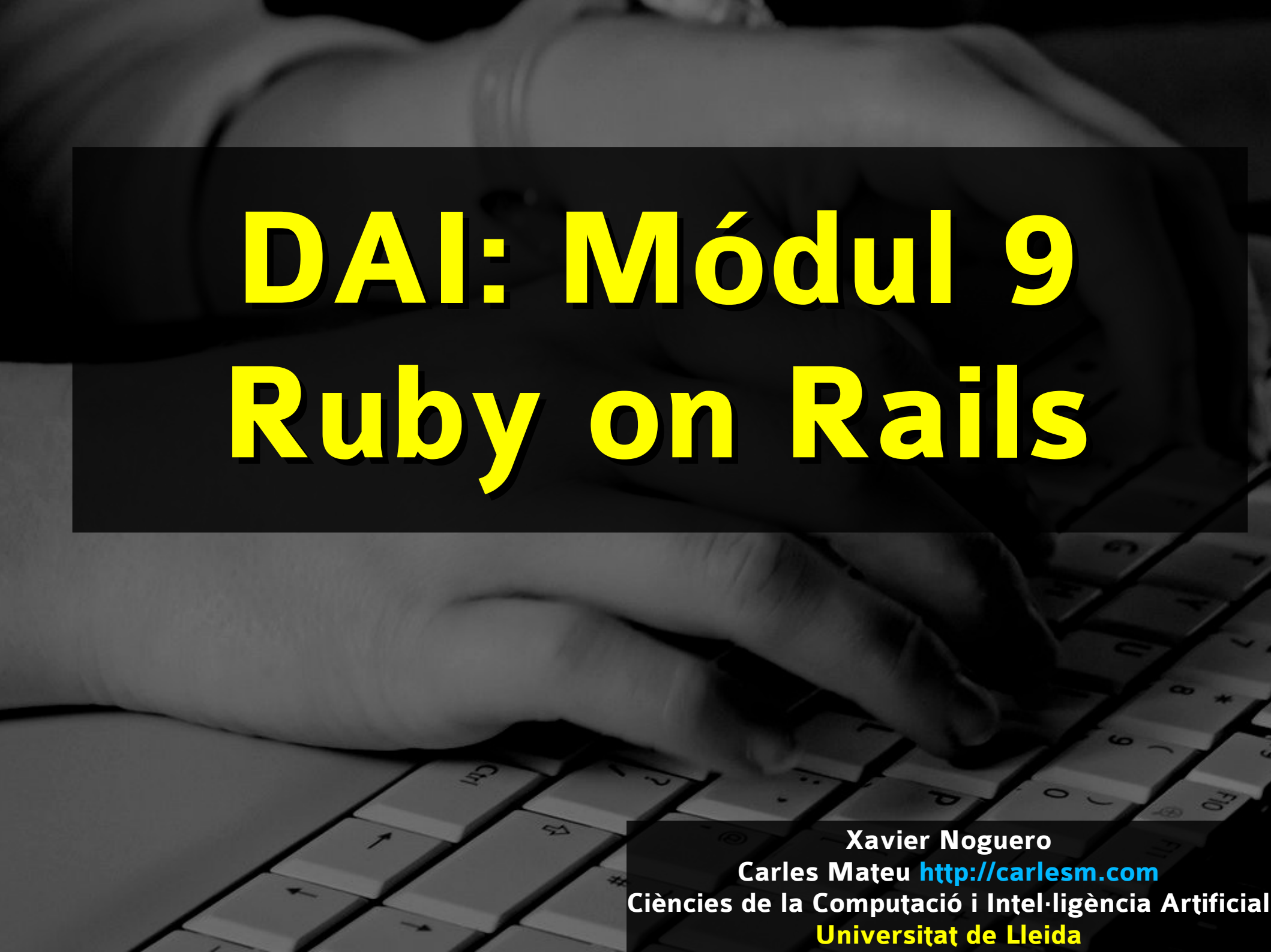


DAI: Mòdul 9
Ruby on Rails

Xavier Noguero
Carles Mateu http://carlesm.com
Ciències de la Computació i Intel·ligència Artificial
Universitat de Lleida

Ruby on Rails, què és?

- Ruby és un llenguatge de programació.
- Rails és un "framework" modern que fa molt fàcil el desenvolupament i manteniment d'aplicacions web.

- Molt escollit per crear aplicacions Web 2.0!!

Ruby on Rails, perquè?

