|  |
| --- |
| Enkel dokumentasjon av PG4300-eksamen |
| Martin Lehmann |
| Trekker frem hvert enkelt krav som stilles av oppgaven, og viser hvordan jeg mener besvarelsen min løser problemene. |

5/28/2013

## 2. Vurderingskriterier

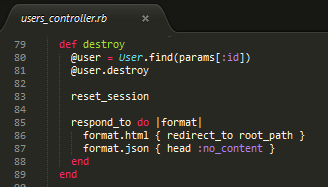
### **Konsis og lesbar kode med riktig innrykk, formatering, og navngiving**

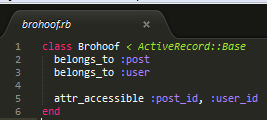
Koden følger standard for både innrykk, formatering, og navngiving i henhold til språk og rammeverk. Det finnes ikke noen klar konvensjon for innrykk for access modifiers (<http://fabiokung.com/2010/04/05/ruby-indentation-for-access-modifiers-and-their-sections/>). Antall spaces per indent varierer mellom 2 og 4 fra fil til fil, men ikke i de enkelte filene; det er rett og slett småplukk som ikke går ut over lesbarheten og tar lang tid å rette opp i.



### **Idiomatisk kode og smarte løsninger**

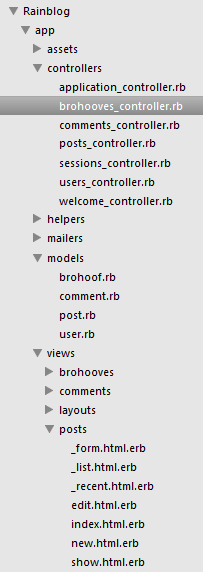
Koden følger standarder for språk og rammeverk, og hjelpefunksjoner som allerede tilbys er stort sett benyttet.

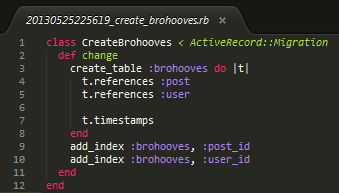




### **Konvensjoner i Rails er benyttet fremfor ekstra konfigurasjon**

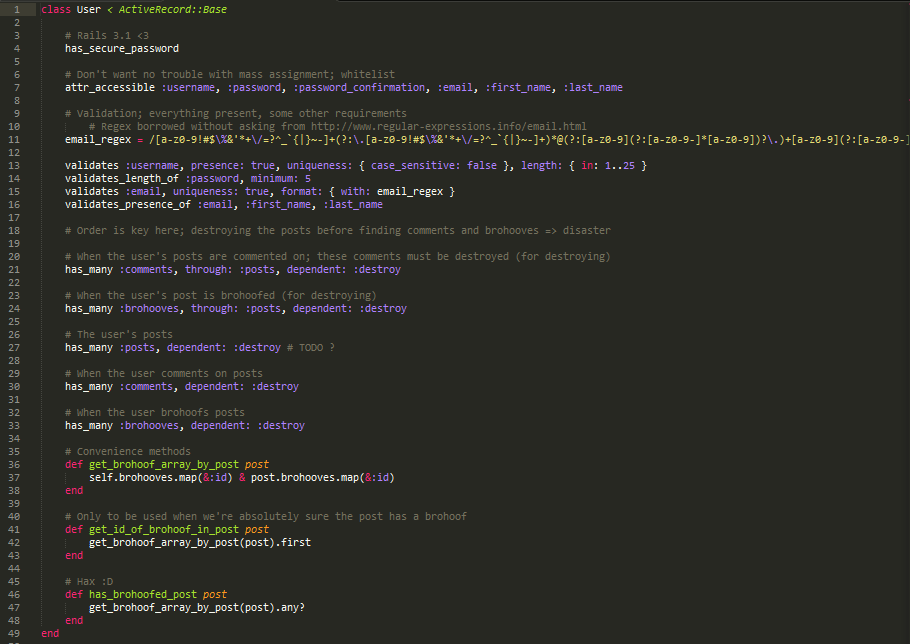
Filstruktur, navngiving, og struktur i klassene følger konvensjoner. To eksempler her er at jeg ikke på noe tidspunkt spesifiserer hva som skal rendres av de standard action-ene i Controllers, og at jeg benytter references-typen for å skape avhengigheter mellom klasser uten å spesifisere foreign key-navn.

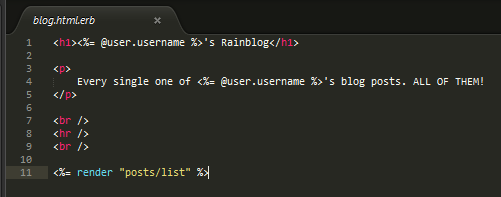


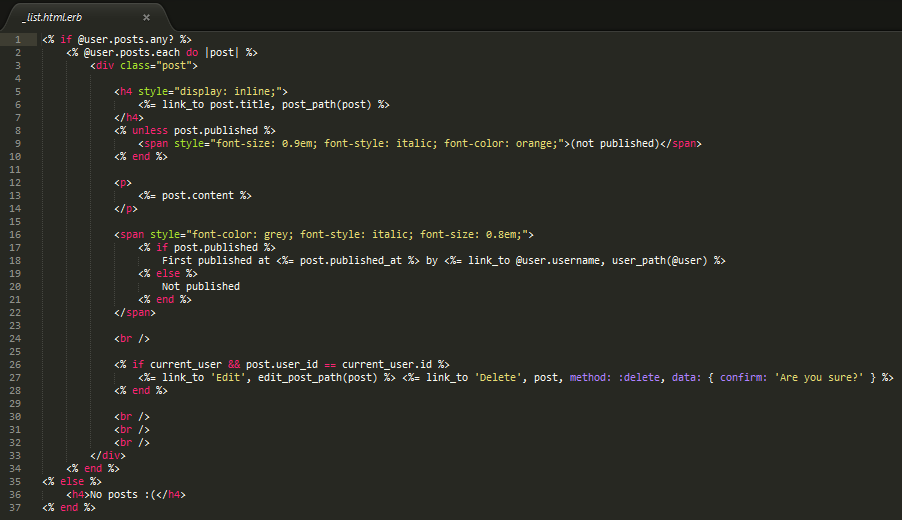


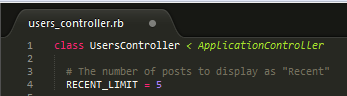
### **Fornuftig fordeling av kode mellom Model, View, og Controller. Tenk DRY, og bruk view partials, view helpers og filtre dersom det er fornuftig.**

I’ll say.





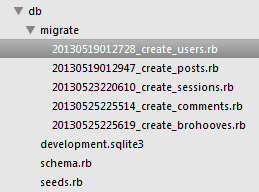


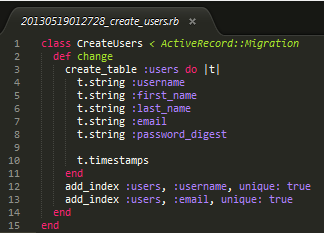




### **Migrations brukes for endringer mot databasestruktur**

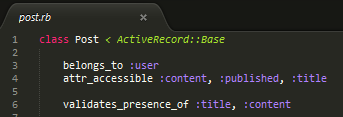
Yup.

