02

Customizando as camadas MVC para exibir jobs premium

MVC

Nós dizemos que o Rails é um framework **MVC**, ou seja, com **Model**, **View** e **Controller**. Nós estudamos o *Controller* e *View* no segundo capítulo, quando customizamos a nossa página *Hello World*, e aprendemos mais sobre o *Model* capítulo anterior.

Porém, quanto estudamos cada um destes componentes, nós tivemos uma visão isolada de cada um deles. Neste capítulo, nós vamos aprender como as camadas **MVC** se comunicam, de forma que possamos acessar modelos diretamente dos *controllers* e passar esses modelos para a *view*. Esse aprendizado vai ser essencial para podemos criar a página que lista os jobs *premium*, já que precisaremos pegar todos os jobs premium do banco de dados e mostrá-los na página html final.

Controllers e views

Quando executamos o *scaffold generator* e acessamos a uri */jobs* no navegador, foi possível efetuar uma série de ações, através de uris diferentes:

-
- listar os *jobs* existentes: */jobs*;
-
- criar um novo *job*: */jobs/new*;
-
- exibir o *job 1*: */jobs/1*;
-
- editar o *job 1*: */jobs/1/edit*.

Nós também já estudamos o comando `rake routes` que mostra para qual *controller* e *action* cada uri acima aponta. Vamos executar o comando novamente para refrescar a nossa memória:

```
$ rake routes
       root GET    /                       jobs#index
       jobs GET    /jobs(.:format)         jobs#index
            POST   /jobs(.:format)         jobs#create
    new_job GET    /jobs/new(.:format)     jobs#new
   edit_job GET    /jobs/:id/edit(.:format) jobs#edit
        job GET    /jobs/:id(.:format)     jobs#show
            PUT    /jobs/:id(.:format)     jobs#update
            DELETE /jobs/:id(.:format)     jobs#destroy
hello_world GET    /hello/world(.:format)  hello#world
```

Por exemplo, podemos identificar que a segunda rota, `/jobs` , aponta para o *controller jobs* e a *action index*. Além do mais, `/jobs` possui uma rota **nomeada** chamada `jobs` , o que quer dizer que é possível utilizar os métodos `jobs_path` ou `jobs_url` quando for necessário referenciar essa rota.

Vamos abrir o `JobsController` em *app/controllers/jobs_controller.rb*:

```
class JobsController < ApplicationController
  # GET /jobs
  # GET /jobs.json
  def index
    @jobs = Job.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @jobs }
    end
  end

  # GET /jobs/1
  # GET /jobs/1.json
  def show
    @job = Job.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @job }
    end
  end

  # GET /jobs/new
  # GET /jobs/new.json
  def new
    @job = Job.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @job }
    end
  end

  # GET /jobs/1/edit
  def edit
    @job = Job.find(params[:id])
  end

  # POST /jobs
  # POST /jobs.json
  def create
    @job = Job.new(params[:job])

    respond_to do |format|
      if @job.save
        format.html { redirect_to @job, notice: 'Job was successfully created.' }
        format.json { render json: @job, status: :created, location: @job }
      else
        format.html { render action: "new" }
        format.json { render json: @job.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /jobs/1
  # PUT /jobs/1.json
```

```
  def update
    @job = Job.find(params[:id])

    respond_to do |format|
      if @job.update_attributes(params[:job])
        format.html { redirect_to @job, notice: 'Job was successfully updated.' }
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @job.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /jobs/1
  # DELETE /jobs/1.json
  def destroy
    @job = Job.find(params[:id])
    @job.destroy

    respond_to do |format|
      format.html { redirect_to jobs_url }
      format.json { head :no_content }
    end
  end
end
```

O código é um pouco mais extenso do que estamos acostumados até então, mas nós vamos desbravá-lo pouco a pouco. Primeiro, podemos verificar que o controller define 7 *actions* diferentes:

```
def index
def show
def new
def edit
def create
def update
def destroy
```

Por agora, vamos estudar a *action index*, responsável por mostrar a lista de jobs toda vez que acessamos a raiz da nossa aplicação */* ou acessamos */jobs*:

```
  # GET /jobs
  # GET /jobs.json
  def index
    @jobs = Job.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @jobs }
    end
  end
```

A primeira instrução que a *action index* executa é obter todos os jobs, através de `Job.all`, e depois armazena essa informação em uma variável chamada `@jobs`. Logo em seguida, o nosso *controller* chama o método `respond_to` que vai responder à uma requisição baseada em seu formato. Um destes formatos é o **html**, que é exatamente o que nós retornamos para o browser. O outro formato é o **json**, geralmente usado por APIs.

