



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.

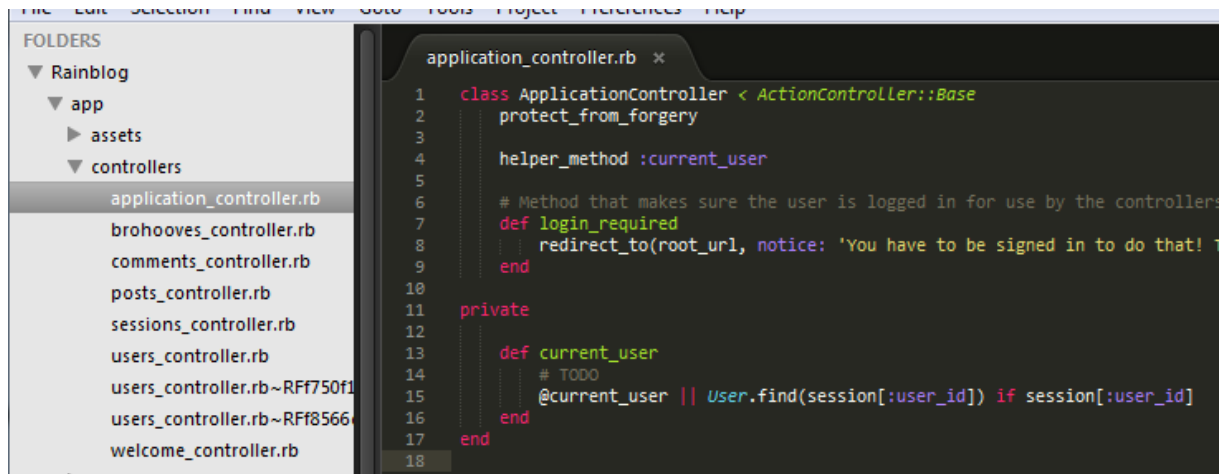
5/28/2013

2. Vurderingskriterier

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

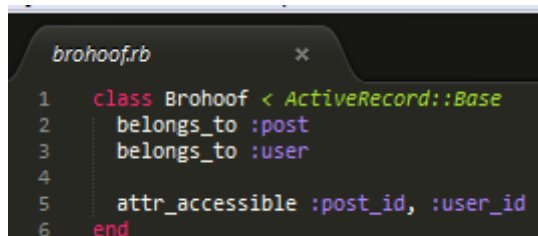
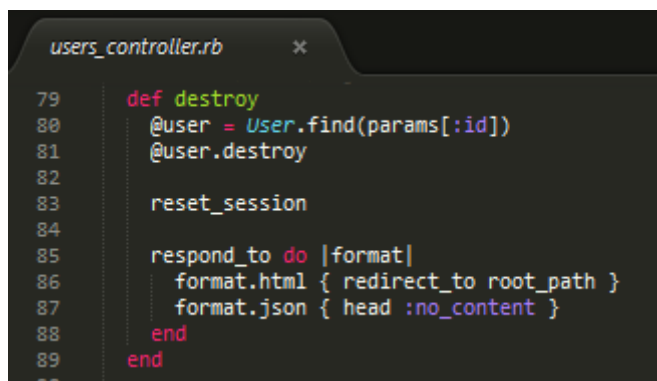
Koden følger standard for både innrykk, formatering, og navngiving i henhold til språk og rammeverk. Det finnes ikke noen klar konvensjon for innrykk for access modifiers (<http://fabiochung.com/2010/04/05/ruby-indentation-for-access-modifiers-and-their-sections/>).

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.



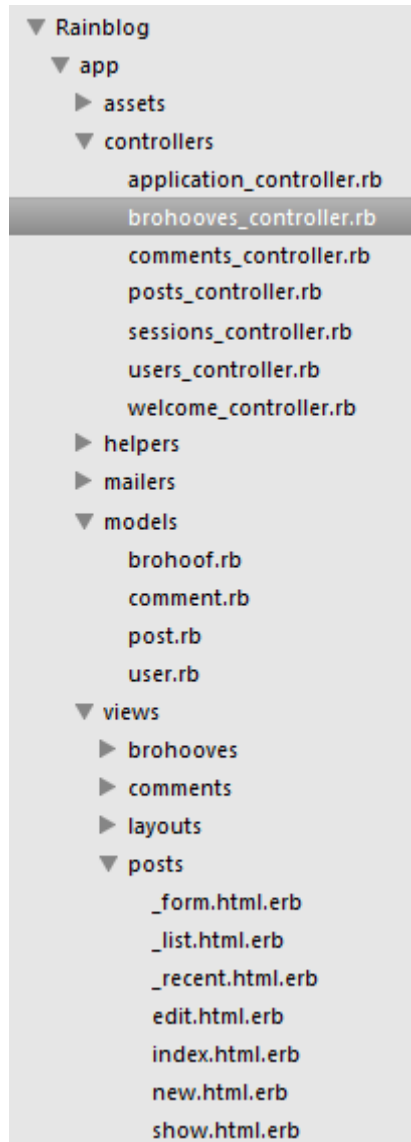
Idiomatisk kode og smarte løsninger

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.



```
20130525225619_create_brohooves.rb x
1  class CreateBrohooves < ActiveRecord::Migration
2    def change
3      create_table :brohooves do |t|
4        t.references :post
5        t.references :user
6
7        t.timestamps
8      end
9      add_index :brohooves, :post_id
10     add_index :brohooves, :user_id
11   end
12 end
```

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.

```

1 class User < ActiveRecord::Base
2
3   # Rails 3.1 <3
4   has_secure_password
5
6   # Don't want no trouble with mass assignment; whitelist
7   attr_accessible :username, :password, :password_confirmation, :email, :first_name, :last_name
8
9   # Validation; everything present, some other requirements
10  |   # Regex borrowed without asking from http://www.regular-expressions.info/email.html
11  email_regex = /[a-z0-9!#$%&'*+\/=?^_`{|}~]+(?:\.[a-z0-9!#$%&'*+\/=?^_`{|}~]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?$/i
12
13  validates :username, presence: true, uniqueness: { case_sensitive: false }, length: { in: 1..25 }
14  validates_length_of :password, minimum: 5
15  validates :email, uniqueness: true, format: { with: email_regex }
16  validates_presence_of :email, :first_name, :last_name
17
18  # Order is key here; destroying the posts before finding comments and brohooves => disaster
19
20  # When the user's posts are commented on; these comments must be destroyed (for destroying)
21  has_many :comments, through: :posts, dependent: :destroy
22
23  # When the user's post is brohoofed (for destroying)
24  has_many :brohooves, through: :posts, dependent: :destroy
25
26  # The user's posts
27  has_many :posts, dependent: :destroy # TODO ?
28
29  # When the user comments on posts
30  has_many :comments, dependent: :destroy
31
32  # When the user brohoofs posts
33  has_many :brohooves, dependent: :destroy
34
35  # Convenience methods
36  def get_brohoof_array_by_post post
37    |   self.brohooves.map(&:id) & post.brohooves.map(&:id)
38  end
39
40  # Only to be used when we're absolutely sure the post has a brohoof
41  def get_id_of_brohoof_in_post post
42    |   get_brohoof_array_by_post(post).first
43  end
44
45  # HAX :D
46  def has_brohoofed_post post
47    |   get_brohoof_array_by_post(post).any?
48  end
49 end

```

```

blog.html.erb
1 <h1><%= @user.username %>'s Rainblog</h1>
2
3 <p>
4   |   Every single one of <%= @user.username %>'s blog posts. ALL OF THEM!
5 </p>
6
7 <br />
8 <hr />
9 <br />
10
11 <%= render "posts/list" %>

```

```

_list.html.erb
1 <% if @user.posts.any? %>
2   <% @user.posts.each do |post| %>
3     <div class="post">
4
5       <h4 style="display: inline;">
6         <%= link_to post.title, post_path(post) %>
7       </h4>
8       <% unless post.published %>
9         <span style="font-size: 0.9em; font-style: italic; font-color: orange;">(not published)</span>
10      <% end %>
11
12      <p>
13        <%= post.content %>
14      </p>
15
16      <span style="font-color: grey; font-style: italic; font-size: 0.8em;">
17        <% if post.published %>
18          First published at <%= post.published_at %> by <%= link_to @user.username, user_path(@user) %>
19        <% else %>
20          Not published
21        <% end %>
22      </span>
23
24      <br />
25
26      <% if current_user == post.user_id %>
27        <%= link_to 'Edit', edit_post_path(post) %> <%= link_to 'Delete', post, method: :delete, data: { confirm: 'Are you sure?' } %>
28      <% end %>
29
30      <br />
31      <br />
32      <br />
33    </div>
34  <% end %>
35 <% else %>
36   <h4>No posts :( </h4>
37 <% end %>

```

```

users_controller.rb
1 class UsersController < ApplicationController
2
3   # The number of posts to display as "Recent"
4   RECENT_LIMIT = 5

```

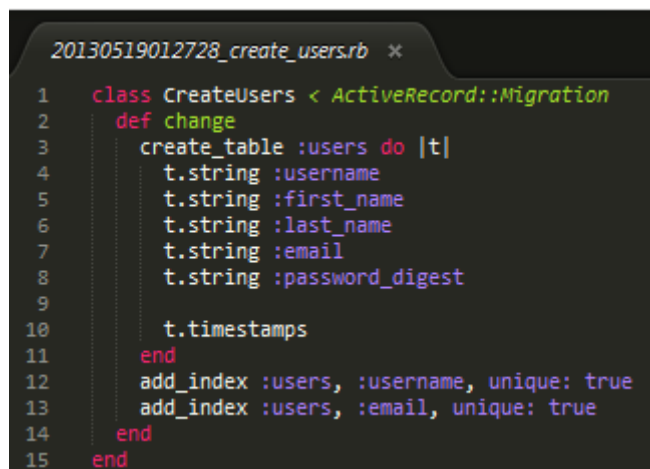
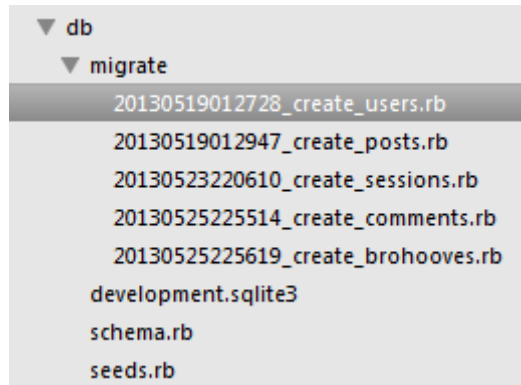
```

users_controller.rb
91 # GET /users/1/blog
92 # GET /users/1/blog.json
93 def blog
94   @user = User.find(params[:id])
95   @posts = Post.available_by_user(@user.id, current_user.nil? ? -1 : current_user.id).limit(RECENT_LIMIT)
96
97   respond_to do |format|
98     format.html # favorites.html.erb
99     format.json { head :no_content }
100   end
101 end

```

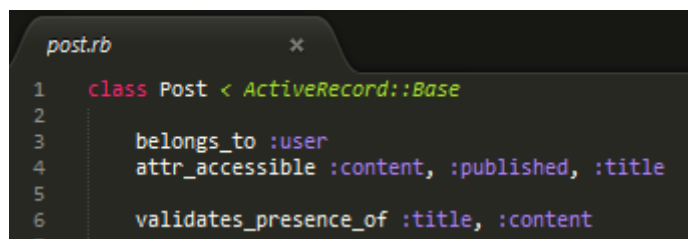
Migrations brukes for endringer mot databasestruktur

