DKuug nr. 175 *September 2014*

NYT

Kører Ruby på skinner? - vi ser på Ruby on Rails,
et framework for web applikationer -
Simple eksempler og
vejledning i installation

Dansk Forum for Åbne Systemer
DKUUG - Unix Brugere (Linux/BSD/AIX mfl) og system administratorer
Community for IT-specialister og IT-interesserede.

DKUUG

Et brev fra redaktionen

Husker du hvor sjovt det er at programmere?

For mange år siden sad jeg i en fysiktime og hørte vores inspirerende lærer fortælle, hvordan tyngdekraftens påvirkning af legemet er en af de ting, som mennesket kan undre sig over og opleve på mange forskellige måder, og at det morer de fleste mennesker at blive rystet og rutsche op-og-ned.

"Det er det, som man udnytter i Tivoli, hvor folk kører i rutschebane, eller roterer i et pariserhjul, støder ind i hinanden i radiobiler og så videre," fortalte han og fortsatte: *"Hvert år* ***tager jeg en tur i Tivoli*** *for at huske, hvordan det er - jeg synes at når man underviser, skal man også kende de følelser og oplevelser, som mennesket har af de fysiske fænomener."*

Oversat til IT-formidling betyder det at en IT-redaktion må slippe alvoren og more sig med et nyt sprog, en ny app-builder eller vanvittige programmer. Det bør man gøre et par gange om året for at holde sig i form ☺

Man må simpelthen lægge sine erfaringer bag sig og tage fat på et emne, som er nyt og aldeles uvant.

Denne gang tog vi fat på Ruby on Rails, et system til app-building.

Rails skulle egentlig køre på skinner, som navnet siger - og det gør det måske også for de, der begynder på en ren maskine uden tidligere installationer, men desværre oplevede vi her på redaktionen et par afsporinger. Det er nu ikke noget, som kan ryste erfarne IT-mennesker med mere end 30 år i branchen og skyldtes simpelthen at systemet kræver bash eller zsh - og vi bruger (somme tider) ksh eller mksh.

Som systemadministrator gennem 30 år ***har*** man oplevet alle fejl og crash - manglende libraries, udefinerede variable og det, der er værre, og man er helt glad og taknemmelig, bare maskinerne kører og fysisk memory fungerer uden glitches.

Ikke desto mindre betød problemerne med eksempler til ***Ruby on Rails*** at artiklen måtte have en ekstra kommentar og det blev til et par sider om alternativ installation af en ældre version ***Ruby on Rails***.

Det tog lang tid (og forsinkede bladet) - men det er lærerigt og morsomt. En installation med Debians apt-get er driftsafdelingens ønskedrøm, det er stabilt! og følger generelle system-regler. Så pyt med at det er en lidt ældre version!

Design af en app-builder

Der dukkede kritiske spørgsmål op undervejs: er embedded Ruby bedre end embedded PHP? Hvilket system tillader den bedste adskillelse af datamodel, flow-kontrol og præsentation (eller user-interface)?

For at besvare det, må man se nærmere på ***Rails*** end vi har gjort i dette nummer af DNyt. Som det vil fremgå, hvis man kigger efter under installation af Rails, er der mange bestanddele som hver især udgør en del af struktureret net-programmering: **Model, View, Control** (MVC) hvor model er en data-model, som repræsenterer det, vi ønsker at registrere og/eller beregne på (klimamodel, budgetmodel eller blog-model, whatever).

Rails er opdelt i forskellige pakker, ***ActiveRecord***, som er et database-mapping system, ***ActiveResource*** som er et web-service system, ActionMailer mfl. som er klasser i et system af program-objekter. ActionController::Base er den kerne, som håndterer web-requests. En *action* er en public method, som styret af Rails Routes giver request videre til webserveren.

I de simple eksempler i dette blad bruges kun de "øverste" dele af abstraktionslagene - hvis man vil have fat i cookies, flash og lignende, skal man have fat i ***ActionController::Base.*** - Men det kan gøres på en systematisk, struktureret måde.

I eksemplerne er det også ***Ruby***, som lytter på port 3000, men ***Rails*** kan også arbejde med Apache og andre webservere, som giver ekstra fleksibilitet og mulighed for finkornet access-kontrol.

Rails udspringer af projekt-management systemerne *Basecamp* og *37signals* (nu navn på web-app virksomhed). David Heinemeier-Hansson releasede en første version 2004, og i 2005 var det blevet Open Source, og han gav andre lov til at committe direkte. I August 2006 nåede *Rails* en milepæl, da Apple besluttede at shippe Rails med Mac OS X v10.5 Leopard. Det blev released i 2007.

Twitter brugte Ruby on Rails, men kritikere af systemet hæftede sig ved, at Twitter i 2007-8 havde nogle driftsstop, og det bevirkede at Twitter gradvist gik over til Scala, som kører på en Java Virtual Maskine, for queueing system og andet. View - oversættelse til HTML, kørte videre til 2011, hvor det blev udskiftet af hensyn til performance.

Nogle af de største websites, som kører *Rails* er GitHub, Yammer, Scribd, Shopify, Hulu og Basecamp. I maj 2014 skønnedes det at 600.000 webservere kørte *Rails.*

(fortsættes side 18)

Dette nummer ...

Der er sket en lille ændring i dette nummer - vi har gjort alvor af at bruge en font med seriffer (fødder, afrundede ender på stregerne)

Det er en font, som ligner den oprindelig Times New Roman der så lyset i den britiske avis *The Times* ønskede i 1932. New Roman tillod en lille skriftstørrelse, som stadig var let at læse. Det er længe siden, og Times er gået over til tabloid format og andre fonte, som dog stadig baseret på Times New Roman.

Nimbus Roman No.9L er også en variation af New Roman. Udover at håbe på lettere læselighed håber vi også på at lægge lidt afstand til fonten Helvetica, sans serif. Det var en Nimbus Sans Serif, som vi brugte. Helvetica virkede forfriskende moderne, da den kom frem omkring 1950. Nutildags virker den mere som noget der hører til ugepressen.

Vi prøvede med Roman No.9L i en enkelt artikel i sidste nummer, og resultatet var OK. Trykkeriet kan godt håndtere den font ned i detaljerne. (serifferne er jo næsten mikroskopiske.)

I dette nummer bringer vi en billedserie fra Labitat på H.C. Ørstedsvej, et fristed for hardware interesserede, et sted hvor man kan snakke om bits og printkort uden at nogen blinker eller misforstår, hvad man siger.

Man er velkommen til en guided rundtur i kælderen på

H.C. Ørsteds Vej 5, 1879 Frederiksberg C, Denmark

Der er åbent for gæster hver tirsdag aften - der er nogen hele dagen, men om tirsdag aften er døren åben for alle. Udenfor dette tidsrum skal man enten kende en adganskode eller kende et medlem, som kan give en rundvisning.

Desuden skriver Lars Sommer om sikring af en CMS installation - Wordpress, det mest udbredte CMS, som er meget nemt at installere og som nemt kan tilpasses til mange, mange behov.

Donald Axel

DKuug-NYT er medlemsblad for DKuug, foreningen for Åbne Systemer og Internet
Nr. 175 - September 2014

Udgiver:
DKUUG
Fruebjergvej 3
2100 København Ø
Tlf. 39 17 99 44
email: blad@dkuug.dk
Redaktion:
Donald Axel (ansvarshavende)

Forsidecredits:
(redaktionen)

Design og layout:
DKUUG/Donald Axel med LibreOffice
Annoncer:
pr@dkuug.dk
Tryk:
Lasertryk i Aarhus
Oplag:
400 eksemplarer

Artikler og inlæg i DKUUG-Nyt er ikke nødvendigvis i overensstemmelse med redaktionens eller DKUUGs bestyrelses synspunkter.

Eftertryk i uddrag med kildeangivelse er tilladt.

Deadline for nr. 176: ons. 18. oktober 2014

Medlem af Dansk Fagpresse
DKUUG-Nyt
ISSN-1395-1440

Vores møder og **foredrag** holdes - med mindre andet udtrykkeligt angives - på vores adresse:

DKUUG
SYMBION
Fruebjergvej 3
2100 København Ø

Hvis man kommer lidt før, er der tid til en snak på kontoret. DKUUG bor i en virksomhedsfarm, Symbion, hvor der er åbne døre indtil kl.18 eller 19 (afhængig af mødetidspunkt). Efter den tid har vi på foredragsaftener en vagt ved døren.

INDHOLD:

Arrangementer:

Onsdag d. 17. september kl.18: Serie & Javascript, CSS. Kombination. Christian Tusborg.

Onsdag d. 1. oktober kl. 18: Serie & Javascript, CSS. Kombination. Christian Tusborg.

Onsdag d. 22. oktober: Serie & Javascript, CSS. Kombination. Christian Tusborg.

Lørdag d. 8. November: OSD community day

Onsdag d. 26. November: Generalforsamling.

Andre arrangementer vil blive annonceret på web og via mail.

Gør-det-selv foredrag:

Vores kontor i Symbion giver mulighed for hands-on workshops, både dag og aften. Skriv til pr@dkuug.dk eller til bestyrelsen i DKUUG, bestyr@dkuug.dk, og hør om lokalet er ledigt den dag du vil arrangere et møde. Der er hurtig internetforbindelse, både wired og wireless. Er der større tilmelding, kan vi leje mødelokaler i Symbions mødested. Spørg os!

Ruby on Rails

Af David Askirk Fotel

Ruby on Rails, Rails eller RoR, kært barn har mange navne

I denne artikel vil vi komme ind på hvad RoR er, og lave en webapp med rails.

Rails er et MVC framework.

MVC står for Model View Control. I denne model er der en skarp adskilling mellem de tre dele.

Lad os undersøge MVC princippet først:

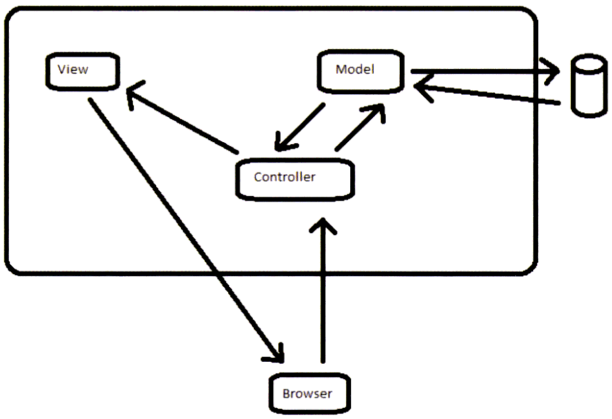
M (Model) er den datamodel der bliver brugt i webapp. I rails bliver der også brugt et database abstraktionslag, så de modeller der arbejdes med fungerer ens uanset hvilken database der bliver benyttet nedenunder. Normalt praksis er at lægge udtræk her, fx. hent alle brugere givet et postnummer ville man lave som en funktion i modellen.

V (View) er det view som bliver brugt. Det er som regel implementeret i flere forskellige view filer. Fælles for alle views er at de ikke indeholder noget logic, højst en if sætning, eller et for loop til at iterere over en samling af data.

C (Control) er selve forretningslogikken i webappen. Det er denne der håndterer request, og det er her logikken til at styre hvilke data der skal gemmes og hentes ud ligger.

Hvis man kigger på det udefra, har det denne opstilling:

Et request kommer ind til en controller, som snakker med en model og laver et view klar, som bliver sendt til brugeren.



Det smarte ved denne model er netop den skarpe opdeling mellem de forskellige elementer, som gør det nemt og overskueligt at udvikle sin rails webapp.

For at komme igang med rails skal rails først installeres.

