**User Authentication: Devise gem**

**Link: https://github.com/plataformatec/devise**

**1. Add devise gem in to Gemfile**

gem 'devise'

**2. Run “bundle install”**

**3. Next, you need to run the generator:**

$ rails generate devise:install

4. **Generate model**

rails generate devise MODEL

MODEL cam be User, Admin, Customer e.t.c

5. **Run “rake** [**db:migrate**](db:migrate)**”**

6. **Restart server**

7. **Add Signount links http://railscasts.com/episodes/209-introducing-devise**

<div id="user\_nav">

<% if user\_signed\_in? %>

Signed in as <%= current\_user.email %>. Not you?

<%= link\_to "Sign out", destroy\_user\_session\_path, :method => :delete %>

<% else %>

<%= link\_to "Sign up", new\_user\_registration\_path %> or <%= link\_to "sign in", new\_user\_session\_path %>

<% end %>

</div>

[https://launchschool.com](https://launchschool.com/)

### …or create a new repository on the command line

echo "# vijayblog" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/vijay133404/vijayblog.git

git push -u origin master

### …or push an existing repository from the command line

git remote add origin https://github.com/vijay133404/vijayblog.git

git push -u origin master

## office

<http://vijaymaurya.herokuapp.com/>

<https://github.com/vijay133404>

<https://github.com/vijay133404/vijayblog>

<https://github.com/vijay133404/vijayapp.git>

https://github.com/vijay133404/vijaymyapp.git

### room

<https://github.com/vijay133404/vijay-room>

<http://vijayroom.herokuapp.com/>

teacher

<https://github.com/vijay133404/teacherblog>

<https://teacherdetails.herokuapp.com/>

# task(blogs)

# upload image file using carrierwave followes few step

1 $ gem install carrierwave -v "1.0.0"

2 n Rails, add it to your Gemfile:

3 gem 'carrierwave'

4 rails generate uploader Avatar

5 rails g migration add\_avatar\_to\_users avatar:string(OPTIONAL)

IN MODEL(Blog) add (mount\_uploader :image, AvatarUploader)

6 rake db:migrate

1 data base

database .yml

default: &default

adapter: postgresql

pool: 5

username: root

password: root

timeout: 5000

development:

<<: \*default

database: teacher\_dev

# Warning: The database defined as "test" will be erased and

# re-generated from your development database when you run "rake".

# Do not set this db to the same as development or production.

test:

<<: \*default

database: teacher\_dev

production:

<<: \*default

database: teacher\_dev

add new coumn

Add new columns:

1. Add new migration:

rails generate migration add\_last\_name\_to\_users last\_name:string

2. rake db:migrate

Permit newParameteres in Devise:

class ApplicationController < ActionController::Base

protect\_from\_forgery with: :exception

before\_action :configure\_permitted\_parameters, if: :devise\_controller?

protected

def configure\_permitted\_parameters

devise\_parameter\_sanitizer.permit(:sign\_up, keys: [:first\_name,

:last\_name])

end

end

1.bittern-109@bittern109-desktop:~/projects/blog3$ rails g migration add\_user\_id\_to\_teachers user\_id:integer

2.controller create method in push

def create

@teacher = Teacher.new(teacher\_params)

@teacher.user\_id = current\_user.id

3.create new method for myblog status

def my\_teachers

@teachers = Teacher.where(:user\_id => current\_user.id)

end

4.in index.html file we put the condtion of user something like that

if current\_user.id == teacher.user\_id =%>

<td><%= link\_to 'edit', edit\_teacher\_path(teacher) %></td>

<td><%= link\_to 'delete', teacher\_path(teacher)

5. create my teachers(blog) action in index or my blog both are same code

6. last step we work routes.rb

get "/my\_teachers" =>"teachers#my\_teachers"

asks to do in the App:

1. Create a new application using database postgres.

2. Integrate devise for signin, signup, signout and forgot password.

3. Integrate active admin(<https://github.com/activeadmin/activeadmin>).

4. Allow admin to create the categories.

5. Allow user to create the blogs with title, image, dropdown category(select\_tag) and description.

6. Allow user to search the blogs with keyword(Reference: <https://rubyplus.com/articles/3381-Simple-Search-Form-in-Rails-5>).

