Rails 入门
2. 路由 (route)

申思维
siwei.me

学习收获

- 知道什么是路由
- 知道 RESTful 思想。
- 知道哪些路由是过时的，不需要学

事实上的 http 请求 : GET/POST

- POST 请求 :
  - 提交表单 (<form>) 时
- GET 请求 :
  - 点击 url (<a> 标签) 时
  - 在浏览器地址栏输入 url, 回车时 .

RESTful 请求

- 认为浏览器请求的任何数据 , 都是资源 (resources).
- 总共有 4 种请求 :
  - GET : 仅用于查询
  - POST: 仅用于新建
  - PUT: 仅用于修改
  - DELETE 仅用于 删除 .

RESTful 请求的几个例子

- 查看数据：
  - GET /books ： 查看 books 的列表
  - GET /books/2 ： 查看 id = 2 的 book
  - GET /books?id=2 同上 .(查看 id=2 的 book)
- 创建数据：
  - POST /books (name = ‘三体’) 创建一本书
- 修改数据：
  - PUT /books/2 (name = ‘黑暗森林’)
- 删除数据：
  - DELETE /books/2 删除 id = 2 的 书 .

RESTful：相同的 URL，不同的操作

- 以下操作的 URL 都是相同的。request 的类型是不同的
- 查看数据：
  - GET /books/2 ： 查看 id = 2 的 book
- 修改数据：
  - PUT /books/2 (name = ‘黑暗森林’)
- 删除数据：
  - DELETE /books/2 删除 id = 2 的 书 .

web 编程的精髓：CRUD

- 就是数据的增删改查（简称 CRUD, create, read, update, delete).
- 在 Rails 中，系统默认每个 controller 有这样的操作：
  - index，列表页（有对应页面）
  - show 详情页（有对应页面）
  - edit 编辑页（有对应页面）
  - new 新建页（有对应页面）
  - update 更新操作
  - create 新建操作
  - destroy 删除操作.

回到 Rails 项目

- 下载第一课的 rails 项目 .
  - $ git clone https://github.com/sg552/rails_lesson_1_setup_and_run
- 为 books controller 增加 new action, 它的作用是处理 /books/new 这个 url.

```
class BooksController < ApplicationController

  # 增加这个新的action
  def new
  end

  def list
  end
end
```

Rails 中的路由

- 回顾下 config/routes.rb 文件：

```
Rails.application.routes.draw do

  # 这里会生成七种路由.
  resources :books do
    collection do
      # 在这里生成第八种路由: GET:  /books/list
      get :list
    end
  end
end
```

回顾路由文件 config/routes.rb

resources :books

上面一句，就直接定义了 7 种路由：

GET /books index 显示 book 的列表页

- GET /books/new new 显示 book 的新建页面。
- GET /books/3 show 显示 id = 3 的 book
- GET /books/3/edit edit 显示 id = 3 的 book 的编辑页面。
- PUT /books/3 update 对 id = 3 的 book 进行修改（后面还会紧跟一大串的参数）
- POST /books create 对 books 进行创建（后面也有一大堆参数）
- DELETE /books/3 destroy 对 id=3 的 book 进行删除操作。

回顾路由文件 config/routes.rb

也可以通过 命令： $bundle exec rake routes 来查看详细的路由：

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

```
    Prefix Verb   URI Pattern                Controller#Action
list_books GET    /books/list(.:format)      books#list
     books GET    /books(.:format)           books#index
           POST   /books(.:format)           books#create
  new_book GET    /books/new(.:format)       books#new
 edit_book GET    /books/:id/edit(.:format)  books#edit
      book GET    /books/:id(.:format)       books#show
           PATCH  /books/:id(.:format)       books#update
           PUT    /books/:id(.:format)       books#update
           DELETE /books/:id(.:format)       books#destroy
```

增加了可读性的路由

为了大家 看的方便:

1. 增加了前缀（prefix）列中原来省略的内容
2. 去掉了 URI pattern 中无用的 .format
3. 删掉了 PATCH(它跟 PUT 一模一样，所以精简掉它)

```
    Prefix Verb   URI Pattern                Controller#Action
list_books GET    /books/list                books#list
     books GET    /books                     books#index
     books POST   /books                     books#create
  new_book GET    /books/new                 books#new
 edit_book GET    /books/:id/edit            books#edit
      book GET    /books/:id                 books#show
           PATCH  /books/:id                 books#update
      book PUT    /books/:id                 books#update
      book DELETE /books/:id                 books#destroy
```

- 所以，一个 resources，就定义了 new,show, index, create, edit, update, destroy 总共 7 个从 url 到 action 的对应关系.
- 因为还有 collection ... get :list ... 语句，所以，就多定义出来一个： list_book 这个 url.

