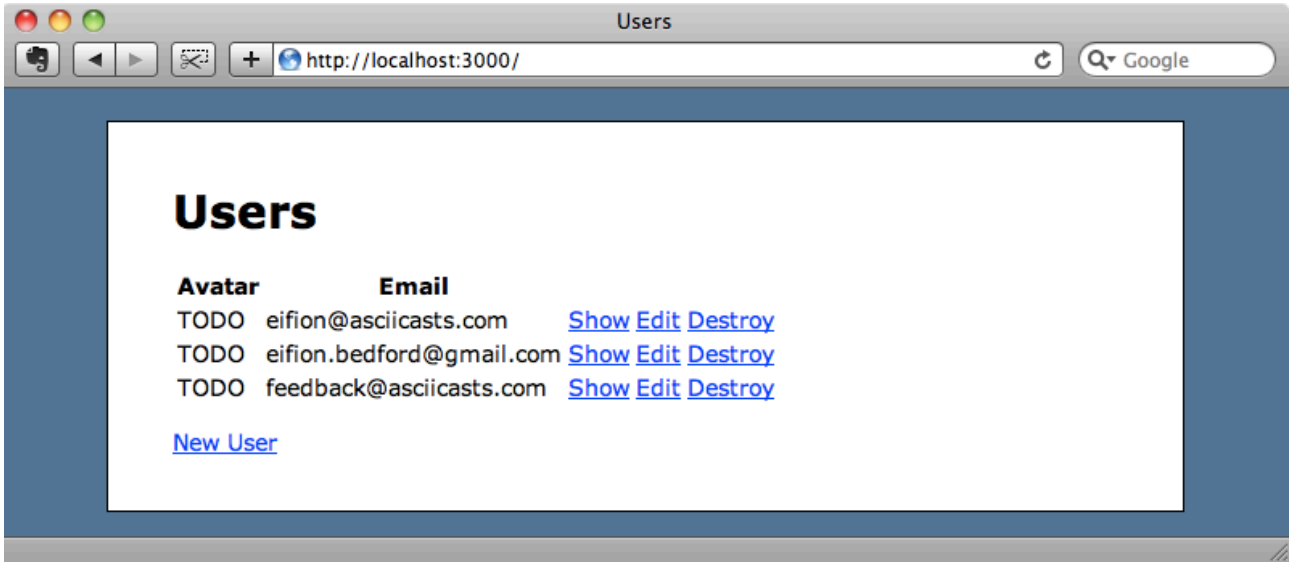
Episode 244

Gravatar

Avatar images are playing a bigger role on the web these days, especially in "social" web applications. If you want to add avatar images to one of your own applications you might want to consider using Gravatar[1] which provides a convenient way to do this. All you need is an email address for each user so there's no need for your application to handle file attachments, image cropping or to worry about users uploading inappropriate images as all of this is handled by Gravatar. Just give it an email address and the avatar for that user is returned.

In this episode we'll add Gravatar to a simple Rails application with a `User` model. It currently has three users, each with a different email address and the `index` page shows an `avatar` column marked "TODO". We'll use Gravatar to add an avatar for each user.



There are a number of plugins available for adding Gravatar to Rails but it's so easy to do from scratch that we won't use one. We'll start by modifying the view code that renders the table shown above, which looks like this:

[1] http://en.gravatar.com/

/app/views/users/index.html.erb

```
<% for user in @users %>
  <tr>
    <td>TODO</td>
    <td><%= user.email %></td>
    <td><%= link_to "Show", user %></td>
    <td><%= link_to "Edit", edit_user_path(user) %></td>
    <td><%= link_to "Destroy", user, :confirm => 'Are you ↵
      sure?', :method => :delete %></td>
  </tr>
<% end %>
```

We'll replace the TODO text in the first table cell with an `image_tag`. The image's URL will come from a helper method that we'll write called `avatar_url` that takes a user an an argument.

