BAZE i MODELI

$ rake db:create - pravi bazu (na osnovu config/database.yml podataka)

$ rails generate migration → stvara tablicu

$ rails generate model User (SINGULAR, CAMEL CASE) → automatski radi i migraciju

↳ APP/MODELS

↳ DB/MIGRATE

↳ EDITIRAT FILE OVOJE PR:

t.string "first_name", :limit => 25

ili

rails generate model User name:string email:string

- RUN-anje migracije

$ rake db:migrate (VERSION = x) ($ rake db:seed)

1. GENERIRAT MODELE (JEDNINA) (CAMEL-CASE)
2. NAPISAT MIGRACIJE - FOREIGN KEYS - INDEX
3. RUN-at MIGRACIJE

MIGRATION METODE ("admin_users", "usern

add_column(table, column, type, options)

remove_column(table, column)

rename_column(table, column, new_name)

change_column(table, column, type, opti

add_index(table, column, options) → migrat na foreign keyove

- DA BI SPRIJEČIO MASS ASSIGNMEN VULNERABILITY U:

config/application.rb odkomentirat

↳ config.active_record.whitelist_attributes = true

- SAD SVAKI MODEL MORA EKSPLICITNO IMAT attr_accessible :property za ONE PROPERTYE KOJIMA JE DOZVOLJEN MASS ASSIGNMEN / update_attributes METODA

SUSTJEPANSKA 4

Model & relations

MODEL NEW/SAVE TEHNIKA → OVO KORISTI

HASH

SUBJECT = Subject.new(:name => "Ime", :pozicija => 1, :visible => true)

subject.save → TRUE / FALSE

CREATE TEHNIKA – automatski sprema

HASH

subject = Subject.create(:....)

UPDATE

ID

subject = Subject.find(1)

HASH

subject.update_attributes(...) → TRUE / FALSE

DELETE

subject = Subject.find(x)

subject.destroy

FIND

Subject.find_by_id(x) ili Subject.find_by_name("Ime")

Subject.all

subject.first ili Subject.last

QUERY

Subject.where(:visible => true).order("position ASC")

MOŽE SE CHAINATI

VRAĆA ARRAY

ORDER, LIMIT, OFFSET

- order ("sql")
- limit (integer)
- offset (integer)

Subject.order("position ASC").limit(20).offset(40)

table_name.column_name ASC/DESC

"subjects.name DESC"

"subjects.visible DESC, subjects.name ASC"

NAMED SCOPE → U MODELU SU STOREANI

- SCOPE :IME, WHERE(...)
- subject = Subject.ime
- .

SEARCH

- scope :search, lambda {|query| where([

"name LIKE ?", "%#{query}%"])}

ili

? scope :sorted, order("last_name ASC", "first_name ASC")

RELACIJE U TABLICAMA

- KLASA sa "belongs_to" ima foreign key
- mijenja definiranje obje strane relacije

↳ u MODELIMA

belongs_to :subject

ONE_TO_MANY

↳ VRAĆA ARRAY OBJEKATA (UMJESTO 1. OBJEKTA)

has_many :pages (PLURAL)

→ subject.pages → sve stranice (array)

- subject.pages << page (dodavanje stranice)
- subject.pages.delete(page)
- subject.pages.clear → čisti sve stranice
- subject.pages.empty?
- subject.pages.size → koliko je stranica?

MANY_TO_MANY - LYNDA & VID MANY_TO_MANY

→ POTREBAN - JOIN TABLE
- JOIN TABLE NEMA PRIMARY KEY (:id ⇒ false)

Page has_and_belongs_to_many :admin_users → obje strane staviti vezu

JOIN TABLE IME
↳ OBJE TABLICE U PLURALU
↳ ALFABETSKI RED

ODIS JOIN TABLICE

→ FILE SIMPLE_CMS
↳ DB
↳ MIGRATE
↳ CREATE ADMIN USERS...

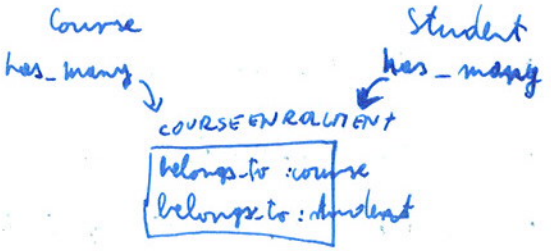
$rails generate migration CreateAdminUsersPagesJoin
NE MODEL

RICH ~~MANY_TO_MANY~~
~~- SIMPLE_CMS → SECTION EDIT MODEL i MIGRACIJA~~

- u migraciji
  - ↳ :id ⇒ false
  - ↳ i samo foreign keyeri
  - ↳ bez timestampova
  - ↳ add_index :admin_users_pages, ["admin_user_id", "page_id"]
    - OVAKAV INDEX TREBA DODAT ZAJEDNO
    - NIJE GREŠKA
- u modelu:

has_and_belongs_to_many :editors, :class_name ⇒ "AdminUser"
- DRUKČIJE IME KOJE JE ZAPRAVO OVO

