- Title: Django 3 Referentie Kaart
- Door: Dion Dresschers
- Doelpubliek: Iedereen
- Inspiratie: Python Django Tutorial
- Licentie: CC BY-SA 4.0
- Versie: 2020-09-23 09:25:38

Basis

Basis - Besturingssysteem

- controlleer het besturingssysteem

```
cat /etc/*release* | grep '^PRETTY_NAME'
```

- resulteert in:

```
PRETTY_NAME="Ubuntu 20.04.1 LTS"
```

Basis - Installeer pip

Installeer pip: `apt install python3-pip`.

Controlleer je pip versie:

- `python3 -m pip --version`

Basis - Installeer venv

- `sudo apt install python3-venv`

Basis - Maak een virtuele omgeving aan

Maak een viruele omgeving aan genaamd `opendib2` in een gewenste directory:

- `python3 -m venv opendib2`

Ga naar die virtuele omgeving

- `cd opendib2`

Activeer die virtuele omgeving:

- `source bin/activate`

Basis - Maak je prompt kleiner en gekleurd

1. Maak je prompt kleiner met `PS1='\W\$ '`, wat resulteert in:

- `PS1='\W $ '`

1. Maak nu optioneel je prompt gekleurd met bijvoorbeedl groen '32' en de rest van de tekst weer wit '37'.

- `PS1='\[\033[32m\]\W $ \[\033[37m\]'`

Django

Django - Basisinstallatie

1. Installeer Django in je viruele omgeving:

    - `pip install django`

    wat resulteert in:

```
Collecting django
Downloading Django-3.1.1-py3-none-any.whl (7.8 MB)
    |████████████████████████████████| 7.8 MB 2.3 MB/s
Collecting asgiref~=3.2.10
Downloading asgiref-3.2.10-py3-none-any.whl (19 kB)
Collecting sqlparse>=0.2.2
Downloading sqlparse-0.3.1-py2.py3-none-any.whl (40 kB)
    |████████████████████████████████| 40 kB 2.5 MB/s
Collecting pytz
Downloading pytz-2020.1-py2.py3-none-any.whl (510 kB)
    |████████████████████████████████| 510 kB 856 kB/s
Installing collected packages: asgiref, sqlparse, pytz, django
Successfully installed asgiref-3.2.10 django-3.1.1 pytz-2020.1 sqlparse-
0.3.1
```

2. Controlleer je Django versie met

    - `python3 -m django --version`

    wat resulteert in:

    - `3.1.1`

3. Bekijk alle commandos van 'django-admin' met:

    - `django-admin`

    wat resulteert in:

```
Type 'django-admin help <subcommand>' for help on a specific subcommand.

Available subcommands:

[django]
    check
```

```
    compilemessages
    createcachetable
    dbshell
    diffsettings
    dumpdata
    flush
    inspectdb
    loaddata
    makemessages
    makemigrations
    migrate
    runserver
    sendtestemail
    shell
    showmigrations
    sqlflush
    sqlmigrate
    sqlsequencereset
    squashmigrations
    startapp
    startproject
    test
    testserver
Note that only Django core commands are listed as settings are not properly
configured (error: Requested setting INSTALLED_APPS, but settings are not
configured. You must either define the environment variable
DJANGO_SETTINGS_MODULE or call settings.configure() before accessing
settings.).
```

4. Maak een 'project' aan met de naam 'project_opendib2':
    - `django-admin startproject project_opendib2`
5. Bekijk die inhoud van dat project:
    - `tree project_opendib2/`

    resultaat:

```
project_opendib2/
├── manage.py
└── project_opendib2
    ├── asgi.py
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

1 directory, 6 files
```

Django - Django files

1. Je kan met `manage.py` commando's uitvoeren.

2. Bekijk die `manage.py` file:

    - `cat manage.py`

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'project_opendib2.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

3. Ga naar de `project_opendib2` directory:

    - `cd project_opendib2`

4. De `__init__.py` is leeg. Dit geeft aan dat het dat het een Python package is.

    - `cat __init__.py`

```

```

5. In de `settings.py` kan je instelling en configuties aanpassen. Bekijk deze file:

    - `cat settings.py`

```
"""
Django settings for project_opendib2 project.

Generated by 'django-admin startproject' using Django 3.1.1.
```

```
For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'cau6@aegbym2*id2u8$8=k2i905aw(*xpq!x)f@z2+dl^105_%'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

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

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'project_opendib2.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
```

```
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'project_opendib2.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'
```

```
USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'
```

