Web development with Ruby

"Show and Tell" MLH, Anna Tselikova

Why Ruby?

Popular?

Ruby, however, is quite well off: according to recently published rankings by RedMonk, Ruby takes the 8th place and belongs to Tier 1 programming languages.

Startups..?

Among a multitude of programming languages, Ruby has been a great choice for startups. With its simple and intuitive code, Ruby facilitates software development and saves time: on average, developers **who use Ruby on Rails build applications 30-40% faster than teams that use other technologies**.

Who uses it?

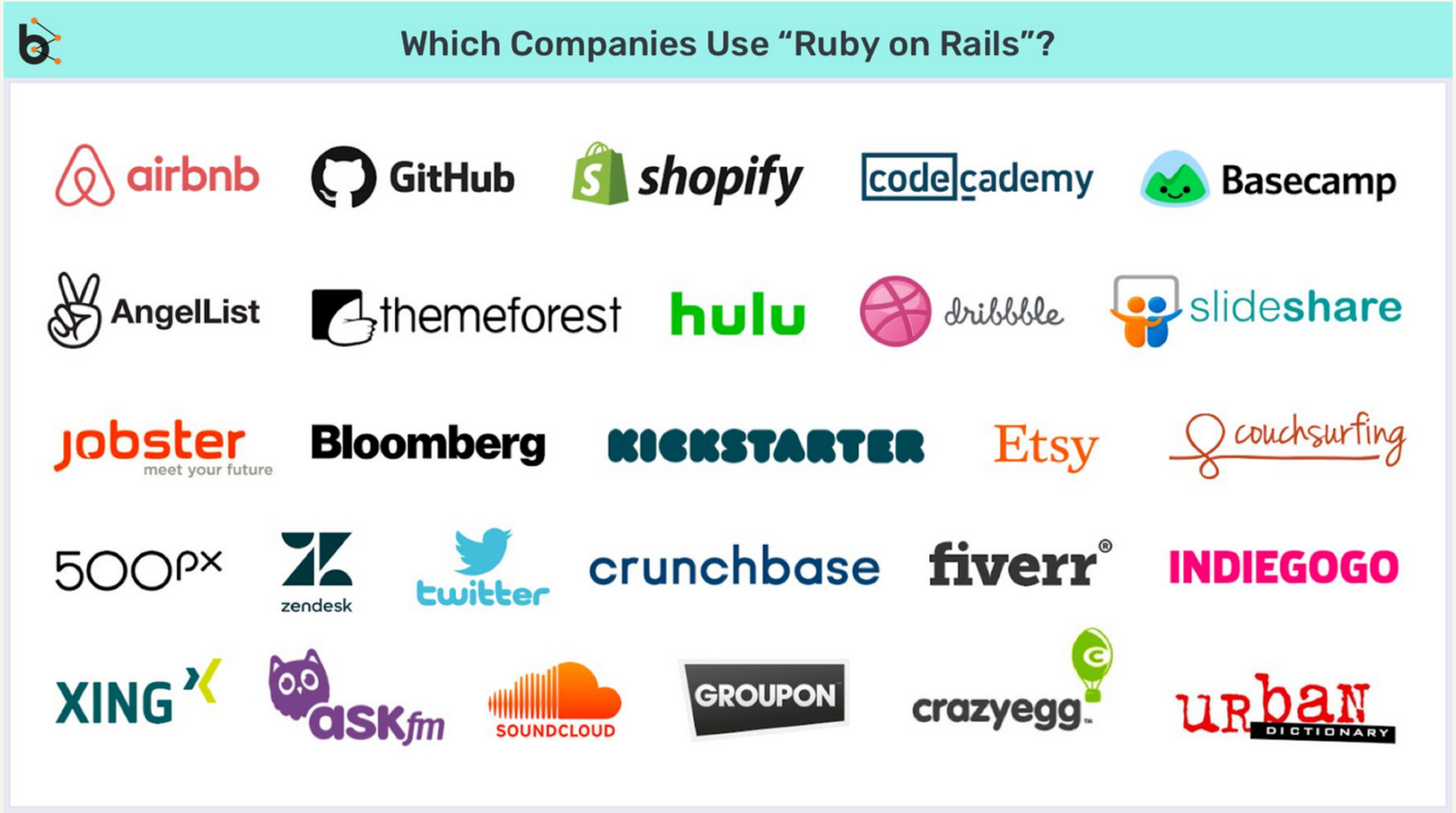
Social Networking Apps

It's also used for apartment sharing and booking apps - **Airbnb** and **Couchsurfing** because it swiftly manages a big amount of daily transactions and property bookings. Also, **Twitter** uses it.

