Ruby on Rails with AJAX

Piers Harding
Business Card
Posted on May. 18, 2006 09:23 AM in SAP NetWeaver Platform, Scripting Languages
Permalink

Since I wrote an article about integration of SAP and Ruby on Rails, it has been great to see the beginnings of a kernel of interest. As a result I decide package up the sap4rails code and distribute it properly on RAA.

Rails and AJAX

One of the interesting things that Ruby on Rails provides is built AJAX functionality by virtue of an API over prototype, and Scriptalicious. In this blog, I would like to show how neatly this integration is implemented in RubyOnRails, using a simple example of Locking, and Unlocking SAP R/3 user account

Installation Requirements

For this example you will need to do your own install of Ruby On Rails. Use the instructions for installing Ruby, Ruby On Rails, the RFCSDK, and saprfc the RadRails blog.
Just be sure to install versions Ruby 1.8.4+ and Rails 1.1.2+.
Once you have got this far, the last thing to install is sap4rails. You can either install the source package or download the gem, and install this with:

```
gem install sap4rails-<version>.gem
```

.

UserAdmin

For this example, most of the standard BAPIs are adequate. We need to be able to list users, with their details, including their lock state. We also need be able to lock and unlock them.
The general functionality of the application is to create two lists of users on a page - locked and unlocked - and for you to be able to drag a user from one to the other to change their lock state in SAP.
The following RFCs are used:

- BAPI_USER_GETLIST - list users, and their address details
- BAPI_USER_LOCK
- BAPI_USER_UNLOCK

BAPI_USER_GETLIST is not quite enough. This, I have had to wrap in another function module and also modify the results table to include the lock sta information of users.
Create a new function module called Z_BAPI_USER_GETLIST, and make sure that you activate it for RFC on the attributes tab (in SE37) (code).

Create a new structure called ZBAPIUSNAME, and include the two structures BAPIUSNAME, and USLOCK like this  .
Make sure that you activate the structure.

The code and interface needs to be completed like this:

```
FUNCTION Z_BAPI_USER_GETLIST.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(MAX_ROWS) TYPE  BAPIUSMISC-BAPIMAXROW DEFAULT 0
*"     VALUE(WITH_USERNAME) TYPE  BAPIUSMISC-WITH_NAME DEFAULT SPACE
*"  EXPORTING
*"     VALUE(ROWS) TYPE  BAPIUSMISC-BAPIROWS
*"  TABLES
*"      SELECTION_RANGE STRUCTURE  BAPIUSSRGE OPTIONAL
*"      SELECTION_EXP STRUCTURE  BAPIUSSEXP OPTIONAL
*"      USERLISTLOCK STRUCTURE  ZBAPIUSNAME OPTIONAL
*"      RETURN STRUCTURE  BAPIRET2 OPTIONAL
*"----------------------------------------------------------------------
*
data:
  LOCKSTATE LIKE  USLOCK,
  userlist like bapiusname occurs 0 with header line.

  refresh userlistlock.


  CALL FUNCTION 'BAPI_USER_GETLIST'
    EXPORTING
```

```
        MAX_ROWS              = 0
        WITH_USERNAME         = with_username
      IMPORTING
        ROWS                  = rows
      TABLES
        SELECTION_RANGE       = selection_range
        SELECTION_EXP         = selection_exp
        USERLIST              = userlist
        RETURN                = return
              .

    loop at userlist.
      move-corresponding: userlist to userlistlock.
      if userlistlock-firstname = space.
         userlistlock-firstname = userlistlock-username.
      endif.

      CALL FUNCTION 'SUSR_USER_LOCKSTATE_GET'
        EXPORTING
          USER_NAME               = userlist-username
        IMPORTING
          LOCKSTATE               = lockstate
        EXCEPTIONS
          USER_NAME_NOT_EXIST     = 1
          OTHERS                  = 2
                .

      move-corresponding: lockstate to userlistlock.
      append userlistlock.

    endloop.

  ENDFUNCTION.
```

Activate the function module and test it.