MANY TO MANY RICH



- ime join tablice zovemo kako hoćemo
- join tablica ima:
  - ↳ :id ⇒ ima ga
  - ↳ dva foreign keya
  - ↳ ima svoj MODEL

(1) napravit model za join table
  - ↳ MIGRACIJA
    - ↳ dodat foreign keyeve
    - ↳ dodat "drugačiji" "index": add_index :section_edits, ['admin_user_id', 'section_id']
    - ↳ ima timestamps

(2) MODEL:
- → u "glavne tablice dodat" JOIN TABLE
  - ↳ has_many : section_edits
  - ↳ has_many : sections, :through ⇒ : section_edits
    - ↑ DRUGA GLAVNA TABLICA SE SAD MOŽE "DOHVATIT" NPR. page.sections
    - JOIN TABLE
- → u join tablicu dodat
  - 2x → belongs_to : editor, : class_name ⇒ "Admin User", : foreign_key ⇒ 'admin_user_id'
    - DRUGO IME
    - PRAVA KLASA
    - TREBA DODAT KOJI JE ID
  - belongs_to : section

Controller & crud

CONTROLLER & CRUD

- TIPIČNI CRUD
  - ↳ jedan controller po modelu
  - ↳ ime u množini (SubjectsController)

$ rails generate controller Subjects (PLURAL) new show ...
↑ ↑ actions u kontrolleru

CRUD
- CREATE: NEW, CREATE
- READ: LIST, SHOW
- UPDATE: EDIT, UPDATE
- DELETE: DELETE, DESTROY

} AKCIJE

READ — LIST

- u controller dodajemo ono što dohvaćamo iz modela + SPREMAMO U INSTANCE VAR. @varijabla (PLURAL)

```
def list
  @subjects = Subject.order('subjects.position ASC')
end
```

- u view dohvaćamo @instance var. => list.html.erb

```
<%= link_to("Show", {:action => 'show', :id => subject.id}, :class => 'action show') %>
```

CRUD SHOW

SHOW

- za ovo treba unijeti objekt iniet :id
- u controlleru

```
def show
  @subject = Subject.find(params[:id])
end
```

- u viewu show.html.erb → popuniti ovisno od @subject

GOOD TODO ZA READ

```
def index          → AKO SE NE UPIŠE SUBJECTS/LIST NEGO SAMO /SUBJECTS
  list             ← VRŠI LIST
  render('list')   ← I PEKICA TI LIST.HTML.ERB (A NE INDEX.HTML.ERB)
end
```

FORME

```
<%= form_for(:subject, :url => {:action => 'create'}) do |f| %>
<%= f.text_field(:name) %>
<%= f.text_field(:position) %>
. . . .
<%= submit_tag("Create Subject") %>
<% end %>
```

ili

```
<%= form_tag(:action => 'create') do %>
<%= text_field(:subject, :name) %>
<%= text_field(:subject, :position) %>
. . . .
<%= submit_tag("Create Subject") %>
<% end %>
```

CREATE

NEW - pripremi formu

- VIEW → NEW.HTML.ERB
  ↳ elementi forme i modela

- CONTROLLER
  → NE TREBA NIŠTA
  ili!
  BOLJE
  → DEF new
  @subject = Subject.new → u formu dodaje pre-definisane elemente (iz migracije)
  end
  ↳ ili Subject.new(:name => 'default') - OVO SAD JE PREDEFINISANO

ACTION CREATE – PROCESIRA FORMU

- CONTROLLER

```
def create
  @subject = Subject.new(params[:subject])
  if @subject.save
    redirect_to(:action => 'list')
  else
    render('new')
  end
end
```

→ INSTANCIRA NOVI OBJEKT

→ OVDJE SE OBJEKT SPREMA + ISPITIVA SA IF

→ AKO JE OK ISPISIVA SVE

→ AKO FORMA NIJE OK PODUNJE VRAĆA NA FORMU KOJA JE PRE-POPULIRANA

- DODAT LINK ZA NEW SUBJECT → U LIST.HTML.ERB

```
<%= link_to("Add new", {:action => 'new'}, :class => 'action new') %>
```

UPDATE edit – displaya formu

↳ ZAHTJEVA ID OBJEKTA DA GA MOŽE UPDATEAT

↳ UPDATE ACTION KORISTI FIND I UPDATE_ATTRIBUTES

- <%= link_to("Edit", {:action => 'edit', :id => subject.id}, :class => 'action edit') %> (OVO U LISTI HTML.ERB)

- CONTROLLER edit
  ↳ def edit → isti kao show

- EDIT.HTML.ERB
  ↳ ISTI KAO NEW.HTML.ERB
  ↳ KOZMETIČKE PROMJENE NEW ⇒ UPDATE i sl.
  ↳ PROMIJENIT {:action => 'update', :id => @subject.id} ← BITNO?

