Developing Facebook Platform

Applications with Rails

Beta Book

Agile publishing for agile developers

The book you're reading is still under development. As part of our Beta book program, we're releasing this copy well before we normally would. That way you'll be able to get this content a couple of months before it's available in finished form, and we'll get feedback to make the book even better. The idea is that everyone wins!

Be warned. The book has not had a full technical edit, so it will contain errors. It has not been copyedited, so it will be full of typos and other weirdness. And there's been no effort spent doing layout, so you'll find bad page breaks, over-long lines with little black rectangles, incorrect hyphenations, and all the other ugly things that you wouldn't expect to see in a finished book. We can't be held liable if you use this book to try to create a spiffy application and you somehow end up with a strangely shaped farm implement instead. Despite all this, we think you'll enjoy it!

Throughout this process you'll be able to download updated PDFs from your account on http://pragprog.com. When the book is finally ready, you'll get the final version (and subsequent updates) from the same address. In the meantime, we'd appreciate you sending us your feedback on this book at http://books.pragprog.com/titles/mmfacer/errata, or by using the links at the bottom of each page.

Thank you for being part of the Pragmatic community!

► **Andy Hunt**

Chapter 7

Profiles and the Facebook REST API

We've looked at how our users interact with our application. Now it's time to look at how we interact with Facebook. We'll start by looking at the Facebook REST API. We'll walk through the details of how it works and then will use it build a dojo page. Next, we'll look at the Facebook Query Language, or FQL and use it to improve our dojo page's performance.

Finally, we'll finish up by looking at one of the most popular uses of API calls; profile updates. We'll catch our profile up to the rest of Karate Poke. Along the way, we'll look at profile actions and the mobile profile as well.

7.1 Using the REST API

We've used the Facebook REST API indirectly several times. Let's look at it in a little more detail. We'll start by looking what happens each time we make a request. Next, we'll walk through some of the API methods that are available to us. Finally, we'll use the REST API to add more detail to our dojo page.

How the REST API works

To really understand how the REST API works, we'll step through an example. In our example, we will use script/console to retrieve our name from Facebook. Start script/console and find your User instance. Once you have your User object, run the code below.

```
    >>  user.facebook_session.user.name
=> "Mike Mangino"
```

That doesn't look like anything special because Facebooker encapsulates the Facebook API behind a very Ruby-like façade. Behind the scenes, Facebooker does a lot of work to retrieve our name. First, Facebooker sends a POST to the Facebook API service. It includes a number of parameters in the request such as the api_key of our application, the session_key of the user making the request and the uid of the user whose albums we want to retrieve. Additionally, Facebooker adds the fb_sig parameter as proof that our application is making the request. We talked about signatures earlier in Section 3.1, *The details of Facebook signatures*, on page 34. By requiring all API calls to be signed, Facebook can verify that requests are coming from an approved application.

In response to our request, Facebook will return an XML document similar to the one shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<users_getInfo_response xmlns="http://api.facebook.com/1.0/" ...>
 <user>
  <uid>12451752</uid>
  <status>
    <message/>
    <time>0</time>
  </status>
  <political/>
  <pic_small>http://profile.ak.facebook.com/profile...</pic_small>
  <name>Mike Mangino</name>
  <quotes/>
  <is_app_user>1</is_app_user>
  <tv/>
  <profile_update_time>0</profile_update_time>
  <meeting_sex list="true"/>
  <hs_info>
    <hs1_name>Westerville - North High School</hs1_name>
    <hs2_name/>
    <grad_year>1996</grad_year>
    <hs1_id>19941</hs1_id>
    <hs2_id>0</hs2_id>
  </hs_info>
  <timezone>-6</timezone>
  <relationship_status>Married</relationship_status>
  ...
```

When Facebooker gets the response, it turns the xml into Ruby objects. Each time Facebooker needs to load more information, it sends an HTTP request to Facebook. Each request takes time, typically between a quarter and a half of a second. We'll want to keep this timing in mind as we use the REST API.

Using the Facebook API

We've looked at several uses of the Facebook API already. Since Facebooker makes the Facebook API look just like regular Ruby objects, we aren't going to spend time looking at every object.[1] Instead, we'll look at a typical example of the API.

We're going to create a dojo page that lists the members of a dojo. We'll use the Facebook API to include our disciples hometown next to their name. Let's start by extending our User model with a hometown method. Our hometown method will need to access the hometown_location attribute on the Facebooker::User object.

```ruby
def hometown
  fb_user = Facebooker::User.new(facebook_id)
  location = fb_user.hometown_location
  text_location = "#{location.city} #{location.state}"
  text_location.blank? ? "an undisclosed location" : text_location
end
```

Our method starts by creating an instance of Facebooker::User to represent our user. From there, we retrieve the hometown_location. If the location is blank, we provide default text.

Now that we have a way of accessing a user's hometown, we need to build a controller and view. Let's create the DojosController for our dojo page. After we create that, we need to add a dojo resource.

Download chapter7/karate_poke/config/routes.rb

```ruby
map.resources :dojos
```

Now we can create our action. We'll use the show action for displaying a user's dojo. Our show action just needs to find a sensei and then retrieve their disciples.

```ruby
def show
  @sensei = User.find(params[:id])
  @disciples = @sensei.disciples
end
```

1. You can find the Facebook API documentation onine at http://wiki.developers.facebook.com/index.php/API. The Facebooker documentation is available at http://facebooker.rubyforge.org.

Now we can use the @disciples array to show our dojo in the show.fbml.erb file.

```
<% if @disciples.blank? %>
  <h2>You don't have any disciples
      <%= link_to "Go Invite Some",new_invitation_path%>
  </h2>
<% else %>
<% for disciple in @disciples %>
  <div class="disciple">
    <%= fb_profile_pic disciple,:size=>:thumb %>
    <%= name(disciple) %>
    From: <%=disciple.hometown%>
  </div>
<% end %>
<% end %>
```

That does exactly what we would expect. Unfortunately, that page takes a long time to load for large Dojos. I have a dojo with 40 disciples in it. It takes 28 seconds to load. Since Facebook will show a timeout page after only 8 seconds, I won't ever be able to see my dojo. In the past, we've used FBML tags to avoid this API performance penalty. Unfortunately, there aren't FBML tags for all Facebook fields. There's got to be a way to speed this up.

7.2 The Facebook Query Language

We've seen how easy it is to use the Facebook API to retrieve data about our users. We've also seen how slow it can be to retrieve more than just a small amount of data. To reduce the need for repeated API calls, Facebook created the Facebook Query Langauge, or *FQL*. FQL is very similar to SQL, the Structured Query Language. In fact, the syntax is almost identical. Instead of retrieving our disciples' hometowns one user at a time, FQL allows us to get the hometowns of all of our disciples with a single request.

Using FQL to retrieve information

Let's look at an example FQL query. To get my Hometown location, you can run the FQL select hometown_location from user where uid=12451752. You can experiment with FQL using the API test console.[2] Select the fql.query method from the "Method" dropdown. You can enter your query in the query box and click "Call Method" to see the result. If you run a

2. Available at http://developer.facebook.com/tools.php