The Rails part

The full application can be downloaded from here - but what I'd like to do is quickly describe the meat of what had to be done to get this type of application working.

Config - sap.yml

As described in the RadRails blog, you need to adjust the configuration in config/sap.yml to point to your SAP system:

```
development:
  ashost: 10.1.1.1
  sysnr: "00"
  client: "010"
  user: developer
  passwd: developer
  lang: EN
  trace: "1"
...
```

Model - sap_user.rb

The SapUser object now inherits from the new SAP4Rails::Base class. This serves to automatically take care of managing RFC connections based on th config done above. The two main class methods for use are function_module, which allows you to declare what RFCs you want to use, and parameter which is a helper method for declaring attributes of a SapUser (or any other Model object).
In the interests of simplifying the application, by reducing the amount of ABAP code to be written, and the number of RFC calls to be made, I have in a way "cheated", with the arrangement of methods defined in SapUser. Instead of having a SapUser#find method, I rely on the use of SapUser#find_all and SapUser#find_cache to reduce a series of SapUser searches down to one RFC call only. In reality this is probably not good practice, but it suits for this example.

Read the code comments below for further details:

```
require_gem "sap4rails"

class SapUser < SAP4Rails::Base
```

```
# You must define a list of RFCs to preload
  function_module :Z_BAPI_USER_GETLIST,
                        :BAPI_USER_LOCK,
                        :BAPI_USER_UNLOCK

# You must define a list of attribute accessors to preload
  parameter :last, :first, :userid, :locked

# do your attribute initialisation for each SapUser instance
        def initialize(last, first, userid, locked)
          @last = last
          @first = first
          @userid = userid
          @locked = locked
          @changed = false
        end

# what is the lock state
        def locked?
          return self.locked ? true : false
        end

# on #save - flip the lock state of the SapUser, calling the
# appropriate RFC to do it
  def save()
    RAILS_DEFAULT_LOGGER.warn("[SapUser]#save what did we get: " + self.inspect)

    if self.locked?
      SapUser.BAPI_USER_LOCK.reset()
      SapUser.BAPI_USER_LOCK.username.value = self.userid
      SapUser.BAPI_USER_LOCK.call()
    else
      SapUser.BAPI_USER_UNLOCK.reset()
      SapUser.BAPI_USER_UNLOCK.username.value = self.userid
      SapUser.BAPI_USER_UNLOCK.call()
    end

    # just so something happens ...
    return true
  end

# one RFC call to get them all
  def self.find_all
    RAILS_DEFAULT_LOGGER.warn("[SapUser]#find_all ")
    SapUser.Z_BAPI_USER_GETLIST.reset()
    SapUser.Z_BAPI_USER_GETLIST.with_username.value = 'X'
    SapUser.Z_BAPI_USER_GETLIST.call()
    users = []
    SapUser.Z_BAPI_USER_GETLIST.userlistlock.rows().each {|row|
                next if row['FIRSTNAME'].strip.length == 0
                state = nil
                if row['WRNG_LOGON'] == "L" ||
                   row['LOCAL_LOCK'] == "L" ||
                   row['GLOB_LOCK'] == "L"
                  state = true
                else
                  state = false
                end
                users.push(SapUser.new(row['LASTNAME'],
                                       row['FIRSTNAME'],
                                       row['USERNAME'],
                                       state))
              }
    return users
  end

# find a user base on the results of a SapUser#find_all
  def self.find_cache(user, cache)
    RAILS_DEFAULT_LOGGER.warn("[SapUser]#find_cache: #{user} ")
    cache.each{|row|
```

```
            return row if user.strip == row.userid.strip
        }
    end

    # get a list of all the locked users
    def self.find_locked
        RAILS_DEFAULT_LOGGER.warn("[SapUser]#find_locked ")
        locked = []
        find_all().each{|user|
            locked.push(user) if user.locked
        }
        return locked
    end

    # get a list of all the unlocked users
    def self.find_unlocked
        RAILS_DEFAULT_LOGGER.warn("[SapUser]#find_unlocked ")
        unlocked = []
        find_all().each{|user|
            unlocked.push(user) unless user.locked
        }
        return unlocked
    end
end
```