- CONTROLLER update

```
↳ @subject = Subject.find(params[:id])
  if @subject.update_attributes(params[:subject])
    redirect_to(:action => 'show', :id => @subject.id)
  else
    redneder('edit')
  end
```

→ PRIKAZUJE OBJEKT

DELETE

AKCIJE
- DELETE - displaya formu (forma se mora ni potrjat)
- DESTROY - procesira

↳ uvijek trebaju :id
↳ destroy koristi find i destroy metode
↳ delete page može biti po izboru < uporovenje / popis atributa / konfirmacija

u LIST.HTML.ERB
- <%= link_to("delete", {:action => 'delete', :id => subject.id}, :class => 'action delete')%>
  ili 'destroy'

- delete.html.erb → po želji forma
  ↳ forma mora imat :id => @subject.id

- CONTROLLER
  ↳ def delete → isti kao edit
  ↳ def destroy
    Subject.find(params[:id]).destroy
    redirect_to(:action => 'list')
  end

HASH FLASH

flash[:notice] = "Subject created successfully"

- ↳ dodat poruke prije redirect_to... u controllerm za
  ↳ def create
  ↳ def update
  ↳ def destroy
- u list.html.erb (ili bilo koji drugi template)

```
<% if !flash[:notice].blank? %>
  <div class="notice">
    <%=flash[:notice] %>
  </div>
<% end %>
```

VIDI TEMPLATE
↳ ovo se dodaje u templateu

LAYOUTS

u kontrolleru:

layout 'admin' → koristi layout admin.html.erb

```
<html>
  <head>
  ...
  </head>
  <body>
    <%= yield %>
  </body>
</html>
```

- u layoutima možemo stavit instance varijable koje se popunjavaju podjer u yield dokumentu
- u layout staviti i flash hash

PARTIALS - partial templates

- koristit kod sličnih html.erb-ova tipa new i edit
- ime im mijenja ima _ispred : _form.html.erb
- u dokumentima s _form poziva se:

```
<%= render(:partial => "form", :locals => {:f => f}) %>
```

"form" BEZ CRTE_

varijable u dokumentu trenutnom

:f → varijable u _form

STYLE SHEETS

↳ nalaze se u /public/stylesheets

↳ ekstenzija .css

- css dodajemo u file views/layouts (u head tag)

↳ `<%= stylesheet_link_tag('public', :media => 'all') %>`

ili još može -ZA drugi CSS 'admin',

JAVA SCRIPTS

↳ locirano /public/javascripts

↳ završavaju na .js

↳ po defaultu skripte u application.js

- u layoutu (views/layouts) dodat

  <%= javascript_include_tag('application' , 'drugo') %>

- POZIVANJE JS PREKO LINKA

  <%= link_to('Roar', '#', :onclick => "js Roar(); return false;") %>

- POZIVANJE JS-a ODTAH

  <%= javascript_tag("js Roar();") %>

- IZVRŠAVANJE JS-a

  <%= javascript_tag('alert('Hello.');") %>

- IZVRŠAVANJE JS-a INPUTANOG OD USERA

  <%= javascript_tag("alert('#{escape_javascript(text)}');") %>

  NEKA VARIJABLA ILI USER INPUT

  OVO OSIGURAVA DA TEXT NIJE MALICIOZNI JAVASCRIPT KOD

IMAGES

↳ /public/images

↳ nema defaultne file extenzije

↳ može bit subfoldera

- slika

  <%= image_tag('rails.png') %>

  naziv.ext

- opcije

  <%= image_tag('rails.png', :size => '50x60', :alt => "Rails logo") %>

  veličina u pix

  alt text

TEXT HELPERI

Text helpers

<%= word_wrap(text, :line_width => 30) %>

↑ varijabla sa tekstom koju želimo ograničiti

→ ograničava širinu retka na 30 znakova

npr. text = "text. . . "

<%= simple_format(text) %> → pretvara \n -> u
 tag radi html-a

→ 2x \n (\n\n) je novi paragraf

<%= truncate(text, :length => 40, :omission => "...") %>

PRIKAŽE PRVIH 40 ZNAKOVA.

