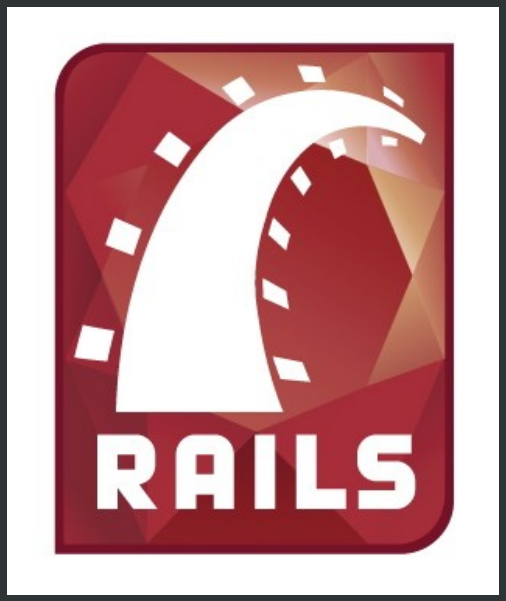
Ruby on Rails con Hobo



Cursos eGhost Julio 2011

Ibon Castilla e Ignacio Huerta

UnoyCero.com

HOBO

¿De qué va esto?

- “Ruby on Rails es un gran avance [...]. Poderosas aplicaciones web que antes habrían llevado semanas o meses desarrollar pueden hacerse en cuestión de días.”
- “Hobo es nuestro intento de responder a una simple pregunta: ¿Hasta dónde podemos llegar?”

Los datos clave de RoR

- Framework libre de aplicaciones web
- Escrito en el lenguaje de programación Ruby
- Sigue la arquitectura Modelo Vista Controlador
- Creado en 2004
- Filosofía: Convención sobre configuración: menos código y más legible
- Active Record abstrae la base de datos: Sqlite, Mysql, PostgreSQL...

Los datos clave de Hobo

- Framework libre que va encima de RoR
- Creado en 2006 por Tom Locke motivado por:
- *¿Otra vez tengo que programar todo eso?*
- Gestión de usuarios y permisos
- Ciclos de vida de los objetos
- Widgets Ajax y tablas inteligentes
- Dryml: Don't Repeat Yourself Markup Language

Reunión 1

- Un amiguete nos ha pedido que hagamos una aplicación para gestionar los cursos de su asociación de tiempo libre.
- Cada **curso** tiene que tener título y fecha.
- Cada **alumno** tiene nombre, teléfono e email y está apuntado a un curso.

Paso 1.1

- Abrimos una terminal y creamos el proyecto:
  - `hobo cursos`
  - `cd cursos`
- Creamos los modelos de curso y alumno:
  - `script/generate hobo_model_resource curso titulo:string fecha:date`
  - `script/generate hobo_model_resource alumno nombre:string email:email_address`
- Generamos nuestra primera migración
  - `script/generate hobo_migration`

Paso 1.2

- Arrancamos el servidor (tenemos un log con colores!)

```
script/server
```

- Abrimos el navegador y vamos a http://localhost:3000
- Añadimos un par de cursos, los editamos, borramos... Las acciones CRUD de los cursos ya funcionan. Ahora viene cuando decís: Ooooo :D
- ¿Y todo esto dónde se guarda? Pues por defecto en sqlite3. Podéis echar un vistazo en `config/database.yml` y a la carpeta `db`.

Paso 1.3

- Ahora vamos a añadir la tabla de alumnos

```
script/generate hobo_model_resource alumno
  nombre:string telefono:string
  email:email_address
```

- Antes de hacer la migración, vamos a relacionar las tablas.
  Editamos *app/models/curso.rb*

```
has_many :alumnos
```

- Y editamos también *app/models/alumno.rb*

```
belongs_to :curso
```

- Y ahora sí, hacemos la migración

```
script/generate hobo_migration
```

Reunión 2

- Estamos muy orgullosos de nuestra aplicación, pero nuestro amiguete enseguida pone pegas:
- Se les ha olvidado un campo en cada curso: quiero añadir una descripción y un lugar
- En la lista de cursos debería aparecer el nombre del curso, no ”Curso 1”
- Cuando añado un nuevo alumno, aparece también ”Curso 1” en vez del nombre del curso. Así no hay quien se aclare.
- En cada curso, quiero poder ver la lista de alumnos apuntados