Controller - lock_controller.rb

There are only 3 basic actions to the only controller in this application. The initial list action, build the starting page presenting the two lists of users (locked and unlocked). From there, as a result of the AJAX enabled calls from the dragndrop feature, two further actions are called - set_locked and set_unlocked.

```
class LockController < ApplicationController

    # gnerate the starting user lists, and hand off to the default list view
    def list
        RAILS_DEFAULT_LOGGER.warn("[LIST] Parameters: " + @params.inspect)
        @locked_users = SapUser.find_locked()
        @unlocked_users = SapUser.find_unlocked()
        RAILS_DEFAULT_LOGGER.warn("[LIST] of Locked: " + @locked_users.inspect)
        RAILS_DEFAULT_LOGGER.warn("[LIST] of UNLocked: " + @unlocked_users.inspect)
    end

    # check through the list of locked users in the locked sortable_element box
    # and set their locked state in necessary
    # on completion - render the partial locked_users
    def set_locked
        RAILS_DEFAULT_LOGGER.warn("[SET_LOCKED] Parameters: " + @params.inspect)
        @locked_users = []
        cache = SapUser.find_all()
        if @params['locked_box']
          @params['locked_box'].each {|locked|
              next if locked.length == 0
              user = SapUser.find_cache(locked, cache)
              next if user.first.strip.length == 0
              if user && ! user.locked?
                user.locked = !user.locked?
                user.save
              end
              @locked_users.push(user)
          }
        end
        render :partial => 'locked_users', :object => locked_users
    end

    # exact opposite/the same as set_locked
    def set_unlocked
        RAILS_DEFAULT_LOGGER.warn("[SET_UNLOCKED] Parameters: " + @params.inspect)
        @unlocked_users = []
        cache = SapUser.find_all()
        if @params['unlocked_box']
```

```
      @params['unlocked_box'].each {|unlocked|
          next unless unlocked.length > 0
          user = SapUser.find_cache(unlocked, cache)
          next if user.first.strip.length == 0
          if user && user.locked?
            user.locked = !user.locked?
            user.save
          end
          @unlocked_users.push(user)
      }
    end
    render :partial => 'unlocked_users', :object => unlocked_users
  end

# called by the rendering action of the partial locked_users
  def locked_users
    RAILS_DEFAULT_LOGGER.warn("[LOCKED_USERS] of Locked: " + @locked_users.inspect)
    @locked_users
  end

# called by the rendering action of the partial unlocked_users
  def unlocked_users
    RAILS_DEFAULT_LOGGER.warn("[UNLOCKED_USERS] of UNLocked: " + @unlocked_users.inspect)
    @unlocked_users
  end

end
```

Views

The the overall page template (layout) defines the shape of the page, and what JavaScript libraries are pulled in for the effects (AJAX). All pages inher from this.

layout/application.rhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <title>User Administration:  <%= controller.controller_name %></title>
    <meta http-equiv="imagetoolbar" content="no" />
    <%= stylesheet_link_tag "administration.css" %>
    <%= javascript_include_tag "prototype", "effects", "dragdrop",
                      "controls" %>
  </head>

  <body>
  <div id="container">
    <div id="header">
    <div id="info">
      <%= link_to "home", :controller=> "/lock", :action => 'list' %>
    </div>
      <h1><%= link_to "UserAdmin - #{controller.controller_name}", :controller => "/lock"
        </h1>
    </div>
    <div id="content">
      <h2><%= @page_heading %></h2>
      <%= @content_for_layout %>
    </div>
  </div>
  </body>
</html>
```

lock/list.rhtml

The two most important things in list are the two container div tags - unlocked_box and locked_box. These in turn, have a corresponding partial (unlocked_users, and locked_users), that are responsible for generating the dragable user items.

