Exame – Padrões de Desenvolvimento

Gabriel Gian, 8° período

Diferenças básicas entre Ruby on Rails e PHP

O **Ruby on Rails** é um *framework* baseado na arquitetura **model-view-controller** (MVC) programado na linguagem de programação interpretada **Ruby**, apresentado famosamente na palestra *How to Build a Blog in 15 Minutes with Ruby on Rails*, ministrada pelo seu criador, David Heinemeier Hansson em 2005. Já a **PHP** (PHP: Hypertext Preprocessor, originalmente Personal Home Page) é uma *linguagem de programação* nascida em 1995 e criada por Rasmus Lerdorf para ser utilizada como uma *template engine*, um motor que geraria templates HTML dinamicamente a partir de tags PHP específicas, tornando-se uma linguagem de programação interpretada muito utilizada no âmbito *back end* para a geração de páginas web, em stacks notórias como a **LAMP** (Linux, Apache, MySQL, PHP).

Ambos Rails e PHP são amplamente utilizados no desenvolvimento web, entretanto, são significativamente diferentes. O Rails é um *framework*, o que significa que ele é um **ecossistema de ferramentas, templates e padrões de desenvolvimento**–pouco configuráveis, uma vez que um dos motes do framework é *convenção sobre configuração*–que permitem a criação extremamente rápida de conteúdo web arquitetado nos padrões MVC *a partir* da linguagem de programação Ruby. Já a PHP é **uma linguagem de programação de propósito geral**, embora seja voltada à programação de sistemas web, que podem ou não seguir os padrões MVC, e que por sua vez *possui frameworks baseados nela*, como **Symfony** e **Laravel**.

Diferenças entre as arquiteturas do Rails e do Laravel

O framework Laravel, baseado na PHP, é um framework que pode se tornar um sistema de aplicações web avançado, uma vez que é composto de uma template engine para a criação de layouts eficiente e relativamente leve e possui uma arquitetura MVC **configurável** e com **diversos built-ins** voltados a segurança e a capacidade de executar múltiplos testes unitários, além de contar com diversas bibliotecas padrão que garantem uma **maior modularidade** e permitem um grau maior de **escalabilidade**.

Já o framework Ruby on Rails, baseado na Ruby, é um framework focado na **criação rápida e eficiente** de serviços web a partir de **convenções**, composto por uma arquitetura MVC **pouco configurável**, padrões enxutos e um grande potencial de **metaprogramação** que auxilia na dinamicidade e coesão do desenvolvimento, além de contar com extras como um coletor de lixo simbólico, argumentos para *keywords*, *turbolinks* para o carregamento mais veloz de conteúdo nas páginas e facilidade na renderização de *views*. Os métodos de desenvolvimento e convenções

propostas pelo Ruby on Rails fazem com que haja uma certa **dificuldade na configuração** do framework para atender a determinadas demandas como as que necessitam de maior escalabilidade, *polêmica tornada notória por engenheiros do Twitter*.

Diferenças entre as *models* do Rails e do Laravel

A camada model, no contexto do Ruby on Rails, é operada **a partir do módulo ActiveRecord**, que é projetado para lidar com todas as operações com bancos de dados da aplicação, como estabelecer a conexão com o servidor do banco de dados, retornar dados a partir de tabelas e armazená-los nas mesmas. Ele inclui **drivers** para diversos bancos de dados, que têm suas tabelas abstraídas de modo que a implementação específica das *queries* é independente do código do framework. O *schema* do banco é mapeado para estruturas de **objetos** e **atributos** Ruby, com inferências de nomes. Sendo assim, é possível criar até mesmo relações entre tabelas:

```
class Story < ActiveRecord::Base
  has_and_belongs_to_many :topics
end
class Topic < ActiveRecord::Base
  has_and_belongs_to_many :stories
end
```