Rails kan installeres på flere måder:

Via ***rvm*** (http://rvm.io/), via pakke manageren i styresystemet, eller via ***railsinstaller*** (http://railsinstaller.org/en). *(Se dog side 18.)*

Jeg anbefaler klart ***rvm***. Den sørger for, eller gør det lettere at sørge for, at man altid har den nyeste udgave af både ruby og rails, så man ikke behøver at vente på at det kommer i den pakke manager, som følger med OS-distributionen.

Her er en meget kort guide til at installere rvm og rails:

Først, installér rvm:

```
$ curl -sSL https://get.rvm.io | bash -s stable
```

Dernæst installér ruby:

```
rvm install 2.1.1
```

(eller hvad den nyeste nu er når du. kære læser, læser dette)

Så er ruby installeret så skal bundler og rails installeres (her uden dokumentation):

```
gem install bundler rails --no-rdoc --no-ri
```

Så er rails installeret og vi er klar til at lave vores første webapp.

Den første webapp

Vi vil lave en lille standard app, som er en database over vores yndlingsspil, med en kommentar om spillet og med en URL, hvis det er relevant.

Når rails er installeret kan man starte et nyt projekt således:

```
rails new gamedatabase
```

Vi kalder det gamedatabase, så vi ved hvad idéen er.

Nu kører rails en masse initaliserings kode og gør ens projekt klar til start.

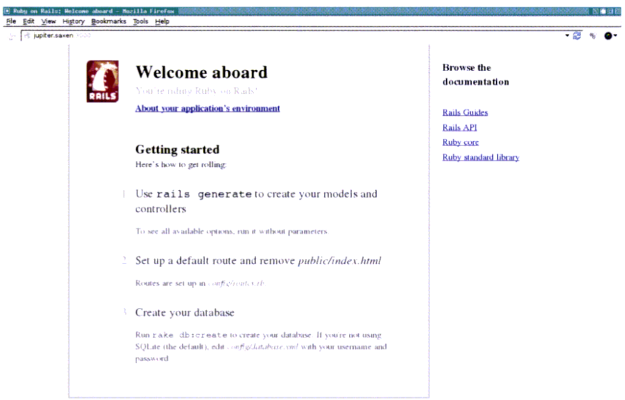
Derefter skiftets der til biblioteket:

```
cd gamedatabase
```

og vi kan starte vores rails app op med :

```
rails server
```

Nu er vores rails server startet og vores webapp kan besøges på **http://localhost:3000/** i en browser.



Lad os nu programmere noget mere

Hvis man gerne vil lave noget, så man kan se noget mere i Rails app'en - og samtidig finde ud af, hvordan det fungerer - skal man oprette et view som absolut minimum. Men vi vil gerne have mere end en statisk tekst-side, så vi må oprette en controller og en model samt view der passer til.

Lad os oprette en controller der passer til at beskrive et spil. Vi laver en spil-registrering, som består af en titel, en beskrivelse samt en url til et billede.

Spil controller oprettes således:

```
rails generate controller games
```

g er kort for generate, så man kan nøjes med at skrive:

```
rails g controller games
```

Nu har vi en controller vi vil også gerne have en model, som er vores datamodel.

Datamodellen repræsenterer overfor rails hvordan vores data ser ud i databasen.

```
rails g model Game title:string \↵
description:text url:string
```

Her genererer vi en model som indeholder tre elementer. En titel, en beskrivelse og en url til et billede. Ved at angive typerne på de tre elementer vil rails generere en database record med de typer. Typecheck forekommer først ved dannelse af databasen, så hvis du skriver en forkert type (fx. tekst med ks) så vil *generate model* ikke brokke sig.

Så har vi generet den ruby kode der skal til at oprette modellen og databasen. Det kan ses i db/migrations mappen. Brug fx.

```
ls -lt db/migrate
```

til at liste indholdet i mappen. Indholdet af filen

```
db/migrate/[tidsstempel]create_games.rb
```

er et skema for databasen:

```
class CreateGames < ActiveRecord::Migration
  def change
    create_table :games do |t|
      t.string :title
      t.text :description
      t.string :url

      t.timestamps
    end
  end
end
~
~
~
```

Man kan genkende vores tre felter, title, description og url.

Ved at køre den næste kommando dannes en tabel kaldet ***games*** i databasen, som kommer til at ligge i ***db/development.sqlite3*** (indtil man evt. bruger en anden database-backend).

Så sker det, nu skal den følgende kommando køres:

```
rake db:migrate
```

Når den er færdig, ligger modellen klar i databasen og vi er klar til at bruge data i vores applikation.

Rake står for R(uby) (M)ake! men betyder oprindeligt en rive.

Migration betyder jo egentlig at flytte data, typisk hvis man har data fra en tidligere applikation. Migrations er en struktureret måde at ændre og vedligeholde sin database. Migrations er - i rail-sammenhæng - klasser, én enkelt migration er en subklasse af `ActiveRecord::Migration` som kan udføre en række operationer på databasen, som fx. create table, add column, drop table (og meget mere).

Nu skal vi gøre en ting mere, nemlig oprette en route entry til vores nyoprettede controller.

I rails bliver alting routed efter forskriften

/<controller>/<action>

For at få en masse default routes baseret på vores controller kan vi kigge i config/routes.rb filen. Den indeholder definitionen for de forskellige routes.

I denne fil tilføjes linjen:

```
resources :games
```

i selve klassen. Så kan de nye routes ses med kommandoen:

```
rake routes
```

Output fra rake routes kan fx. se sådan ud:

```
Jup2:/hjem/src/ruby/gamesdatabase #rake routes
    games GET    /games(.:format)          games#index
          POST   /games(.:format)          games#create
 new_game GET    /games/new(.:format)      games#new
edit_game GET    /games/:id/edit(.:format) games#edit
     game GET    /games/:id(.:format)      games#show
          PUT    /games/:id(.:format)      games#update
          DELETE /games/:id(.:format)      games#destroy
Jup2:/hjem/src/ruby/gamesdatabase #
```

Vi kan nu prøve en af ruterne, fx, den der skal bruges hvis vi gerne vil oprette et nyt spil i vores database:

```
http://localhost:3000/games/new
```

Men uha - vi er ikke færdige!

Det vil fejle, da godt nok er vores game controller oprettet, men vi har endnu ikke defineret den action der hedder new.

Dette gøres i vores controller som ligger i:

```
app/controllers/games_controller.rb
```

I denne fil er der lige nu en bar klasse definition, som kun inderholder definition på vores controller - hvor vi kan se at den nedarver fra ApplicationController, som også ligger i vores projekt.

I denne klasse oprettes nu en ny metode kaldet new således:

```
def new
end
```

Så ser hele vores klasse således ud:

```
class GamesController < ApplicationController
        def new
        end
end
```

Dog brokker sig rails sig stadig over at tingene ikke er i orden.

Vi mangler stadig at oprette et view som er selve den form vi skal bruge til at oprette et nyt spil i vores database.

I mappe strukturen i vores rails app er der en mappe der hedder app. Den inderholder alt logikken som vi laver til vores app.

I controllers mappen ligger vores controllers. I models mappen ligger vores modeller og i views mappen ligger alle vores view. Denne mappe er så igen inddelt i undermapper inddelt efter controller navn. Controller navnet matcher model navnet.

Lad os oprette et view kaldet new.html.erb i app/views/games. Lad os starte med filnavnet. Det slutter på erb som står for embedded ruby, altså muligheden for at ligge ruby kode i et view.

Vores view skal blot indeholde dette:

```
<h1>Nyt spil</h1>
<%= form_for :game, url: games_path do |f| %>
  <p>
    <%= f.label :title %><br>
    <%= f.text_field :title %>
  </p>

  <p>
    <%= f.label :description %><br>
    <%= f.text_area :description %>
  </p>

   <p>
    <%= f.label :image_url %><br>
    <%= f.text_field :image_url%>
  </p>

  <p>
    <%= f.submit %>
  </p>
<% end %>
```

Her kommer en lille detajle: Vi bruger ***game*** og ikke games, som vi ellers har gjort! Det er fordi vores ***form*** (webskema-input) binder sig til vores model. Den anden del url: er for at vi peger på /games når vi laver en post fra formen istedet for at pege på games/new.

Lad os prøve at køre det samme igen i vores browser. Nu kan vi udfylde informationen i vores form og trykke på submit. Dog fejler rails igen, denne gang på at den ikke kender den action der hedder create. Lad os oprette den.

Start med at tilføj en metode i vores controller kaldet create:

def create

end

Nu har vi de to metoder. Dog sker der stadig ikke noget når vi poster.

Ved at ændre vores new metode og tilføje en ny metode til vores controller kan vi gemme data.

```
def create
    @game = Game.new(game_params)
    @game.save
    redirect_to @game
end

private
  def game_params
    params.require(:game).permit(:title,
    :description, :url)
  end
```

Den metode der hedder game_params angiver hvilke parametere vi gerne vil modtage fra browseren. Dette er for undgå en sikkerhedsfejl der ellers har været i rails.

Lad os undersøge koden i create metoden:

```
@game = Game.new(game_params)
@game.save
redirect_to @game
```

I linje 1 opretter vi et nyt object kaldet game. @'et foran game angiver at det er instans variabel. Værdierne som vi bruger til at oprette objektet kommer gennem vores parameter, som vi har indtastet i vores form.

Linje 2 gemmer objektet ned i databasen.

I linje 3 redirekter vi, dvs. sender browseren til en side som viser det objekt. I rails kalder den automatisk show aktionen, som vi hellere må lave med det samme.

```
def show
       @game = Game.find(params[:id])
end
```

I denne metode henter vi det spil ud fra databasen med det id vi har fået givet i browseren.

Til vores show aktion skal der laves et view der passer til.

Opret nu ***app/views/game/show.html.erb*** med følgende indhold:

```
<p>
  <strong>Title:</strong>
  <%= @game.title %>
</p>

<p>
  <strong>Description:</strong>
  <%= @game.description %>
</p>

<p>
  <strong>image:</strong>
  <img src="<%= @game.url %>">
</p>
```

En liste over indhold i databasen

Nu kan vi oprette en spil beskrivelse, og se den listet, men vi vil gerne se en liste over alle vores spil. Dette er også standard aktion, det som man får hvis man kalder /game/ på vores server.

Først skal der laves en action:

```
def index
        @games = Game.all
end
```

Det var det. Game.all henter datarecords fra databasen. Så skal vi lave et view der hører til:

Opret en fil, der hedder
`app/views/game/index.html.erb`

og skriv i den:

```
<h1>Listing games</h1>

<table>
  <tr>
    <th>Title</th>
<th>Tools</th>
  </tr>

  <% @games.each do |game| %>
    <tr>
      <td><%= game.title %></td>
      <td><%= link_to 'More info', game %></td>
    </tr>
  <% end %>
</table>
<%= link_to 'New game', new_game_path %>
```

I dette view sker der en del ting.

Der bliver lavet en table der viser vores indhold, herunder listen af spil. Ved hvert spil bliver der lavet et link til det pågældende spil. Her bliver brugt en rails funktion (nogle ville sige magi) nemlig link_to, hvor vi bare giver en tekst med der skal vises "More info" og det object den skal linke til. Så ved rails selv hvad den skal skrive som url.

Til sidst bliver der lavet endnu en ting, en ***link_to*** hvor der bliver linket til vores form til at oprette et nyt spil.

Som den sidste krølle på halen vil vi gerne have at vores applikation starter med at vise oversigten over vores spil. Dette gøres i vores routeconfig.

I ***config/routes.rb*** skal vi tilføje:

```
root 'games#index'
```

Det fortæller rails at når browseren går ind og gerne vil have / (root) så skal man have games/index.

Nu er vores spil-database komplet, ihvertfald indtil videre ✌ og vi kan gå til det sidste emne, nemlig omkring hosting af vores rails app.

