

[IF977] Engenharia de Software

Prof. Vinicius Cardoso Garcia
vcg@cin.ufpe.br :: [@vinicius3w](https://twitter.com/vinicius3w) :: assertlab.com



Licença do material

Este Trabalho foi licenciado com uma Licença

**Creative Commons - Atribuição-NãoComercial-
Compartilhualgual 3.0 Não Adaptada.**

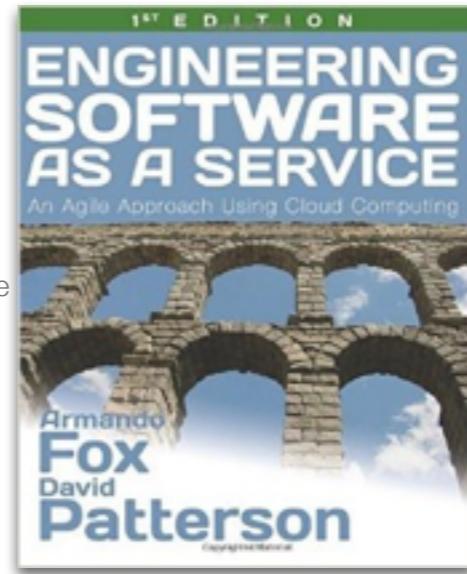
Mais informações visite

<http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt>



Referências

- A biblioteca do Desenvolvedor de Software dos dias de hoje
 - <http://bit.ly/TDOA5L>
- SWEBOK
 - Guide to the Software Engineering Body of Knowledge (SWEBOK): <http://www.computer.org/web/swebok>
- Engineering Software as a Service: An Agile Approach Using Cloud Computing (Beta Edition)
 - <http://www.saasbook.info/>



Utilizando recursos públicos/abertos

- Ruby on Rails Tutorial
 - <http://ruby.railstutorial.org/ruby-on-rails-tutorial-book/>
- Ruby on Rails Guides on rubyonrails.org
 - <http://guides.rubyonrails.org/>
- Railscasts
 - <http://railscasts.com/>
- RailsTestingForZombies@CodeSchool
 - <http://www.codeschool.com/courses/rails-testing-for-zombies>
- Documentação: <http://api.rubyonrails.org/>
- Site de apoio: <http://bit.ly/IF977-Apoio>

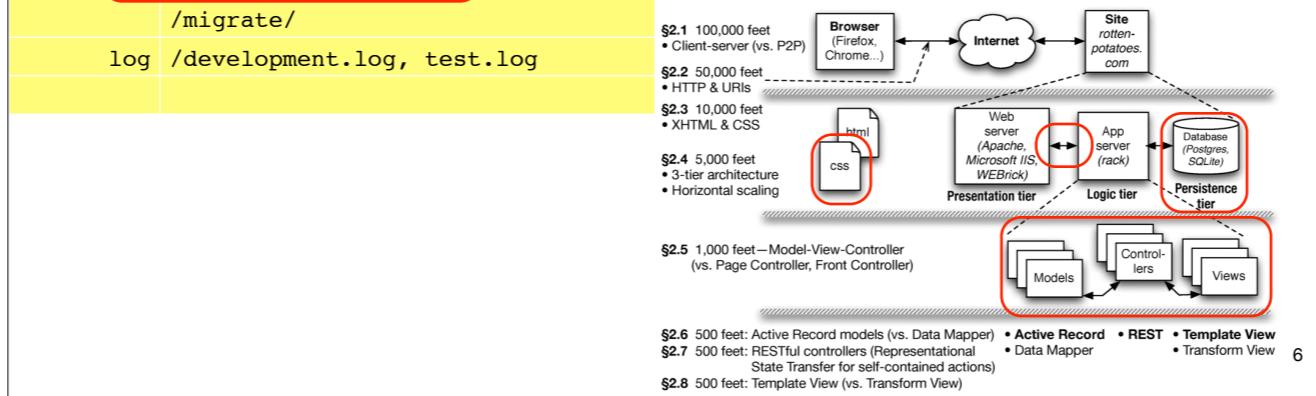
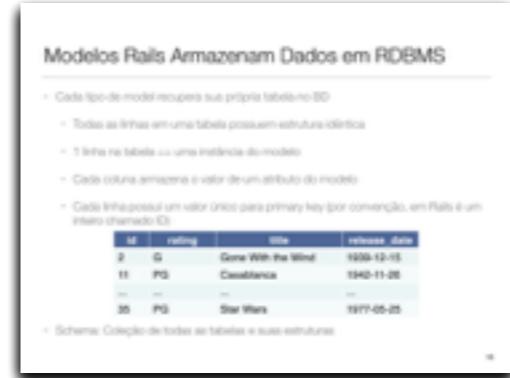


