```
class ApplicationController < ActionController::Base
  include AuthenticatedSystem

  if LOG_DETAIL
    include Oink::MemoryUsageLogger
    include Oink::InstanceTypeCounter
  end

  helper :all # include all helpers, all the time

  prepend_before_filter :login_required

  protect_from_forgery

  # purge "password" from logs
  filter_parameter_logging :password

  # rescue errors and return HTTP status codes
  rescue_from 'PermissionError' do |e| http_status_code(:forbidden, e) end
  rescue_from 'ArgumentError' do |e| http_status_code(:expectation_failed, e)
end
  rescue_from ActiveRecord::RecordNotFound do |e| http_status_code(:bad_request,
e) end
  rescue_from ActiveRecord::RecordInvalid do |e| http_status_code(:bad_request,
e) end
  rescue_from LibXML::XML::Error do |e| http_status_code(:bad_request, e) end
  rescue_from REXML::ParseException do |e| http_status_code(:bad_request, e) end

  if ENV['RAILS_ENV'] == 'production'
    rescue_from Exception do |e| notify_and_handle(e) end
  end

  def http_status_code(status, exception)
    respond_to do |format|
      format.any { head status }
    end
  end

  def force_xml
    params.merge!(:format => :xml) unless params.include?(:format)
  end

  def admin_required
    redirect_back_or_default('/') unless current_user && current_user.admin?
  end

  private
    def notify_and_handle(e)
      @requested_page= e.to_s
      if e.class == ActionController::RoutingError || e.class ==
ActionController::UnknownAction
        @error = "File or location does not exist."
        render :template => '/home/error'
      else
        SystemNotifier.deliver_exception(self, request, current_visit, e)
```

```
        flash.now[:error] = "We encountered an internal error. We have been
notified and are working to fix the problem."
        render :template => '/home/index'
      end
    end
end
class HomeController < ApplicationController
  skip_before_filter :login_required, :only => [:learn, :api, :about]

  def index
  end

  def api
  end

  def about
  end

  def learn
  end

  # Delete all data associated with the logged in user.
  def delete
    if request.post?
      current_user.destroy_data
      flash[:notice] = 'All data destroyed.'
    end
    redirect_to root_path
  end

  # Delete all items associated with the logged in user.
  def delete_items
    if request.post?
      current_user.destroy_items
      flash[:notice] = 'All item data destroyed.'
    end
    redirect_to root_path
  end
endclass ItemsController < ApplicationController
  include Algorithms::Rank::Elo

  before_filter :force_xml

  # GET /items/1
  # ==== Return
  # Item by id.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of item.
  # ==== Raises
  # PermissionError:: If item does not belong to user.
  def show
    @item = Item.find(params[:id], :conditions => { :user_id => current_user.id
}, :include => [ :questions ])
  end
```

```
  # POST /items/add
  # ==== Return
  # Added Item.
  # ==== Options (params)
  # active<String>:: Converted to boolean. If not nil item is activated.
  # tracking<String>:: String of data to be stored with item.
  # ==== Post
  # Formatted XML of item to add and question to add item to.
  # ==== Raises
  # PermissionError:: If any questions do not belong to user.
  def add
    return unless request.post?
    xml = LibXML::XML::Parser.parse(request.raw_post)
    active = !params[:active].nil?
    @items = xml.find("/items/item").inject([]) do |items, item|
      questions = item.find("questions/question").inject([]) do |arr, question|
        arr << current_user.questions.find(question.attributes["id"])
      end
      attributes = {:user_id => current_user.id, :data =>
item.find("data").first.content, :active => active}
      attributes.merge!(:tracking => params[:tracking]) if params[:tracking]
      current_item = Item.create(attributes)
      current_item.questions << questions
      items << current_item
    end
  end

  # POST /items/delete
  # ==== Return
  # ID of item deleted and whether or not item was succesfully deleted.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of item.
  # ==== Raises
  # PermissionError:: If item does not belong to user.
  def delete
    return unless request.post?
    item = Item.find(@id = params[:id], :conditions => { :user_id =>
current_user.id })
    item.destroy
    @success = !Item.exists?(@id)
  end

  # GET /items/list
  # ==== Return
  # Array of items.
  # ==== Options (params)
  # limit<String>:: Converted to integer. Number of items to return.
  # offset<String>:: Converted to integer. Item to begin returning with.
  # order<String>:: Order to return items. If ASC items returned in ascending
  # order, otherwise items returned in descending order.
  # question_id<String>:: Converted to integer. Question to return items for.
  # data<String>:: Converted to boolean. If exists return data for items.
  # rank_algorithm<String>:: Name or ID of rank algorithm.  Default order is
  # by created at date.
  # ==== Raises
```

```
  # PermissionError:: If question does not belong to user.
  def list
    limit = params[:limit].to_i
    offset = params[:offset].to_i
    order = (params[:order] && params[:order].downcase == 'asc') ? 'ASC' :
'DESC'
    question_id = params[:question_id].to_i
    options = {
      :conditions => { :user_id => current_user.id },
      :include => [:items_questions]
    }
    options[:limit] = limit if limit > 0
    options[:offset] = offset if offset > 0
    if question_id > 0
      raise PermissionError unless
current_user.question_ids.include?(question_id)
      options[:joins] = "INNER JOIN items_questions ON
items.id=items_questions.item_id AND items_questions.question_id=#{question_id}"
    end
    @data = !params[:data].nil?
    rank_algo = params[:rank_algorithm]
    rank_algo_id = rank_algo.to_i
    if rank_algo_id == 0 && !(rank_algo.nil? || rank_algo.blank?)
      rank_algo = RankAlgorithm.first(:conditions => { :name => rank_algo })
      rank_algo_id = rank_algo.id if rank_algo
    end
    case rank_algo_id
    when 1
      options[:order] = 'items_questions.position'
      @score = lambda { |i| i.items_questions.first.position }
    when 2
      options[:order] = 'items_questions.wins / (items_questions.wins +
items_questions.losses)'
    when 3
      # calculate expected winning percentage as (1/n)SUM_j(||i>j||/||i,j||)
      options.delete(:limit)
      @items = Item.all(options)
#      prompt_options = {
#        :include => [:items, { :vote => :items }],
#        :conditions => "votes.id IS NOT NULL AND
items.user_id=#{current_user.id}"
#      }
#      prompt_options[:conditions] += " AND question_id=#{question_id}" if
question_id > 0
#      prompts = Prompt.all(prompt_options)
      stat_options = { :select => "stats.votes, items_stats2.item_id" }
      stat_options[:conditions] = { :question_id => question_id } if question_id
> 0
      prompt_options = {
        :select => "COUNT(items_prompts.item_id) AS wins,
items_prompts.item_id",
        :group => "items_prompts.item_id"
      }
      for item in @items
```

```
        i_cmp = Stat.all(stat_options.merge(:joins => "INNER JOIN items_stats ON
(items_stats.stat_id=stats.id
          AND items_stats.item_id=#{item.id}) INNER JOIN items_stats AS
items_stats2
          ON (items_stats2.stat_id=stats.id AND
items_stats2.item_id!=#{item.id})"
        )).inject({}) do |hash, stat|
          hash[stat.item_id.to_i] = stat.votes.to_f
          hash
        end
        i_wins = Prompt.all(prompt_options.merge(:joins => "INNER JOIN
items_prompts ON (items_prompts.prompt_id=prompts.id
          AND items_prompts.item_id!=#{item.id}) INNER JOIN votes ON
          (votes.prompt_id=prompts.id) INNER JOIN items_votes ON
(items_votes.vote_id=votes.id
          AND items_votes.item_id=#{item.id})"
        )).inject({}) do |hash, prompt|
          hash[prompt.item_id.to_i] = prompt.wins.to_i
          hash
        end
        cmp = 0
        item.score = (@items - [item]).inject(0) do |sum, j_item|
          i_j_cmp = i_cmp[j_item.id]
          if i_j_cmp && i_j_cmp > 0
            cmp += 1
            wins = i_wins[j_item.id]
            wins ? sum + wins / i_j_cmp : sum
          else
            sum
          end
        end
        item.score = (100 * (item.score / cmp)).round if cmp > 0
      end
      @items = @items.sort_by { |el| -el.score }
      @items = @items.first(limit) if limit > 0
      @score = lambda do |i|
        i.score
      end
    else
      options[:order] = 'items.created_at'
    end
    unless @score
      @score = lambda do |i|
        iq = i.items_questions.first
        total = iq.wins + iq.losses
        total.zero? ? 0 : (100 * (iq.wins.to_f/(total))).round
      end
    end
    unless @items
      options[:order] += " #{order}"
      @items = Item.all(options)
    end
  end

  # GET /items/suspend/1
```