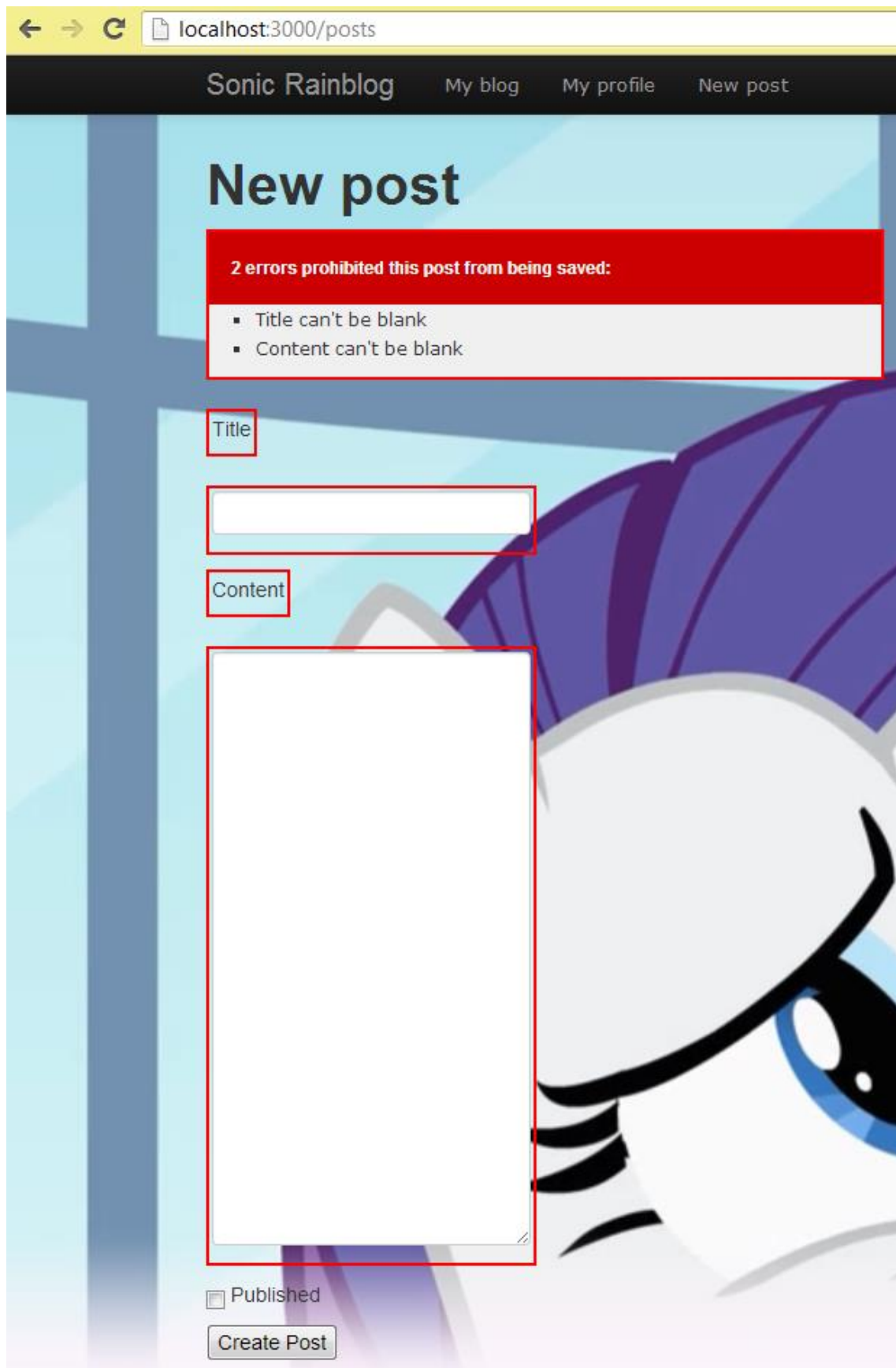
Yup.



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

Benytter Rails' standardmetoder for validering.





The screenshot shows a web browser window with the address bar displaying 'localhost:3000/posts'. The page has a dark header with the text 'Sonic Rainblog' and navigation links 'My blog', 'My profile', and 'New post'. The main content area is titled 'New post' and features a red error box with the message '2 errors prohibited this post from being saved:'. Below this, a list of errors is shown: 'Title can't be blank' and 'Content can't be blank'. The form includes a 'Title' label and an empty text input field, and a 'Content' label and a large empty text area. At the bottom, there is a 'Published' checkbox and a 'Create Post' button. The background of the page features a stylized illustration of Sonic the Hedgehog.

localhost:3000/posts

Sonic Rainblog My blog My profile New post

New post

2 errors prohibited this post from being saved:

- Title can't be blank
- Content can't be blank

Title

Content

☐ Published

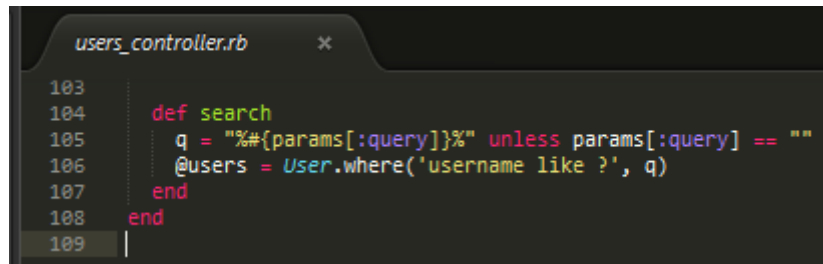
Create Post

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.

```
scope :available, lambda { |current_user_id| where('published = ? OR user_id = ?', true, current_user_id).order 'published_at'  
scope :available_by_user, lambda { |user_id, current_user_id| available(current_user_id).where('user_id = ?', user_id) }
```



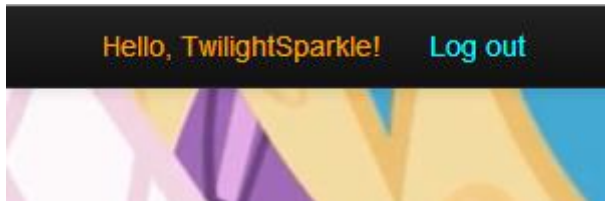
```
users_controller.rb  x  
  
103  
104   def search  
105     q = "%#{params[:query]}%" unless params[:query] == ""  
106     @users = User.where('username like ?', q)  
107   end  
108 end  
109 |
```


4. Oppgavebeskrivelse

Oppgave 1: Brukerregistrering og autentisering

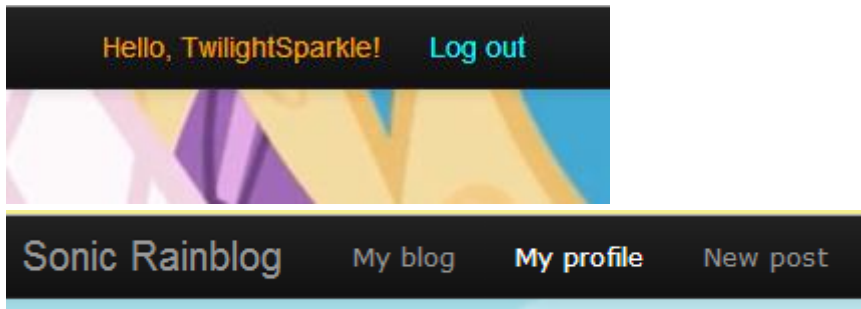
Hvis logget inn, link til utlogging

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



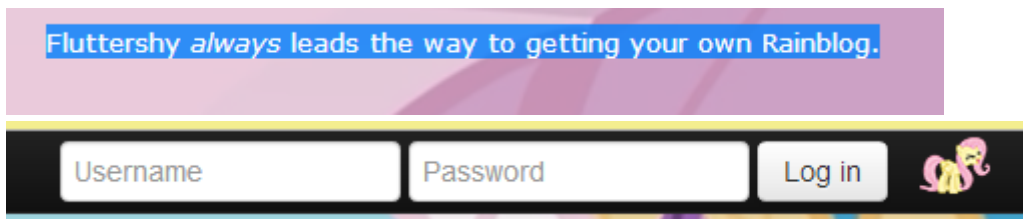
Hvis logget inn, link til profilsiden for innlogget bruker

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

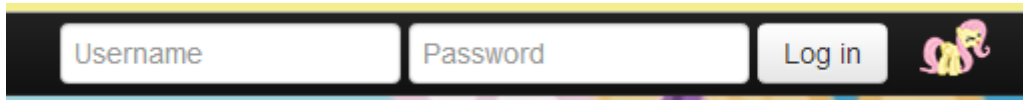


Hvis ikke logget inn, link til registrering av ny bruker

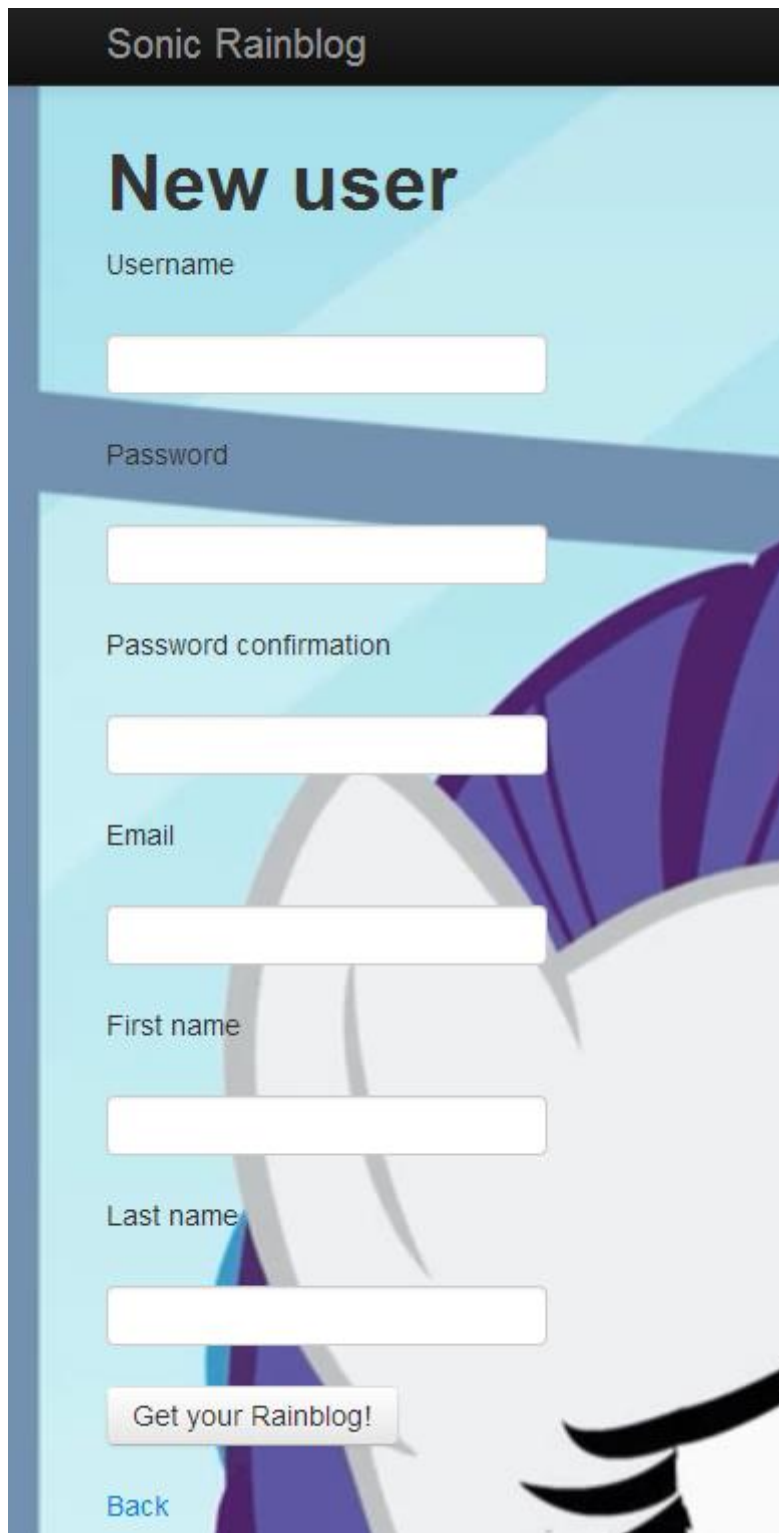
Fluttershy er å finne oppe til høyre (i /app/views/layouts/application.html.erb), og er tilgjengelig fra alle undersider.



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

The image shows a web form for creating a new user account on 'Sonic Rainblog'. The form is set against a background featuring a stylized illustration of Sonic the Hedgehog. The form fields are arranged vertically, each with a label to its left. The labels are: 'Username', 'Password', 'Password confirmation', 'Email', 'First name', and 'Last name'. Each label is followed by a white rectangular input field. At the bottom of the form, there is a button labeled 'Get your Rainblog!' and a link labeled 'Back'.

Sonic Rainblog

New user

Username

Password

Password confirmation

Email

First name

Last name

Get your Rainblog!

[Back](#)

New user

Username

Password

Password confirmation

Email

First name

Last name

Get your Rainblog!