Hello Rails

From Zero to CRUD

Conectando Conceitos Arquiteturais a apps rails

Gemfile	
Rakefile	
app	/models/, views/, controllers/ /helpers /assets/stylesheets/application.css
config	/routes.rb /database.yml
db	/development.sqlite3, test.sqlite3
	/migrate/
log	/development.log, test.log

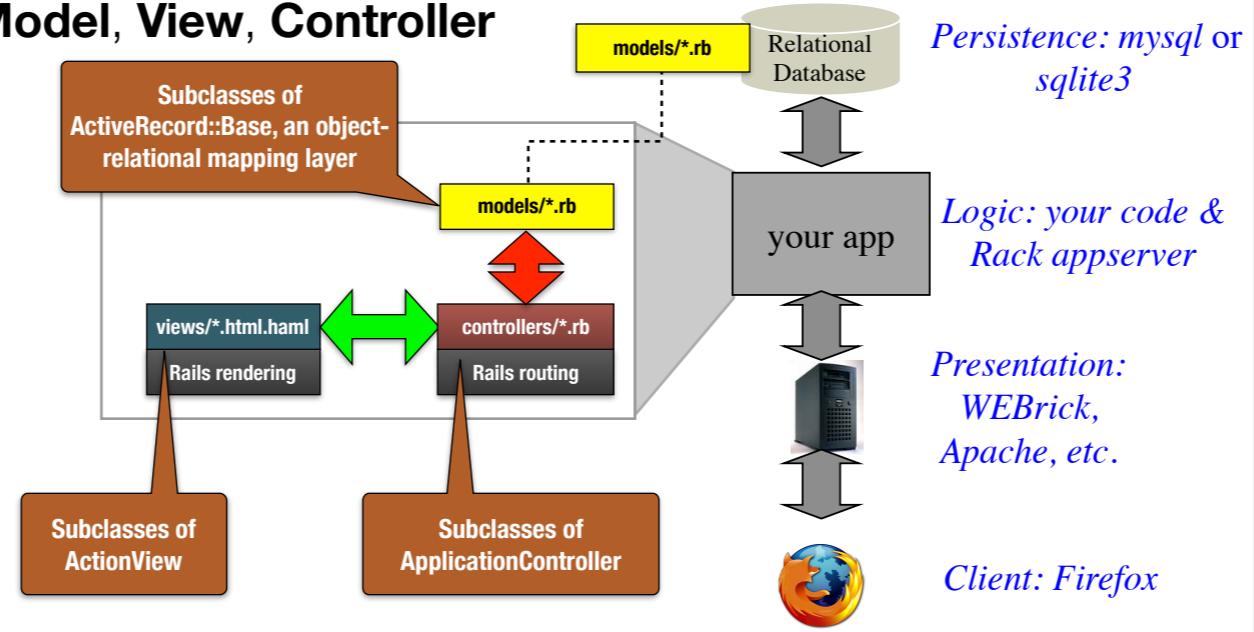


Gemfile: which libraries does app depend on, so can ensure same versions installed in production

Rakefile: you've already used 'rake routes' in book and lecture examples

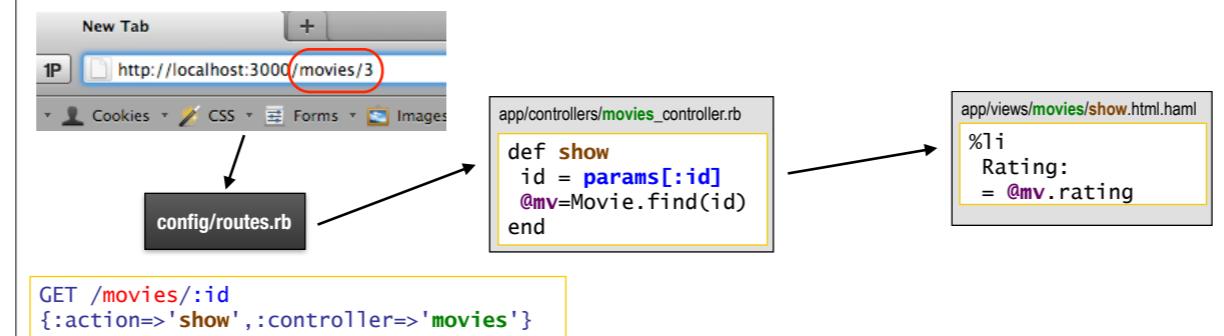
Rails como um Framework MVC

Model, View, Controller



Uma viagem por uma app Rails

- **Routes** (em `routes.rb`) mapeiam URL's para **ações de controladores** e extraem qualquer **parâmetro** opcional
 - Parâmetros “wildcard” da route (`:id`) + qualquer coisa após “?” na URL, são colocados no hash `params[]` que é acessível nas ações controladores
 - Ações controladores definem **variáveis de instância**, visíveis às **views**
 - Subdiretórios e filenames do `views/` casam com os nomes dos controladores & ações
 - Ação controladora eventualmente **processa** uma view



