

# Ruby on Rails - Application

## Inhalt

<b>Ruby Installation auf Ubuntu</b>	3
• Voraussetzung:	3
• Hilfreiches:	3
• Installation:	3
<b>Ein neues Project anlegen</b>	4
1. Projekt erstellen:	4
2. Gem Foundation einbinden:	4
3. Foundation aktivieren:	4
4. Model und Controller anlegen:	4
5. Index-Action im Controller anlegen:	4
6. Index.html.erb im View/Project erstellen:	4
7. Route anlegen (routes.rb):	4
8. Model "Datenbank" Tabelle "Project" mit Inhalt füllen:	5
9. New-Action, Create-Action und new.html.erb in View/Project anlegen!	5
10. Project_controller eintrag:	5
11. new.html.erb ergänzen:	5
12. index.html.erb ergänzen:	5
13. Formular in View/Project anlegen (_form.html.erb):	5
14. Validations eintragen (Im Model/project.rb):	6
15. Im View/Project/_form.html.erb die Fehler anzeigen lassen:	6
16. Error-Anzeige auslagern:	6
17. Neue Datein _errors.html.erb:	6
18. _errors.html.erb in _form.html.erb Rendern:	6
19. Show, Destroy, Update und Edit im Controller Einrichten:	6
20. edit.html.erb im View/Project erstellen:	7
21. show.html.erb im View/Project erstellen:	7
22. Verlinkungen in der View/Project/Index.html.erb einfügen, wenn nicht bereits vorhanden!	7
23. Per Scaffold die verschachtelte und referenzierte Resource anlegen:	7
24. In der routes.rb die Ressourcen verschachteln:	7
25. In der Show von Projects einen Link zu den Tasks hinzufügen:	7
26. Im Tasks Controller eine neue before_action anlegen:	7
27. Index.html.erb von View/Tasks anpassen:	8
28. Im Model/Project.rb anpassen:	8
29. Im Model/Tasks.rb anpassen:	8
30. Im View/Tasks/_form.html.erb anpassen:	8
31. Im View/Tasks/new.html.erb link_to ergänzen:	9
32. Im Controller/tasks_controller.rb create action anpassen:	9
33. Im View/Tasks/show.html.erb link_to ergänzen:	9
34. In View/Tasks/edit.html.erb link_to ergänzen:	9
35. Im controller/tasks_controller.rb update action ergänzen:	9
36. Im controller/tasks_controller.rb destroy und def new ergänzen:	9
37. Und zum Ende wollen wir nun den Task zum jeweiligen Project zuweisen!	9
38. Keine Tasks vorhanden, daher soll man über den Klick auf "Tasks" zum anlegen neuer Tasks gelangen.	10
39. Eingabespalten sollen auf Inhalt bzw. mindesteingabe überprüft werden!	10
<b>User Login</b>	11