```
  # ==== Return
  # Item in suspended state.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of item.
  # ==== Raises
  # PermissionError:: If item does not belong to user.
  def suspend
    update_item_state(params[:id], 0)
  end

  # GET /items/activate/1
  # ==== Return
  # Item in active state.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of item.
  # ==== Raises
  # PermissionError:: If item does not belong to user.
  def activate
    update_item_state(params[:id], 1)
  end

  private
    def update_item_state(id, state)
      @item = Item.find(id, :conditions => { :user_id => current_user.id })
      @item.update_attribute(:active, state)
      render :action => 'show'
      return
    end
endclass PromptAlgorithmsController < ApplicationController
  before_filter :force_xml, :only => :list
  before_filter :admin_required, :except => :list

  # GET /prompt_algorithms
  # GET /prompt_algorithms.xml
  def index
    @prompt_algorithms = PromptAlgorithm.find(:all)

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @prompt_algorithms }
    end
  end

  # GET /prompt_algorithms/1
  # GET /prompt_algorithms/1.xml
  def show
    @prompt_algorithm = PromptAlgorithm.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @prompt_algorithm }
    end
  end

  # GET /prompt_algorithms/new
```

```
  # GET /prompt_algorithms/new.xml
  def new
    @prompt_algorithm = PromptAlgorithm.new

    respond_to do |format|
      format.html # new.html.erb
      format.xml  { render :xml => @prompt_algorithm }
    end
  end

  # GET /prompt_algorithms/1/edit
  def edit
    @prompt_algorithm = PromptAlgorithm.find(params[:id])
  end

  # POST /prompt_algorithms
  # POST /prompt_algorithms.xml
  def create
    @prompt_algorithm = PromptAlgorithm.new(params[:prompt_algorithm])

    respond_to do |format|
      if @prompt_algorithm.save
        flash[:notice] = 'PromptAlgorithm was successfully created.'
        format.html { redirect_to(@prompt_algorithm) }
        format.xml  { render :xml => @prompt_algorithm, :status => :created,
:location => @prompt_algorithm }
      else
        format.html { render :action => "new" }
        format.xml  { render :xml => @prompt_algorithm.errors, :status =>
:unprocessable_entity }
      end
    end
  end

  # PUT /prompt_algorithms/1
  # PUT /prompt_algorithms/1.xml
  def update
    @prompt_algorithm = PromptAlgorithm.find(params[:id])

    respond_to do |format|
      if @prompt_algorithm.update_attributes(params[:prompt_algorithm])
        flash[:notice] = 'PromptAlgorithm was successfully updated.'
        format.html { redirect_to(@prompt_algorithm) }
        format.xml  { head :ok }
      else
        format.html { render :action => "edit" }
        format.xml  { render :xml => @prompt_algorithm.errors, :status =>
:unprocessable_entity }
      end
    end
  end

  # DELETE /prompt_algorithms/1
  # DELETE /prompt_algorithms/1.xml
  def destroy
```

```
    @prompt_algorithm = PromptAlgorithm.find(params[:id])
    @prompt_algorithm.destroy

    respond_to do |format|
      format.html { redirect_to(prompt_algorithms_url) }
      format.xml  { head :ok }
    end
  end

  # GET /prompt_algorithms/list
  def list
    @algorithms = PromptAlgorithm.all
  end
endclass PromptsController < ApplicationController
  before_filter :force_xml

  # GET /prompts/1
  # ==== Return
  # Prompt.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of prompt.
  # ==== Raises
  # PermissionError:: If prompt does not belong to user.
  def show
    @prompt = Prompt.find(params[:id])
    raise PermissionError unless
current_user.question_ids.include?(@prompt.question_id)
  end

  # GET /prompts/view/1
  # ==== Return
  # Nothing.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of prompt to register view for.
  # ==== Raises
  # PermissionError:: If prompt does not belong to user.
  def view
    prompt = Prompt.find(params[:id])
    raise PermissionError unless
current_user.question_ids.include?(prompt.question_id)
    Stat.view(prompt.question_id, prompt.items)
    head :ok
  end

  # GET /prompts/list
  # ==== Return
  # Array of length n. Prompts matching parameters
  # ==== Options (params)
  # question_id<String>:: Converted to integer. Must be greater than 0 and
  # belong to the current user.  Must belong to user.
  # item_ids<String>:: Comma seperated list of items to include. May only
  # include commas and digits.  Must belong to user.  Optional value.
  # data<String>:: Flag for whether to include item data.  Data included
  # if value is not nil.
  # ==== Raises
```

```
  # PermissionError:: If question or any item doesn't belong to current user.
  def list
    question_id = params[:question_id].to_i
    item_ids = params[:item_id]
    @data = !params[:data].nil?
    item_ids = valid_item_ids(item_ids)
    options =  { :include => :items, :conditions => {} }
    if question_id > 0
      options[:conditions].merge!('prompts.question_id' => question_id)
      raise PermissionError unless
current_user.question_ids.include?(question_id)
      unless item_ids.empty?
        raise PermissionError unless (item_ids - current_user.item_ids).empty?
        options[:conditions].merge!({ 'items.id' => item_ids })
      end
      @prompts = Prompt.all(options)
    else
      @prompts = []
    end
  end

  # GET /prompts/create
  # ==== Return
  # Array of length n. Prompts created for the question and voter.
  # ==== Options (params)
  # question_id<String>:: Converted to integer. Must be greater than 0 and
belong to the current user.
  # voter_id<String>:: Converted to integer. Must be 0 or belong to the current
user.
  # n<String>:: Converted to integer. Set to 1 if less than 1. Set to
MAX_BATCH_PROMPTS if greater
  # than MAX_BATCH_PROMPTS.
  # item_id<String>:: Nil or comma seperated list of items to include in the
prompt.
  # be returned.
  # ==== Raises
  # ArgumentError:: If question less than 1 or doesnt belong to current user or
if voter greater than 0
  # and doesn't belong to current user.
  def create
    @question_id = params[:question_id].to_i
    voter_id = params[:voter_id].to_i
    num = params[:n].to_i
    prime = params[:prime].to_i > 0
    @data = !params[:data].nil?
    if num < 1
      num = 1
    elsif num > Constants::MAX_BATCH_PROMPTS
      num = Constants::MAX_BATCH_PROMPTS
    end
    raise ArgumentError unless current_user.questions.exists?(@question_id) &&
(voter_id < 1 || current_user.voters.exists?(voter_id))
    @prompt_item_ids = Prompt.fetch(@question_id, voter_id, num, prime)
    @algorithm_id = prime ? 3 : 2
    raise ArgumentError if @prompt_item_ids.keys.empty?