7. Allow user to filter the blogs within a date range.

8. Allow user to comment on the blogs.

9. Integrate the template in bootstrap (after completing the above points, refenrece: <https://github.com/twbs/bootstrap-sass>).

10. Push the application to github.

11. Deploy the application to heroku.

Regards

Arvind

database yml

# SQLite version 3.x

# gem install sqlite3

#

# Ensure the SQLite 3 gem is defined in your Gemfile

# gem 'sqlite3'

#

default: &default

adapter: postgresql

pool: 5

username: root

password: root

timeout: 5000

development:

<<: \*default

database: teacher\_dev

# Warning: The database defined as "test" will be erased and

# re-generated from your development database when you run "rake".

# Do not set this db to the same as development or production.

test:

<<: \*default

database: teacher\_dev

production:

<<: \*default

database: teacher\_dev

3. meta serch finite loop(error)

gem 'activeadmin', github: 'activeadmin'

=============

rails version above 5.0

gem 'activemodel-serializers-xml', github: 'rails/activemodel-serializers-xml'

gem 'draper'

============

4 forigin key dependency show then put model in

dependent: :destroy

# 1.Integrate active admin(<https://github.com/activeadmin/activeadmin>).

# Flow these link and create these admin pannl and in admin pannl we add blogs model through migration and add scaffold commnd and add category and as well as product and next step we category and product model create.

## 2.When we create active admin the run admin then show the these error

## then we show the active controller

## permit\_params :name, :description, :attributes, :on, :model

## ActiveModel::ForbiddenAttributesError

# 1.add blogs table category\_id column

Add new columns:   
  
 1. Add new migration:   
    rails generate migration add\_last\_name\_to\_users last\_name:string   
  
 2. rake db:migrate

# 1.show

<p> <strong>Category:</strong> <%=Category.find(@blog.category\_id) %> </p>

# 2. new

<%= f.select(:category\_id, Category.all.collect {|u| [u.name, u.id]}, :prompt => 'Select Category') %>

# 3.index

<td> <%if blog.category\_id.present? %> <%= Category.find(blog.category\_id).name %> <% end %> </td>

4. edit page show then put

<%= f.select(:category\_id, Category.all.collect {|u| [u.name, u.id]}, :prompt => 'Select Category') %>

https://github.com/twbs/bootstrap-sass

10. Push the application to github.

11. Deploy the application to heroku.

Search:

1. In view:

<%=form\_tag "/blogs" ,:method=>"GET" do %>

<%= text\_field\_tag :search, params[:search] , :placeholder => "Type here to search"%>

<%= submit\_tag "Search"%>

<%end%>

2.In controller:

if params[:search].present?

@blogs= Blog.where('title LIKE ? OR description LIKE ?', "%#{params[:search]}%", "%#{params[:search]}%")

elsif params[:start\_date].present? && params[:end\_date].present?

@blogs = Blog.where('created\_at > ? AND created\_at < ?', params[:start\_date].to\_date, params[:end\_date].to\_date)

else

@blogs=Blog.all

end

# [Rails Active Admin css conflicting with Twitter Bootstrap css](http://stackoverflow.com/questions/10180418/rails-active-admin-css-conflicting-with-twitter-bootstrap-css)

For me changing application.css to following solves the problem:

\*= stub "active\_admin"

# **Bootstrap for Sass using gem**

**Link:**[**https://github.com/twbs/bootstrap-sass**](https://github.com/twbs/bootstrap-sass)

**In your Gemfile you need to add the bootstrap-sass gem, and ensure that the sass-rails gem is present - it is added to new Rails applications by default.**

gem 'bootstrap-sass', '~> 3.3.6'

gem 'sass-rails', '>= 3.2'