- Java? PHP? .NET? Molta feina per a desenvolupar...
- Rails, és fàcil!
  - Tècniques modernes i professionals.
- Un exemple: Patró Model-Vista-Controlador
  - En Java: Struts, Tapestry...
  - Amb Ruby i Rails:
    - L'esquelet de l'aplicació es crea automàticament.
    - Només ens cal crear el codi just de la lògica de negoci.
    - Tests, es creen automàticament!

La base, Ruby

- És un llenguatge interpretat orientat a objectes.
- No és nou, va ser creat als anys 90 per un japonés, en Yukihiro Matsumoto.
- S'ha fet popular durant els últims anys a mesura que han aparegut llibres i documentació en anglès.

  (i en gran mesura gràcies a Rails).

irb, un shell interactiu de Ruby

- És intérpret de comandes que es pot utilitzar per a testejar codi ràpidament:

```
$irb
irb(main):001:0> puts "Hello, World"
Hello, World
=> nil
irb(main):002:0> 2+3
=> 5
irb(main):003:0>
```

rdoc, per a fer documentació

- És l'eina que ens permet generar la documentació del codi que escrivim:

```
$rdoc [options] [names...]
```

- Es pot especificar de quin o quins fitxers volem que es generi la documentació.
- El format de sortida és HTML.

RubyGems

- És l'eina per a gestionar el sistema de paquets de Ruby:
  - Un format estàndard per a distribuir programes i llibreries Ruby.
  - Gestiona la instalació de paquets gem (de manera similar com ara yum o apt-get per a distros linux).

Rails is Agile

- L'any 2001, 17 crítics dels models de desenvolupament de programari, van signar l'Agile Manifesto.

  http://agilemanifesto.org/

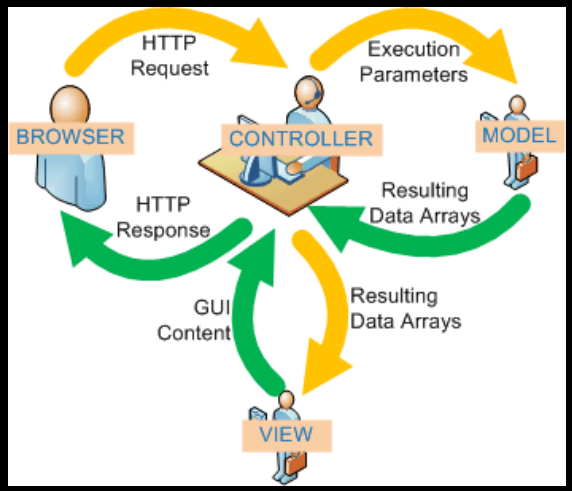
- Aquest, defineix nous mètodes de desenvolupament de programari com una alternativa a les metodologies formals existents, considerades massa “pesades” i rígides pel seu caràcter normatiu i la seva alta dependència en les planificacions prèvies a l'etapa de desenvolupament.
- Alguns exemples: Extreme Programming, Scrum...
- Rails, per tal com s'ha concebut, segueix aquestes pràctiques!

Extremme Programing (XP)

- Considerada per molts com “la manera com realment volen treballar els programadors”
- Es basa principalment en:
  - Fer releases petites i freqüents.
  - Desenvolupar en cicles iteratius.
  - Implementar només el que hi ha l'especificació.
  - Escriure sempre primer el codi de test.
  - Seguir la planificació de manera realista.
  - Refactoritzar sempre que es pugui.
  - Fer-ho tot simple.
  - Programar en parelles, i intercanviar rols dins el projecte, per tal que tothom conegui millor el codi.