```

```
  end

  private
    # If non "," or digit is passed as an "item_id" param all item params are
ignored
    def valid_item_ids(item_ids)
      (item_ids && !item_ids.empty? && item_ids.gsub(/,|\d/, '').empty?) ?
item_ids.split(',').map(&:to_i) : []
    end
endclass QuestionsController < ApplicationController
  before_filter :force_xml

  # GET /questions/1
  # ==== Return
  # Question and stats.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of question.
  # ==== Raises
  # PermissionError:: If question does not belong to user.
  def show
    @question = Question.first(:conditions => { :id => params[:id], :user_id =>
current_user.id })
    if @question
      @items_count = Item.count(
        :joins => "INNER JOIN items_questions ON
(items_questions.item_id=items.id AND
items_questions.question_id=#{@question.id})",
        :conditions => { :active => true }
      )
      @all_items_count = Item.count(:joins => "INNER JOIN items_questions ON
(items_questions.item_id=items.id AND
items_questions.question_id=#{@question.id})")
      @votes_count = Prompt.count(:joins => "INNER JOIN votes ON
(votes.prompt_id=prompts.id)", :conditions => { :question_id => @question.id })
    end
  end

  # POST /questions/add
  # ==== Return
  # Added question.
  # ==== Post
  # Formatted XML of question to add.
  def add
    return unless request.post?
    xml = LibXML::XML::Parser.parse(request.raw_post)
    @questions = []
    xml.find("/questions/question").each do |question|
      @questions << Question.create(:user_id => current_user.id, :name =>
question.content)
    end
    GC.start
  end

  # POST /questions/delete
  # ==== Return
```

```
  # Question and deletion status.
  # ==== Options (params)
  # id<String>:: Converted to integer. ID of question.
  # ==== Raises
  # PermissionError:: If question does not belong to user.
  def delete
    return unless request.post?
    question = Question.find(@id = params[:id], :conditions => { :user_id =>
current_user.id })
    question.destroy
    @success = !Question.exists?(@id)
  end

  # GET /questions/list
  # ==== Return
  # Array of user questions.
  def list
    @questions = current_user.questions
    @items_count = Item.count(:conditions => { :user_id => current_user.id,
:active => true })
    @all_items_count = Item.count(:conditions => { :user_id => current_user.id
})
    @votes_count = Prompt.count(:joins => "INNER JOIN votes ON
(votes.prompt_id=prompts.id)", :conditions => { :question_id =>
current_user.question_ids })
  end
endclass RankAlgorithmsController < ApplicationController
  before_filter :force_xml, :only => :list
  before_filter :admin_required, :except => :list

  # GET /rank_algorithms
  # GET /rank_algorithms.xml
  def index
    @rank_algorithms = RankAlgorithm.find(:all)

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @rank_algorithms }
    end
  end

  # GET /rank_algorithms/1
  # GET /rank_algorithms/1.xml
  def show
    @rank_algorithm = RankAlgorithm.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @rank_algorithm }
    end
  end

  # GET /rank_algorithms/new
  # GET /rank_algorithms/new.xml
  def new
```

```
    @rank_algorithm = RankAlgorithm.new

    respond_to do |format|
      format.html # new.html.erb
      format.xml  { render :xml => @rank_algorithm }
    end
  end

  # GET /rank_algorithms/1/edit
  def edit
    @rank_algorithm = RankAlgorithm.find(params[:id])
  end

  # POST /rank_algorithms
  # POST /rank_algorithms.xml
  def create
    @rank_algorithm = RankAlgorithm.new(params[:rank_algorithm])

    respond_to do |format|
      if @rank_algorithm.save
        flash[:notice] = 'RankAlgorithm was successfully created.'
        format.html { redirect_to(@rank_algorithm) }
        format.xml  { render :xml => @rank_algorithm, :status => :created,
:location => @rank_algorithm }
      else
        format.html { render :action => "new" }
        format.xml  { render :xml => @rank_algorithm.errors, :status =>
:unprocessable_entity }
      end
    end
  end

  # PUT /rank_algorithms/1
  # PUT /rank_algorithms/1.xml
  def update
    @rank_algorithm = RankAlgorithm.find(params[:id])

    respond_to do |format|
      if @rank_algorithm.update_attributes(params[:rank_algorithm])
        flash[:notice] = 'RankAlgorithm was successfully updated.'
        format.html { redirect_to(@rank_algorithm) }
        format.xml  { head :ok }
      else
        format.html { render :action => "edit" }
        format.xml  { render :xml => @rank_algorithm.errors, :status =>
:unprocessable_entity }
      end
    end
  end

  # DELETE /rank_algorithms/1
  # DELETE /rank_algorithms/1.xml
  def destroy
    @rank_algorithm = RankAlgorithm.find(params[:id])
    @rank_algorithm.destroy
```

```
    respond_to do |format|
      format.html { redirect_to(rank_algorithms_url) }
      format.xml  { head :ok }
    end
  end

  # GET /rank_algorithms/list
  def list
    @algorithms = RankAlgorithm.all
  end
end# This controller handles the login/logout function of the site.
class SessionsController < ApplicationController
  skip_before_filter :login_required

  # render new.rhtml
  def new
  end

  def create
    logout_keeping_session!
    user = User.authenticate(params[:login], params[:password])
    if user
      # Protects against session fixation attacks, causes request forgery
      # protection if user resubmits an earlier form using back
      # button. Uncomment if you understand the tradeoffs.
      # reset_session
      self.current_user = user
      new_cookie_flag = (params[:remember_me] == "1")
      handle_remember_cookie! new_cookie_flag
      redirect_back_or_default('/')
      flash[:notice] = "Logged in successfully"
    else
      note_failed_signin
      @login       = params[:login]
      @remember_me = params[:remember_me]
      render :action => 'new'
    end
  end

  def destroy
    logout_killing_session!
    flash[:notice] = "You have been logged out."
    redirect_back_or_default('/')
  end

  protected
    # Track failed login attempts
    def note_failed_signin
      flash[:error] = "Couldn't log you in as '#{params[:login]}'"
      logger.warn "Failed login for '#{params[:login]}' from
#{request.remote_ip} at #{Time.now.utc}"
    end
endclass UsersController < ApplicationController
  skip_before_filter :login_required
```

```
  before_filter :admin_required, :only => [:index, :suspend, :unsuspend,
:destroy, :purge]
  before_filter :find_user, :only => [:suspend, :unsuspend, :destroy, :purge]
  before_filter :force_xml, :only => :add

  def index
    @users = User.find(:all)
  end

  # render new.rhtml
  def new
    @user = User.new
  end

  # Create user account but do not give out activation code.
  def create
    logout_keeping_session!
    @user = User.new(params[:user])
    @user.register! if @user && @user.valid?
    success = @user && @user.valid?
    if success && @user.errors.empty?
      # UserMailer.deliver_signup_notification(@user)
      UserMailer.deliver_admin_notification(@user)
      redirect_back_or_default('/')
      flash[:notice] = "Thanks for signing up!  We'll send an email upon account
activation."
    else
      @user.password = @user.password_confirmation = nil
      flash[:error]  = "We couldn't set up that account, sorry.  Please try
again, or contact an admin (link is above)."
      render :action => 'new'
    end
  end

  def add
    return unless request.post? && params[:key] && Base64.decode64(params[:key])
== Constants::USER_KEY
    xml = LibXML::XML::Parser.parse(request.raw_post)
    password = Base64.decode64(xml.find("/user/password").first.content)
    @user = User.new(:login => xml.find("/user/login").first.content,
      :email => xml.find("/user/email").first.content,
      :password => password, :password_confirmation => password)
    @user.register! if @user && @user.valid?
    success = @user && @user.valid?
    if success && @user.errors.empty?