**bundle install**

Import Bootstrap styles in app/assets/stylesheets/application.scss:

// "bootstrap-sprockets" must be imported before "bootstrap" and "bootstrap/variables"

@import "bootstrap-sprockets";

@import "bootstrap";

**application/layout**

**<%=form\_tag "/blogs" ,:method=>"GET" do %>**

**<%= text\_field\_tag :search, params[:search] , :placeholder => "Type here to search"%>**

**<%= text\_field\_tag :start\_date, params[:start\_date] , :placeholder => "Start date", :id=>"start-date"%>**

**<%= text\_field\_tag :end\_date, params[:end\_date] , :placeholder => "end\_date" , :id=>"end-date"%>**

**<%= submit\_tag "Search", :class=>"btn btn-success"%>**

**<%end%>**

**layout script for datepicker rails form**

**<script>**

**$( function() {**

**$("#start-date" ).datepicker({ dateFormat: 'dd-mm-yy' });**

**$("#end-date" ).datepicker({ dateFormat: 'dd-mm-yy' });**

**} );**

**</script>**

**index page result found**

**<% if params[:search].present? %>**

**<h3> <%= @blogs.count%> Results found.</h3>**

**<%else%>**

**<h3>no result found<h3>**

**<%end%>**

for a table bootstap

<table class="table table-striped">

<http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-tables.php>

19/4

Blog show page

<p>

<strong>Title:</strong>

<%= @blog.title %>

</p>

<p>

<strong>description:</strong>

<%= @blog.description %>

</p>

<p>

<strong>image:</strong>

<%= image\_tag @blog.image, :height=>"100", :width=>"100" %>

</p>

**Display Comment:**

<% @blog.comments.each do |comment| %>

<p>

<strong>Commenter:</strong>

<%= comment.commenter %>

</p>

<p>

<strong>Comment:</strong>

<%= comment.body%>

</p>

<%= link\_to 'Edit Comment',edit\_blog\_comment\_path(@blog, comment)%>

<%#= link\_to 'Destroy Comment',article\_comment\_path(@article,comment),method: :delete, data: { confirm: 'Are you sure,you want to delete' } %>

<%= link\_to 'Destroy Comment',"/blogs/#{@blog.id}/comments/#{comment.id}",method: :delete, data: { confirm: 'Are you sure,you want to delete' } %>

<%end%>

**Comment form:**

<h2>Add a comment:</h2>

<%= form\_for([@blog, @blog.comments.build]) do |f| %>

<p>

<%= f.label :commenter %><br>

<%= f.text\_field :commenter %>

</p>

<p>

<%= f.label :body %><br>

<%= f.text\_area :body %>

</p>

<p>

<%= f.submit %>

</p>

<% end %>

<%= link\_to 'Back', blogs\_path %>

edit comments view

<h3>Edit tg67hy8nupl.;</h3>

<%= form\_for([@blog, @comment] , :method =>"patch" ) do |f| %>

<%#= form\_for(:comment, url: "/blogs/#{@blog.id}/comments/#{@comment.id}/edit") do |f| %>

<p>

<%= f.label :commenter %><br>

<%= f.text\_field :commenter %>

</p>

<p>

<%= f.label :body %><br>

<%= f.text\_area :body %>

</p>

<p>

<%= f.submit %>

</p>

<% end %>

controller

class CommentsController < ApplicationController

def create

@blog = Blog.find(params[:blog\_id])

@comment = @blog.comments.create(comment\_params)

redirect\_to blog\_path(@blog)

end

def edit

@blog = Blog.find(params[:blog\_id])

@comment = Comment.find(params[:id])

end

def destroy

#@article = Article.find(params[:id])

#@comment = @article.comments.find(params[:article\_id])

@blog = Blog.find(params[:blog\_id])

