We gaan een website maken

intro

Doel

Meerdere:

1. welke vragen (en antwoorden) zijn belangrijk **voordat** je een website gaat maken
2. kennis maken met **een** techniek om een website te realiseren
3. beter begrip andere technieken

Programma

Deze 1e bijeenkomst:

- vragen / antwoorden mbt het maken van een website
- beginnen met opzetje

Volgende bijeenkomsten:

- registratie en login/logout mogelijk maken (denk wel nodig..)
- betere setup
- database integratie
- formulieren
- templates
- code in git
- maken van de toepassing (?)

Eerst wat vragen beantwoorden

1) Wat voor soort website?
- Statische of dynamische website?
- Wat voor data / database?
- Welke taal?
- Frontend / backend?
- Community ondersteuning (martkaandeel)?, perspectief?
- CMS ondersteuning?

Wat voor website?

Statisch:

- Genoeg aan html / JavaScript? / css? / cookies?

  Let op: met bv. een formulier al vrij snel een programmeertaal nodig.

Veelal behoefte aan meer dynamiek met dynamische data (bv. uit een database)

Wat voor data? Database?

- “SQL data” -> RDMS (Oracle, MySQL, Postgresql, SQLite!)
  - Voor gestructueerde data (rij/kolom waarden, afhankelijkheden)
  - Consistentie, data-integriteit
  - Minder schaalbaar dan NoSQL

- NoSQL -> mongodb, elasticsearch, timeseries db? (influxdb, prometheus?), …
  - Voor niet-gestructureerde data (emails, social media data, logging, time-series, …)
  - Beter schaalbaar
  - Geen limiet aan “data types” (bv. alles in json)

Welke taal?

Voorkeur gecompileerd boven scripting?

go?

Scripting talen goede interpreters en caching: php-fpm, OPcache, WSGI, .pyc, …

Denk ook aan: ondersteuning voor templating: scheiden van logic en layout

-> scripting taal framework!

Frontend en/of backend? Framework?

Is je website meer een “desktop applicatie” (veel “widgets”): meer mogelijkheden met een frontend taal als JavaScript.

- Veel Server side JavaScript libraries / frameworks: JQuery, Angular, React, Vue, …
- complex..

Minder desktop maar meer “website”: keuze voor een goede backend:

- php - veel “frameworks”, grote community, veel modules (wel grotendeels “web-georienteerd”)
- python: ook frameworks (Django, Flask), opkomende communitiy, zeer uiteenlopende module ondersteuning (bv. voor data analyse (SciPy, NumPy, pandas), machine learning (TensorFlow), Graphs (matplotlib, Plotly),...)

Combinatie frontend + backend komt ook vaak voor.

Community ondersteuning / perspectief

Stack Overflow Trends (insights.stackoverflow.com)

(aan de hand van “tags” op stack overflow)

Niet zaligmakend: denk aan je eigen toepassing (en kunde)!

CMS?

- Soms heb je genoeg aan een standaard CMS (met eventueel extensies)
- Makkelijk uit te breiden?
- Blijf je flexibel genoeg?
- Meest populair: Wordpress (nog steeds meer blog dan CMS maar veel extensies)
- Maar ook Joomla, Drupal, ...
- Extensies liever niet: “broken dependencies met upgrades”

Conclusie

- Kies de tooling die bij je eindprodukt past
- Zorg dat je oplossing “flexibel” is: dat je makkelijk kan inspelen op veranderende wensen
- Denk aan community support
- Doe wat je leuk vindt
- Wees thuis in je oplossing
- Bij coderen: goede afspraken!

Hoe nu verder?

We gaan een website bouwen! :)

Waarvoor?

- blog? 6
- ticket systeem? 10
- webshop?
- md -> html
- iets anders?

Technieken:

- taal: python
- framework: flask
- database: MySQL

Begin

Maken van een (virtuele) werkomgeving:

```
$ python[3] -V
Python 3.8.5
$ pip3 install virtualenv
$ virtualenv venv
$ source venv/bin/activate
(venv) $ mkdir website; cd website
(venv) website$ vi hello.py
```

hello.py

```
    -------------------------
    from flask import Flask
    app = Flask(__name__)


    @app.route('/')
    def hello_world():
        return 'Hello, World!'
    -------------------------
(venv) $ export FLASK_APP=hello.py
(venv) $ flask run
(venv) $ export FLASK_ENV=development
```

Een eerste template

```
<html><head></head><title>Hello world</title>
<body>
<h1>Hallo wereld</h1>
</body></html>
```

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def hello_world():
    return render_template('hello.html')
```

lets flexibeler

```
<html><head></head><title>Hello world</title>
<body>
<h1>{{ greeting }}</h1>
</body></html>
```

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def hello_world():
    greeting="Hallo, daar zijn we weer ;)"
    return render_template('hello.html',  greeting=greeting)
```

Hoe verder?

Login

Nodig:

- database design (tabel bv.: users)
- database integratie (SQLAlchemy)
- formulieren (registratie, login)
- template met register/login/logout menu
- verschillende “routes” (/register, /login, /logout, …)
- flexibeler setup
- code in git