Por enquanto, vamos focar somente no formato **html**. Lembra do "Hello World", que tinha o *controller Hello* e a *action world*? O Rails renderizava por padrão a *view hello/world.html.erb*, e aqui o processo é o mesmo, o Rails vai **por padrão** renderizar uma *view*, que nesse caso está em **app/views/jobs/index.html.erb**. Vamos abrí-la em nosso editor:

```
<h1>Listing jobs</h1>

<table>
  <tr>
    <th>Title</th>
    <th>Description</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @jobs.each do |job| %>
  <tr>
    <td><%= job.title %></td>
    <td><%= job.description %></td>
    <td><%= link_to 'Show', job %></td>
    <td><%= link_to 'Edit', edit_job_path(job) %></td>
    <td><%= link_to 'Destroy', job, method: :delete, data: { confirm: 'Are you sure?' } %></td>
  </tr>
<% end %>
</table>

<br />
<%= link_to 'New Job', new_job_path %>
<br />
<%= link_to 'Hello World', hello_world_path %>
```

A *view* possui conteúdo em **html** com código **ERB** que nós já aprendemos em capítulos anteriores. Observe também que ela está acessando a mesma variável `@jobs` que definimos no controller. Essas variáveis são chamadas **variáveis de instância** e são automaticamente compartilhadas entre o *controller* e as *views*!

A *view* usa a variável `@jobs` para obter todos os jobs e, para cada `job` em `@jobs`, ela imprime informações como o título do job (*title*) e sua descrição (*description*).

Agora que entendemos o que a view *jobs/index.html.erb* faz, já temos uma idéia dos passos que precisamos para criar a nossa própria página de listagem de jobs:

1) Criar uma *action* no `JobsController` chamada `premium`; 2) Criar uma view *jobs/premium.html.erb* que vai ser usada pela nossa nova *action*; 3) Adicionar ou modificar uma rota existente para apontar para *jobs#premium*.

Então mãos à obra!

Premium jobs

O primeiro passo é adicionar uma nova *action* ao `JobsController` chamada `premium`:

```
def premium
  @jobs = Job.where(premium: true).all
end
```

Essa action é bem simples. Nós fazemos uma busca por todos os **jobs** os quais **premium** está marcado como **true** (verdadeiro) e armazenamos essa informação na variável de instância `@jobs`.

Nota: Observe que nós não adicionamos o método `respond_to` como na *action index* porque, por enquanto, vamos nos preocupar apenas com o formato **html**.

Após definir a **action**, vamos criar uma nova view em *app/views/jobs/premium.html.erb*. Vamos usar a *jobs/index.html.erb* como base e simplesmente trocar o título (em `h1`) para "Premium jobs". O resultado final será:

```
<h1>Premium jobs</h1>

<table>
  <tr>
    <th>Title</th>
    <th>Description</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @jobs.each do |job| %>
  <tr>
    <td><%= job.title %></td>
    <td><%= job.description %></td>
    <td><%= link_to 'Show', job %></td>
    <td><%= link_to 'Edit', edit_job_path(job) %></td>
    <td><%= link_to 'Destroy', job, method: :delete, data: { confirm: 'Are you sure?' } %></td>
  </tr>
<% end %>
</table>

<br />
<%= link_to 'New Job', new_job_path %>
<br />
<%= link_to 'Hello World', hello_world_path %>
```