Til hosting af rails findes der forskellige muligheder. Hvis man gerne vil hoste gratis kan Heroku anbefales. Her er det muligt at begynde gratis, og det er muligt at betale for udvidelse, hvis man har brug for flere kræfter at rulle op.

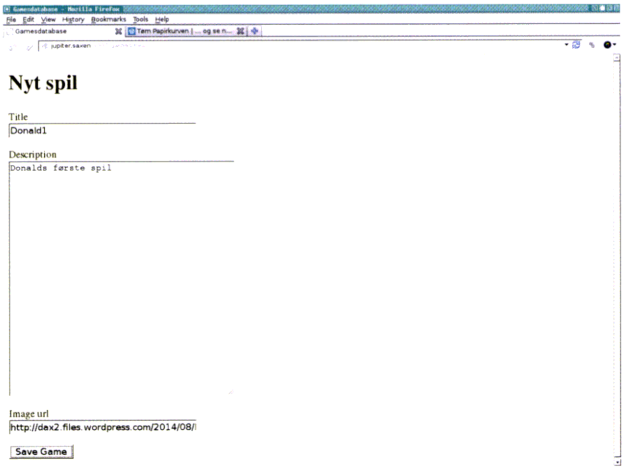
Hvis man vil hoste selv kan man benytte sig af ***nginx*** eller ***apache*** op som reverse proxy.

Hvis man gerne videre med rails kan med fordel gå ind rails guides og gå videre derfra.

`http://guides.rubyonrails.org/`

Hvis man er til lidt underholdning kan man se følgende video og de efterfølgende:

http://www.youtube.com/watch?v=PQbuyKUaKFo



Installation af Rails m. Debian pakke management

Af Donald Axel

Driftsafdelingens Rails - med Debians apt-get

Hvis man ikke lige ved det, lyder ***apt*** for en englænder som noget, der er passende effektivt, godt. Hun var hurtig til at lære - *she proved an apt pupil.*

Eller et værktøj, som er velegnet: *It was an apt tool.*

Debian pakke-system har mange specialiserede pakke-management programmer, men under dem alle ligger ***dpkg,*** Debian Pakage system; oven på ligger fx. ***aptitude, debsums,*** m.fl. - ***apt-get*** er det, som oftest nævnes i artikler og vejledninger, og det reddede redaktionen fra flere afteners søgen efter fejl. **A**dvanced **P**ackage **T**ool - **APT** - ***It has proven to be an apt tool!***

Det er imidlertid ikke pakke-management programmerne, som er afgørende for om tingene virker, men derimod pakke-arkivets måde at håndtere afhængigheder, dependencies:

Debian distributionens stabilitet er anerkendt gennem mere end 10 år, og andre distro'er bygges ovenpå Debian, netop fordi det er så stabilt. Der findes som bekendt mange andre pakke-systemer, og i et kommende nummer vil der være et tema om distributioner og dependencies.

Vi prøver som regel alle eksempler af

Men jeg lagde apt-get på hylden for en stund. Et system til at holde styr på versionerne, oven i købet flere versioner, sådan som **R**uby **V**ersion **M**anagement, (***rvm)*** kan gøre det, lød lovende og skulle afprøves.

Jeg studsede først over installationsmetoden.

Behøver jeg at nævne, at det kan være usikkert at køre et script, som man henter fra en webserver, uden at se det igennem? Uden at checke URL'ens ægthed?

Installationen af ***rvm*** begynder med at man kører et script, som man henter fra en URL - uden at læse scriptet igennem. Alene det fik tæerne til at motionere lidt:

```
curl -sSL https://get.rvm.io | bash -s stable
```

Men det er et kendt website, og med wget kan man hente scriptet og kigge det igennem. 20 kb 800 linier script er nu ikke lige det hurtigste at kigge igennem, og det var et test-system, som kunne ofres. Scriptet med option stable henter *stable* versionen og installerer fra source, noget der var ganske almindeligt for blot 10 år siden. ***rvm*** scriptet udpakker source og kompilerer i ***/usr/local/rvm***

Jeg gik derfor igang - og den følgende installation gik uden problemer:

```
# curl -sSL https://get.rvm.io | bash -s stable
# rvm install 2.1.2
# gem install bundler rails --no-rdoc --no-ri
# source /usr/local/rvm/scripts/rvm
```

Derefter vejledes man i at der skal oprettes en gruppe "rvm" i /etc/group, enten med kommando linie ***addgroup*** eller ved editering, ***grpconv*** osv. og administrator og ***Rails***-programmøren indmeldes i denne gruppe - derved har alle adgang til de filer, som udgør ***Ruby***, og ***Gem,*** betegnelsen for ruby-moduler, (gem på engelsk betyder smykkesten).

Eftersom der stadig lå både en Debian Ruby fra forrige test og en række *gem*-moduler i /usr/lib, er det egentlig ikke så underligt at noget kunne gå galt, men foreløbig var alt vel. Men nu afinstallere jeg den gamle Debian-Ruby:

```
# apt-get purge ruby1.9.1
```

og alt dertil hørende bliver fjernet, incl. config filer.

Ruby blev afsporet!

Desværre opstod der en fejl. PATH til installationstræet blev ikke sat af opsætnings-scriptet ***/etc/profiles.d/rvm.sh***

Til syvende og sidst kunne jeg ikke lade være med at debugge opsætningsscriptet, og det viste at ***rvm*** opsætning forventer bash eller zsh - tester på dem og laver ingenting hvis ikke det er en af dem. På testmaskinen kører jeg mksh.

Man kan se fx. hvilken ***gem*** kommando, der bruges med følgende kommando:

```
# type gem
gem is /usr/bin/gem
#
```

Men hvis man har det rigtige miljø til ***rvm,*** så skal ***gem*** være en funktion! og det var den ikke, den ***gem.***

Efter login med bash (eller bare ***bash -l***) får man ***rvm***-miljøet:

```
Jup2 gamesdatabase # echo $PATH
/usr/local/rvm/gems/ruby-2.1.2/bin:/usr/local/rvm/gems/ruby-2.1.2@global/bin:/usr/local/rvm/rub
/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/rvm/bin
Jup2 gamesdatabase # type gem
gem is a function
gem ()
{
    typeset result;
    ( typeset rvmrc;
    rvm_rvmrc_files=("/etc/rvmrc" "$HOME/.rvmrc");
    if [[ -n "${rvm_prefix:-}" ]] && ! [[ "$HOME/.rvmrc" -ef "${rvm_prefix}/.rvmrc" ]]; then
        rvm_rvmrc_files+=("${rvm_prefix}/.rvmrc");
    fi;
    for rvmrc in "${rvm_rvmrc_files[@]}";
    do
        [[ -s "${rvmrc}" ]] && source "${rvmrc}" || true;
    done;
    unset rvm_rvmrc_files;
    command gem "$@" ) || result=$?;
    hash -r;
    return ${result:-0}
}
Jup2 gamesdatabase #
```

Egentlig er jeg heller ikke begejstret for, at ***rvm*** installerer en permanent environment for alle brugere, for hele systemet, i ***/etc/profile.d/rvm.sh***

For at slippe for at køre ***rvm.sh*** ved login for alle brugere/scripts, kan man lave en wrapper for kørsel af alt vedrørende **rvm** og det er lige så godt - eller bedre - end at udvide systemwide environment på et system, som ikke er dedikeret til at køre rails.

Op på Debian apt skinner!

apt-get install rails

Debian pakkesystemet har også både ***Ruby*** og ***Rails,*** men det er ofte ældre versioner. Det skulle også prøves

For lige at være på den sikre side, skulle rvm-installationen fjernes. Det er imidlertid rimeligt nemt, - hvis man har et backup-system på /mnt/b52 udføres fx. flg. kommandoer:

```
# mkdir /mnt/b52/slet
# mv /usr/local/rvm /mnt/b52/slet/
# mv /etc/profile.d/rvm.sh /mnt/b52/slet/
```

Alternativt: ***rvm remove*** # hvis rvm kan køre ...

Apt-get med forsigtighed

Man kan se, hvad apt-systemet vil udføre for at installere en eller flere pakker ved at bruge simulering, --simulate optionen (-s).

```
# apt-get -s install rails
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  bundler rake ruby ruby-actionmailer-3.2 ruby-actionpack-3.2 ruby-activemodel-3.2 ruby-activerecord-3.2
  ruby-activeresource-3.2 ruby-activesupport-3.2 ruby-arel ruby-blankslate ruby-builder ruby-dev ruby-erubis ruby-hike
  ruby-i18n ruby-journey ruby-mail ruby-mime-types ruby-multi-json ruby-net-http-persistent ruby-polyglot ruby-rack
  ruby-rack-cache ruby-rack-ssl ruby-rack-test ruby-rails-3.2 ruby-railties-3.2 ruby-sprockets ruby-thor ruby-tilt
  ruby-treetop ruby-tzinfo ruby-yajl ruby1.9.1 ruby1.9.1-dev rubygems-integration
Suggested packages:
  ri ruby-bcrypt treetop ruby1.9.1-examples ri1.9.1 graphviz ruby-switch rubygems
The following NEW packages will be installed:
  bundler rails rake ruby ruby-actionmailer-3.2 ruby-actionpack-3.2 ruby-activemodel-3.2 ruby-activerecord-3.2
  ruby-activeresource-3.2 ruby-activesupport-3.2 ruby-arel ruby-blankslate ruby-builder ruby-dev ruby-erubis ruby-hike
  ruby-i18n ruby-journey ruby-mail ruby-mime-types ruby-multi-json ruby-net-http-persistent ruby-polyglot ruby-rack
  ruby-rack-cache ruby-rack-ssl ruby-rack-test ruby-rails-3.2 ruby-railties-3.2 ruby-sprockets ruby-thor ruby-tilt
  ruby-treetop ruby-tzinfo ruby-yajl ruby1.9.1 ruby1.9.1-dev rubygems-integration
0 upgraded, 38 newly installed, 0 to remove and 1802 not upgraded.
Need to get 3318 kB/3615 kB of archives.
After this operation, 17.6 MB of additional disk space will be used.
```

[...] der kommer meget mere output af hvilke eksakte kommandoer, Debian pakkesystemet vil udføre.

Som man ser, kan man gennemgå nøjagtigt hvad der sker. Hvis der er pakker, som man ikke ønsker ændret, eller installeret, fordi man med sikkerhed ved at man ikke får brug for dem, så kan man låse og forbyde pakker, enten med apt-get eller ved direkte redigering i

```
/etc/apt/preferences
/etc/apt/preferences.d
```

På administrators systemer installerer man somme tider hjælpepakken wajig, der med nogle intuitive kommandoer kan liste indholdet og forklaringer til software-pakker. Således fx. hvis man bliver forbavset over at apt-get vil installere pakker, man ikke ønsker:

```
# wajig show ruby1.9.1
Package: ruby1.9.1
State: installed
Automatically installed: yes
Version: 1.9.3.194-8.1
Priority: optional
Section: ruby
Maintainer: akira yamada <akira@debian.org>
Architecture: i386
Uncompressed Size: 258 k
Depends: libruby1.9.1 (= 1.9.3.194-8.1), libc6 (>= 2.3.6-6~)
Suggests: ruby1.9.1-examples, ri1.9.1, graphviz, ruby1.9.1-dev, ruby-switch
Conflicts: irb1.9.1 (< 1.9.1.378-2~), rdoc1.9.1 (< 1.9.1.378-2~), ri (<= 4.5), ri1.9.1 (< 1.9.2.180-3~), ruby (<= 4.5),
        rubygems1.9.1
Breaks: apt-listbugs (< 0.1.6)
Replaces: irb1.9.1, rdoc1.9.1, rubygems1.9.1
Provides: irb1.9.1, rdoc1.9.1, ruby-interpreter, rubygems1.9.1
Description: Interpreter of object-oriented scripting language Ruby
 Ruby is the interpreted scripting language for quick and easy object-oriented programming.
It has many features to process text files and to do system management tasks (as in perl).
It is simple, straight-forward, and extensible.

 In the name of this package, `1.9.1' indicates the Ruby library compatibility version. This package
currently provides the
 `1.9.3' branch of Ruby, which is compatible with the `1.9.1' branch.
Homepage: http://www.ruby-lang.org/
```

- der som tidligere nævnt viser, at det er library-versionsnummeret, som identificerer pakken og at selve ruby-versionsnummeret er 1.9.3

Jadak - vi vælger at sige ja til apt-get install!