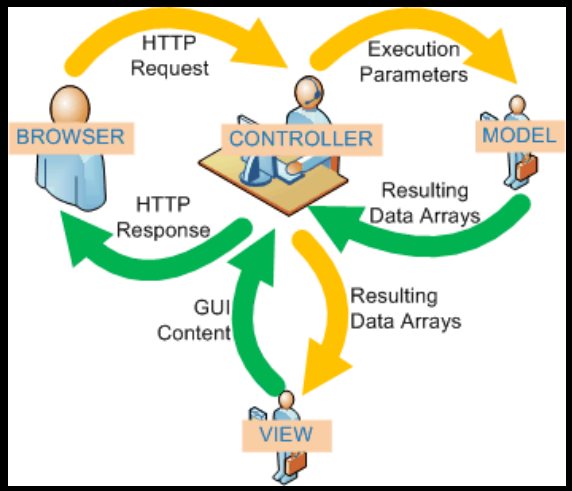
Arquitectura MVC

- Patró Model-Vista-Controlador



Arquitectura MVC - Model

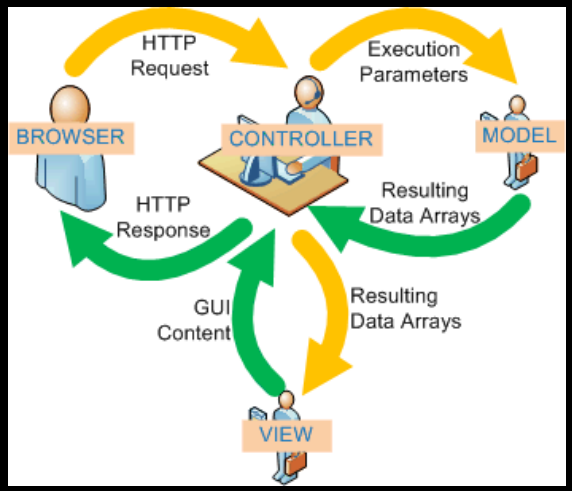
- Patró Model-Vista-Controlador



- Model:
  - Part responsable de mantenir l'estat de l'aplicació.
  - Engloba:
    - Dades: manipulació de bases de dades, fitxers, etc.
    - Regles per mantenir coherents les dades en tot moment.

Arquitectura MVC - Vista

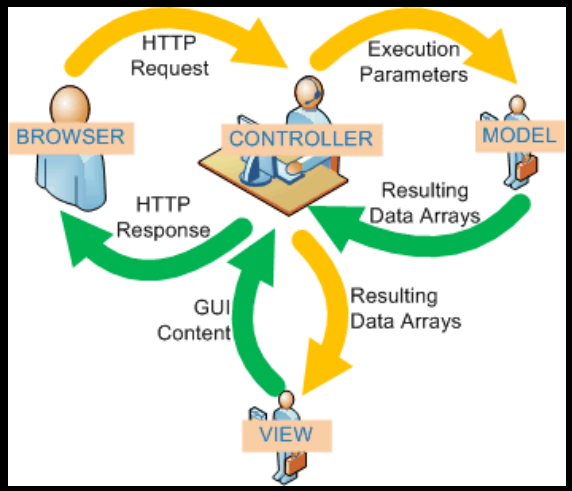
- Patró Model-Vista-Controlador



- Vista:
  - Part responsable de generar la interfície d'usuari.
  - Accedeix al model per a presentar les dades a l'usuari de l'aplicació