OSTALO ZAMIJENI S OVIM

<%= excerpt(text, 'riječ', :radius => 7, :omission => "...") %>

OVO TRAŽI I PRIKAZUJE 7 ZNAKOVA OKO TE RIJEČI

OSTALO ZAMIJENI S OVIM

HIGHLIGHTA OVO KAD NAĐE U TEKSTU

<%= highlight(text, 'text example') %>

ili

OVO ZNAČI PRVA RIJEČ U NIZU, U OVOM PRIMJERU JE TO 'EXAMPLE'

<%= highlight(text, ['example', 'ramble'], :highlighter => '\1') %>

NIZ RIJEČI HIGHLIGHTA

OVAKO HIGHLIGHTA

<%= auto_link(text) %> - linkove u textu automatski pretvara u "klikabilne" linkove

NAPOMENA

DA BI AUTO LINK RADIO - STAVIT TEXT = "....", HTML_SAFE

DA RAILS ZNA DA JE VARIJABLA OK

pluralize(n, 'octopus') → pametni pluralize

Number & date helperi

NUMBER HELPERI

IZNOS — BROJ DECIMALNIH — JEDINICA — JEDINICA VALUTE — PRVO BROJ

```
number_to_currency (34.5, :precision => 0, :unit => "kr", :format => "%n %u")
```

→ 35 kr

DATE & TIME HELPERI

second	seconds
minute	minutes
hour	⋮
day	⋮
week	<SVI U
month	MNOŽINI>
year	ISTA STVAR

30.days ⇒ 30 dana u sekundama

⋮ itd

~~Time.now~~

Time.now + 30.days − 23.minutes

MOŽE SE ZBRAJAT (SVE JE U SEKUNDAMA)

- ago — npr. 30.days.from_now
- from_now
- beginning_of_ day / week / month / year
- yesterday / tomorrow
- prev_month / next_month
- prev_year / next_year
- end_of_ day / week / month / year

- chainable je ne! NPR. Time.now.prev_year.beginning_of_month

- strftime ⇒ string from time

```
Time.now.strftime("%B %d, %Y %H:%M")
```

%B MJESEC, %H SAT, %M MINUTA

- Time.now.to_s(:long) : time "13:36"

ili

```
Time::DATE_FORMATS[:custom] = "%B %e, %Y at %I:%M %p"
Time.now.to_s(:custom)
```

UZIMA OVAJ FORMATTING

OVU LINIJU STAVIT U FILE: /config/initializers/date_formats.rb

NAPRAVIT NOVI FILE

CUSTOM HELPERI

/app/helpers

↳ tu se nalaze helperi za modele koje mo već napravili

↳ kad mo u npr. Subjects controllern → može se invisat helper metoda iz Subjects helpera (ali NE iz drugih helpera)

↳ isto i za viewe

↳ application_helper je globalan (za sve)

STATUS TAG → POGLEDAT U SIMPLE_CMS APP

FORME

Forme

FORM HELPERS

- form_for
- text_field
- password_field
- text_area
- hidden_field
- file_field
- radio_button
- check_box

OVO MORA BIT AKO IMATO UPLOAD BUTTON

```
form_for(:subject, :url => {:action => "create"},
    :html => {:multipart => true}) do |f|
  f.text_field(:name, :size => 40, :maxlength => 40)
  text_field(:subject, :alt_name, :size => 40, :max length => 40)
  f.password_field(:password, :size => 40, :maxlength => 40)
```

OVO JE ISTA STVAR KAO RED PRIJE

MOGAO JE BIT I NEKI DRUGI OBJEKT

NE VIDI SE PASSWORD

DIMENZIJE POLJA

```
  f.text_area(:description, :size => "40 x 5")
  f.hidden_field(:token)
  f.file_field(:logo)
  f.radio_button(:content_type, "text")
  f.radio_button(:content_type, "HTML")
  f.check_box(:visible)
  submit_tag("Submit")
end
```

ZA OGRANIČAVANJE UNOSA TREBA JAVASCRIPT

-> skriveno polje

-> zahtijeva ono :multipart => true

ODABIR IZMEĐU OVE DVIJE OPCIJE

NE ZABORAVIT STAVIT HTTML RIJEČ IZA KUĆICA

-> super je za boolean

FORM SELECT HELPERS

(padajući izbornik)

f.select (attribute, choices, options = {}, html_options = {})
ili
select (object, attribute, ...)

Opcije
:include_blank => false
: selected => object.attribute

f.select (:position, 1..5, {}, {:style => "width: 4em;"})
- 1..5: IZBOR OD 1 DO 5
- {}: OVO TREBA BIT TU
- {:style => "width: 4em;"}: HTML

- f.select (:content_type, ['text', 'HTML'])
  - ['text', 'HTML']: 2 OPCIJE U NIZU
- f.select (:visible, {"Visible" => 1, "Hidden" => 2})
  - {"Visible" => 1, "Hidden" => 2}: HASH
  - USERU VIDI "VISIBLE" I "HIDDEN", A SPREMAJU SE 1 I 2
- f.select (:page_id, Page.all.collect {|s| [s.name, s.id]})
  - [s.name, s.id]: OVO JE JEDNA OPCIJA
  - [name, id], [name, id]
  - name: PRIKAZUJE SE USERU
  - id: SUBMITTA SE