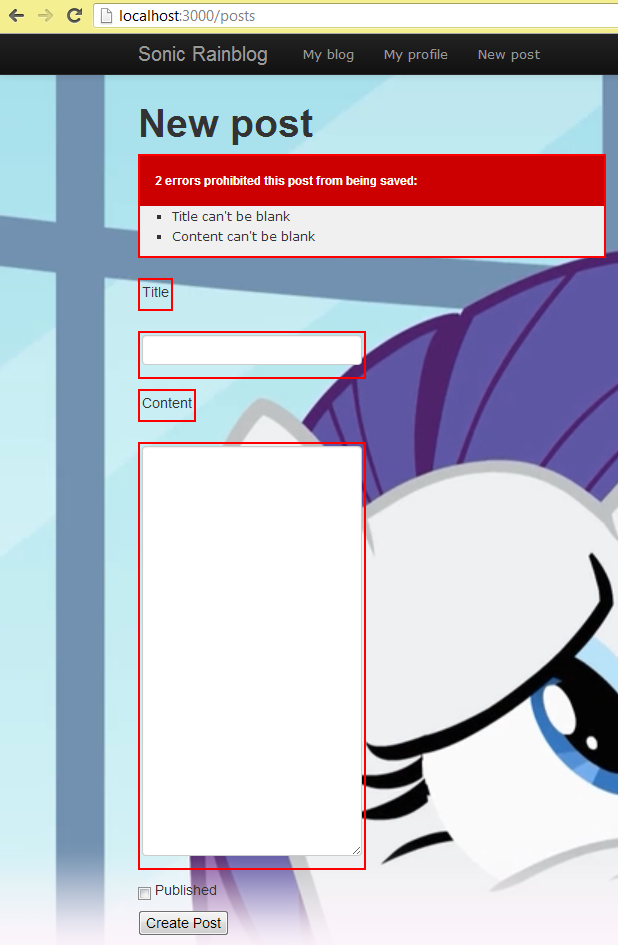




### Validering av input benyttes for å sørge for at bruker sender inn gyldig input

Benytter Rails’ standardmetoder for validering.



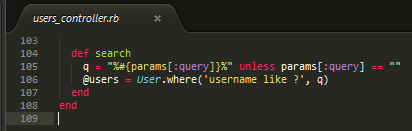


### Applikasjonen er sikret mot SQL injections og XSS

Rails 3 har XSS prevention på som default (<http://railscasts.com/episodes/204-xss-protection-in-rails-3>).

Det er hele veien benyttet prepared statements for databasespørringer.



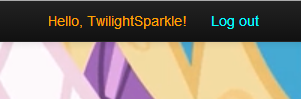


## 4. Oppgavebeskrivelse

### Oppgave 1: Brukerregistrering og autentisering

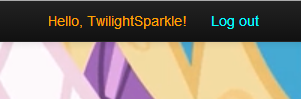
#### Hvis logget inn, link til utlogging

Det er mulig å trykke på Log out. Dette destroy-er sesjonen, og logger på denne måten brukeren ut av systemet.



#### Hvis logget inn, link til profilsiden for innlogget bruker

Det er mulig å trykke på både brukernavnet oppe til høyde og «My profil»-lenka oppe til venstre for å komme til egen profil.

#### Hvis ikke logget inn, link til registrering av ny bruker

Fluttershy er å finne oppe til høyre (i /app/views/layouts/application.html.erb), og er tilgjengelig fra alle undersider.

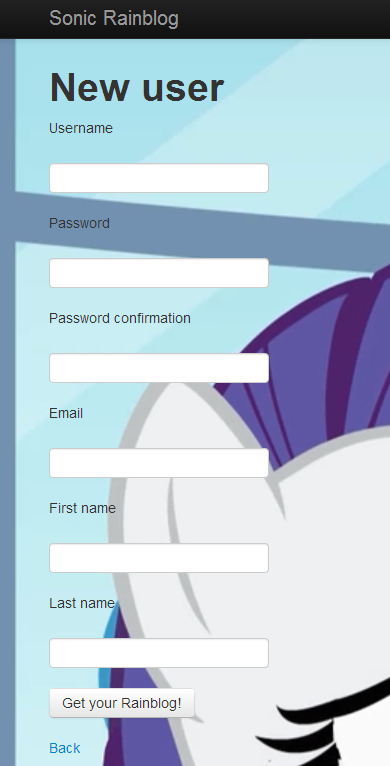
 

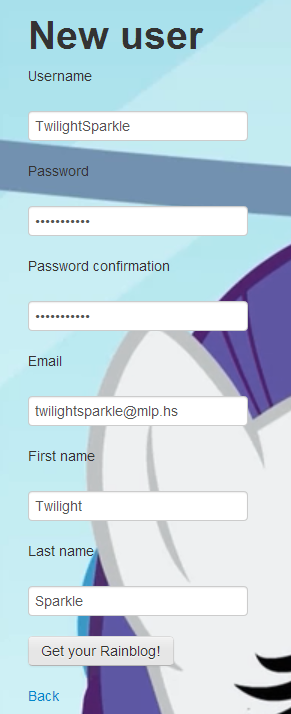
#### Hvis ikke logget inn, link til innlogging av bruker

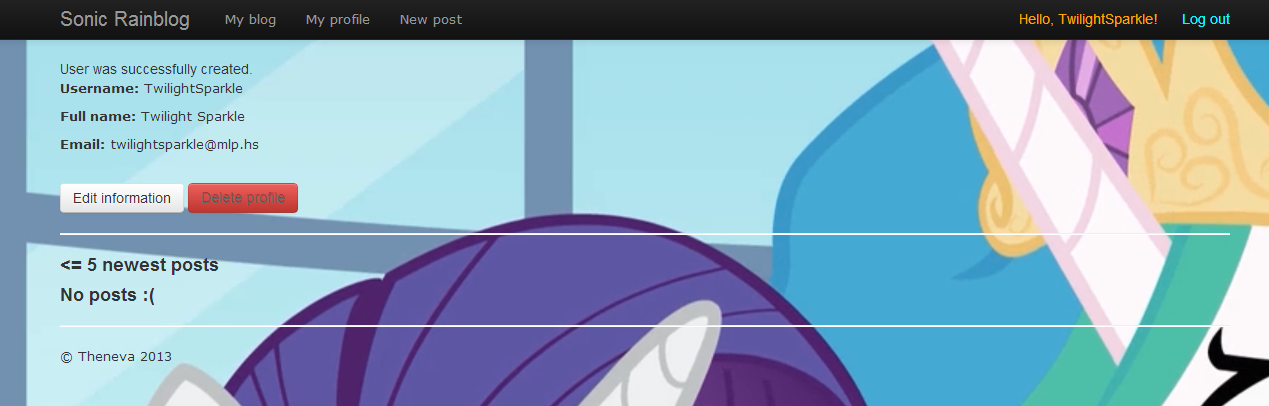


Se link til registrering av ny bruker; samme tankegang gjelder.