Filosofia Rails

- Convenção sobre configuração
 - Se os nomes seguem certas convenções, não necessitamos de arquivos de configuração
MoviesController#show in movies_controller.rb
→ **views/movies/show.html.haml**
- Don't Repeat Yourself (DRY)
 - Mecanismos para extrair funcionalidades comuns
- Ambos dependem fortemente de recursos do Ruby
 - Introspection & metaprogramming
 - Blocks (closures)
 - Modules (mix-ins)



Pergunta

- Por que cada interação com um aplicativo SaaS, eventualmente, causa algo a ser processado??
 - A. Por causa da convenção sobre configuração
 - B. Por que HTTP é um protocolo requisição-resposta
 - C. Porque o MVC implica que toda ação processa sua própria View
 - D. Todas as anteriores

Pergunta

- Por que cada interação com um aplicativo SaaS, eventualmente, causa algo a ser processado??
 - A. Por causa da convenção sobre configuração
 -  B. Por que HTTP é um protocolo requisição-resposta
 - C. Porque o MVC implica que toda ação processa sua própria View
 - D. Todas as anteriores



<http://www.phdcomics.com/comics/archive.php?comicid=673>

Quando as coisas dão errado

Depuração



Depurar SaaS pode ser complicado

- Nem sempre o terminal (STDERR) está disponível
- Erros no início do fluxo podem se manifestar mais tarde
- **URI -> route -> controller -> model -> view -> render**
- Erros podem ser difíceis de localizar/reproduzir se afetar apenas alguns usuários, rotas, etc

O quê?	Dev?	Prd?
Imprimir no terminal (“printf debugging”)	✓	
Logging	✓	✓
Depuração interativa	✓	

13

In computer programming, standard streams are preconnected input and output communication channels between a computer program and its environment when it begins execution. The three I/O connections are called standard input (stdin), standard output (stdout) and standard error (stderr). Originally I/O happened via a physically connected system console (input via keyboard, output via monitor), but standard streams abstract this. When a command is executed via a shell, the streams are typically connected to the text terminal in which the shell is running, but can be changed with redirection, particularly via a pipeline. However, daemons do not have an associated terminal or standard streams. More generally, a child process will inherit the standard streams of its parent process.

RASP (ou LPBP em pt-br)

- Depuração é um fato da vida
- **L**eia a mensagem de erro. Realmente leia
- **P**eça a um colega uma questão bem formada, esclarecida
- **B**usque informações no StackOverflow (por exemplo)
 - Especificamente por erros envolvendo versões específicas de gems, SO, etc.
- **P**ublique sua dúvida em fóruns, etc.
 - Todo mundo é bastante ocupado, ajude os outros a te ajudar informando o mínimo, mas de forma completa

Lendo mensagens de erro em Ruby

- O registo de chamadas (*backtrace*) mostra a pilha de chamadas (de onde você veio) no ponto de parada
- Uma mensagem muito comum:

```
undefined method 'foo' for nil:NilClass
```

- Muitas vezes, isso significa que uma atribuição falhou e você não verificou o erro

```
@m = Movie.find_by_id(id) # pode ser nil
```

```
@m.title # vai falhar: 'undefined method'
```

Instrumentação (“Imprimindo o valor das coisas”)

- Nas visões:

```
= debug(@movie)
```

```
= @movie.inspect
```

- No log, geralmente a partir de um método de controle:

```
logger.debug(@movie.inspect)
```

- Não use `puts` ou `printf`! Eles não fazem sentido quando estiver em produção

Busca: Use a internet para responder as perguntas

- Google it
 - “How do I **format** a **date** in **Ruby**? ”
 - “How do I **add Rails routes** beyond **CRUD**? ”
- Consulte a documentação
 - api.rubyonrails.org, documentação completa e “pesquisável” do Rails
 - ruby-doc.org, documentação completa e “pesquisável” do Ruby (incluindo bibliotecas)
- Consulte o **StackOverflow**

Use o console do Rails

- Similar ao irb, mas carrega o Rails + o código da sua app
- Mas o contexto pode não estar muito certo para investigar a fundo os controladores e visões
 - Controles atuam no ambiente preparado pela camada de apresentação
 - Visões atuam no contexto configurado pelos controladores

