Ruby On Rails

Boris Nadion
astrails.com
Ruby On Rails
why Ruby on Rails?
it is beautiful
it is almost pure English
read the **english**,

not a code
Account.transaction(quentin, alice) do
  quentine.withdraw(100)
  alice.deposite(100)
end
class Account < ActiveRecord::Base
  validates_presence_of :subdomain, :name, :email, :password
  validates_uniqueness_of :subdomain
  validates_acceptance_of :terms_of_services, :on => :create
  validates_confirmation_of :password, :email, :on => :create
end
class Project < ActiveRecord::Base
  belongs_to :portfolio
  has_one  :project_manager
  has_many  :milestones
end
Project.last.milestones.create({:due_date => 10.days.from_now, :name => "html integration"})
class UsersController < ApplicationController
  before_filter :login_required, :only => [:edit, :update]
  session :off, :only => :feed
end
really beautiful, isn’t it?
"When I am working on a problem, I never think about beauty ... but when I have finished, if the solution is not beautiful, I know it is wrong."

Buckminster Fuller
ruby
ruby - is a language
• created by Yukihiro “matz” Matsumoto

• initial public release at 1995

• free to use, copy, modify, and distribute
www.ruby-lang.org
www.jruby.org
ruby runs on desktops
ruby runs on servers
rails
rails - is a framework
• created by David Heinemeier Hansson

• initial public release at 2003

• more then 1400 contributors
a lot of ways
really really good framework to create in a simple, easy, and a beautiful way
productivity
easier and faster
less code
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

public class TimeServer {
    private static class TellTime extends Thread {
        private Socket soc;

        public TellTime(Socket soc) {
            super();
            this.soc = soc;
        }

        public void run() {
            try {
                this.soc.getOutputStream().write(new Date().toString().getBytes());
            } catch (Exception e) {
            } finally {
                try {
                    this.soc.close();
                } catch (IOException e1) {
                }
            }
        }
    }

    public static void main(String args[]) throws Exception {
        ServerSocket server = new ServerSocket(12345);
        while (true) {
            new TellTime(server.accept()).start();
        }
    }
}
less code - less bugs
less code - less maintenance
less effort
perfect for web startups
maintainability
• make it quickly
• ask your users
• change and update often
get to the point
MVC
model
controller
view
separation of concerns
business logic in models
data representation in views
requests handling in controllers
opinionated
rails tells you how the things should be done
convention over configuration
you can always do it in your way
if you’re stuck
probably you are doing something wrong
MVC
ActiveRecord - model

http://en.wikipedia.org/wiki/Active_record_pattern
talks with database via adapters
all major DBs

- MySQL
- SQLite
- PostgreSQL
- MS SQL
- DB2
- Oracle

and many others
ActiveRecord builds complex SQL queries for you, efficiently.
three environments

- development
- production
- test
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: mysql
  username: sqluser
  password: ilovemysql123
  encoding: utf8
  socket: /tmp/mysql.sock
company = Company.new
company.name = "Astrails"
company.save
company.id # => 12

user = company.employees.create(:name => "Boris")
user.name # => "Boris"
associations
one-to-one
one-to-many
many-to-many
class Project < ActiveRecord::Base
  belongs_to :portfolio
  has_one :project_manager
  has_many :milestones
  has_many :tasks, :through => :milestones
end
validations
class Account < ActiveRecord::Base
  validates_presence_of :subdomain, :name, :email, :password
  validates_uniqueness_of :subdomain
  validates_acceptance_of :terms_of_services, :on => :create
  validates_confirmation_of :password, :email, :on => :create
end
callbacks
class Account < ActiveRecord::Base

  attr_accessor :password

  def before_create
    self.crypted_password = crypt(password)
  end

end
before_update
after_create
before_validation_on_update
after_find
....
migrations
class RemoveUnnecessaryItemAttributes < ActiveRecord::Migration
  def self.up
    remove_column :items, :incomplete_items_count
    remove_column :items, :completed_items_count
  end