1.	User scaffold erstellen: .....	11
2.	Den ersten Admin User anlagen: .....	11
3.	Session controller erstellen:.....	11
4.	Routes.rb ergänzen .....	11
5.	Für die Anmeldung muss ein einzelner Ordner "session" und eine new.html.erb in dem Ordner angelegt werden. ....	11
6.	Signup und Sighin pfad in die routs.rb hinzufügen.....	11
7.	Helper Methoden zur Ermittlung der Userrechte im application controller anlegen.....	12
8.	Admin Checkbox in der User/_form.html.erb anlagen: .....	12
9.	Folgende privaten methoden im usercontroller einfügen .....	12
10.	Folgende beforeactions im usercontroller einfügen .....	12
11.	Folgende Änderungen im usermodel vornehmen.....	13
12.	Folgende beforeactions im events controller hinzufügen .....	13
13.	Anlegen der Sighin und Sighup buttons in der _navigation.html.erb .....	13
14.	b-crypt im gem file aktivieren .....	13
15.	Anpassen der User _form.html.erb.....	13
16.	Anpassen der User show.html.erb .....	13
Registration und Likes .....		14
1.	Rails g scaffold registration how_heard:string user: references event: references .....	14
2.	Änderung in der routes.rb.....	14
3.	Registration controller folgendermaßen anpassen .....	14
4.	Anpassen des User-Models .....	14
5.	Anpassen des Event-Models .....	14
6.	spots_left method anlegen .....	14
7.	Anpassen des Registrations-Models .....	14
8.	How_Heard Mehtode anlegen im Registrations-Model.....	15
9.	Folgende buttons hinzufügen in der event-show.html.erb .....	15
10.	Registrations index.html.erb und new.html.erb anlegen .....	15
11.	Gemeinsam genutzte _errors.html.erb im Ordner /views/shared anlegen .....	16
12.	Registrierungen anzeigen lassen: .....	16
13.	Likes Tabelle erstellen .....	16
14.	Likes model anpassen .....	16
15.	User model anpassen.....	16
16.	Event model anpassen .....	16
17.	Likes controller anpassen .....	16
18.	Show methode in dem User controller anpassen .....	17
19.	Show methode in dem Event controller anpassen.....	17
20.	Likes Darstellung und Buttons in der show.html.erb von Event einbinden .....	17
21.	Likers anzeigen lassen: .....	17
22.	Routes.rb anpassen.....	17
Scopes .....		18
1.	Scopes werden in der Model/event.rb definiert .....	18
2.	In der controller.rb werden die einzelnen Methoden des Scopes erstellt .....	18
3.	In der roots.rb müssen die Pfade für die einzelnen Methoden erstellt werden .....	18
4.	Den Inhalt der Event "index.html.erb" in eine "_event.html.erb" Datei auslagern:.....	18
5.	Folgende weiterleitung in die index.html.erb und die neuen scope.html.erb einbinden.....	18
6.	Dropdown für die Scopes einrichten .....	18
Hilfsbefehle .....		19
1.	SQL/MSQL.....	19
2.	Validations .....	19

## Ruby Installation auf Ubuntu

- **Voraussetzung:**

Die Voraussetzung um mit Ruby on Rails arbeiten zu können ist eine Linux Oberfläche. Diese kann mit VMware oder aufsetzen eines Systems realisiert werden.

- **Hilfreiches:**

Online findet man sehr viele hilfreiche Seiten.

<http://guides.rubyonrails.org/index.html>

<http://www.tutorialspoint.com/index.htm>

<http://railsapps.github.io/>

- **Installation:**

Terminal öffnen.

Terminal ist die Eingabeaufforderung unter Linux!

`gem install rails` löst die Installation von Ruby on Rails aus.

Voraussetzung ist die Anbindung an das Internet!

## Ein neues Project anlegen

### 1. Projekt erstellen:

```
rails new <projektname>
```

### 2. Gem Foundation einbinden:

#### Ordner von der App/Gemfile

```
gem 'foundation-rails', '~> 5.5.3.2'
group :development do
  gem 'rails_layout'
end
```

Optional:

```
gem 'foundation-icons-sass-rails'
```

#### Ausführung

```
"bundle install" zum ausführen
```

### 3. Foundation aktivieren:

```
rails generate layout:install foundation5 --force
```

### 4. Model und Controller anlegen:

```
rails g resource <Name> title:string description:text start_date:datetime
Resource besteht aus Model und Controller (keine Actions im Controller)
```

#### In Datenbank schreiben:

```
rake db:migrate
```

### 5. Index-Action im Controller anlegen:

```
def index
  @projects = Project.all
end
```

### 6. Index.html.erb im View/Project erstellen:

```
<h1>
  <%= if @projects.size == 1 %><%= "Projekt" %><%= else %><%= "Projekte" %><%= end %>
</h1>
<table>
  <thead>
    <tr>
      <th>Title</th>
      <th>Description</th>
      <th>Start Date</th>
      <th>Tasks</th>
      <th></th>
      <th></th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <%= @projects.each do | var | %>
      <tr>
        <td><%= var.title %></td>
        <td><%= var.description %></td>
        <td><%= var.start_date %></td>
        <td><%= link_to "Show", project_path(var.id) %></td>
        <td><%= link_to "Edit", edit_project_path(var.id) %></td>
        <td><%= link_to "Destroy", project_path(var.id), method: :delete, data:
          {confirm: "Sind sie sicher?"} %></td>
      </tr>
    <%= end %>
  </tbody>
</table>
```

### 7. Route anlegen (routes.rb):

```
root "projects#index"
resources :projects
```

**8. Model "Datenbank" Tabelle "Project" mit Inhalt füllen:**

```
Project.create(title: "Test", description: "Test", start_date: "2016-03-03")
Oder
a = Project.new(title: "Test", description: "Test", start_date: "2016-03-03")
a.save
```