/app/views/users/index.html.erb

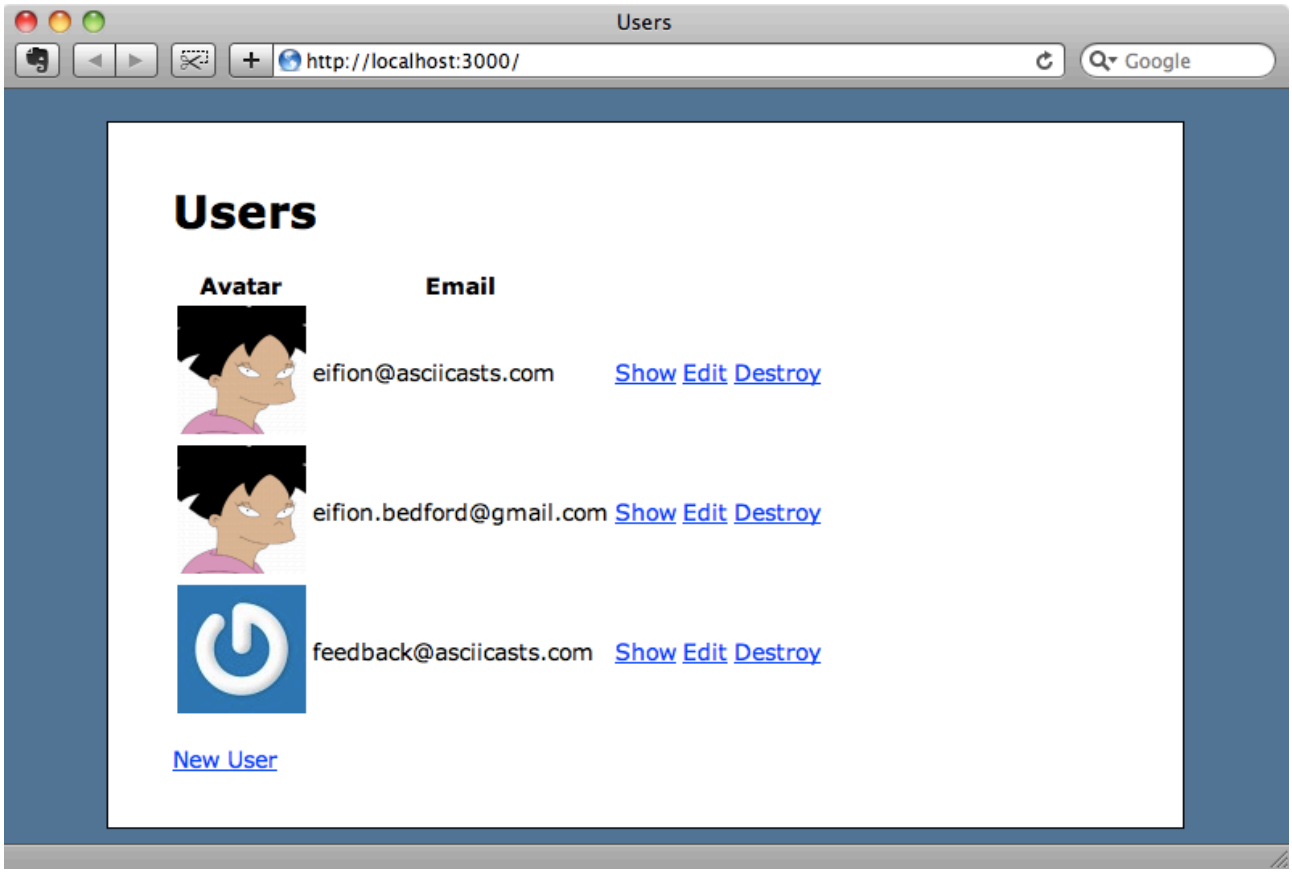
```
<% for user in @users %>
  <tr>
    <td><%= image_tag avatar_url(user) %></td>
    <td><%= user.email %></td>
    <td><%= link_to "Show", user %></td>
    <td><%= link_to "Edit", edit_user_path(user) %></td>
    <td><%= link_to "Destroy", user, :confirm => 'Are you ↵
      sure?', :method => :delete %></td>
  </tr>
<% end %>
```

Next we'll write the `avatar_url` method in the `ApplicationHelper` class. The only slightly tricky part of this is generating the `id` that gravatar expects. This is a lowercase MD5 hash of the user's email address, so even this is fairly straightforward.