Pergunta

- Se você usar `puts` ou `printf` para imprimir mensagens de depuração em uma app em produção:
 - A. Sua app irá gerar uma exceção e ficará paralisada
 - B. Sua app vai continuar, mas as mensagens irão de perder
 - C. Sua app vai continuar e as mensagens irão para um arquivo de log
 - D. Os Deuses mitológicos do SaaS irão te atacar com uma chuva de raios e trovões em um processo de ódio eterno

Pergunta

- Se você usar `puts` ou `printf` para imprimir mensagens de depuração em uma app em produção:
 - A. Sua app irá gerar uma exceção e ficará paralisada
 -  B. Sua app vai continuar, mas as mensagens irão de perder
 - C. Sua app vai continuar e as mensagens irão para um arquivo de log
 - D. Os Deuses mitológicos do SaaS irão te atacar com uma chuva de raios e trovões em um processo de ódio eterno

```
class Post < ActiveRecord::Base
  has_many :comments
end

class Comment < ActiveRecord::Base
  def self.search(query)
    find(:all, :conditions => [ "body = ?", query ])
  end
end

# SELECT * from comments WHERE post_id = 1 AND body = 'hi'
Post.find(1).comments.search "hi"
```

MODELS: Noções básicas do ActiveRecord

Como os recursos de uma linguagem podem simplificar o projeto e implementação de padrões de projeto?

Como os recursos de uma linguagem podem simplificar o projeto e implementação de padrões de projeto?

Neste caso, o Active Record, que "faz a ponte" entre objetos Ruby na memória a a sua representação armazenada em um banco de dados.

CRUD em SQL

- Structured Query Language (SQL) é a linguagem de consulta utilizada pelos RDBMS's
- Rails gera instruções SQL em runtime, baseado no seu código Ruby
- 4 operações básicas em uma linha de uma tabela:
Create, **R**ead, **U**pdate attributes, **D**elete



"Ted" Codd



```
INSERT INTO users
(username, email, birthdate)
VALUES ("vcg", "vcg@cin.ufpe.br", "1978-09-02"),
"kiev", "kiev@cin.ufpe.br", "????")  
  
SELECT * FROM users
WHERE (birthdate BETWEEN "1987-01-01" AND "2000-01-01")  
  
UPDATE users
SET email = "viniciusgarcia@gmail.com"
WHERE username="vcg"  
  
DELETE FROM users WHERE id=1
```

O lado Ruby de um modelo

- Subclasse de `ActiveRecord::Base`

- “conecta” ao banco de dados

- provê conveniências para manipular o banco de dados

- Nome da classe deve ser igual ao nome do modelo:

- Nome da classe deve ser igual ao nome do modelo

```
1 class Movie < ActiveRecord::Base
2 end
3 # 3 ways to create ActiveRecord objects
4 # (the constructor checks to see what arguments it got)
5 movie = Movie.new
6 movie.title = 'The Help'
7 movie.rating = 'PG-13'
8
9 movie = Movie.new do |m|
10   m.title = 'The Help'
11   m.rating = 'PG-13'
12 end
13
14 movie = Movie.new(:title => 'The Help', :rating => 'PG-13')
```

- **Atenção: os getters & setters não bastam para modificar as variáveis de instância!**

25

<http://pastebin.com/ruu5y0D8>

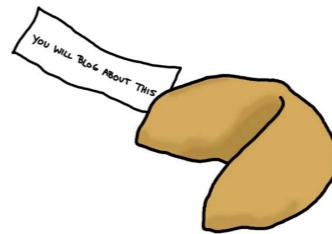
Creating: new ≠ save

- Deve-se chamar `save` ou `save!` em uma instância do modelo AR para realmente salvar as alterações para DB
 - versão `'!'` lança exceção se a operação **falhar**
 - `create` apenas **combina** `new` e `save`
- Uma vez **criado**, o objeto **adquire** uma chave primária (coluna `id` em cada tabela do modelo AR)
 - se `x.id` é nula (`nil`) ou `x.new_record?` é `true`, `x` nunca foi salvo
 - Esses comportamentos são herdados de `ActiveRecord::Base` - não são verdade para objetos Ruby em geral

Pergunta

Considere que a tabela `fortune_cookies` tem uma coluna `fortune_text`. Qual destes métodos de instância de `FortuneCookie < ActiveRecord::Base` não vai retornar um texto da sorte besta (se houver)?

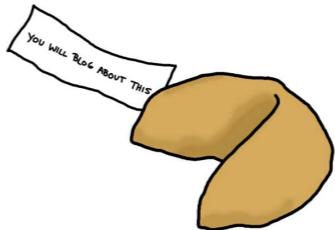
- A `def silly_fortune_1
 @fortune_text + 'in bed'
 end`
- B `def silly_fortune_2
 self.fortune_text + 'in bed'
 end`
- C `def silly_fortune_3
 fortune_text + 'in bed'
 end`
- D Todos irão retornar um texto da sorte besta!

