

Введение в Ruby on Rails

Владимир Пенкин
@mindwork
<http://shell.github.com>



Ruby on Rails

- “Ruby on Rails - open source web-фреймворк который оптимизирован под облегчения жизни программиста и поддержания его постоянной продуктивности. Он позволяет вам писать великолепный код используя всю мощь ruby а так же следуя пути Rails.

- www.rubyonrails.org

План

- Кратчайшая история Rails
- Путь Rails
- Компоненты Rails
- Agile и Rails
- Скринкаст (блог или todo-лист)

История Rails



- Ruby on Rails был создан Дэвидом Хейнемеером Ханссоном на основе его работы над средством управления проектами Basecamp и открыт сообществу в 2004 году.
- 23 декабря 2008 года команда проекта Merb объединилась с командой Rails с целью создания следующей версии Rails 3, которая объединит в себе лучшие черты обоих фреймворков.
- 29 Августа 2010 года вышел Rails 3.0

Путь Rails

- Соглашения по конфигурации
- DRY(не повторяйся, не повторяйся)
- Всё приложение - Ruby код(SQL и Javascript абстрагированы)
- Встроенный AJAX
- Веб сервисы - RESTful
- Тестирование в коробке

Путь Rails

means that Rails makes assumptions about what you want to do and how you're going to do it, rather than letting you tweak every little thing through

- Соглашения по конфигурации
 - ★ Соглашения о именовании таблиц в базе
 - ★ Соглашения о именах файлов
 - ★ Фиксированная структура директорий(названия контроллеров)
 - ★ Минимум настроек

=>

меньше кода, легче поддержка

Путь Rails

- DRY (Don't Repeat Yourself)

Один и тот же код в 15 разных местах? а если нужно что-то изменить?!

- ★ Повторное использование данных

незачем объявлять атрибуты классов, когда они могут быть прочитаны из БД или из файла схемы `schema.rb`

- ★ Повторное использование кода

В представлениях `views`, в хелперах

- ★ Метапрограммирование

Динамические методы

Путь Rails

- Тестирование в коробке

“Не то чтобы Rails заставляет вас применять test-driven development, просто здесь его его сложнее не делать.”

—Brian Eng, В интервью ‘Rails podcast’

- Все приложение - Ruby код(SQL и Javascript абстрагированы)

```
page.insert_html :bottom, :comments, :partial => @comment
page[@comment].visual_effect :highlight
page[:new_comment].reset
```


Путь Rails

- Встроенный AJAX

```
<% form_for [@post, Comment.new] do |f| %> |
```

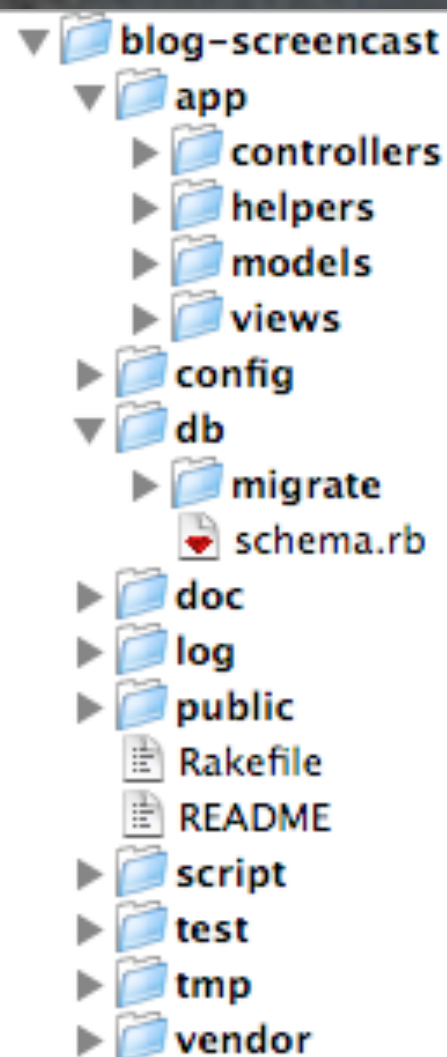
```
<% form_for [@post, Comment.new], :remote => true do |f| %>
```

- Веб сервисы - RESTful

Вся система разделена на ресурсы, которые адресуются соответствующими ссылками. Все ресурсы имеют общий интерфейс обращения и predetermined действия над ними. Соединение между сервером и клиентом не имеет состояния. Формат представления одних данных может быть разным(html, json, xml)

Структура папок приложения Rails

```
MacBook-Pro: [code]:$ rails new blog-screencast
```



Компоненты Rails

- Active Record
- Action Controller
- Action View
- Action Mailer
- Active Resource
- Active Support

