
Making Twitter mobile client with Ruby using Rhodes Rhosync (Part 3)

21Feb09

Previously

- Part One - Syncing public Tweet on Rhosync server
- Part Two - Displaying public Tweet on iPhone Simulator

At the previous 2 posts, we went through the very basic of how you write code in Rhosync and Rhodes.

This time, we go through the similar steps,but additional functionalities to read and create your own tweet.

Creating New source at Rhosync

Let's first create new source, just in the same way as you created the public timeline at Part 1, but new name as "MyTimeline"

```
$ rhogen source MyTimeline
Generating with source generator:
     [ADDED]  lib/my_timeline.rb
```

Authentication and My Tweet.

I initially thought I have to implement some authentication logic at "login" method, but realized that none of REST based examples (Basecamp and Lighthouse) have logic at login method.

Instead, it uses Net::HTTP's authentication mechanism within query method.
Here is an example of Basecamp adapter.

```
def login
  # intentionally left blank
end

def query
  uri = URI.parse(@source.url+"/projects.xml")
  req = Net::HTTP::Get.new(uri.path, 'Accept' => 'application/xml')
  req.basic_auth @source.login, @source.password
  response = Net::HTTP.start(uri.host,uri.port) do |http|
    http.request(req)
  end
  xml_data = XmlSimple.xml_in(response.body);
  @result = xml_data["project"]
end
```

According to Net::HTTP rdoc, you can call "basic_auth" instance method with login and password parameter.

The below is query method for MyTimeline. It's pretty much identical. Only the differences are url and response parameter.

Do you remember that you can access url info you added at Rhosync Web interface through @source? Both login and password are also available through it, so you don't need to hardcode at the source code.

The rest (sync and other private methods) are exactly same as PublicTimeline

```
def query
  url = "/statuses/friends_timeline.xml"
  uri = URI.parse(@source.url+url)

  req = Net::HTTP::Get.new(uri.path, 'Accept' => 'application/xml')
  req.basic_auth @source.login, @source.password

  res = Net::HTTP.start(uri.host,uri.port) do |http|
    http.request(req)
  end

  xml_data = XmlSimple.xml_in(res.body);
  @result = xml_data["status"]
end

def sync
  log "#{self.class} sync, with #{@result.length} results"
  @result.each do |item|
    item_id = item["id"].first.to_i
    iterate_keys(:item => item, :item_id => item_id)
  end
end

private
  def iterate_keys(option)
    item = option[:item]
    item_id = option[:item_id]
```

```
  prefix = option[:prefix] || ""

  # item.keys => ["user", "favorited", "truncated"...]
  item.keys.each do |key|
    value = item[key] ? item[key][0] : ""
    if value.kind_of?(Hash) && value != {}
      # eg. :user => {:url => 'foo} becomes user_url
      iterate_keys(:prefix => (prefix + underline(key) + "_"), :item => value, :item_id => item_id)
    else
      # This method is from rest_api_helper
      add_triple(@source.id, item_id, prefix + underline(key), value)
    end
  end
end

def underline(key)
  key.gsub('-','_')
end
```

Let's start up the server and add the source as you did for PublicTimeline, **plus actual login and password of your twitter account**. If you click "Show Records", you should be able to see your friend's tweets.

Creating Tweet

Create method does not have a lot of examples at sample apps. Most of REST adapters (BaseCamp and Lighthouse) do not have create method for some reason.

I finally found that "lighthouse_tickets.rb" has some examples.

```
def create(name_value_list)
  log "LighthouseTickets create"

  get_params(name_value_list)
  xml_str = xml_template(params)

  uri = URI.parse(base_url)
  Net::HTTP.start(uri.host) do |http|
    http.set_debug_output $stderr
    request = Net::HTTP::Post.new(uri.path + "/projects/#{params['project_id']}/tickets.xml", {'Content-type' => 'application/xml'})
    request.body = xml_str
    request.basic_auth @source.credential.token, "x"
    response = http.request(request)
    # log response.body

    # case response
    # when Net::HTTPSuccess, Net::HTTPRedirection
    #   # OK
    # else
    #   raise "Failed to create  ticket"
    # end
  end
end
```

What they are doing here is passing name_value_list into requrest.body in xml format. Looks get_params method does all conversion work. Let's look at the actual method within RestAPIHelpers

```
module RestAPIHelpers
  # convert name_value_list to a params hash
  # name_value_list example, "[{'name' => 'title', 'value' => 'testing'},{'name' => 'state', 'value' => 'new'}]"
  # => params['title'] = 'testing', etc.
  def get_params(name_value_list)
    @params = {}
    name_value_list.each do |pair|
      @params.merge!(Hash[pair['name'], pair['value']])
    end
    log @params.inspect
  end
```

Ah, here is a useful comment.
The comment says that get_params iterates through each array and converts {'name' => 'title', 'value' => 'testing'} hash into {'title' => 'testing'} hash and set the result into @params instance variable. If you see further bottom of the RestAPIHelpers, you can see "params" method which simply returns @params

```
def params
  @params
end
```

The below is how I implemented the create method for MyTimeline.

```
def create(name_value_list)
  url = "/statuses/update.xml"
  uri = URI.parse(@source.url+url)

  get_params(name_value_list)

  req = Net::HTTP::Post.new(uri.path, 'Accept' => 'application/xml')
  req.basic_auth @source.login, @source.password

  req.set_form_data('status'=> params["text"])

  res = Net::HTTP.start(uri.host,uri.port) do |http|
    http.request(req)
  end
end
```

This is very similar to “query” method, but here are the few differences.