Kommando sekvens

Her følger kommandoer til en rails-installation med apt-get. Første kommando vil også installere *Ruby* i en version, der passer til *Rails:*

```
# apt-get install rails
[...]
# gem install bundler rails --no-rdoc --no-ri
[... installerer bundler, rake og json]
# bundle install --local
[...]
# rails new minegem
[ create
  create  README.rdoc
  ...

#
```

```
# rails --version
Rails 3.2.13
# apt-get install nodejs nodejd-dev
[...]
#
```

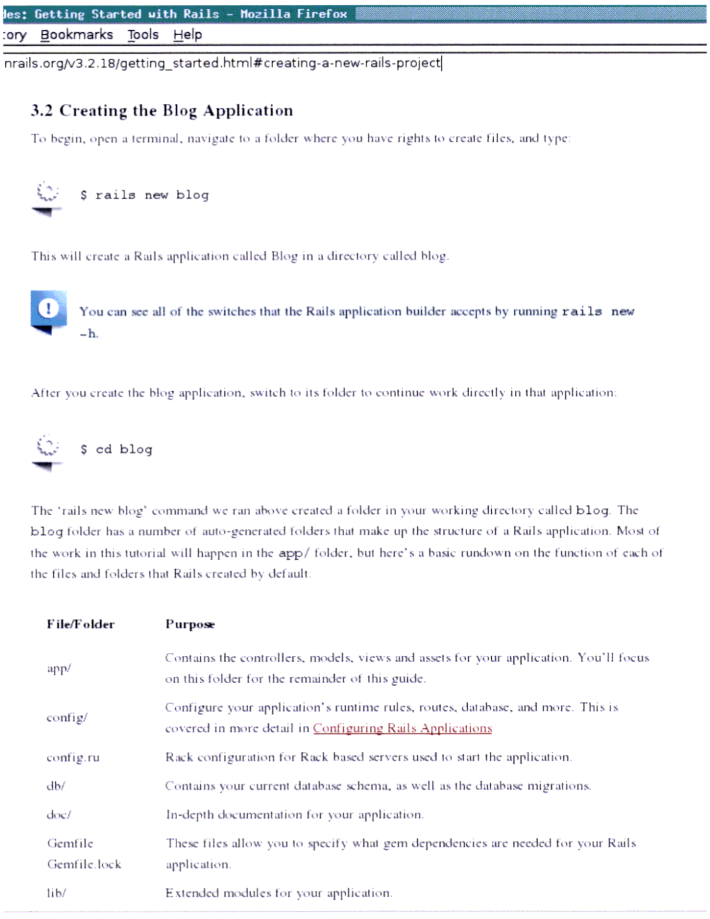
Hvis man støder på mangler eller fejl undervejs efter denne serie installations-kommandoer, må man læse fejlmeddelelserne og rette sig efter dem. Det kunne fx. være at der mangler en sqlite3, databasen, en simpel og hurtig database, som bruges af mange open source applikationer, blandt andet Firefox.

Forudsætninger for ***Rails*** er altså: Ruby, ***RubyGems*** pakke-system, ***SQLite3*** og en Javascript fortolker, som jeg fik med ***apt-get nodejs nodejs-dev***

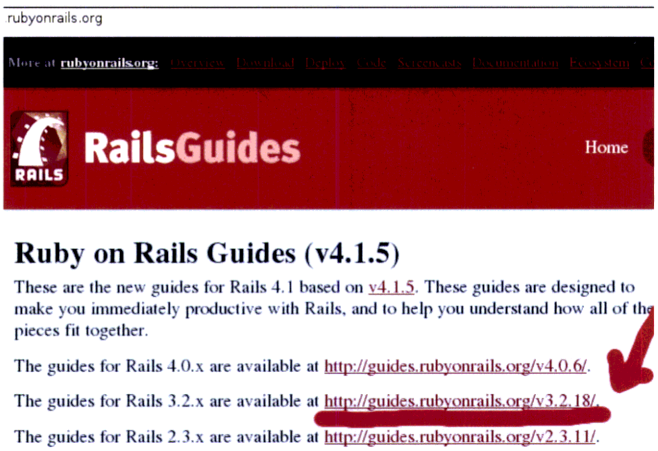
Til slut skal det understreges, at den installation, som jeg brugte til at lave en rails-demonstration, var en rails-3.2.13, men det var nu ikke noget, jeg mærkede så meget til.

En primitiv blog

Generering af en simpel blog-applikation beskrives på ruby-on-rails websitet.



På *rubyonrails.org* findes en version af *Getting Started"*for hver version af ***Rails.***



I rvm-eksemplet brugte David ***rails generate model*** - men med denne installation brugte jeg ***generate scaffold.*** Det er nogle andre MVC-filer, man får med ***rails g scaffold*** kommandoen, men det fungerede hele vejen.

Scaffold løsning er nem, og den passer fint i her på redaktionen, hvor vi hylder princippet om at springe over, hvor gærdet er lavest i stedet for at risikere brækkede ben og forvredne ankler. Så her følger, *ta-daaa!* ♬ ♫

Quick and dirty oprettelse af simpel en-dimensional register applikation:

1. rails new blog
2. cd blog
3. rails server (+ browser test, default webside)

   Luk serveren ned igen med ^C ***(control C)***
4. rake db:create
5. rails generate controller home index

   Her genereres en ny forside i en fil app/views/home/index.erb

   Denne fil indeholder blot en meddelelse om at den ligger i app/views/home og hedder index.html.erb

   .erb står for embedded ruby.

6. gvim app/views/home/index.html.erb

```
<h1>Home#index</h1>
<p>Find me in
app/views/home/index.html.erb</p>
<p>Donax added this comment Fri Aug
22 16:29:15 CEST 2014</p>
```

Når man ikke kender et auto-genererings-system særlig godt, kan det være vanskeligt at gennemskue, hvilke filer, der bliver kørt og vist i applikationen.

Derfor kan det være en god idé at tilføje et par linier i filen, så man kan se at det rent faktisk er vores egen fil, vi har fat i.

Det er en art printf-debugging, fejlfinding ved status-output. Tillføjelse af output på denne måde er en af de mest effektive metoder til at få overblik over, hvad der sker under programafvikling.

6. gvim config/routes.rb

Her indsættes route til ***home:***

```
  #   end

  # You can have the root of your site routed with "root"
  # just remember to delete public/index.html.
  # root :to => 'welcome#index'
  root :to => "home#index"

  # See how all your routes lay out with "rake routes"

  # This is a legacy wild controller route that's not recomme
  # Note: This route will make all actions in every controlle
  # match ':controller(/:action(/:id))(.:format)'
end
```

Filen er meget lang og det er nede på linie 57 at man indsætter en route til hom#index. Desuden skal man fjerne eller "skjule" det gamle, public index:

7. mv public/index.html public/old-index.html

8. rake routes

Denne kommando viser de "routes" - egentlig URL'er til browseren - og man kan evt. fange nogle fejl.

9. rails generate scaffold Post \
name:string title:string content:text

[der kommer meget output, - create ...]

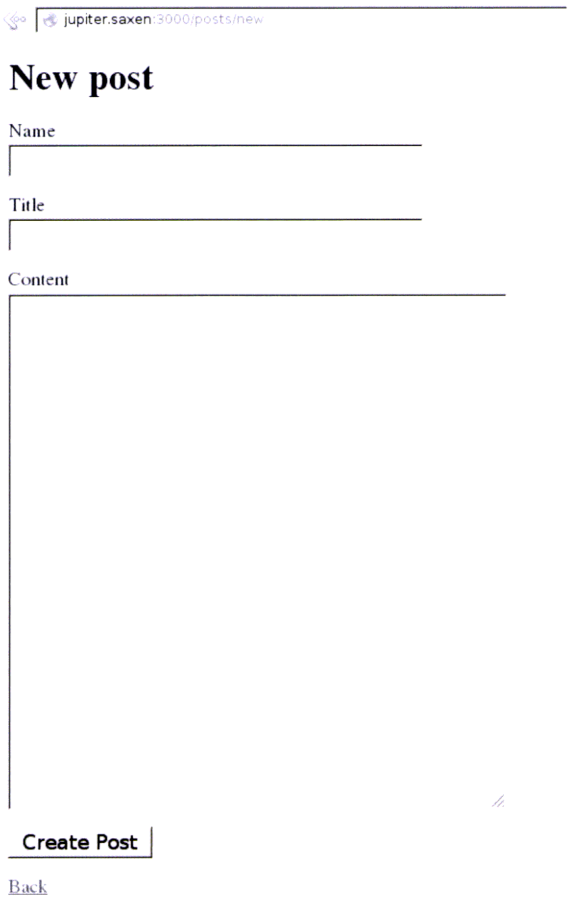
10. rake routes
11. rake db:migrate
12. rails server

Så endelig kommer en færdig applikation igang.

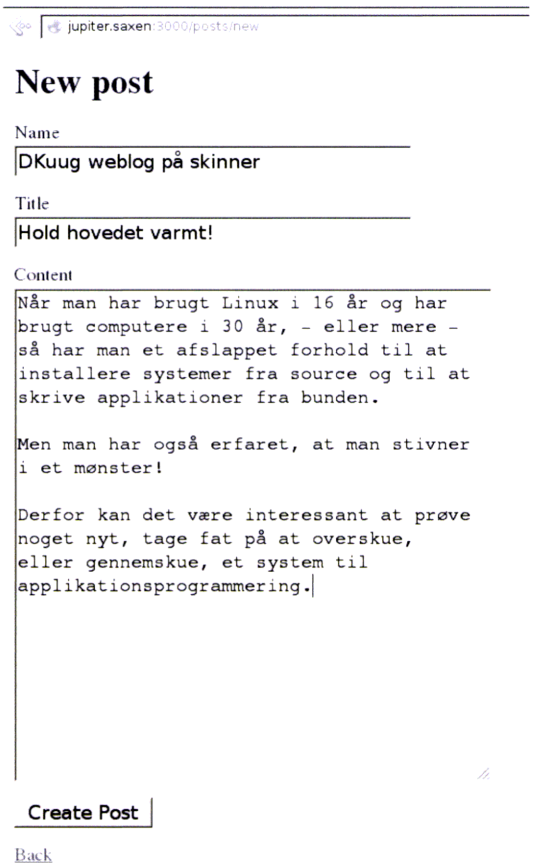
Linket nederst på index-websiden er fremkommet ved tilføjelse af følgende i filen ***app/views/home/index.html.erb***

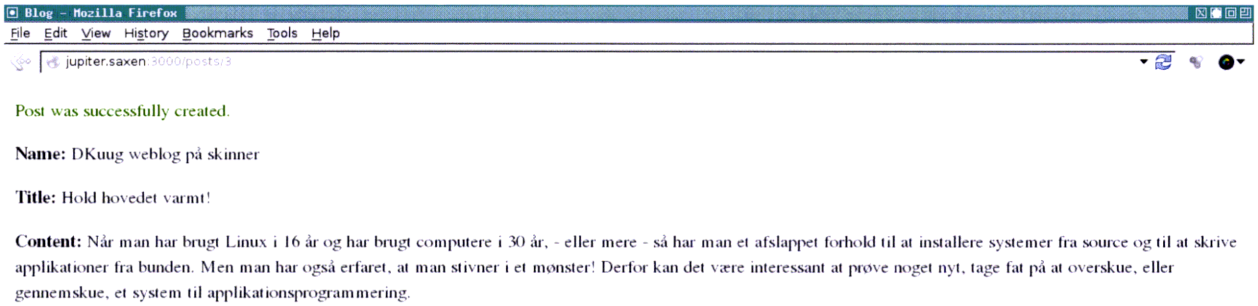
```
<h1>LINK to application post form:</h1>
<%= link_to "My Blog", posts_path %>
```

