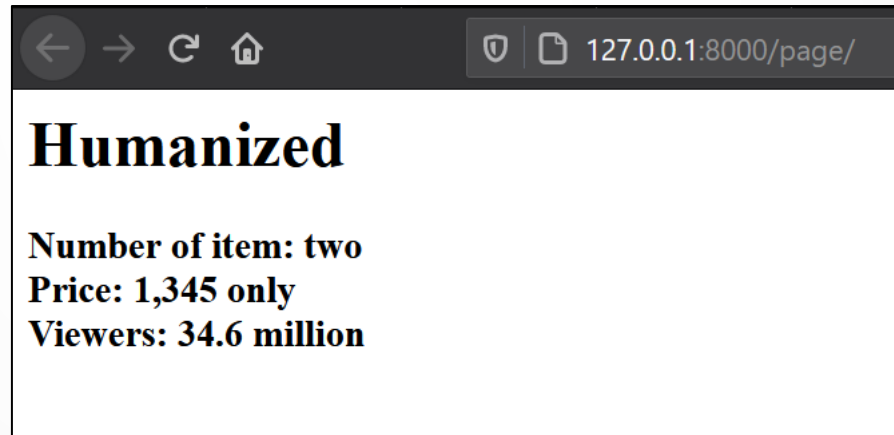
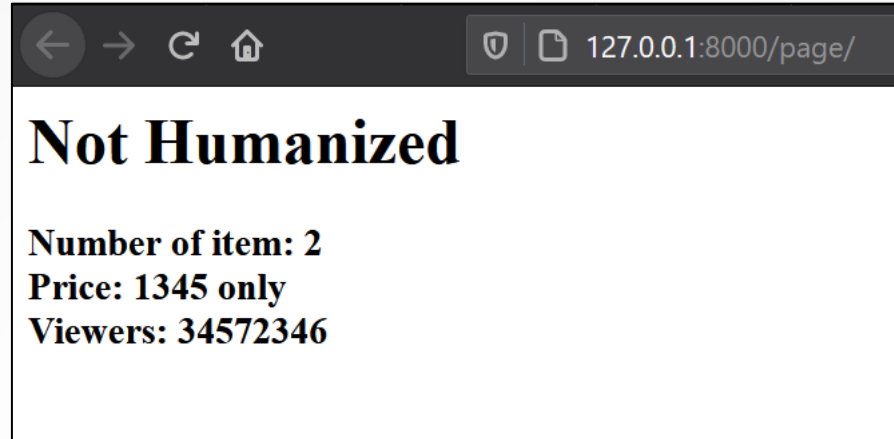
```
{% if student.name == 'jessie' %}
```

```
{{ student.count|add:1 }}
```

```
{% endif %}
```

```
{% endfor %}
```

Humanization



Ref:

<https://docs.djangoproject.com/en/3.1/ref/contrib/humanize/>

Humanization

- ▶ In Installed apps add → *django.contrib.humanize*
- ▶ Add context in *views.py*

views.py

```
from django.shortcuts import render

# Create your views here.
context = {
    'num_of_item': '2',
    'price': 1345,
    'viewers': 34572346
}

def page(request):
    return render(request,
        'backend/index.html', context)
```

Humanization

► Edit *index.html*

index.html

```
<h1> My Page </h1>

{% load humanize %}

<h3>
    Number of item:
    {{ num_of_item|apnumber }}
    <br>

    Price:
    {{ price|intcomma }} only
    <br>

    Viewers:
    {{ viewers|intword }}
    <br>

</h3>
```

custom tags and filters

-

Tags

- ▶ Create a directory called *templatetags* in app directory
- ▶ Create an empty *__init__.py* in templatetags to treat the directory as a python package
- ▶ Create any py file in templatetags, in this slide we are taking it as *mycustomtags.py*

```
myproject
├── db.sqlite3
├── manage.py
├── myapp
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── views.py
│   └── __init__.py
├── migrations
├── templatetags
│   ├── mycustomtags.py
│   └── __init__.py
```

custom tags and filters

-

Tags

- ▶ Edit mycustomtags.py

mycustomtags.py

```
from django import template

register = template.Library()

@register.simple_tag
def count_list(lst):
    return len(lst)
```

- ▶ Add to `INSTALLED_APPS` →
`'myapp.template_tags.mycustomtags'`

custom tags and filters

- Tags

► Context in *views.py*

views.py

```
context = {  
    'data' : [1,2,3,4]  
}
```

► In *index.html*

index.html

```
<h1> Custom Tags </h1>  
  
{% load mycustomtags %}  
  
{% count_list data %}
```


custom tags and filters

-

filters

- ▶ Create a new python file for a custom filter - example *mycustomfilters.py*

mycustomfilters.py

```
from django import template

register = template.Library()

@register.filter
def cut(value, arg):
    return value.replace(arg, '')
```

- ▶ Add the custom filter to *INSTALLED_APPS* → *'myapp.template_tags.mycustomfilters'*

custom tags and filters

- filters

- ▶ Make sure you have a string value in *context* in *views.py*

views.py

```
context = {  
    'data' : [1,2,3,4],  
    'name' : 'malini'  
}
```

