



Rails 入门

1. 初体验

申思维
siwei.me

学习 Rails 的理由

- web 开发的王者 . 完虐 PHP 及其他框架 .
- 性能不差 . 用于优酷无线后端 (7 亿用户)
- 极好找工作 . 市面上 10 个职位空缺对应一个候选人 .
- Ruby 的平均工资在国外最高 . 国内也不低 .
- java/oc/php 让人郁闷 , Ruby 让人快乐 .
-

架构小知识：MVC

- 几乎是最经典的架构。10 个框架,9 个 MVC
- Model: 操作数据库
- View: 在浏览器端显示的视图
- Controller: 把 request 分发给 action 处理。
- (一个 controller 包含多个 action.)

Rails 架构：MVC

- 几乎是最经典的架构。10 个框架, 9 个 MVC
- Model: 操作数据库
- View: 在浏览器端显示的视图
- Controller: 把 request 分发给 action 处理。
- (一个 controller 包含多个 action.)

安装 Rails

- (命令都以在 Linux 端为例)
- 1. 安装了 ruby (见之前的视频)
- 2. `$ gem install bundler`
- 3. `$ gem install rails -v 4.1.6`
- 安装时会出现一系列的命令，只要最后你输入：
- `$ rails -v`，看到显示版本号，就说明安装好了。
- ```
siwei@siwei-linux:/workspace$ rails -v
Rails 4.1.6
```

# 创建一个 Rails 项目 .

- 使用命令：\$ rails new < 项目名 >, 例如：
- 
- \$ cd /workspace
- \$ rails new library
  
- 就创建了一个项目，名为：library.

# 命令截图：

```
siwei@siwei-linux:~/workspace$ rails new library
create
create README.rdoc
create Rakefile
create config.ru
create .gitignore
create Gemfile
create app
create app/assets/javascripts/application.js
create app/assets/stylesheets/application.css
create app/controllers/application_controller.rb
create app/helpers/application_helper.rb
create app/views/layouts/application.html.erb
create app/assets/images/.keep
create app/mailers/.keep
create app/models/.keep
create app/controllers/concerns/.keep
create app/models/concerns/.keep
create bin
create bin/bundle
create bin/rails
create bin/rake
create config
```

## 命令截图：

- ```
create test/mailers/.keep
```
- ```
create test/models
```
- ```
create test/models/.keep
```
- ```
create test/helpers
```
- ```
create test/helpers/.keep
```
- ```
create test/integration
```
- ```
create test/integration/.keep
```
- ```
create test/test_helper.rb
```
- ```
create tmp/cache
```
- ```
create tmp/cache/assets
```
- ```
create vendor/assets/javascripts
```
- ```
create vendor/assets/javascripts/.keep
```
- ```
create vendor/assets/stylesheets
```
- ```
create vendor/assets/stylesheets/.keep
```
- ```
run bundle install
```

- 可以看到，创建了一系列的文件。然后它会自动执行‘`bundle install`’命令。

Rails 使用 bundler 来管理各种依赖

- 跟 java 的 jar 一样 . ruby 中也有很多第三方包 , 我们管它叫 gem (小宝石)
- 每个 Rails 项目 , 都要依赖很多 gems. 一个一个的管理会特别麻烦 .
- 在 java 中 , 使用 maven, ivy 来管理 . 那么在 Rails 中 , 就使用 bundler 来管理 .
- bundler 通过一个文件和一个命令来管理 .
 - 文件 : Gemfile. 定义了所有的 gem 版本 .
 - 命令 : `$ bundle install` , 会自动安装所有的 gems.

使用 Gemfile 安装各种依赖包。

- 编辑你的 Gemfile(根目录下) 文件, 让它的内容看起来如下:
- `source 'https://ruby.taobao.org/'`
- `gem 'rails', '4.1.6'`
- `gem 'sqlite3'`
- `gem 'sass-rails', '~> 4.0.3'`
- `gem 'uglifier', '>= 1.3.0'`
- `gem 'therubyracer', platforms: :ruby`
- `gem 'jquery-rails'`
- `gem 'turbolinks'`

安装各种依赖, gems.

