[2.2.2 MVC in action](http://ruby.railstutorial.org/chapters/a-demo-app" \l "sec-mvc_in_action)

Now that we’ve completed a quick overview of the Users resource, let’s examine one particular part of it in the context of the Model-View-Controller (MVC) pattern introduced in [Section 1.2.6](http://ruby.railstutorial.org/chapters/beginning#sec-mvc). Our strategy will be to describe the results of a typical browser hit—a visit to the user index page at[/users](http://localhost:3000/users)—in terms of MVC ([Figure 2.11](http://ruby.railstutorial.org/chapters/a-demo-app#fig-mvc_detailed)).

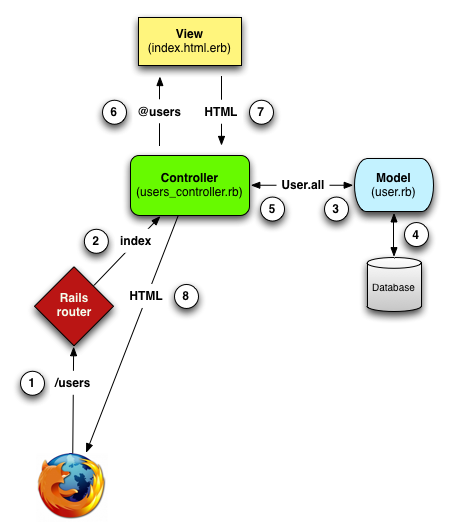


Figure 2.11: A detailed diagram of MVC in Rails. [(full size)](http://railstutorial.org/images/figures/mvc_detailed-full.png)

1. The browser issues a request for the /users URL.
2. Rails routes /users to the **index** action in the Users controller.
3. The **index** action asks the User model to retrieve all users (**User.all**).
4. The User model pulls all the users from the database.
5. The User model returns the list of users to the controller.
6. The controller captures the users in the **@users** variable, which is passed to the **index**view.
7. The view uses embedded Ruby to render the page as HTML.
8. The controller passes the HTML back to the browser.[3](http://ruby.railstutorial.org/chapters/a-demo-app#fn-2_3)

We start with a request issued from the browser—i.e., the result of typing a URL in the address bar or clicking on a link (Step 1 in [Figure 2.11](http://ruby.railstutorial.org/chapters/a-demo-app#fig-mvc_detailed)). This request hits the *Rails router* (Step 2), which dispatches to the proper *controller action* based on the URL (and, as we’ll see in [Box 3.3](http://ruby.railstutorial.org/chapters/static-pages#sidebar-get_etc), the type of request). The code to create the mapping of user URLs to controller actions for the Users resource appears in [Listing 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#code-rails_routes); this code effectively sets up the table of URL/action pairs seen in[Table 2.1](http://ruby.railstutorial.org/chapters/a-demo-app#table-user_urls). (The strange notation **:users** is a *symbol*, which we’ll learn about in [Section 4.3.3](http://ruby.railstutorial.org/chapters/rails-flavored-ruby#sec-hashes_and_symbols).)

**Listing 2.2.** The Rails routes, with a rule for the Users resource.   
**config/routes.rb**

DemoApp::Application.routes.draw **do**

resources :users

.

.

.

**end**

The pages from the tour in [Section 2.2.1](http://ruby.railstutorial.org/chapters/a-demo-app#sec-a_user_tour) correspond to *actions* in the Users *controller*, which is a collection of related actions; the controller generated by the scaffolding is shown schematically in[Listing 2.3](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_users_controller). Note the notation **class UsersController < ApplicationController**; this is an example of a Ruby *class* with *inheritance*. (We’ll discuss inheritance briefly in [Section 2.3.4](http://ruby.railstutorial.org/chapters/a-demo-app#sec-inheritance_hierarchies)and cover both subjects in more detail in [Section 4.4](http://ruby.railstutorial.org/chapters/rails-flavored-ruby#sec-ruby_classes).)