Når vi aktiverer link *("klikker på")* får vi en tom liste over blog-indlæg, blog-postings:

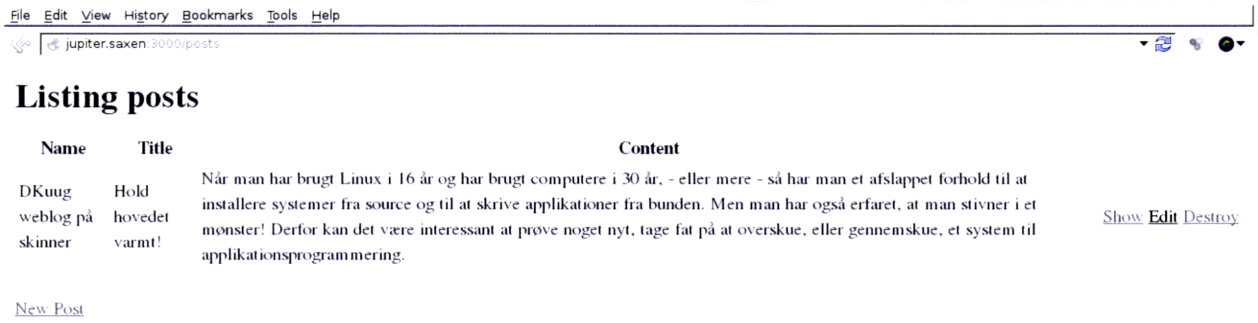


Vi skriver nogle uforglemmelige ord:



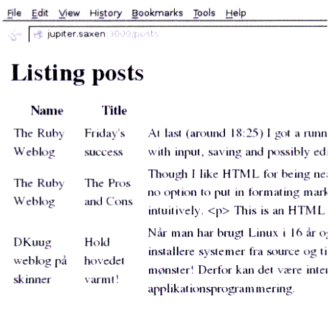


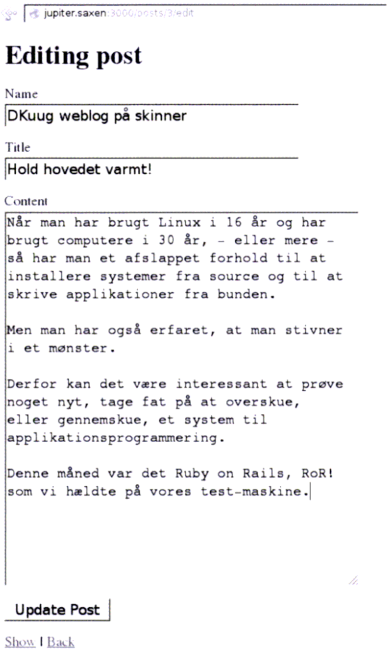
Vælg ***Back:***



Der er kommet indhold i webloggen. Man kan undre sig over at det ser så råt ud. Ruby on Rails skal - selvfølgelig - også kunne ændres med forskellige design, skins, temaer, themes, hvad man nu kalder det, men for programmøren er udseendet i første omgang ikke væsentligt.

Imidlertid kan det være afgørende for at sælge udviklingsmiljøet, at man kan lave en weblog med passende nydeligt design, - et problem med dette er imidlertid, at man kan blive anset for at være grafisk designer og ikke programmør.





For den gennemsnitlige chef eller kunde er det meget vanskeligt at vurdere en IT-medarbejders arbejde og helt umuligt at forstå, hvor kræfterne bliver lagt - indtil chefen en dag brænder sig på en smart sælger med et "smart design".

Man kan både tilføje, og rette og slette. Efter tilføjelse af et par indlæg ser det endnu mere hjælpeløst ud. Afsnit, d.v.s. linieskift, vises ikke i listningen, - men vi ved at det kan gøres på andre måder. Nu vil vi blot prøve editering og sletning, og heldigvis fungerer det alt sammen som det skal.

Ruby on Rails har en stejl indlæringskurve, når man kommer forbi de indledende, grundlæggende trin, og vi her i DKuug kan somme tider blive lidt i tvivl om det er så god en idé at begynde med systemer til generering af applikationer, eller om man i stedet kan komme hurtigere til kørende, vel-designede apps ved at bruge sit gode gamle favorit-programmeringssprog og danne en datamodel og en source-templates.

Men der er ikke tvivl om at der er behov for rammer - en styring af designet i projekter, som er større end en håndfuld funktioner. Som hovedregel kan man sige at efter 4-5 ugers programmeringsarbejde bør resultaterne modulariseres.

Ruby on Rails er et glimrende eksempel på hvordan modularisering kan realiseres.

■

IPv6 i medierne

Hvad siges der om IPv6 - er betydningen og forståelsen af betydningen til- eller aftagende?

Som bekendt var der mange skeptiske røster, selv blandt uddannede IT-experter, vedrørende nødvendigheden af IPv6.

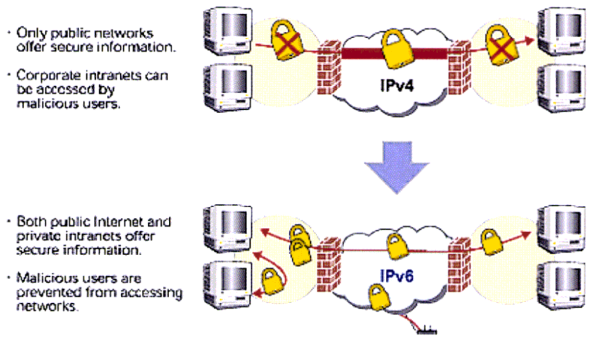
De, der har fulgt med diskussioner om Internettet, har også mærket at der er to trends, en for brug af Internettet som avanceret, *"interaktivt fjernsyn"*, og en anden, som er nærmere den oprindelige idé, at bruge Internettet som en peer-to-peer forbindelse, hvor vi hver især på vores computersite (eller i dag håndholdte computer/smartphone) kan tilbyde andre at logge på og læse hvad vi har liggende.

Der er mange misforståelser, der trives. Har man Internet på en smartphone, hvis man har en browser, der kan "gå på nettet"? Hvad er forskellen på Internet og World Wide Web?

Internettet er et netværk, hvor alle er lige. En smartphone, som ikke har sit eget, globale IP-nummer, kan ikke ses af andre. Det svarer til at have en telefon uden nummer - en, mankan ringe fra, men som ikke kan ringes op. (Det kan være en fordel, indrømmet! ;-))

En smartphone kan kun ses af andre, hvis der er installeret en (spion-) applikation på enheden, så denne transmitterer data og evt. lytter på kommandoer, og vel at mærke selv "trækker kommandoern ind". Den slags kan forlede folk til at tro, at en smartphone kan det samme som de systemer, der udgør det egentlige Internet.

Det er i værste fald en passificering, en begrænsning af indlysende, lette muligheder. Det er imidlertid muligt at forbinde en smartphone med en "rigtig" server, typisk sociale services, - man almindelige brugere kan også forholdsvis let opsætte egen server og derved øge funktionaliteten af smartphones, og det er nok derfor, de fleste ikke tænker så meget over betydningen af at have sit eget, globale IP-nummer.



IPv6 com: øverst IPv4 NAT lokal-netværk, nederst IPv6 med beskyttet lokalnetværk

Denne trend har ændret opfattelsen af IPv6. Det er derfor at selskaber som TDC har kunnet sige at "vi har nok IPv4 adresser indtil engang i 2020."

IPv6 kan skabe bedre sikkerhed for lokale net og smartphones. Derfor har fremsynede organisationer og regeringer sat tryk på IPv6.

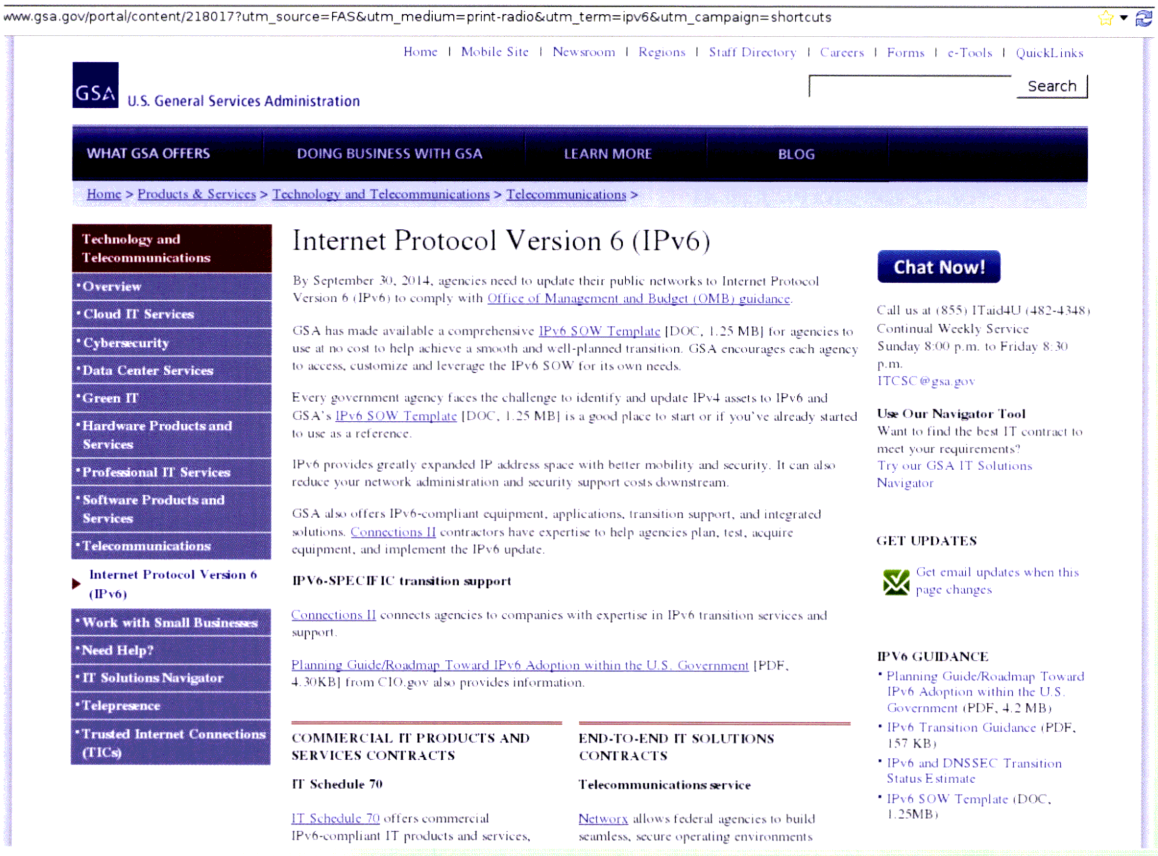
US General Services Administration

http://www.gsa.gov/portal fortæller at pr. 30 september 2014 er det nødvendigt at administrationer under United States skal have netværk, som kan kommunikere via IPv6. Begrundelsen er, at antallet af brugere, som ***kun*** har IPv6 vil begynde at stige, og klart nok skal administrationerne ikke halte bagefter.

Desuden anfører GSA, hvordan man kan få hjælp til IPv6 - med link til virksomheder, som kan give bistand eller komplette løsninger.