- 先安装好 `sqlite3` 的依赖：
 - `$ sudo apt-get install libsqlite3-dev`
- 然后通过 命令：`$ bundle install` 即可。

```
Using activemodel 4.1.6
Using actionpack 4.1.6
Using activerecord 4.1.6
Using actionmailer 4.1.6
Using railties 4.1.6
Using sprockets-rails 2.3.3
Using jquery-rails 3.1.4
Using rails 4.1.6
Using sass-rails 4.0.5
Bundle complete! 7 Gemfile dependencies, 40 gems now installed.
Use `bundle show [gemname]` to see where a bundled gem is installed.
```

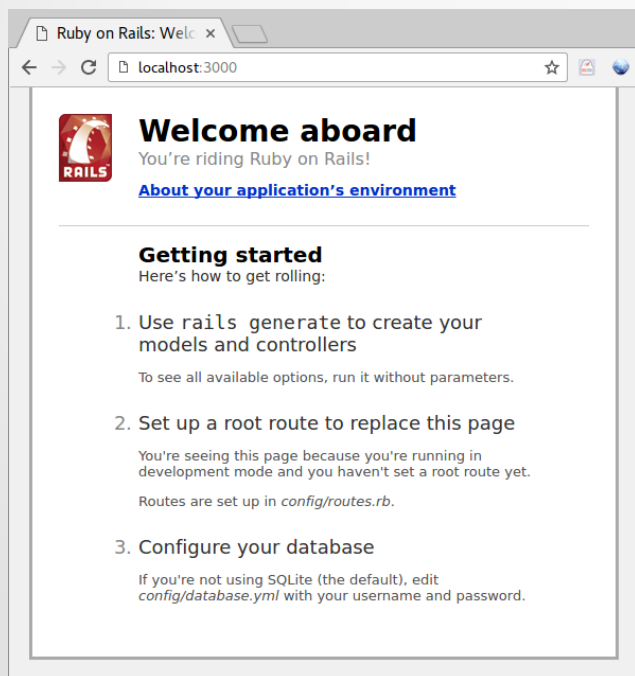
运行 Rails !

- 使用命令： `$ bundle exec rails server`
- `bundle exec`: 是 rails 命令的前缀。它会告诉 rails, 以后用到的 gem, 都是 Gemfile 中规定的版本。
- 下图就是启动后的命令。

```
siwei@siwei-linux:/workspace/library$ bundle exec rails s
=> Booting WEBrick
=> Rails 4.1.6 application starting in development on http://0.0.0.0:3000
=> Run `rails server -h` for more startup options
=> Notice: server is listening on all interfaces (0.0.0.0). Consider using 127.0.0.1 (--binding option)
=> Ctrl-C to shutdown server
[2016-09-21 10:24:52] INFO WEBrick 1.3.1
[2016-09-21 10:24:52] INFO ruby 2.0.0 (2015-12-16) [x86_64-linux]
[2016-09-21 10:24:52] INFO WEBrick::HTTPServer#start: pid=1206 port=3000
```

访问项目首页

- 打开浏览器，访问 `localhost:3000`，能看到首页。（对于好奇宝宝来说，这个页面是存在于 `gem` 文件中的。在刚才生成的 `library` 目录下找不到它）



开始动手第一步：

- 现在项目没有任何内容。只是有了一个骨架。
- 所以我们需要给他增加内容，丰满它。
- 例如：输入一个网址，显示一个页面。

开始动手第一步：

- 假设，用户需要访问的 URL:
- localhost:3000/books/list
- 那么，我们只需要：
 1. 修改路由 (router)
 2. 创建对应的 controller, action
 3. 创建对应的页面 (erb)

修改路由 config/routes.rb

- 向该文件中，增加：
- 下面的代码：(resources ... do ... end)
- 它会生成一段”路由”，也就是说，会识别 /books/list 这个 URL，并且把它交给 books controller 的 list action 来处理。

```
Rails.application.routes.draw do
  resources :books do
    collection do
      get :list
    end
  end
end
```


创建 controller/action

- 新建一个文件：app/controllers/books_controller.rb ,(见下面截图)
- 一个 controller 是由多个 action(也就是普通方法组成的)
- 不用的 action, 处理不同的 url .
- 这个 list action 里没有任何代码 . 它会直接跳转到对应的 erb 页面 (app/views/books/list.html.erb)

```
class BooksController < ApplicationController  
  
  def list  
  end  
end
```