6. In de `urls.py` file zet je URL verbindingen:
    - `cat urls.py`

```
"""project_opendib2 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please
see:
    https://docs.djangoproject.com/en/3.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path

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

7. Zie dat als je naar de url `/admin` achter je URL zet, je gestuurd wordt naar `admin.site.urls`

- cat urls.py | grep -A2 'urlpatterns ='

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

1. Bekijk de `wsgi.py` file waar je normaliter geen veranderingen in maakt. Deze file is de manier waarop het django project en de web server met elkaar communiceren.
    - `cat wsgi.py`

```
"""
WSGI config for project_opendib2 project.

It exposes the WSGI callable as a module-level variable named
``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'project_opendib2.settings')

application = get_wsgi_application()
```

Django - Testserver

1. Je kan het web project starten door het `manage.py` commando `runserver`:
    - `../manage.py runserver`

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until
you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
September 23, 2020 - 07:23:25
Django version 3.1.1, using settings 'project_opendib2.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

2. Open nu je favoriete browser de URL http://127.0.0.1:8000
3. Je ziet ook dat je in die `/admin` URL achter de URL kan zetten om in het admin gedeelte te komen via http://127.0.0.1:8000/admin
4. Je moet nu een nieuwe terminal gebruiken op verder te kunnen werken in de terminal.

5. Wil je de testserver stoppen gebruik dan deze toetscombinatie:
    - `<CTRL> + <C>`

TODO_DD vanaf hier gaat het fout. Maak de App aan in de folder waar ook 'manage.py' zit.

Django - App, maak een losse 'app' aan in je project

Je hebt reeds een 'project' gemaakt. In dat project kan je meerdere losse 'apps' maken. Dit is een goede methode om de verschillende onderdelen in je 'project' op te splitsen. Je kan zelfs die losse 'app' in andere projecten gaan gebruiken.

1. Maak een nieuwe app aan genaam 'blog':
    - `../manage.py startapp blog`

```
project_opendib2 $ tree
.
├── asgi.py
├── blog
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── __init__.py
├── __pycache__
│   ├── __init__.cpython-38.pyc
│   ├── settings.cpython-38.pyc
│   ├── urls.cpython-38.pyc
│   └── wsgi.cpython-38.pyc
├── settings.py
├── urls.py
└── wsgi.py

3 directories, 16 files
```

Django - View

1. Bekijk de `views.py` file:
    - `cat blog/views.py`

```
from django.shortcuts import render

# Create your views here.
```

2. We gaan die file nu aanpassen zodat een HTTP Response terug gegeven kan worden. Daarvoor moet je wel eerst de `HttpResponse` importeren in de file.

Dan maak je een functie aan (bijvoorbeeld genaamd `home`) en daarin `return` je een `HttpResponse` in HTML. Zorg dat je file er nu zo uit ziet:

```
```
from django.shortcuts import render # deze stond er al
from django.http import HttpResponse # deze hebben we zelf toegevoegd

Create your views here.

def home(request): # de functie hebben we `home` genoemd
    return HttpResponse('<h1>Blog Thuisbasis</h1>')

Er wordt HTML geretourneerd)
```
```

1. Dan moet je de URL nog mappen naar de functie die je net gemaakt hebt. Maak eerst de file `urls.py` aan in be `blog` directory. Let op dat er al een `urls.py` bestaat in het 'project' (dus niet in de 'app').
    - `touch blog/urls.py`
2. Deze `blog/urls.py` heeft elementen die lijken op die van de 'project' `urls.py` in de bovenliggende directory. Zorg dat dit het resultaat is:
    - `cat blog/ulrs.py`

```
from django.urls import path
from . import views # van de huidige directory importeer de `views.py`-file

urlpatterns = [
    path(''), views.home, name='blog-home'),
]

# de `views` refereert naar de `views.py`-file.
# de `home` is de functie `home` die we in `views.py` aangemaakt hebben
# de `path` is leeg, dus als er geen /url achter komt gebruik deze view
# de `name` is handig om naar te referen
```