# https://www.youtube.com/watch?v=F5mRW0jo-U4&themeRefresh=1

# Settings.py:

Middleware: Dit is voor beveiliging van de website.

INSTALLED\_APPS: Hier zet je je eigen en third party apps in.

# starten

createsuperuser

Sterk wachtwoord nodig voor live code

Elke app moet echt 1 ding doen,

# models

Registreer je model in admin.py om model te zien in django admin.

Elke field kan een default krijgen.

In python shell kan je ook objecten voor je model maken.

python manage.py shell

from app.models import model

product.object.all() (geeft een queryset)

products.objects.create(<dingen>)

Als je de model en opgeslagen objecten niks vind kan je alles verwijderen door in de pycache en migrations map en je database te verwijderen.

Documentatie: Field-types

Blank=True, null=True, default=

Blank geeft aan hoe django admin het weergeeft (bold is requered?).

Null is de dataabase waarde.

Kijk lekker in de docs om te weten wat elk field nodig heeft.

In shell kan je alle waarden in 1 keer veranderen als je een var type veranderd in je model.

# views

Altijd het woord view in al je view functions zetten.

Respons: gooit iets naar frontend

Kwars = keyword args = dict

Input view: Request, \*args, \*\*kwargs

# urls

Takes request, template, dict

# templates

Hier zet path van templates in: settings.py > TEMPLATES > DIRS

os.path.join(BASE\_DIR, ‘templates/’)

Om template te laten werk moet view dit returnen.

return render(request, “<template>.html”, {})

## template engine

Kan je boel modulair mee maken, zoals een menu bar.

Je maakt een blue page (blue print, lol).

Base.html

<constante zooi>

{% block content %}

{% endblock %}

<constante zooi>

{% block whateverthefuck %}

{% endblock %}

Home.html

{% extends ‘base.html’ %}

{% block content %}

<alle zooi die je dan tussen base.html wil hebben>

{% endblock %} (of {% endblock content %})

{% block whateverthefuck %}

<constante shit>

{% endblock %}

# include template tag

Als je html hebt gemaakt met 1 constant stukje (navbar), dan kan je die in elke html zetten met:

<html shizzle>

{% include ‘navbar.html’ %}

<html shizzle>

Globale temples zet je buiten alle apps in een template folder

# context

Dictionary met zooi er in.

In dict:

{“mytext”: “wjhazaal”}

In de html:

{{ mytext }}

# for loops in template

{% for my\_sub\_item in my\_list %}

{{ my\_sub\_item }}

{% endfor %}

LET OP DAT ER IN JE CONTEXT NIET EEN KEY ZIT DIE my\_sub\_item HEET!!!!

{{ forloop.counter }} geeft de index van het huidige element in de loop.

{% for my\_sub\_item in my\_list %}

{{ forloop.counter }} {{ my\_sub\_item }}

{% endfor %}

# condictions in template

{% if var == 222 %}

{% elif var == 3 %}

{% else %}

{% endif %}

{% if var == True %}

# template tag filter

Een template tag gevolgd door een pipe en dan argumenten met waarden.

{{ var|filter1:waarde|filter2:waarde|filter3:waarde }}

{{ abcvar|add:22 }} // this adds 22 to the variable (temporirly?)

Het ding tussen {% <ding> %} is een template tag.

**Er zijn veel template tags! Belangrijk! Kijk ze allemaal door!**

[Built-in template tags and filters | Django documentation | Django (djangoproject.com)](https://docs.djangoproject.com/en/5.0/ref/templates/builtins/)

Kan je ook dingen bold maken, capitalize en meer. Heb je minder html nodig.

Wordt altijd gerenderd als tekst.

|safe zet tekst om in pure html.

# render data form database with a model

X = <model naam>.objects.get(id=<id>)

X.<naam van colom>

**Beste manier om je templets in the delen is in een map buiten je app.**

* **Templates/app1/home.html**
* **Templates/app2/home.html**
* **Templates/app3/home.html**
* **Etc.**

**In je view roep je hem dan aan als “/app1/home.html”.**

Je kan gewoon je object meegeven in de context dict. In de html kan je dan het object aanroepen.

# how django templates load with apps

Je kan ook in je app de de html zetten. Als de app heet app1, dan gooi je de html in de folder:

/app1/templates/app1/home.html

DEZE OVERSCHRIJFT DAN /templates/app1/home.html!!!

Handig om de boek modulair te maken.

Maar de meeste mensen gooien app templetes in de app folder.

# django model forms

Die maak je in forms.py en laad je in bij views.py als from .forms import ProductForm.

form = ProductForm(request.POST or None)

geef je form mee in je content dictionary.

In je html template kan je de form zo invoeren:

<form method=’POST’>

{% csrf\_token %}

{{ form.as\_p }}

<input type =’submit’ value=’Save’>

</form>

Je maakt een form leeg na opslaan door een nieuwe form aan te maken.

form = ProductForm(request.POST or None)

if form.is\_valid():

form.save()

form = ProductForm()

# raw html form