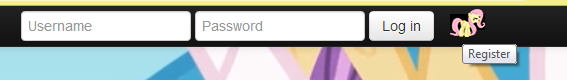
#### Brukergrensesnitt for registrering av nye brukere





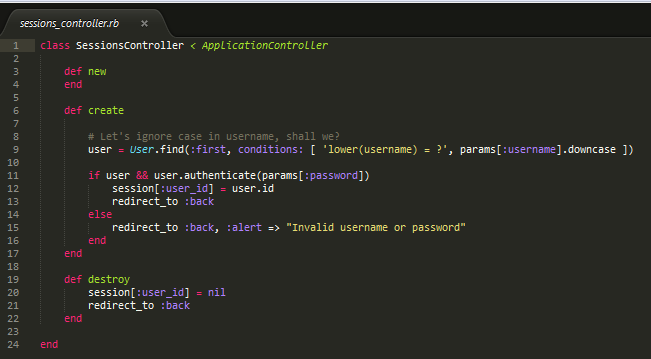


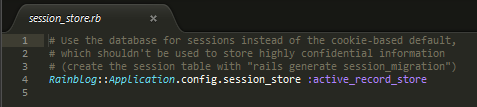
#### Brukergrensesnitt for innlogging av brukere (i /app/views/layouts/application.html.erb)



Applikasjonen benytter ActiveModel::SecurePassword og metoden has\_secure\_password, så passordet i klartekst blir aldri sett (bortsett fra når det sendes til server ved registrering/innlogging; applikasjonen kjører ikke på HTTPS).

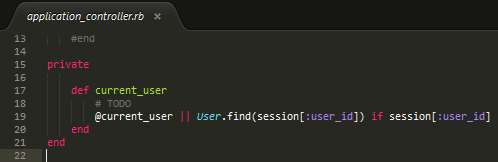


Sesjoner håndteres med en SessionsController og :active\_record\_store for lagring av sesjonsdata. 



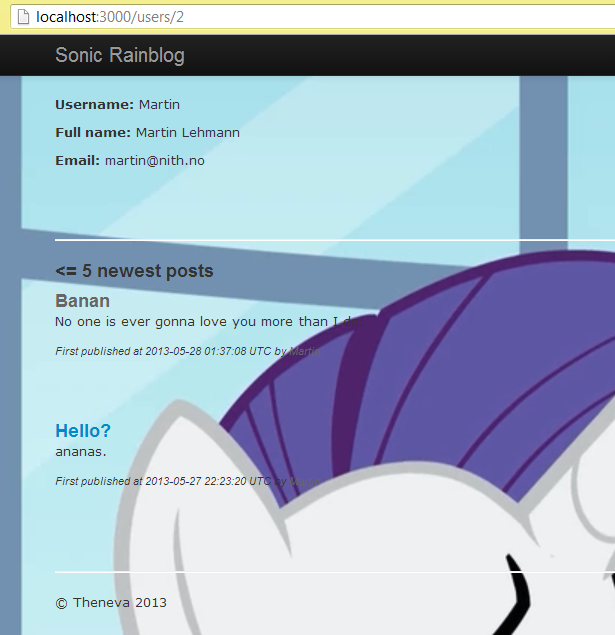
Den innloggede brukeren kan hentes ut med hjelpemetoden current\_user, som konvensjon tilsier.





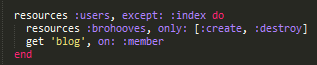
#### Profilside for brukere

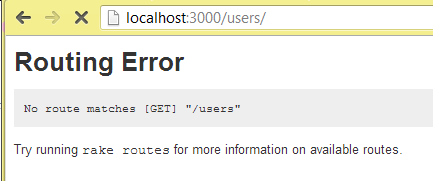
Profilsiden til brukeren Martin, med id = 2:



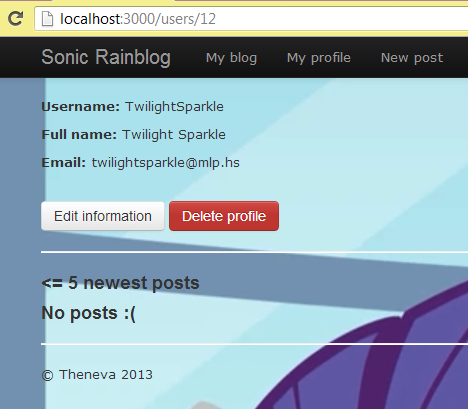
#### Ikke opprettes liste over alle brukere

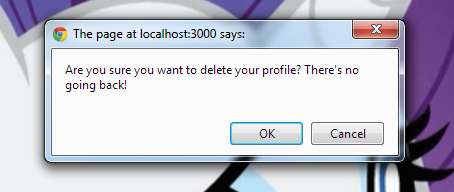
Nope. Har også unngått å la bruker søke etter <emptystring> i oppgave 7.





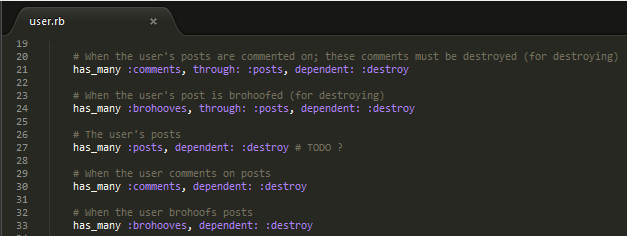
#### Brukere skal kunne slette sin egen brukerkonto. Alle brukerens blogginnlegg skal gå ned med samme skip.



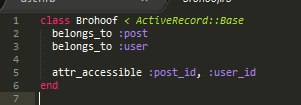


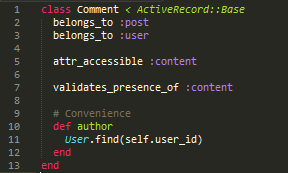


Alle poster (med tilhørende kommentarer og brohooves) tas ned sammen med brukeren ved hjelp av associations. Databasen inneholder naturligvis de nødvendige feltene (gjennom typen references i migrasjoner).





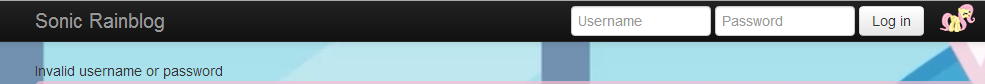




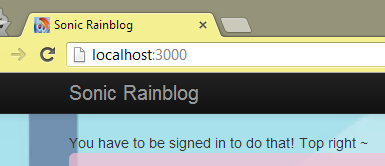
### Oppgave 2: Meldinger til brukeren

Brukeren får i alle tilfeller der noe kan gå galt melding om hva som eventuelt gikk galt i én eller annen form.

Logge inn med feil brukernavn/passord.



Forsøke å besøke eksempelvis <server>/posts, som har begrenset tilgang gjennom bruk av et before-filter.



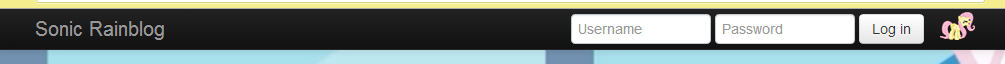
Forsøke å poste en kommentar uten innhold. Her burde errors-hashen vært benyttet, men jeg endte med å måtte løse dette ved en alert som fanges opp av og vises frem som en flash.



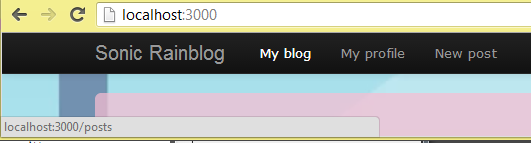
### Oppgave 3: Blogginnlegg

#### Dersom man er innlogget, skal det på en hver side vises en link til brukerens blogg (Posts#index)

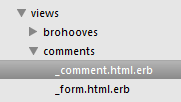
Linken vises ikke hvis man ikke er innlogget.



Hvis man *er* logget inn, derimot, vises alltid «My blog» som en lenke definert i /app/views/layouts/application.html.erb, som fører til at denne vises på alle undersider.