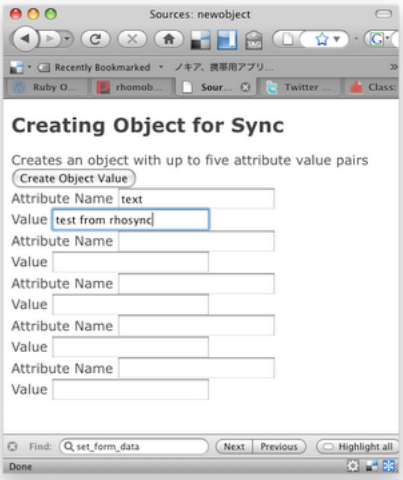
- Specified url as “update.xml”
- Net::HTTP::Post.new. Note this is Post class, not Get class.
- Passed text value using set_form_data method in Hash format. This way, you don’t have to convert your data into xml format and you can post to either “update.xml” or “update.json” url

Final Ryoshnc code

```
class MyTimeline < SourceAdapter

  include RestAPIHelpers

  def initialize(source, credential = nil)
    super
  end

  # Looks not in use
  # def login
  #
  # end

  def query
    url = "/statuses/friends_timeline.xml"
    uri = URI.parse(@source.url+url)

    # Login authentication. The logic is taken from basecamp_projects.rb
    req = Net::HTTP::Get.new(uri.path, 'Accept' => 'application/xml')
    req.basic_auth @source.login, @source.password

    res = Net::HTTP.start(uri.host,uri.port) do |http|
      http.request(req)
    end

    xml_data = XmlSimple.xml_in(res.body);
    @result = xml_data["status"]
  end

  def sync
    log "#{self.class} sync, with #{@result.length} results"
    @result.each do |item|
      item_id = item["id"].first.to_i
      iterate_keys(:item => item, :item_id => item_id)
    end
  end

  def create(name_value_list)
    url = "/statuses/update.xml"
    uri = URI.parse(@source.url+url)

    get_params(name_value_list)

    req = Net::HTTP::Post.new(uri.path, 'Accept' => 'application/xml')
    req.basic_auth @source.login, @source.password

    req.set_form_data('status'=> params["text"])

    res = Net::HTTP.start(uri.host,uri.port) do |http|
      http.request(req)
    end
  end

private
  def iterate_keys(option)
    item = option[:item]
    item_id = option[:item_id]
    prefix = option[:prefix] || ""

    # item.keys => ["user", "favorited", "truncated"...]
    item.keys.each do |key|
      value = item[key] ? item[key][0] : ""
      if value.kind_of?(Hash) && value != {}
        # eg. :user => {:url => 'foo} becomes user_url
        iterate_keys(:prefix => (prefix + underline(key) + "_"), :item => value, :item_id => item_id)
      else
        # This method is from rest_api_helper
        add_triple(@source.id, item_id, prefix + underline(key), value, @source.current_user.id)
      end
    end
  end

  def underline(key)
    key.gsub('-','_')
  end
end
```

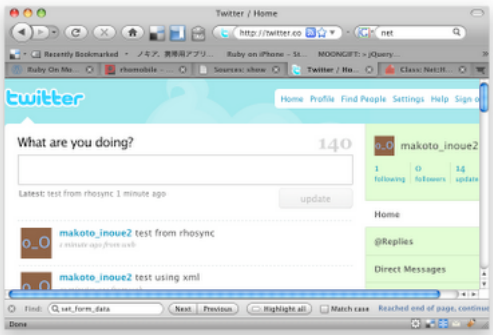
Demo

Let’s go back to Rhosync server. You should have left it when you displayed your friends tweet. Click “Create object-value record”, and add “text” attribute and your own tweet message like below.



Once you click “Create Object Value”, then you should be redirected to show page with “Created Object” flash message.

Now you go to your Twitter website. *Congratulations!!* Your message from Rhosync is successfully updated into Twitter itself!!



Next

You can finally post your own post from iPhone simulator. I will also go through some technics to tidy up your view pages.

Filed under: Rhodes | 0 Comments
Tags: tutorial

Aplix: Converting NTT Docomo i-appli to Windows Mobile Android, iPhone, S60 and even to portable game machine

17Feb09

It’s not ruby related, but I thought this is interesting news. The google translation of this Japanese article is actually not bad, but I will summarize key points.