Arquitectura MVC - Controlador

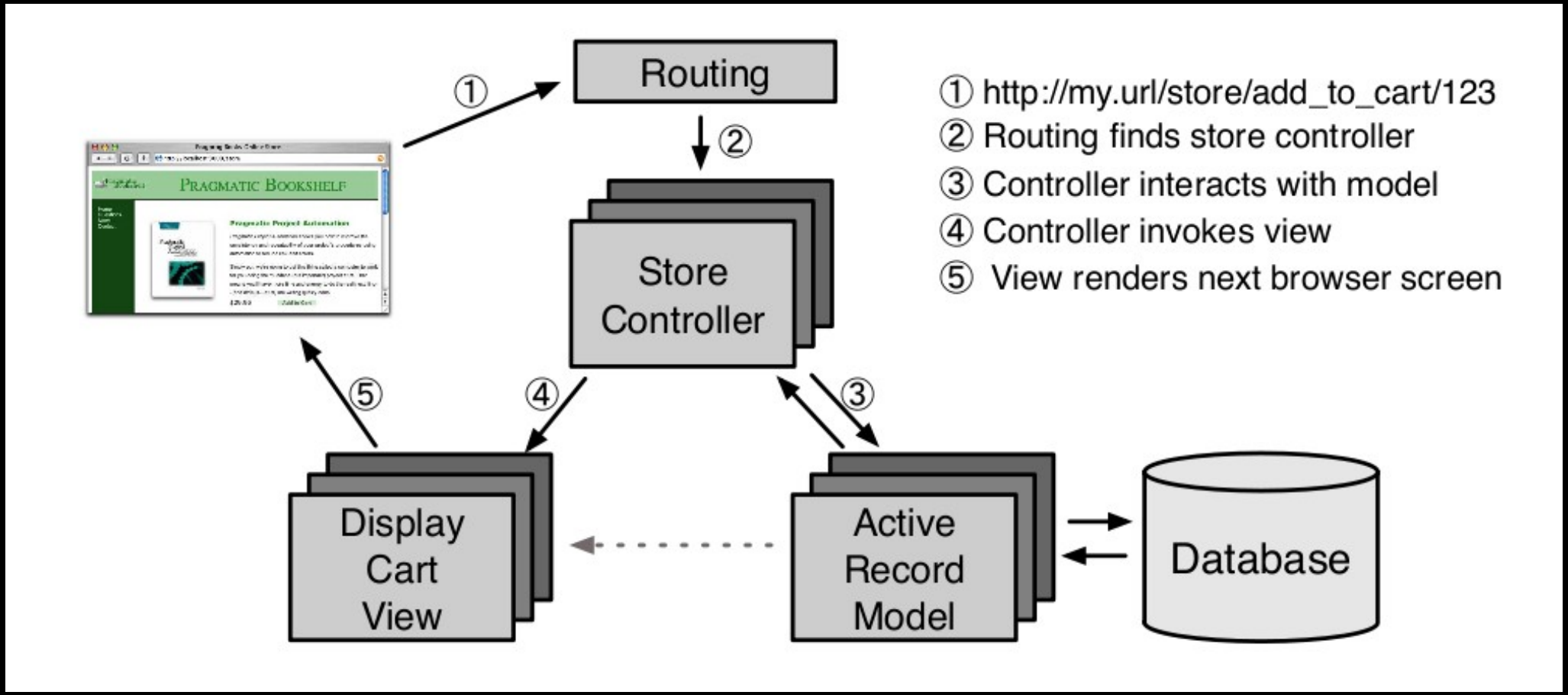
- Patró Model-Vista-Controlador



- Controlador:
  - Part responsable d'orquestrar el funcionament de l'aplicació.
  - Rep events de l'usuari, interacciona amb el model i tria una vista adecuada per a donar resposta de nou a l'usuari.

Rails és un framework M-V-C

- Patró Model-Vista-Controlador amb Rails:



- Model: Active Record
- Vista i controlador: Action Pack

Les URL's amb Rails

- Imaginem una URL d'una aplicació web Rails:

  http://laNostraAplicacio.com/Saludar/hello

- En aquest exemple:
  - laNostraAplicacio.com: adreça base de l'aplicació.
  - Saludar: és el controlador.
  - hello: és l'acció que s'executarà.

- Per tant, les URL's en Rails es mapegen a controladors i accions!

Programant Ruby on Rails

- Primer hem d'instal·lar les eines adients:
  - Ruby
  - Ruby on Rails (Gem)
  - Base de dades (SQLITE3, MySQL)
  - Servidor Web
- Opcions:
  - Bitnami: (http://bitnami.org) Kits de programació (Stacks) per Linux/Windows/Mac/Solaris
  - Locomotive (Mac)
  - InstantRails (Windows)
  - Distribució (Linux)
  - Distribució (Mac)

La primera aplicació Rails

- Eines de Desenvolupament:
  - Aptana RadRails (http://www.radrails.org)
  - Xcode (http://developer.apple.com)
  - TextMate (http://macromates.com) $$
  - Netbeans (http://netbeans.org/features/ruby/index.html)
  - jEdit (http://www.jedit.org)
  - Vim (http://www.vim.org/scripts/script.php?script_id=1567)
  - Gedit (http://github.com/mig/gedit-rails)

Programant Ruby on Rails

- Eines de Desenvolupament:
  - Aptana RadRails (http://www.radrails.org)
  - Xcode (http://developer.apple.com)
  - TextMate (http://macromates.com) $$
  - Netbeans (http://netbeans.org/features/ruby/index.html)
  - jEdit (http://www.jedit.org)
  - Vim (http://www.vim.org/scripts/script.php?script_id=1567)
  - Gedit (http://github.com/mig/gedit-rails)

La primera aplicació

- Crearem una aplicació que mantindrà una base de dades de Restaurants, una de crítics.
- Tindrem a més crítiques per cada restaurant, amb puntuació.
- Farem les crítiques amb AJAX (sense recarregar planes).
- Serà una aplicació ràpida.
- No correcta (només per anar fent a classe).