/app/helpers/application_helper.rb

```
module ApplicationHelper
  def avatar_url(user)
    gravatar_id = Digest::MD5::hexdigest(user.email).downcase
    "http://gravatar.com/avatar/#{gravatar_id}.png"
  end
end
```

When we reload the page now we'll see the avatar images for the two accounts below that have Gravatar images associated with them. For the one that hasn't we instead get the default Gravatar image.



Customizing Gravatar Images

There are a number of parameters that can be passed to the querystring in the Gravatar URL to customize its behaviour and you can find details of these on the Gravatar site[2]. For example, passing an `s` parameter will change the size of the avatar, like this:

```
http://gravatar.com/avatar/9a295994737a47683a4da4ed47aef7dd.png?s=200
```

The `d` option can be used to supply an alternative default image if no avatar is found for a user, like this:

[2] http://en.gravatar.com/site/implement/images

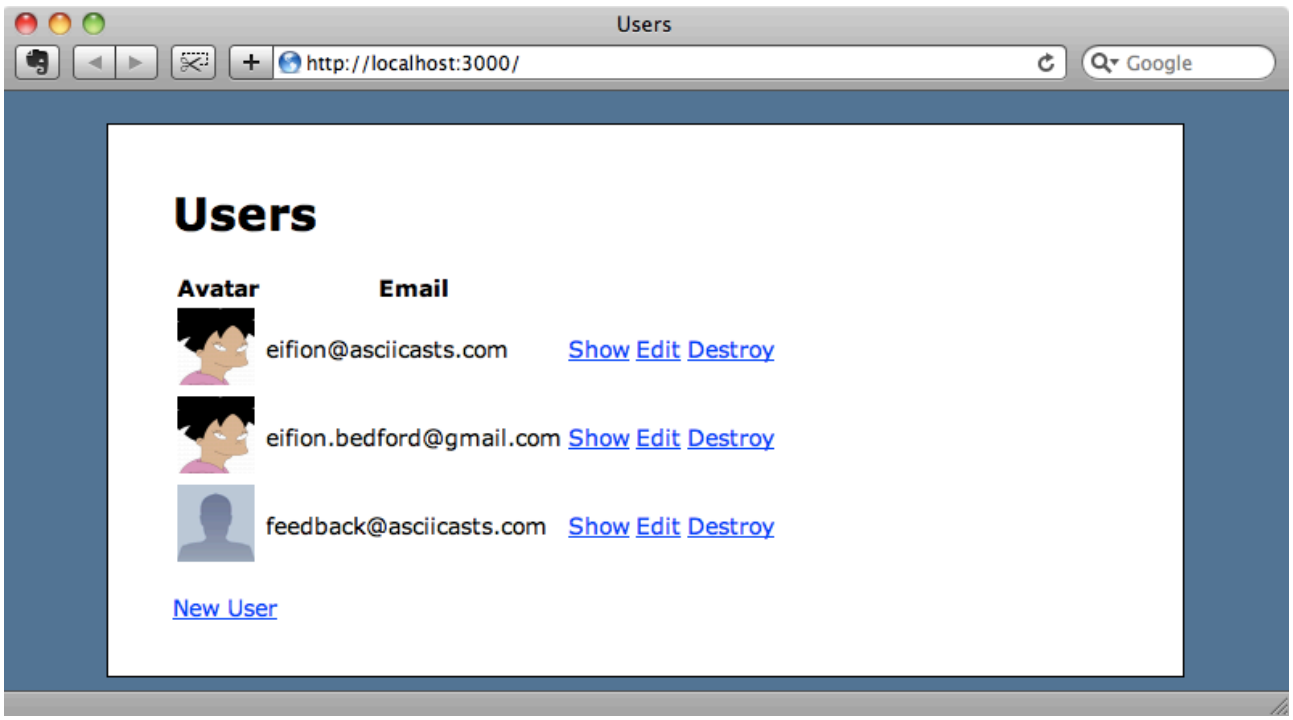
```
http://gravatar.com/avatar/84ce1adb94b67014236a909fa4c1269d.png?
d=http%3A%2F%2Fasciicasts.com%2Fimages%2Frails.png
```

We can also pass in an `r` option to change the rating and if our site uses HTTPS then we can get images securely by using `https://secure.gravatar.com/` at the start of the URL. We'll use a couple of these options now to resize our application's avatars to 48 pixels, as it's the same size that Twitter uses, and to provide a different default image.