DATE & TIME FORM HELPERS

- date_select
- time_select
- datetime_select

↳ napravi nenovezna polja za unos za padajućim izbornicima

FORM TAG HELPERS (POGLAVLJE 11)

↳ razdvojeni od active record upisa

- prije je bilo form_for sad → form_tag
  ↳ i radar za sve elemente od forme

- mogu se miksat sa form_for
  - select_tag ("gender", options_for_select (["male", "female"], selected_text)

KOJA VRIJEDNOST JE PRE-SELECTIRANA

FORM ERRORS

- validates_presence_of :name (ATRIBUT) → IDE U MODEL

- object.errors → ubacuje greške u ovo
  (subject company)

- object.errors.clear
  .size
  .each {|attr, msg| ...}
  .full_messages.each {|msg| ...}

NAZIV ATRIBUTA — TEKST GREŠKE — 1. ATRIBUT MOŽE IMAT VIŠE GREŠAKA

- napisat vlastiti HTML za greške views/shared/_error_messages.html.erb

OVAJ DIR SAMI NAPRAVIMO

OVAJ FILE IMA VARIJABLE KOJE "VUČE" OSIGURAT DA IH DOBIJE IZ NPR _FORM.HTML.ERB

- u _form.html.erb na vrh

<%= render(:partial => 'shared/error_messages', :locals => {:object => @subject}) %>

RENDERIRA NEKI PARTIAL LAYOUT FILE

LOKALNE VARIJABLE TU U FILEU

OVAKO SE ZOVE U _ERROR_MESSAGES

OVAKO SE ZOVE U _FORM.HTML.ERB

JOŠ BOLJE

- STAVIT U APPLICATION HELPER → TU DEFINIRAMO POZIVANJE OVOG PARTIAL LAYOUTA

```
def error_messages_for(object)
  render(:partial => 'shared/error_messages', :locals => {:object => object})
end
```

- U _FORM.HTML.ERB POZIVAMO HELPER

~~<%= error_messages_for(@object) %>~~ → OVO JE DEPRECATED → VIDIT NOVO NA KRAJU OVIH BILJEŠKI

→ SAD TO LAKO NAPRAVIMO I U DRUGIM KLASAMA, A PROMJENIMO SAMO VARIJABLU

→ NE ZABORAVIT DODAT views/shared/_error_messages.html.erb

→ NE ZABORAVIT DA OVA DATOTEKA IMA _ ISPRED IMENA

FORM FIELD LABELS

```
label(object, attribute, text)
f.label(attribute, text)
label_tag(name, text).
```

- npr
  `f.label(:name)` → u formi da se moramo pisat <th> (Name) </th> UMJESTO OVOGA
  → KAO BONUS IME SE ZACRVENI AKO JE GREŠKA !
  `f.label(:name, "Ime")`
  CSS ⇒ field_with_errors

ESCAPE INPUT

↳ da se svi input od korisnika besrsleno renderira (da nije SQL injection)

- urlovi
- form parametri
- cookie data
- database data

- rails escapea data za Hash i array tipa
  - `subject.where(:name => user_query, :visible => 1)`
    ili
  - `subject.where(["name = ? AND visible = 1", user_query])`

MOŽE BIT I VIŠE UPITNIKA, ONDA TREBA BIT VIŠE VRIJEDNOSTI TU

- rails escapea automatski dodavanje atributa
  - subject.update_attributes(:name => "subjekt 1")

ESCAPING OUTPUT

↳ IZBJEGAVANJE IZVRŠAVANJA JAVASCRIPTA KOJI JE UBAČEN UMJESTO VARIJABLE ILI POLJA

METODE

- html_escape() , h()
- raw() → izbjegava escape (html i JS se izvršavaju)
- html_safe , html_safe? — ispituje ima li tekst JS-a i HTML-a

- u railsima 3 sav se text automatski escapea
  ↳ mi trebamo reć railsu da je html safe

- strip_links(html) → miče linkove
- strip_tags(html) → miče tagove tipa <p>, , <a href...>
- sanitize(html, options) → miče sav [redacted] JS

  ↙
  :tags => ['p', 'br', ...], :attributes => ['id', 'class', 'style']
  ↳ ONDA SE PROPUSTA (WHITELISTA) NAVEDENO, A SVE OSTALO SE BLACK-LISTA

- text = "tecnnn...", html_safe → sad je sanjalica uvijek safe

DATA VALIDATION

Data validation

↳ u MODELU!