La primera aplicació

- Instal·lem el rubystack.
- Dins de rubystack:
  - `./rubyconsole`
  - Això carrega variables d'entorn.
- Crearem ara l'aplicació. Ho farem a: projects/ (dins de rubystack).

La primera aplicació

- Creem aplicació:

  `rails baretos`
- Bareţos és el nom de l'aplicació
- Crearà amb SQLITE3 per defecţe. Per mysql:

  `rails -d mysql baretos`

- El creem amb SQLITE3.

La primera aplicació

- Podem provar l'aplicació creada:

  `script/server`

- Això arranca un servidor web (mongrel) al port 3000 i desplega la nostra aplicació. Ens hi connectem amb el navegador:

  http://localhost:3000/

La primera aplicació

- Començarem generant un model pels restaurants.
- Podríem començar generant un controlador, etc.
- El restaurant tindrà:
  - nom: Nom del restaurant.
  - adreca: Adreça del carrer.

```
script/generate scaffold restaurant nom:string
adreca:string
```

```
bash-4.0$ script/generate scaffold restaurant nom:string adreca:string lng:float lat:float
      exists  app/models/
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/restaurants
      exists  app/views/layouts/
      exists  test/functional/
      exists  test/unit/
      create  test/unit/helpers/
      exists  public/stylesheets/
      create  app/views/restaurants/index.html.erb
      create  app/views/restaurants/show.html.erb
      create  app/views/restaurants/new.html.erb
      create  app/views/restaurants/edit.html.erb
      create  app/views/layouts/restaurants.html.erb
      create  public/stylesheets/scaffold.css
      create  app/controllers/restaurants_controller.rb
      create  test/functional/restaurants_controller_test.rb
      create  app/helpers/restaurants_helper.rb
      create  test/unit/helpers/restaurants_helper_test.rb
       route  map.resources :restaurants
  dependency  model
      exists    app/models/
      exists    test/unit/
      exists    test/fixtures/
      create    app/models/restaurant.rb
      create    test/unit/restaurant_test.rb
      create    test/fixtures/restaurants.yml
      create    db/migrate
      create    db/migrate/20100511103805_create_restaurants.rb
bash-4.0$
```

```
bash-4.0$ script/generate scaffold restaurant nom:string adreca:string lng:float lat:float
      exists  app/models/
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/restaurants
      exists  app/views/layouts/
      exists  test/functional/
      exists  test/unit/
      create  test/unit/helpers/
      exists  public/stylesheets/
      create  app/views/restaurants/index.html.erb
      create  app/views/restaurants/show.html.erb
      create  app/views/restaurants/new.html.erb
      create  app/views/restaurants/edit.html.erb
      create  app/views/layouts/restaurants.html.erb
      create  public/stylesheets/scaffold.css
      create  app/controllers/restaurants_controller.rb
      create  test/functional/restaurants_controller_test.rb
      create  app/helpers/restaurants_helper.rb
      create  test/unit/helpers/restaurants_helper_test.rb
       route  map.resources :restaurants
  dependency  model
      exists    app/models/
      exists    test/unit/
      exists    test/fixtures/
      create    app/models/restaurant.rb
      create    test/unit/restaurant_test.rb
      create    test/fixtures/restaurants.yml
      create    db/migrate
      create    db/migrate/20100511103805_create_restaurants.rb
bash-4.0$
```

VISTA

CONTROLADOR

MODEL

La primera aplicació

- Tenim a més un mecanisme de migració (actualització) de la Base de Dades.

  ```
  rake db:migrate
  ```

- Executem de nou l'aplicació

  ```
  script/server
  ```

- I anem al navegador (http://localhost:3000/restaurants/)