/application/helpers/application_helper.rb

```
module ApplicationHelper
  def avatar_url(user)
    default_url = "#{root_url}images/guest.png"
    gravatar_id = Digest::MD5::hexdigest(user.email).downcase
    "http://gravatar.com/avatar/#{gravatar_id}.png?s=48&d=#
{CGI.escape(default_url)}"
  end
end
```

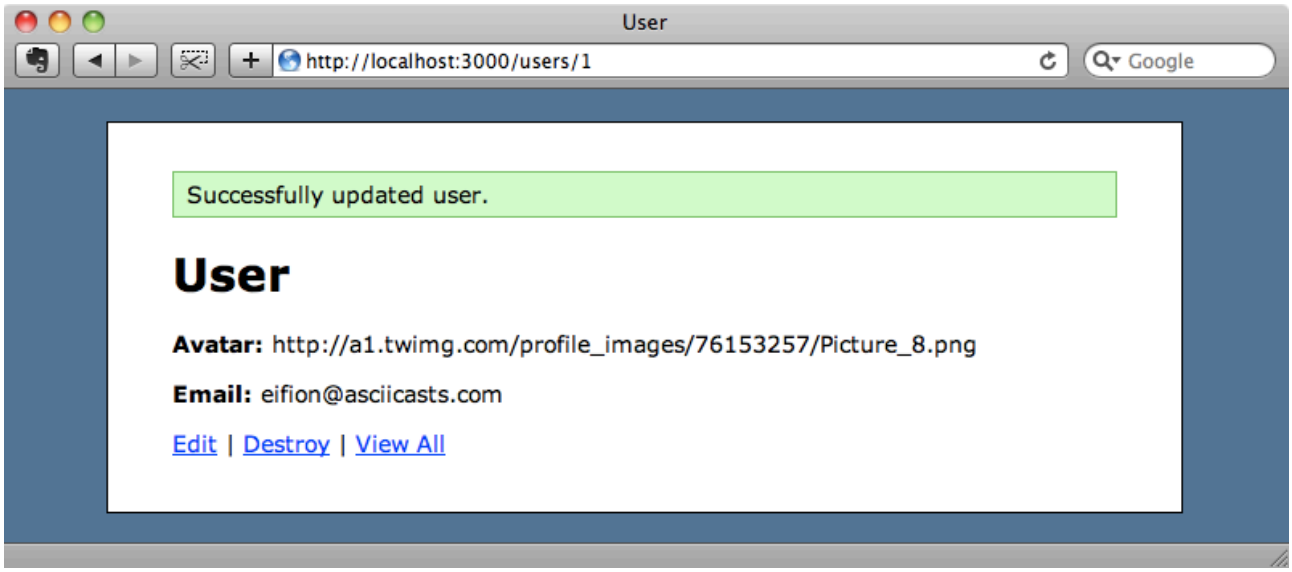
When we reload the page now the images will be smaller and our custom default image is shown for the account that has no Gravatar image.



Using Images From Other Sites With OmniAuth

This avatar solution works especially well when it's paired up with OmniAuth. If you're not familiar with OmniAuth episode 241 [watch[3], read[4]] covered how to get started with using it in a Rails application. The reason OmniAuth works well here is that if a user signs in to your website with Facebook or Twitter then they'll already have an avatar image from those services and this image can be fetched from the `image` property of `user_info` hash that OmniAuth returns. If a user hasn't signed in with one of these services them we can use Gravatar as a fallback so that each user always has an avatar.

We won't add OmniAuth to this application, instead we've added an `avatar_url` field to our `User` model to simulate this part of the OmniAuth data. First we'll modify one of the users to add an avatar image from Twitter.



Next we need to modify our `avatar_url` method in the `ApplicationHelper` so that it will use a user's stored avatar if one is present.