- validates_presence_of (:name, :option)

  :message => "can't be blank" → MOŽE SE CUSTOMIZIRAT

- validates_length_of (:name, :option)

  ATRIBUT

  :is, :minimum, :maximum (integer)

  :within, :in (range) (ISTO SINONIMI)

  :wrong_length => "is the wrong length (should be {{count}} chars)"

  :too_short => "is too short ..."

  :too_long => "is too long ..."

- validates_numericality_of (:number, :option,...)

  :equal_to, :greater_than, :less_than, :greater_than_or_equal_to,

  :less_than_or_equal_to (numeric)

  :odd, :even, :only_integer (boolean)

  :message => "is not a number"

- validates_inclusion_of → atribut mora biti u listi (array ili range)

  :in (enumerable)

  :message => "is not included in the list"

- validates_exclusion_of → ISTO KAO OVO GORE

  :message => "is reserved"

- validates_format_of

  :with (regular expression) ← EMAIL_REGEX

  :message => "is invalid"

VELIKA SLOVA

OVO DODATI KAO KONSTANTU U MODEL

```
EMAIL_REGEX = /\A[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\z/i
```

- validates_uniqueness_of
  - ↳ atribut ne smije postojat u bazi za taj objekt
  - :case_sensitive (boolean)
  - :scope (column symbols for limiting scope)
  - :message ⇒ "has already been taken"

- validates_acceptance_of — za checkbox npr "I agree to the services of license..."
  - ↳ stvara virtualni atribut ako nema stupca za atribut
  - NE SPREMA SE U BAZU
  - :accept (expected value, "1")
  - :message ⇒ "must be accepted"

- validates_confirmation_of → za emailove ili passworde
  - ↳ atribut se mora potvrdit unošenjem dva puta
  - ↳ stvara virtualni atribut npr :email ⇒ :email_confirmation
  - ↳ validira potvrdu ako atribut nije nil
  - U FORMAMA SE TO KORISTI KAO u. email_confirmation
  - :message ⇒ "doesn't match confirmation"

OPCIJE ZA VALIDACIJE

- :allow_nil ⇒ true → ne validira ako atribut ⇒ nil (nije potvrđeno)
- :allow_blank ⇒ true → ISTO PRESKAČE ako je atribut (nil, false, "", [], {})

- :on ⇒ :save / :create / :update
  - ↳ validira samo ako se radi o odabranoj akciji
  - :save → validira u sva slučaja (default)
- :if ⇒ :method / :unless ⇒ :method
  - :method je metoda definirana u modelu koja vraća true ili false

s = Subject.new

- s.errors → vraća hash sa greškama
- s.valid? → provjerava validacije i fraća true ili false (PROŠAO SVE VALIDACIJE / NIJE PROŠAO VALIDACIJE)
- s.errors.full_messages ⇒ ["Name can't be blank"] (Name = IME ATRIBUTA)

VALIDATES METHOD

↳ very validation

- isto kao "normalne" validacije, ali kraće

- validates :email, : presence => true,
  : length => {: maximum => 50},
  : uniqueness => true,
  : format => {: with => EMAIL_REGEX},
  : confirmation => true

→ trenutno je teško ići nikako promijeniti error message...

VALIDATES ASSOCIATED METHOD

- validates_associated (association, options)
  ↳ associani objekti moraju biti valid
  ↳ nema very validations
  ↳ ako nema associranog objekta => neće pasti greška. → DOBAR PRIJE validates_presence_of :object

! infinite loops & long cascades

- class Auction < ActiveRecord::Base
  has_one :car
  validates_presence_of :car
  validates_associated :car
  end

USER AUTHENTICATION

HASHING PASSWORD

↳ ne spremat passworde kao text !

↳ algoritmi : MD5, SHA-1, SHA-2

↳ SHA1 ?

u IRB-u

```
Digest::SHA1.hexdigest(variable)   'text'
```

- u modelu, vrh, prije class
  → require 'digest/sha1'

```
def self.hash(password="")
    Digest::SHA1.hexdigest(password)
end
```

- prvi se:

MORATE OVO POZVAT JER SMO TU DEFINIRALI PASSWORD HASH METODU

```
user.hashed_password = Admin User.hash('password text')
```

SALTING PASSWORDS

- password + "salt string" => SHA1 => ~~kad~~ kad je password pre-dug za rainbow tables
- salt koji ukljucuje i username => SHA1 za koji treba i pass i username
- random salt → ukljucuje i random broj
- password + Time.now
  ↳ store salt in database => hash it
- unique, random, enkripted salt

Callbacks

U MODELU

- na vrh: require 'digest/sha1'
  ⋮
- def self.make_salt (username = "")
  Digest::SHA1.hexdigest ("Use #{username} with #{Time.now} to make salt")
  end

  OVAJ CIJELI SALT SPREMAMO U BAZU

  ~~OVO SPREMITI U BAZU~~

  def self.hash_with_salt (password = "", salt = "")
  Digest::SHA1.hexdigest ("Put #{salt} on the #{password}")

- upotreba:
  user.salt = AdminUser.make_salt ("username string ili variable")

  OVO JE RANDOM I SPREMLJENO ZA SVAKOG USERA

  user.hashed_password = AdminUser.hash_with_salt ("password string", user.salt)

MASS ASSIGNMENT

- attr_protected
  ↳ atributi se preskaču u mass assignmentu
- attr_accessible
  ↳ definiraju se dostupni → svi ostali su zaštićeni

- SAMO U MASS ASSIGNMENTU
- DIRECT ASSIGNMENT JE ISTI
  'User.name = ...'

