

Extracted from:

Developing Facebook Platform

Applications with Rails

This PDF file contains pages extracted from Developing Facebook Platform, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.

Copyright © 2008The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.



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**

Creating Your First Application

So far, we've used the Facebook developer tool to create keys for an application. We've also configured Facebook to talk to our computer and to run a pre-built application. After all that setup, let's get started writing Karate Poke.

In this chapter we're going to add two features found in many social networking applications. We'll start by building an invitation system to allow our users to tell their friends about Karate Poke. From there, we'll add some content to our users' profiles.

Before we do that, let me explain what the "Poke" in Karate Poke means. Poke is a very simple Facebook application that lets you let another know you're thinking about them. When you poke a user, they just receive a message telling them that they were poked by you. Because of the simplicity of the application, developers quickly created a number of variations on the poke concept. Poke applications like "Super Poke" and "X Me" are some of the most popular applications on Facebook.

3.1 Creating a Facebook Rails Application

In the last chapter, we created an application with the Developer tool. Now we're going to write some code for Karate Poke.

Configuring Rails

First, we're going to need a Rails application. Let's create an application called `karate_poke`

```
$ rails karate_poke
```

```
...
```

Now that we have a Rails application, let's turn it into a Facebook application. We're going to need to install the Facebooker plugin. Facebooker is a Ruby library that knows how to talk to Facebook. It provides access to the Facebook API, some view helpers and the glue that makes our application part of Facebook. Run `script/plugin` to install the Facebooker. You will need to specify the full path to the Subversion repository like I did in the example below.

```
$ script/plugin install \
    http://facebooker.rubyforge.org/svn/trunk/facebooker/
+ ./CHANGELOG.txt
+ ./COPYING
```

It's time to do a little configuration to our Facebook application. This next step will look familiar to you. Installing Facebooker will create a `config/facebooker.yml`. Open this file and fill in the development section. You can either use the API key and secret key that came from the application you created in Chapter 2, *Getting Started with the Facebook Platform*, on page 18, or you can follow the same steps and create a new application. I'm lazy, so I'm going to use my existing Facebook application.

Since you've had to do this step twice already, you've probably guessed that these lines of code are important to a Facebook application. Let's look at what they actually do. The first two lines are our application's username and password. They authenticate our application to Facebook. Just as importantly, they let our application verify that requests are coming from Facebook. It sounds odd that we need to verify that requests come from Facebook. I'll explain why in Section 3.1, *The details of Facebook signatures*, on the following page.

The next line tells Facebooker what to use for our application's canvas path. We talked about the canvas path in Section 2.4, *Configuring our new Facebook Application*, on page 26. Facebooker will automatically include our canvas paths in all of our application's links.

The last line is used to tell Facebooker where to find our server. While all requests for our canvas pages go through Facebook, our images will be requested directly from our servers. By setting the `callback_url` parameter, Rails knows to use the hostname of our server instead of `apps.facebook.com`.

The details of Facebook signatures

One thing we won't do in this book is build a login controller. We can depend on Facebook to handle authentication for us. In fact, Facebook sends us the id of the current user and their whole list of friends on every request. That makes our life quite a bit easier. It also can cause some security problems.

In typical web development, your application would never include a logged in user's ID as a part of the URL. After all, a malicious user could change the user ID in the URL to access our application as a different user. Facebook development is a little different.

In Facebook development, we never talk to our users directly. All requests come from Facebook. To make sure this is the case, we can verify the signature that is sent by Facebook on every request. A digital signature is a way to use cryptography to verify that something actually came from the person who it appears to be from.¹ Facebook sends a number of parameters that start with `fb_sig`. All of these parameters are used in the signature validation.

When Facebook sends our applications a request, it builds a string that includes all of the `fb_sig` parameters in alphabetical order. It then add our Secret Key to the end of that string and calculates the MD5 sum.² Facebook then adds this signature to the request in the `fb_sig` parameter. When Facebooker receives a request, it goes through the same steps to re-calculate the signature. If the value that Facebooker calculates matches the one in our request it proves that the request came from somebody who knows our secret key. We also know that the request wasn't changed after it was sent (If the request was changed, the signatures wouldn't match.) If they don't match, then we know that the key the sender used to calculate the signature doesn't match our key. When that happens Facebooker will raise an exception. You will probably see this exception from time to time. It doesn't normally come from a forged request. It normally shows up when you've got the wrong key in your `facebooker.yml` file.