[Back](#)

Sonic Rainblog My blog My profile New post Hello, TwilightSparkle! [Log out](#)

User was successfully created
Username: TwilightSparkle
Full name: Twilight Sparkle
Email: twilightsparkle@mlp.hs

[Edit information](#) [Delete profile](#)

<= 5 newest posts
No posts :(

© Theneva 2013

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

Applikasjonen benytter ActiveSupport::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).

```
irb(main):003:0> User.find_by_username('TwilightSparkle')
=> #<User id: 12, username: "TwilightSparkle", first_name: "Twilight", last_name: "Sparkle", email: "twilightsparkle@mlp.hs", password_digest: "$2a$10$C5G.83REhrgnaDimfd1400NbMEh5LbxxnUL7s2npm/4...", created_at: "2013-05-28 16:45:19", updated_at: "2013-05-28 16:45:19">
irb(main):004:0> User.find_by_username('TwilightSparkle').password_digest
=> "$2a$10$C5G.83REhrgnaDimfd1400NbMEh5LbxxnUL7s2npm/44jABZh1UK"
irb(main):005:0>
```

Sesjoner håndteres med en SessionsController og :active_record_store for lagring av sesjonsdata.

```
sessions_controller.rb
1 class SessionsController < ApplicationController
2
3   def new
4   end
5
6   def create
7
8     # Let's ignore case in username, shall we?
9     user = User.find(:first, conditions: [ 'lower(username) = ?', params[:username].downcase ])
10
11     if user && user.authenticate(params[:password])
12       session[:user_id] = user.id
13       redirect_to :back
14     else
15       redirect_to :back, :alert => "Invalid username or password"
16     end
17   end
18
19   def destroy
20     session[:user_id] = nil
21     redirect_to :back
22   end
23
24 end
```

```
session_store.rb
1 # Use the database for sessions instead of the cookie-based default,
2 # which shouldn't be used to store highly confidential information
3 # (create the session table with "rails generate session_migration")
4 Rainblog::Application.config.session_store :active_record_store
5
```

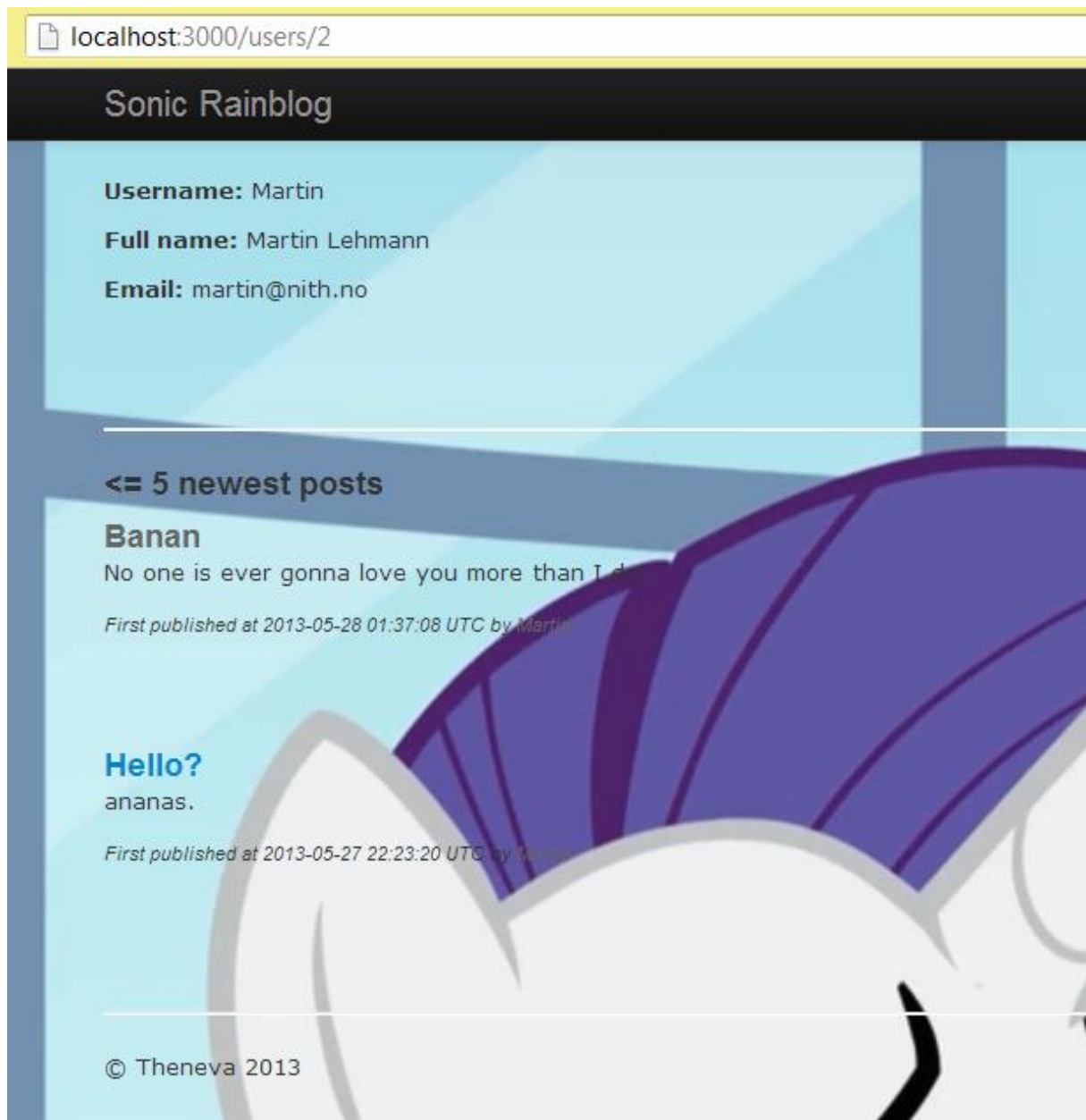
Den innloggede brukeren kan hentes ut med hjelpemetoden current_user, som konvensjon tilsier.

```
application_controller.rb ✕  
1  class ApplicationController < ActionController::Base  
2    protect_from_forgery  
3  
4    helper_method :current_user  
5
```

```
application_controller.rb ✕  
13  | #end  
14  
15  private  
16  
17    def current_user  
18      # TODO  
19      @current_user || User.find(session[:user_id]) if session[:user_id]  
20    end  
21  end  
22
```

Profilside for brukere

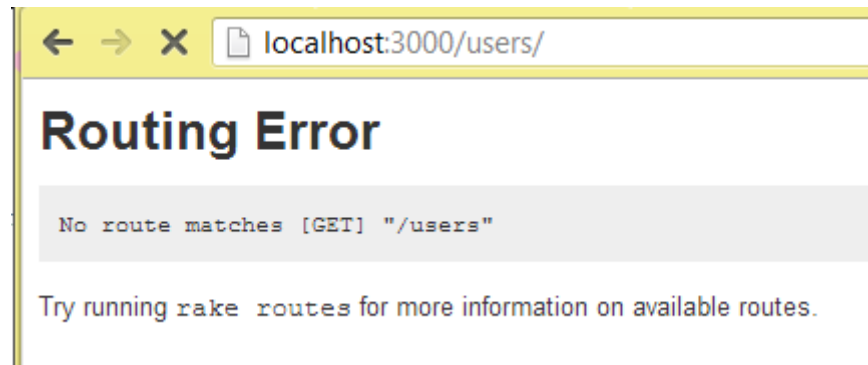
Profilsiden til brukeren Martin, med id = 2:



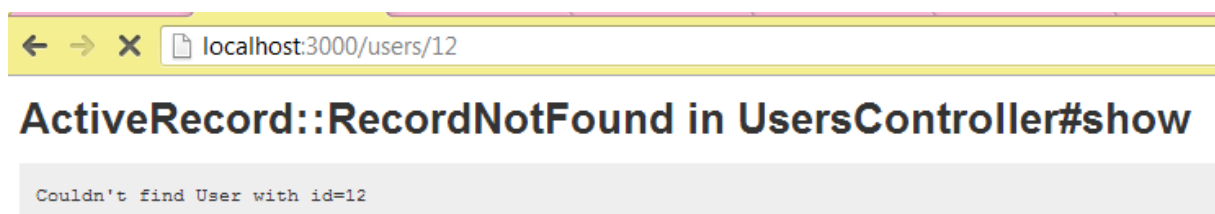
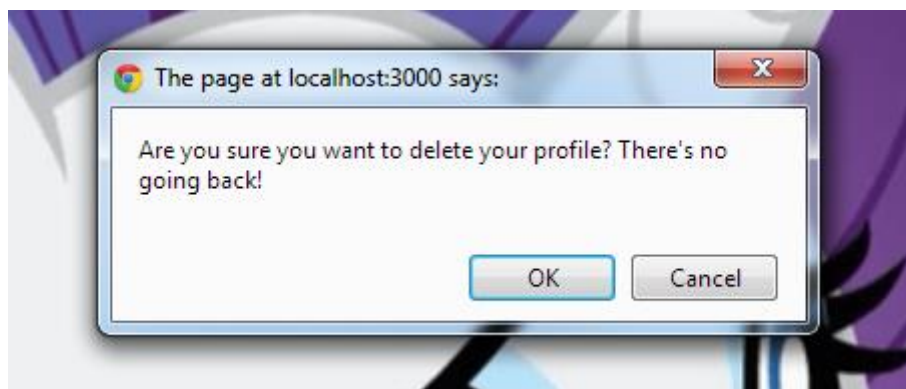
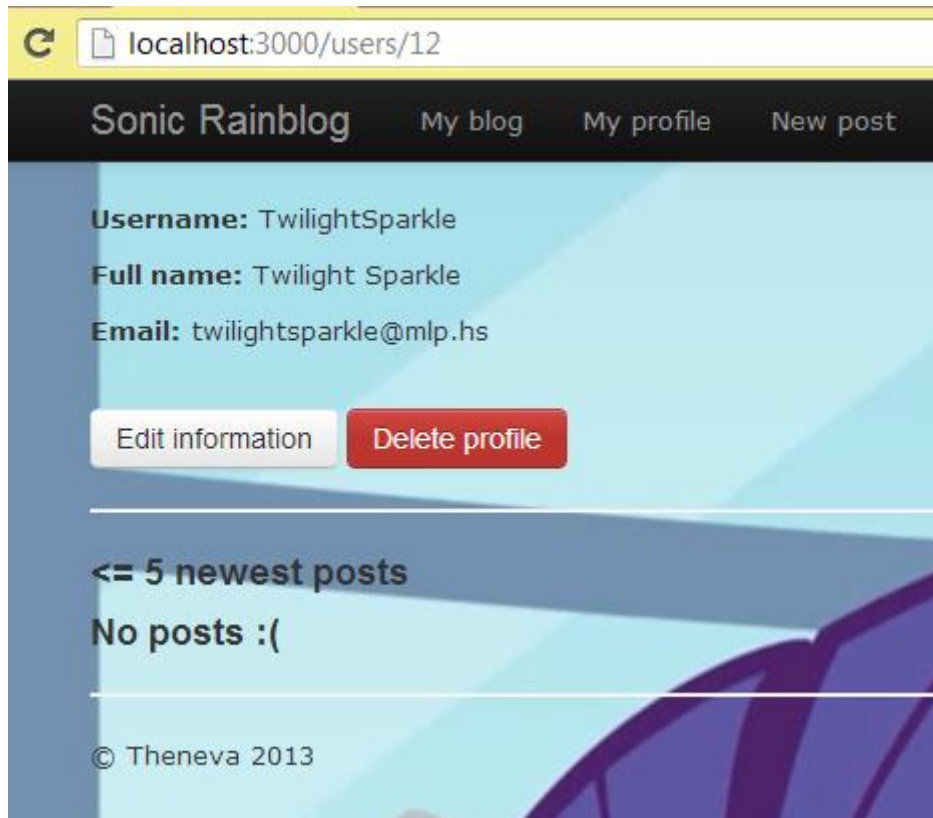
Ikke opprettes liste over alle brukere

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

```
resources :users, except: :index do
  resources :brohooves, only: [:create, :destroy]
  get 'blog', on: :member
end
```



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).

```
user.rb
19
20 # When the user's posts are commented on; these comments must be destroyed (for destroying)
21 has_many :comments, through: :posts, dependent: :destroy
22
23 # When the user's post is brohoofed (for destroying)
24 has_many :brohooves, through: :posts, dependent: :destroy
25
26 # The user's posts
27 has_many :posts, dependent: :destroy # TODO ?
28
29 # When the user comments on posts
30 has_many :comments, dependent: :destroy
31
32 # When the user brohoofs posts
33 has_many :brohooves, dependent: :destroy
```

```
1 class Post < ActiveRecord::Base
2
3   belongs_to :user
```

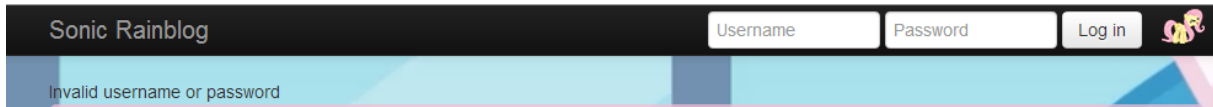
```
1 class Brohoof < ActiveRecord::Base
2   belongs_to :post
3   belongs_to :user
4
5   attr_accessible :post_id, :user_id
6 end
7
```

```
1 class Comment < ActiveRecord::Base
2   belongs_to :post
3   belongs_to :user
4
5   attr_accessible :content
6
7   validates_presence_of :content
8
9   # Convenience
10  def author
11    User.find(self.user_id)
12  end
13 end
```

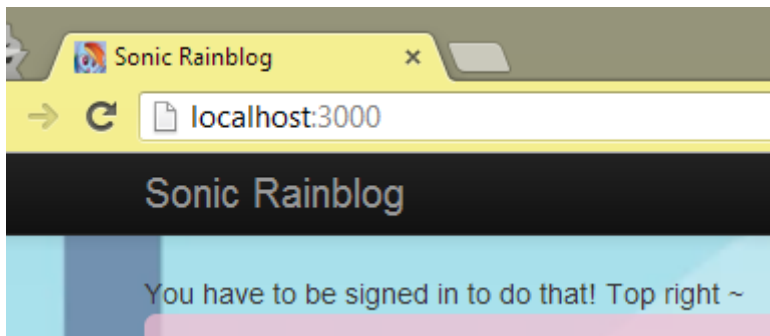
Oppgave 2: Meldinger til brukeren

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.