- u modelu
  attr_protected :hashed_password, :salt

CALLBACKS – sve ji definiramo u modelu

	CREATE	UPDATE	DESTROY
PRIJE VALIDACIJE	before_validation	before_validation	
	← VALIDACIJA		
POSLIJE VALIDACIJE	after_validation	after_validation	

PRIJE BAZE	before_save before_create	before_save before_update	before_destroy
			← INTERAKCIJA S BAZOM
POSLIJE BAZE	after_create after_save	after_update after_save	after_destroy

class AdminUser < ActiveRecord::Base
  before_save :create_hashed_password

  private ← nemoš ovu clasu ima pristup
  def create_hashed_password
  ...
  end
end

nadi validacija u modelu

- attr_accessor :password → non database atribut → ne sprema se u bazu, ali se koristi User.password
- before_save :create_hashed_password
  after_save :clear_password
  } callbacks ↳ idu ispod validacija RADE I ZA NEW I ZA UPDATE

- private → SVE ISPOD OVOGA ZNAČI DA SE METODE MOGU ZVAT SAMO IZ TE KLASE
- def create_hashed_password — METODA STVARA HASHED PASSWORD SAMO AKO NIJE PRAZAN
    unless password.blank?
      self.salt = AdminUser.make_salt(username) if salt.blank?
      self.hashed_password = AdminUser.hash_with_salt(password, salt)
    end
- end

- def clear_password → ZBOG SIGURNOSTI I DA SE PASSWORD NE BI SVAKI PUT HASHIRAO
    self.password = nil
  end

- validates_length_of :password, :within => 8..25, :on => :create

AUTENTIFIKACIJA
↳ u modelu

- def self.authenticate(username="", password="")
    user = AdminUser.find_by_username(username)
    if user && user.password_match?(password) — METODA KOJA VRAĆA BOOLEAN VRIJEDNOSTI (TRUE, FALSE)
      return user
    else
      return false
    end
  end
- def password_match?(password="")
    hashed_password == AdminUser.hash_with_salt(password, salt)
  end

LOGIN & LOGOUT

↳ u kontroler

- rails generate controller Access menu login (generira se admin i template)
- dodat akcije def index, attempt_login, logout
  - menu
  - render ('menu')
  - dodat na vrh: layout 'admin'
- menu.html.erb → uredit da ima linkove koji trebaju imati
- login → form_tag forma koja ima action attempt_login
- routat config/routes.rb

  match 'admin' : to => 'access# menu'

- def attempt_login

  authorized_user = Admin User. authenticate (params [:username] , params [:password])

  if authorized_user

    flash [: notice] = "You are logged..."

    redirect_to (: action => 'menu')

    session[:user_id] = authorized_user. id

    session [:username] = authorized_user. username

  else

    flash[: notice] = " Neispravno korisničko ime ili lozinka"

    redirect_to (: action => 'login')

  end

- def logout

  flash [:notice] = "You have odlogiran"

  redirect_to (: action => "login")

  session [:user_id] = nil

  session [:username] = nil

end

COOKIES & SESSIONS

↳ kukiji ~ 4kB

```
cookies[:username] = "jsmith"
cookies[:username] = {:value => "jsmith", :expires => 1.week.from_now}
```

SESSIONS

```
session[:username] = "jsmith"
```

↳ super-cookie ⇒ 4kb, enkriptirano

↳ cookie storage — THE BEST

/config/initializers
- session_store.rb → odredujemo ime cookia i gdje se session sprema
- secret_token.rb → sadrži key za enkripciju cookiea

- primjer korištenja u controlleru:

```
session[:user_id] = authorized_user.id
```

RESTRICTING ACCESS - BEFORE FILTERS

- callbacks su za modele - ekskluzivno
- before filters su za kontrolere ekskluzivno

- class AccessController < ApplicationController

```
before_filter :confirm_logged_in,
    :except => [:login, :attempt_login, :logout]
    :only => [:metoda x]

private
def confirm_logged_in
```