Tænk hvis danske virksomheder også fik den slags ordrer og vi derved var på forkant med ny teknik!

■



IT Sikkerhed

Hvordan beskytter du din WordPress-blog?

Af Lars Sommer

Foto: Claus Bech

WordPress er verdens mest populære CMS i brug på internettet [1], og personligt har jeg også en lille række af sites der kører WordPress. Tilbage i 2007-2008 blev en hel del sårbarheder opdaget, vendt, drejet og lappet, og siden da, har grundsystemet været rimeligt. Men med populariteten følger også et enormt antal af moduler, som ikke alle er sikkerhedstestede lige godt. Som en del af mit job følger jeg dagligt med i, hvad der opdages af sårbarheder, og hvad der bygges af exploits til at udnytte sårbarhederne med. Den sidste måned, er der kommet 60 exploits til moduler i WordPress. Det er 2 om dagen, og det er et enormt højt tal.

Så udover at følge med i, hvilke moduler der opdages sårbarheder i, har jeg følt det nødvendigt, at finde muligheder for at sikre mine sites, uden selv at skulle holde øje med opdateringer i tide og utide. Jeg har kravlet bjerget af plugins til WordPress igennem, i forsøg på at komme rundt i de fleste aspekter af sikkerhed.

Her er hvad jeg har fundet anvendeligt, og pt. benytter:

Meta Generator and Version Info Remover, tidligere *Secure WordPress* [2]: En all-round-forhøjning af sikkerheden på sitet. Det blokerer underlige forespørgsler, fjerner informationer om versionsnumre og detaljerede fejlbeskeder, der typisk gør det nemmere for hackere at finde informationer omhvilke exploits der er brugbare.

Mute Screamer [3]: En implementation af PHP-IDS, et Intrusion Detection System til PHP-applikationer. Mute Screamer inspicerer de forespørgsler brugerne, og angriberne, sender til WordPress. Det er nemt at skrue på forskellige tærskelværdier, for f.eks. hvor meget der skal til for at sende en angriber til en fejlside, eller direkte på en blacklist. Det er også til at få tilsendt emails ved forsøg på angreb.

Akismet [4]: Et meget kendt modul, der tjekker alle kommentarer for spam. Deter ikke direkte beskyttende mod hacking, men det er effektivt til at forhindre atder ender en masse spam på dit site. Hvis en angriber ser et site fyldt med spam i kommentarerne, virker det oplagt at sitet måske er mindre vedligeholdt, og dermed mere sårbart overfor alvorligere angreb.

Update Notifier [5]: Tjekker dagligt alle dine installerede moduler (og eventuelt temaer) for opdateringer, og sender dig en email, hvis der er opdateringer duendnu ikke har installeret. Det er en nem måde at vedligeholde WordPress på, uden selv aktivt at skulle holde øje hele tiden.

WordPress Database Backup [6]: dækker det backup-mæssige aspekt af sikkerheden. De flade filer i WordPress laver jeg backup af via FTP. Men databasen synes jegdet var lidt mere bøvlet at få jævnlig backup af. Da det typisk er databasen der ryger, ved et SQL Injection angreb, er det vigtigt at tage backup af denne. WordPress Database Backup kan indstilles til f.eks. at sende en databasebackup afsted premail, hver uge.

Liste over sikkerheds-plugins, screen-dump fra Wordpress.org

Et andet plugin, Look-See Security Scanner, kan kontrollere om der er filer, som er blevet modificeret. Er det tilfældet, er hele installationen formentlig kontamineret.

Herudover bruger jeg en række andre plugins til bl.a. kontaktform, Search Engine Optimization, automatiske Google-oversættelser, caching og XML-sitemaps. Men de har væsenligt mindre med sikkerhed at gøre.

Hvad gør du for at beskytte din WordPress?

Har du andre gode WordPress-moduler, eller har du gode tips til at sikre WordPress-serveren andre steder? -- F.eks. et IDS/IPS til databasen eller lignende? Så rapporter til Wordpress.org.

---

[1]: http://en.wikipedia.org/wiki/WordPress

[2]: https://wordpress.org/plugins/meta-generator-and-version-info-remover/

[3]: http://ampt.github.com/mute-screamer

[4]: http://akismet.com/

[5]: https://wordpress.org/plugins/wp-updates-notifier/

[6]: http://austinmatzko.com/wordpress-plugins/wp-db-backup/

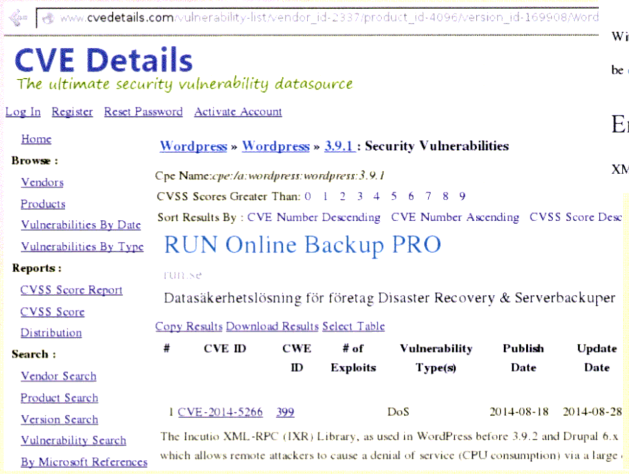
Sikkerhedshuller fra supportlibraries kan ses registreret på

CVE: http://www.cvedetails.com/vulnerability-list/

CERT: https://www.us-cert.gov/

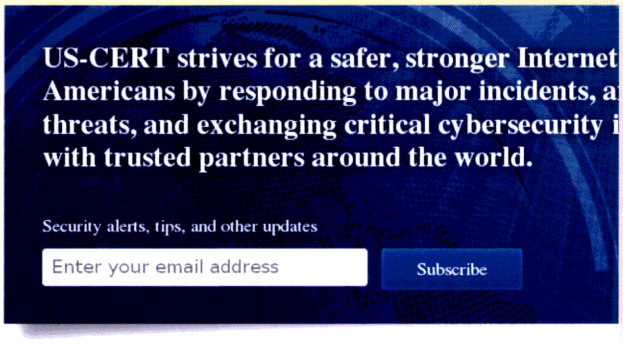
Seruriteam: http://www.securiteam.com/

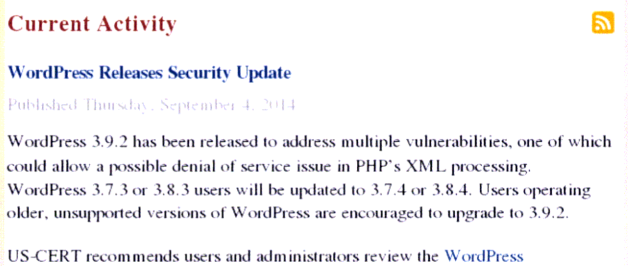
Nogle af de sikkerhedshuller, man har fundet indenfor XML og XML-RPC support, er lukket fra Wordpress version 3.9.2 som har egen implementation af XML. XML bruges bl.a. ved indlæsning af indhold fra gamle weblogs, som skal tilrettes til ny version af Wordpress. Nyeste version er 4.0 (så CERTs nyhed er lidt bagefter). Der er poleret lidt mere på 4.0, skriver Matt Mullenweg, ellers er 4.0 bare et tal som andre tal, og ét, som kommer før 4.1 ☺



Et eksempel på en weblog, baseret på Wordpress, er websitet med plugin til backup af databasen, som Wordpress bruger:

Da vi gerne ville vise mange screen-dumps har vi måttet klippe lidt i kanten nogle af billederne.





CERTs nyhed kommer lige før release af 4.0 med indbygget XML support

Sikring af Wordpress er et første skridt, men der vil altid være mulighed for at en cyber-kriminel finder nogle andre sårbarheder eller - at maskineriet bryder sammen og man taber alle sine data.

I så fald er backup en ekstra sikring, så har man i det mindste en rimelig frisk backup.

Backup til en extern disk, der fjernes og opbevares et andet sted, og ikke i en oversvømmelig kælder, er at foretrække.

■

Labitat laver ting

Labitat, navnet spiller på flere dimensioner, stedet er laboratorium for gør det selv-lystne, og det er et levested, et habitat, for nørder

I bio-rummet er der eksperimenter med forskellige enzymer

På H.C.Ørstedsvej kan man hver aften, og somme tider også om dagen, finde medlemmer af Labitat, som arbejder med hardware og typisk med digital hardware. Der er dog også andre ting, så som biologiske experimenter i bio-rummet.

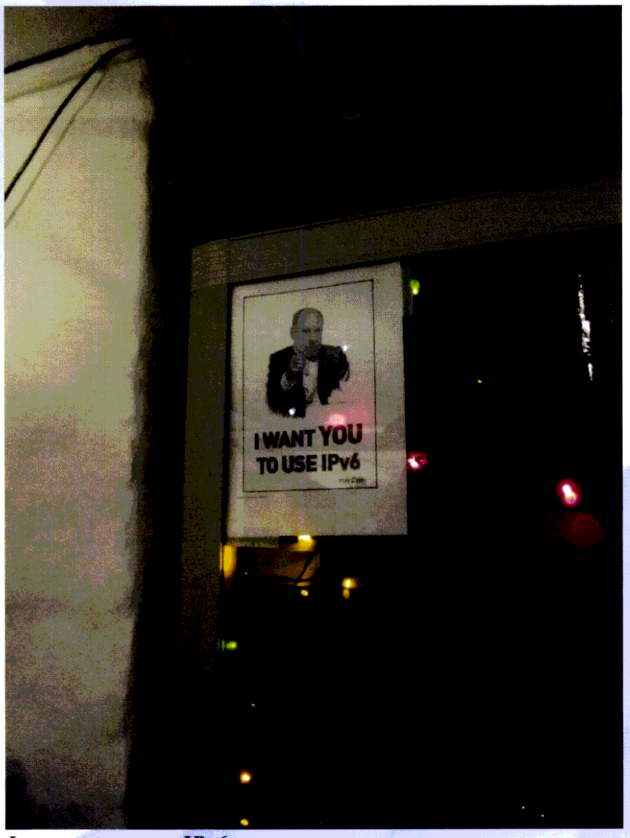
Døren til foreningens lokaler på H.C. Ørstedsvej på Frederiksberg (i København) har kodelås (også af egen produktion, programmeret og opsat af Labitat medlemmer) men den er åben om tirsdagen, så er alle velkomne uden aftale forud. De andre af ugens tidspunkter kan man som regel også komme ind som gæst, blot man kender én af medlemmerne og får ham/hende til at åbne døren.

Labitat har deltaget på de seneste Open Source Days, og vi har i DKuug støttet dem med mindre beløb.

3D-printer samles

Der er serverskab og køleskab i driftsrummet

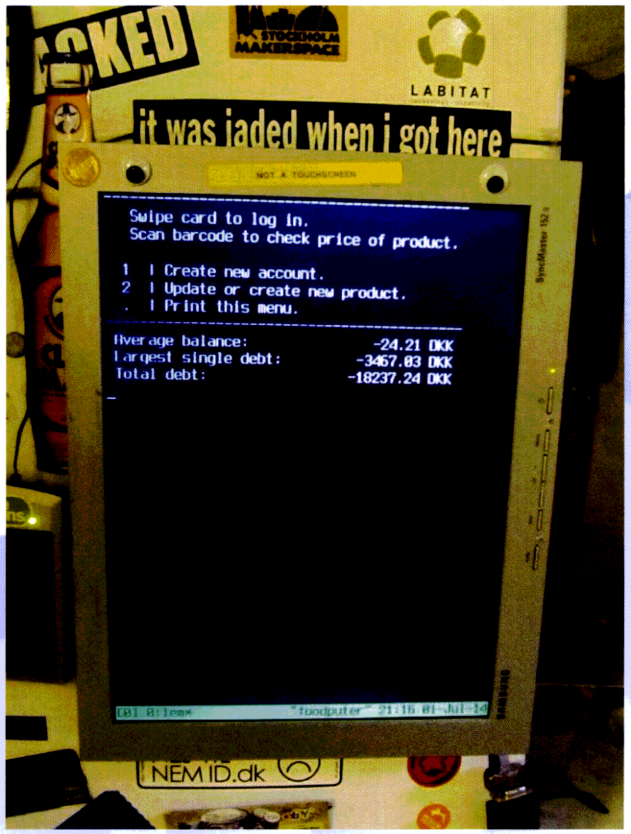
Labitat har haft en holdning til moderne tekniske landvindinger, som er meget god til at afmystificere den moderne teknik, såsom fx. mikro-videocam til sonder, 3D printere, routere, der er lavet af almindelig hardware og Linux-systemer.