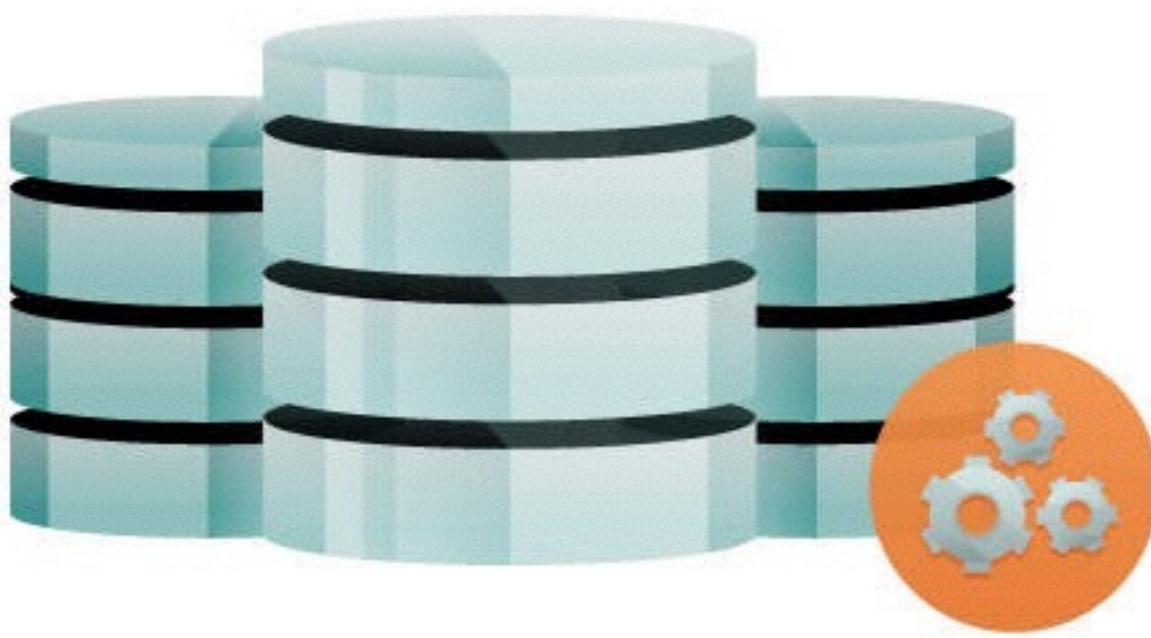


Pergunta

Considere que a tabela `fortune_cookies` tem uma coluna `fortune_text`. Qual destes métodos de instância de `FortuneCookie < ActiveRecord::Base` não vai retornar um texto da sorte besta (se houver)?

- A `def silly_fortune_1
 @fortune_text + 'in bed'
end`
- B `def silly_fortune_2
 self.fortune_text + 'in bed'
end`
- C `def silly_fortune_3
 fortune_text + 'in bed'
end`
- D Todos irão retornar um texto da sorte besta!

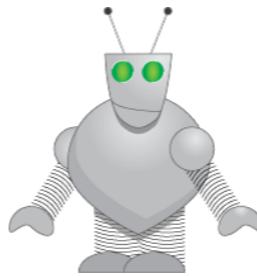




Databases & Migrations

Os dados do seu cliente valem ouro!

- Como é que vamos **evitar estragar tudo** ao experimentar/ desenvolver novas funcionalidades?
- Como é que podemos **controlar e gerenciar** as mudanças de **esquema**?
- ... a resposta para ambas é a automação!

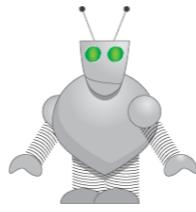


Múltiplos ambientes, múltiplos bancos de dados

- Solução Rails: ambientes de **[development]**, **[production]** & **[test]**, cada um com seu próprio DB
 - Diferentes tipos de DB's apropriados para cada propósito!
 - Como fazer alterações no DB, uma vez que teremos que repetir as alterações no DB de produção?
- Solução Rails: **migration** - ajudam a gerenciar a evolução de um esquema utilizado por diversos bancos de dados
 - Solução para propagar mudanças nos DBs

Migrations :: Vantagens

- Podemos identificar cada migration, e saber onde e quando cada uma é aplicada
 - Podem ser criadas para ser reversíveis
 - Podem ser gerenciadas com controle de versão
 - Automatizada == sistematizada e confiável
-
- Lema: não faça isso, automatize isso!
 - Especifique o que fazer, crie ferramentas para automatizar



Prazer, gerador de código.