**Listing 2.3.** The Users controller in schematic form.   
**app/controllers/users\_controller.rb**

**class** **UsersController** < ApplicationController

.

.

.

**def** index

.

.

.

**end**

**def** show

.

.

.

**end**

**def** new

.

.

.

**end**

**def** create

.

.

.

**end**

**def** edit

.

.

.

**end**

**def** update

.

.

.

**end**

**def** destroy

.

.

.

**end**

**end**

You may notice that there are more actions than there are pages; the **index**, **show**, **new**, and**edit** actions all correspond to pages from [Section 2.2.1](http://ruby.railstutorial.org/chapters/a-demo-app#sec-a_user_tour), but there are additional **create**, **update**, and **destroy** actions as well. These actions don’t typically render pages (although they sometimes do); instead, their main purpose is to modify information about users in the database. This full suite of controller actions, summarized in [Table 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#table-demo_RESTful_users), represents the implementation of the REST architecture in Rails ([Box 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#sidebar-REST)), which is based on the ideas of *representational state transfer*identified and named by computer scientist [Roy Fielding](http://en.wikipedia.org/wiki/Roy_Fielding).[4](http://ruby.railstutorial.org/chapters/a-demo-app#fn-2_4) Note from [Table 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#table-demo_RESTful_users) that there is some overlap in the URLs; for example, both the user **show** action and the **update** action correspond to the URL /users/1. The difference between them is the [HTTP request method](http://en.wikipedia.org/wiki/HTTP_request#Request_methods) they respond to. We’ll learn more about HTTP request methods starting in [Section 3.2.1](http://ruby.railstutorial.org/chapters/static-pages#sec-TDD).

|  |  |  |  |
| --- | --- | --- | --- |
| **HTTP request** | **URL** | **Action** | **Purpose** |
| GET | /users | **index** | page to list all users |
| GET | /users/1 | **show** | page to show user with id **1** |
| GET | /users/new | **new** | page to make a new user |
| POST | /users | **create** | create a new user |
| GET | /users/1/edit | **edit** | page to edit user with id **1** |
| PATCH | /users/1 | **update** | update user with id **1** |
| DELETE | /users/1 | **destroy** | delete user with id **1** |

Table 2.2: RESTful routes provided by the Users resource in [Listing 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#code-rails_routes).

**Box 2.2.REpresentational State Transfer (REST)**

If you read much about Ruby on Rails web development, you’ll see a lot of references to “REST”, which is an acronym for REpresentational State Transfer. REST is an architectural style for developing distributed, networked systems and software applications such as the World Wide Web and web applications. Although REST theory is rather abstract, in the context of Rails applications REST means that most application components (such as users and microposts) are modeled as *resources* that can be created, read, updated, and deleted—operations that correspond both to the [CRUD operations of relational databases](http://en.wikipedia.org/wiki/Create,_read,_update_and_delete) and four fundamental [HTTP request methods](http://en.wikipedia.org/wiki/HTTP_request#Request_methods): POST, GET, PATCH, and DELETE. (We’ll learn more about HTTP requests in [Section 3.2.1](http://ruby.railstutorial.org/chapters/static-pages#sec-TDD) and especially [Box 3.3](http://ruby.railstutorial.org/chapters/static-pages#sidebar-get_etc).)

As a Rails application developer, the RESTful style of development helps you make choices about which controllers and actions to write: you simply structure the application using resources that get created, read, updated, and deleted. In the case of users and microposts, this process is straightforward, since they are naturally resources in their own right. In [Chapter 11](http://ruby.railstutorial.org/chapters/following-users#top), we’ll see an example where REST principles allow us to model a subtler problem, “following users”, in a natural and convenient way.

To examine the relationship between the Users controller and the User model, let’s focus on a simplified version of the **index** action, shown in [Listing 2.4](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_index_action). (The scaffold code is ugly and confusing, so I’ve suppressed it.)

**Listing 2.4.** The simplified user **index** action for the demo application.   
**app/controllers/users\_controller.rb**

**class** **UsersController** < ApplicationController

.