- Aplix is a technology solution company to provide Java development platform to Japanese mobilie network providers, such as NTT Docomo, KDDI and Softbank mobile (Used to be J -phone and vodafone).
- i-appli is Java development platform for NTT Docomo. More detail is here
- Aplix developed a tool to convert binary files of i-appli to native application of target platforms.
- It automatically support touch panel, view rotation. Also supports flick operation for iPhone only.
- Aplix is showing the demo at WMC right now.

After a bit more research on this topic, I found interesting note from “ITPro“(Japanese)

- Aplix business model is commission based to software vendor per sales basis(up to 5~6 %)

Looks their business model is similar to Rhodes, though they do not seem open source.

When I checked Aplix’s website, there was press release only in Japanese

There are not much technical detail on the press release, but it’s worth noting that there are lots of comment from Japanese Game development company such as Capcon, Sega, Bandai and so on.

Looks their main target is Japanese game development companies. Since iPhone game applications are getting so popular, there could be huge business opportunities for Japanese game companies to get into all smartphone market.

But what will be the benefit for NTT docomo which offered technical advice to Aplix?

At a first glance, letting all game companies to provide a way to develop iphone/android apps rather than iAppli sounds silly move , as they can no longer “lock in” all vendors. However, I saw one Japanese blog speculating that that’s the strategy of NTT docomo to take the standard of mobile application development using their Java platform, so that they can protect game vendors to switch their platform out of iAppli.

If you are attending WMC in Barcelona, please visit their booth and let me know how smooth their conversion program is.

It seems Adobe is also making progress for their Flash support to smartphone (iPhone is not to be supported though).

It is interesting that many companies are now trying to support cross mobile development platform.

What would you choose?

Filed under: Uncategorized | 0 Comments
Tags: poll

Making Twitter mobile client with Ruby using Rhodes & Rhosync (Part 2)

13Feb09

What we have done at previous tutorial.

At the previous post, we created an Rhosync adapter to fetch public timeline of Twitter. Take a note of the url where you displayed the record, because you will need it when you build rhodes app.

In my case, the url is like this.

```
http://localhost:3000/apps/6/sources/11
```

“6” is id for the application(Tweeter), and 11 is id of the source(Public Timeline). The numbers very in your environment.

Setup

As specified at Rhomobile tutorial, let’s download the latest Rhodes from github.

```
git clone git://github.com/rhomobile/rhodes.git rhodes
```

Before building new app

When I first played with SugarCRM sample which rhomobile provided, I really struggled running it on my iPhone emulator. You can see the full detail at this thread, but I will also summarize it again with some visual aid.

Unify source url into one.

You will notice that there are lots of sample application directories (Lighthouse, SugarCRM, etc) under “apps” directory. I recommend that you either remove them, move to different directory, or edit “config.rb” files of every single models under every single probjects pointing source url from “http://rhosyncdev.rhohub.com” to “http://localhost:3000” . This is because Rhodes try to connect to each url at login and fails if it fails to connect to any url specified at the config file.

```
$cd rhodes
$cd apps
$grep source */*/config.rb
Basecamp/People/config.rb:#Rho::RhoConfig::add_source("People", {"url"=>"http://rhosyncdev.rhohub.com/apps/1/sources/6", "source_id"=>6})
```

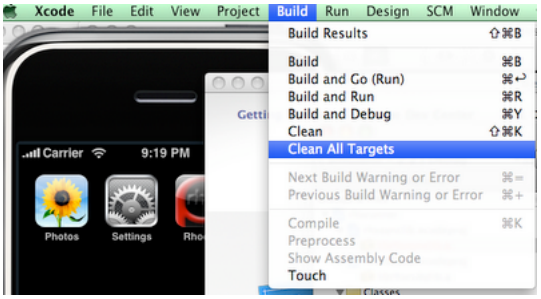
```
Lighthouse/LighthouseSettings/config.rb:Rho::RhoConfig::add_source("LighthouseSettings", {"url"=>"http://rhosyncdev.rhohub.com/apps/4/sources/10",
Lighthouse/Milestones/config.rb:Rho::RhoConfig::add_source("Milestone", {"url"=>"http://rhosyncdev.rhohub.com/apps/4/sources/6", "source_id"=>6})
Lighthouse/Project/config.rb:Rho::RhoConfig::add_source("Project", {"url"=>"http://rhosyncdev.rhohub.com/apps/4/sources/5", "source_id"=>5})
Lighthouse/Ticket/config.rb:Rho::RhoConfig::add_source("Ticket", {"url"=>"http://rhosyncdev.rhohub.com/apps/4/sources/7", "source_id"=>7})

$mv  Basecamp/ Lighthouse/ RhoSiebel/ RhoSugarCRM/ Phonebook/ ..
```

Do a clean build in xcode.

sqlite local db holds session information unless you completely wipe
out app from the simulator. Here are the steps to wipe out completely.
Make sure you do this every time you modified your code under rhodes directory
(not just when you are tweaking rhodes framework itself, but also when you write your code under “app”
directory)

1. Do “reset content and settings” then from the iphone simulator
   menu

2. Remove iphone/build/Debug-iphonesimulator director from Rhodes
   project directory.
3. “Build” => “Clear all targets” on Xcode.



