02

Expondo os jobs através de uma API

Até esse momento, temos nosso servidor Rails, que atende as requisições que vem de um cliente "da internet". Ou seja, é o usuário que abre o browser e acessa a página web. Mas nem sempre precisa ser assim; uma outra aplicação pode querer consumir nossa aplicação, por exemplo.

Uma das maneiras de distribuir nossa aplicação é usar o que conhecemos por **REST**. Agora estudaremos um pouco de REST, usando Rails. Nosso primeiro passo será listar as empresas cadastradas em nosso sistema, usando XML.

Veja que em nosso `CompaniesController` já temos um método para mostrar o formulário, e outro para adicionar a empresa. Vamos começar com um método que lista uma empresa dado o seu id:

```
def show
  @company = Company.find(params[:id])
  render @company
end
```

Com o método implementado como assim, teríamos renderizado a empresa. Mas como a queremos em XML, avisamos isso para o método `render` :

```
def show
  @company = Company.find(params[:id])
  render :xml => @company
end
```

Precisamos agora configurar a rota. Abrimos o `routes.rb` . Veja que para ":companies", temos só a opção de new e create. Vamos adicionar agora o nosso show:

```
resources :companies, only: [:new, :create, :show]
```

Podemos testar que essa nova rota funciona, usando o `rake routes` no terminal. Se abrirmos o browser e acessarmos, por exemplo, `/companies/1` , vemos que agora nosso browser exibe um XML. Veja que o Rails faz tudo, pega o objeto, o transforma em XML, e manda para o cliente.

Mas temos uma coisa estranha ali. Estamos mandando a senha do usuário. Mesmo que criptografada, não é bom trafegar esse tipo de informação. Vamos então ensinar o Rails a exibir somente os atributos que temos interesse. De volta ao código, basta dizermos quais atributos queremos excluir:

```
    render :xml => @company, except: 'encrypted_password'
```

Pronto. Ao atualizarmos no navegador, não temos mais a senha aparecendo.

E do mesmo jeito que excluímos um campo, podemos adicionar outros. Por exemplo, podemos incluir a lista de empregos. Por padrão, o Rails não inclui no XML atributos "has_many". Vamos lá:

```
render :xml => @company, except: 'encrypted_password', include: :jobs
```

Como é possível passar uma lista de atributos em ambos "except" e "include", podemos padronizar e passar um array para ambos, para deixar isso mais claro:

```
render :xml => @company, except: [:encrypted_password], include: [:jobs]
```

Se buscarmos agora uma empresa que tenha empregos cadastrados, vemos que no XML, o Rails nos mostrará tanto a empresa quanto a vaga de emprego.

Além de XML, é bem comum também transferirmos a informação no formato JSON. Ele é geralmente mais leve, e o preferido entre desenvolvedores web, pois Javascript se integra bem com ele. Para isso, basta trocarmos "xml" por "json":

```
render :json => @company, except: [:encrypted_password], include: [:jobs]
```

Agora pronto. Os dados foram renderizados em JSON. Veja então como é fácil criar serviços web em Rails!