.

.

**def** index

@users = User.all

**end**

.

.

.

**end**

This **index** action has the line **@users = User.all** (Step 3), which asks the User model to retrieve a list of all the users from the database (Step 4), and then places them in the variable**@users** (pronounced “at-users”) (Step 5). The User model itself appears in [Listing 2.5](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_user_model); although it is rather plain, it comes equipped with a large amount of functionality because of inheritance ([Section 2.3.4](http://ruby.railstutorial.org/chapters/a-demo-app#sec-inheritance_hierarchies) and [Section 4.4](http://ruby.railstutorial.org/chapters/rails-flavored-ruby#sec-ruby_classes)). In particular, by using the Rails library called *Active Record*, the code in [Listing 2.5](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_user_model) arranges for **User.all** to return all the users.

**Listing 2.5.** The User model for the demo application.   
**app/models/user.rb**

**class** **User** < ActiveRecord::Base

**end**

Once the **@users** variable is defined, the controller calls the *view* (Step 6), shown in [Listing 2.6](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_index_view). Variables that start with the **@** sign, called *instance variables*, are automatically available in the view; in this case, the **index.html.erb** view in [Listing 2.6](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_index_view) iterates through the **@users** list and outputs a line of HTML for each one. (Remember, you aren’t supposed to understand this code right now. It is shown only for purposes of illustration.)

**Listing 2.6.** The view for the user index.   
**app/views/users/index.html.erb**

**<h1>**Listing users**</h1>**

**<table>**

**<tr>**

**<th>**Name**</th>**

**<th>**Email**</th>**

**<th></th>**

**<th></th>**

**<th></th>**

**</tr>**

<% @users.each **do** |user| %>

**<tr>**

**<td>**<%= user.name %>**</td>**

**<td>**<%= user.email %>**</td>**

**<td>**<%= link\_to 'Show', user %>**</td>**

**<td>**<%= link\_to 'Edit', edit\_user\_path(user) %>**</td>**

**<td>**<%= link\_to 'Destroy', user, method: :delete,

data: { confirm: 'Are you sure?' } %>**</td>**

**</tr>**

<% **end** %>

**</table>**

**<br** **/>**

<%= link\_to 'New User', new\_user\_path %>

The view converts its contents to HTML (Step 7), which is then returned by the controller to the browser for display (Step 8).

[2.2.3 Weaknesses of this Users resource](http://ruby.railstutorial.org/chapters/a-demo-app#sec-weaknesses_of_this_users_resource)

Though good for getting a general overview of Rails, the scaffold Users resource suffers from a number of severe weaknesses.

* **No data validations.** Our User model accepts data such as blank names and invalid email addresses without complaint.
* **No authentication.** We have no notion of signing in or out, and no way to prevent any user from performing any operation.
* **No tests.** This isn’t technically true—the scaffolding includes rudimentary tests—but the generated tests are ugly and inflexible, and they don’t test for data validation, authentication, or any other custom requirements.
* **No layout.** There is no consistent site styling or navigation.
* **No real understanding.** If you understand the scaffold code, you probably shouldn’t be reading this book.

[2.3 The Microposts resource](http://ruby.railstutorial.org/chapters/a-demo-app#sec-microposts_resource)

Having generated and explored the Users resource, we turn now to the associated Microposts resource. Throughout this section, I recommend comparing the elements of the Microposts resource with the analogous user elements from [Section 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#sec-demo_users_resource); you should see that the two resources parallel each other in many ways. The RESTful structure of Rails applications is best absorbed by this sort of repetition of form; indeed, seeing the parallel structure of Users and Microposts even at this early stage is one of the prime motivations for this chapter. (As we’ll see, writing applications more robust than the toy example in this chapter takes considerable effort—we won’t see the Microposts resource again until [Chapter 10](http://ruby.railstutorial.org/chapters/user-microposts#top)—and I didn’t want to defer its first appearance quite that far.)