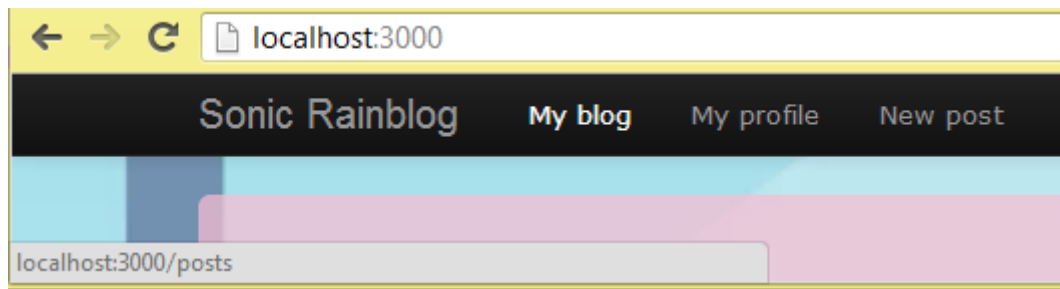
Oppgave 3: Blogginnlegg

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

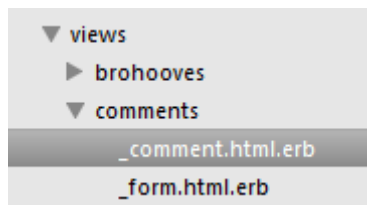
Linken vises ikke hvis man ikke er innlogget.



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).



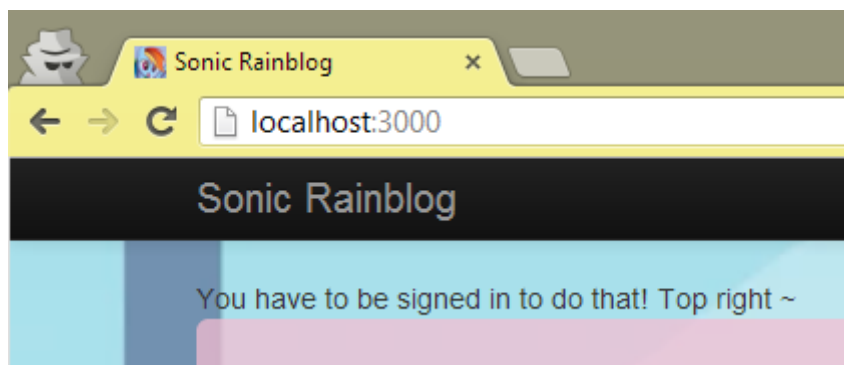


Hvis en ikke innlogget bruker forsøker å gå til /posts (Post#index), vil et `before_filter` i `PostsController` sørge for at ha/un redirectes til root og få beskjed om å logge in. Denne beskjeden kunne med fordel vært stilsatt og plassert annerledes, men oppgaven stiller ikke eksplisitt krav til dette.

```
posts_controller.rb  x
1  class PostsController < ApplicationController
2
3    before_filter :login_required, only: [:index, :new, :create]
```

(`login_required` er definert i `ApplicationController`.)

```
application_controller.rb  x
6
7  # Method that makes sure the user is logged in for use by the controllers
8  def login_required
9    redirect_to(root_url, notice: 'You have to be signed in to do that! Top right ~') unless current_user
10 end
```



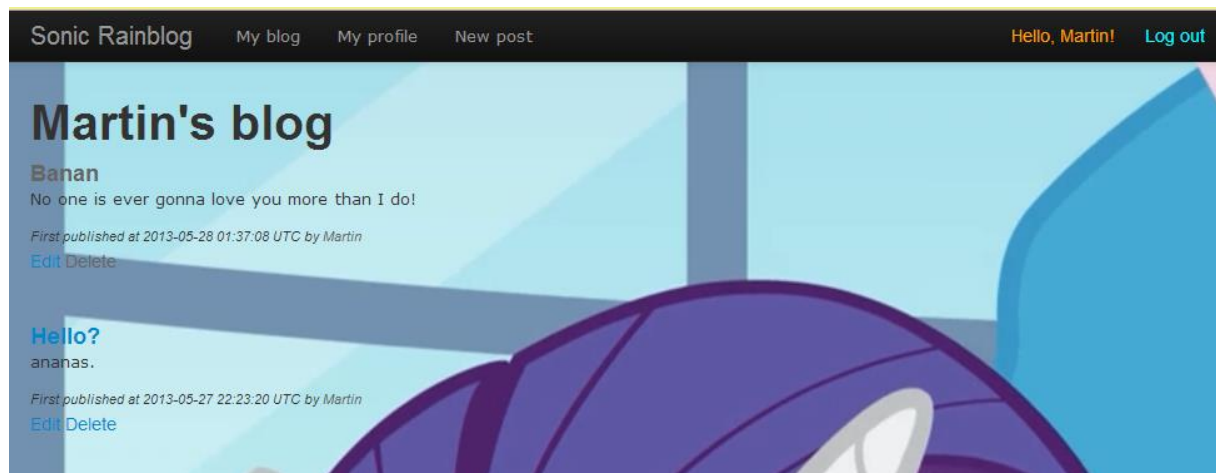
Blogginlegg skal minimum ha attributtene tittel, innhold, publiseringstid, og -dato

```
irb(main):008:0> Post
=> Post(id: integer, title: string, content: text, published: boolean, published_at: datetime, user_id: integer, created_at: datetime, updated_at: datetime)
```

Det skal kunne opprettes, endres, og slettes blogginlegg 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 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

Sonic Rainblog My blog My profile

Editing post

Title

Banan

Content

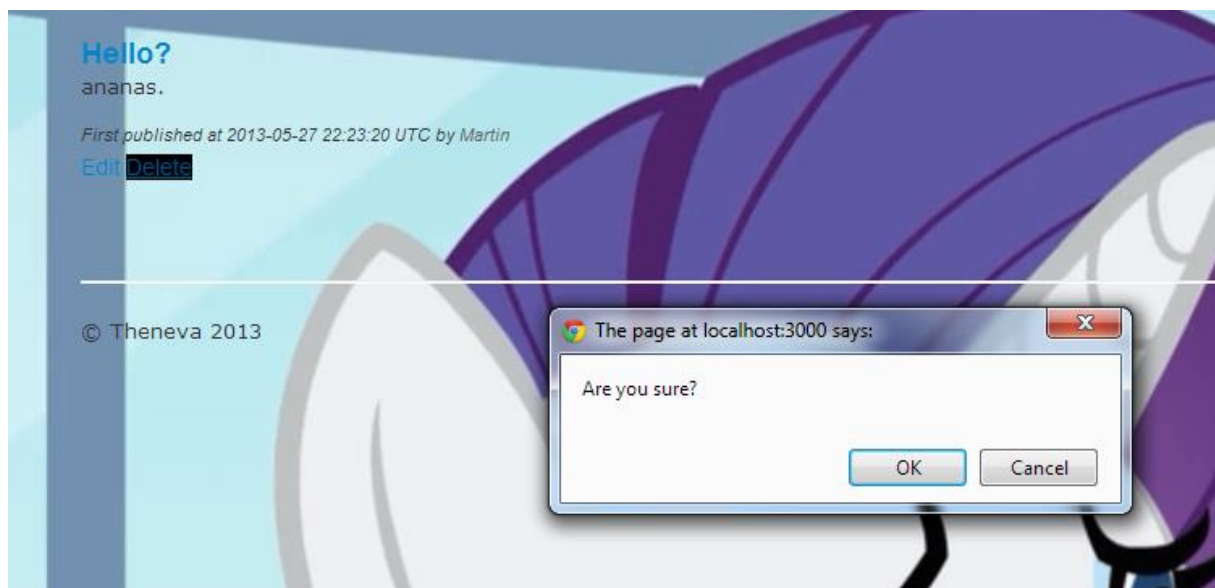
No one is ever gonna love you more than I do!