La primera aplicació

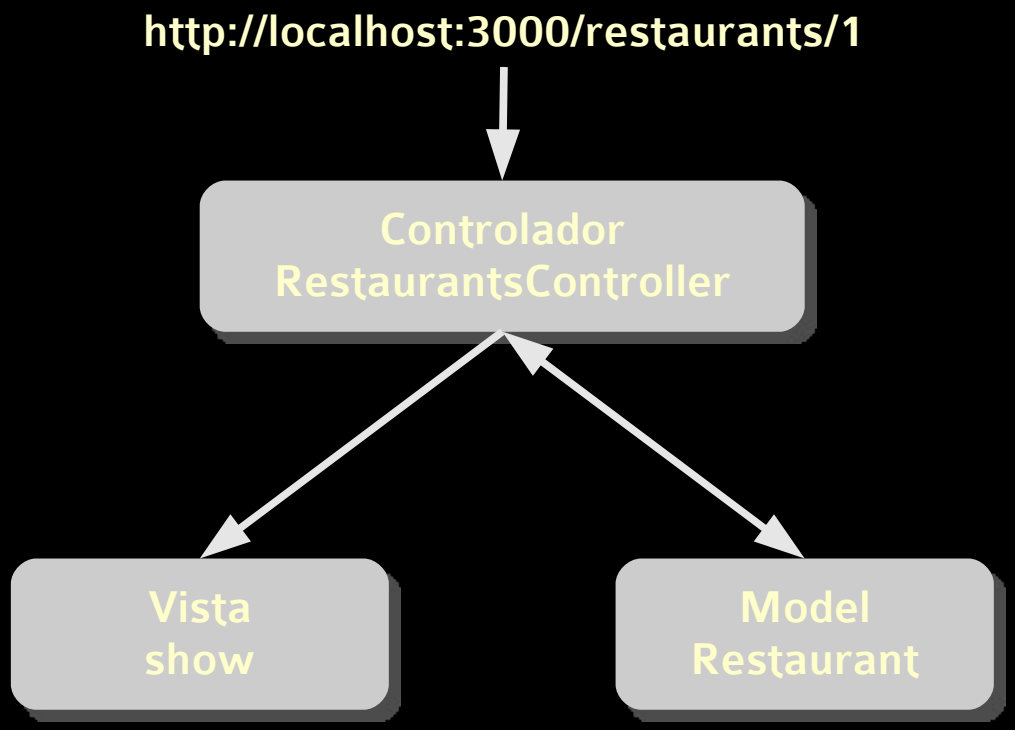
- Generem també un model pels nostres crítics de restaurant.
- El crític tindrà:
  - nom: Nom del restaurant.
  - email: Adreça de mail

```
script/generate scaffold critic nom:string
email:string
```

```
rake db:migrate
```

Com funciona?

- app/
  - controllers/
    - restaurants_controller.rb
  - models/
    - restaurant.rb
  - views/
    - restaurants/
      - show.html.erb



- Segueix REST

La primera aplicació

- Els recursos que farem accessibles per REST declarats a `config/routes.rb`:

```
map.resources :critics

map.resources :restaurants
```

- Les rutes que queden aleshores (rake routes):

```
       critics GET    /critics(.:format)                {:action=>"index", :controller=>"critics"}
               POST   /critics(.:format)                {:action=>"create", :controller=>"critics"}
    new_critic GET    /critics/new(.:format)            {:action=>"new", :controller=>"critics"}
   edit_critic GET    /critics/:id/edit(.:format)       {:action=>"edit", :controller=>"critics"}
        critic GET    /critics/:id(.:format)            {:action=>"show", :controller=>"critics"}
               PUT    /critics/:id(.:format)            {:action=>"update", :controller=>"critics"}
               DELETE /critics/:id(.:format)            {:action=>"destroy", :controller=>"critics"}
   restaurants GET    /restaurants(.:format)            {:action=>"index", :controller=>"restaurants"}
               POST   /restaurants(.:format)            {:action=>"create", :controller=>"restaurants"}
new_restaurant GET    /restaurants/new(.:format)        {:action=>"new", :controller=>"restaurants"}
edit_restaurant GET   /restaurants/:id/edit(.:format)   {:action=>"edit", :controller=>"restaurants"}
    restaurant GET    /restaurants/:id(.:format)        {:action=>"show", :controller=>"restaurants"}
               PUT    /restaurants/:id(.:format)        {:action=>"update", :controller=>"restaurants"}
               DELETE /restaurants/:id(.:format)        {:action=>"destroy", :controller=>"restaurants"}
                      /:controller/:action/:id
                      /:controller/:action/:id(.:format)
```

La primera aplicació

- Això provoca crides al Controller (app/controllers/restaurants_controller.rb):

```
class RestaurantsController < ApplicationController
  # GET /restaurants
  # GET /restaurants.xml
  def index
    @restaurants = Restaurant.all

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @restaurants }
    end
  end

  # GET /restaurants/1
  # GET /restaurants/1.xml
  def show
    @restaurant = Restaurant.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @restaurant }
    end
  end
 [....deletia....]
end
```

La primera aplicació

- El Model (app/models/restaurant.rb):

```
class Restaurant < ActiveRecord::Base
end
```