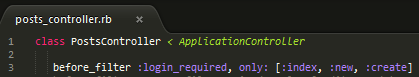


Bloggen bruker partial-en /app/views/posts/\_list.html.erb. Her kunne jeg for eksempel benyttet hjelpemetoden <%= render current\_user.posts %> med tilhørende /app/views/posts/\_post.html.erb, men partial-en benyttes av andre views og ha samme sjekk av hvorvidt lista er tom, så jeg har valgt å ikke løse det på denne måten. Jeg har dog valgt å liste kommentarer på denne måten (i bloggpostens show-view).

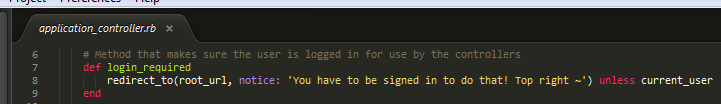


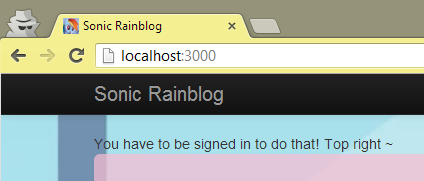


Hvis en ikke innlogget bruker forsøker å gå til /posts (Post#index), vil et before\_filter i PostsController sørge for at ha/un redirectes til root og få beskjed om å logge in. Denne beskjeden kunne med fordel vært stilsatt og plassert annerledes, men oppgaven stiller ikke eksplisitt krav til dette.



(login\_required er definert i ApplicationController.)



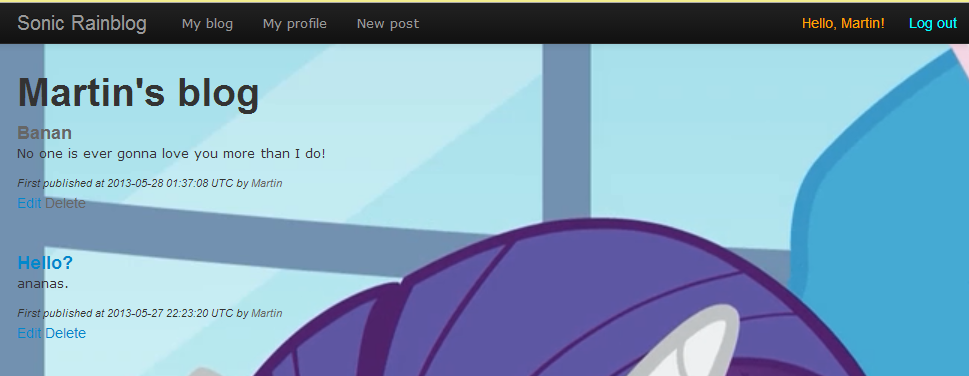


#### Blogginnlegg skal minimum ha attributtene tittel, innhold, publiseringstid, og –dato

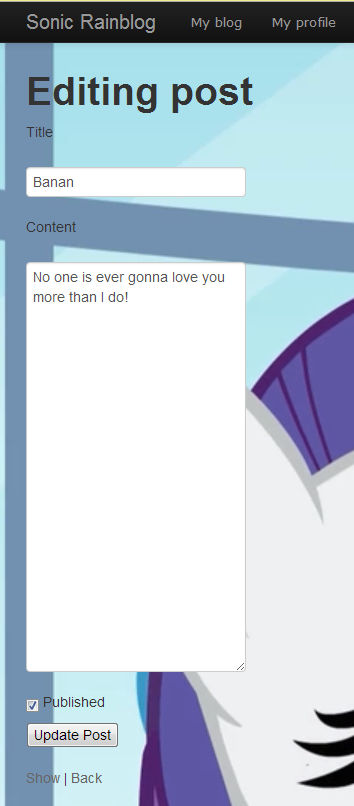


#### Det skal kunne opprettes, endres, og slettes blogginnlegg fra et grafisk grensesnitt i systemet

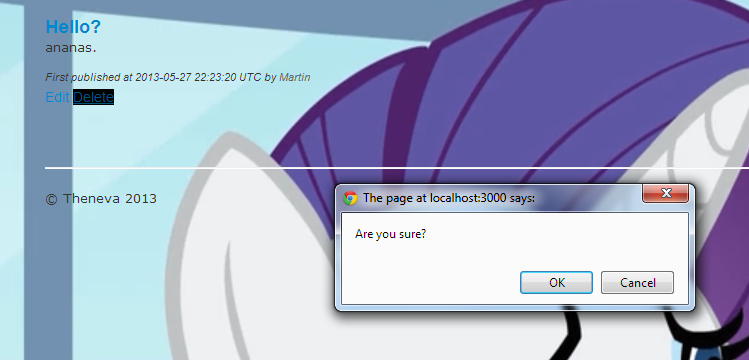
Jeg har her tatt høyde for at brukere kan publisere innlegg flere ganger. published\_at-kolonnen til Post blir oppdatert uavhengig av om posten var publisert fra før hvis den endres, men oppgaveteksten kan tolkes til at du publiserer innlegget på nytt hvis du oppdaterer det. Kolonnen kunne kanskje med fordel hett noe sånt som «last\_published\_at».

Her ligger Edit og Delete tilgjengelig, samt «New post» som lenke i headeren som finnes i /app/views/layouts/application.html.erb.

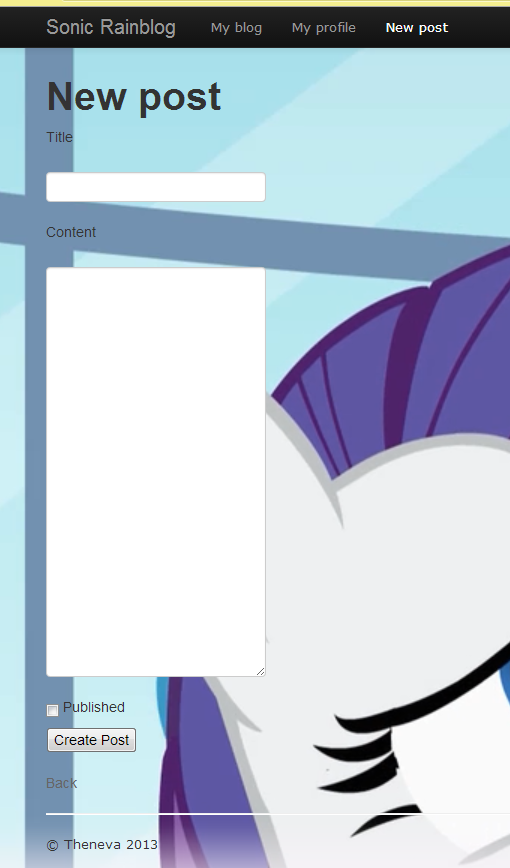
Grensesnitt for å endre post



Grensesnitt for å slette post



Grensesnitt for å opprette ny post

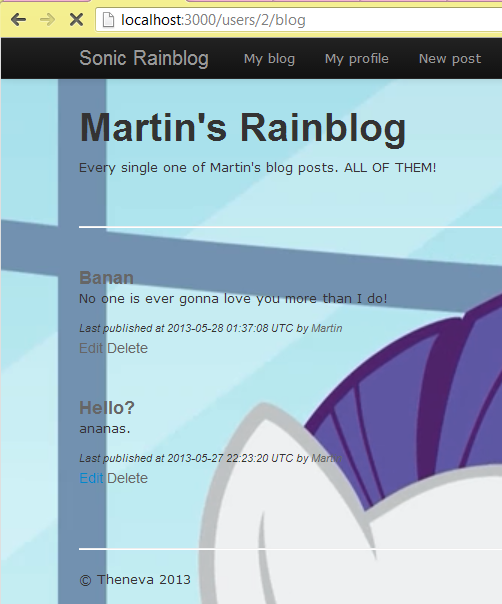


#### Oversiktsside som viser alle blogginnlegg for en blogg, sortert etter publiseringstidspunkt (sist publisert)

Jeg har valgt å matche url-en <server>/users/:id/blog som en non-resourceful route nøstet i users-ressursen for å liste opp alle blogginnleggene til hver enkelt bruker.