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

这里 会定义一个新的URL：

GET /books/list
对应： list_books_path

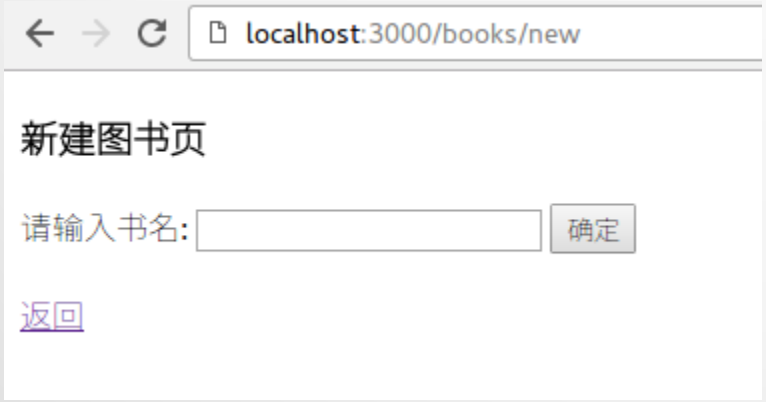
为 new 这个 action 增加一个新页面 .

- 前面的路由知识铺垫完了，我们继续创建第二个页面。
- 我们新建文件： app/views/books/new.html.erb
- 这是个最普通的 html 文件 .
- 这个页面，可以跳转到： 第一个页面。 （ /books/list)

```
<h3>新建图书页</h3>

<form action="" method="POST">
  请输入书名：
  <input type='text' name='book_name'/>
  <input type='submit' value='确定'/>
</form>

<br/>

<a href='/books/list'>返回</a>
```

新建图书页

- 启动 rails（ 还记得命令吗？ $ bundle exec rails server ）
- 访问 url: /books/new
- 可以看到，出现了页面：



实现页面的互相跳转 .

- 同时，我们修改 /books/list 页面，

```
<h3> 下面的代码,都是由 RUBY代码生成的</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 %>

<a href='/books/new'>新建图书</a>
```

- 可以看到，多出了一个 <a> 链接，

重点来了： Rails 中的 url 写法。

- 跟 茴香豆的四种写法 <a> 标签的下面四种写法，都是一样的.

```
<!-- 下面四种写法，都会产生 同样的<a>内容 -->
<a href='/books/new'>跳转到新建页 </a>
<a href='<%= new_book_path %>'>跳转到新建页 </a>
<%= link_to "跳转到新建页", '/books/new' %>
<%= link_to "跳转到新建页", new_book_path %>
```

- 它们生成的 HTML 内容都是一样的：

```
16 <h3>注意： 使用了ruby的代码来显示HTML</h3>
17
18   <p>三体1 - 地球往事</p>
19   <p>三体2 - 黑暗森林</p>
20   <p>三体3 - 死神永生</p>
21
22 <!-- 下面四种写法，都会产生 同样的<a>内容 -->
23 <a href='/books/new'>跳转到新建页 </a>
24 <a href='/books/new'>跳转到新建页 </a>
25 <a href="/books/new">跳转到新建页</a>
26 <a href="/books/new">跳转到新建页</a>
27
28
```

回答问题

- new_book_path 是从哪儿来的？来自于 route 的定义。
- new_book_path 是动态生成的方法，返回一个 URL （可以认为是字符串）
- 同理， 还可以从下面的配置文件看出，还有其他的 url: edit_book_path， books_path，以及 book_path

```
    Prefix Verb   URI Pattern                Controller#Action
list_books GET    /books/list                books#list
     books GET    /books
     books POST   /books
  new_book GET    /books/new                 books#new
 edit_book GET    /books/:id/edit            books#edit
      book GET    /books/:id                 books#show
           PATCH  /books/:id                 books#update
      book PUT    /books/:id                 books#update
      book DELETE /books/:id                 books#destroy
```

这里的 new_book, 就是 前缀。 后缀： _path, _url，所以我们就可以在 rails 中，使用： new_book_path/new_book_url

为什么弄这个新东西?

- 太长不看版答案： 为了我们的方便。
- 详细答案:
- 如何编辑某个 book?

  /books/1/edit （编辑 id = 1 的 book)
- 写成 ruby 代码 : (假设 book_id = 1)
  - 1. "/books/" + book_id + "/edit"
  - 2. "/books/#{book_id}/edit" # string interpolation 插入插值

  实际上上面这两种形式，都是外行的风格（比如，之前做 java 的同学，来写 ruby，就是这个风格）

为什么弄这个新东西?