☒ Published

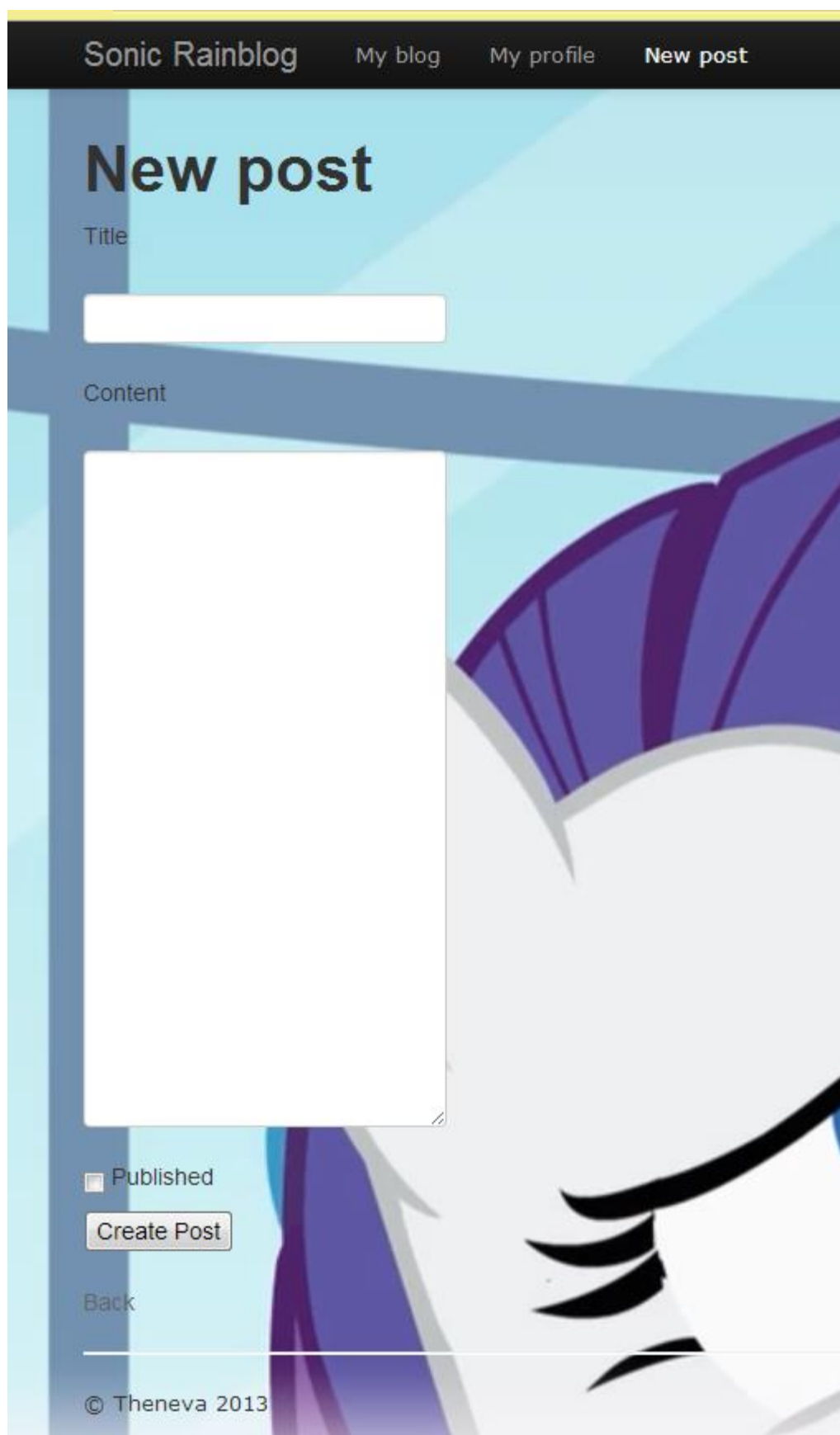
Update Post

Show | Back

Grensesnitt for å slette post



Grensesnitt for å opprette ny post



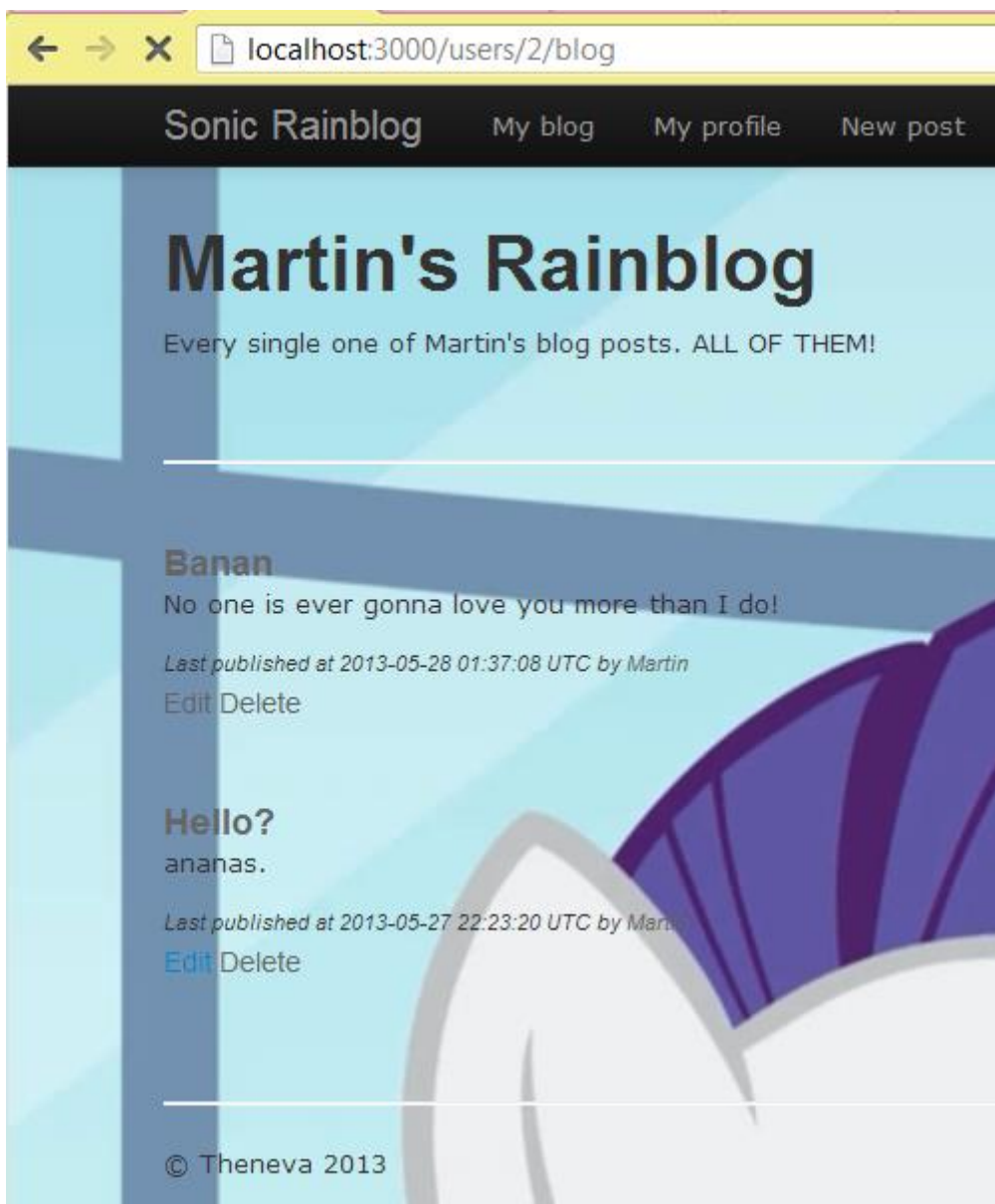
The screenshot shows a web interface for creating a new blog post. At the top, there is a navigation bar with the following links: **Sonic Rainblog**, [My blog](#), [My profile](#), and **New post**. The main heading is **New post**. Below the heading, there is a form with two main sections: **Title** and **Content**. The **Title** section has a single-line text input field. The **Content** section has a large, multi-line text area. At the bottom of the form, there is a checkbox labeled **Published** and a **Create Post** button. Below the form, there is a **Back** link. At the very bottom, there is a copyright notice: **© Theneva 2013**. The background of the interface features a stylized illustration of a character with purple hair and a white face.

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.

```
routes.rb
15 resources :users, except: :index do
16   resources :brohooves, only: [:create, :destroy]
17   get 'blog', on: :member
18 end
```

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.

```
post.rb
13 scope :available, lambda { |current_user_id| where('published = ? OR user_id = ?', true, current_user_id).order 'published_at DESC' }
14 scope :available_by_user, lambda { |user_id, current_user_id| available(current_user_id).where('user_id = ?', user_id) }
```

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

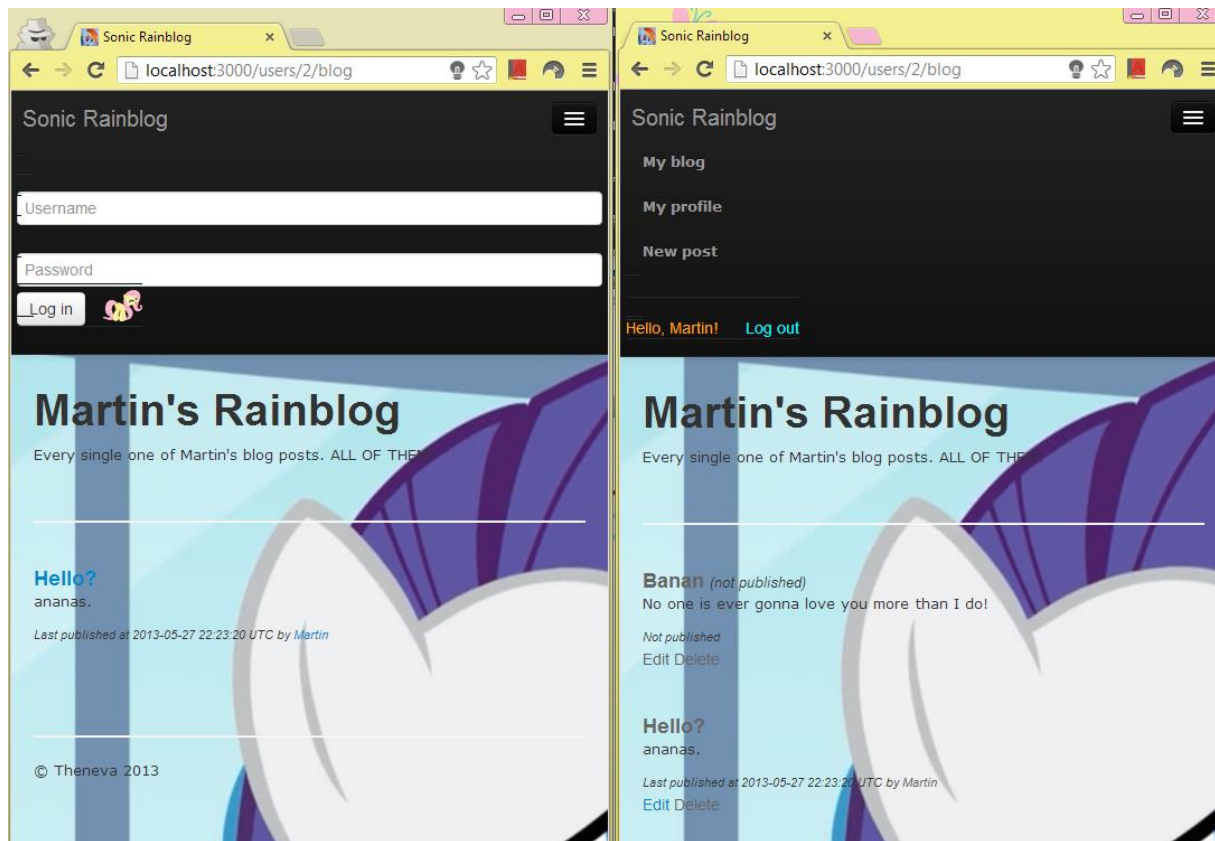
```
blog.html.erb
1 <h1><%= @user.username %>'s Rainblog</h1>
2
3 <p>
4   Every single one of <%= @user.username %>'s blog posts. ALL OF THEM!
5 </p>
6
7 <br />
8 <hr />
9 <br />
10
11 <%= render "posts/list" %>
```

... 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.

```
_list.html.erb
1 <% if @user.posts.any? %>
```

```
_list.html.erb
34 <% end %>
35 <% else %>
36   <h4>No posts :( </h4>
37 <% end %>
```

Blogginlegg 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 ikke 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.

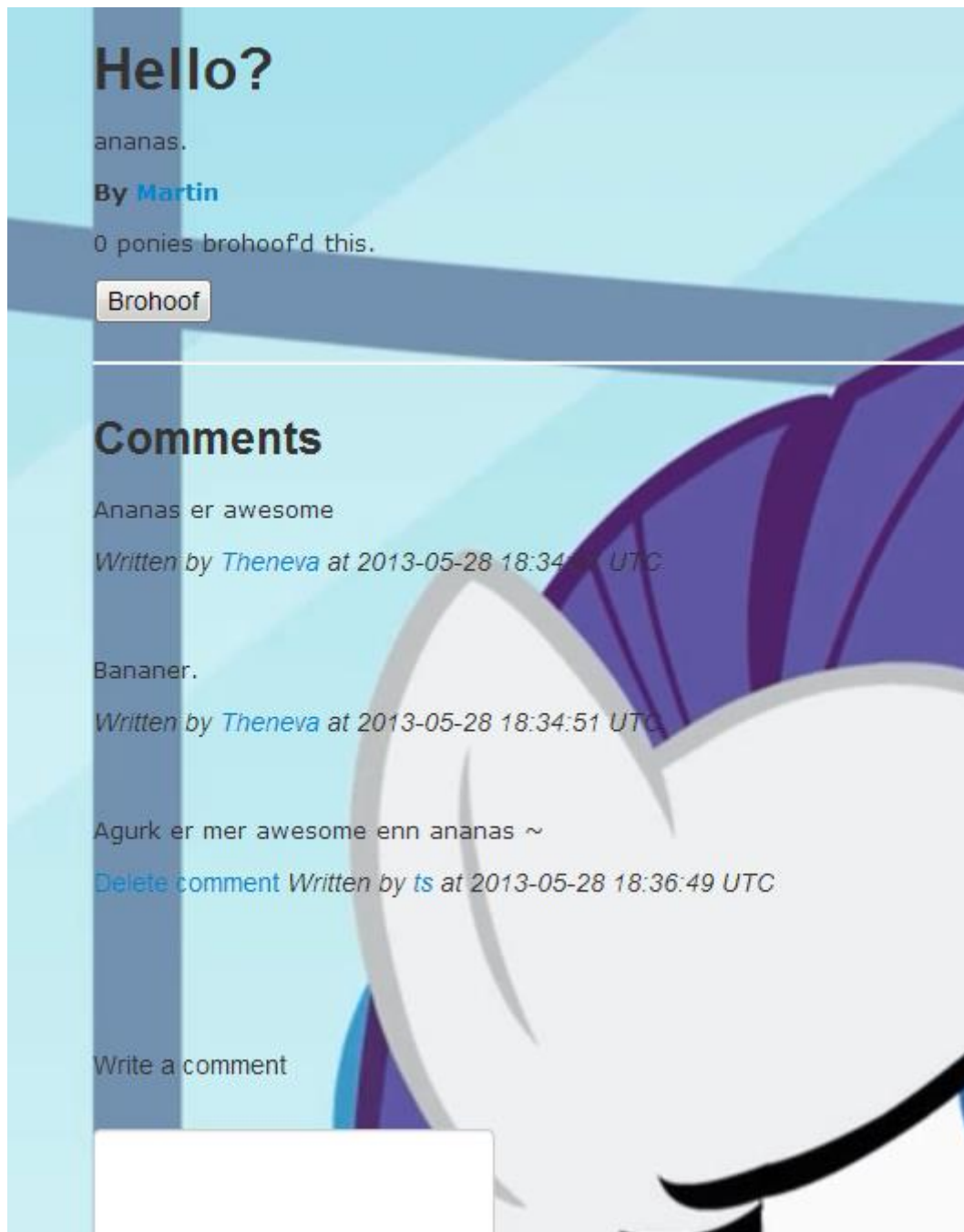
Oppgave 4: Kommentarer

Kommentarer skal minst inneholde innhold

```
irb(main):023:0> Comment  
=> Comment(id: integer, content: text, post_id: integer, user_id: integer, creat  
ed_at: datetime, updated_at: datetime)
```