1. For more information see http://en.wikipedia.org/wiki/Digital_signature

2. MD5 is a cryptographic one way hash function. You can learn more at <http://en.wikipedia.org/wiki/MD5>

Setting up the Controllers

Now that we have Rails configured, we need to do a little setup in the Application controller. Since Facebook limits the information we can learn about users who haven't installed our application, let's force all of our users to add Karate Poke. Facebooker provides a filter to do that. Let's add that to our Application controller.

[Download](#) chapter3/karate_poke/app/controllers/application.rb

```
# Filters added to this controller apply to all controllers in the application.
# Likewise, all the methods added will be available for all controllers.
```

```
class ApplicationController < ActionController::Base
  helper :all # include all helpers, all the time

  # See ActionController::RequestForgeryProtection for details
  # Uncomment the :secret if you're not using the cookie session store
  protect_from_forgery :secret => 'a7cabcdf1499df9ded55d8a3797d9387'
  ensure_application_is_installed_by_facebook_user
end
```

Now that we have our filter in place, we'll also need to change the way Rails handles sessions. By default, Rails uses cookies to store session information. Since Facebook sits between us and our users, we won't be able to count on cookies for session storage. Instead, we'll store our session information in the database.

First, we'll need to run `script/generate session_migration` to create all of the right tables. Next, we'll need to uncomment a line in `environment.rb`

```
config.action_controller.session_store = :active_record_store
```

With that in place, we're ready to start coding.

3.2 Sending an Invitation

Now that we've configured our application it's time to actually write code. Since we're going to start by building an invitation system, let's start by creating the Invitations controller.

Creating the invitation controller

We can create our controller using the Rails generator.

```
$ script/generate controller invitations
```

We're going to use the new RESTful³ features of Rails to build Karate Poke. REST stands for Representational State Transfer. Starting with version 1.2, Rails provides support for creating applications using REST principals. These conventions tells us that we should use the new action to show a form. Before we do that, let's set up our routes.

```
map.resources :invitations
```

With that little bit of setup out of the way we're ready to create our view.

Creating the invitation form using FBML

Now that we've got a controller and a route, we need to create a view.⁴

```
Download chapter3/karate_poke/app/views/invitations/new.erb
```

```
<fb:fbml>
  <fb:request-form
    action="<%=new_invitation_path%>"
    method="POST"
    invite="true"
    type="Karate Poke"
    content="I added a cool application." >
    <fb:multi-friend-selector
      showborder="false"
      actiontext="Invite your friends to use Karate Poke." />
  </fb:request-form>
</fb:fbml>
```

That looks a little like HTML, but I bet you've never seen those tags before. What you're seeing is FBML, the Facebook Markup Language. FBML is one of the most powerful features of the Facebook platform. It acts as an extension to HTML that gives you some prebuilt user interface components. FBML is translated into normal HTML when Facebook process your page.

Before we get into the details of how the view works, let's see what it does. Start up `script/server` and then open a browser to your invitation page.⁵ Make sure you're logged in as one of your test users. You should see something that looks like Figure 3.1, on the following page. If it doesn't, make sure you've copied the code exactly. You should also make sure that the test user you're using has at least one friend.

3. If you haven't used the new RESTful Rails features, Geoffrey Grossenbach provides a great overview at <http://peepcode.com/products/restful-rails>

4. The naming convention for views changed between Rails 1.2 and Rails 2.0. Templates now end in `.erb` instead of `.rhtml` by default.