→ IDE ISPOD LAYOUT 'ADMIN' U KONTROLERU

→ TREBAJU BIT PRIVATE ? DA SE NE MOGU POZIVAT KAO AKCIJE

- must return false to halt filter chain:
  - private
  - def confirm_logged_in

```
unless session[:user_id]
    flash[:notice] = "Please log in."
    redirect_to(:action => 'login')
    return false
else
    return true
end
```

- confirm_logged_in metoda + private ide u ApplicationController (da ni kontroleri mogu koristit)
  ↳ potopć: protected
- dodat:
  before_filter : confirm_logged_in
  za sve kontrolere
  ↳ promijenit:
  redirect_to(:controller => 'access', :action => 'login')

OVO DODAJEMO JER AKCIJA LOGIN POSTOJI SAMO U OVOM KONTROLERU

(14) IMPROVING THE SIMPLE CMS

NESTING

↳ ni redirekti moraju računat ID "višeg" objekta

- SUBJECTS / LIST.HTML.ERB

```
<%= link_to("View Pages", {:controller => 'pages', :action => 'list', :subject_id => subject.id},
            :class => 'action show') %>
```

PAGES CONTROLLER

- before_filter :find_subject

```
...
private
def find_subject
  if params[:subject_id]
    @subject = Subject.find_by_id(params[:subject_id])
  end
end
```

- def list dodat
  ... .where(:subject_id => @subject.id)
- def new izmjenit
  @page = Page.new(:subject_id => @subject.id)
  → DA OVO ODMAH POSTAVI
- def create izmjenit
  ...
  redirect_to(:action => 'list', :subject_id => @page.subject_id)
  ↓ AKO JE SUBJECT REFIJENJEN

def update - izmjenit
⋮
redirect_to (:action => 'show', :id => @page.id, :subject_id => @page.subject

def destroy - izmjenit
⋮
redirect_to (:action => 'list', :subject_id => @subject.id)

- u svim viewsima dodat u linkove ..., :subject_id => @subject.id }
- u pages/list.html.erb dodat link na ⇒ subject

<%= link_to ("<< Back to Subjects List", {:controller => "subjects",
:action => 'list' }, :class => 'back-link') %>

MODULI

lib/ ~~position_mover~~.rb — ovdje ne radi!
module PositionMover
def move_to_position (new_position)
⋮
end
end

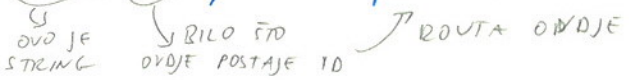
app/models/subject.rb
class Subject < ActiveRecord::Base
include PositionMover ← MODULE JE KAKO INSTANCIRAT U MODELIMA (ISTO KAO DA SMO KOPIRALI KOD)
end

- rails loada sve iz app/helpers, app/models/, ~~lib/~~ NEKAD!

- ZA INCLUDAT MODUL
↳ prije class u modelu
require 'lib/position_mover'
↳ nakon class
include PositionMover

PUBLIC AREA

↳ controller Public index show
↳ novi layout Public.html.erb
↳ index.html.erb će statičan tekst
↳ def show controller da prikazuje stranicu s :idem
↳ routes ⇒ match 'show/:id', :to ⇒ 'public#show'



↳ napravit show view
↳ promijenit glavni route u applikaciji

PUBLIC AREA NAVIGATION

subjects ||= [] → pretvara objekte u array ⇒ tako ako ne baca grešku ako je prazno (nil)

link_to_unless_current ⇒ isto kao link_to

ERRORS

LOG FILES

- logger.debug ("The name is #{@subject.name}")
- logger.info ("Starting the subject update...")
- logger.warn ("Invalid log in attempted for #{params[:username]}")
- logger.error ("Page with permalink #{params[:id]} not found")
- logger.fatal ("Position Mover module not loaded")

DEBUGGING

- object.inspect
- <%= debug(object) %>
- puts "*** Got here ***"
- ruby-debug

GREŠKE NA PRODUKCIJI

config/environments/production.rb
config.consider_all_requests_local = false → pokazuje error pages

public/404.html
/500.html
/422.html

- exception notification
  šalje email kad se dogodi greška na aplikaciji

↳ GIT ili GEM
EXCEPTION_NOTIFICATION

DEPLOYING

1. ISP sa Rails → HEROKU, JOYENT, LINODE, RAILS MACHINE, ENGINEYARD, RAILSPLAYGROUND, RIMUHOSTING, SLICEHOST
2. INSTALL software na server → RUBY, RUBYGETS, RAILS, MYSQL, MYSQL GEM
   ↳ SSH username@server.com
3. choose web server
   - FULL-FEATURED: APACHE, NGINX, LIGHTTPD
   - LAGANI: PASSENGER / MOD_RAILS, MONGREL, THIN, UNICORN
   - APACHE + PASSENGER
4. get deployment tools
   - FTP, GIT
   - DEPLOY SCRIPTING
     ↳ CAPISTRANO HTTP://CAPIFY.COM