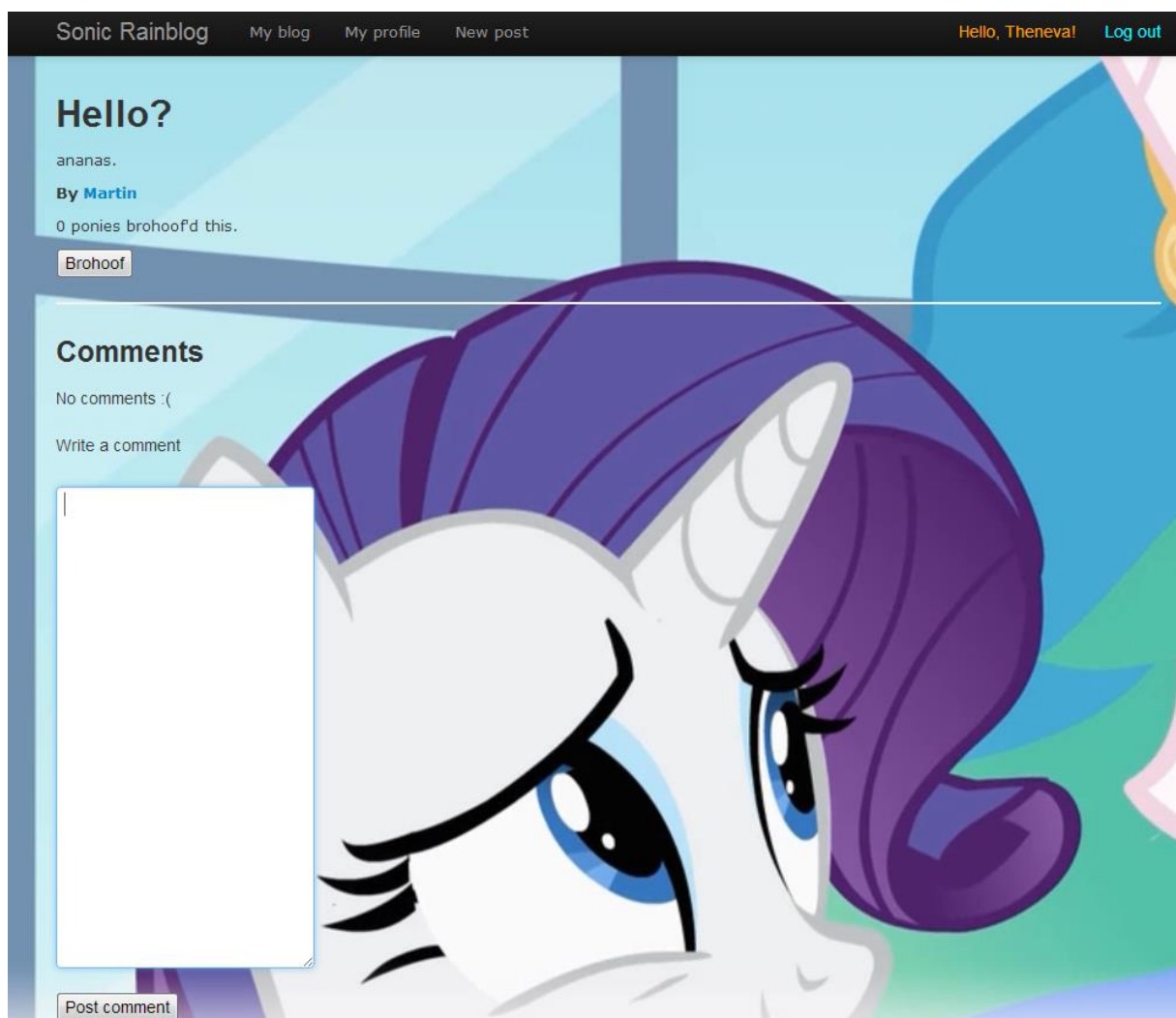
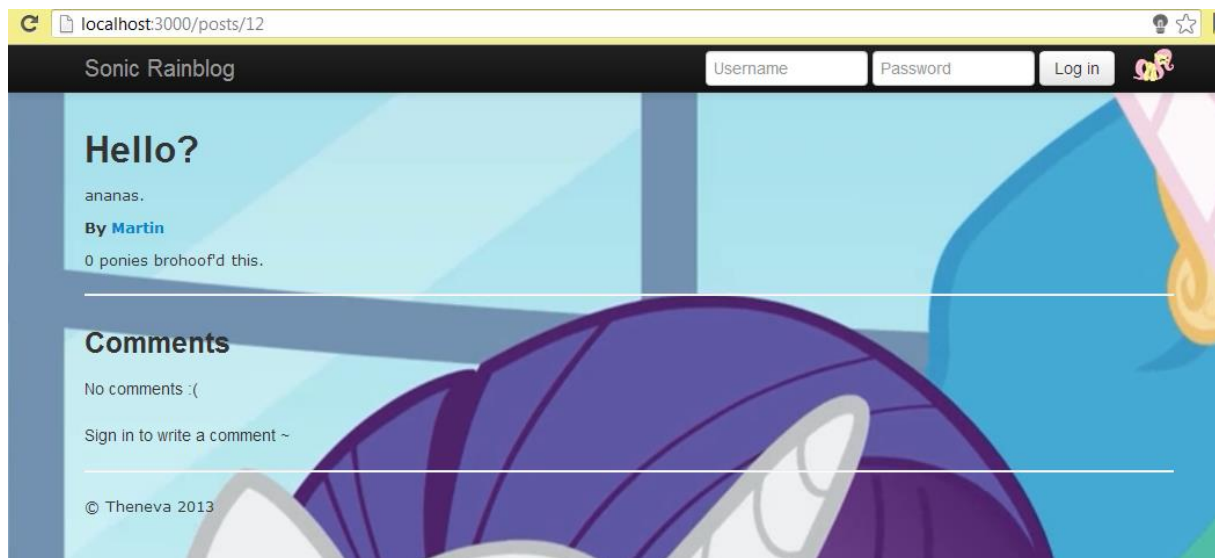
Det skal vises en oversikt over alle kommentarer som tilhører et innlegg på samme siden som innlegget vises (show-view)

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



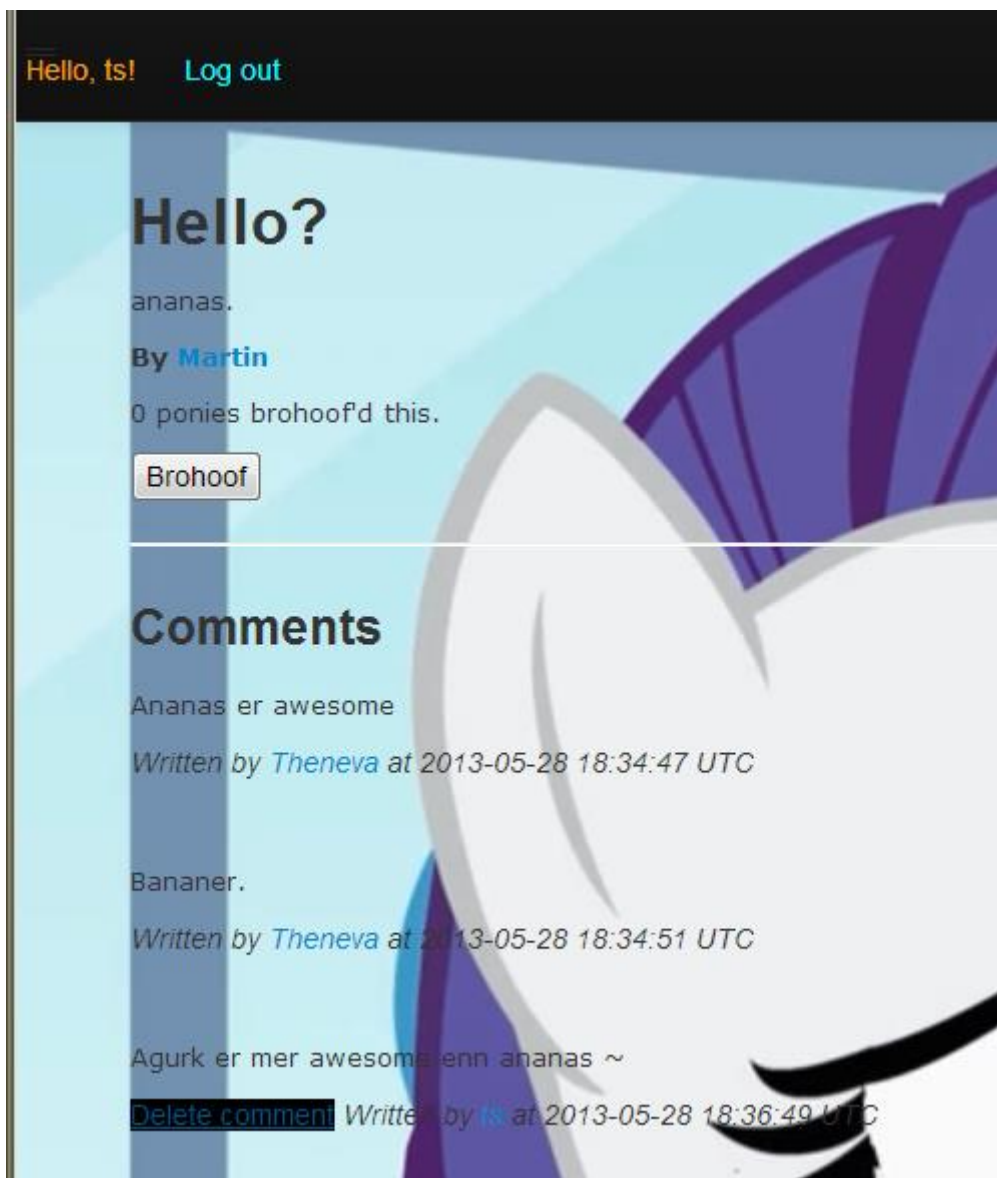
Dersom innlegget ikke har noen kommentarer, skal dette vises med en forklarende tekst & kun innloggede brukere skal ha lov til å kommentere på innlegg

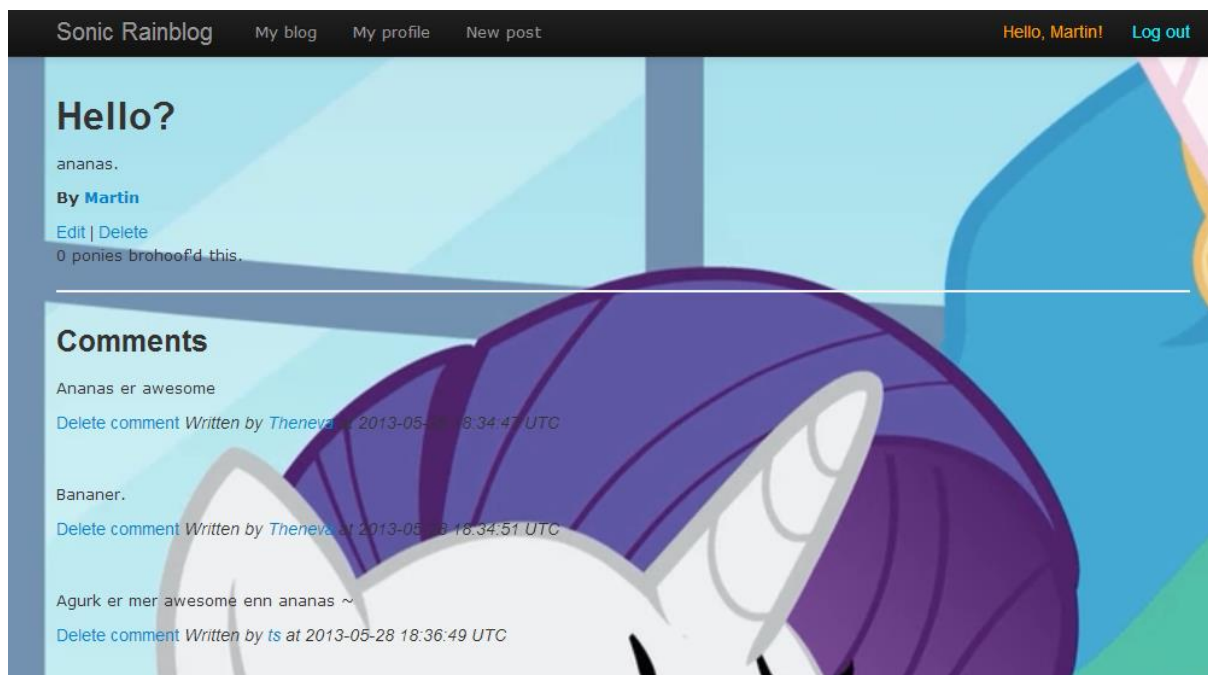
Det finnes ikke noen *new*-action for kommentarer, så jeg lar view-et avgjøre om form-et skal vises.



