DESARROLLO DE APLICACIONES PARA LA WEB CON RUBY ON RAILS3X

Guia: 3
Motor de base de datos: PostgreSQL 9.2.1
Ruby: 1.9.3-p194 en adelante, Rails: 3.2.8 en adelante
Tema: Paginación

TEMAS

- **Instalación y configuración de la gema will_paginate**
- **Validaciones en la entrada de datos**
- **Creación de un módulo para instructores**

1: INSTALACIÓN Y CONFIGURACIÓN DE LA GEMA WILL_PAGINATE

Es muy útil a la hora de desplegar muchos registros en una pagina web poder mostrarlos por páginas, además, hacer una caja donde podamos hacer un filtro básico, como aparece en la imagen siguiente:

Para realizar este ejercicio de paginación realizaremos el ejercicio con un proyecto de prueba, ubicamos la ruta de nuestro proyecto y en la raíz y seguimos los siguientes pasos:

1.1: Generemos un proyecto nuevo

Recuerde crear una carpeta llamada railsapp, o si ya la tiene cree el proyecto dentro de esta, creelo con el nombre de prueba, y si ya tiene uno llamado prueba, cambiele el nombre por prueba2 al que existe.

Genere los siguientes pasos para empezar:

1.2: Ubiquese dentro de la carpeta

cd railsapp

1.3: Cree el proyecto

rails new prueba -d postgresql

1.4: Aplique lo de la guia 2 “Configuración del idioma Español”

1.5: genere el scaffold de Estudiante con Bootstrap y Simple_form(ver guia 1 y 2)

rails generate scaffold estudiante nombres:string cedula:string email:string direccion:string fch_nacimiento:date --skip-stylesheets

1.6: Editamos el archivo llamado Gemfile, lo podemos hacer con gedit, vi o nano, o con su editor preferido(Sublime text 2, geany, notepad ++ ,etc), agregamos la linea:

gem 'will_paginate'

Puede hacerlo En cualquier parte del archivo después de

source 'http://rubygems.org', Pero les recomiendo que lo hagan en la parte inferior.

Con este paso le decimos a rails que nuestro proyecto va a utilizar la gema de paginación.

1.7: Para instalar esta nueva gema en la consola escribimos

ruta_proyecto$ **bundle install**

Esta instrucción instala la nueva gema que matriculamos para el proyecto llamada will_paginate.

1.8: Modelo Estudiante agregamos el siguiente método de cuerdo a su versión de rails DENTRO DE LA CLASE .

```
#para rails 3.2.8 e inferiores
  def self.search(search, page)
     paginate :per_page => 5, :page => page,
          :conditions => ['nombres like ?', "%#{search}%"], :order => 'nombres'
   end
#para rails 3.2.9 hasta 3.2.12(última versión a abril-2013)
```

```
  def self.search(search)
    where('nombres like ?', "%#{search}%")
  end
```

1.9: En el Controller de estudiantes cambie en método index de acuerdo a su versión de rails que esté utilizando por este otro:

```
def index
 #para rails 3.2.8 e inferiores   ->
 @estudiantes = Estudiante.search(params[:search], params[:page])
   respond_to do |format|
    format.html # index.html.erb
    format.xml  { render :xml => @estudiantes }
   end
end
```

ó

```
def index
 #para rails 3.2.9 hasta 3.2.12(última versión a abril-2013)
 @estudiantes = Estudiante.search(params[:search]).page(params[:page]).per_page(5)
   respond_to do |format|
    format.html # index.html.erb
    format.xml  { render :xml => @estudiantes }
   end
end
```

1.10: En la vista index.html.erb de estudiantes agregue después del titulo <h1></h1> las siguientes lineas de código que construyen la caja de consulta:

```
<%= form_tag request.path, :method => 'get' do %>
  <%= content_tag :label do %>
    Nombre Aprendiz:
    <%= text_field_tag :search, params[:search]   %>
    <%= submit_tag "Consultar", :name => nil %>
  <% end %>
<% end %>

<br />
```

Como bien aprecia, casi todo es código html

1.11: Bueno viene la última parte, al final de este mismo archivo index.html.erb agregue las líneas de código que cuenta las páginas:

```
<br />
<br />

<div align="center">
   <%= will_paginate @estudiantes, :previous_label => '&laquo; Anterior', :next_label => 'Siguiente &raquo;' %>
</div>

<p>
  Total Aprendices : <%= @estudiantes.total_entries %>
</p>
```

1.12: Suba el server y pruebe, recuerde que después de subir una gema hay que reiniciar el server.

```
rails s
```

Ingrese en su browser http://localhost:3000/estudiantes

si todo salió bien estará viendo la paginación.

2: VALIDACIONES EN LA ENTRADA DE DATOS

Los datos deben ser válidos a la hora de tenerlos en una base de datos, para ellos nos debemos asegurar de su validación para que sean bien ingresados con los formatos válidos y su correspondiente dato.

Recomiendo leer la siguiente entrada es este blog:

http://lindsaar.net/2010/1/31/validates_rails_3_awesome_is_true

La validación se hace en los modelos:

Validaciones

- :acceptance => Boolean
- :confirmation => Boolean
- :exclusion => { :in => Ennumerable }
- :inclusion => { :in => Ennumerable }
- :format => { :with => Regexp }
- :length => { :minimum => Fixnum, :maximum => Fixnum, }
- :numericality => Boolean
- :presence => Boolean
- :uniqueness => Boolean

Validar nombres que sea obligatorio y que tenga un valor máximo

```
validates :nombres, :presence => true,
        :length => { :maximum => 80 }
```

Validar un email obligatorio, que sea único y que tenga el formato de correo, leer sobre expresiones regulares en ruby http://rubytutorial.wikidot.com/expresion.

```
validates :email, :presence => true,
        :uniqueness => true,
        :format => { :with => /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i }
```

Validar una cédula que sea obligatoria, que tenga una longitud mínima y una longitud máxima, además debe ser numérica.

```
validates :cedula, :presence => true,
        :length => { :minimum => 6, :maximum => 15 },
    :numericality => true
```

Por favor investigue y consulte el api de rails3x:
http://api.rubyonrails.org/
buscando por validations deberá encontrar Ruby on Rails v3.2.13 Module ActiveModel::Validations

2.1: Como tarea debe Validar todos los atributos del modelo estudiantes

3: Tarea

Construya un módulo para instructores y realice todas las validaciones, limpie los controller de comentarios innecesarios y haga la paginación de instructores. Aplique estilos CSS con Boostrap, formularios con Simple_form y aplique lazybox a las operaciones CRUD de ambos formularios.

Instructor

- cedula:string
- nombres:string
- email:string
- direccion:string
- telefono:string
- fch_ncto:date
- modalidad:string

Mucha suerte.....

Paulo Andrés Carmona Zapata
Instructor ADSI
Centro de Formación en Disweño, Confección y Moda, Itaguí
pcarmona@misena.edu.co
V2, actualizado a abril 2013