Paso 2.1

- Vamos a añadir un par de campos a la tabla de cursos. Editamos *app/models/curso.rb*. Dentro de ”fields” añadimos los campos que necesitamos:

```
lugar :string
descripcion :text
```

- Guardamos y hacemos una migración
- En el mismo fichero (el modelo del curso) vamos a marcar el titulo para que se utilice como nombre en la aplicación:

```
titulo :string, :name => true
```

Paso 2.2

- Hobo tiene algunas ”magias” que son realmente útiles. En este caso vamos a ver los ”viewhints” de cada curso: *app/viewhints/curso_hints.rb*:

```
children :alumnos
```

- Guardamos y observamos que en cada curso ya podemos ver los alumnos, y que en cada alumno tenemos una miga de pan de vuelta al curso.
- Como nos ha sobrado tiempo, vamos a hacer que ”Alumno 1” sea el nombre del alumno.

Reunión 3

- El muy desagradecido de nuestro amigo nos sigue pidiendo cosas:
- Hay algún zoquete que pone mal su email, o que directamente no escribe su nombre. Necesitamos **validaciones**.
- Como le han dado una subvención, necesita saber el género de los alumnos: hay que guardarlo como boolean.
- No necesita la pestaña de alumnos, desde cada curso debería poder añadir nuevos alumnos

Paso 3.1

- Ahora vamos a añadir algunas validaciones a cada alumno. En el modelo de alumno:

```
nombre :string, :required, :name => true
validates_presence_of :telefono, :email
```

- Para saber el género de los alumnos no vamos a añadir otra tabla de generos, ya que solo son dos y no varía de forma regular (oficialmente).

```
genero :enum_string(:masculino, :femenino)
```

Paso 3.2

- Ahora vamos a meternos con los controladores, para entender un poco las acciones CRUD: *app/controllers/alumnos_controller.rb*

```
auto_actions :all, :except => :index
```

- Para crear nuevos alumnos tenemos que añadir una acción especial

```
auto_actions_for :curso, :create
```

Reunión 4

- Como ya viene siendo costumbre, nos piden más cosas:
- Los cursos pertenecen a una o varias categorías y las categorías tienen muchos cursos (N a M)
- Y ahora, queremos que el mismo alumno se apunte a varios cursos (cambiar 1 a N por N a M)
- Quieren que junto a la descripción del curso aparezca una pequeña ayuda. Y que ”email” se llame ”correo electrónico”.

Paso 4.1a

- Creamos dos tablas: categorias y categoria_cursos

```
script/generate hobo_model_resource categoria
  nombre:string

script/generate hobo_model categoria_curso
```

- app/models/curso.rb

```
has_many :categoria_cursos, :dependent => :destroy

has_many :categorias, :through =>
  :categoria_cursos, :accessible => true
```

Paso 4.1b

- app/models/categoria.rb

```
has_many :categoria_cursos, :dependent => :destroy
has_many :cursos, :through => :categoria_cursos
```

- app/models/categoria_curso.rb

```
belongs_to :categoria
belongs_to :curso
```

Paso 4.2a

- Ahora vamos a modificar la relación entre cursos y alumnos para ver que no rompemos nada
- Antes que nada dejamos el controlador de alumnos como estaba

```
script/generate hobo_model curso_alumno
```

- *app/models/curso_alumno.rb*

```
belongs_to :curso

belongs_to :alumno
```

Paso 4.2b

- *app/models/curso.rb*

```
has_many :curso_alumnos, :dependent => :destroy

has_many :alumnos, :through => curso_alumnos
```

- *app/models/alumno.rb*

```
has_many :curso_alumnos, :dependent => :destroy

has_many :cursos, :through => curso_alumnos,
  :accessible => true
```

Paso 4.3

- Volvemos a los viewhints para unos detalles:
- *app/viewhints/curso_hints.rb*

```
field_help :descripcion => ”Aquí acordaros de
  meter el horario e instrucciones de cómo llegar”
```

- *app/viewhints/alumno_hints.rb*

```
field_names :email => ”Correo electrónico”
```

Paso 4.4

- Un detalle: nos gustaría poder ver los cursos en cada categoría. A ver cómo lo hacéis :).