ActiveRecord

- Active Record
 - ★ Отображение БД в Ruby классы(таблицы/ колонки => классы/атрибуты)
 - ★ Соглашения о именовании(таблицы/id/ foreign_key)
 - ★ Динамические get, set, поиск
 - ★ Абстрагированные операции с БД (CRUD, find)

Прежде чем таблица
появится в БД,
рассмотрим такую
штуку как миграции

ActiveRecord

Миграции

```
class CreatePosts < ActiveRecord::Migration
  def self.up
    create_table :posts do |t|
      t.string :title
      t.text :body

      t.timestamps
    end
    add_index :posts, :id
  end

  def self.down
    drop_table :posts
  end
end
```


ActiveRecord Запросы

- Find, where, join, destroy, create

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

```
Post.all  
Post.find(:id)  
Post.first  
Post.join(:comments).where(['created_at > ?', Date.today])  
Post.join(:comments => {:users => :address})  
Post.find_by_title("Ruby on Rails")  
Post.create({:title => "Ruby on Rails", :text => "Rollin' Rollin'"})
```


ActiveRecord::Associations

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

```
class Comment < ActiveRecord::Base
  belongs_to :post
end
```

```
class Physician < ActiveRecord::Base
  has_many :appointments
  has_many :patients, :through => :appointments
end
```

```
class Appointment < ActiveRecord::Base
  belongs_to :physician
  belongs_to :patient
end
```

```
class Patient < ActiveRecord::Base
  has_many :appointments
  has_many :physicians, :through => :appointments
end
```


ActiveRecord

- Validations

`validates_presence_of`

`validates_length_of`

`validates_format_of`

`validates_numericality_of`

`validates_inclusion_of`

`validates`

- Callback

`before_save`

`after_save`

`before_create`

`after_create`

`before_validation`

`before_destroy`

Action View

- Erb, Haml, Markdown, Textile
- Билдеры XML, json, RJS
- Паршиалы

Action View::Erb

```
<h1>Listing comments</h1>

<table>
  <tr>
    <th>Post</th>
    <th>Body</th>
  </tr>

  <% for comment in @comments %>
    <tr>
      <td><%=h comment.post_id %></td>
      <td><%=h comment.body %></td>
      <td><%= link_to 'Show', comment %></td>
      <td><%= link_to 'Edit', edit_comment_path(comment) %></td>
      <td><%= link_to 'Destroy', comment, :confirm => 'Are you sure?',
    </td>
    </tr>
  <% end %>
</table>

<br />

<%= link_to 'New comment', new_comment_path %>
```


Action View::Partials

- `_post.html.erb`

```
<% div_for post do %>
  <h2><%= link_to_unless_current h(post.title), post %></h2>
  <%= simple_format h(post.body) %>
<% end %>
```

- `index.html.erb`

```
<h1>Listing posts</h1>

<%= render :partial => @posts %>

<%= link_to 'New post', new_post_path %>
```

ВОТ ЭТО DRY

ActionController

- действие это публик метод соответствующий файлу в представлении
- Стандартный роутинг
<http://localhost.ru/users/show/7>
- callbacks
- простой доступ к кукам, сессии, и параметрам запроса

ActionController

```
class PostsController < ApplicationController
  before_filter :authenticate, :except => [:index, :show]

  # GET /posts
  # GET /posts.xml
  def index
    @posts = Post.find(:all)

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @posts }
      format.json { render :json => @posts }
      format.atom
    end
  end
end
```

ActionController

DRY

<pre>def delete @post = Post.find(params[:id]) . . . def update @post = Post.find(params[:id]) . . . def show @post = Post.find(params[:id]) end def edit @post = Post.find(params[:id]) end</pre>	<pre>before_filter :find_post, :except => [:index, :new, :create] def delete . . . def update . . . def show; end def edit; end private def find_post @post = Post.find(params[:id]) end</pre>
	<p>=> w00t! w00t! =></p>

Рассмотрели
Контроллер, теперь
рассмотрим как же
идет обращение к
контроллеру

Маршрутизация

```
Blog::Application.routes.draw do
```

```
  resources :users
```

```
  resources :posts, :has_many => :comments
```

```
  root :to => "posts#index"
```

```
end
```

Опять же валидный
ruby-код

Маршрутизация

resources :photos

Verb	Path	action	used for
GET	/photos	index	display a list of all photos
GET	/photos/new	new	return an HTML form for creating a new photo
POST	/photos	create	create a new photo
GET	/photos/:id	show	display a specific photo
GET	/photos/:id/edit	edit	return an HTML form for editing a photo
PUT	/photos/:id	update	update a specific photo
DELETE	/photos/:id	destroy	delete a specific photo