`rails generate migration CreateMovies`

<http://pastebin.com/VYwbc5fq>

- Atenção, somente criamos o script de migração, não o aplicamos ainda
- Aplicar a migração ao desenvolvimento:
`rake db:migrate`
- Aplicar a migração à produção:
`heroku rake db:migrate`
- Aplicando a migração também salva no DB quais migrações foram aplicadas

Rails Cookery #1



- Aumentando funcionalidade da app == Adicionando modelos, visualizações e ações do controlador
 - Para adicionar um novo modelo em uma aplicação Rails:
 - (ou mudar/adicionar atributos a um modelo já existente)
1. Crie um script de migração descrevendo as mudanças: **rails generate migration**
 2. Aplique a migração: **rake db:migrate**
 3. Se for um novo modelo, crie o arquivo do modelo
app/models/model.rb
 4. Atualize o esquema de teste do DB: **rake db:test:prepare**

Pergunta

Com base no que já foi visto de Rails, que tipo de objeto é provavelmente sendo rendido (yielded) no código de migração:

```
def up
  create_table 'movies' do |t|
    t.datetime 'release_date' ...
  end
end
```

- A. Um objeto representando um banco de dados
- B. Um objeto representando a instância de um modelo
- C. Um objeto representando uma tabela
- D. Dá um tempo, não deve ser nada!

Pergunta

Com base no que já foi visto de Rails, que tipo de objeto ("t") é provavelmente sendo rendido (**yielded**) no código de migração:

```
def up
  create_table 'movies' do |t|
    t.datetime 'release_date' ...
  end
end
```

- A. Um objeto representando um banco de dados
- B. Um objeto representando a instância de um modelo
-  C. Um objeto representando uma tabela
- D. Dá um tempo, não deve ser nada!

11010010100100010010001001000000111010110100
01001000100100010001000111010101011100010010
011011001010110001000111000111010101011100010010
01101000011000010001011110001000000111011100
001010110001000100010100001110000100100000011100
01000001000000010001010000011100000011100000011100
00011000100010001000101110000100100000011100000011100
110011000100010001000101110000100100000011100000011100
011101000100010001000101110000100100000011100000011100
011100000100010001000101110000100100000011100000011100
011011000100010001000101110000100100000011100000011100
0001010010010001000101110000100100000011100000011100
010010110001101000101110000100100000011100000011100

danger

Modelos: Buscando,
Atualizando e Removendo

Read: buscando coisas no banco de dados

- Método de classe `where` seleciona objetos com base nos seus atributos

```
Movie.where("rating='PG'")  
  
Movie.where('release_date < :cutoff and  
            rating = :rating',  
            :rating => 'PG', :cutoff => 1.year.ago)  
  
Movie.where("rating=#{rating}") # BAD IDEA!
```

- Podem ser encadeados de forma **eficiente**

```
kiddie = Movie.where("rating='G'")  
  
old_kids_films =  
    kiddie.where("release_date < ?", 30.years.ago)
```

Read: find_*

- busque pelo id:

```
Movie.find(3) #exception if not found
```

```
Movie.find_by_id(3) # nil if not found
```

- dynamic attribute-based finders:

```
Movie.find_all_by_rating('PG')
```

```
Movie.find_by_rating('PG')
```

```
Movie.find_by_rating!('PG')
```

Update: 2 maneiras

- Tomemos `m=Movie.find_by_title('The Help')`

- Modificando atributos, e então salvando o objeto

```
m.release_date='2011-Aug-10'
```

```
m.save!
```

- Atualizando os atributos no objeto existente

```
m.update_attributes
```

```
:release_date => '2011-Aug-10'
```

- **Transacional: ou todos os atributos são atualizados, ou nenhum é**

Deleting é simples...

- Lembre-se! `destroy` é um metodo de instância

```
m = Movie.find_by_name('The Help')
```

```
m.destroy
```

- Há também o `delete`, que não provoca retornos de chamada de ciclo de vida, que discutiremos mais tarde (assim, é melhor evitá-lo por enquanto)
- Uma vez que objetos AR são destruídos, você pode acessar o objeto na memória, mas não modificá-lo!

```
m.title = 'Help' # FAILS
```

Resumindo, Introdução ao ActiveRecord

- Subclasse de `ActiveRecord::Base` “conecta” um modelo a um banco de dados
 - `C(save/create), R(where, find), U(update_attributes), D(destroy)`
- Convenção sob configuração mapeia:
 - nome do modelo para nome da tabela do banco
 - getters/setters para colunas da tabela do banco
 - Objeto na memória ≠ linha no banco!
 - `save` deve ser usado para a **persistência!**
 - `destroy` **não destrói a cópia** do objeto na memória!