Denne bloggen har to poster. Innleggene vises sortert på publiseringstidspunkt, så nyest publiserte poster kommer øverst.

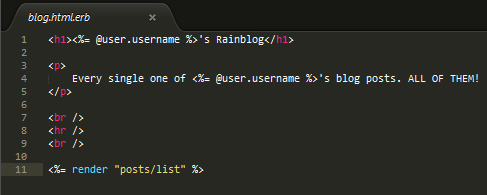


#### Ingen publiserte innlegg skal forklares, med mindre eieren av bloggen ser på sin egen blogg; da skal ikke-publiserte innlegg vises også.

Jeg henter ut alle tilgjengelige innlegg for brukeren som er logget inn, og setter instansvariabelen @posts lik lista som returneres. Denne skrives ut i view-et.

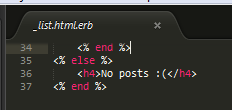


User#blog gjør ikke annet enn å rendre posts/list

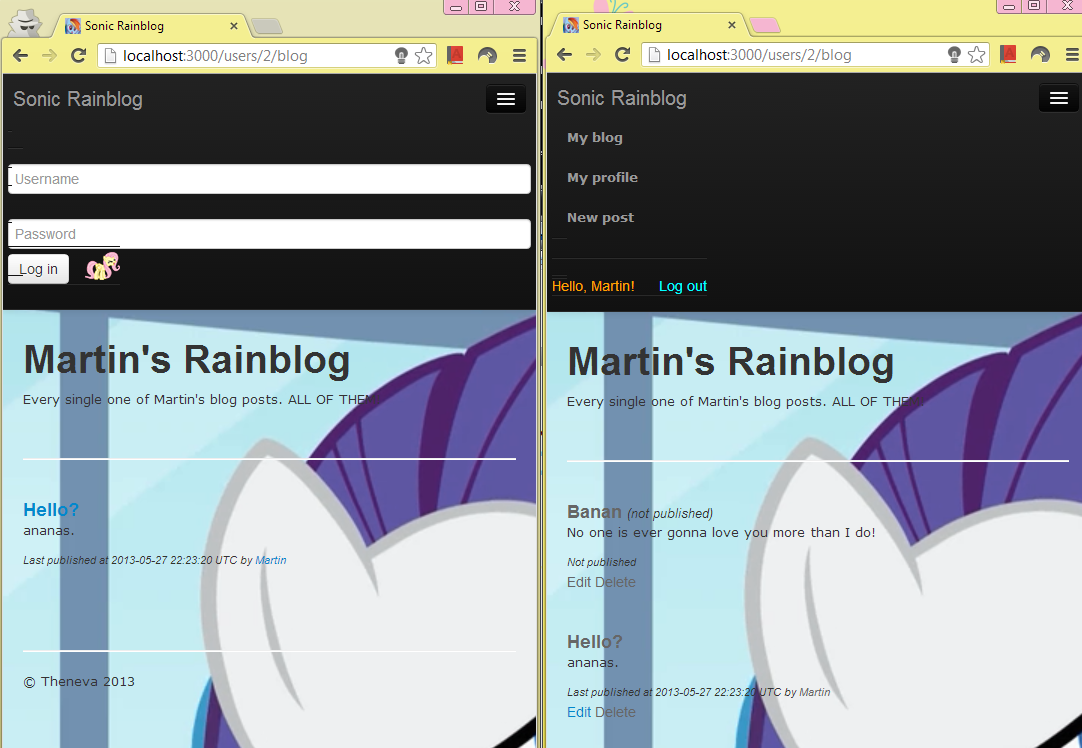


… og posts/list-partial-en sørger for å gi en oppklarende beskjed hvis ingen innlegg matcher kriteriene, eller skrive ut alle innleggene som finnes i lista.



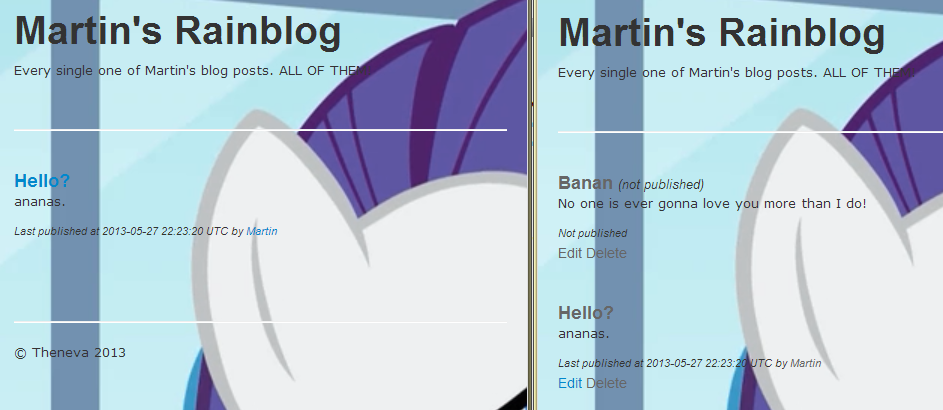


Blogginnlegg som ikke er publiserte vises bare frem hvis den innloggede brukeren er eier av bloggen, og ikke-publiserte innlegg får «*(not published)*» lagt til ved siden av tittel. På grunn av oppsettet der jeg henter ut tilgjengelige innlegg i Controlleren, vil partial-en sørge for at det vises en beskjed dersom det ikke er noen tilgjengelige innlegg uavhengig av hvorvidt eieren av bloggen er logget inn.



#### Kun eieren av bloggen skal ha lov til å opprette, endre, og slette blogginnlegg

Fordi Post#new lager en ny post for brukeren, og verken Edit eller Delete benytter get-forespørsler, lar jeg view-et alene avgjøre hvorvidt den innloggede brukeren skal få tilgang til å endre/slette innleggene.



#### Kun eieren av bloggen skal ha lov til å vise innlegg i sine private blogger

Går ut fra at dette ikke er et krav til oppgaven, da «private blogger» ikke omtales noe annet sted i oppgaveteksten.

#### Kun eieren av bloggen skal ha lov til å vise innlegg som enda ikek er publiserte

Dette er allerede dekket; den samme view partial-en benyttes til å vise frem poster gjennom hele applikasjonen, så dette vil alltid fungere som allerede dokumentert.