      @user.activate!
      UserMailer.deliver_admin_notification(@user)
    end
  end

  def activate
    logout_keeping_session!
    user = User.find_by_activation_code(params[:id]) unless params[:id].blank?
    case
    when (!params[:id].blank?) && user && !user.active?
```

```
      user.activate!
      flash[:notice] = "Signup complete! Please sign in to continue."
      redirect_to '/login'
    when params[:id].blank?
      flash[:error] = "The activation code was missing.  Please follow the URL
from your email."
      redirect_back_or_default('/')
    else
      flash[:error]  = "We couldn't find a user with that activation code --
check your email? Or maybe you've already activated -- try signing in."
      redirect_back_or_default('/')
    end
  end

  def suspend
    @user.suspend!
    redirect_to users_path
  end

  def unsuspend
    @user.unsuspend!
    redirect_to users_path
  end

  def destroy
    @user.delete!
    redirect_to users_path
  end

  def purge
    @user.destroy
    redirect_to users_path
  end

  protected
    def find_user
      @user = User.find(params[:id])
    end
endclass VotersController < ApplicationController
  before_filter :force_xml

  # POST /voters/add
  # ==== Return
  # Added voter.
  # ==== Post
  # XML of voter to add.
  def add
    return unless request.post?
    xml = LibXML::XML::Parser.parse(request.raw_post)
    @voters = xml.find("/voters/voter").inject([]) do |voters, voter|
      cur_voter = Voter.create(:user_id => current_user.id)
      voter.find("features/feature").each do |feature|
        Feature.create(:voter_id => cur_voter.id, :name =>
feature.attributes["name"], :value => feature.content.to_i)
      end
```

```
      voters << cur_voter
    end
  end

  # GET /voters/list
  # ==== Return
  # List of user's voters.
  def list
    @voters = Voter.all(:conditions => { :user_id => current_user.id }, :include
=> :features)
  end

  # GET /voters/set/1
  # ==== Return
  # Voter.
  # ==== Options (params)
  # id<String>:: Converted to integer. Id of voter to set. Must belong to user.
  # {}<Hash>:: Name, value pairs of features to set. Any non spaces or
  # alphanumerics plus _ are removed from name.
  # ==== Raises
  # PermissionError:: If voter does not belong to user.
  def set
    p = params.clone
    @voter = Voter.find(p.delete(:id).to_i, :conditions => { :user_id =>
current_user.id }, :include => :features)
    raise PermissionError unless @voter
    p.delete(:action)
    p.delete(:controller)
    p.delete(:format)
    p.each do |name, value|
      # only space, alphanumerics, or _
      name = name.gsub(/[^(' '|\w)]/, '')
      feature = Feature.first(:conditions => { :voter_id => @voter.id, :name =>
name })
      if feature
        feature.update_attribute(:value, value.to_i)
      else
        feature = Feature.new(:name => name, :value => value.to_i)
        @voter.features << feature
      end
    end
  end
endclass VotesController < ApplicationController
  before_filter :force_xml

  # GET /votes/list
  # ==== Return
  # Array of votes.
  # ==== Options (params)
  # question_id<String>:: Converted to integer. Optional question id of votes.
  # item_id<String>:: Converted to integer. Optional item id of votes. Must
  # belong to user.
  # ==== Raises
  # PermissionError:: If question of item does not belong to user.
  def list
```

```
    conditions = { 'questions.user_id' => current_user.id }
    conditions.merge!('questions.id' => params[:question_id]) if
params[:question_id].to_i > 0
    item_id = params[:item_id].to_i
    conditions.merge!('items_prompts.item_id' => item_id) if item_id > 0
    options = {
      :include => [{:prompt => [:items, :question]}],
      :conditions => conditions,
      :order => 'votes.id'
    }
    options[:limit] = params[:limit] if params[:limit].to_i > 0
    @votes = Vote.all(options)
    if item_id > 0
      options[:conditions].delete('items_prompts.item_id')
      options[:conditions].merge!('items.id' => item_id)
      options[:joins] = "INNER JOIN items_prompts ON
(items_prompts.prompt_id=prompts.id AND items_prompts.item_id=#{item_id})"
      options[:include] = [:items, {:prompt => :question}]
      @votes_items = Vote.all(options)
    end
  end

  # GET /votes/add
  # ==== Return
  # Vote.
  # ==== Options (params)
  # prompt_id<String>:: Converted to integer. Prompt id of vote. Must belong
  # to user.
  # skip<String>:: Vote is skip if value '1'.
  # item_id<String>:: Item ids of winners.  Integer or comman seperated list
  # integers.  Any zero values are removed.
  # voter_id<String>:: Converted to integer.  Voter id or anonymous 0 voter.
  # Must belong to user.
  # ==== Raises
  # ArgumentError:: If no prompt id, prompt doesn't belong to user, or no
  # voter id.
  # PermissionError:: If voter does not belong to user.
  def add
    prompt_id = params[:prompt_id].to_i
    item_ids = (params[:item_id] &&
params[:item_id].split(',').map(&:to_i).reject(&:zero?)) || []
    skip = (params[:skip] == '1' || item_ids.empty?)
    voter_id = params[:voter_id].to_i
    # can only vote on your questions' prompts, requires voter
    raise ArgumentError unless prompt_id > 0 &&
current_user.prompts.exists?(prompt_id) && voter_id
    # if non-anonymous voter user must own voter
    raise PermissionError if voter_id > 0 && Voter.find(voter_id).user_id !=
current_user.id
    prompt = Prompt.find(prompt_id)
    question_id = prompt.question_id
    all_items = prompt.items
    old_elos = all_items.inject({}) do |hash, item|
      hash[item.id] = item.iq(question_id).position
      hash
```

```
    end
    @adj = true
    attributes = { :prompt_id => prompt_id, :voter_id => voter_id }
    attributes[:tracking] = params[:tracking] unless params[:tracking].nil?
    attributes[:response_time] = params[:response_time].to_i if
params[:response_time].to_i > 0
    @vote = Vote.new(attributes)
    raise ArgumentError unless @vote.valid?
    if skip == true
      all_items.each do |item|
        iq = item.iq(question_id)
        iq.increment!(:ratings)
        (all_items - [item]).each do |other|
          iq.increment!(:position,
Algorithms::Rank::Elo.adjust_elo(Algorithms::Rank::Elo::DRAW_SCORE, iq.position,
old_elos[other.id]))
        end
      end
    else
      items = Item.find(item_ids)
      # vote invalid if any items not in prompt
      raise ArgumentError unless (items - all_items).empty?
      items.each do |item|
        iq = item.iq(question_id)
        iq.update_attributes({ :ratings => iq.ratings + 1, :wins => iq.wins + 1
})
        (all_items - items).each do |loser|
          iq.increment!(:position,
Algorithms::Rank::Elo.adjust_elo(Algorithms::Rank::Elo::WIN_SCORE,
old_elos[item.id], old_elos[loser.id]))
          loser.iq(question_id).increment!(:position,
Algorithms::Rank::Elo.adjust_elo(Algorithms::Rank::Elo::LOSS_SCORE,
old_elos[loser.id], old_elos[item.id]))
        end
      end
      (all_items - items).each do |item|
        iq = item.iq(question_id)
        iq.update_attributes({ :ratings => iq.ratings + 1, :losses => iq.losses
+ 1 })
      end
    end
    Stat.vote(question_id, all_items)
    @vote.save!