Reunión 5

- Nuestro amigo ya está embalado, y nosotros a punto de sufrir un ataque de nervios. Esto nos pasa por hacer las cosas gratis :)
- Las **búsquedas** Ajax no funcionan! ¿Por qué?
- Perfiles de usuario: los **visitantes** solo pueden ver los cursos y categorías, los **usuarios** pueden ver cursos, alumnos y crear nuevos alumnos, los **administradores** pueden hacer todo
- Modificar la **lista de alumnos** de cada curso para que tenga otro aspecto (con DRYML)

Paso 5.1

- Por defecto Hobo viene con una acción que busca en todas las tablas de la aplicación, pero hemos de indicarle sobre qué columnas: app/models/curso.rb:

```
set_search_columns :titulo, :lugar
```

- Podemos hacer lo mismo con los alumnos :)

Paso 5.2

- Un visitante no debería poder ver los alumnos:
  app/models/alumno.rb:

```
def view_permitted?(field)
  acting_user.signed_up?
end
```

- Un usuario registrado puede crear nuevos alumnos:

```
def create_permitted?(field)
  acting_user.administrator? ||
  acting_user.signed_up?
end
```

Paso 5.3

- Abrimos `app/views/taglibs/auto/rapid/cards.dryml`
- Copiamos y modificamos la card de Alumno a `app/views/taglibs/application.dryml`

```
<def tag="card" for="Alumno">
  <card class="alumno" param="default" merge>
    <header: param>
      <h4 param="heading">
        <a><name/></a>:<this.telefono />
      </h4>
    </header:>
  </card>
</def>
```

Reunión 6

- Nos llama por teléfono y nos pide mas cambios: personalizar el menu: que la pestaña alumnos solo aparezca si estás logueado, modificar la home (que muestre el total de cursos y alumnos) y luego que por defecto vayas a la lista de cursos y no a la home
- No quiero que aparezca el Género porque es un dato solo estadístico.
- Quiero modificar el Teléfono sin tener que pulsar Editar.

Paso 6.1

- Para esto modificamos el menú por defecto, nos fijamos en `app/views/taglibs/auto/rapid/pages.dryml` y modificamos así

```
<!-- ====== Main Navigation ====== -->

<def tag="main-nav">

<navigation class="main-nav" merge-attrs param="default">

<nav-item href="#{base_url}/">Home</nav-item>

<nav-item with="&Alumno" if="&logged_in?"><ht
   key="alumnos.nav_item">Alumnos</ht></nav-item>

<nav-item with="&Categoria"><ht
   key="categorias.nav_item">Categorias</ht></nav-item>

<nav-item with="&Curso"><ht key="cursos.nav_item">Cursos</ht></nav-
   item>

</navigation>

</def>
```

Paso 6.2

- Primero el controlador `app/controllers/front_controller.rb`

```
def index
@total_alumnos = Alumno.count
@total_cursos = Curso.count
end
```

- Y luego la vista `app/views/front/index.dryml`

```
<h3>Totales</h3>
<p>El total de alumnos es <%= @total_alumnos %></p>
<p>El total de cursos es <%= @total_cursos %></p>
```

Paso 6.3

- Modificamos las rutas `config/routes.rb`

```
map.root :controller => 'cursos', :action => 'index
```

- Y por otro la vista de las pestañas
  `app/views/front/index.dryml`

```
<nav-item href="/front/index">Home</nav-item>
```

¡¡Mas tareas, que es esto es la guerra!!

Paso 6.4

- Modificamos el `app/views/alumnos/show.dryml`

```
<show-page>
<field-list: skip="genero"/>
</show-page>
```

- Por el mismo precio le cambiamos la edición de Teléfono:

```
<show-page>
<field-list: skip="genero">
<telefono-view:><editor/></telefono-view:>
</field-list:>
</show-page>
```

Reunión 7

- A nuestro amiguete le gustaría que los cursos estuvieran en una tabla, y que pudiera ordenar los cursos por nombre.
- También le gustaría empezar a ver la aplicación en castellano.
- Por último nos cuenta que los cursos tienen dos estados: abierto y cerrado. Entre esos estados pasamos mediante dos botones: ”Abrir curso” y ”Cerrar curso”. Lo que le gustaría también es que al cerrar el curso se guardase la fecha fin de curso.

