



Ruby on Rails

Suzyanne Oliveira



Como você aprende e
sabe tanto de Ruby on
Rails Suzy ?

+You **Search** Images Maps Play YouTube Gmail Drive Calendar Orkut More -

SIGN IN



Google Search

I'm Feeling Lucky

Google.com.br offered in: [português \(Brasil\)](#)

Introdução ao Framework Rails

- o O rails é um framework de aplicação web para a linguagem de programação Ruby.
- o É elaborado, prático, ajudando a construir sites poderosos rapidamente, com código limpo e de fácil manutenção.

O bom framework web

- Framework é uma coleção de bibliotecas e ferramentas feitas para auxiliar o desenvolvimento.
- Ter um bom framework é quase como ter uma parte de sua aplicação pronta. Em vez de começar do zero , você pode partir de uma base já estabelecida.

Surge o Rails

- o Rails é um framework de ponta para construções de aplicações web.
- o É completo, código aberto e tem compatibilidade de plataforma cruzada.
- o Oferece uma poderosa camada de abstração de dados, chamada de Active Record.
- o Vem com um conjunto de considerável de padrões e oferece um sistema comprovado em múltiplas camadas ,para organização e arquivos de programas e interesses.

- O rails é um software de opinião, com uma filosofia que leva a sério a arte do desenvolvimento web. Esta filosofia está centrada em beleza e produtividade.
- Criado originalmente por David Heinmeier , o rails surgiu primeiro como uma aplicação wiki chamada Instiki. A primeira versão ,lançada em julho de 2004,do que hoje é o framework Rails foi extraída de uma aplicação Funcional do mundo real : A Basecamp ,da signals. Os criadores do rails removeram todas as partes específicas da Basecamp, e o que restou foi o Rails.



- ✓ Sua meta como framework é solucionar 80 % dos problemas que ocorrem no desenvolvimento web, presumindo que os 20 % restantes são problemas individuais do domínio da aplicação.
- ✓ Quando você escrever uma aplicação Rails , espera-se que você siga as convenções, em vez de se concentrar nos detalhes sobre como unificar toda a sua aplicação , você pode se voltar aos 20 % que realmente importam.



Rails é Ruby

- Antes do Rails surgir, não havia muitas pessoas escrevendo aplicações Web com Ruby . Outras linguagens como PHP e ASP, eram mais utilizadas , grande parte da Web era construída com elas.
- O fato do Rails usar Ruby é significativo , pois o Ruby é consideravelmente mais poderosa que PHP ou ASP , em termos de suas habilidades com linguagens de programação.

O ruby é a parte chave do êxito do rails. O rails utiliza o ruby para criar o que é conhecido como uma linguagem de domínio específico(Domain Specific Language,ou DSL).

O Ruby foi inventado por Yukihiro Matsumoto em 1994, no que se refere a linguagem de programação ,a ruby está entre as mais belas .



Ruby é ...

- Interpretada;
- Orientada à Obejtos;
- Elegante e Expressiva;
- Poderosos recursos de metaprogramação;

UM POUCO DE
RUBY

Um pouco de Ruby

- Interpretando diretamente um comando

```
PS C:\Users\suzyanne oliveira> irb  
irb(main):001:0> puts"olá!"  
olá!
```

Um pouco de **Ruby**

o Criando uma classe

Em Ruby :

```
class User  
end
```

Em java :

```
public class User {  
}
```


Um pouco de **Ruby**

o Criando um objeto :

Ruby:

```
user = User.new
```

Java:

```
User user = new User ();
```

Note a sintaxe simplificada! Não é necessário declarar , a variável é criada quando utilizada.

Um pouco de **Ruby**

o Variáveis

Local : user

De instância : @user

De Classe: @@user

Variáveis de classe são comparadas a atributos estáticos(Static) em java.

Um pouco de **Ruby**

o Métodos

- *De instância :*

```
def equal?(name)
  @name==name
End
```

- *De Classe :*

```
def self.compare(name1,name2)
  name1==name2
end
```

- *Protegidos e Privados*

Private

```
def method
End
```

Protected

```
def method
end
```


Um pouco de **Ruby**

- Criando métodos de acesso as variáveis :

`attr_accessor` : name

`attr_reader` : age

- `attr_accessor` e `attr_reader` são métodos que geram os métodos `get` / `set` dinamicamente.

Rails estimula a agilidade

- Aplicações web não são conhecidas tradicionalmente por sua agilidade, tendo a reputação de dificultarem o trabalho e de serem de difícil manutenção. Talvez seja como resposta a essa noção que o rails surgiu ,ajudando a impulsionar o movimento em direção a metodologias ágeis de programação no desenvolvimento Web.

Menos Software

- o Um dos princípios centrais da filosofia do Rails é o de menos Software.
- o Utilizar convenção , em vez de configuração , escrever menos código e eliminar pontos que levem desnecessariamente a complexidade de um sistema .
- o Menos software , menos código , menos complexidade , menos bugs.