## 9. New-Action, Create-Action und new.html.erb in View/Project anlegen!

### 10. Projet\_controller eintrag:

```
before_action :set_project, only: [:show, :edit, :update, :destroy]

def new
  @project = Project.new
end

def create
  @project = Project.new(project_params)
  if @project.save
    flash[:notice] = "Eintrag erfolgreich erstellt :)"
    redirect_to projects_url
  else
    flash[:alert] = "Leider ist etwas schief gegangen :("
  end
end

private

def project_params
  params.require(:project).permit(:title, :description, :start_date)
end

def set_project
  @project = Project.find(params[:id])
end
```

**11. new.html.erb ergänzen:**

```
<h3>Neues Projekt</h3>
<%= render "form" %>
```

**12. index.html.erb ergänzen:**

```
<%= link to "Neues Project anlegen", new project path %>
```

### 13. Formular in View/Project anlegen (\_form.html.erb):

```
<%= form_for(@project) do |f| %>
  <div class="row">
    <div class="medium-3 columns">
      <%= f.label :title %>
    </div>
    <div class="medium-9 columns">
      <%= f.text_field :title %>
    </div>
  </div>
  <div class="row">
    <div class="medium-3 columns">
      <%= f.label :description %>
    </div>
    <div class="medium-9 columns">
      <%= f.text_area :description %>
    </div>
  </div>
  <div class="row">
    <div class="medium-3 columns">
      <%= f.label :start_date %>
    </div>
    <div class="medium-9 columns">
      <%= f.datetime_select :start_date %>
    </div>
  </div>
  <div class="row">
    <div class="medium-3 columns">&nbsp;
    </div>
    <div class="medium-9 columns">
      <%= f.submit "Speichern", class: "button" %>&nbsp;
      <%= link_to "Zurück", projects_path, class:"button" %>
    </div>
  </div>
end %>
```

```

    </div>
  <% end %>

```

**14. Validations eintragen (Im Model/project.rb):**

```

validates :title, length: { minimum: 5, message: "Der Titel muss mindestens 5 Zeichen
enthalten!" }
validate :start_date_cannot_be_in_the_past
  def start_date_cannot_be_in_the_past
    errors.add(:start_date, "Das Datum muss in der Zukunft liegen!")
    if !start_date.blank? && start_date < Date.today
    end
  end

```

Der Befehl "validate" ruft die Methode auf die hinter ihm steht, in diesem Falle ":start\_date\_cannot\_be\_in\_the\_past"

**15. Im View/Project/\_form.html.erb die Fehler anzeigen lassen:**

```

<% if @project.errors.any? %>
  <h3>Oops! Der <%= @project.class.name.titleize.downcase %> konnte nicht
    gespeichert werden.</h3>
  Korrigieren sie folgenden <%= pluralize(@project.errors.count, "error") %>:
  <ul>
    <% @project.errors.full_messages.each do |message| %>
    <li><%= message %></li>
    <% end %>
  </ul>
<% end %>
<div class=.....

```

**16. Error-Anzeige auslagern:**

Neuen Ordner in den Views -> "shared"

**17. Neue Datei \_errors.html.erb:**

```

<% if object.errors.any? %>
<h3>Oops! Der <%= object.class.name.titleize.downcase %> konnte nicht gespeichert
werden.</h3>
  Korrigieren sie folgenden <%= pluralize(object.errors.count, "error") %>:
  <ul>
    <% object.errors.full_messages.each do |message| %>
    <li><%= message %></li>
    <% end %>
  </ul>
<% end %>

```

**18. \_errors.html.erb in \_form.html.erb Rendern:**

```

...form_for(@project) do |f| %>
  <%= render "shared/errors", object: @project %>
  <div class=.....

