Informationsteknologi A: IT Eksamensprojekt

Afleveres: 04 / 04 2017

John Gerhard Jensen

Udvikling af en blog

Christian Påbøl

Indhold

1 Indledning

Dette er en rapport over det arbejde jeg har lavet de sidste 30 elevtimer i faget Informationsteknologi(IT A) under lærer John Gerhard.
I IT fik vi stillet et meget åbent eksamensprojekt, uden egentlige rammer. Dette efterlod mig med mange muligheder og efter noget brainstorm og nogle fravalg kom jeg frem til mit endelige projekt: en blog.
Med mindre man har boet under en sten i de sidste 10-15 år så er man bekendt med termet. Det kom til som en forkortelse for "weblog"og er en slags offentlig dagbog. Justin Hall er krediteret til at være en af de første bloggere,[1] som siden 1994 og ind til 2005. Han delte mange af sine tanker, lige fra intime fantasier til klager over kedsomhed og fyldt op med billeder, videoer og andre former for medier. I dag starter og slutter der dagligt blogs verden over, mange af dem dør ud, nogle få skaffer sig en samling følgere og de helt unikke bliver verdensomspændende med indtægter på flere millioner(Eksempelvis "Pewdiepie", en kendt videoblogger). Dette er takket være menneskets medfødte narcissisme og de mængdevis af forskellige platforme tilgængelige. Fra mikrobloggingplatformen Twitter til giganten Wordpress, den første med en karaktergrænse på 140 tegn per "tweet"og den sidste der startede som et blog framework og har udviklet sig til et flyvefærdigt Content Management System.

1.1 begrundelse

Jeg har valgt at lave en blog fordi det er en nem måde at udvise mine færdigheder i IT. Der er mange interresante problemer at tage fat i og løse og der er meget dokumentation for optimale løsninger tilgængelige. Derudover fra en mere personlig synsvinkel, har jeg valgt en blog fordi det er en god måde at starte en portefølje på. Da dette er mit tredie år på gymnasiet skal jeg snart ud og finde et arbejde i it-verdenen og en blog, hvor jeg kan skrive om projekter jeg laver eller ting jeg finder er en god og ofte brugt metode til at vise teknisk kunnen og interesse.

2 Teknologier

I dette underemne vil jeg redegøre for de forskellige teknologier jeg har brugt til udviklingen.

2.1 Python

Python er et programmeringssprog som startede i 1994. Det er et High-level sprog som er meget læseligt og nemt at lære, men samtidig uffatteligt stærkt og nemt at udvide. Et eksempel på sproget kan ses:

```
1 def fib(n):
2     a, b = 0, 1
3     while a < n:
4         print(a, end=' ')
5         a, b = b, a+b
6     print()
7 fib(1000)
8 >>>0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

[1] Se "Time to get a life[...]"artikel

Ovenstående skriver alle fibonacci tal op til 1000 på kommandolinien. Man kan se dets lidt ukkorante struktur. Istedet for de klassiske tuborgparenteser bruges der indentering til at definere kodeblokke. En funktion `fib(n)` bliver defineret med syntaxen `def fib(n):` hvor `def` bruges til deklarationen, parametrene er i parenteser og `:` definerer at en kodeblock starter
Dernest bliver variablerne a, b deklareret som $a = 0, b = 1$ da man i Python ikke behøver initialisere variabler før man bruge dem. Som forlængelse af dette har Python et løst variabelomfang. Alle variabler er lokale med mindre de bliver initialiseret med syntaxen `global var`. Et klassisk `while` loop bliver kaldt med betingelsen $a < n$. Bemærk her at man ikke behøver parenteser og at næste kodeblock bliver startet med `:` og næste linie som følge er indenteret. Udregningen bliver skrevet til konsollen med kommandoen `print(a, end='')` hvor a er variablen der skrives og `end=' '` overskriver `print` standardværdien, som er at printe en ny linie efter hvert kald. Til sidst bliver der kaldt `print()` for at skrive en ny linie til konsollen. Så køres funktionen med `fib(1000)` og outputtet kan ses i bunden efter `>>>`

2.2 Django

Django er et web-framework skrevet til python, det er open source og bygger på en såkaldt MVT struktur. Det er en udvidelse af det klassiske Model-View-Controller system og står for Model-View-Template. Templating er nemt, og bruger deres eget template system de selv har udviklet. Det tilbyder HTML blandet med Python, på en måde der minder om php.
Django er struktureret om tre filer: "Models.py", "Views.py", og "Urls.py". Models bruges til at sætte databasemodeller op, og definerer det meste af dataen der bliver brugt i de forskellige "views"[2], nemt defineret i views.py. Urls.py er et routing table, der modtager requests til hjemmesiden, og redirecter til de relevante views.

3 Produkt

Mit projekt er en blogplatform, der på forsiden viser en liste over de tre nyeste posts. Klikker du på en posts forfatter kommer du til den forfatters side, og klikker du på en post bliver du redirectet til en permanent url til den post. Posts bliver oplagt gennem djangos indbyggede back-end system, og kan planlægges frem i tiden, samt indeholde diverse formateringer.

3.1 Eksisterende løsninger

Der eksisterer som tidligere nævnt allerede mange forskellige løsninger til at skrive en blog, mest anerkendt er nok "Wordpress". Denne blog sigter ikke efter at erstatte de løsninger, men efter at fremvise mine egne evner.

3.2 Hvorfor har jeg valgt de værktøjer

Jeg har valgt Django frameworket fordi det er nemt at prototype samt nemt at skalere og integrere på store servere. Django har en indbygget udviklingsserver som kører lokalt, men i produktion integrerer den nemt med klassiske webservere som Apache© eller Nginx som er designet til at håndtere så mange requests som computerens hardware kan håndtere, fyldt med mange års optimiseringer.

[2]Et view tager noget data fra en request og serverer data dynamisk

3.3 Hvordan har jeg lavet produktet

Jeg har lavet produktet med en development server opsat lokalt, og derfor brugt en Postgres database. Dette er ikke så skalerbart, men Django gør det nemt og modulært at skifte til en klassisk database som mysql, ved at ændre i konfigurationsfilen.
Jeg har tre primære routes til min blog: en til posts, en til indlægsforfattere og en til permalinks. Derudover kan adminpanelet bagved tilgås ved en bestemt url med et sikkert login. Jeg har sepereret back-enden og frontenden i to django "Apps". Alt databasehåndtering i "back"og alt front-end og Human computer interaction i "blog". Dette er resultatet af Djangos modulære tilgang til tingene.

4 Konklusion

Produktet er færdigt, men skulle jeg gøre det igen har jeg lært følgende: Grafisk design er en hel opgave for sig selv, her ville en gruppe hjælpe. Jeg har lært at databasehåndtering er noget man skal tænke over, har man allerede sat en tabel eller model op skal man passe på når man laver ændringer. Alt i alt er Django dog et godt framework, især til denne slags arbejde og med mindre end 2000 linier kode for at sætte skelettet op er det tydeligt at Django kan trække en masse.

5 Kilder

6 Bilag

Den fulde kildekode er tilgængelig på github: http://www.github.com/mrcpj1998/BlogFrame og den kildekode jeg bruger til eksamen kan hentes ved commit `cb38859` evt ved kommandoen:
`git clone https://github.com/mrcpj1998/BlogFrame.git ; git checkout cb38859`