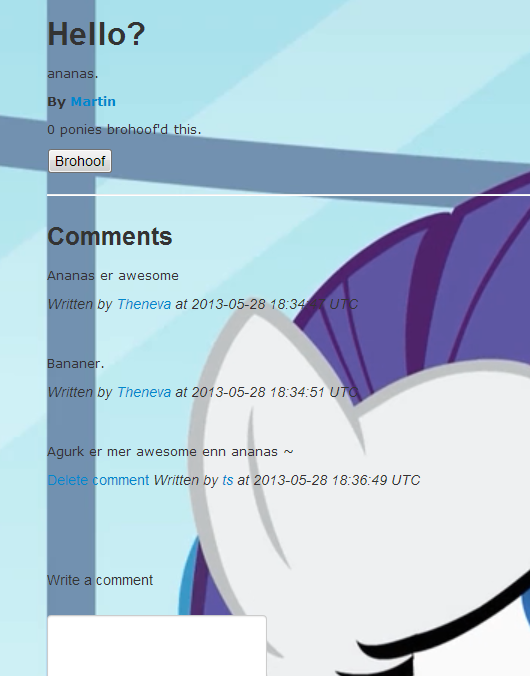
### Oppgave 4: Kommentarer

#### Kommentarer skal minst inneholde innhold



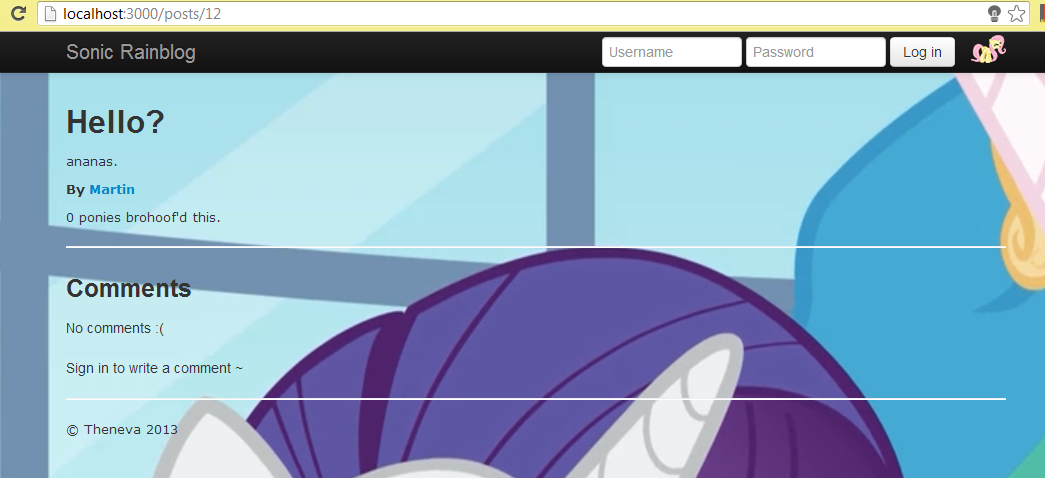
#### Det skal vises en oversikt over alle kommentarer som tilhører et innlegg på samme siden som innlegget vises (show-view)

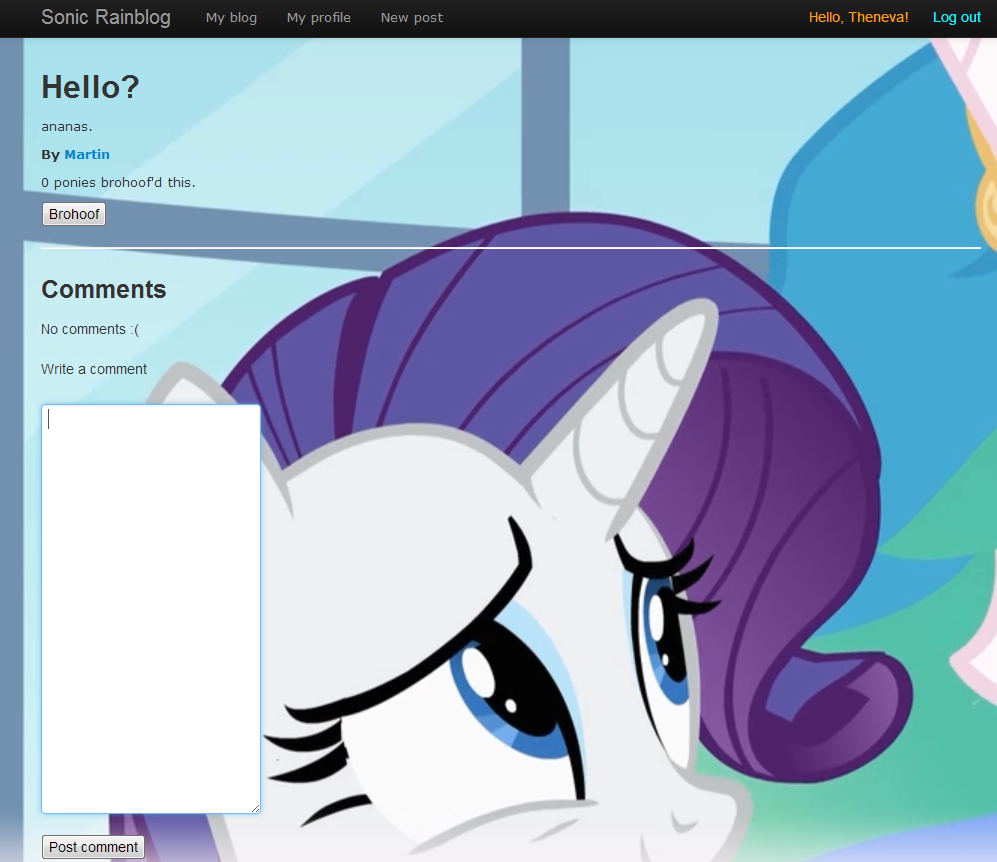
Her ser man at det står oppført sammen med kommentaren hvem som har kommentert, og når kommentaren ble opprettet.



#### Dersom innlegget ikke har noen kommentarer, skal dette vises med en forklarende tekst & kun innloggede brukere skal ha lov til å kommentere på innlegg

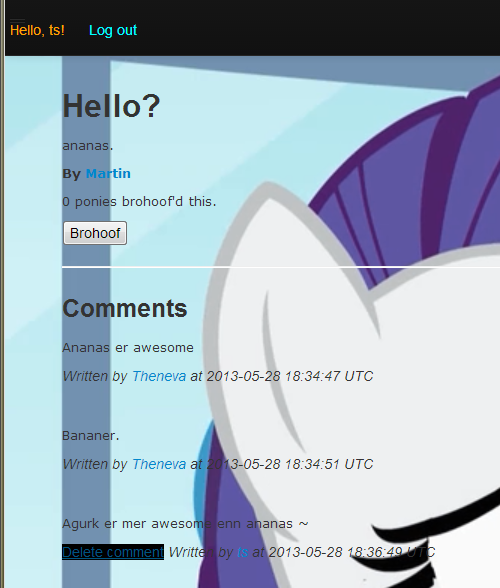
Det finnes ikke noen *new*-action for kommentarer, så jeg lar view-et avgjøre om form-et skal vises.