```

**19. Show, Destroy, Update und Edit im Controller Einrichten:**

```

def edit
end

def update
  if @project.update(project_params)
    flash[:notice] = "Update erfolgreich :)"
    redirect_to projects_url
  else
    flash[:alert] = "Leider ist etwas schief gegangen :("
    render :edit
  end
end

def destroy
  @project.destroy
  redirect_to projects_url
end

def show
end

```

**20. edit.html.erb im View/Project erstellen:**

```
<h2>Editieren von <%= @project.title %></h2>
<%= render "form" %>
```

**21. show.html.erb im View/Project erstellen:**

```
<h2>Editieren von <%= @project.title %></h2>
<%= render "form" %>

show.html.erb im View/Project erstellen:
<h1><%= @project.title %></h1>

<p><%= @project.description %></p>

<p><%= @project.start_date %></p>

<p>
  <ul class="button-group">
    <li>
      <%= link_to "Alle Projekte anzeigen", projects_path, class: "button" %>
    </li>
  </ul>
</p>
```

**22. Verlinkungen in der View/Project/Index.html.erb einfügen, wenn nicht bereits vorhanden!**

```
<% @projects.each do | var | %>
  <tr>
    <td><%= var.title %></td>
    <td><%= var.description %></td>
    <td><%= var.start_date %></td>
    <td></td>
    <td><%= link_to "Show", project_path(var.id) %></td>
    <td><%= link_to "Edit", edit_project_path(var.id) %></td>
    <td><%= link_to "Destroy", project_path(var.id), method: :delete, data: {confirm:
      "Sind sie sicher?" } %></td>
  </tr>
<% end %>
```

**23. Per Scaffold die verschachtelte und referenzierte Resource anlegen:**

```
rails g scaffold tasks title description:text start_date end_date project:references
rake db:migrate
```

**24. In der routes.rb die Ressourcen verschachteln:**

```
resources :projects do
  resources :tasks
end
```

**25. In der Show von Projects einen Link zu den Tasks hinzufügen:**

```
<%= link_to "Tasks", project_tasks_path(@project.id), class: "button" %>
```

**26. Im Tasks Controller eine neue before\_action anlegen:**

```
before_action :set_project

private
  def set_project
    @project = Project.find(params[:project_id])
  end
```

**27. Index.html.erb von View/Tasks anpassen:**

```

<p id="notice"><%= notice %></p>

<h1>Listing Tasks for <%= @project.title %></h1>

<table>
  <thead>
    <tr>
      <th>Title</th>
      <th>Description</th>
      <th>Start date</th>
      <th>End date</th>
      <th colspan="3"></th>
    </tr>
  </thead>

  <tbody>
    <% @tasks.each do |task| %>
      <tr>
        <td><%= task.title %></td>
        <td><%= task.description %></td>
        <td><%= task.start_date %></td>
        <td><%= task.end_date %></td>
        <td><%= link_to 'Show', project_task_path(@project.id, task.id) %></td>
        <td><%= link_to 'Edit', edit_project_task_path(@project.id, task.id) %></td>
        <td><%= link_to 'Destroy', project_task_path(@project.id, task.id),
          method: :delete, data: { confirm: 'Are you sure?' } %></td>
      </tr>
    <% end %>
  </tbody>
</table>

<br>

<%= link_to 'New Task', new_project_task_path(@project.id) %>

```

**28. Im Model/Project.rb anpassen:**

```
has_many :tasks, dependent: :destroy
```

**29. Im Model/Tasks.rb anpassen:**

```
belongs_to :project
```

**30. Im View/Tasks/\_form.html.erb anpassen:**

```

<%= form_for [@project, @task] do |f| %>
  <% if @task.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@task.errors.count, "error") %> prohibited this task
        from being saved:</h2>

      <ul>
        <% @task.errors.full_messages.each do |message| %>
          <li><%= message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :title %><br>
    <%= f.text_field :title %>
  </div>

  <div class="field">
    <%= f.label :description %><br>
    <%= f.text_area :description %>
  </div>

  <div class="field">
    <%= f.label :start_date %><br>

```



```

<%= f.datetime_select :start_date %>
</div>
<div class="field">
<%= f.label :end_date %><br>
<%= f.datetime_select :end_date %>
</div>
<div class="actions">
<%= f.submit %>
</div>
<% end %>

```

**31. Im View/Tasks/new.html.erb link\_to ergänzen:**

```

<%= link_to 'Back', project_tasks_path %>

```

**32. Im Controller/tasks\_controller.rb create action anpassen:**

```

def create
  @task = @project.tasks.new(task_params)

  respond_to do |format|
    if @task.save
      format.html { redirect_to project_tasks_path(@project.id), notice: 'Task
        was successfully created.' }
      format.json { render :show, status: :created, location: @task }
    else
      format.html { render :new }
      format.json { render json: @task.errors, status: :unprocessable_entity }
    end
  end
end
end