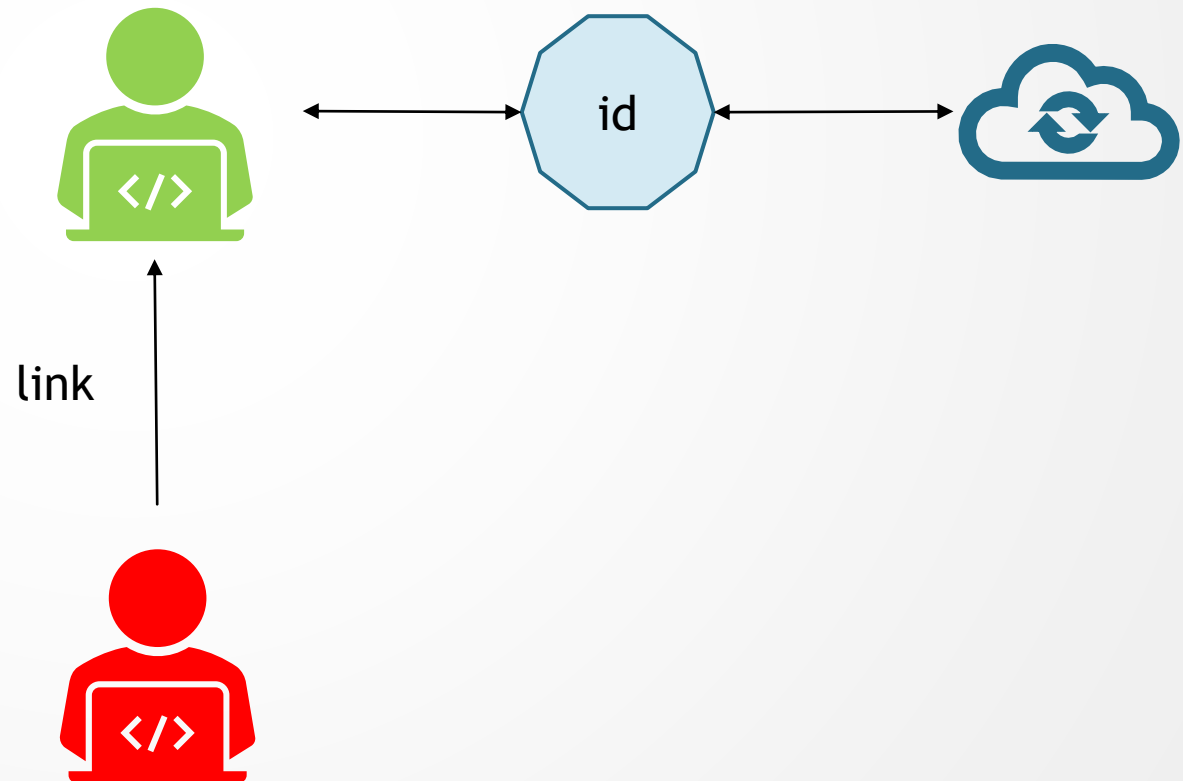
- ▶ In *index.html*

index.html

```
<h1> Custom Filters </h1>  
  
{% Load mycustomfilters %}  
  
{{ name|cut:'a' }}
```

Csrf token

- ▶ Csrp - Cross site request forgery
- ▶ To know what's csrf - <https://www.youtube.com/watch?v=hW2ONyxAySY>



View Layer

View functions

URL confs

Shortcuts and decorators

Request and response objects

File upload

Class based views

Mixins

Generating csv and pdf

View functions

- ▶ In project directory, in *views.py*

views.py

```
from django.http import HttpResponse

def page(request):
    html = "<html><body>Hello there !
</body></html>"
    return HttpResponse(html)
```

URL confs

- ▶ In project directory, in *settings.py*, ***ROOT_URLCONF*** where Django sees the urls patterns
- ▶ In app directory, create a new py file to store the app urls. Example: ***myappurls.py***

myappurls.py

```
from django.urls import path

from .views import page

urlpatterns = [
    path('page/', page),
]
```

URL confs

- ▶ In project directory, *urls.py*

urls.py

```
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.myappurls'))
]
```

- ▶ Try editing project dir settings.py urlconf
- ▶ Undo ! 😊

Shortcuts and decorators

- ▶ `render`
- ▶ `redirect()`
- ▶ `get_object_or_404()`
- ▶ `get_list_or_404()`

Official Doc Ref:

<https://docs.djangoproject.com/en/3.1/topics/http/shortcuts/>

Shortcuts and decorators

- ▶ Using redirect
- ▶ In app dir, *views.py*

views.py

```
from django.shortcuts import render, redirect

def page2(request):
    return redirect('../page/')
```

- ▶ In *urls.py*

urls.py

```
from .views import page, page2

urlpatterns = [
    path('page/', page),
    path('page2/', page2)
]
```

Request and response objects

- ▶ `HttpRequest.scheme`
- ▶ `HttpRequest.path`
- ▶ `HttpRequest.method`
- ▶ `HttpRequest.is_secure()`
- ▶ `HttpRequest.user >> request.user.is_authenticated`

More:

<https://docs.djangoproject.com/en/3.1/ref/request-response/>

- ▶ Inclusion tag - on completion of models
- ▶ Csrf - after forms or post get methods
- ▶ Views decorators - after get post methods

To cover