---

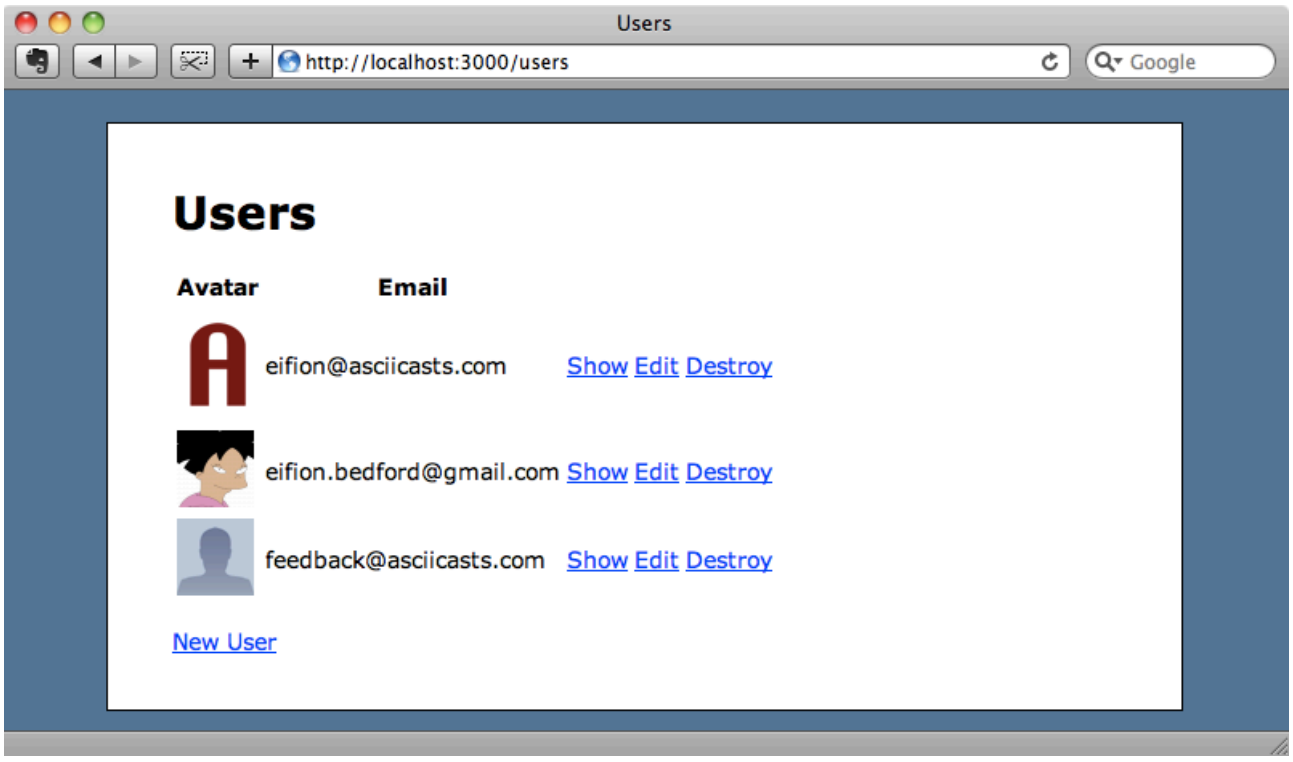
[3] http://railscasts.com/episodes/241-simple-omniauth

[4] http://asciicasts.com/episodes/241-simple-omniauth

/app/helpers/application_helper.rb

```
module ApplicationHelper
  def avatar_url(user)
    if user.avatar_url.present?
      user.avatar_url
    else
      default_url = "#{root_url}images/guest.png"
      gravatar_id = Digest::MD5::hexdigest(user.email).downcase
      "http://gravatar.com/avatar/#{gravatar_id}.png?s=48&d=#
{CGI.escape(default_url)}"
    end
  end
end
```

When we reload the page now the account with its own avatar image will display that image while the others will still be fetched from Gravatar.



That's it for this episode. It's been a short one but hopefully it has covered enough to get you started if you want to use avatars in your Rails applications.