4. “Build” => “Build and Go” from xcode.

The steps are quite tedious. I’d love to know if there are any other ways to simplify the steps. This is when I miss the simplicity of scripting environment…

Log out /login from simulator.

For some reason, rhodes thinks I was logged in at iphone simulator. If
first login did not work, cancel the login page, go back to main page,
click “Logout”, then login again (This used to happen at 0.2.0, but seems fixed at 0.3.0).

Make sure port 8080 is not in use.

The iphone simulator uses http://localhost:8080 for it’s internal web server to render user interface. I was once running Tomcat at port 8080 for my work related product, and my mobile app window at the simulator was showing this message. Took a few minutes figuring out why it was showing such a message.

rhogen app

You must remember that you used “rhogen” to create an adapter at Rhosync.
You will use the same command, but now with different option.

```
$rhogen app Twitter
Generating with app generator:
     [ADDED]  Twitter/application.rb
     [ADDED]  Twitter/index.html
```

Before moving to the directory, please insert a link to the app page at “apps/index.erb” like below.

```
<h4>Sample Apps</h4>
```

```
<a href="Twitter">Twitter</a></br>
```

rhogen model

Now is the time to create a controller which sync with the resource model you created at Rhosync.

```
$cd Twitter
$rhogen model PublicTimeLine "http://localhost:3000/apps/6/sources/11" 11 text,user_screen_name,user_name,user_profile_image_url,source
Generating with model generator:
[ADDED] PublicTimeLine/config.rb
[ADDED] PublicTimeLine/index.erb
[ADDED] PublicTimeLine/edit.erb
[ADDED] PublicTimeLine/new.erb
[ADDED] PublicTimeLine/controller.rb
```

The syntax should be straightforward. it specify controller name (PublicTimeLine), resource url(http://localhost:3000/apps/6/sources/11), resource id(11, actually I feel this is duplicated info as resource url contains the resource id itself), and all the attributes(text,user*screen*name,user*name*,*user*profile*image*url,source).
Make sure you put no space between commas.

You don't really need to touch the controller you just created for now, as rhogen made controller with basic CRUD. Just insert link to the "PublicTimeLine/index" page at at "Twitter/index.html" like below.

```
<ul id="home" selected="true" title="Twitter">
  <li>Something interesting here...</li>
  <li><a  href="PublicTimeLine"> Public Tweet</a></li>
```

View

Last part is a view. The below is PublicTimeLine/index.erb created by rhogen.

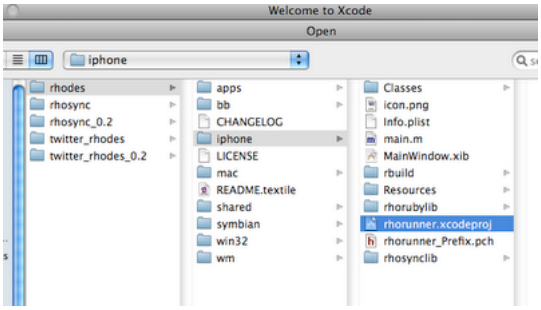
```
<ul id="PublicTimeLines" title="PublicTimeLines">
<%@PublicTimeLines.each do |x|%>

<li><%=link_to "#{x.text}", "edit", x.object%></li>

<%end%>
<li><font color="blue"><%=link_to "New PublicTimeLine", "new"%></font></li>
</ul>
```

Let's change it to show all the info we fetched from rhosync.

```
<ul id="PublicTimeLines" title="PublicTimeLines">
<%@PublicTimeLines.each do |x|%>
  <li class ="row">
    <img src=<%= x.user_profile_image_url %> alt=<%= x.user_screen_name %> />
     <%= x.text %>
     <%= x.user_name %> <%= x.created_at %> via <%= x.source %>
  </li>
<%end%>
</ul>
```

Displaying on Emulator

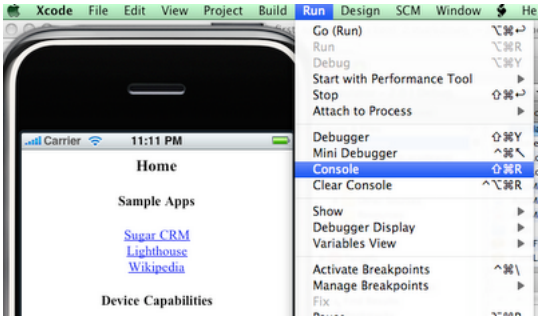
Now it's time to open the project at Xcode.