- i la db:migrate (db/migrate/....create_restaurants.rb):

```
class CreateRestaurants < ActiveRecord::Migration
  def self.up
    create_table :restaurants do |t|
      t.string :nom
      t.string :adreca

      t.timestamps
    end
  end

  def self.down
    drop_table :restaurants
  end
end
```

La primera aplicació

- El respond_to crida la vista (apps/views/restaurants/show.html.erb):

```
<p>
  <b>Nom:</b>
  <%=h @restaurant.nom %>
</p>

<p>
  <b>Adreca:</b>
  <%=h @restaurant.adreca %>
</p>

<p>
  <b>Lng:</b>
  <%=h @restaurant.lng %>
</p>

<p>
  <b>Lat:</b>
  <%=h @restaurant.lat %>
</p>


<%= link_to 'Edit', edit_restaurant_path(@restaurant) %> |
<%= link_to 'Back', restaurants_path %>
```

Validació de dades

- Validarem ara les dades:

```
------> app/models/critic.rb
class Critic < ActiveRecord::Base
    validates_presence_of :nom
end
------> app/models/restaurant.rb
class Restaurant < ActiveRecord::Base
    validates_presence_of :nom
end
```

- Tenim moltes validacions:

  validates_associated, validates_confirmation_of, validates_format_of, validates_length_of, validates_uniqueness_of, ...

- http://guides.rubyonrails.org/activerecord_validations_

Vinculació de dades

- Ara lligarem crítics i restaurants (via la crítica que facin):

```
script/generate resource resenya restaurant_id:integer
critic_id:integer puntuacio:decimal comentari:string
```

```
rake db:migrate
```

- Important restaurant_id (amb aquest nom exacte) és refereix a l'identificador de cada restaurant.

Vinculació de dades

- Declarem les associacions al model:

```
----> app/models/resenya.rb
class Resenya < ActiveRecord::Base
    belongs_to  :restaurant
    belongs_to  :critic
end
----> app/models/restaurant.rb
class Restaurant < ActiveRecord::Base
    validates_presence_of :nom


    has_many :resenyas
    has_many :critics, :through => :resenyas
end
----> app/models/critic.rb
class Critic < ActiveRecord::Base
    validates_presence_of :nom

    has_many :resenyas
    has_many :restaurants, :through => :resenyas

end
```

Lògica d'aplicació

- Afegim al model (restaurants) dos funcions:

```
----> app/models/restaurant.rb
class Restaurant < ActiveRecord::Base
    validates_presence_of :nom


    has_many :resenyas
    has_many :critics, :through => :resenyas

    def mitja_punts
        resenyas.average(:puntuacio) || BigDecimal("0.0")
    end

    def suspes?
        mitja_punts < 5
    end

end
```

Tests

- Rails proporciona entorn de test unitari. Afegim a test/unit/restaurant_test.rb:

```
----> test/unit/restaurant_test.rb
require 'test_helper'

class RestaurantTest < ActiveSupport::TestCase

  def test_notes
      restaurant = Restaurant.new(:nom=>"Prova")
      restaurant.resenyas.build(:puntuacio=>0.5)
      restaurant.resenyas.build(:puntuacio=>0.5)

      assert restaurant.save
      assert_equal BigDecimal("0.5"), restaurant.mitja_punts
      assert restaurant.suspes?
  end

  test "the truth" do
    assert false
  end
end
```

Plantilles parcials

- Permeten pintar “troços”. Pintarem resenyes sota cada restaurant.
- Creem un fitxer app/views/restaurants/_resenyas.erb
  - _ és important

Plantilles parcials

```
----> app/views/restaurants/_resenyas.erb
<table>
<% for resenya in resenyas -%>
  <tr id="<%= dom_id(resenya ) %>">
    <td>
      <%= link_to resenya.critic.nom, resenya.critic %>
    </td>
    <td align="right">
        <%= number_with_precision(resenya.puntuacio, :precision => 1)  %>
    </td>
  </tr>
<% end -%>
  <tr>
    <td align="right" colspan="2">
      <strong>Mitja</strong>:
        <%= number_with_precision(restaurant.mitja_punts, :precision => 1)  %>
    </td>
  </tr>
</table>
```

Plantilles parcials

- Afegim ara el display de la plantilla parcial. Al final de app/views/restaurants/show.html.erb:

```
----> app/views/restaurants/show.html.erb

[....]

<h3>Resenyes</h3>

<div id="resenyas">
  <%= render :partial => 'resenyas',
             :locals  => {:resenyas=> @restaurant.resenyas,
                          :restaurant => @restaurant} %>
</div>

</table>
```