No contexto do Laravel, a camada model é usualmente representada por uma **classe** com propriedades que mapeiam para a estrutura de uma tabela em um banco de dados. Para isto, a classe deve importar e estender o módulo **Model** do **Eloquent**, um mapeador objeto-relacional (ORM) que faz o mesmo processo de abstração do banco de dados que o Rails:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    @var string
    protected $table = 'my_flights';

    @var string
    protected $primaryKey = 'flight_id';

    @var bool
    public $incrementing = false;

    @var string
    protected $keyType = 'string';

}
```

Diferenças entre as *views* do Rails e do Laravel

A camada view, no contexto do Ruby on Rails, é manipulada pelo módulo **ActionView**. O código das views consiste em **templates HTML** com código Ruby embutido, o chamado **embedded Ruby** (ERb). O ERb permite que código do lado do servidor seja embutido no código HTML por meio de tags especiais: `<%= %>`, que são utilizadas para **saídas** comuns de expressões Ruby, e `<% %>`, que são **puramente para execução**—sua saída não surte efeito nas exibições do browser. O módulo ActionView cuida completamente da manipulação das views, sendo a programação necessária **somente nos templates**.

No contexto do Laravel, as views utilizam a template engine **Blade** para executar código PHP em templates HTML utilizando as interpolações com chaves `{{}}` e as tags `<?php >` padrões, assim como anotações com o sigilo `@` que permitem **metaprogramação**, **importações** e **reaproveitamento** de código.

Diferenças entre os *controllers* do Rails e do Laravel

Os controllers do Ruby on Rails são operados pelo módulo ActionController, que é responsável por gerir as regras de negócio da aplicação, agindo como uma junta entre a camada de models e a camada de views. Toda aplicação Rails é gerada com um ApplicationController, que herda do ActionController base, e pode possuir diversos outros controllers para as diferentes regras de negócio.

```
class StoriesController < ApplicationController
  def index
  end

  def show
  end
end
```

Já no Laravel, a camada de controllers é definida em classes específicas que lidam com as diferentes regras de negócio, em arquivos com código PHP comum.

```
<?php

namespace App\Http\Controllers;

use App\Models\User;

class UserController extends Controller
{
    @param  int  $id

    @return \Illuminate\View\View
    public function show($id)
    {
        return view('user.profile', [
            'user' => User::findOrFail($id)
        ]);
    }
}
```

Diferenças entre o *deployment* de apps Rails e Laravel

O deployment de aplicações desenvolvidas utilizando Ruby on Rails é relativamente simples, iniciando com o comando `rails new <nome do app>`, que gera a estrutura de diretórios necessária para o desenvolvimento da aplicação. Após a criação dos modelos, views, controllers e rotas, assim como a instalação e atualização das `gems` necessárias, o banco de dados escolhido pode ser migrado utilizando-se o comando `rake db:migrate`. Por fim, a aplicação pode ser executada e testada com `rails server` ou `rails s`.

Já no caso de aplicações Laravel, a criação da aplicação começa pelo comando `composer create-project laravel/laravel <nome do app>`, que cria o esqueleto de um projeto Laravel. Para o deployment, primeiramente deve-se certificar que o autoloader de classes do **Composer** está otimizado, por meio do comando `composer install --optimize-autoloader --no-dev`. Adicionar um arquivo `composer.lock` para fazer com que a instalação de dependências seja mais granular. Também é uma boa prática otimizar o carregamento de configurações por meio do comando `php artisan config:cache`, assim como o carregamento de rotas com `php artisan route:cache` e de views com `php artisan view:cache`. Por fim, o projeto pode ser visualizado em localhost por meio da sequência de comandos `php artisan key:generate`, para gerar uma `APP_KEY` no arquivo de ambiente `.env`; `php artisan migrate`, que faz as devidas migrações do banco de dados; `php artisan db:seed`, para rodar *seeders*—caso algum esteja sendo utilizado—; e, por fim, `php artisan serve` para que a aplicação comece a servir páginas. O Laravel também oferece a ferramenta **Forge** para facilitar o deployment nas *mais diversas infraestruturas em nuvem*.