@comment = Comment.find(params[:id])

@comment.destroy

#redirect\_to article\_path(@article)

redirect\_to @comment

end

def update

@blog = Blog.find(params[:blog\_id])

@comment = Comment.find(params[:id])

@comment.update(comment\_params)

flash[:notice] = "Comment Updated"

redirect\_to blog\_path(@blog)

end

private

def comment\_params

params.require(:comment).permit(:commenter, :body)

end

end

**Working with Rails Console:**

**1. Open Terminal**

**$ User.find(1)**

**$ User.find\_by\_id(1)**

**User.f $ rails c**

**1. Get record by id**

**$ User.find(1)**

**$ User.find\_by\_id(1)**

**User.find\_by\_column\_name(column value)**

**Find will always return single active record object.**

**User.where(:email =>** [**'arvind@bittern.co**](mailto:'arvind@bittern.co)**')**

**User.where(:email =>** [**'arvind@bittern.co**](mailto:'arvind@bittern.co)**', first\_name => “arvind”)**

**@comments = Comment.where('blog\_id = ? ', "19")**

**@comments = Comment.where(:blog\_id => 19)**

**@blogs= Blog.where('title LIKE ? AND description LIKE ?', "test", "%test%")**

**Blog.create!(:title => "test", description: "test")**

**blog = Blog.first**

**blog.title = “New value”**

**blog.save**

**blog.update(:title => "new value")**

**blog.update\_attributes(:title => "new value")**

**Comment.create(:body => "hello", blog\_id: blog.id)**

**has\_many association with forienkey:**

**has\_many :blogs, :foreign\_key => "categry\_id", :class\_name => "Blog"**

**<% @blog.comments.each do |comment| %>**

**<p><%= comment.body %></p>**

**<%end %>**

**render comment**

**<h2>Comments</h2>**

**<%= render @blog.comments%>**

**1. multiple image insert :crate image model with blog id blog model we insert has many relation and put images after that image belongs to blog**

in controller

**params[:blog][:image].each do |image|**

**Image.create(:image=>image,blog\_id:@blog.id)**

**end**

in show

<%@blog.images.each do |image|%>

<%= image\_tag image.image\_url(:thumb) if image.image? %>

<%end%>

destroy image

images = @blog.images

images.destroy\_all

2 comments controller in comments show associations

https://u.osu.edu/hasnan.1/2014/03/30/rails-4-multiple-file-upload-with-carrierwave-nested-form-and-jquery-file-upload/

## nested form (new page)

<%= nested\_form\_for @blog do |f| |f| %>

p>   
  
 <table id="images">   
  <%= f.fields\_for :images, :wrapper => false do |image\_form| %>   
    <tr class="fields">   
      <td><%= image\_form.file\_field :image %></td>   
      <td><%= image\_form.link\_to\_remove "Remove this image" %></td>   
    </tr>   
  <% end %>   
</table>   
<p><%= f.link\_to\_add "Add a image", :images, "data-target" => "#images" %></p>   
 </p>

if params[:blog][:image].present?

<%=#form\_for :blog, url:blogs\_path do |f|%>

if params[:blog][:image].present?

# images = Image.where(:article\_id=>@article.id)

images = @blog.images

images.destroy\_all

params[:blog][:image].each do |image|

Image.create(:image=>image,blog\_id:@blog.id)

end

**AJAX: make request without page loading:**  
**Add remote= true in form or Anchor tag:**  
<%= form\_for([@blog, @blog.comments.build], :remote => true) do |f| %>    
    <%= f.label :body %><br>   
    <%= f.text\_area :body %>      
    <%= f.submit %>   
<% end %>  
  
<a herf=”url” data-remote=true> Hello </a>  
<%= link\_to “hello”, root\_url, :remote => true %>  
  
In Controller: request will be come in AS JS format  
def create   
    @blog = Blog.find(params[:blog\_id])   
    @comment = @blog.comments.create(comment\_params)   
    render 'blogs/create\_comments'   
  end  
  