    @vote.items << items if defined?(items) && items
  end
end
module ApplicationHelper
  def cdata(str)
    "<![CDATA[\n  #{str.to_s.gsub!("\n", "\n  ") || str}\n]]>"
  end

  def analytics
    render(:partial => 'shared/analytics') if PRODUCTION
  end
end
```

```
module HomeHelper
end
module ItemsHelper
  def active?(i)
    i.active ? 1 : 0
  end
end
module PromptAlgorithmsHelper
end
module PromptsHelper
end
module QuestionsHelper
end
module RankAlgorithmsHelper
end
module SessionsHelper
endmodule UsersHelper

  #
  # Use this to wrap view elements that the user can't access.
  # !! Note: this is an *interface*, not *security* feature !!
  # You need to do all access control at the controller level.
  #
  # Example:
  # <%= if_authorized?(:index,   User)  do link_to('List all users', users_path)
end %> |
  # <%= if_authorized?(:edit,    @user) do link_to('Edit this user',
edit_user_path) end %> |
  # <%= if_authorized?(:destroy, @user) do link_to 'Destroy', @user, :confirm =>
'Are you sure?', :method => :delete end %>
  #
  #
  def if_authorized?(action, resource, &block)
    if authorized?(action, resource)
      yield action, resource
    end
  end

  #
  # Link to user's page ('users/1')
  #
  # By default, their login is used as link text and link title (tooltip)
  #
  # Takes options
  # * :content_text => 'Content text in place of user.login', escaped with
  #   the standard h() function.
  # * :content_method => :user_instance_method_to_call_for_content_text
  # * :title_method => :user_instance_method_to_call_for_title_attribute
  # * as well as link_to()'s standard options
  #
  # Examples:
  #   link_to_user @user
  #   # => <a href="/users/3" title="barmy">barmy</a>
  #
  #   # if you've added a .name attribute:
```

```
  #  content_tag :span, :class => :vcard do
  #    (link_to_user user, :class => 'fn n', :title_method => :login,
:content_method => :name) +
  #          ': ' + (content_tag :span, user.email, :class => 'email')
  #   end
  #   # => <span class="vcard"><a href="/users/3" title="barmy" class="fn
n">Cyril Fotheringay-Phipps</a>: <span
class="email">barmy@blandings.com</span></span>
  #
  #   link_to_user @user, :content_text => 'Your user page'
  #   # => <a href="/users/3" title="barmy" class="nickname">Your user page</a>
  #
  def link_to_user(user, options={})
    raise "Invalid user" unless user
    options.reverse_merge! :content_method => :login, :title_method => :login,
:class => :nickname
    content_text      = options.delete(:content_text)
    content_text    ||= user.send(options.delete(:content_method))
    options[:title] ||= user.send(options.delete(:title_method))
    link_to h(content_text), user_path(user), options
  end

  #
  # Link to login page using remote ip address as link content
  #
  # The :title (and thus, tooltip) is set to the IP address
  #
  # Examples:
  #   link_to_login_with_IP
  #   # => <a href="/login" title="169.69.69.69">169.69.69.69</a>
  #
  #   link_to_login_with_IP :content_text => 'not signed in'
  #   # => <a href="/login" title="169.69.69.69">not signed in</a>
  #
  def link_to_login_with_IP(content_text=nil, options={})
    ip_addr           = request.remote_ip
    content_text    ||= ip_addr
    options.reverse_merge! :title => ip_addr
    if tag = options.delete(:tag)
      content_tag tag, h(content_text), options
    else
      link_to h(content_text), login_path, options
    end
  end

  #
  # Link to the current user's page (using link_to_user) or to the login page
  # (using link_to_login_with_IP).
  #
  def link_to_current_user(options={})
    if current_user
      link_to_user current_user, options
    else
      content_text = options.delete(:content_text) || 'not signed in'
      # kill ignored options from link_to_user
```

```
      [:content_method, :title_method].each{|opt| options.delete(opt)}
      link_to_login_with_IP content_text, options
    end
  end
end
module VotersHelper
end
module VotesHelper
end
class Feature < ActiveRecord::Base
  belongs_to :voter

  validates_presence_of :voter_id
  validates_presence_of :name
  validates_presence_of :value
end
class Item < ActiveRecord::Base
  belongs_to :user
  has_many :items_questions, :dependent => :destroy
  has_many :questions, :through => :items_questions
  has_and_belongs_to_many :prompts
  has_and_belongs_to_many :stats
  has_and_belongs_to_many :votes
  has_and_belongs_to_many :prompt_requests

  validates_presence_of :user_id

  before_destroy :destroy_habtms

  attr_accessor :score

  def iq(question_id)
    items_questions.find_by_question_id(question_id)
  end

  def destroy_habtms
    Prompt.destroy(prompts)
    prompts.clear
    Stat.destroy(stats)
    stats.clear
    Vote.destroy(votes)
    votes.clear
    PromptRequest.destroy(prompt_requests)
    prompt_requests.clear
  end
end
class ItemsQuestion < ActiveRecord::Base
  belongs_to :item
  belongs_to :question

  validates_presence_of :item_id
  validates_presence_of :question_id
end
class Prompt < ActiveRecord::Base
  belongs_to :question
```

```
  belongs_to :prompt_algorithm
  belongs_to :voter
  has_one :vote, :dependent => :destroy
  has_and_belongs_to_many :items

  validates_presence_of :question_id
  validates_presence_of :prompt_algorithm_id
  validates_presence_of :voter_id

  class << self
    include Algorithms::Prompt

    # Create prompts based on conditions. If prime is true items used in prompts
    # are relative to their stats values.  Otherwise items used are randomly
    # chosen.
    # ==== Return
    # Array of prompt ids.
    # ==== Parameters
    # question_id<int>:: The question id.
    # voter_id<int>:: The voter id.
    # count<int>:: The number of prompts to generate.
    # prime<boolean>:: If true prompts generated using prime algorithm.
    # Otherwise not.
    def fetch(question_id, voter_id, count, prime)
      Item.count(conditions(question_id)) > 1 ? create_prompts(question_id,
voter_id, count, prime) : {}
    end
  end
endclass PromptAlgorithm < ActiveRecord::Base
  has_many :prompts

  validates_presence_of :name
  validates_presence_of :data
end
class PromptRequest < ActiveRecord::Base
  belongs_to :question
  belongs_to :voter
  has_and_belongs_to_many :items

  validates_numericality_of :question_id
  validates_numericality_of :voter_id
end
class Question < ActiveRecord::Base
  belongs_to :user
  has_many :items_questions, :dependent => :destroy
  has_many :items, :through => :items_questions
  has_many :prompts, :dependent => :destroy
  has_many :prompt_requests, :dependent => :destroy

  validates_presence_of :user_id
  validates_presence_of :name

end
class RankAlgorithm < ActiveRecord::Base
  validates_presence_of :name
```

```
  validates_presence_of :data
end
class Stat < ActiveRecord::Base
  belongs_to :question
  belongs_to :rank_algorithm
  has_and_belongs_to_many :items

  validates_presence_of :question_id
  validates_presence_of :views
  validates_presence_of :votes
  validates_presence_of :score

  class << self
    include Constants::Stat

    # Update view for or create stat for items
    def view(question_id, items)
      return if items.empty? || !RankAlgorithm.exists?(DEFAULT_RANK_ALGO)
      stat = for_items(items)
      if stat
        views = stat.views + 1
        stat.update_attributes({ :views => views, :score =>
score(stat.rank_algorithm.data.to_i, stat.votes, views) })
      else
        stat = create(default(question_id))
        stat.items << items
      end
      stat
    end

    # Update vote for or create stat for items. Assume items are linked to the
    # passed question id as they should be.
    def vote(question_id, items)
      return if items.empty? || !RankAlgorithm.exists?(DEFAULT_RANK_ALGO)
      stat = for_items(items)
      if stat
        votes = stat.votes + 1
        # if fewer views thatn votes a view wasn't counted, set to number of
votes
        views = stat.views < votes ? votes : stat.views
        options = { :votes => votes, :score =>
score(stat.rank_algorithm.data.to_i, votes, views) }
        options.merge!(:views => views) if views != stat.views
        stat.update_attributes(options)
      else
        stat = create(default(question_id, 1))
        stat.items << items
      end
      stat
    end