```

**33. Im View/Tasks/show.html.erb link\_to ergänzen:**

```

<%= link_to 'Edit', edit_project_task_path(@project.id, @task.id) %> |
<%= link_to 'Back', project_tasks_path %>

```

**34. In View/Tasks/edit.html.erb link\_to ergänzen:**

```

<%= link_to 'Show', project_task_path(@project.id, @task.id) %> |
<%= link_to 'Back', project_task_s_path(@project.id) %>

```

**35. Im controller/tasks\_controller.rb update action ergänzen:**

```

def update
  respond_to do |format|
    if @task.update(task_params)
      format.html { redirect_to project_task_path, notice: 'Task was
        successfully updated.' }
      format.json { render :show, status: :ok, location: @task }
    else
      format.html { render :edit }
      format.json { render json: @task.errors, status: :unprocessable_entity }
    end
  end
end
end

```

**36. Im controller/tasks\_controller.rb destroy und def new ergänzen:**

```

def new
  @task = @project.tasks.new
end

def destroy
  @task.destroy
  respond_to do |format|
    format.html { redirect_to project_tasks_url, notice: 'Task was successfully
      destroyed.' }
    format.json { head :no_content }
  end
end
end

```

**37. Und zum Ende wollen wir nun den Task zum jeweiligen Project zuweisen!****Im controller/tasks\_controller.rb index action ergänzen:**

```

def index
  @tasks = @project.tasks
end

```

**38. Keine Tasks vorhanden, daher soll man über den Klick auf “Tasks” zum anlegen neuer Tasks gelangen.**

**In der tasks\_controller.rb def index erweitern:**

```
def index
  @tasks = @project.tasks
  if @tasks.blank?
    redirect_to new_project_task_path(@project.id)
  end
end
```

**39. Eingabespalten sollen auf Inhalt bzw. mindesteingabe überprüft werden!**

**In dem Ordner models/project.rb oder task.rb die validates einrichten:**

```
....class Validation < ActiveRecord::Base
  belongs_to :project

  validates :title, :price, :start_date, :end_date, presence: true
  validates :description, length: {minimum: 15 }
end
```

## User Login

### 1. User scaffold erstellen:

```
rails g scaffold user name:string email:string password_digest:string  
admin:Boolean  
rake db:migrate
```

### 2. Den ersten Admin User anlagen:

```
Console aufrufen  
rails c  
  
u = User.new(name: "admin", email: "admin@admin.de", password: "admin", admin: true)  
u.save
```

### 3. Session controller erstellen:

```
rails g controller sessions (MIT SSSSSSSSSSSS)  
controller inhalt  
    def new  
    end  
  
    def create  
      if user = User.authenticate(params[:email], params[:password])  
        session[:user_id] = user.id  
        flash[:notice] = "Welcome back, #{user.name}!"  
        redirect_to session[:intended_url] || root_url  
        session[:intended_url] = nil  
      else  
        flash.now[:alert] = "Invalid email/password combination!"  
        render :new  
      end  
    end  
  
    def destroy  
      session[:user_id] = nil  
      redirect_to root_url, notice: "You're now signed out!"  
    end
```

### 4. Routes.rb ergänzen

```
resource :session
```

### 5. Für die Anmeldung muss ein einzelner Ordner "session" und eine new.html.erb in dem Ordner angelegt werden.

```
<%= form_tag(session_path) do %>  
  <p>  
    <%= label_tag :email %>  
    <%= email_field_tag :email, nil, autofocus: true %>  
  </p>  
  <p>  
    <%= label_tag :password %>  
    <%= password_field_tag :password %>  
  </p>  
  <p>  
    <%= submit_tag 'Sign In' %>  
  </p>  
<% end %>
```

### 6. Signup und Signin pfad in die routs.rb hinzufügen

```
get 'signup' => 'users#new'  
get 'signin' => 'sessions#new'
```

## 7. Helper Methoden zur Ermittlung der Userrechte im application controller anlegen

```
def current_user
  @current_user ||= User.find(session[:user_id]) if session[:user_id]
end

helper_method :current_user
#-----
def require_signin
  unless current_user
    session[:intended_url] = request.url
    redirect_to new_session_url, alert: "Please sign in first!"
  end
end
#-----
def require_admin
  unless current_user_admin?
    redirect_to root_url, alert: "Unauthorized access!"
  end
end

def current_user_admin?
  current_user && current_user.admin?
end

helper_method :current_user_admin?
```