Conversão em vez de configuração

- Significa que você deve definir apenas configurações que não sejam convencionais.
- O rails permite que inicie imediatamente ,oferecendo um conjunto de decisões inteligentes sobre a forma como seu programa deve funcionar e reduzindo a quantidade de decisões de baixo nível que você deve tomar de início.

Não se repita

- o O Rails se vale muito do princípio DRY(Não se repita, ou don't repeat yourself), que declara que informação em um sistema deve ser expressa em um único local.
- o Quanto mais duplicações existem em um sistema, maior é o espaço para os bugs se esconderem. Quanto maior o número de locais em que reside uma informação, mais a aplicação tem de ser modificada quando uma alteração é necessária, e mais difícil se torna de monitorar essas mudanças.
- o O rails é organizado de uma forma que utiliza ao máximo este princípio. Você especifica informações em um único local e segue para trabalhar em algo mais importante.

Rails é código aberto

- o A cultura do rails está imersa na tradição do código aberto.
- o Seu código-fonte é , evidentemente , aberto.
- o O Rails está licenciado sob uma licença MIT, uma das mais “livres” existentes.

Padrão MVC

- o O Rails emprega um notório e reconhecimento padrão de arquitetura que defende a divisão lógica do trabalho da aplicação em três categorias distintas :
- o Model (Modelo)
- o Controller (Controlador)
- o View (Visualização)

No padrão MVC , o Model representa os dados , a view representa a interface do usuário e o controller dirige toda a ação.



O usuário interage com a interface e dispara um evento

O controller recebe a entrada de dados da interface

GERA EVENTOS

Controller

ALTERAÇÕES

Model

FORNECE OS
DADOS

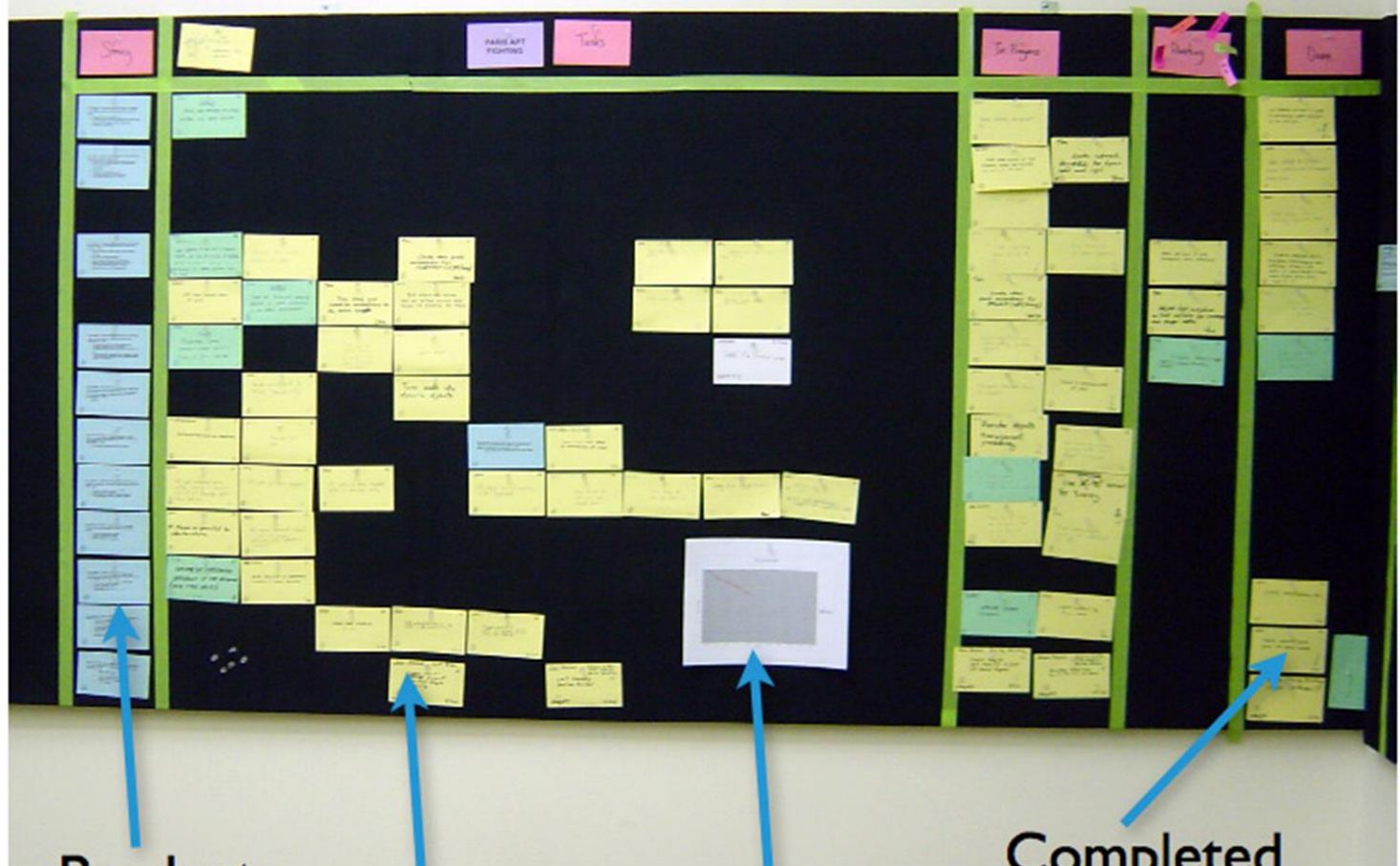
O controller
acessa o model
, atualizando