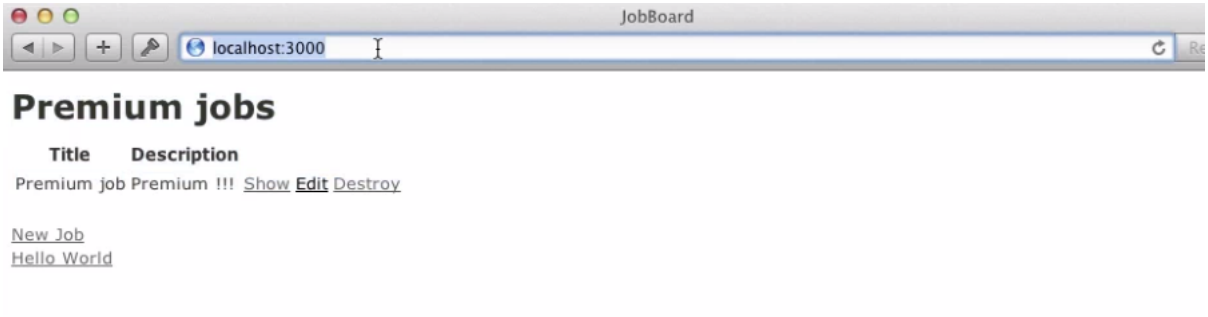
Precisamos também alterar as nossas rotas de forma que alguma url aponte para *jobs#premium*, mas ao invés de adicionar uma nova rota, vamos simplesmente alterar a rota *root* para apontar para *jobs#premium*. Logo, abra o *config/routes.rb* e modifique a linha:

```
root to: "jobs#index"
```

Para:

```
root to: "jobs#premium"
```

Com isso, inicie novamente o seu servidor Rails (caso ele ainda não esteja rodando) e acesse a página inicial da aplicação. Você verá a página de *premium jobs*:



E se acessarmos a página */jobs* (http://localhost:3000/jobs), ainda continuamos vendo a listagem de todos os jobs. Para tornar a navegação mais simples vamos adicionar mais dois links à nossa aplicação. Vamos fazer com que a página de *premium jobs* aponte para a página com todos os jobs e vice-versa. Lembre-se que podemos sempre consultar `rake routes` para relembrarmos quais são os nomes das rotas nomeadas:

```
$ rake routes
       root GET    /                          jobs#premium
       jobs GET    /jobs(.:format)            jobs#index
            POST   /jobs(.:format)            jobs#create
    new_job GET    /jobs/new(.:format)        jobs#new
   edit_job GET    /jobs/:id/edit(.:format)   jobs#edit
        job GET    /jobs/:id(.:format)        jobs#show
            PUT    /jobs/:id(.:format)        jobs#update
            DELETE /jobs/:id(.:format)        jobs#destroy
hello_world GET    /hello/world(.:format)     hello#world
```

Observe que a nossa rota raiz agora aponta para *jobs#premium*, como esperado. Também observe que a rota *nomeada* para *jobs#premium* é **root**, logo devemos usar `root_path` . De forma similar, verificamos que devemos usar `jobs_path` para mostrar todos os jobs.

Com isso, abra novamente o arquivo *app/views/jobs/premium.html.erb* e adicione ao final:

```
<%= link_to 'All jobs', jobs_path %>
```

Salve o arquivo e depois abra o arquivo *app/views/jobs/index.html.erb* e adicione ao final:

```
<%= link_to 'Premium jobs', root_path %>
```

Agora, acesse novamente a sua aplicação e verifique que a navegação funciona como esperado. Excelente!

Don't repeat yourself

Quando discutimos rotas **nomeadas**, nós aprendemos uma filosofia bastante importante do Rails chamada *Don't Repeat Yourself*, ou seja, **não se repita**. Porém, se analizarmos bem as *views jobs/index.html.erb* e *jobs/premium.html.erb*,

podemos verificar que exatamente o oposto está ocorrendo! As *views* são muito similares e grande parte do código em uma *view* se repete em outra.

Toda vez que nos repetimos, nós estamos ignorando um problema que pode aparecer mais tarde. Por exemplo, se quisermos alterar como os *jobs* são mostrados, agora teremos que alterar em duas *views* diferentes. Se nos esquecermos de fazer tais mudanças, o site pode ficar inconsistente e talvez até resultar em páginas de erro!

Logo, para resolver o problema de duplicação em views, o Rails providencia duas ferramentas principais: **partials** e **helpers**. Neste capítulo vamos nos focar apenas na primeira.

Evitando duplicação de views com *partials*

Na verdade, a idéia de *partial* não é nenhuma novidade para nós, já que no capítulo anterior nós usamos e modificamos uma *partial*. Lembre-se como as *views* para criar e editar jobs são bastante similares, e que para adicionar o *check box premium* no capítulo anterior nós editamos um arquivo chamado *app/views/jobs/_form.html.erb*?