Pergunta

Suponha que fizemos:

```
movie = Movie.where("title='Amelie'")
```

Então, outra aplicação modificou o título do filme diretamente na tabela do banco. Logo após deste instante, o valor de `movie`:

- A. será atualizado automaticamente, porque um modelo ActiveRecord "conecta" seu aplicativo ao banco de dados
- B. será atualizado automaticamente porque o ActiveRecord usa metaprogramação
- C. não será atualizado automaticamente, mas pode ser atualizado manualmente por re-execução de
`movie = Movie.where("title='Amelie'")`
- D. pode ser indefinida ou dependente de implementação

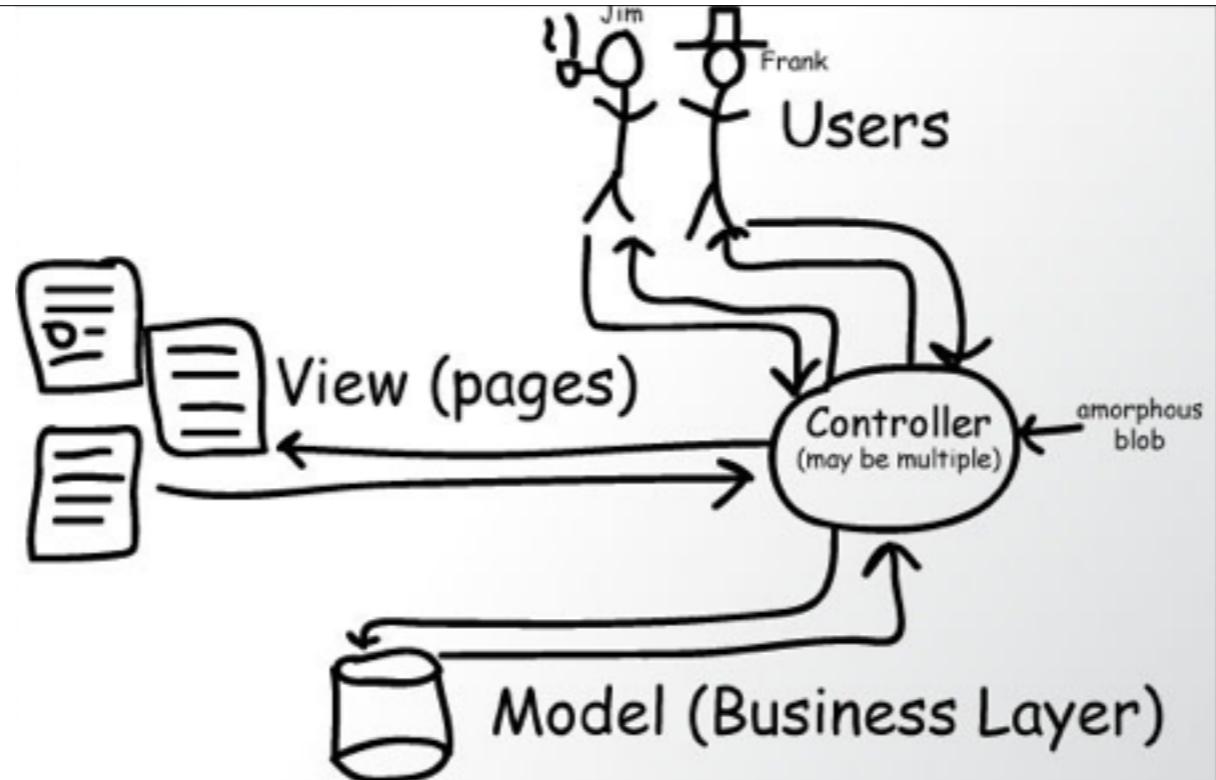
Pergunta

Suponha que fizemos:

```
movie = Movie.where("title='Amelie'")
```

Então, outra aplicação modificou o título do filme diretamente na tabela do banco. Logo após deste instante, o valor de `movie`:

- A. será atualizado automaticamente, porque um modelo ActiveRecord "conecta" seu aplicativo ao banco de dados
- B. será atualizado automaticamente porque o ActiveRecord usa metaprogramação
-  C. não será atualizado automaticamente, mas pode ser atualizado manualmente por re-execução de
`movie = Movie.where("title='Amelie'")`
- D. pode ser indefinida ou dependente de implementação



Controladores e Visões

Rails Cookery #2



- Para adicionar uma nova ação em uma app Rails
 - 1. Crie uma rota em `config/routes.rb` se necessário
 - 2. Adicione a ação (método) no `app/controllers/*_controller.rb` apropriado
 - 3. Garanta que existe alguma coisa para a ação processar em `app/views/model/action.html.haml`

Responsabilidades do MVC

- Model: métodos para recuperar/manipular dados

```
Movie.where(...), Movie.find(...)
```

Variáveis de instância
atribuídas no Controller
ficam disponíveis na View

- Controller: recupera dados do Modelo, disponibiliza

```
def show
  @movie = Movie.find(p)
end
```

- View: exibe os dados

- Mostre o que é necessário

- Mas...