Paso 7.1a

- Crear fichero `app/views/cursos/index.dryml`

```
<index-page>
  <collection: replace>
    <table-plus: fields="titulo, lugar, fecha">
      <titulo-view:><a with="&this_parent"/></titulo-view:>
    </table-plus:>
  </collection>
</index-page>
```

Paso 7.1b

- Para poder ordenar cambiamos el controlador

  app/controllers/cursos_controller.rb

```
def index
  hobo_index Curso.apply_scopes(
    :search => [params[:search], :titulo],
    :order_by => parse_sort_param(:titulo, :lugar, :fecha)
  )
end
```

Paso 7.2

- Generar el fichero es.yml y dejarlo en la carpeta de idiomas: otra vez, convención sobre configuración :) y modificamos `config/environment.rb`

```
config.i18n.load_path += Dir[Rails.root.join('my', 'locales',
  '*.{rb,yml}')]

config.i18n.default_locale = :es

# HOBO_VERBOSE_TRANSLATIONS = true

# HOBO_SHOW_LOCALE_KEYS = true
```

- Nos llama y nos dice que quiere que la aplicación se llame ”Cursos a mogollón, molan un montón”. Se lo traducimos en un pis-pas:

```
<def tag="app-name">Cursos de tu asociación</def>
```

Paso 7.3a

- En el model de curso `app/models/curso.rb`

```
# --- Ciclo de vida --- #

lifecycle :state_field => :estado do

state :cerrado, :default => :true

state :abierto

transition :abrir_curso, { :cerrado => :abierto },
   :available_to => :all

transition :cerrar_curso, { :abierto => :cerrado },
   :available_to => :all

end
```

- Por último generamos la migración y reiniciamos el servidor: `script/generate hobo_migration`

Paso 7.3b

- Creamos el siguiente fichero para añadir los botones:
  `app/views/cursos/show.dryml`

```
<show-page>
  <after-heading:>
    <transition-buttons/>
  </after-heading:>
</show-page>
```

- Y añadimos el estado en `app/views/cursos/index.dryml`

```
<table-plus: fields="titulo, lugar, fecha, estado">
```

Paso 7.3c

- Para modificar la fecha de cierre del curso, modificamos un poco el modelo `app/models/curso.rb`

  Añadir campo:

```
fecha_cierre :date

...
```

  Y modificar la transición (ciclo de vida):

```
transition :cerrar_curso, { :abierto => :cerrado },
   :available_to => :all do

self.update_attribute(:fecha_cierre, Date.today)

end
```

Reunión 8

- Necesitamos corregir un bug en la validación de nuevos Usuarios desde Cursos: en un paso anterior había un bug que no ejecutaba las validaciones de Alumnos.
- Los cursos abiertos tienen un fondo verde
-
-
- Ajax: al añadir un alumno se actualice la lista por Ajax
- Ajax: ordenar alumnos con Ajax
- Jquery: datepicker

Paso 8.1a

- Controlador:

```
auto_actions_for :alumno, :create
def create_for_orden
  hobo_create_for :alumno do
    render :template => "alumnos/edit" if !valid?
  end
end
def create
  hobo_create do
    render :template => "alumnos/edit" if !valid?
  end
end
```

Paso 8.1b

- Y editamos el DRYML del modelo necesario (en nuestro ejemplo en Alumno): `edit.dryml`

```
<edit-page:>
<form:>
<field-list: skip="orden" />
</form:>
</edit-page:>
```

Paso 8.2

- Modificamos la vista: `app/views/cursos/index.dryml`

```
<estado-view: style="background-
   color:#{this_parent.color};" />
```

- Y el modelo: app/models/curso.rb

```
# --- Asignamos colores a los estados --- #

def color

  case estado

    when "cerrado"

      "#FFF8B0" #Amarillo

    when "abierto"

      "#A5FFAA" #Verde

    else

      ""

  end

end
```

Recursos de Hobo

- **Hobocentral.net**
  - Tutoriales
  - Dos libros
  - Recetas con trucos
  - Lista HoboUsers
- **Puesta en producción (Apache)**
  - modrails.com