Platforms that support complex databases.

Some of the best examples of such platforms are **GitHub** - the biggest Git repository hosting platform for developers and **Bloomberg** - a multi-platform financial news and analytics hub.

stackshare.io/ruby



Introduction

Part 01

A few facts about Ruby

Scripting

A Ruby application is always run at an interpreter, both during development and during end-user use.

Object oriented

- An object is a basic element of an application that has a state and methods that use this state
- The structure of objects is described by using classes
- To create objects (class instances) the class method ***new*** is used, the constructor

Dynamically typified

Variables may contain references to objects of any type

Objects in Ruby

What we should know

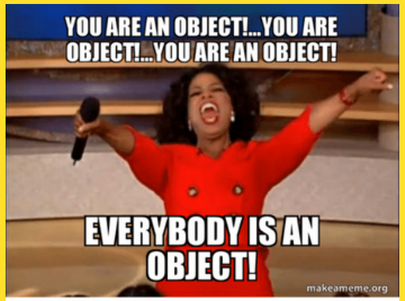
Each object has a unique identifier, even if there are no data

---

Objects contain instance variables describing its state
Objects contain methods that can modify instance variables

Methods may be public, private and protected

- Public methods can be caused by any objects
- Private methods can be called only by this object
- Protected methods can be triggered within the inheritance tree



Ruby is the same as Python, right?

Code samples

Part 02

Gem Bundler

manage dependencies

Bundler is a standard in the Ruby community for managing dependencies of a project under development.

- Allows you to create a list of gems used in the development, testing and delivery of the application
- Allows you to run the application in the environment of gems that are listed, isolating them from garbage
- Allows you to fix specific versions of necessary gems
- Allows you to update versions of fixed gems
- Allows saving gems locally to the project for transfer to other systems

To start using bundler, you must install it

$ gem install bundler


```
source 'https://rubygems.org'

gem 'tty-prompt', '~> 0.21.0'

gem 'rubocop', '~> 0.82.0'
```

```
Gemfile
Gemfile.lock
```

Dependencies

Shortly:

$ gem install bundler
$ bundle init
$ bundle install

01

Phase 01

Install bundler gem:
$ gem install bundler

02

Phase 02

Create a Gemfile file in the project directory, if it is not available you can use the command: $ bundle init

03

Phase 03

Specify in the file the source for downloading gems

04

Phase 04

Specify a list of gem dependencies in the file

05

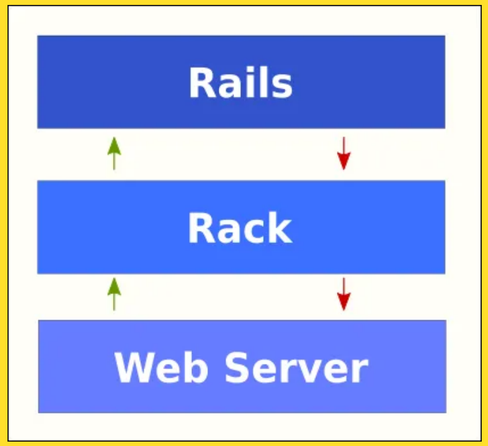
Phase 05

Run the command
$ bundle install

Show Bundler usage

Part 03

Implementation of server HTTP-stack in Ruby



Rack Library

The implementation of all modern server HTTP-stack for Ruby applications is based on the Rack library:

- Offers a unified approach for processing incoming requests and forming answers to them
- Describes the standard interfaces for both web servers and applications

There are many web servers: puma, unicorn, thin, etc., which have different properties

On the basis of Rack library created libraries and frameworks: Ruby on Rails, Sinatra, Hanami, RODA, Cuba

A new framework under development

Rack shortly

Few facts

Rack is most interesting for developers

- low-level tools (servers, libraries)
- universal handlers: query logging, filtering, etc.

Rack::Builder
The Rack::Builder class provides a subject programming language that allows you to connect intermediate layers to each other.