- O que mais o usuário pode fazer

- Como o usuário chegar a essa página

Não encontrando
informações específicas,
procurar em app/views/
show.html.haml

```
-# app/views/movies/show.html.haml

%h2 Details about #{@movie.title}

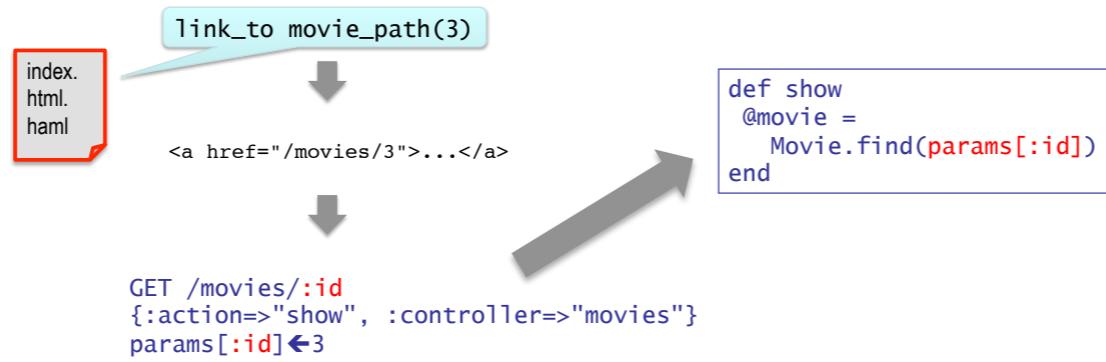
%ul
  %li
    Rating:
    = @movie.rating
  %li
    Released on:
    = @movie.release_date.strftime("%B %d, %Y")

%h3 Description:

%p= @movie.description
```

Como chegamos até aqui: URI assistentes

Helper method	URI returned	RESTful Route and action	
<code>movies_path</code>	/movies	GET /movies	index
<code>movies_path</code>	/movies	POST /movies	create
<code>new_movie_path</code>	/movies/new	GET /movies/new	new
<code>edit_movie_path(m)</code>	/movies/1/edit	GET /movies/:id/edit	edit
<code>movie_path(m)</code>	/movies/1	GET /movies/:id	show
<code>movie_path(m)</code>	/movies/1	PUT /movies/:id	update
<code>movie_path(m)</code>	/movies/1	DELETE /movies/:id	destroy



O que mais podemos fazer?

- Que tal deixar que o usuário retorne à lista de filmes?
- URI RESTful assistente ao resgate de novo:
- `movies_path` sem argumentos conecta à ação Index
`=link_to 'Back to List', movies_path`

Helper method	URI returned	RESTful Route and action	
<code>movies_path</code>	/movies	GET /movies	index
<code>movies_path</code>	/movies	POST /movies	create
<code>new_movie_path</code>	/movies/new	GET /movies/new	new
<code>edit_movie_path(m)</code>	/movies/1/edit	GET /movies/:id/edit	edit
<code>movie_path(m)</code>	/movies/1	GET /movies/:id	show
<code>movie_path(m)</code>	/movies/1	PUT /movies/:id	update
<code>movie_path(m)</code>	/movies/1	DELETE /movies/:id	destroy

49

Pergunta

Quais afirmações são verdadeiras:

- i) Uma rota consiste de **ambos** URI e um método HTTP
- ii) A rota URI **deve** ser gerada por URI assistentes do Rails
- iii) A rota URI **pode** ser gerada por URI assistentes do Rails
 - A. Somente a (i) é verdadeira
 - B. Somente a (iii) é verdadeira
 - C. Somente (i) e (ii) são verdadeiras
 - D. Somente (i) e (iii) são verdadeiras

Pergunta

Quais afirmações são verdadeiras:

- i) Uma rota consiste de **ambos** URI e um método HTTP
- ii) A rota URI **deve** ser gerada por assistentes URI do Rails
- iii) A rota URI **pode** ser gerada por assistentes URI do Rails
 - A. Somente a (i) é verdadeira
 - B. Somente a (iii) é verdadeira
 - C. Somente (i) e (ii) são verdadeiras
 - D. Somente (i) e (iii) são verdadeiras