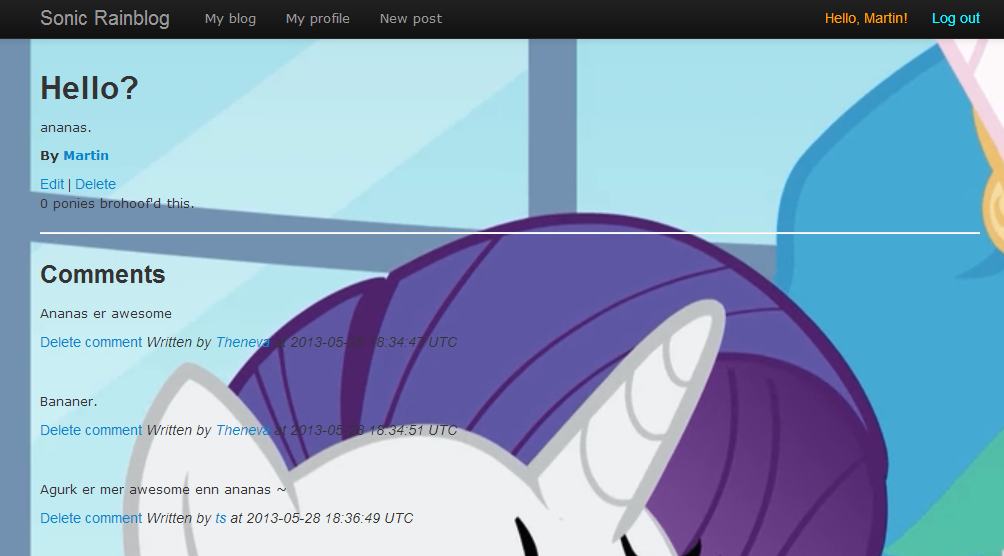




#### Det skal være mulig å slette sine egne kommentarer

Både forfatter av kommentaren og eier av blogginnlegget har mulighet til å slette kommentarer.





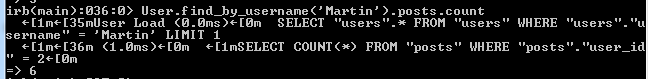
### Oppgave 5: Profilsiden

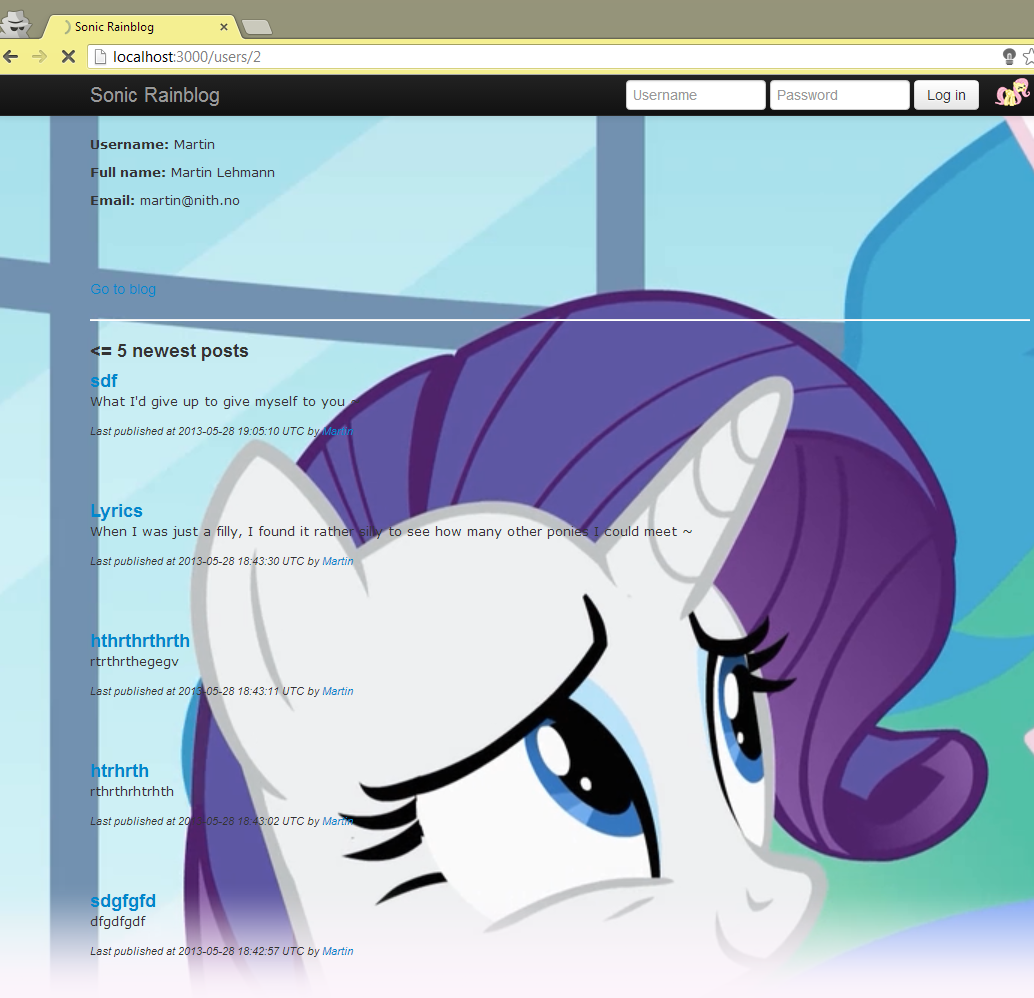
#### Profilsiden skal vise brukernavn, fullt navn, vise brukerens 5 siste publiserte innlegg (sortert på publiseringstidspunkt), og ingen innlegg i bloggen skal forklares med tekst; innleggene skal listes opp på samme måte som i oversiktssiden; innlogget bruker som ser på egen profil skal ha mulighet til å endre og slette profilen sin

Denne bloggen har bare én post, som er grunnen til at kun én vises. Det er mulig å endre/slette profil fordi brukeren ser på sin egen blogg.

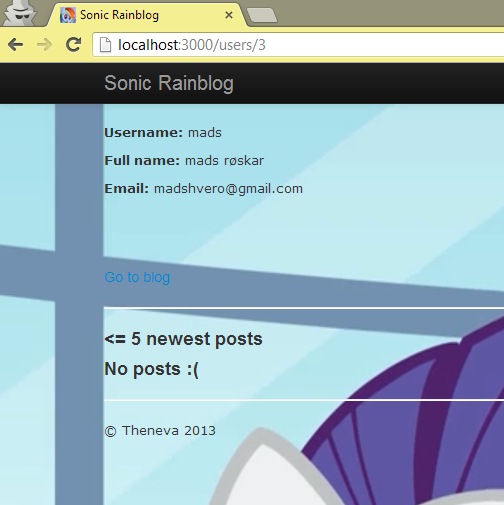
#### 

Her ser en ikke innlogget bruker på bloggen. En kan se av konsollen at brukeren Martin har 6 poster, men bare 5 av dem vises i «recent posts»-delen av profilen.



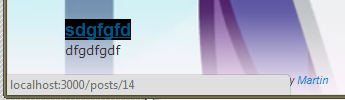


Her er en blogg uten noen poster. Fordi view partial-en benyttes, trenger man ikke mer kode enn det som allerede er dokumentert.



#### Det skal være mulig å besøke hvert enkelt innlegg fra denne oversikten

Yep.

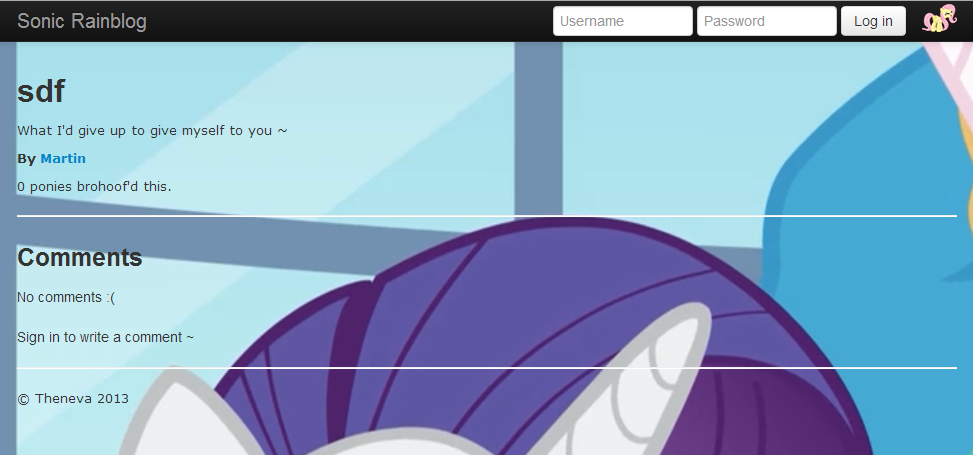


### Oppgave 6: Fist bumps

Nope; brohooves (som ga meg en unnskyldning for å leke med Inflector-klassen fordi pluralizeren hardnakket hevder av «brohoof» blir til «brohoofs» i flertall).