I want you to use IPv6 now

Memory-logo

Labitat har sit eget website og er til stede på Facebook, IRC, Flickr, Twitter, og har releaset videoer på Youtube, *Labitat on a Tuesday evening, Kulturnat 2013, Moving the Roskilde Container to Krausesvej, Today Riiiis made in Labitat,* og mange flere.



Labitats eget drikkevare-regnskab

På labitat.dk fortælles: Labitat is a **hackerspace** in CPH. We are a group of people with diverse interests in technology. We are an independent physical space, working creatively with techonology, bridging interactive technology with design and art.

... and we love new faces!

Just Visit Us!

No need to hesitate. We have coffee! If you have found this website, you're probably interested in our space. And: If you are interested, just drop by.

Labitat har deltaget i OSD 2012 og 13, og har udviklet en platform for undervisning i grundlæggende lodning og programmering af et stand-alone modul, baseret på Arduino (en single board microcontroller/computer baseret på flere CPU'er - her skal blot nævnes 32-bit ARM.)

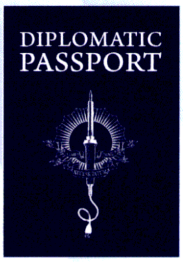
Der har været afholdt workshops og i kalenderen kan man følge med i kommende events.

På https://github.com/labitat/doorduino kan man finde programmet til dørens adgangssystem, og til andre projekter, bl.a. drinkomatic til *foodputeren!*

Zapro's NixieClock (fra Labitat Copenhagen, video på YouTube,* Labitat on a Tuesday evening*: A look around the space during one of our regular open night Tuesdays)

Hackerspace pas kan udstedes! De ligner US pas, men de er lavet for at opfordre til at krydse grænser. Filosofien er:

At opfordre til at folk besøger mange hackerspaces rundt om i verden, øge samarbejde, øge krydsbestøvning med gode idéer, og anspore hackerspaces til at hjælpe hinanden samt - selvfølglelig - at have det sjovt! ■



Fotograf: Malthe Borch

Labitats adresse:

H. C. Ørsteds Vej 5, kld

1879 Frederiksberg C

Denmark

Website: https://labitat.dk/

LABITAT
technology · creativity

Logo fra Wikipedia og Labitats website, med tak til Renolm

(fortsat fra side 2 - Et brev fra redaktionen)

Det mest charmerende ved Rails er, at det er nemt at komme igang med. Derefter bliver indlæringskurven stejlere. Hvis man vil ud over de grundlæggende applikationers muligheder, skal man være dybt inde i klasserne og sprogets syntax for at forstå, hvad der sker. Mange websites, bøger og video'er hjælper én igang.

Den dybere forståelse af, hvad der sker, kan man få her:

http://andrewberls.com/blog/post/rails-from-request-to-response-part-1--introduction

og https://www.railstutorial.org/book/beginning

Man kan somme tider spørge sig selv, om ikke *Rails* kunne præsenteres som peg-og-klik IDE (Integreret Development Environment) og minsandten jo, det kan man også få, endda kvit og fri Open Source: ***Aptana RadRails*** IDE for *Rails* applikationer (engelsk). ***RadRails*** inkluderes som del af ***Aptana Studio 3***.

http://www.aptana.com/products/radrails

Andre Software Managere

Software (eller pakke-) management generelt er et system, der sørger for at man kan køre de applikationer, man har brug for. Hvis ikke software-manageren kan håndtere afhængigheder af versioner - "mindt version 6.1" og den slags - så er brugeren ilde stedt.

Selv om det så kun er for *Ruby*, så er ***rvm*** nyttigt og imponerende, for der er masser af pakker og source til klasser, som man inkluderer, når de skal bruges - ligesom ***Perl***, der har ***Comprehensive Perl Archive Network*** (CPAN) og tilhørende applikation, som kan opdatere de fleste ting automatisk.

Det har længe været et ønske at få en eller flere skribenter til at give et bidrag til en serie om software management. Det skulle først og fremmest være for at finde den grundlæggende design idé bag de mange software management systemer: Debians ***dpkg*** med front end ***apt-get, aptiture*** m.fl. ***Red Hat Package Manager*** (RPM), ***yum, pacman, paludis*** og Gentoo ***Portage***, der ligesom RPM og dpkg kan køre på flere distroer.

Derudover er der nogle, der overskrider grænsen mellem software management og distribution, fx. ***pkgsrc***. Og der er flere end de nævnte - Bladet kunne fyldes flere gange med facts og fordele og bagsider ved disse pakkesystemer.

Den ønske-artikel skal have eksempler og illustrationer og kommentarer om, hvordan systemerne håndterer afhængigheder, dependencies og her får vi brug for læsernes erfaringer. Skriv! Eller gå på IRC!

Men er softwaremanagement nødvendigt? ***Hvor*** nødvendigt? Som eksempel på systemer, som medbringer det nødvendige selv, og sætter ***ld_library_path*** mv. kan nævnes Firefox og Open Office. Begge systemer kan hentes som arkiver, der blot skal udpakkes et passende sted, de lægger sig i eget subdirectory. Det fungerer så godt, at pakkerne kan køre på stort set alle kurante Linux-distro'er.

Hvorfor standardiserer man ikke?

Men mange distro'er har deres ***egen*** version af fx. Firefox, plantet i /usr/bin og med hjælpefiler splattet ud over /usr/lib med videre. Er det rationelt?

Hør hvad Bryan Lunduke mener om det i et foredrag, som hedder "Linux sucks" 2014 (find det på Youtube ved at søge på Linux Sucks), der er mere end vi citerer her, og det er kærligt ment) om at de forskellige distro'er pakker og pakker og pakker:

Idet han peger på kurver over hvor mange downloads der er via Distrowatch på de forskellige distributioner siger han:

"I et span af 2 år kan hele Linux øko-systemet ændre billede - en lille nobody distribution (Elementary) overhaler en stor del af disse bedstefædre-distro'er, Fedora, Redhat osv.

Det er forbløffende, når man tager i betragtning hvor meget arbejde de store leverandører bruger på at lave nye versioner - det er et enormt arbejde, som hvert år - måske et par gange om året - går med at teste, pakke, brande, tilføje nye features, rette fejl, og for alle de distributioner må de repetere det - pakke - om og om igen.

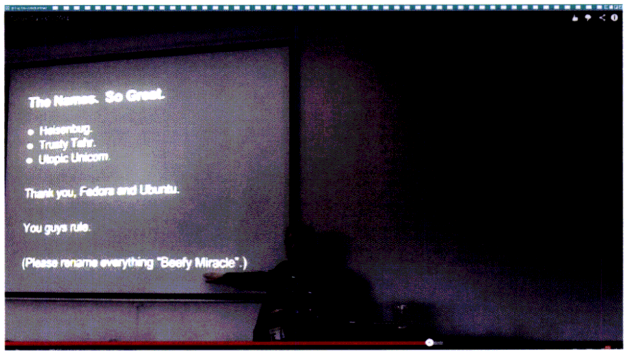
Og for hver af disse mange, mange distributioner må der være nogen, som **gentager disse ting om og om igen.** Så vi kan konstatere, at vi har en stor mængde arbejde i Linux-community, som går med at pakke, ompakke, pakke igen og igen. Det er gentagelsesarbejde i stedet for kreativt arbejde."

"Der er to problemer med det", siger Bryan Lunduke, "1) det indlysende, vi kunne ønske at denne hærskare af mennesker arbejdede på nyttige features i Linux, - 2) det er kedeligt arbejde!"

Han spørger publikum, hvor der er to, der pakker for en distro: "Ville pakning af 2000 programmer stå højest på en liste over ting, du allerhelst ville foretage dig næste week-end? "

"Har Apple det problem? Har Microsoft det problem?"

"Man kunne standardisere på et interface - fx. Red Hat eller Fedora, og det siger jeg skønt jeg ikke er særlig begejstret for Red Hat!" grynter Bryan Lunduke.



Bryan Lunduke gør grin med versionernes kælenavne, Utopic Unicorn - wott? Please rename everything "Beefy Miracle"!

Men det er jo klart at distributionerne har et ønske om at tjene penge - med undtagelse af Debian og måske Gentoo - og at man af en eller flere grunde ønsker at beskytte sit pakke-system mod ting, som ikke passer ind i og ender med at få andre ting til ikke at fungere. Det er - kort sagt - versionshelvedet, man prøver at undgå.

For at være retfærdig må vi også huske at Microsoft efter nogle år, hvor de ***frøs*** en MSwindows version, begyndte at lave service packs, sikkerhedsopdateringer med videre - måske for at have mere hånd i hanke med deres kunder.

Det samme er ikke tilfældet for Linux-distroer, her tjener opdateringerne altid to formål: Flere features, færre fejl. Typisk fx. ved installation af wireless-drivere, noget som for bare 4-5 år siden kunne volde problemer; i dag går det som en leg, siger Bryan Lunduke - undtagen hvis man køber rådden hardware!

I anden halvdel af foredraget vender Bryan Lunduke synspunktet og går gennem de samme slides, denne gang med en ros til den opfindsomhed og den energi, som ligger til grund for de mange distributioner.

Det er historien om det halvt fulde glas vand, ændret til at handle om Linux og udstrakt over en timestid. Meget opløftende! ■

Donald Axel

Kommandocentralen

Hvor meget skrives der på disk?

Programmet ***top*** giver et hurtigt overblik over hvilken applikation, som lige nu er den mest CPU-belastende:

Men hvordan får vi at vide, hvor meget systemets diske er belastet? og hvor meget disk den enkelte proces bruger?