Sub-Forms

- Afegirem ara l'entrada de resenyes a la pantalla de restaurants. Modificarem la funció show al controlador.

```
----> app/controllers/restaurants_controller.rb
[....]
  # GET /restaurants/1
  # GET /restaurants/1.xml
  def show
    @restaurant = Restaurant.find(params[:id])

    @resenya = Resenya.new
    @critics = Critics.find(:all, :order=>'nom')

    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @restaurant }
    end
  end
[...]
```

Sub-Forms

- Modifiquem la ruta:

```
----> config/routes.rb
[....]

   map.resources :restaurants, :has_many=> :resenyas

[...]
```

- Això fara que interactuem amb les resenyes sempre via un restaurant.

Sub-Forms

- Afegim a sota del show de restaurants:

```
----> app/views/restaurants/show.html.erb
[...]
<h3>Afegir resenya al restaurant</h3>

<p style="color: red"><%= flash[:error] %></p>

<% form_for [@restaurant, @resenya] do |f| -%>
  <p>
    <%= f.collection_select :critic_id, @critics, :id, :nom%>
    puntua amb <%= f.text_field :puntuacio, :size => 3 %>
    <br />
    i comenta <%= f.text_field :comentari %>
  </p>
  <%= f.submit 'Afegir Resenya' %>
<% end -%>
```

Sub-Forms

- Creem métode per controlar creació:

```
----> app/controllers/resenyas_controller.rb
class ResenyasController < ApplicationController

    def create
      @restaurant= Restaurant.find(params[:restaurant_id])
      @resenya = @restaurant.resenyas.build(params[:resenya])

      respond_to do |format|
        if @resenya.save
          flash[:notice] = 'Critica creada OK.'
          format.html { redirect_to @restaurant }
        else
          flash[:error] = @resenya.errors.full_messages.to_sentence
          format.html { redirect_to @restaurant }
        end
      end
    end
end
```

AJAX

- Farem que la plana no recarregui al afegir crítiques i farem animació.
- Afegim a app/views/layouts/restaurants.html, a <head>:

```
<%= javascript_include_tag :defaults %>
```

- Modifiquem la vista: app/views/restaurants/show.html, canviant form_for per remote_form_for:

```
<% remote_form_for [@restaurant, @resenya] do |f| -%>
```

AJAX

- Canviem l'acció create al controlador de resenyes:

```
----> app/controllers/resenyas_controller.rb
class ResenyasController < ApplicationController

    def create
      @restaurant= Restaurant.find(params[:restaurant_id])
      @resenya = @restaurant.resenyas.build(params[:resenya])
      respond_to do |format|
        if @resenya.save
          flash[:notice] = 'Critica creada OK.'
          format.html { redirect_to @restaurant }
          format.js   # Javascript create.js.rjs
        else
          #ELIMINAR flash[:error] = @resenya.errors.full_messages.to_sentence
          format.html { redirect_to @restaurant }
          format.js do
              render :update do |page|
                  page.redirect_to @restaurant
              end
          end
        end
      end
    end
end
```

AJAX

- Creem ara create.js.rjs

```
----> app/views/resenyas/create.js.rjs
page[:resenyas].replace_html :partial => 'restaurants/resenyas',
                             :locals  => {:resenyas=> @restaurant.resenyas,
                                          :restaurant => @restaurant}
page[@resenya].highlight
page[:mitja].highlight
```

- Posem la puntuació en un SPAN amb id = mitja

```
----> app/views/restaurants/_resenyas.erb
<% end -%>
  <tr>
    <td align="right" colspan="2">
      <strong>Mitja</strong>:
      <span id="mitja">
          <%= number_with_precision(restaurant.mitja_punts, :precision => 1)  %>
      </span>
    </td>
  </tr>
</table>
```

Webgrafia i Bibliografia

- Ruby i Ruby on Rails:

  http://rubyonrails.org/

  http://www.ruby-lang.org/es/

  http://www.rubyonrails.org.es
- Programming Ruby 1.9: The Pragmatic Programmers' Guide:

  http://www.pragprog.com/titles/ruby3/programming-ruby-1-9
- Agile Web Development with Rails, Third Edition:

  http://www.pragprog.com/titles/rails3/agile-web-development