- 下面是 Rails 风格 (把可读性发挥到极致）的写法:
- edit_book_url({:id => book_id})
- 这个写法，可以引申成: （ruby 方法最外层圆括号可以省略，方法的最后一个参数如果是 hash 的话，{} 也可以省略)
- edit_book_url :id => book_id
- 继续省略:
- edit_book_url book_id

可读性：

- book_id = 1
  - 1. "/books/" + book_id + "/edit"
  - 2. "/books/#{book_id}/edit"
  - 3. edit_book_path book_id # rails 的推荐写法。
- 第三种形式是可读性最好的，也是 Rails 程序员最喜欢读写的。
- 所以，记住： 希望生成某个 url:

  /books/list?a=1&b=2

  就写成：
- list_books_path(:a =>1, :b => 2)

带有 namespace 的路由：

- namespace 也就是一个前缀 . 有 2 种常见用途 :
- 1. 不同角色使用的 url:
- /books 普通用户看到的 url
- /admin/books 专门给管理员访问的 url
- 2. 对于接口的 url:
- /interface/books

写接口的例子．步骤 1-router

- 在 router.rb 中，增加下面红色边框的语句：

```
1  Rails.application.routes.draw do
2
3    # 这里会生成七种路由.
4    resources :books do
5      collection do
6        # 在这里生成第八种路由: GET:  /books/list
7        get :list
8      end
9    end
10
11   resources :interface do
12     resources :books do
13       collection do
14         get :all
15       end
16     end
17   end
18 end
```

会生成路由:
/interface/books/all

交给 app/controllers/interface/books_controller.rb 中的 all action 来处理.

写接口的例子．步骤 2-controller

- 注意它跟 app/controllers/books_controller.rb 不是一个文件 .
- 新建一个文件 : app/controllers/interface/books_controller.rb
  - render :json => {} 表示 返回一个 json
  - 继承的类是 ActionController::Base
  - 类名前面有个 Interface:: 前缀 .

```
class Interface::BooksController < ActionController::Base
  def all
    render :json => {
      :name => '小王',
      :sex => 'male',
      :age => 18,
      :address => '北京市朝阳区...'
    }
  end
end
```

打开浏览器，可以看到结果

- 打开浏览器 http://localhost:3000/interface/books/all
- 可以看到结果： 一个 json.

特殊的路由： root_path

- root_path 决定了你的项目根目录的页面是哪个 .
- 例如，我增加下面的路由 (表示 , 对于 根目录的请求 ， 会由 books controller 的 welcome action 来处理。 :

```
# 根路径
root :to=>  'books#welcome'
```

特殊的路由： root_path

- 修改对应的 controller: (app/controllers/books_controller.rb)

```
# 专门为根目录准备
def welcome
end
```

- 修改对应的 view: (app/views/books/welcome.html.erb)

```
<h1>欢迎来到Rails的世界</h1>

<p>Ruby 让编程变得快乐。</p>
```

特殊的路由： root_path

- 现在，访问咱们的根路径 （ / ）， 就可以看到结果了： （注意： 由于我们没有在 resources :books 中配置 get :welcome, 所以，我们访问 /books/welcome 是会报错的）

古老的路由： match

- 有一种最古老的路由： match . 随着 RESTful 的崛起，这个路由越发显得过时。但是可能在一些遗留项目中会出现：
- 在目前的 Rails4 中， 它的写法如下：
  - match: 指定了所匹配的 url
  - to : 指定了由哪个 controller#action 处理
  - via: 这个请求，是 get 还是 post.

```
# 直接指定一个url ， 由哪个controller#action 来处理。
match '/welcome_you', to: 'books#welcome', via: [:get, :post]
```

古老的路由： match

- 现在，访问 /welcome_you 就可以看到结果了。

总结 1： 需要学习的 route 形式

- 今天通过两个页面的跳转，学习了路由。
- 1. resources， 重中之重。
- 2. 可以加上 namespace
- 3. 可以看懂： match ... （回头补上）
- 4. 其他的， 嵌套的 routes, resource（ 单数）， 都不用看。

总结 2： 不需要学习的 route 形式

- 不要使用嵌套路由 , 太傻了 . 嵌套路由的例子:
- 中国 (86)- 辽宁省 (041x)- 沈阳市 (010)- 和平区 (1)
- 路由可以是:
- /countries/86/provinces/041x/cities/010/districts/1
-

```
resources :countries do
  resources :provinces do
    resources :cities do
      resources :districts do
          ...
      end
    end
  end
end
```

总结 2： 不需要学习的 route 形式

- 但是，不要写的这么傻。 区的 id 是固定的，有它一个就足够了。 所以，我们只要写：
- /districts/1
- 就可以查询到： id = 1 的 district

总结 3： 不需要学习的 route 形式

- 其他形式的，都不用专门学。
- 遇到的时候，能看懂代码就可以。

本节源代码

- 本节中的例子，对应的源代码，可以来这里下载：
-
- https://github.com/sg552/rails_lesson_2_route

谢谢！

微信公众账号
软件实用主义

个人网站：　http://siwei.me