## 8. Admin Checkbox in der User/\_form.html.erb anlagen:

```
<% if current_user_admin? %>
  <div class="field">
    <%= f.label :admin %>
    <%= f.check_box :admin %>
  </div>
<%end%>
```

## 9. Folgende privaten methoden im usercontroller einfügen

```
private
  def require_correct_user
    @user = User.find(params[:id])
    unless current_user == @user
      redirect_to root_url
    end
  end

  # Use callbacks to share common setup or constraints between actions.
  def set_user
    @user = User.find(params[:id])
  end

  # Never trust parameters from the scary internet, only allow the white list
through.
  def user_params
    params.require(:user).permit(:name, :email, :password,
:password_confirmation)
  end
end
```

## 10. Folgende beforeactions im usercontroller einfügen

```
before_action :set_user, only: [:show, :edit, :update, :destroy]
before_action :require_signin, except: [:new, :create]
before_action :require_correct_user, only: [:edit, :update, :destroy]
```

**11. Folgende Änderungen im usermodel vornehmen**

```
has_secure_password

  validates :name, presence: true

  validates :email, presence: true,
    format: /\A\S+@\S+\z/,
    uniqueness: { case_sensitive: false }

  def self.authenticate(email, password)
    user = User.find_by(email: email)
    user && user.authenticate(password)
  end
```

**12. Folgende beforeactions im events controller hinzufügen**

```
before_action :require_signin, except: [:index]
before_action :require_admin, only: [:destroy, :edit, :update]
```

**13. Anlegen der Sighin und Sighup buttons in der \_navigation.html.erb**

```
<ul class="right">
  <% if current_user %>
    <li><%= link_to current_user.name, user_path(current_user.id) %></li>
    <li><%= link_to "Log Out", session_path, method: :delete %></li>
  <%else%>
    <li><%=link_to "Log In", signin_path %></li>&nbsp;
    <li><%=link_to "SignUp", signup_path %></li>
  <%end%>
</ul>
```

**14. b-crypt im gem file aktivieren**

```
gem 'bcrypt', '~> 3.1.7'
```

**15. Anpassen der User \_form.html.erb**

```
<div class="field">
  <%= f.label :password %><br>
  <%= f.text_field :password %>
</div>

<div class="field">
  <%= f.label :password_confirmation %>
  <%= f.password_field :password_confirmation %>
</div>
```

**austauschen des Password\_digest****16. Anpassen der User show.html.erb**

```
<table>
  <tr>
    <td>
      <p>
        <strong>Name:</strong>
        <%= @user.name %>
      </p>
      <p>
        <strong>Email:</strong>
        <%= @user.email %>
      </p>
      <% if current_user == @user %>
        <%= link_to 'Edit', edit_user_path(@user) %> &nbsp;
        <%= link_to 'Destroy', @user, method: :delete, data: {
confirm: 'Are you sure?' } %>
      <%end%>
      &nbsp;
      <%= link_to 'Back', users_path %>
    </td>
  </tr>
</table>
```

## Registration und Likes

### 1. Rails g scaffold registration how\_heard:string user: references event: references

```
rake db:migrate
```

### 2. Änderung in der routes.rb

```
resources :events do
  resources :registrations
end
```

### 3. Registration controller folgendermaßen anpassen

```
before_action :set_event
before_action :require_signin, except: [:new, :create]
def index
  @registrations = @event.registrations
end

def new
  @registration = @event.registrations.new
end

def create
  @registration = @event.registrations.new(registration_params)
  @registration.user_id = current_user.id
  if @registration.save
    redirect_to event_path(@event.id), notice: "Die Registrierung
wurde gespeichert"
  else
    render :new, notice: "Fehler beim Speichern"
  end
end

def destroy
  @registration = Registration.find(params[:id])
  @registration.destroy
  redirect_to event_registrations_url(@event.id), notice: "Registrierung
von #{@registration.user.name} erfolgreich gelöscht"
end

private
def registration_params
  params.require(:registration).permit(:how_heard, :event_id,
:user_id)
end
def set_event
  @event = Event.find(params[:event_id])
end
end
```