创建对应的视图文件

- 新建 `app/views/books/list.html.erb`
- (所有 `books controller` 中的视图, 都要放到 `app/views/books` 目录下)
- (action 叫什么名字, 视图就叫什么名字. `list action`, 对应的视图, 就是: `list.html.erb`)
- 为了简单起见, 它只显示 HTML 代码.

```
<p> 三体 1 - 地球往事 </p>  
<p> 三体 2 - 黑暗森林 </p>  
<p> 三体 3 - 死神永生 </p>
```

看到结果

- 访问浏览器：<http://localhost:3000/books/list>
- 可以看到结果：



在视图中，使用 ruby 代码。

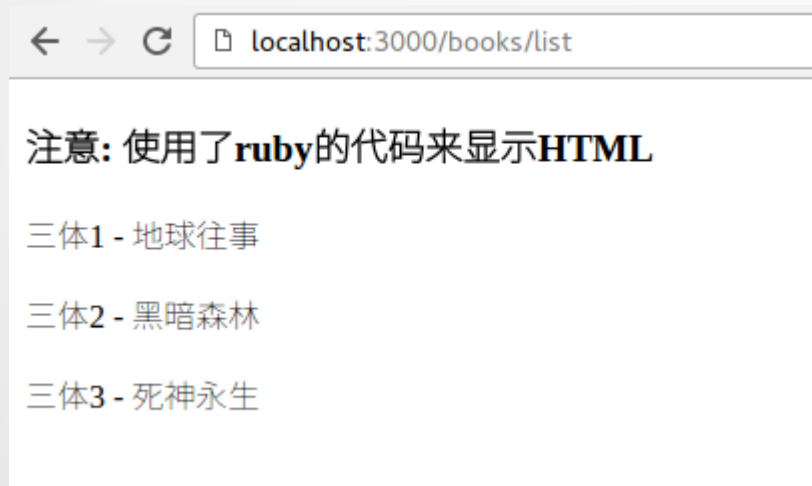
- 修改 `app/views/books/list.html.erb`
- 可以看到，语法与 PHP, JSP 一样。
- `<% %>` 之间来执行 ruby 代码。
- `<%= %>` 来显示返回值。

```
<h3>注意：使用了 ruby 的代码来显示 HTML</h3>
<% books = ['三体1 - 地球往事', '三体2 - 黑暗森林', '三体3 - 死神永生'] %>

<% books.each do |book| %>
  <p><%= book %></p>
<% end %>
```

看到结果

- 访问浏览器 : `http://localhost:3000/books/list`
- 可以看到结果 :



总结 1.

- `http://host/books/list`
- （上面的 url, 如果是按照 rails 的约定惯例来看的话, 就是:
- 显示 books 的 list 页面 .
- 所以, rails 会把这个请求, 根据路由 (router), 发送给对应的方法 (也就是 controller 中的 action) 。

总结 2:

- router, 根据配置文件: `config/routes.rb` 中的配置:

```
Rails.application.routes.draw do
  resources :books do
    collection do
      # 下面这个路由, 让 rails 可以处理 URL: /books/list
      get :list
    end
  end
end
```

总结 3:

- list action 做一些处理, 显示对应的 erb (JSP, PHP 也是一样的)
- class BooksController < ApplicationController
- def list
- # 1. 渲染一段字符串
- #render :text => 'hihihi'
- # 2. 渲染一个json
- #render :json => {
- # key: 'value',
- # name: 'dashi',
- # sex: 'male'
- #}
- # 3. 啥也不写, 就渲染对应的
- # app/views/books/list.html.erb
- end
- end

总结 4:

- 如果要渲染的是一个 erb 页面，我们可以在 erb 中直接写 ruby 代码。
- `<% %>`: 仅执行
- `<%= %>`: 执行并显示结果到页面中。
- 例如:

`<h3>` 注意：使用了 ruby 的代码来显示 HTML `</h3>`

```
<% books = ['三体 1 - 地球往事', '三体 2 - 黑暗森林', '三体 3 - 死神永生'] %>
```

```
<% books.each do |book| %>
```

```
  <p><%= book %></p>
```

```
<% end %>
```

源代码

- 本节源代码可以在下列地址下载：
- https://github.com/sg552/rails_lesson_1_setup_and_run
-

谢谢！

微信公众账号
软件实用主义



个人网站：<http://siwei.me>

明创软件：<http://siwei.tech>