It provides the following methods:
- **use** allows you to connect the intermediate layer
- **map** allows you to bind handlers to specific paths
- **run** starts the application to process requests

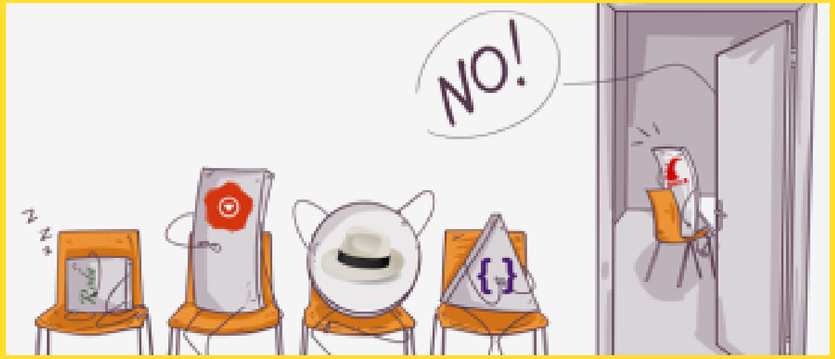
Usage

File config.ru

Gem rack delivers a rackup application that is designed to run Rack applications.
Its configuration file is config.ru.

```
require_relative 'app'
require_relative
'middleware_logger'

use MiddlewareLogger
run App.new
```



Roda

Roda Library

RODA is a library that makes easier coding web applications on Ruby.

Key features

- Easy to understand, focused on the task of processing requests
- Key element - routing tree
- Reliability due to the ability to apply a constant state, which allows you to easily start processing requests in multiple threads
- Built on a modular system, which allows you to choose the right add-ons for your specific task.

One of the fastest Rake-libraries available on the Ruby Stack

Roda. Not shortly. Modules.

Roda is module system

Roda is built on the principle of modular system, with the help of modules you can solve specific tasks of application development

Modular system allows you to make radical changes to the API by adding a new module and marking the old module as obsolete

To connect modules inside a RODA application, the plugin method is used

```
class App < Roda
  plugin :csrf
end
```

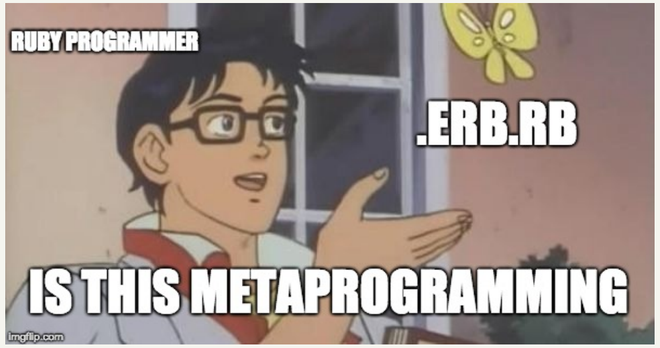
Available modules

The list of modules supplied with the roda jam is described in the official documentation.

The modules are divided into the following groups:

- Routing - routing, processing requests from clients
- Rendering/View - server response generation, usually for HTML content
- Request/Responce - processing requests and responses at the HTTP protocol level
- Matchers - expand the routing tree (route do |r|).
- Mail - forming e-mail from the application
- Middleware - application work with intermediate layers Rack-stack

Other - various modules that were not included in the previous categories



Roda. Pages.

Dynamic pages

To form HTML documents, the **tilt library** (tilt gem) is used, which supports multiple template engines. This library should be added, depending on the application

On the Roda side, we will use the following modules:

- Render - basic module for forming HTML pages
- Partials - a module for displaying templates that begin with stressing

Render offers 2 methods for displaying information:

- view - display the page template inside the layout
- render - display the template without layout

Partials adds a partial method that works as render

Layout system

When forming pages, we always have common elements: title, navigation bar, foot and technical elements. It is convenient to place them in the layout and replace only the central part.

The layout file is called layout.erb and is located next to the page templates in the views directory.

```
r.on 'tests' do
  r.get do
    view('tests')
  end
end
```

Roda.
Variables.

Send data to ERB template

There are two options for transferring information to ERB:

Local variables:

In Ruby:

```
view('tests', locals: { info: 'Some' }).
```

In ERB:

```
<%= info %>
```

Instance variables:

In Ruby:

```
@info = 'Some info'
view('tests')
```

In ERB:

```
<%= @info %>
```

Which one is better?

- Syntax of local variables is more complex than instance variables
- If no local variable is found, the template engine will end up with the exception
- If no instance variable is found, the data on the page will not be displayed.
- Template "speed" is higher when using instance variables

Show Roda usage

Part 04

Thanks for your attention!

GitHub Repo Link

HTTPS://GITHUB.COM/ANIATS/MLH-SHOW-AND-TELL

Questions?

ASK ME ON DISCORD OR EMAIL ME

ania.tselikova@gmail.com