Det skal være mulig å slette sine egne kommentarer

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

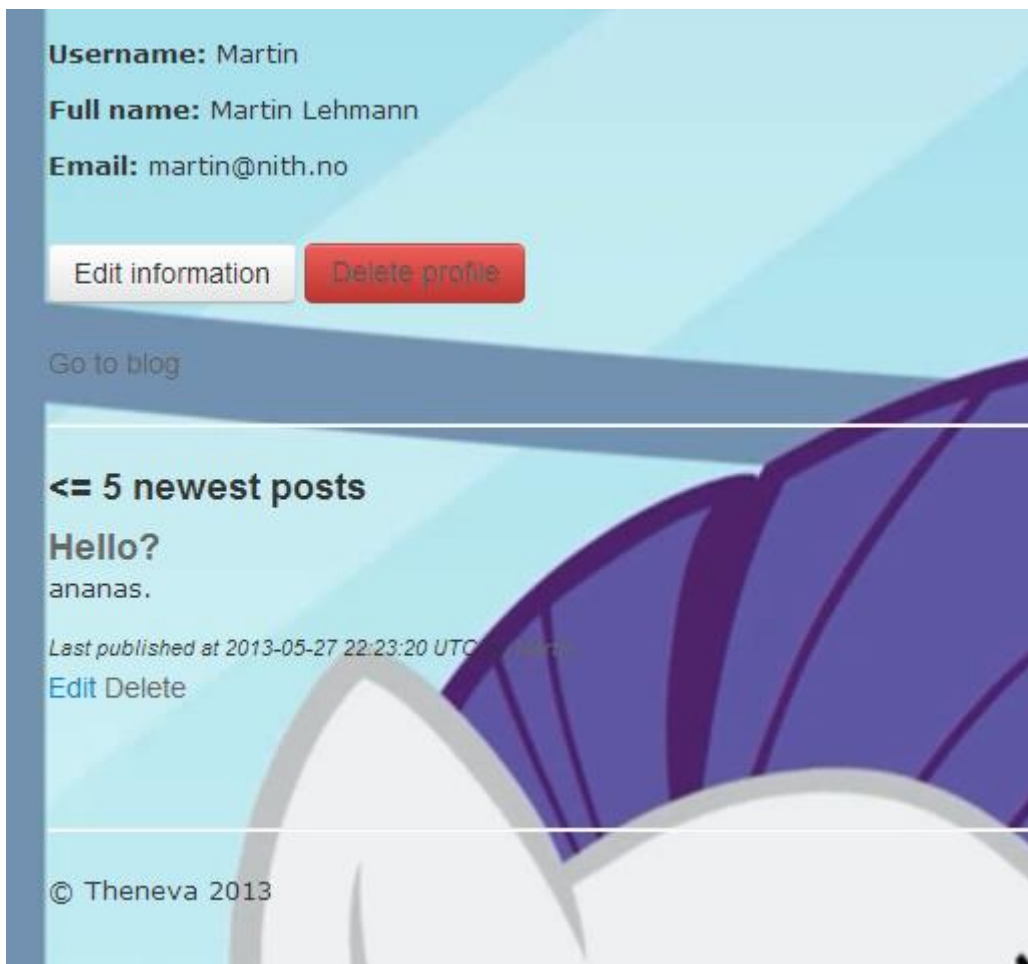




Oppgave 5: Profilsiden

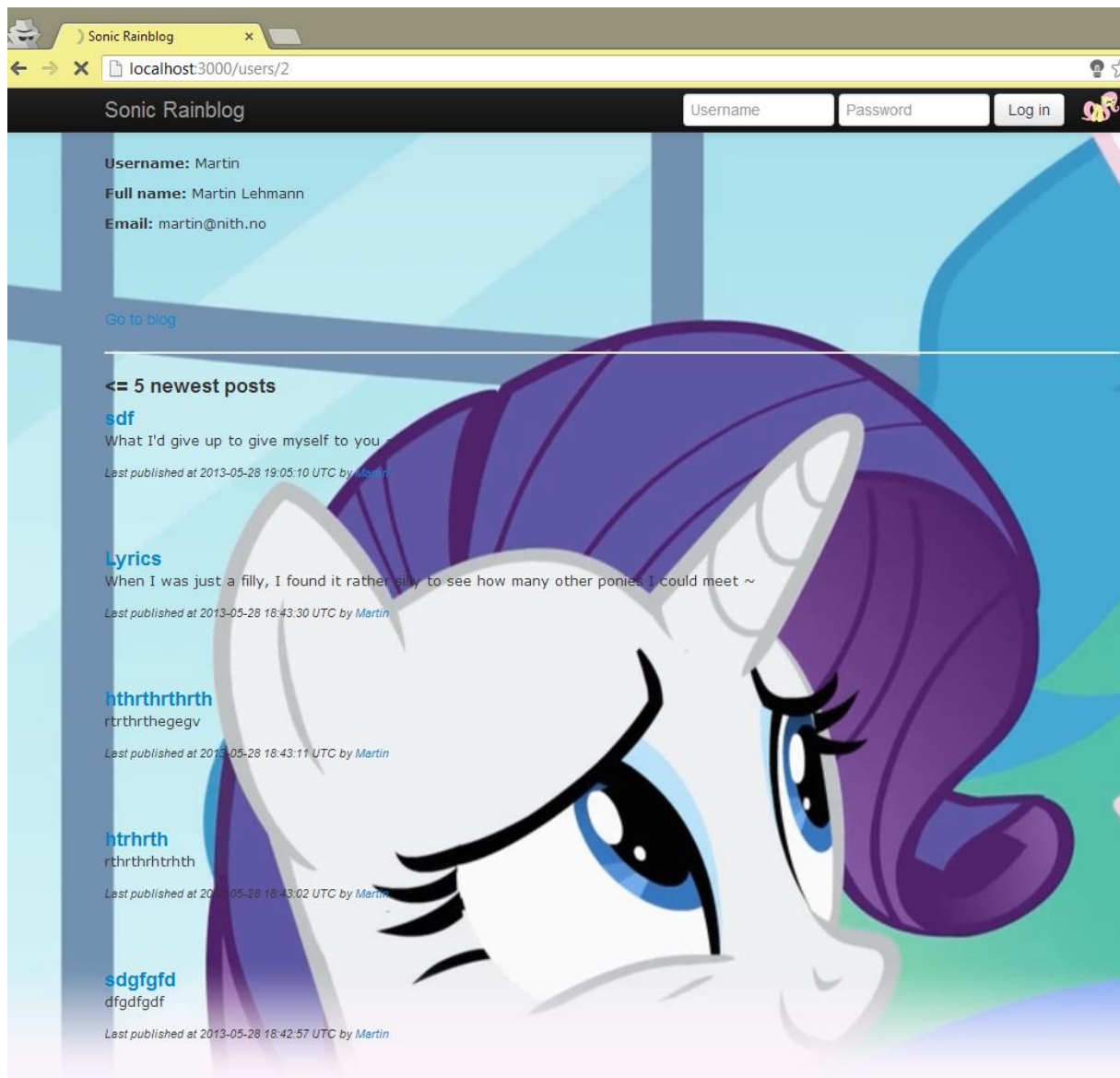
Profilsiden skal vise brukernavn, fullt navn, vise brukerens 5 siste publiserte innlegg (sortert på publiseringstidspunkt), og ingen innlegg i bloggen skal forklares med tekst; innleggene skal listes opp på samme måte som i oversiktssiden; innlogget bruker som ser på egen profil skal ha mulighet til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

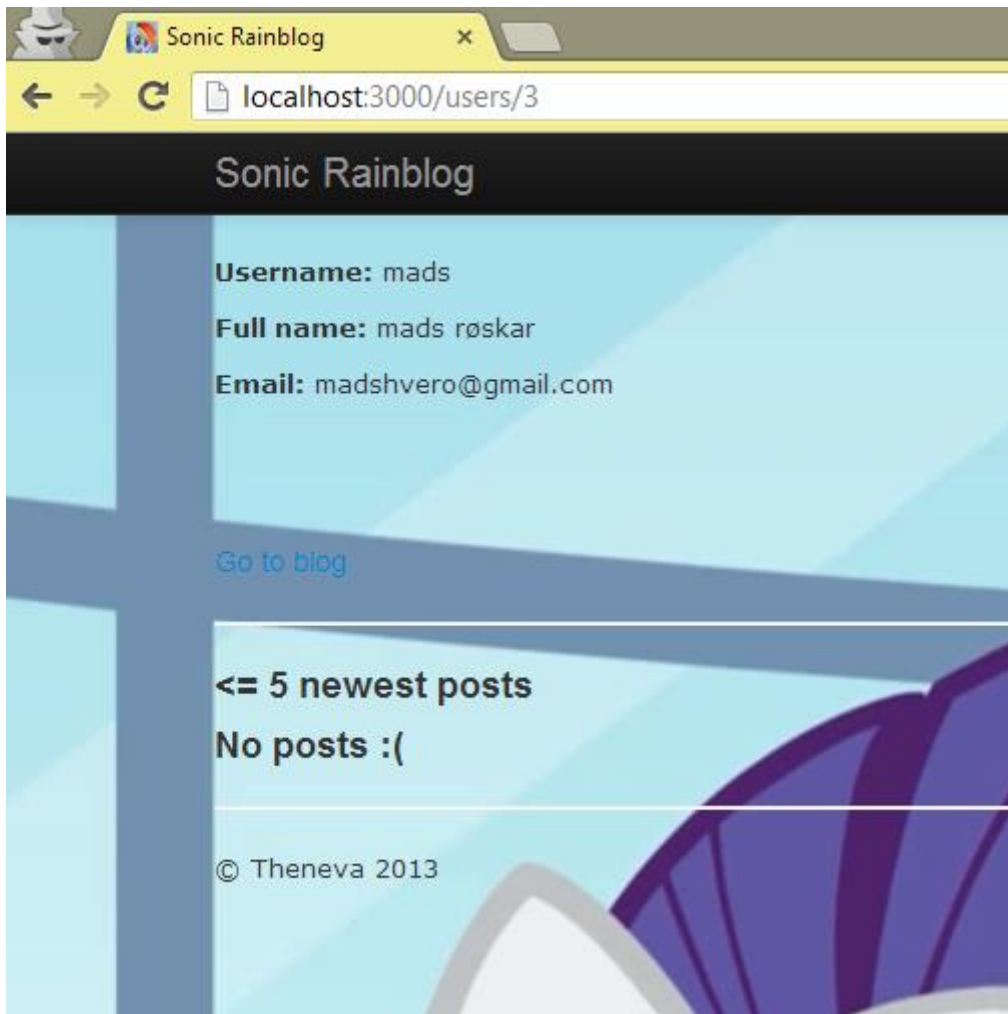


Her ser en ikke innlogget bruker på bloggen. En kan se av konsollen at brukeren Martin har 6 poster, men bare 5 av dem vises i «recent posts»-delen av profilen.

```
irb(main):036:0> User.find_by_username('Martin').posts.count
=> 6
irb(main):036:0> User.find_by_username('Martin').posts
=> [#<User Load (0.0ms)> SELECT "users".* FROM "users" WHERE "users"."username" = 'Martin' LIMIT 1, #<User Load (1.0ms)> SELECT COUNT(*) FROM "posts" WHERE "posts"."user_id" = 2<0ms>]
```



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.

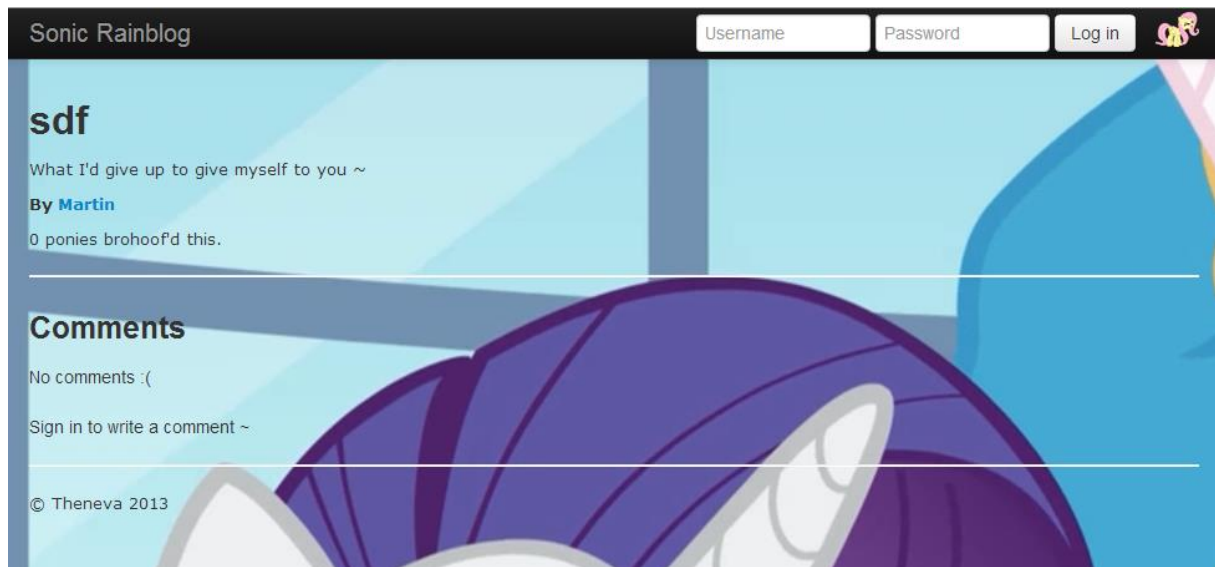


Oppgave 6: Fist bumps

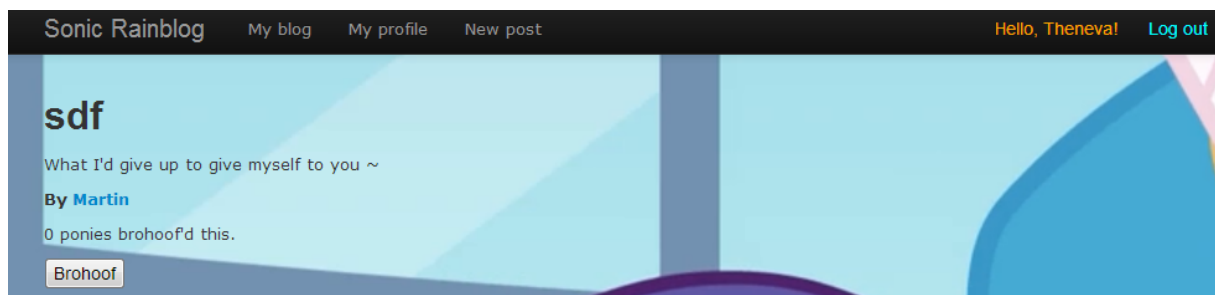
Nope; brohooves (som ga meg en unnskyldning for å leke med Inflector-klassen fordi pluralizeren hardnakket hevder av «brohoof» blir til «brohoofs» i flertall).

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

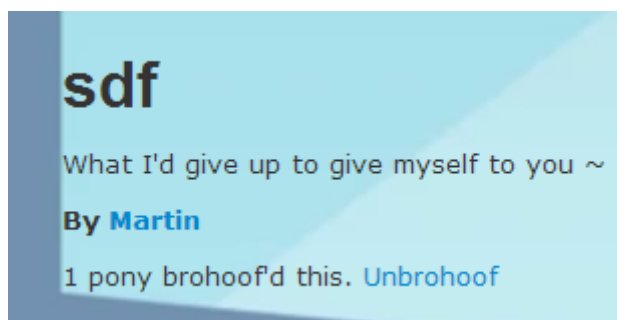
Ikke innlogget: får se antall brohooves, men ikke brohoofe innlegget selv.