Before running "Build & Go", I recommend that you go through the clean up process I explained earlier.

Once build is complete, iPhone simulator will popup, and Rhodes application will launch itself. Also, make sure you Rhosync Rails server is up and running.

It's good time to open up console, as well as displaying script/server log, so you can see what's going on at the backend.



Rhodes on iPhone Simulator

Finally we managed to display public tweet on our iphone simulator!!

The view is rendered by iui css and javascript. It will show like normal iPhone UI, as long as you specify each tweet within `<li class="row">`. I also made other view changes such as displaying date in time distance (eg: 3 hours ago), but I will cover that at later time.

Next

At next post, I will go through similar step we did at Part 1 and 2, but we will display your own friend feed, as well as being able to post tweet from iphone simulator.

Filed under: Rhodes | 0 Comments
Tags: tutorial

Making Twitter mobile client with Ruby using Rhodes & Rhosync (Part 1)

10Feb09

Once you play with sample applications provided from rhomobile, now you might want to make something on your own.

I've seen lots of desktop client app tutorial with "Creating Sample Twitter App" examples, so I also tried something similar.

I will try to explain my code examples step by step including error messages and mistakes I made, so that you can not only experience what it is like to develop mobile app using Rhodes/Rhosync, but also debug similar problems when you make your own application.

In this tutorial, I will start from minimum functionality to display Twitter's public timeline on iPhone simulator. I will then implement authentication and basic Read/Create actions , so that you can show your friends's tweet, as well as post your own tweet from the iPhone simulator.

All the codes will be tested against 0.3.1 tags for both rhosync and rhodes.

Before I start, special thanks to Rhomobile support team for answering so many questions at google group.

What is Twitter API?

Twitter is one of the most post popular Ruby On Rails based web app. It is a micro blogging service where you can post your status in small text(140 chars).

Following Ruby On Rails convention, they have very clean (
REST based APIs). If you have never tried their API, click http://twitter.com/statuses/public_timeline.xml.
You will see list of 20 public tweets in xml format like below.

```
<status>
  <created_at>Sat Feb 07 23:49:04 +0000 2009</created_at>
  <id>1187577382</id>
  <text>
    @ceetee dude! How's it like out there! Been there before?
  </text>
  <source>
    <a href="http://www.atebits.com/software/tweetie/">Tweetie</a>
  </source>
  <truncated>false</truncated>
  <in_reply_to_status_id>1187570116</in_reply_to_status_id>
  <in_reply_to_user_id>8620122</in_reply_to_user_id>
  <favorited>false</favorited>
  <user>
    <id>11686132</id>
    <name>bombayaddict</name>
    <screen_name>b50</screen_name>
    <description>Bombay's Addict</description>
    <location>Bombay. Always. </location>
    <profile_image_url>
    http://s3.amazonaws.com/twitter_production/profile_images/63468508/photo2_normal.jpg
    </profile_image_url>
    <url>http://www.bombayaddict.com</url>
    <protected>false</protected>
    <followers_count>462</followers_count>
  </user>
</status>
```

If you change file extension from .xml to .json or .rss, they will show in each format. Unlike your private tweets which I cover later, accessing public tweets does not require authentication.

There are a few Ruby Libraries to handle their API, but the APIs are so easy that you can do most of what you want to do by just using basic Ruby network library, such as net/http.

Using Rhosync

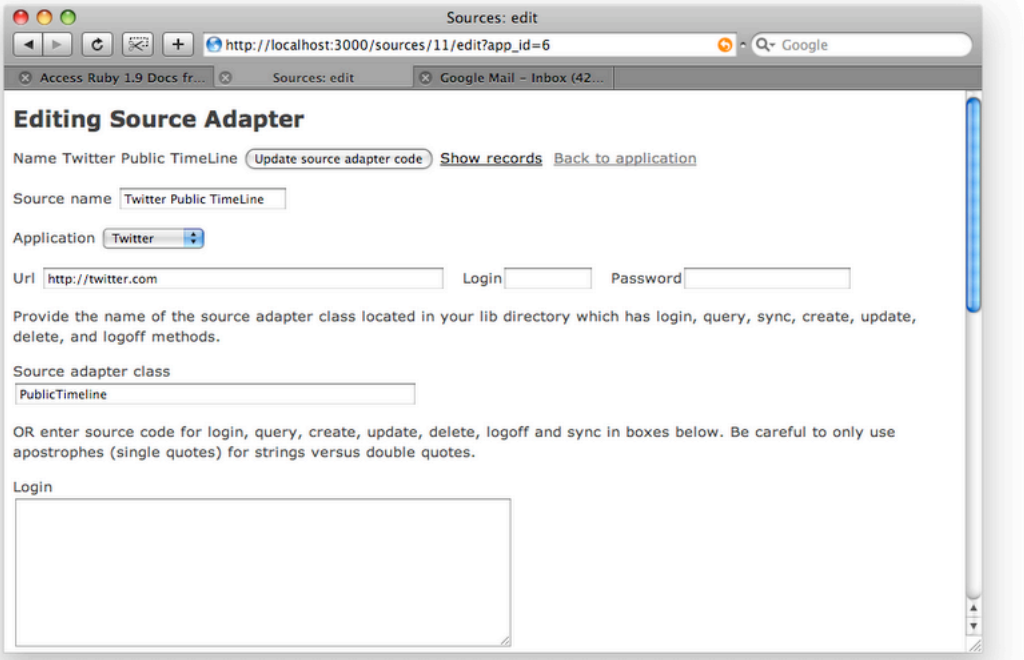
Rhosync will let you define common CRUD(Create, Read, Update, Delete) interfaces for mobile device via Rhodes.

