Ruby on Rails - Application

Inhalt

Ruby Installation auf Ubuntu

- **Voraussetzung:**

  Die Vorraussetzung um mit Ruby on Rails arbeiten zu könne 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 :("render :new
		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"> 
		</div>
		<div class="medium-9 columns">
			<%= f.submit "Speichern", class: "button" %> 
			<%= link_to "Zurück", projects_path, class:"button" %>
		</div>
```

```
        </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
```

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>Opps! 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 Datein _errors.html.erb:

```
<% if object.errors.any? %>
<h3>Opps! 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 Resourcen 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
```

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
```

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
```

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 SSSSSSSSSSS)
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 Sighin 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
```

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> 
            <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) %>  
                        <%= link_to 'Destroy', @user, method: :delete, data: {
confirm: 'Are you sure?' } %>
                  <%end%>
                   
                  <%= 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
```

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 Mehtode 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 Regiertrierung 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 %> <%= 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>Opps! Der <%= object.class.name.titleize.downcase %> konnte nicht gespeichter
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 Pfads 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!!!

Dateiupload

http://www.rubydoc.info/gems/carrierwave/0.11.2

7. rmagick installieren

console:

```
sudo apt-get update
sudo apt-get install libmagickwand-dev libmagickcore-dev imagemagick
```

8. Änderungen in der gemfile

```
gem 'carrierwave'
gem 'rmagick'
gem 'mini_magick'
```

console:

```
bundle install
```

9. Neues uploader Verzeichnis im Projekt erstellen

console:

```
rails g uploader Avatar
```

nun wurde neuer Ordner "uploaders" und Datei "avatar_uploader.rb" erstellt

10. Änderung in der event.rb

In der schema.rb ist das file als image_file bezeichnet.

Hieraus ergibt sich dann die Änderung in der event.rb:

```
class Event < ActiveRecord::Base

        mount_uploader :image_file, AvatarUploader

...
end
```

11. Als nächstes wird die Möglichkeit geschaffen, Dateien für den Upload zu bestimmen und hochzuladen.

Änderung in der _form.html.erb von /views/events:

```
...
        <p>
        <%= f.label :image_file %><br />
        <%= f.file_field :image_file, size: 30, placeholder: "GIF,
        JPG, PNG als Dateityp" %>
        </p>
...
```

Aus f.text_field wurde f.file_field

12. Bildgröße ändern (es werden tatsächlich weitere Files generiert)

```
...
        # Include RMagick or MiniMagick support:
        include CarrierWave::RMagick
        # include CarrierWave::RMagick
...
```

13. Im avatar_uploader.rb:

```
...
    version :large do
        process :resize_to_limit => [600,600]
    end
    version :thumb do
        resize_to_fill(150,150)
    end
    version :standard do
        resize_to_fill(300,300)
    end
...
```

14. Änderungen in der _events.html.erb um die Thumbnails bereits in der Tabelle anzuzeigen:

```
...
    <thead>
        <tr>
        <th>NAME</th>
-->--->  <th></th>
        <th>Beschreibung</th>
        <th>Datum</th>
        <th>Location</th>
        <th>Price</th>
        <th>Registrierungen</th>
        <th>Anzeigen</th>
        <% if current_user_admin? %>
            <th>Editieren</th>
            <th>Löschen</th>
        <% end %>
        </tr>
    </thead>
    <tbody>
        <% @events.each do |event| %>
        <tr>
        <td><%= event.name %></td>
-->--->  <td><%= image_tag event.image_file.url(:thumb) %></td>
        <td><%= event.description %></td>
...
```

Befehlssammlung

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 Datanbank übertragen

```
rake db:migrate
```

db/seeds.rb

```
Examples:
cities = City.create([{ name: 'Chicago' }, { name: 'Copenhagen' }])
Mayor.create(name: 'Emanuel', city: cities.first)

Country.create({"name"=>"Deutschland", "population"=>81831000})
```

```
rake db:seed
Load the seed data from db/seeds.rb
```

```
rake db:setup
Create the database, load the schema, and initialize with the seed data
```

```
rake db:reset
Same as rake db:setup, but drop the database first
```

Column types

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

2. Validations

```
validates :name,  presence: true, length: { maximum: 50 }

validates :password, presence: true, length: { minimum: 6 }

validates_format_of :email
   :with => /^(+)@((?:[-a-z0-9]+\.)+[a-z]{2,})$/i

validates_numericality_of   :value
                            :only_integer => true
                            :allow_nil => true

validates_inclusion_of  :gender,
                        :in => %w( m, f )

validates_exclusion_of  :age
                        :in => 13..19
```

```
validates_acceptance_of :option
                        :accept => 'Y'
```

3. **Rails**

```
SELECT
<%= select("Contact", "email_provider", Contact::PROVIDERS, {:include_blank => true}) %>

<% cities_array = City.all.map { |city| [city.name, city.id] } %>
<%= options_for_select(cities_array) %>
```

```
f.collection_select :event_id, @events, :id, :name
```

```
RADIO BUTTON
radio_button("post", "category", "rails")
```

4. **DateTime Formats**

Code	Output	Description
`t.strftime("%H")`	=> "22"	# Gives **Hour** of the time **in 24 hour** clock format
`t.strftime("%I")`	=> "10"	# Gives **Hour** of the time **in 12 hour** clock format
`t.strftime("%M")`	=> "49"	# Gives **Minutes** of the time
`t.strftime("%S")`	=> "27"	# Gives **Seconds** of the time
`t.strftime("%Y")`	=> "2013"	# Gives **Year** of the time
`t.strftime("%m")`	=> "09"	# Gives **month** of the time
`t.strftime("%d")`	=> "12"	# Gives **day** of month of the time
`t.strftime("%w")`	=> "4"	# Gives **day of week** of the time
`t.strftime("%a")`	=> "Thu"	# Gives name of **week day** in **short** form of the
`t.strftime("%A")`	=> "Thursday"	# Gives **week day** in **full** form of the time
`t.strftime("%b")`	=> "Sep"	# Gives **month** in **short** form of the time
`t.strftime("%B")`	=> "September"	# Gives **month** in **full** form of the time
`t.strftime("%y")`	=> "13"	# Gives **year without century** of the time
`t.strftime("%Y")`	=> "2013"	# Gives **year without century** of the time
`t.strftime("%Z")`	=> "IST"	# Gives **Time Zone** of the time
`t.strftime("%p")`	=> "PM"	# Gives **AM / PM** of the time

These are the almost all formats that are required.

Anwendung:

You can try these formats in combination too, For example,

`t.strftime("%H:%M:%S") > "22:49:27"`

5. Strings zusammenführen

```
"peter" + " "+"müller"
```

wird zu

"peter müller"

```
def name
firstname+" "+lastname
end
```

Für User Peter Müller kommt ebenfalls Peter Müller raus