Mod_xml_curl_Ramaze/Sequel_groups_example

```
 This is the portion of TinyQueue (http://code.rubyists.com/projects/tq)
```

Which returns the dial groups. This is written using Sequel for the database layer and Ramaze for the web framework. First, the configuration, in /usr/local/freeswitch/conf/autoload_confs/xml_curl.conf.xml

```
<configuration name="xml_curl.conf" description="cURL XML Gateway">
  <bindings>
    <binding name="dynamic_group_conf">
      <param name="gateway-url" value="http://127.0.0.1:8080/directory.xml" bindings="directory"/
    </binding>
  </bindings>
</configuration>
```

And in /usr/local/freeswitch/conf/dialplan/default/04_group_dial.xml

```
<include>
   <extension name="Group IT">
       <condition field="destination_number" expression="9988">
         <action application="set" data="ringback=${texas-ring}"/>
         <action application="set" data="hangup_after_bridge=true"/>
         <action application="set" data="continue_on_fail=true"/>
         <action application="set" data="call_timeout=10"/>
         <action application="bridge" data="group/it@$${domain}+F"/>
         <action application="transfer" data="9988 XML default" />
         <action application="hangup" />
       </condition>
     </extension>
</include>
```

Now for the Sequel models, migration for agents: 001_agents_table.rb

```
require "sequel"
class AgentsTable < Sequel::Migration
  def up
    create_table :agents do |t|
      primary_key :id
      varchar :first_name, :null => false
      varchar :last_name, :null => false
      varchar :extension, :null => false
    end
  end

  def down
    drop_table :agents if DB.table_exists?(:agents)
  end
end
```

Our table to hold queues, which have agents: 002_tiny_queues_table.rb

```
class TinyQueuesTable < Sequel::Migration
  def up
    create_table :tiny_queues do |t|
      primary_key :id
      varchar :name, :null => false
      text :description
    end
  end

  def down
```

```
    drop_table :tiny_queues if DB.table_exists?(:tiny_queues)
  end
end
```

And the join table for these: 003_agents_tiny_queues_table.rb

```
class AgentsTinyQueuesTable < Sequel::Migration
  def up
    create_table :agents_tiny_queues do |t|
      primary_key :id
      foreign_key :tiny_queue_id, :references => :tiny_queues, :null => false
      foreign_key :agent_id, :references => :agents, :null => false
      varchar :status, :null => false, :default => "Idle"
      timestamp :status_stamp, :default => "now()"
      timestamp :last_call_stamp
      timestamp :login_stamp, :null => false, :default => "now()"
      boolean :login, :null => false, :default => false
    end
  end

  def down
    drop_table :agents_tiny_queues if DB.table_exists?(:agents_tiny_queues)
  end
end
```

Load those with sequel -m (see docs for full command line) The models which correspond with the above tables follow agent.rb

```
class Agent < Sequel::Model
  one_to_many :agents_tiny_queues
  many_to_many :tiny_queues, :join_table => :agents_tiny_queues

  def logged_in?
    self.agents_tiny_queues.first.login
  end

  def login
    now = Time.now
    self.agents_tiny_queues.each do |agent|
      agent.update_with_params(:status => "Ready", :status_stamp => now, :login => true, :login_s
    end
  end

  def logoff
    now = Time.now
    self.agents_tiny_queues.each do |agent|
      agent.update_with_params(:status => "Offline", :status_stamp => now, :login => false)
    end
  end

  def fullname
    "%s %s" % [first_name, last_name]
  end

end
```

agents_tiny_queue.rb

```
class AgentsTinyQueue < Sequel::Model
```

```
  many_to_one :tiny_queue
  many_to_one :agent
end
```

and tiny_queue.rb

```
class TinyQueue < Sequel::Model
  many_to_many :agents, :join_table => :agents_tiny_queues
  one_to_many :unavailable_agents, :class => "AgentsTinyQueue" do |ds|
    ds.filter(:login => false).order(:last_call_stamp.desc, :login_stamp.asc)
  end
  one_to_many :consumers, :class => "AgentsTinyQueue" do |ds|
    ds.filter(:login => true).order(:last_call_stamp.asc, :login_stamp.asc)
  end

end
```

The Ramaze controller - controller/main.rb

```
# Default url mappings are:
#  a controller called Main is mapped on the root of the site: /
#  a controller called Something is mapped on: /something
# If you want to override this, add a line like this inside the class
#  map '/otherurl'
# this will force the controller to be mounted on: /otherurl

Ramaze::Route["/directory.xml"] = "/directory"
class MainController < Controller
  # the index action is called automatically when no other action is specified
#  scaffold_all_models :except => ['init']

  def index
  end

  def directory
    if request[:section] == "directory" and request[:group]
      @queues = TinyQueue.all
      if this_queue = @queues.detect { |n| n.name.downcase == request[:group_name] } # request[:g
        Ramaze::Log.info "Sending Group.conf, requested #{request[:group_name]} matched #{this_qu
        @queues = TinyQueue.all
        response['Content-Type'] = 'text/xml'
        output = render_template("groups.rhtml")
        got_called = this_queue.consumers.first
        got_called.update_with_params(:last_call_stamp => Time.now)
        output
      else
        not_found # MUST return not_found or fs gets an error parsing the empty response as xml
      end
    else
      not_found # If it's not a directory request, return not_found so the static files get looke
    end
  end

  # the string returned at the end of the function is used as the html body
  # if there is no template for the action. if there is a template, the string
  # is silently ignored
  def notemplate
    "there is no 'notemplate.xhtml' associated with this action"
  end
```

```
  def not_found
    respond '<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <include>
      <document type="freeswitch/xml">
        <section name="result">
          <result status="not found" />
        </section>
      </document>
    </include>', 200, "Content-Type" => "text/xml"
  end
end
```

And the view: view/groups.rhtml (Erubis templating)

```
<document type="freeswitch/xml">
  <section name="directory" description="group test">
    <domain name="$${domain}">
      <groups>
        <% @queues.each do |queue| %>
        <group name="#{queue.name}">
          <users>
          <% queue.consumers.each do |consumer| %>
            <% agent = consumer.agent %>
            <% if agent.extension.to_s.match(/^sofia/) %>
            <user id="#{agent.fullname.to_s.gsub(/\s/,'_')}">
              <params>
                <param name="dial-string" value="#{agent.extension}"/>
              </params>
              <variables>
                <variable name="user_context" value="default"/>
              </variables>
            </user>
            <% else %>
            <user id="#{agent.fullname.to_s.gsub(/\s/,'_')}">
              <params>
                <param name="dial-string" value="user/#{agent.extension}@$${domain}"/>
              </params>
              <variables>
                <variable name="user_context" value="default"/>
              </variables>
            </user>
            <% end %>
          <% end %>
          </users>
        </group>
        <% end %>
      </groups>
    </domain>
  </section>
</document>
```