    # Get stats for specific items.  Any set of items should have one stat.
    # Merge together multiple stats.
    def for_items(items)
```

```
      sql = "SELECT items_stats.stat_id FROM items_stats WHERE
items_stats.item_id="
      stat_ids = items.map { |item|
find_by_sql("#{sql}#{item.id}").map(&:stat_id) }.closure
      if stat_ids.length > 1
        # merge multiple stats for same item group
        stats = find(stat_ids)
        stat = stats.shift
        stats.each do |el|
          stat.votes += el.votes
          stat.views += el.views
        end
        stat.score = score((stat.rank_algorithm || default_rank_algo).data.to_i,
stat.votes, stat.views)
        stat.save!
        stat
      else
        !stat_ids.empty? && find(stat_ids.first)
      end
    end

    private
      # Compute score as |p|^alpha, with p = 2 * votes - views, sign = sign(p).
      def score(alpha, votes, views) #:doc:
        n = 2 * votes - views
        n != 0 ? (n / n.abs) * (n.abs) ** alpha : 0
      end

      def default(question_id, votes = 0)
        rank_algo = default_rank_algo
        { :views => 1,
          :votes => votes,
          :question_id => question_id,
          :score => score(rank_algo.data.to_i, votes, 1),
          :rank_algorithm_id => rank_algo.id
        }
      end

      def default_rank_algo
        RankAlgorithm.find(DEFAULT_RANK_ALGO)
      end
  end
end
class SystemNotifier < ActionMailer::Base
  SYSTEM_EMAIL_ADDRESS = %{"Error Notifier" <error@photocracy.org>}
  EXCEPTION_RECIPIENTS = %w{error@photocracy.org}

  def exception(controller, request, visit, exception, sent_on = Time.now)
    @subject = sprintf("[ERROR] %s\#%s (%s) %s", controller.controller_name,
controller.action_name, exception.class, exception.message.inspect)
    @body = {
      :controller => controller,
      :request => request,
      :exception => exception,
      :backtrace => sanitize_backtrace(exception.backtrace),
```

```
      :host => request.env["HTTP_HOST"], "rails_root" => rails_root,
      :visit => visit
    }
    @sent_on = sent_on
    @from = SYSTEM_EMAIL_ADDRESS
    @recipients = EXCEPTION_RECIPIENTS
    @content_type = "text/html"
  end

  private
    def sanitize_backtrace(trace)
      re = Regexp.new(/^#{Regexp.escape(rails_root)}/)
      trace.map do |line|
        Pathname.new(line.gsub(re, "[RAILS_ROOT")).cleanpath.to_s
      end
    end

    def rails_root
      @rails_root ||= Pathname.new(RAILS_ROOT).cleanpath.to_s
    end
endrequire 'digest/sha1'

class User < ActiveRecord::Base
  include Authentication
  include Authentication::ByPassword
  include Authentication::ByCookieToken
  include Authorization::AasmRoles

  # associations
  has_many :questions, :dependent => :destroy
  has_many :items, :dependent => :destroy
  has_many :voters, :dependent => :destroy
  has_many :prompts, :through => :questions, :dependent => :destroy

  validates_presence_of     :login
  validates_length_of       :login,    :within => 3..40
  validates_uniqueness_of   :login
  validates_format_of       :login,    :with => Authentication.login_regex,
:message => Authentication.bad_login_message

  validates_format_of       :name,     :with => Authentication.name_regex,
:message => Authentication.bad_name_message, :allow_nil => true
  validates_length_of       :name,     :maximum => 100

  validates_presence_of     :email
  validates_length_of       :email,    :within => 6..100 #r@a.wk
  validates_uniqueness_of   :email
  validates_format_of       :email,    :with => Authentication.email_regex,
:message => Authentication.bad_email_message

  # HACK HACK HACK -- how to do attr_accessible from here?
  # prevents a user from submitting a crafted form that bypasses activation
  # anything else you want your user to change should be added here.
  attr_accessible :login, :email, :name, :password, :password_confirmation
```

```
  # Authenticates a user by their login name and unencrypted password.  Returns
the user or nil.
  #
  # uff.  this is really an authorization, not authentication routine.
  # We really need a Dispatch Chain here or something.
  # This will also let us return a human error message.
  #
  def self.authenticate(login, password)
    return nil if login.blank? || password.blank?
    u = find_in_state(:first, :active, :conditions => {:login => login}) ||
find_in_state(:first, :admin, :conditions => {:login => login}) # need to get
the salt
    u && u.authenticated?(password) ? u : nil
  end

  def login=(value)
    write_attribute :login, (value ? value.downcase : nil)
  end

  def email=(value)
    write_attribute :email, (value ? value.downcase : nil)
  end

  # Delete all the user's items and their connections to stats, votes, prompts.
  # Delete all stats, items_questions, prompt_requests, and prompts for the
  # user's questions.
  def destroy_items
    item_ids_str = item_ids.join(',')
    prompt_ids_str = prompt_ids.join(',')
    question_ids_str = question_ids.join(',')
    Vote.delete_all("prompt_id IN (#{prompt_ids_str})") unless
prompt_ids_str.empty?
    unless item_ids_str.empty?
      ActiveRecord::Base.connection.execute("DELETE FROM items_stats WHERE
item_id IN (#{item_ids_str})")
      ActiveRecord::Base.connection.execute("DELETE FROM items_votes WHERE
item_id IN (#{item_ids_str})")
      ActiveRecord::Base.connection.execute("DELETE FROM items_prompts WHERE
item_id IN (#{item_ids_str})")
    end
    Item.delete_all("user_id=#{id}")
    unless question_ids_str.empty?
      Stat.delete_all("question_id IN (#{question_ids_str})")
      ItemsQuestion.delete_all("question_id IN (#{question_ids_str})")
      PromptRequest.delete_all("question_id IN (#{question_ids_str})")
      Prompt.delete_all("question_id IN (#{question_ids_str})")
    end
  end

  # Delete all data connected to the user.
  def destroy_data
    item_ids_str = item_ids.join(',')
    prompt_ids_str = prompt_ids.join(',')
    question_ids_str = question_ids.join(',')
    voter_ids_str = voter_ids.join(',')
```

```
    Vote.delete_all("prompt_id IN (#{prompt_ids_str})") unless
prompt_ids_str.empty?
    unless item_ids_str.empty?
      ActiveRecord::Base.connection.execute("DELETE FROM items_stats WHERE
item_id IN (#{item_ids_str})")
      ActiveRecord::Base.connection.execute("DELETE FROM items_votes WHERE
item_id IN (#{item_ids_str})")
      ActiveRecord::Base.connection.execute("DELETE FROM items_prompts WHERE
item_id IN (#{item_ids_str})")
    end
    Item.delete_all("user_id=#{id}")
    unless question_ids_str.empty?