#### Innloggede brukere skal kunne gi fist bumps til andre brukeres blogginnlegg; eier av et innlegg skal ikke kunne fist bumpe sitt eget innlegg; det skal være mulig å trekke tilbake fist bumps; hver bruker kan gi én fist bump per innlegg; en brukers favoritter skal kun kunne fjernes av brukeren selv

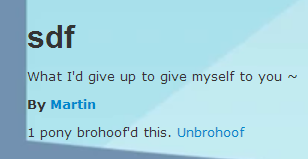
Ikke innlogget: får se antall brohooves, men ikke brohoofe innlegget selv.



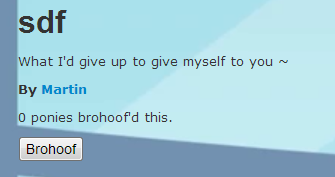
Innlogget på en annen profil enn eieren av innlegget; kan brohoofe innlegget



\*Brohoof\*



\*Unbrohoof\*

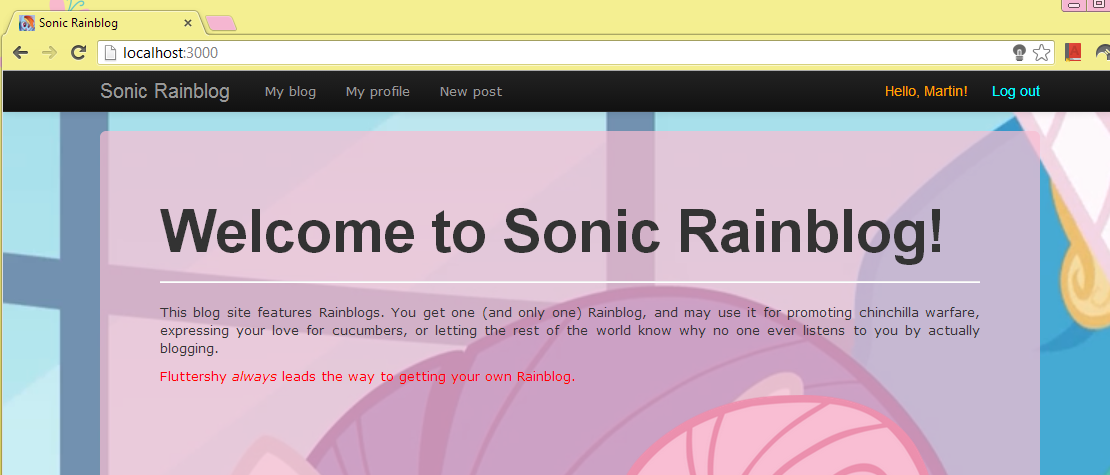


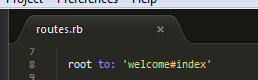
Det tilbys ikke noe grensesnitt for oversikt over hvem som har brohoofet innlegget, og det er dermed heller ikke mulighet for å endre andres favoritter; Brohoof har kun create og destroy som actions.

View-et tar for seg at det ikke er mulig å gi brohoof til egne innlegg ved å sjekke om innlogget bruker er postens eier.

### Oppgave 7: Forside med brukersøk

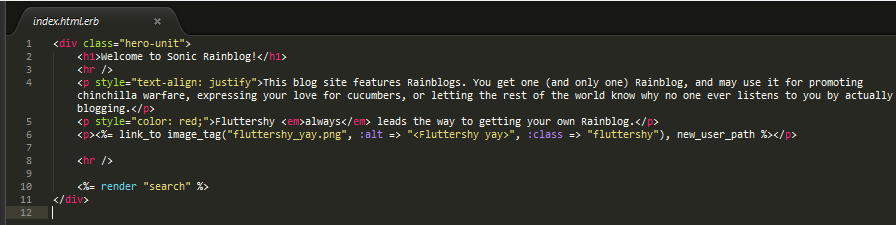
#### Forside tilgjengelig på rot







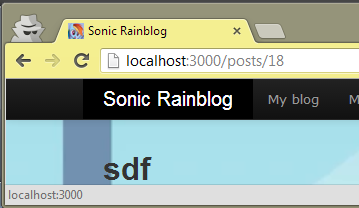
Welcome#index:



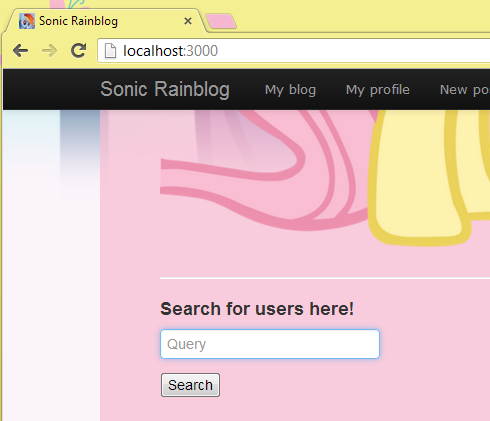
Det finnes så vidt jeg har klart å finne ikke noen konvensjon for hva en «landing page»/forside-resource skal kalles, så jeg gikk med Welcome fordi dette er eksempelverdien i routes-fila.

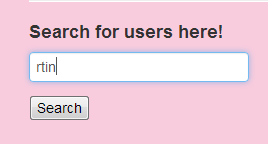
#### Det skal på en hver side vises en link til forsiden

Dette håndterer /app/views/layouts/application.html.erb.

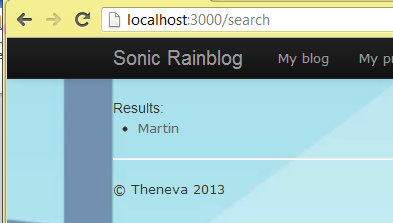


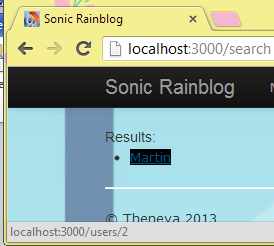
#### Forsiden skal inneholde et søkefelt, og ved å bruke dette skal en kunne søke på brukernavn; søket skal returnere alle brukernavn som inneholder søkestrengen, ved å bruke SQL-kodeordet LIKE; brukerne som blir funnet i søket skal presenteres på resultatsiden; det skal være mulig å klikke på brukernavnet til en bruker for å besøke brukerens profil



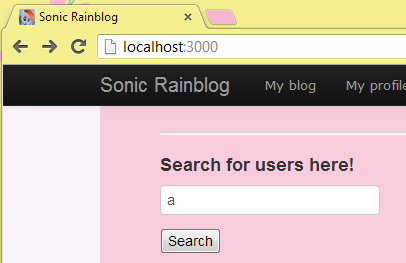


\*Search\*





Eksempel 2:



\*Search\*