Pois é, este arquivo **é** uma *partial*, que contém o formulário de *jobs* e é utilizado tanto pela *view* de criação *jobs/new.html.erb* como de edição *jobs/edit.html.erb*. Se abrirmos *app/views/jobs/new.html.erb* podemos ver o seguinte:

```
<h1>New job</h1>

<%= render 'form' %>

<%= link_to 'Back', jobs_path %>
```

O comando `render 'form'` é exatamente o responsável por renderizar a *partial jobs/form.html.erb. O fato que o nome da view começa com* é um indicador de que o arquivo é uma *partial*.

Porém, não se engane com o nome, uma *partial* é uma *view* como qualquer outra e podemos usar código **ERB** normalmente. A única diferença é que as *partials* geralmente são renderizadas através de outras *views* e não diretamente do *controller*, como ocorre com *jobs/new.html.erb* ou *jobs/index.html.erb*.

Logo, sem mais demora, que tal criarmos uma *partial* para removermos a duplicação de código que existe entre as views **jobs/index.html.erb** e **jobs/premium.html.erb**?

Partial para jobs

Já que a nossa *partial* vai mostrar um job, vamos criar uma *view* em *app/views/jobs/_job.html.erb* (atenção ao underscore!) com o seguinte conteúdo:

```
<h3>
  <%= job.title %>
  <% if job.premium %>
   <strong>(premium)</strong>
  <% end %>
</h3>

<%= simple_format job.description %>
```

```
<p>
  <%= link_to 'Show', job %> |
  <%= link_to 'Edit', edit_job_path(job) %> |
  <%= link_to 'Destroy', job, method: :delete, data: { confirm: 'Are you sure?' } %>
</p>
```

A nossa *partial* mostra agora o título do *job* seguido de "(premium)" se o *job* for marcado como *premium*. Em seguida, nós mostramos a descrição do *job* utilizando um método do Rails chamado *simple_format*. Este método formata a descrição em diversas parágrafos, ao invés de mostrar todo o conteúdo em uma linha só. E por último, nós exibimos os links de *CRUD* para visualizar, editar e remover jobs, como os que tínhamos na *view* **jobs/index.html.erb**.

Agora que a nossa *partial* existe, só precisamos alterar as *views* para utilizarem a nossa *partial*. Abra a *view* **jobs/premium.html.erb** e altere-a para o seguinte:

```
<h1>Premium jobs</h1>

<%= render @jobs %>

<br />
<%= link_to 'New Job', new_job_path %>
<br />
<%= link_to 'Hello World', hello_world_path %>
<%= link_to 'All jobs', jobs_path %>
```

Observe que trocamos todo o código relacionado a gerar uma tabela por simplesmente `<%= render @jobs %>`. Podemos fazer uma modificação similar na página que lista todos os jobs, abra a *view* **jobs/index.html.erb** e altere-a conforme abaixo:

```
<h1>Listing jobs</h1>

<%= render @jobs %>

<br />
<%= link_to 'New Job', new_job_path %>
<br />
<%= link_to 'Hello World', hello_world_path %>
<%= link_to 'Premium jobs', root_path %>
```

Com as views alteradas, acesse novamente a página inicial da sua aplicação e observe que a nossa partial está sendo renderizada com sucesso:

A nossa aplicação está mais elegante e com menos linhas de código que antes! Excelente!

Finalmente, observe que simplesmente usamos `render @jobs` e em nenhum momento tivemos que pedir para o Rails explicitamente utilizar a *partial* em *app/views/jobs/_job.html.erb*. Isso acontece porque, ao passarmos a variável `@jobs` para o método `render`, ele` automaticamente detecta que recebeu uma listagem de jobs e a convenção do Rails é renderizar exatamente: *app/views/jobs/_job.html.erb*. Ou seja, *partials* são um perfeito exemplo de duas filosofias fortes no Rails: **Don't Repeat Yourself** e **Convention over Configuration**.

Para saber mais

-
- Neste capítulo utilizamos o método `simple_format`, você pode aprender mais sobre ele na documentação do Rails (http://api.rubyonrails.org/classes/ActionView/Helpers/TextHelper.html#method-i-simple_format).