Agile и Rails

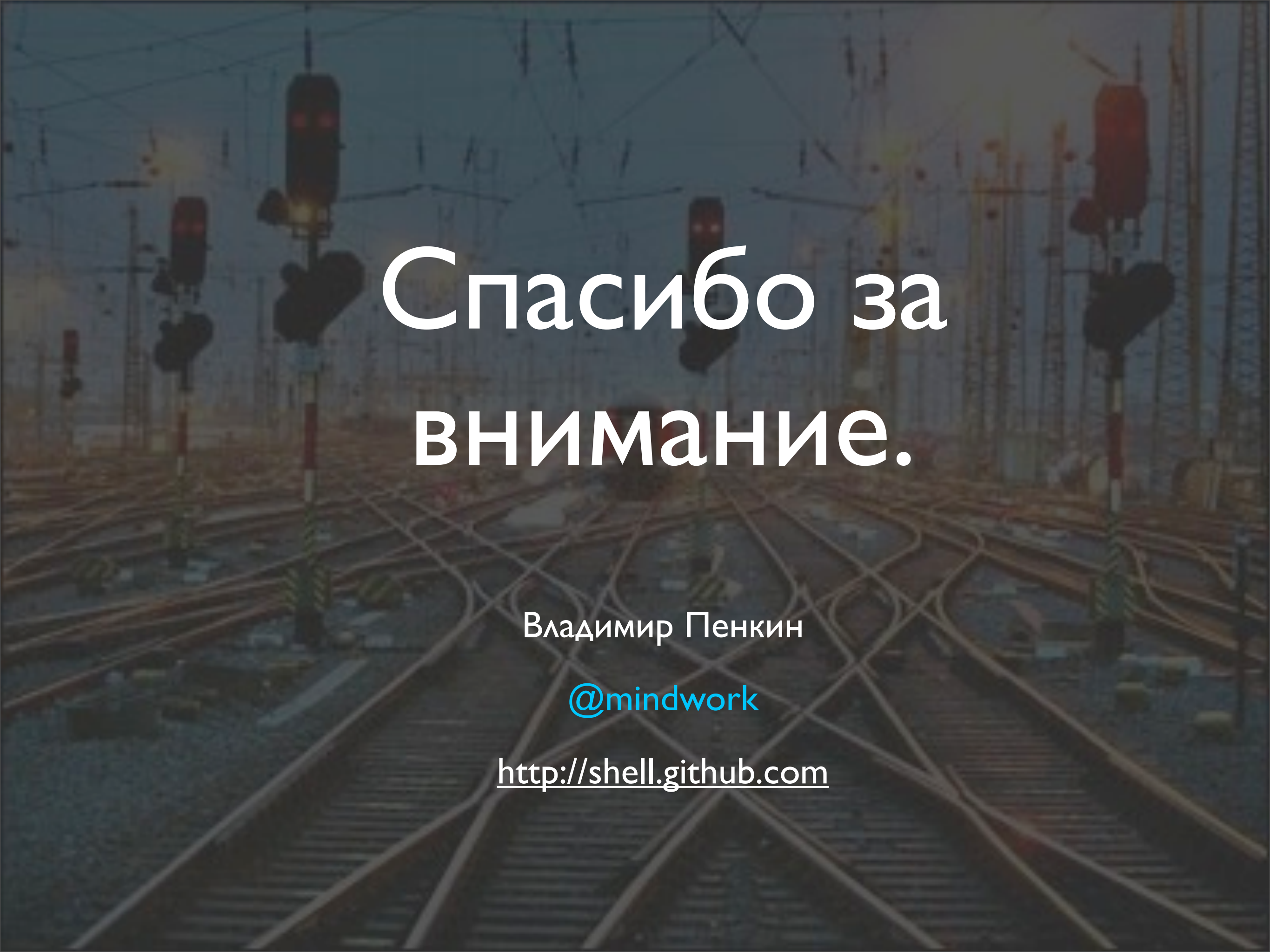
- Встроенный веб-сервер
- Мощные генераторы
- Скаффолдинг
- плагины, gems
- debugger (если что-то пошло не так)
- автоматическое развертывание-
capistrano/heroku
- Rake

Agile и Rails

- rails generate
- rails console
- rails server
- rails plugin (for those who haven't heard of bundler)

Community и Документация

- <http://rubyonrails.org/>
- <http://guides.rubyonrails.org/>
- <http://apidock.com/>
- <http://railscasts.com/>
- <http://ruby-toolbox.com/>



Спасибо за
внимание.

Владимир Пенкин

[@mindwork](#)

<http://shell.github.com>

Даже врачи
могут писать на
Ruby on Rails.