      Stat.delete_all("question_id IN (#{question_ids_str})")
      ItemsQuestion.delete_all("question_id IN (#{question_ids_str})")
      PromptRequest.delete_all("question_id IN (#{question_ids_str})")
      Prompt.delete_all("question_id IN (#{question_ids_str})")
    end
    Question.delete_all("user_id=#{id}")
    Feature.delete_all("voter_id IN (#{voter_ids_str})") unless
voter_ids_str.empty?
    Voter.delete_all("user_id=#{id}")
  end

protected
  def make_activation_code
    self.deleted_at = nil
    self.activation_code = self.class.make_token
  end
end
class UserMailer < ActionMailer::Base
  def signup_notification(user)
    setup_email(user)
    @subject    += 'Please activate your new account'
    @body[:url]  = "#{Constants::BASE_URL}activate/#{user.activation_code}"
  end

  def admin_notification(user)
    setup_email(user)
    @recipients = Constants::ADMIN_EMAILS
  end

  def activation(user)
    setup_email(user)
    @subject    += 'Your account has been activated!'
    @body[:url]  = Constants::BASE_URL
  end

  protected
    def setup_email(user)
      @recipients  = "#{user.email}"
      @from        = "pairwise@photocracy.org"
      @subject     = "[pairwise] "
      @content_type = "text/html"
      @sent_on     = Time.now
      @body[:user] = user
```

```
    end
end
class Vote < ActiveRecord::Base
  belongs_to :prompt
  belongs_to :voter
  has_and_belongs_to_many :items

  validates_presence_of :prompt_id
  validates_uniqueness_of :prompt_id
  validates_presence_of :voter_id
end
class Voter < ActiveRecord::Base
  has_many :features, :dependent => :destroy
  belongs_to :user
  belongs_to :prompt

  validates_presence_of :user_id
end
# Don't change this file!
# Configure your app in config/environment.rb and config/environments/*.rb

RAILS_ROOT = "#{File.dirname(__FILE__)}/.." unless defined?(RAILS_ROOT)

module Rails
  class << self
    def boot!
      unless booted?
        preinitialize
        pick_boot.run
      end
    end

    def booted?
      defined? Rails::Initializer
    end

    def pick_boot
      (vendor_rails? ? VendorBoot : GemBoot).new
    end

    def vendor_rails?
      File.exist?("#{RAILS_ROOT}/vendor/rails")
    end

    def preinitialize
      load(preinitializer_path) if File.exist?(preinitializer_path)
    end

    def preinitializer_path
      "#{RAILS_ROOT}/config/preinitializer.rb"
    end
  end

  class Boot
    def run
```

```
      load_initializer
      Rails::Initializer.run(:set_load_path)
    end
  end

  class VendorBoot < Boot
    def load_initializer
      require "#{RAILS_ROOT}/vendor/rails/railties/lib/initializer"
      Rails::Initializer.run(:install_gem_spec_stubs)
    end
  end

  class GemBoot < Boot
    def load_initializer
      self.class.load_rubygems
      load_rails_gem
      require 'initializer'
    end

    def load_rails_gem
      if version = self.class.gem_version
        gem 'rails', version
      else
        gem 'rails'
      end
    rescue Gem::LoadError => load_error
      $stderr.puts %(Missing the Rails #{version} gem. Please `gem install -
v=#{version} rails`, update your RAILS_GEM_VERSION setting in
config/environment.rb for the Rails version you do have installed, or comment
out RAILS_GEM_VERSION to use the latest version installed.)
      exit 1
    end

    class << self
      def rubygems_version
        Gem::RubyGemsVersion if defined? Gem::RubyGemsVersion
      end

      def gem_version
        if defined? RAILS_GEM_VERSION
          RAILS_GEM_VERSION
        elsif ENV.include?('RAILS_GEM_VERSION')
          ENV['RAILS_GEM_VERSION']
        else
          parse_gem_version(read_environment_rb)
        end
      end

      def load_rubygems
        require 'rubygems'
        min_version = '1.1.1'
        unless rubygems_version >= min_version
          $stderr.puts %Q(Rails requires RubyGems >= #{min_version} (you have
#{rubygems_version}). Please `gem update --system` and try again.)
          exit 1
```

```
        end

      rescue LoadError
        $stderr.puts %Q(Rails requires RubyGems >= #{min_version}. Please
install RubyGems and try again: http://rubygems.rubyforge.org)
        exit 1
      end

      def parse_gem_version(text)
        $1 if text =~
/^[^#]*RAILS_GEM_VERSION\s*=\s*["']([!~<>=]*\s*[\d.]+)["']/
      end

      private
        def read_environment_rb
          File.read("#{RAILS_ROOT}/config/environment.rb")
        end
    end
  end
end

# All that for this:
Rails.boot!
set :keep_releases, 5
set :application,   "pairwise"
set :ip,            "173.45.234.238"
set :repository,    "svn+pssh://#{ip}/home/svn/repos/pairwise/"
set :user,          'photocracy'
set :password,      'c3rd0FASC1STA'
set :scm_username,  'svn'
set :scm_password,  'f3y3rab3nd'
set :scm,           :subversion
set :monit_group,   'pair'
set :deploy_to,     "/var/www/#{application}/"
set :deploy_via,    :copy

ssh_options[:port] = 11235
ssh_options[:paranoid] = false
default_run_options[:pty] = true

task :stage do
  set :deploy_to,  "#{deploy_to.chop}_stage/"
  set :monit_group, "#{monit_group}_s"
  set :repository, "#{repository}trunk"
end

task :prod do
  set :tag, "0.2.4.0"
  set :repository, "#{repository}tags/#{tag}"
end

role :web, ip
role :app, ip
role :db, ip, :primary => true
```

```
# ========================================================================
# TASKS
after "deploy", "deploy:cleanup"
after "deploy:update_code", "deploy:symlink_configs"

# ========================================================================
namespace :deploy do
  task :symlink_configs, :roles => :app, :except => {:no_symlink => true} do
    run <<-CMD
      cd #{release_path} &&
      ln -nfs #{shared_path}/log #{release_path}/config/log &&
      ln -nfs #{shared_path}/config/mongrel.yml
#{release_path}/config/mongrel.yml &&
      ln -nfs #{shared_path}/config/database.yml
#{release_path}/config/database.yml
    CMD
  end

  desc "Restart the Monit processes on the app server by calling monit."
  task :restart, :roles => :app do
    monit.restart
  end
end

namespace :monit do
  desc <<-DESC
  Start Monit processes on the app server.  This uses the :use_sudo variable to
determine whether to use sudo or
  not. By default, :use_sudo is set to true.
  DESC
  task :start, :roles => :app do
    sudo "/usr/sbin/monit start all -g #{monit_group}"
  end

  desc <<-DESC
  Restart the Monit processes on the app server by starting and stopping the
cluster. This uses the :use_sudo
  variable to determine whether to use sudo or not. By default, :use_sudo is set
to true.
  DESC
  task :restart, :roles => :app do
    sudo "/usr/sbin/monit restart all -g #{monit_group}"
  end

  desc <<-DESC
  Stop the Monit processes on the app server.  This uses the :use_sudo
  variable to determine whether to use sudo or not. By default, :use_sudo is
  set to true.