  def self.down
    add_column :items, :incomplete_items_count
    add_column :items, :completed_items_count
  end
end
many other features
controller
class ProjectsController < ActionController::Base
  
def create
    @project = Project.new(params[:project])
    if @project.save
      flash[:notice] = "Project created"
      # The entry was saved correctly, redirect to index
      redirect_to projects_path
    else
      flash.now[:notice] = "Fix errors and try again"
      render :action => :new
    end
  end
end
CRUD

http://en.wikipedia.org/wiki/Create,_read,_update_and_delete
# create
def create
def show
def update
def destroy
# create
def create
def create
end

def show
def show
end

def update
def update
end

def destroy
def destroy
end

# form for new object
def new
def new
end

def edit
def edit
end

# list of objects
def index
def index
end
unique URL for every action
GET /books               { :action=>"index" , :controller=>"books" }
POST /books               { :action=>"create" , :controller=>"books" }
GET /books/new               { :action=>"new" , :controller=>"books" }
GET /books/:id/edit      { :action=>"edit" , :controller=>"books" }
GET /books/:id               { :action=>"show" , :controller=>"books" }
PUT /books/:id               { :action=>"update" , :controller=>"books" }
DELETE /books/:id             { :action=>"destroy" , :controller=>"books" }
filters
class UsersController < ApplicationController

  before_filter :login_required, :only => [:edit, :update]

protected
  def login_required
    # ...
  end

end
web APIs
def index
    @people = Person.find(:all)

    respond_to do |format|
        format.html
        format.xml { render :xml => @people.to_xml }
        format.js { render :json => @people.to_json }
    end
end
views
templates
• HAML
• ERB
• Liquid
• Amrita
• Markaby
• HAML
• ERB
• Liquid
• Amrita
• Markaby
User = @user.name

<ul>
  <li>Age = @user.age</li>
  <li>Interests = @user.interests.join(', ', )</li>
  <li>Occupation = @user.occupation</li>
</ul>
User <%= @user.name %>

<ul>
  <li>Age: <%= @user.age %></li>
  <li>Interests: <%= @user.interests.join(', ') %></li>
  <li>Occupation: <%= @user.occupation %></li>
</ul>
helpers
simplify html generation
<% form_for @person do |f| %>
  <%= f.text_field :first_name %>
  <%= f.text_field :last_name %>
  <%= f.submit 'Create' %>
<% end %>

<form action="/people/create" method="post">
  <input id="person_first_name" name="person[first_name]" type="text" />
  <input id="person_last_name" name="person[last_name]" type="text" />
  <input name="commit" type="submit" value="Create" />
</form>
xml
# Builder

```ruby
xml.instruct! :xml, :version=>'1.0'
xml.instruct! :rss, :version=>'2.0'
xml.channel{
  for hat in @hats
    xml.hat do
      xml.title(hat.name)
      xml.description(hat.description)
    end
  end
}
```
putting all together
short demo
test your code
JUnit
RSpec
describe "Sum computation" do
  it "should return 2" do
    (1 + 1).should == 2
  end
end
describe "Sum computation" do
  it "should return 2" do
    (1 + 1).should == 2
  end
end

> spec ./spec/basic_test.rb

.
Finished in 0.040745 seconds
1 example, 0 failures
describe "Sum computation" do
  it "should return 2" do
    (1 + 2).should == 2
  end
end

desc...
describe User do
  before(:each) do
    @user = User.new
  end

  it "should be invalid without a username" do
    @user.should_not be_valid
    @user.username = 'someusername'
    @user.should be_valid
  end
end
TDD & BDD

http://en.wikipedia.org/wiki/Test-driven_development
http://en.wikipedia.org/wiki/Behavior_Driven_Development
lucky
software
Ruby On Rails
• MVC
• maintainability
• time to market
• lower development cost
• fun
what's next?
• download and install ruby
• download and install rails
• build your rails application
welcome to the club
thank you
q&a