### 4. Anpassen des User-Models

```
has_many :registrations
```

### 5. Anpassen des Event-Models

```
has_many :registrations, dependent: :destroy
```

### 6. spots\_left method anlegen

```
def spots_left
  if capacity.zero?
    0
  else
    capacity - registrations.size
  end
end
```

### 7. Anpassen des Registrations-Models

```
belongs_to :event
belongs_to :user
```

## 8. How\_Heard Methode anlegen im Registrations-Model

```
HOW_HEARD_OPTIONS = ['Newsletter', 'Blog Post', 'Twitter', 'Web Search', 'Other']
validates :how_heard, inclusion: {in: HOW_HEARD_OPTIONS}
#validates :email, format: {with: /(\S+)@(\S+)/}
#validates :name, presence: true
```

## 9. Folgende buttons hinzufügen in der event-show.html.erb

```
<% if @event.spots_left > 0 %>
  <%= link_to "Registrieren?", new_event_registration_path(@event.id), class:
"button tiny" %>
<%end%>
```

## 10. Registrations index.html.erb und new.html.erb anlegen

### Index

```
<h3><%= @event.name %></h3>
<p><%= "#{pluralize(@event.registrations.size, "Registrierung", "Registrierungen")}" %>
für <%= @event.name %></p>
<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>HowHeard</th>
      <% if current_user_admin? %>
      <th>Delete</th>
      <%end%>
    </tr>
  </thead>
  <tbody>
    <% @registrations.each do |registration| %>
    <tr>
      <td><%= registration.user.name %></td>
      <td><%= registration.user.email %></td>
      <td><%= registration.how_heard %></td>
      <td>
        <%= link_to "<i class='fi-trash icon-red'></i>".html_safe,
event_registration_path(@event.id, registration.id), method: :delete, data: {confirm:
"Möchten Sie wirklich löschen"} if current_user_admin? %></td>
      </tr>
    <% end %>
  </tbody>
</table>
```

### NEW

```
<h3>Neue Registrierterung für Event: <%= @event.name %></h3>
<%= render "shared/errors", object: @registration %>
<%= form_for [@event, @registration] do |f| %>
  <%# Select how_heard %>
  <div class="row">
    <div class="medium-3 columns">
      <%= f.label :how_heard %>
    </div>
    <div class="medium-9 columns">
      <%= f.select :how_heard, Registration::HOW_HEARD_OPTIONS, prompt:
"Bitte wählen" %>
    </div>
  </div>
  <%# Submit Button %>
  <div class="row">
    <div class="medium-12 columns">
      <%= f.submit %>&nbsp;<%= link_to 'Back',
event_registrations_path(@event.id) %>
    </div>
  </div>
<% end %>
```

**11. Gemeinsam genutzte \_errors.html.erb im Ordner /views/shared anlegen**

```
<% if object.errors.any? %>
  <h3>Oops! Der <%= object.class.name.titleize.downcase %> konnte nicht gespeichert
werden.</h3>
  Korrigieren Sie die folgenden <%= pluralize(object.errors.count, "error")
%>:
  <ul>
    <% object.errors.full_messages.each do |message| %>
      <li><%= message %></li>
    <% end %>
  </ul>
<% end %>
```

**\_errors.html.erb ist ein PARTIAL => führenden Unterstrich nicht vergessen!****12. Registrierungen anzeigen lassen:**

```
<td>
  <% if @user.registrations.any? %>
    <h4>Registriert für:</h4>
    <ul>
      <% @user.registrations.each do |registration| %>
        <li><%= link_to registration.event.name, registration.event %>
        </li>
      <%end%>
    </ul>
  <%end%>
</td>
```

**13. Likes Tabelle erstellen**

```
rails g resource like user:references event:references
rake db:migrate
```

**14. Likes model anpassen**

```
belongs_to :event
belongs_to :user
```

**15. User model anpassen**

```
has_many :likes, dependent: :destroy
has_many :liked_events, through: :likes, source: :event
```

**16. Event model anpassen**

```
has_many :likes, dependent: :destroy
has_many :likers, through: :likes, source: :user
```