Rhosync is built as a pure Ruby On Rails based application, so you should not have any problems if you know how to run and develop using Ruby On Rails.

Registering new source at Rhosync web form.

After you install Rhosync following the instruction, let's startup the server, register your account, login to the server, and create an application called "Twitter". Once application is created, click "Add Source" and create an source called "Twitter Public Timeline"

Add "http://twitter.com" as url, but you can leave login and password for now, as Public Timeline does not require authentication.



To setup adapter, you can either generate the adapter using “rhogen” command, or type source code directly from the web form. I choose generating adapter myself for better debugging purpose. If your adapter code has some bugs, any exception will point the line number which contains the bug. If you type code in the form, your code is kept at database and get evaluated dynamically when required, but exception only says your eval failed, which is harder to debug.

If you go for generating adapter, just type “PublicTimeline” at “Source Adapter Class”

rhogen

you need to create an adapter to talk to Twitter. Rhosync has some built in code generator, so creating the skeleton is very easy.

```
$ rhogen source PublicTimeline
Generating with source generator:
     [ADDED]  lib/public_timeline.rb
```

This creates long list of code defining login, query, sync, create, update, delete, and logoff.

For this very first application, all you need to worry about are “query” and “sync” methods.

query method

Here is my mimimalist code to fetch query from twitter.

```
def query
  uri = URI.parse(@source.url)
  res = Net::HTTP.start(uri.host, uri.port) {|http|
    http.get("/statuses/public_timeline.xml")
  }
  xml_data = XmlSimple.xml_in(res.body);
  @result = xml_data["status"]
end
```

I think it can’t be as simple as this. It take url from @source instance variable. @source.url is the url you added when you first created the source from web page. Then it fetches tweets and put the body of the response into XMLSimple library. Lastly I put each entry under into @result instance variable as Array.

sync method

Next is “sync” method. The code repopulated by rhogen command iterates through each entry of @result and put value into ObjectValue object. ObjectValue is a table where you can store the result of API response as key => value pair. This way, you can

The below are the difference between normal ORM(Object Relational Mapping) object table and ObjectValue table.

Normal table

```
ID                                   |  Industry  | Name
44e804f2-4933-4e20-271c-48fcecd9450d | Technology | Mobio India
69ae8551-c0bd-0cb3-5c25-48ff9bae965a | Finance    | vSpring
```

ObjectValue table

```
Object                               | Attribute | Value
44e804f2-4933-4e20-271c-48fcecd9450d   | industry  | Technology
44e804f2-4933-4e20-271c-48fcecd9450d   | Name      | Mobio India
69ae8551-c0bd-0cb3-5c25-48ff9bae965a | industry  | Finance
69ae8551-c0bd-0cb3-5c25-48ff9bae965a | Name      | vSpring
```

What I did not understand much was where “entry*list” and “name*value_list” attributes are coming.

At the rhomobile tutorial
, @result is set like below at query method.

```
 result = client.get_entry_list(session_id,'Account','','',0,['name','industry'],10000,0)
```

And it also explains that “client” variable is available and not for REST API.

The Rhomobile tutorial page shows example using SugarCRM which uses SOAP API. Since Twitter uses REST API, I referred to Lighthouse and BaseCamp examples which are also Rails based app and uses REST API.

By examining these codes, you notice that there is a helper called RestAPIHelpers . Let’s use that. I included the helper at the top of the class like this

```
class PublicTimeline < SourceAdapter

  include RestAPIHelpers
```

They use “add_triple” method which also adds your API result into ObjectValue, so I decided to use the method to inject my Tweets.

```
def sync
  @result.each do |item|
```

```
    item_id = item["id"].first.to_i
    item.keys.each do |key|
      value = item[key] ? item[key][0] : ""
      add_triple(@source.id, id, key.gsub('-','_'), value)
      # convert "-" to "_" because "-" is not valid in ruby variable names
    end
  end
end
```