[2.3.1 A micropost microtour](http://ruby.railstutorial.org/chapters/a-demo-app#sec-a_micropost_microtour)

As with the Users resource, we’ll generate scaffold code for the Microposts resource using **rails generate scaffold**, in this case implementing the data model from [Figure 2.3](http://ruby.railstutorial.org/chapters/a-demo-app#fig-demo_micropost_model):[5](http://ruby.railstutorial.org/chapters/a-demo-app#fn-2_5)

$ rails generate scaffold Micropost content:string user\_id:integer

invoke active\_record

create db/migrate/20130307005528\_create\_microposts.rb

create app/models/micropost.rb

invoke test\_unit

create test/models/micropost\_test.rb

create test/fixtures/microposts.yml

invoke resource\_route

route resources :microposts

invoke jbuilder\_scaffold\_controller

create app/controllers/microposts\_controller.rb

invoke erb

create app/views/microposts

create app/views/microposts/index.html.erb

create app/views/microposts/edit.html.erb

create app/views/microposts/show.html.erb

create app/views/microposts/new.html.erb

create app/views/microposts/\_form.html.erb

invoke test\_unit

create test/controllers/microposts\_controller\_test.rb

invoke helper

create app/helpers/microposts\_helper.rb

invoke test\_unit

create test/helpers/microposts\_helper\_test.rb

invoke jbuilder

exist app/views/microposts

create app/views/microposts/index.json.jbuilder

create app/views/microposts/show.json.jbuilder

invoke assets

invoke coffee

create app/assets/javascripts/microposts.js.coffee

invoke scss

create app/assets/stylesheets/microposts.css.scss

invoke scss

identical app/assets/stylesheets/scaffolds.css.scss

To update our database with the new data model, we need to run a migration as in [Section 2.2](http://ruby.railstutorial.org/chapters/a-demo-app#sec-demo_users_resource):

**$** bundle exec rake db:migrate

== CreateMicroposts: migrating ===============================================

-- create\_table(:microposts)

-> 0.0023s

== CreateMicroposts: migrated (0.0026s) ======================================

Now we are in a position to create microposts in the same way we created users in [Section 2.2.1](http://ruby.railstutorial.org/chapters/a-demo-app#sec-a_user_tour). As you might guess, the scaffold generator has updated the Rails routes file with a rule for Microposts resource, as seen in [Listing 2.7](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_microposts_resource).[6](http://ruby.railstutorial.org/chapters/a-demo-app#fn-2_6) As with users, the **resources :microposts** routing rule maps micropost URLs to actions in the Microposts controller, as seen in [Table 2.3](http://ruby.railstutorial.org/chapters/a-demo-app#table-demo_RESTful_microposts).

**Listing 2.7.** The Rails routes, with a new rule for Microposts resources.   
**config/routes.rb**

DemoApp::Application.routes.draw **do**

resources :microposts

resources :users

.

.

.

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| **HTTP request** | **URL** | **Action** | **Purpose** |
| GET | /microposts | **index** | page to list all microposts |
| GET | /microposts/1 | **show** | page to show micropost with id **1** |
| GET | /microposts/new | **new** | page to make a new micropost |
| POST | /microposts | **create** | create a new micropost |
| GET | /microposts/1/edit | **edit** | page to edit micropost with id **1** |
| PATCH | /microposts/1 | **update** | update micropost with id **1** |
| DELETE | /microposts/1 | **destroy** | delete micropost with id **1** |

Table 2.3: RESTful routes provided by the Microposts resource in [Listing 2.7](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_microposts_resource).

The Microposts controller itself appears in schematic form [Listing 2.8](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_microposts_controller). Note that, apart from having**MicropostsController** in place of **UsersController**, [Listing 2.8](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_microposts_controller) is *identical* to the code in[Listing 2.3](http://ruby.railstutorial.org/chapters/a-demo-app#code-demo_users_controller). This is a reflection of the REST architecture common to both resources.