  
  
  
  
In Views:  
  
app/blogs/create\_comments.js.erb  
$('.comments-container').html('<%= escape\_javascript( render "comments/comments")%>')   
$("#new\_comment")[0].reset();   
$("#new\_comment").hide();  
  
  
In App/views/comments/\_comments.html.erb:  
  
<%@blog.comments.each do |comment|%>    
<p>   
  <strong>Commenter:</strong>   
  <%= comment.commenter %>   
</p>   
    
<p>   
  <strong>Comment:</strong>   
  <%= comment.body %>   
</p>   
    
<p>   
  <%= link\_to 'Destroy Comment', [comment.blog, comment],   
               method: :delete,   
               data: { confirm: 'Are you sure?' } %>   
</p>   
<p>   
  <%= link\_to 'Edit Comment', edit\_blog\_comment\_path(comment.blog, comment) %>   
</p>   
   
<h2> reply a comment</h2>   
<% @replies=Reply.where(:comment\_id=> [comment.id](http://comment.id/))   
%>   
   
<%@replies.each do |reply|%>   
  <p><strong>Reply:</strong>><%=reply.body%></p>   
<%end%>   
   
   
<%= render 'replies/form', :comment=>comment %>   
<%end%>

**AJAX:**

**make request without page loading:**

Add

**remote= true** in form or Anchor tag:

<%=

form\_for([@blog, @blog.comments.build], :remote => true) do |f|

%>

<%= f.label :body %><br>

<%= f.text\_area :body %>

<%= f.submit %>

<%

end %>

<a

herf=”url” data-remote=true> Hello </a>

<%=

link\_to “hello”, root\_url, :remote => true %>

**In**

**Controller:** request will be come in

AS JS format

def

create

@blog = Blog.find(params[:blog\_id])

@comment =

@blog.comments.create(comment\_params)

render 'blogs/create\_comments'

end

mycode

<%@blog.comments.each do |comment|%>

<p>

<strong>Commenter:</strong>

<%= comment.commenter %>

</p>

<p>

<strong>Comment:</strong>

<%= comment.body %>

</p>

<tr>

<p>

<%= link\_to 'Destroy Comment', [comment.blog, comment],

method: :delete,

data: { confirm: 'Are you sure?' } %>

</p>

<p>

<%= link\_to 'Edit Comment',edit\_blog\_comment\_path(id: comment.id,blog\_id: @blog.id) %> </p>

</tr>

<h2> reply a comment</h2>

<% @replies= Reply.where(:comment\_id=> comment.id)%>

<%@replies.each do |reply|%>

<p><strong>Reply:</strong>

<%=reply.body%>

</p>

<%end%>

<%= render 'replies/form', :comment=>comment %>

reply

respond\_to do |format|

format.js { render "blogs/create\_comments"}

format.html {redirect\_to :back}

end

25/04

**1. Create a feature for vote.**

 1.1. Voting will be start a predefined time and will be finished at defined time(After 30 mins)

 1.2 Candidate will be add by Admin.

 1.3 User can give vote only once.

 1.4 Show result of voting.

 1.5 Admin can view all voting details with which user voted to which candidate.

Vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

important link for loading

https://kimmobrunfeldt.github.io/progressbar.js/

<http://jsbin.com/huyusu/3/edit?html,css,js,output>

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Hi Guy's

Create app with following features.

1. User login/signup.

2. Boostrap UI.

3. User can manage our projects.

4. User can mark project as Pending, Available or Completed.

5. User can inactive project.

6. User can upload documents for project.

8. User can comments and replies on Project with files or images.

9. Projects with be associated with category.

10. Filter projects by name or Category.

11. Project Owner can add members into project.

12. Admin Panel.

13. Push on Heroku and Git.

**Note:** user can use Bootstrap UI from here <http://bootsnipp.com/tags>

Regards

Arvind