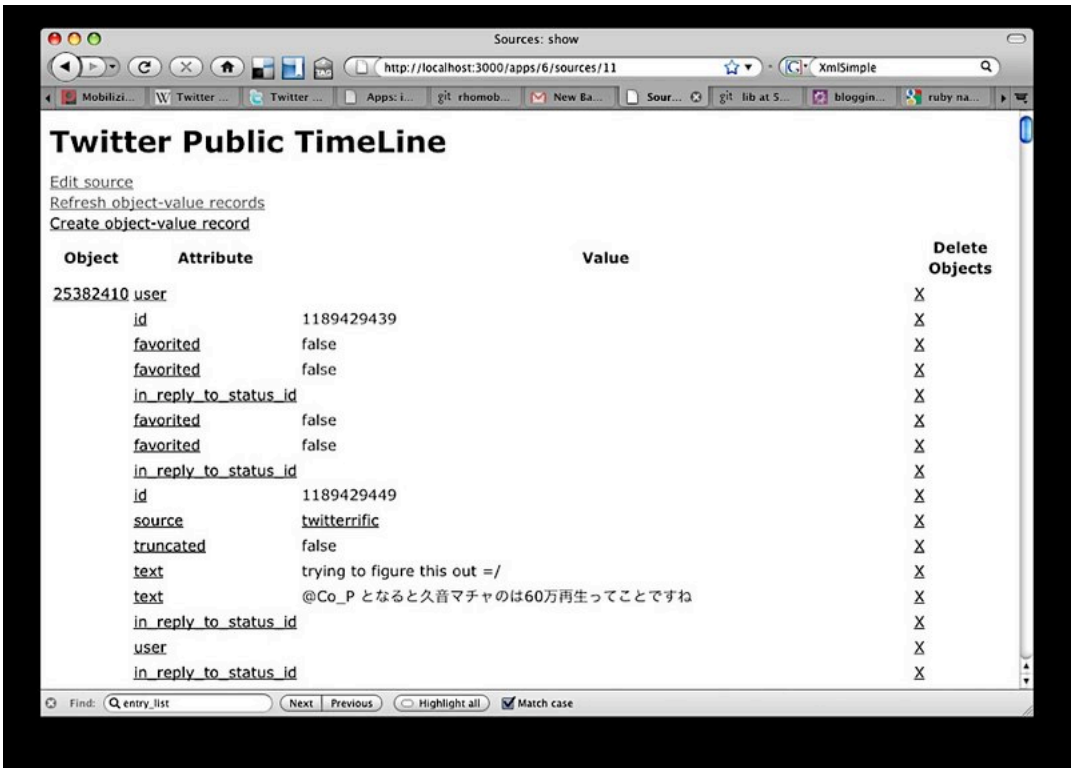
This is also simple one. @result contains array of Tweet where each tweet contains data in hash format like below.

```
  {
    "text"=>
    ["@ceetee dude! How's it like out there! Been there before?"],
    "id"=>["1187577382"],
    "created_at"=>["Sat Feb 07 23:49:04 +0000 2009"]
  }
```

I iterated through each hash key and send its key and value into “add_triple” methods

```
  @result.first.keys
  => ["text", "id", "created_at"]
```

Now it’s done. Go back to “Editing Source Adapter” page, click “Show records”, then “Refresh object-value records”. This shouldwill show all recent tweets.



Ummm, certain values are missing, such as “user”. By looking at original xml, you should see that certain attributes are nested, so just going through each key of hash won’t work. You need to iterate recursively every time value contains hash.