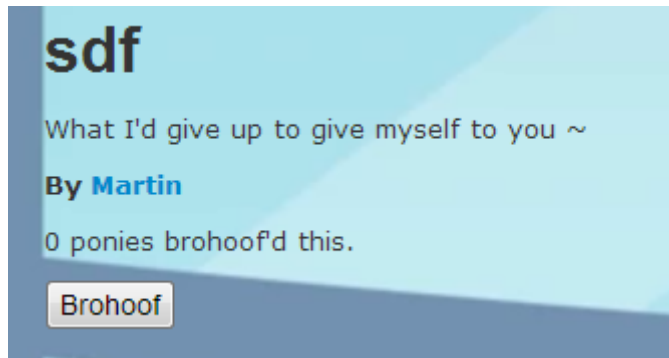
Innlogget på en annen profil enn eieren av innlegget; kan brohoofe innlegget



Brohoof



Unbrohoof

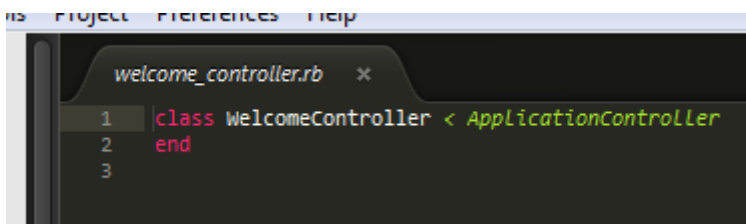
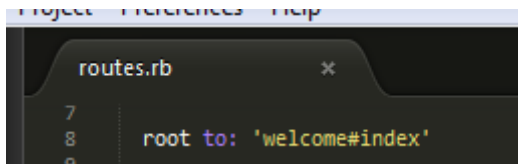
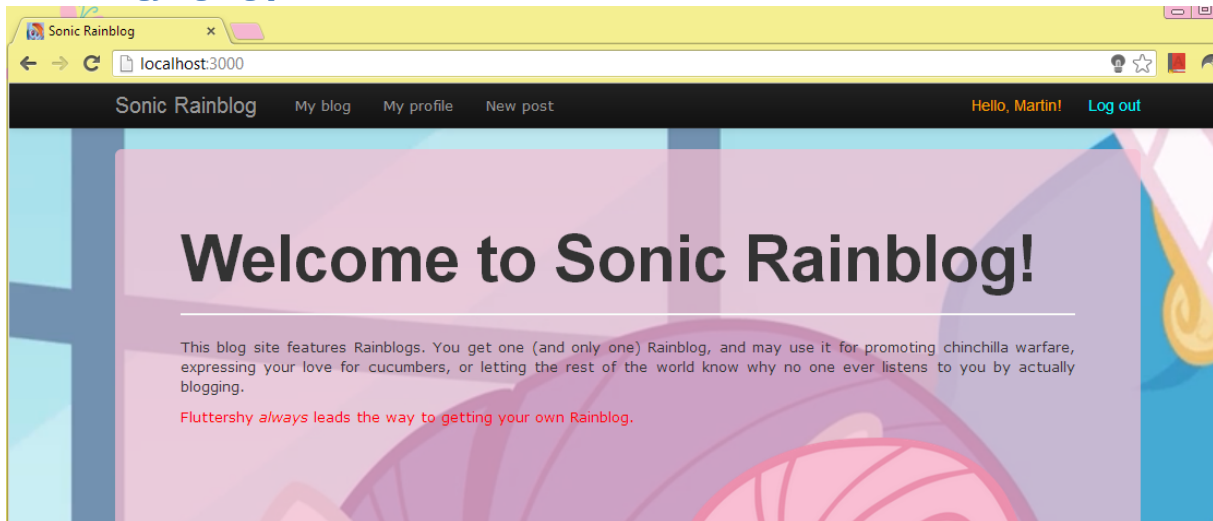


Det tilbys ikke noe grensesnitt for oversikt over hvem som har brohoofet innlegget, og det er dermed heller ikke mulighet for å endre andres favoritter; Brohoof har kun create og destroy som actions.

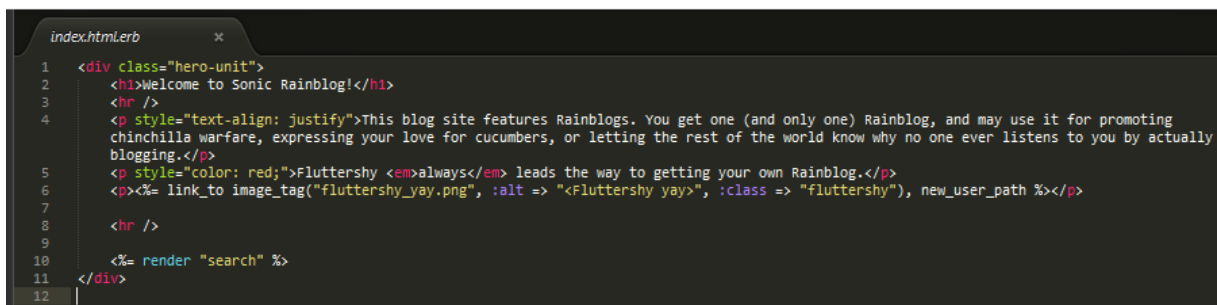
View-et tar for seg at det ikke er mulig å gi brohoof til egne innlegg ved å sjekke om innlogget bruker er postens eier.

Oppgave 7: Forside med brukersøk

Forside tilgjengelig på rot



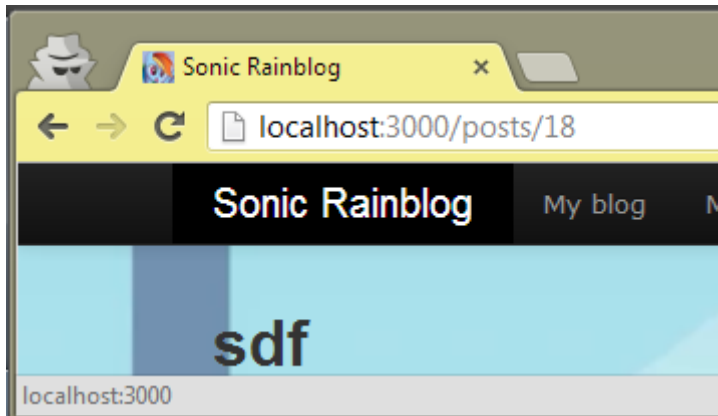
Welcome#index:



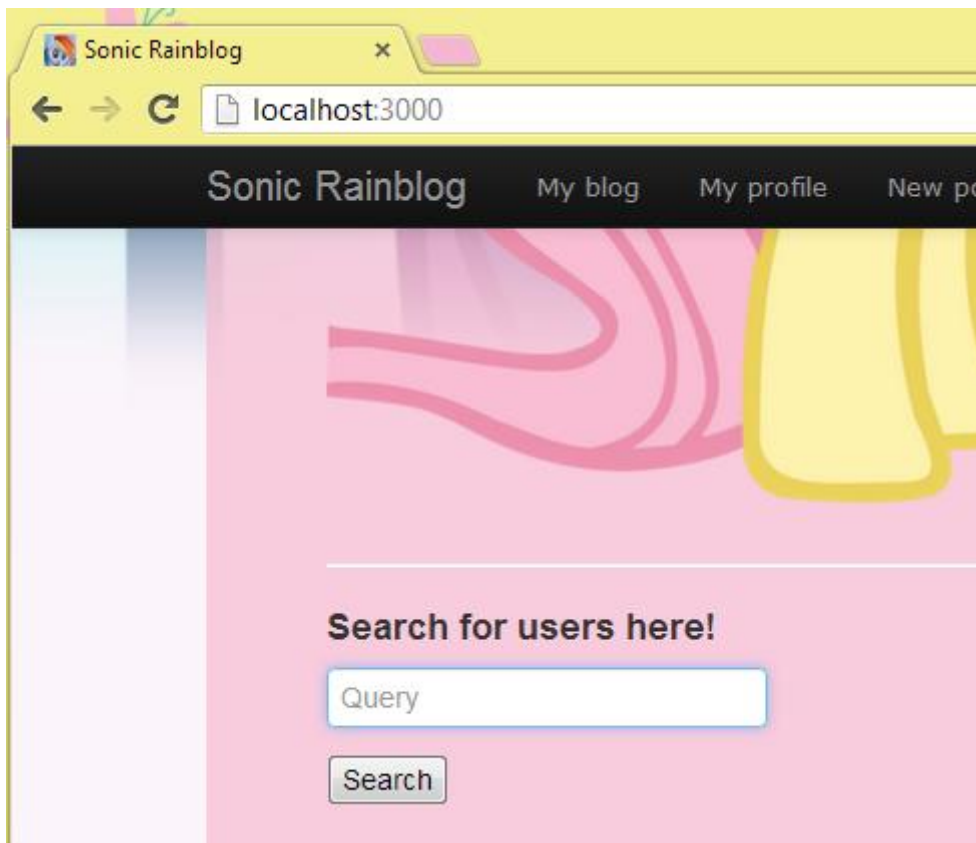
Det finnes så vidt jeg har klart å finne ikke noen konvensjon for hva en «landing page»/forside-resource skal kalles, så jeg gikk med Welcome fordi dette er eksempelverdien i routes-fila.

Det skal på en hver side vises en link til forsiden

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

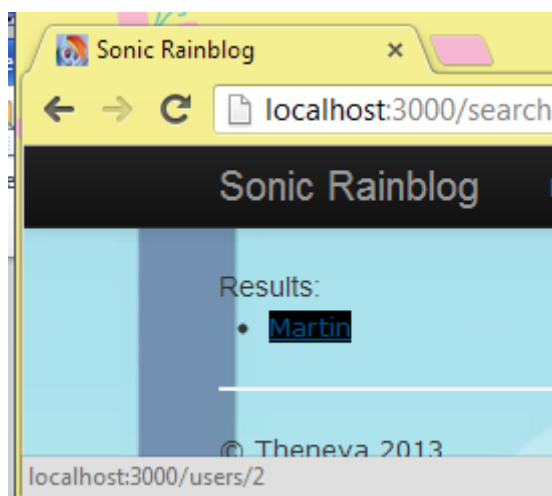
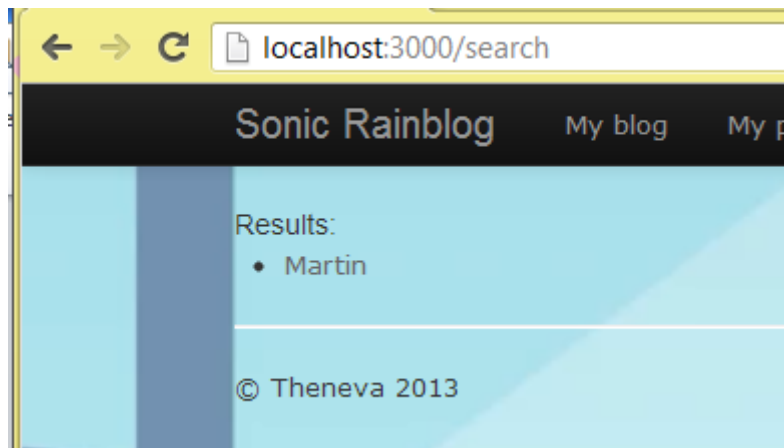


Forsiden skal inneholde et søkefelt, og ved å bruke dette skal en kunne søke på brukernavn; søket skal returnere alle brukernavn som inneholder søkestrengen, ved å bruke SQL-kodeordet LIKE; brukerne som blir funnet i søket skal presenteres på resultatsiden; det skal være mulig å klikke på brukernavnet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

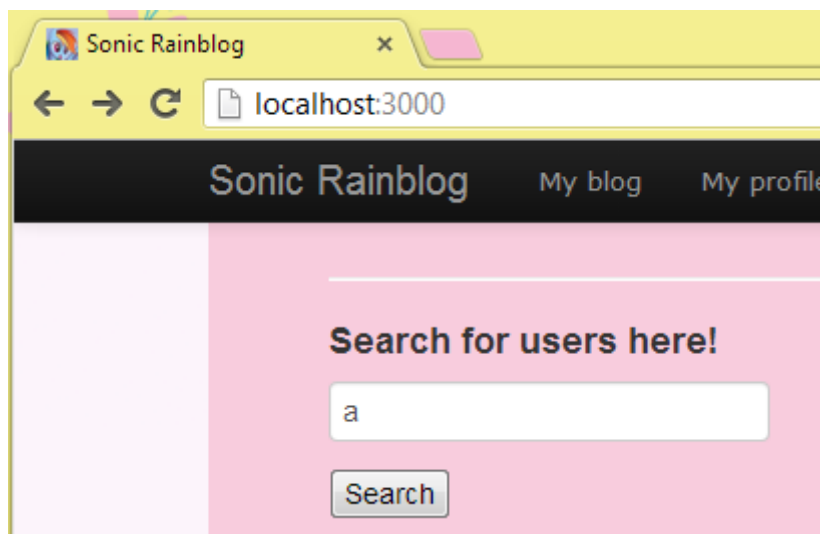


Search for users here!

Search



Eksempel 2:



Search