**17. Likes controller anpassen**

```
before_action :require_signin
before_action :set_event
def create
  #@event.likes.create(user: current_user)
  @event.likers << current_user
  @current_like = current_user.likes.find_by(event_id: @event.id)
  respond_to do |format|
    format.html {redirect_to @event, notice: "Super, kannst dir eine
anstecken"}
    format.js
  end
end
def destroy
  @like = current_user.likes.find(params[:id])
  @like.destroy
  respond_to do |format|
    format.html {redirect_to @event, notice: "Gut gemacht. war ätzend
dort"}
    format.js
  end
end
private
def set_event
  @event = Event.find(params[:event_id])
end
```



**18. Show methode in dem User controller anpassen**

```
def show
  @liked_events = @user.liked_events
end
```

**19. Show methode in dem Event controller anpassen**

```
def show
  @likers = @event.likers
  if current_user
    @current_like = current_user.likes.find_by(event_id: @event.id)
  end
end
```

**20. Likes Darstellung und Buttons in der show.html.erb von Event einbinden**

```
<h3>Likers</h3>
<% if @likers.any? %>
  <ul>
    <% @likers.each do |user| %>
      <li><%= link_to user.name, user %></li>
    <% end %>
  </ul>
<% else %>
  <p>Sei der erste, dem das Gefällt, du Penner.</p>
<% end %>
<% if @current_like %>
  <%= button_to "Unlike", event_like_path(@event, @current_like), class: "button",
method: :delete %>
<% else %>
  <%= button_to "Like", event_likes_path(@event), class: "button" %>
<% end %>
```

**21. Likers anzeigen lassen:**

```
<td>
  <% if @liked_events.any? %>
    <h4>Liked Events:</h4>
    <ul>
      <% @liked_events.each do |event| %>
        <li><%= link_to event.name, event %></li>
      <%end%>
    </ul>
  <%end%>
</td>
```

**22. Routes.rb anpassen**

```
resources :sites do
  resources :registrations
  resources :likes
end
```

## Scopes

### 1. Scopes werden in der Model/event.rb definiert

```
Beispiel:      Model/event.rb

scope :free, ->{where("price <= 0").order(:name)}
scope :past, ->{where("date <= ?", Time.now).order(:name)}
scope :future, ->{where("date >= ?", Time.now).order(:name)}
```

### 2. In der controller.rb werden die einzelnen Methoden des Scopes erstellt

```
def past
  @events = Event.past
end

def free
  @events = Event.free
end

def future
  @events = Event.future
end
```

### 3. In der roots.rb müssen die Pfade für die einzelnen Methoden erstellt werden

```
resources :events do
  collection do #-----Bezogen auf den Scope Filter-----
    get 'past' #route zu event_controller "past"
    get 'free' #route zu event_controller "free"
    get 'future' #route zu event_controller "future"
  end

  resources :registrations
  resources :likes
end
```

### 4. Den Inhalt der Event "index.html.erb" in eine "\_event.html.erb" Datei auslagern:

Es werden für jede scope Methode eine .html.erb Datei erstellt.

```
past.html.erb
free.html.erb
future.html.erb
```

### 5. Folgende weiterleitung in die index.html.erb und die neuen scope.html.erb einbinden.

```
<%= render 'events' %>
```

### 6. Dropdown für die Scopes einrichten.

Beispiel für \_navigation.html.erb

```
.....<div class="top-bar-section">
  <ul>
    <li><%=link_to "Alle Events", events_path %></li>

    <li class="has-dropdown">
      <a href="#">Events gefiltert anzeigen</a>
      <ul class="dropdown">
        <ul><%= link_to "Alte Events", past_events_path %></ul>
        <ul><%= link_to "Kommende Events", future_events_path %></ul>
        <ul><%= link_to "Kostenlose Events", free_events_path %></ul>

      </ul>

    </li>
  </ul>
  <ul class="right">.....
```

Diese Dropdown Auswahl kann ebenso in der Event index oder show stehen, muss dann entsprechend angepasst werden!!!

# Hilfsbefehle

## 1. SQL/MSQL

```
rails generate migration AddColumnToProducts column:type
```

### Erzeugte Migration

```
class addColumnToProducts < ActiveRecord::Migration
  def change
    add_column :products, :column, :type
  end
end
```

### Nach Prüfung der neuen Migration in die Datenbank übertragen

```
rake db:migrate
```

### Column types

```
user:references
  :primary_key,
  :string,
  :text,
  :integer,
  :float,
  :decimal,
  :datetime,
  :timestamp,
  :time,
  :date,
  :binary,
  :boolean
```

## 2. Validations