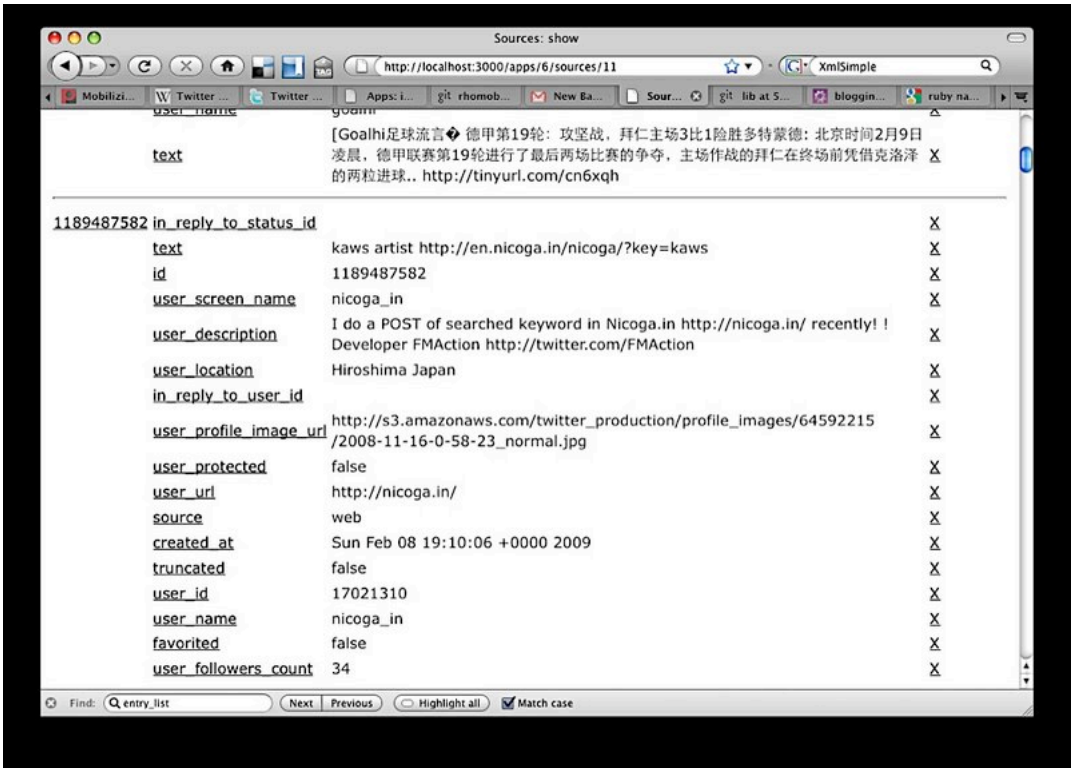
Final code

Here is the complete code including the change to recursively iterate through each tweets. I refactored the code by separating iteration part into “iterate*keys” and calls the method recursively if the hash value contains another hash. When you call iterate*keys, it also adds prefix, so “user” => {”screen*name”} will be stored into “user*screen_name” key name at ObjectValue

```
class PublicTimeline < SourceAdapter

  include RestAPIHelpers

  def initialize(source)
    super
  end

  def query
    uri = URI.parse(@source.url)
    res = Net::HTTP.start(uri.host, uri.port) {|http|
      http.get("/statuses/public_timeline.xml")
    }
    xml_data = XmlSimple.xml_in(res.body);
    @result = xml_data["status"]
  end
```

```
  def sync
    @result.each do |item|
      item_id = item["id"].first.to_i
      iterate_keys(:item => item, :item_id => item_id)
    end
  end

private
  def iterate_keys(option)
    item = option[:item]
    item_id = option[:item_id]
    prefix = option[:prefix] || ""

    # item.keys => ["user", "favorited", "truncated"...]
    item.keys.each do |key|
      value = item[key] ? item[key][0] : ""
      if value.kind_of?(Hash) && value != {}
        # eg. :user => {:url => 'foo} becomes user_url
        iterate_keys(:prefix => (prefix + underline(key) + "_"), :item => value, :item_id => item_id)
      else
        # This method is from rest_api_helper
        add_triple(@source.id, item_id, prefix + underline(key), value)
      end
    end
  end

  def underline(key)
    key.gsub('-','_')
  end
end
```

This is the complete “Show Record”, including nested attributes, such as user_id/name/url



Next

Since we now have a source adapter which loads public tweet, I will explain the code I wrote to display these tweets on iPhone simulator using Rhodes.

Filed under: Rhodes | 2 Comments
Tags: tutorial

Hello world!

07Feb09

This is my first post at Ruby On Mobile.

I do currently work as Ruby On Rails Web developer, but got interested in mobile app development, even though I don’t own any smartphones, such as iPhone, T-mobile G1 nor Blackberry.

Since I don’t know anything about mobile development nor any of mobile app development languages such as ObjectiveC(iPhone), Java(Android, Blackberry), nor C++ (Symbian), first thing I did was to search if

there were any way I could write mobile app using web technologies, like AdobeAir does for desktop app.

- To WebKit or not to WebKit within your iPhone app?
- JavaScript iPhone Apps

Embedding WebKit web browser into your mobile app locally seems interesting way to achieve mobile app development, but it seems not matured yet.

Then, I found this article.

- Rhodes Brings Ruby Apps to iPhone, Windows Mobile, BlackBerry

Not only being able to use HTML/CSS/Javascript, you can use Ruby to write mobile app. That sounds great!! The Framework (Rhodes and Rhosync) are inspired by Ruby On Rails, so they use same MVC , ORM, and so on.

As soon as I read the article, I started following tutorial provided from Rhomobile, the company behind Rhodes and Rhomobile.

At this blog, I will write about my experience playing with Rhodes/Rhosync.

Since I named this blog as “Ruby On Mobile”, I will write about other Ruby related mobile projects and libraries as I find. I am also thinking about learning basic of Android, iPhone and Mobile Website development, so that I can compare which approach suits developing what type of mobile services.

Makoto

Filed under: Uncategorized | 1 Comment

Recent Entries

- Making Twitter mobile client with Ruby using Rhodes Rhosync (Part 3)
- Aplix: Converting NTT Docomo i-appli to Windows Mobile Android, iPhone, S60 and even to portable game machine
- Making Twitter mobile client with Ruby using Rhodes & Rhosync (Part 2)
- Making Twitter mobile client with Ruby using Rhodes & Rhosync (Part 1)
- Hello world!

Categories

- Rhodes (3)
- Uncategorized (2)

Archives

- February 2009