**Listing 2.8.** The Microposts controller in schematic form.   
**app/controllers/microposts\_controller.rb**

**class** **MicropostsController** < ApplicationController

.

.

.

**def** index

.

.

.

**end**

**def** show

.

.

.

**end**

**def** new

.

.

.

**end**

**def** create

.

.

.

**end**

**def** edit

.

.

.

**end**

**def** update

.

.

.

**end**

**def** destroy

.

.

.

**end**

**end**

To make some actual microposts, we enter information at the new microposts page,[/microposts/new](http://localhost:3000/microposts/new), as seen in [Figure 2.12](http://ruby.railstutorial.org/chapters/a-demo-app#fig-demo_new_micropost_rails_3).

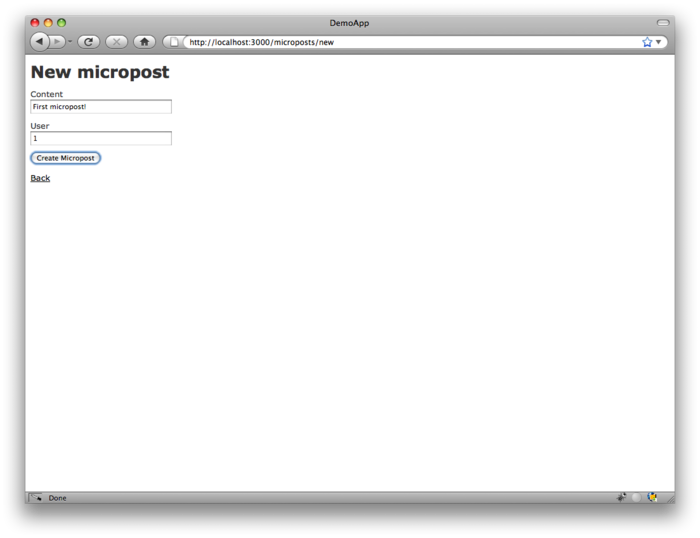


Figure 2.12: The new micropost page ([/microposts/new](http://localhost:3000/microposts/new)). [(full size)](http://railstutorial.org/images/figures/demo_new_micropost-full.png)

At this point, go ahead and create a micropost or two, taking care to make sure that at least one has a **user\_id** of **1** to match the id of the first user created in [Section 2.2.1](http://ruby.railstutorial.org/chapters/a-demo-app#sec-a_user_tour). The result should look something like [Figure 2.13](http://ruby.railstutorial.org/chapters/a-demo-app#fig-demo_micropost_index_rails_3).

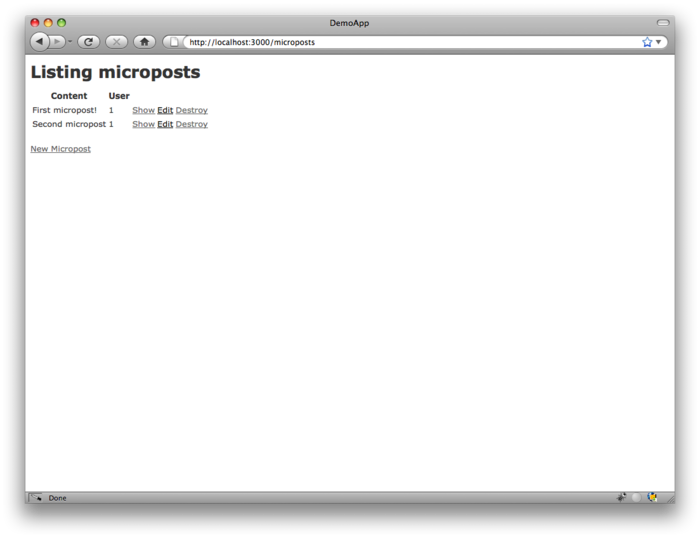


Figure 2.13: The micropost index page ([/microposts](http://localhost:3000/microposts)). [(full size)](http://railstutorial.org/images/figures/demo_micropost_index_rails_3-full.png)