```
<% @heading = "User Admin - Lock/UnLock" %>

  <div id="user-admin">
```

```
    <div id="unlocked" class="dropbox">
      <h3>Unlocked Users</h3>
      <div id="unlocked_box">
        <%= render :partial => 'unlocked_users', :object => @unlocked_users %>
      </div>
    </div>

    <div id="cnt-locked" class="dropbox">
      <h3>Locked Users</h3>
      <div id="locked_box">
        <%= render :partial => 'locked_users', :object => @locked_users %>
      </div>
    </div>
    <br clear="all" />

  </div>
```

lock/_unlocked_users.rhtml

the partial unlocked_users either displays a place holder element if there are no users, or calls the render of unlocked_user for each user. It also uses AJAX function sortable_element which dictates what div container holds the sortable drag and drop elements, and what actions to take when an event fired with them. This is how we trigger the call to the set_unlocked or set_locked action of the list controller for updating the individual "boxes" of user

```
<% if unlocked_users.empty? %>
  <div class="target">  You have no Unlocked SAP Users.... </div>
<% else %>
  <%= render :partial => 'unlocked_user', :collection => unlocked_users %>
<% end %>

<%= sortable_element "unlocked_box",
  :update => "unlocked_box",
  :url => {:action=>'set_unlocked'},
  :tag => 'div', :handle => 'handle', :containment => ['unlocked_box','locked_box'] %>

<%= sortable_element "locked_box",
  :update => "locked_box",
  :url=> {:action=>'set_locked'},
  :tag => 'div', :handle => 'handle', :containment => ['locked_box','unlocked_box'] %>
```

lock/_unlocked_user.rhtml

the partial unlocked_user renders a dragble_element for each SapUser.

```
<div id="unlockeduser_<%= unlocked_user.userid %>" class="dragitem">
  <h4 class="handle"><%= unlocked_user.userid %></h4>
  <p><%= unlocked_user.last + ", " + unlocked_user.first %></p>
</div>
<%= draggable_element "unlockeduser_#{unlocked_user.userid}" %>
```

lock/_locked_users.rhtml

the same as for the partial unlocked_users

```
<% if locked_users.empty? %>
  <div class="target">  You have no Locked Users ...  </div>
<% else %>
 <%= render :partial => 'locked_user', :collection => locked_users %>
<% end %>

<%= sortable_element "unlocked_box",
  :update => "unlocked_box",
  :url => {:action=>'set_unlocked'},
  :tag => 'div', :handle => 'handle', :containment => ['unlocked_box','locked_box'] %>

<%= sortable_element "locked_box",
  :update => "locked_box",
  :url=> {:action=>'set_locked'},
  :tag => 'div', :handle => 'handle', :containment => ['locked_box','unlocked_box'] %>
```

lock/_locked_user.rhtml

the same as for the partial unlocked_user

```
<div id="lockeduser_<%= locked_user.userid %>" class="dragitem">
  <h4 class="handle"><%= locked_user.userid %></h4>
  <p><%= locked_user.last + ", " + locked_user.first %></p>
```

```
</div>
<%= draggable_element "lockeduser_#{locked_user.userid}" %>
```

In config/routes.rb add:

```
map.connect '', :controller => "lock", :action => 'list'
```

and make sure that you delete public/index.html

This makes sure that any requests to the root of the server eg. http://localhost:3000, are forwarded onto the list action of the lock controller.

firing Up

Start the Rails WEBrick server by running the script:

```
ruby scripts/server
```

Open your broswer and point to http://localhost:3000.

When you connect to http://localhost:3000 (there will be an inital delay as sap4rails caches the RFC calls), you should get a screen that looks like this

A Flash movie of this in action can be seen here.

Piers Harding is just another old SAP Hack

Comment on this weblog

Showing messages 1 through 8 of 8.

Titles Only | Main Topics | Oldest First

- Ruby on Rails and Unicode
  2006-06-03 16:52:22 Gregor Wolf Business Card [Reply]

  Hi Piers,

  I try to run your example aginst our Web AS 6.20 Unicode system. Here I have a problem with german umlauts öäü and ÖÄÜ. In the view you've s
  the encoding to utf-8 but the codepage for the RFC is 1100 which is ISO-8859-1 Western Europe. I tried to switch Rails to Unicode by adding:

  $KCODE = 'u'
  require 'jcode'

  to config/environment.rb as suggested in HowToUseUnicodeStrings. But this results in this error:

  Showing app/views/lock/_unlocked_user.rhtml where line #5 raised:
  malformed UTF-8 character
  Extracted source (around line #5):
  5: <%= draggable_element "unlockeduser_#{unlocked_user.userid}" %>

  I've tried to add:

  codepage: "4103"
  unicode: "1"

  to config/sap.yml but the error was the same. As I see from the trace the codepage is still 1100. I'm using saprfc-0.19 on a Debian Sarge system.

  Hope you can help me out.

  Regards
  Gregor

  - Ruby on Rails and Unicode

2006-06-03 21:55:17 Piers Harding Business Card [Reply]

Hi Gregor,

I think I've fixed this. Not all the possible RFC connection options were being passed through. Now you can pass the UNICODE related options (and everything else). Please try sap4rails 0.02, and let me know how you get on.
(http://www.piersharding.com/download/ruby/rails/)
Cheers,
Piers Harding.

- Ruby on Rails and Unicode
  2006-06-05 11:01:27 Gregor Wolf Business Card [Reply]

  Hi Piers,

  wow that was rearly quick. Unfortenately I wasn't so quick to test because of the weekend and holliday in germany. So I've just tested and had a bad result. When I set:

  codepage: "4103"
  unicode: "1"

  in config/sap.yml and access the application website after starting script/server this error message is returned:

  RFC Call/Exception: Connection Failed Error group: 104 Key: C

  in the trace file I see:

  .....C.A.L.L._.F.U.N.C.T.I.O.N._.N.O.T._.F.O.U.N.D.....@
  F.u.n.c.t.i.o.n..m.o.d.u.l.e..".RFCPIN"..n.o.t.f.o.u.n.d....

  I can send you the logs if you need further information.

  Regards
  Gregor

  - Ruby on Rails and Unicode
    2006-06-06 03:55:08 Piers Harding Business Card [Reply]

    Hi Gregor -
    is it possible for you to separate out a test case to run outside of Rails - ie. take the code out and run it in a separate Ruby script so th we can remove Rails from the equassion - I would test this myself but I do not have access to a UNICODE system to do it. Also - it wo probably be a good idea to move this to a Forum discussion under Scripting Languages - I watch all the threads, so I'll know when you post. Please add in the traces too.
    Thanks,
    Piers Harding.

    - Ruby on Rails and Unicode
      2006-06-06 16:01:05 Gregor Wolf Business Card [Reply]

      Hi Piers,

      I've now posted a Topic:
      Ruby and Unicode

      Regards
      Gregor

- Z_SUSR_USER_LOCKSTATE_GET
  2006-05-18 12:13:20 Gregor Wolf Business Card [Reply]

  Hi Piers,

  could you provide the source of function module Z_SUSR_USER_LOCKSTATE_GET which is used in Z_BAPI_USER_GETLIST.

  Regards
  Gregor

  - Z_SUSR_USER_LOCKSTATE_GET
    2006-05-18 13:27:02 Piers Harding Business Card [Reply]

    Hi Gregor - good to hear from you. I forgot about that as the Z version was in relation to some early work I was doing. I've changed the example, and the source code file to revert to SUSR_USER_LOCKSTATE_GET. The Z version was only so I could make the SAP standard RFC enabled - there is no functionality difference.
    Cheers,
    Piers Harding.

  - Z_SUSR_USER_LOCKSTATE_GET
    2006-05-18 12:20:08 Gregor Wolf Business Card [Reply]

    Hi Piers,

    found a solution on my own. On my IDES ERP 2004 I've found the standard function module SUSR_USER_LOCKSTATE_GET.

    Regards
    Gregor

Showing messages 1 through 8 of 8.