EXIBE
O STATUS

View

O controller invoca uma view que redimensiona a interface

SCRUM

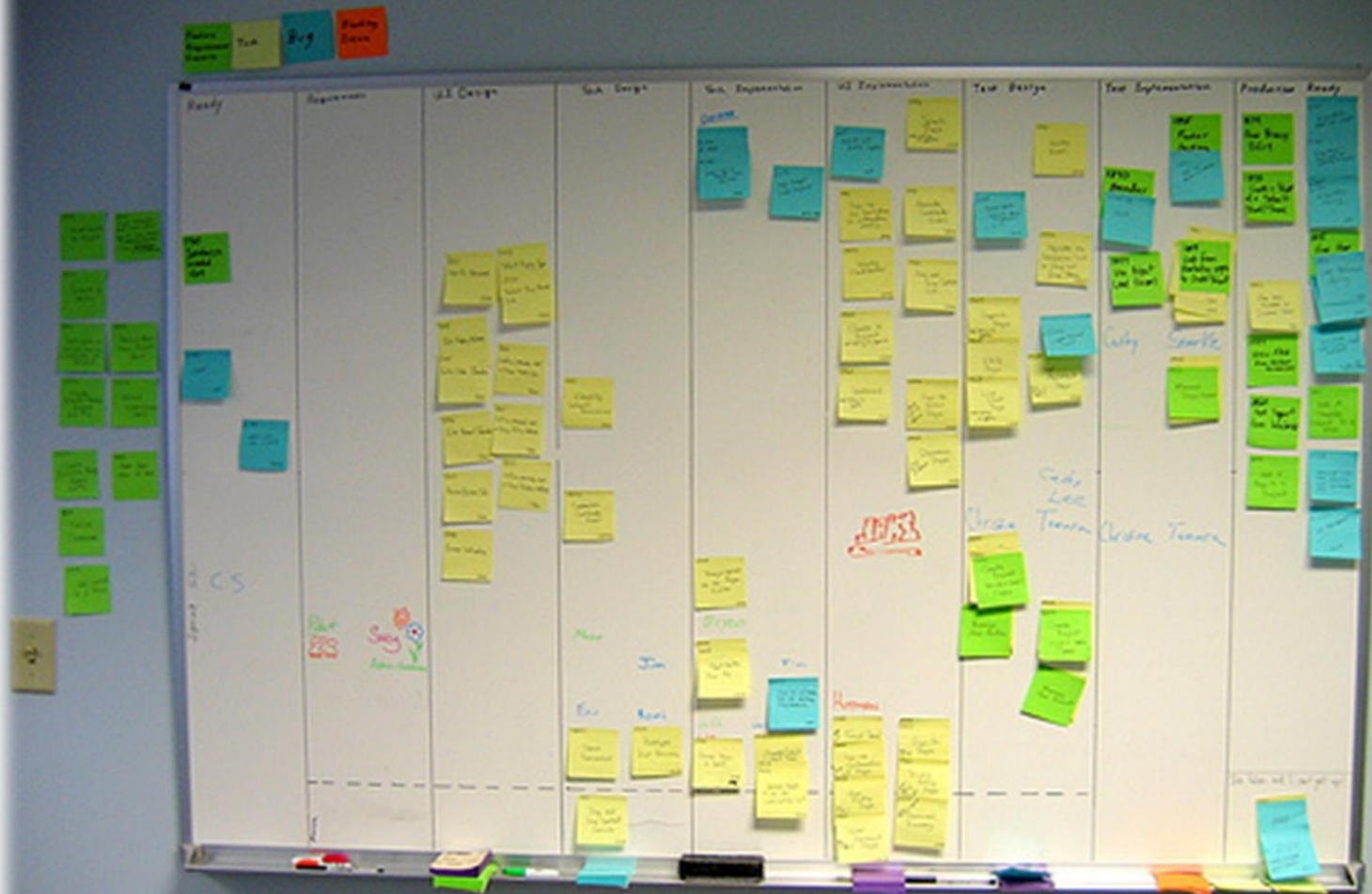


Product
backlog

Tasks
to do

Burndown
chart

Completed
tasks



Ciclo de trabalho do Scrum