Atop er en "luksus-udgave" af top. Lige når programmet starter op, giver det en oversigt over systemet fra boot til nu, men efter 10 sek. får man den aktuelle status. Atop venstr kolonne viser system-udnyttelse (PRC og CPU), memory, disk-og netbelastning:

```
ATOP - sat2                              2014/09/10  11:31:14                                  ------                                  10s elapsed
PRC |  sys    1.48s |  user   3.54s |  #proc    175 |  #trun      1 |  #tslpi    312 |  #tslpu      1 |  #zombie     1 |  clones     1 |  #exit       0 |
CPU |  sys      11% |  user     34% |  irq       0% |  idle     88% |  wait     67% |  steal      0% |  guest      0% |  avgf  955MHz |  avgscal   31% |
cpu |  sys       7% |  user     25% |  irq       0% |  idle      7% |  cpu001 w 60% |  steal      0% |  guest      0% |  avgf  984MHz |  avgscal   32% |
cpu |  sys       4% |  user      9% |  irq       0% |  idle     81% |  cpu000 w  7% |  steal      0% |  guest      0% |  avgf  927MHz |  avgscal   30% |
CPL |  avg1   0.37 |  avg5   0.15 |  avg15  0.08 |                  |               |  csw    17594 |  intr    8985 |               |  numcpu      2 |
MEM |  tot    3.6G |  free 137.7M |  cache  1.7G |  dirty    0.4M |  buff  273.3M |  slab  973.9M |              |               |               |
SWP |  tot    0.0M |  free   0.0M |              |                  |               |               |              |  vmcom   1.4G |  vmlim   1.8G |
PAG |  scan   7532 |              |  stall     0 |                  |               |               |  swin      0 |               |  swout       0 |
DSK |          sdb |  busy     82% |  read   2680 |  write      0 |  KiB/r     63 |  KiB/w      0 |  MBr/s  16.52 |  MBw/s   0.00 |  avio 3.07 ms |
DSK |          sda |  busy      0% |  read      0 |  write     17 |  KiB/r      0 |  KiB/w      4 |  MBr/s   0.00 |  MBw/s   0.01 |  avio 0.00 ms |
NET |  transport   |  tcpi     54 |  tcpo     54 |  udpi       0 |  udpo       0 |  tcpao      0 |  tcppo      0 |  tcprs       0 |  udpip       0 |
NET |  network     |  ipi      59 |  ipo      59 |  ipfrw      2 |  deliv     57 |               |              |  icmpi       0 |  icmpo       3 |
NET |  eth1    0%  |  pcki      5 |  pcko      5 |  si    0 Kbps |  so    0 Kbps |  erri       0 |  erro       0 |  drpi        0 |  drpo        0 |
NET |  eth0    0%  |  pcki      1 |  pcko      1 |  si    0 Kbps |  so    0 Kbps |  erri       0 |  erro       0 |  drpi        0 |  drpo        0 |
NET |  lo    ----  |  pcki     54 |  pcko     54 |  si   14 Kbps |  so   14 Kbps |  erri       0 |  erro       0 |  drpi        0 |  drpo        0 |
NET |  ppp0  ----  |  pcki      4 |  pcko      4 |  si    0 Kbps |  so    0 Kbps |  erri       0 |  erro       0 |  drpi        0 |  drpo        0 |

  PID  RUID        EUID        THR    SYSCPU    USRCPU    VGROW    RGROW    RDDSK    WRDSK  ST  EXC  S  CPUNR  CPU  CMD                1/1
15779  root        root          1     0.49s     2.32s       0K       0K   165.5M       0K  --    -  D      1  28%  md5sum
15214  root        root         34     0.12s     0.93s       0K    5536K       0K      20K  --    -  S      1  10%  firefox
 4323  root        root          1     0.35s     0.24s     -16K       0K       0K       0K  --    -  S      0   6%  Xorg
   51  root        root          1     0.23s     0.00s       0K       0K       0K       0K  --    -  S      1   2%  kswapd0
 4521  root        root          1     0.09s     0.02s       0K       0K       0K       0K  --    -  S      1   1%  xosview
15748  root        root          1     0.05s     0.01s       0K       0K       0K       0K  --    -  R      0   1%  atop
 4491  donax       donax         2     0.03s     0.00s       0K       0K       0K       0K  --    -  S      0   0%  xchat
    3  root        root          1     0.03s     0.00s       0K       0K       0K       0K  --    -  S      0   0%  ksoftirqd/0
    8  root        root          1     0.03s     0.00s       0K       0K       0K       0K  --    -  S      1   0%  kworker/1:0
    9  root        root          1     0.02s     0.00s       0K       0K       0K       0K  --    -  S      1   0%  ksoftirqd/1
 8983  root        root          1     0.02s     0.00s       0K       0K       0K       0K  --    -  S      0   0%  kworker/0:0
 3672  mysql       mysql        17     0.00s     0.01s       0K       0K       0K       0K  --    -  S      0   0%  mysqld
11315  root        root          1     0.00s     0.01s       0K       0K       0K       0K  --    -  S      1   0%  xv
```

Problemet med den slags programmer er, at de er overvældende, de giver alt for meget information.

Man skal være god til at bruge man-pages for at finde ud af hvad øverste linie betyder:

Sys, user og angiver hvor meget tid systemet har brugt i system (kerne-) modus, user-modus og #proc er antallet af processer, som kører lige nu.

Derefter kommer trun, antal threads, som kører lige nu, tslpi, threads som sover (sleep->slp) og som er interruptible - tslpu uninterruptible.

En forståelse af de tal kræver at man ved, hvilken virkning tråd-processer har på system-performance (de giver bedre response i GUI og belaster mindre end en selvstændigh proces) og hvad interrupt, afbrydelser, bruges til og hvad de "koster" i tid (fx. interrupt fra en disk: IO-operation er færdig). Hvis det er noget, som man får brug for, må man læse en grund-læggende bog om CPU arkitektur og lære bare en lille smule assembler-instruktioner.

Et godt sted at begynde er

https://en.wikipedia.org/wiki/Machine_code

Desværre har jeg ikke kunnet finde en dansk web-introduktion. Emnet hører til på ingeniør-uddannelser. Der burde være en dansk Open Source introduktion til emnet frit tilgængelig. DTU:

http://www.kurser.dtu.dk/2014-2015/62515.aspx?menulanguage=da

Derefter kommer antal *zombie* - processer, som er afsluttet, men som forældreprocessen ikke har taget exitkoden fra endnu. *Clones* er antallet af clone() system kald, typisk for at generere en thread. Exit-koder er den 8-bit kode, som et program afleverer efter afslutning.

CPU-linien viser at 1.48 sekunder er ca 11 procent af de ca. 10 sekunder, som er brugt til måling af CPU-belastning. *irq* tallet er procent tid brugt på interrupt-service (midlertidig afbrydelse fx. for at hente input fra tastatur) Desuden vises tomgang, IO-venten, frekvens og scaling. Guest er ikke relevant på det viste output - det er tiden brugt af virtuelle hosts under den aktuelle.

Der er to kerner på den maskine, som er vist, og der vil da være en linie for hver af kernerne - kolonnen med IOwait har også CPU-nr: cpu001 kommer før cpu000 i eksemplet.

Man kan se en linie, som er rød. Det er en ny linie i forhold til forrige status, og den viser at disk-systemet på disk B er gået igang og har været aktivt 82% af tiden. Der er læst 63 gange (read requests) og der er læst 17 MB i sekundet.

System-overblikket viser ikke temperatur - man er nødt til at køre LMsensors og hddtemp (som omtalt i DNyt-174).

Hvor hurtigt er disk-systemet

Hvis disk systemet skriver 1 MB i sekundet, hvor længe tager det så at skrive 1 GB? ***16,7 MINUTTER!*** ovf. står at systemets læst 16.5 MB i sekundet - men det er jo et gennemsnit, og hvis vi vil vide, hvad systemet ***kan,*** må vi bruge programmet hdparm. Prøv flg.:

```
# ls -l /dev/sd?
# # hvis der ikke er nogen diske, så stop!
# hdparm -I /dev/sda
/dev/sda:
ATA device, with non-removable media
        Model Number:       WDC WD20EURS-63Z9B1
        Serial Number:      WD-WCAVY6066531
        Firmware Revision:  80.00A80
        Transport:          Serial, SATA II ...
Standards:
        Supported: 8 7 6 5
        Likely used: 8
Configuration:
        Logical         max     current
        cylinders       16383   16383 [klip!]
# hdparm -tT /dev/sda/
dev/sda:
 Timing cached reads:   2972 MB in  2.00
seconds = 1485.97 MB/sec
 Timing buffered disk reads: 298 MB in  3.01
seconds =  98.97 MB/sec
#time dd if=/dev/zero of=/nulf bs=1M count=1000
1048576000 bytes (1.0 GB) copied, 9.45604 s,
111 MB/s [...klip]
```

SUPERUSERS

To kurser der rykker udviklere og projektledere op i superligaen

SU-0159 Git Versionsstyring

Om Git:
Git blev skabt af Linus Torvalds til udvikling af Linuxkernen og er siden blevet verdens mest anvendte distribuerede versionsstyringssystem (DVCS) på tværs af platforme og udviklingsmiljøer (IDE'er). Git bruges bl.a. i Visual Studio fra Microsoft, i Eclipse og Android Studio fra Google samt i Xcode fra Apple ...

Hver eneste Git-rodmappe er et komplet versionsstyret repo som ikke afhænger af netværksadgang eller en central server.

GIT bruges i dag både til versionsstyring af software-projekter og systemadministration.

Forudsætninger:
Følgende kurser eller tilsvarende viden:
- SU-199 Introduktion til programmering

Om kurset:
Du lærer at anvende Git til versionsstyring af software-projekter, konfigurationsfiler m.m. samt at opsætte Git repo og administrere Git. På kurset gennemgås alle de ting, som du skal vide for at bruge Git i dit daglige arbejde.

Lærebøger:
SuperUsers kursusmateriale på dansk.

Varighed / Pris:
2 dage / 7.400,- kr. (ekskl. moms)

Der er endnu ledige pladser på følgende datoer:
- 27.–28. november

SU-0205 UML (Unified Modelling Language)

Om UML:
Den objektorienterede tankegang udvider og forbedrer de traditionelle metoder til systemudvikling. Unified Modeling Language (UML) bygger på mange udvikleres praktiske anvendelse af OO-metoder, og som den vigtigste OO-metode danner den grundlag for fremtidens standarder i objektorienteret udvikling. UML anvendes både til design, analyse og dokumentation.

Forudsætninger:
Følgende kurser eller tilsvarende viden:
- MS-483 Programming in C# *eller:*
- SU-203 C++ Programmering Grundkursus *eller:*
- SU-208 Objective C Programmering *eller:*
- SU-210 Java Programmering Grundkursus

Om kurset:
Kurset dækker hele udviklingsprocessen med UML, og du lærer at opbygge en model – som udtrykker og afklarer brugerkrav – og så videreudvikle den i designfasen og dermed forberede den til implementering.
Kurset indeholder praktiske øvelser – både individuelle og teamopgaver – og henvender sig til både udviklere og projektledere.

Lærebøger:
SuperUsers kursusmateriale på engelsk.

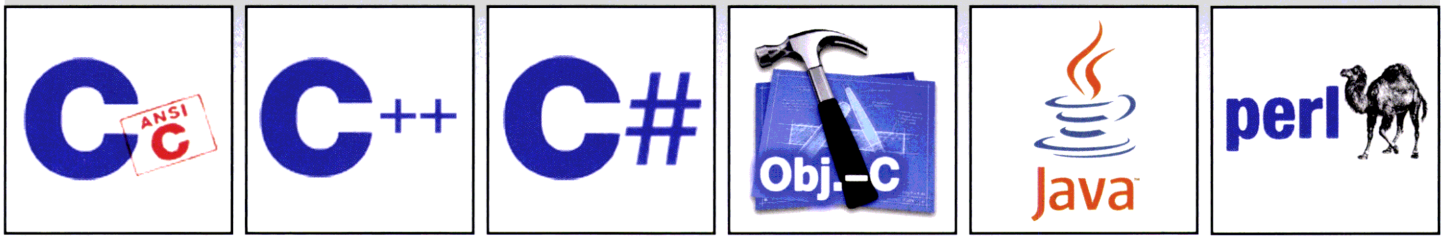
Varighed / Pris:
2 dage / 7.400,- kr. (ekskl. moms)

Der er endnu ledige pladser på følgende datoer:
- 9.–10. oktober
- 15.–16. december

Du kan tilmelde dig på flere måder:

- **Via web:** **superusers.dk**
- **Via mail:** **super@superusers.dk**
- **Via tlf./fax:** **48 28 07 06 / 48 28 07 05**

Har du brug for kurser i sprog? Så se superusers.dk/programmering.htm



Karlebogaard: Karlebovej 91 • 3400 Hillerød & **Kampehøjgaard**: Krajbjergvej 3, Vorre • 8541 Skødstrup
Tlf.: 48 28 07 06 • **Mail:** super@superusers.dk • **Web:** www.superusers.dk • **Alle priser er ekskl. moms**