5. http://apps.facebook.com/«canvas_path»/invitations/new

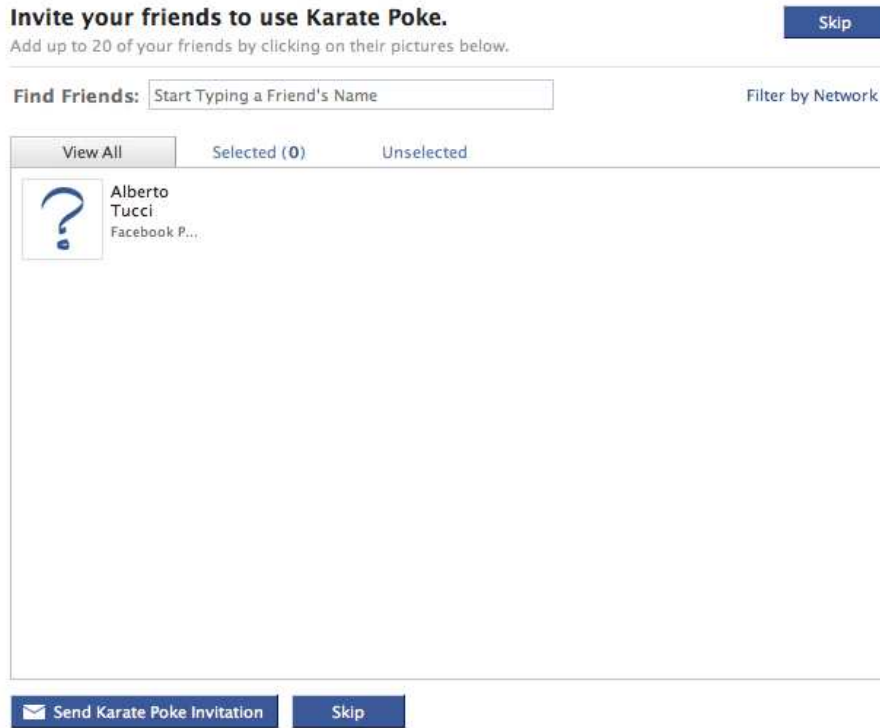


Figure 3.1: Our Facebook Invitation Page

That's pretty impressive for just a few simple lines of code! Since there are only three different FBML tags in our view, let's look at each one. The first tag, `<fb:fbml>` is very similar to the `<html>` tag. It marks the start of an FBML document. Your canvas pages will still work without the `<fb:fbml>` tag, but it is required for profile pages. It's best to get into the habit of always wrapping your pages in an `<fb:fbml>` tag.

The next tag, `<fb:request-form>`, starts a special type of Facebook form. This special form allows your application to create Facebook requests. It takes most of the normal form parameters like `action` and `method` along with two additional parameters, `type` and `content`. The `type` parameter specifies what text shows up on the submit button for the form. It's used to tell the user what type of request they are sending. The `content` parameter gives the body of the message that is sent to the selected users.

Working with multiple users

Since Facebook is a social platform, you should plan to spend a lot of time using two accounts. We'll spend a lot of time sending requests and notifications between two users. You'll get really frustrated if you have to log out of one account and log in to another every time you want to view the invitation you just sent.

To make life a little easier, I normally use two different browsers, either Firefox and Safari or Firefox and IE depending upon my current development platform. This will allow you to stay logged in to both accounts at once. As a bonus, you can test your CSS in multiple browsers at the same time too!

The final tag, `<fb:multi-friend-selector>` is responsible for rendering the actual friend selector. Facebook provides several different friend selectors. They all achieve the same goal, but have different interfaces. This selector allows you to click on images for your friends and includes multiple tabs. We'll look at another friend selector in Section 5.1, *Building our first form*, on page 67.

If it feels like I'm going really fast, don't worry. I'll cover FBML in a lot more detail in Chapter 5, *Getting Into the Facebook Canvas*, on page 66. Facebook also provides a wiki⁶ that includes detailed FBML documentation.

So let's try out the invitation functionality. Use the friend selector to send a request to one of your test accounts. We set the action of the form to `new_invitation_path()` so we'll end up right back on this page after sending an invitation. Try that out now. It's pretty hard to tell that your request was sent. Let's make a note to fix that later.

Now that we've sent a request, let's see what it looks like for the recipient. Log in as the user you sent the request to and take a look at the upper right corner of your home page. You should see a message telling you that you have a new request. When you click on that request, you will be taken to a page where you can read the message. It's nice that you can view the message, but it's definitely a little boring. It certainly

6. <http://wiki.developers.facebook.com/index.php/FBML>

The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style, and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

Developing Facebook Platform Applications With Rails' Home Page

<http://pragprog.com/titles/mmfacer>

Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates

<http://pragprog.com/updates>

Be notified when updates and new books become available.

Join the Community

<http://pragprog.com/community>

Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy

<http://pragprog.com/news>

Check out the latest pragmatic developments in the news.

Buy the Book

If you liked this PDF, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragprog.com/titles/mmfacer.

Contact Us

Phone Orders:	1-800-699-PROG (+1 919 847 3884)
Online Orders:	www.pragprog.com/catalog
Customer Service:	orders@pragprog.com
Non-English Versions:	translations@pragprog.com
Pragmatic Teaching:	academic@pragprog.com
Author Proposals:	proposals@pragprog.com