  DESC
  task :stop, :roles => :app do
    sudo "/usr/sbin/monit stop all -g #{monit_group}"
  end
endRAILS_GEM_VERSION = '2.3.2' unless defined? RAILS_GEM_VERSION

PRODUCTION = false
```

```
LOG_DETAIL = !PRODUCTION

# Bootstrap the Rails environment, frameworks, and default configuration
require File.join(File.dirname(__FILE__), 'boot')
require 'hodel_3000_compliant_logger' if LOG_DETAIL

Rails::Initializer.run do |config|
  config.gem 'rubyist-aasm', :lib => 'aasm'
  config.gem 'libxml-ruby', :lib => 'libxml'

  # Use the Hodel3000 compliant logger
  config.logger = Hodel3000CompliantLogger.new(config.log_path) if LOG_DETAIL

  # Use default local time.
  # config.time_zone = 'UTC'

  # Your secret key for verifying cookie session data integrity.
  # If you change this key, all old sessions will become invalid!
  # Make sure the secret is at least 30 characters and all random,
  # no regular words or you'll be exposed to dictionary attacks.
  config.action_controller.session = {
    :key => '_pairwise_session',
    :secret      =>
'c6dccd828c3249f9cbc83e0682f2986442594a794bce2ae18461ea443acfb52411c3d9117e7899f
7503195a5334db8d4898c9c9eac2420db1a4ce37f1f1b9c97'
  }
  config.action_controller.session_store = :active_record_store
  config.action_mailer.delivery_method = :smtp
end# Settings specified here will take precedence over those in
config/environment.rb

# In the development environment your application's code is reloaded on
# every request.  This slows down response time but is perfect for development
# since you don't have to restart the webserver when you make code changes.
config.cache_classes = false

# Log error messages when you accidentally call methods on nil.
config.whiny_nils = true

# Show full error reports and disable caching
config.action_controller.consider_all_requests_local = true
config.action_view.debug_rjs                         = true
config.action_controller.perform_caching             = false

# Don't care if the mailer can't send
config.action_mailer.raise_delivery_errors = false# Settings specified here will
take precedence over those in config/environment.rb

# The production environment is meant for finished, "live" apps.
# Code is not reloaded between requests
config.cache_classes = true

# Use a different logger for distributed setups
# config.logger = SyslogLogger.new
```

```
# Full error reports are disabled and caching is turned on
config.action_controller.consider_all_requests_local = false
config.action_controller.perform_caching             = true
config.cache_classes                                 = true

# Use a different cache store in production
# config.cache_store = :mem_cache_store

# Enable serving of images, stylesheets, and javascripts from an asset server
# config.action_controller.asset_host                  =
"http://assets.example.com"

# Disable delivery errors, bad email addresses will be ignored
# config.action_mailer.raise_delivery_errors = false
# Settings specified here will take precedence over those in
config/environment.rb

# The test environment is used exclusively to run your application's
# test suite.  You never need to work with it otherwise.  Remember that
# your test database is "scratch space" for the test suite and is wiped
# and recreated between test runs.  Don't rely on the data there!
config.cache_classes = true

# Log error messages when you accidentally call methods on nil.
config.whiny_nils = true

# Show full error reports and disable caching
config.action_controller.consider_all_requests_local = true
config.action_controller.perform_caching             = false

# Disable request forgery protection in test environment
config.action_controller.allow_forgery_protection    = false

# Tell Action Mailer not to deliver emails to the real world.
# The :test delivery method accumulates sent emails in the
# ActionMailer::Base.deliveries array.
config.action_mailer.delivery_method = :test
class ActionController::Request
  # patched version to not raise on xml parameter parse errors
  def parse_formatted_request_parameters
    return {} if content_length.zero?

    content_type, boundary =
self.class.extract_multipart_boundary(content_type_with_parameters)

    # Don't parse params for unknown requests.
    return {} if content_type.blank?

    mime_type = Mime::Type.lookup(content_type)
    strategy = ActionController::Base.param_parsers[mime_type]

    # Only multipart form parsing expects a stream.
    body = (strategy && strategy != :multipart_form) ? raw_post : self.body

    case strategy
```

```
      when Proc
        strategy.call(body)
      when :url_encoded_form
        self.class.clean_up_ajax_request_body! body
        self.class.parse_query_parameters(body)
      when :multipart_form
        self.class.parse_multipart_form_parameters(body, boundary,
content_length, env)
      when :xml_simple, :xml_node
        @no_raise = true
        body.blank? ? {} : Hash.from_xml(body).with_indifferent_access
      when :yaml
        YAML.load(body)
      when :json
        if body.blank?
          {}
        else
          data = ActiveSupport::JSON.decode(body)
          data = {:_json => data} unless data.is_a?(Hash)
          data.with_indifferent_access
        end
      else
        {}
    end
  rescue Exception => e # YAML, XML or Ruby code block errors
    if @no_raise
      {}
    else
      raise
      { "body" => body,
        "content_type" => content_type_with_parameters,
        "content_length" => content_length,
        "exception" => "#{e.message} (#{e.class})",
        "backtrace" => e.backtrace }
    end
  end
endclass PermissionError < Exception
end# Be sure to restart your server when you modify this file.

# Add new inflection rules using the following format
# (all these examples are active by default):
# ActiveSupport::Inflector.inflections do |inflect|
#   inflect.plural /^(ox)$/i, '\1en'
#   inflect.singular /^(ox)en/i, '\1'
#   inflect.irregular 'person', 'people'
#   inflect.uncountable %w( fish sheep )
# end
# Be sure to restart your server when you modify this file.

# Add new mime types for use in respond_to blocks:
# Mime::Type.register "text/richtext", :rtf
# Mime::Type.register_alias "text/html", :iphone
# These settings change the behavior of Rails 2 apps and will be defaults
# for Rails 3. You can remove this initializer when Rails 3 is released.
```

```
if defined?(ActiveRecord)
  # Include Active Record class name as root for JSON serialized output.
  ActiveRecord::Base.include_root_in_json = true

  # Store the full class name (including module namespace) in STI type column.
  ActiveRecord::Base.store_full_sti_class = true
end

# Use ISO 8601 format for JSON serialized times and dates.
ActiveSupport.use_standard_json_time_format = true

# Don't escape HTML entities in JSON, leave that for the #json_escape helper.
# if you're including raw json in an HTML page.
ActiveSupport.escape_html_entities_in_json = falseclass Array
  # evaluate on outside of loop
  def sumup(on = nil)
    on ? self.inject(0) { |sum, el| sum += el.send(on).to_f } : self.inject(0) {
|sum, el| sum += el.to_f }
  end

  def closure
    self.inject(self.first) { |closure, el| closure &= el; closure }
  end
end# A Site key gives additional protection against a dictionary attack if your
# DB is ever compromised.  With no site key, we store
#   DB_password = hash(user_password, DB_user_salt)
# If your database were to be compromised you'd be vulnerable to a dictionary
# attack on all your stupid users' passwords.  With a site key, we store
#   DB_password = hash(user_password, DB_user_salt, Code_site_key)
# That means an attacker needs access to both your site's code *and* its
# database to mount an "offline dictionary
attack.":http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/web-
authentication.html
#
# It's probably of minor importance, but recommended by best practices: 'defense
# in depth'.  Needless to say, if you upload this to github or the youtubes or
# otherwise place it in public view you'll kinda defeat the point.  Your users'
# passwords are still secure, and the world won't end, but defense_in_depth -=
1.
#
# Please note: if you change this, all the passwords will be invalidated, so DO
# keep it someplace secure.  Use the random value given or type in the lyrics to
# your favorite Jay-Z song or something; any moderately long, unpredictable
text.
REST_AUTH_SITE_KEY         = 'd8de37c85f588d8414e1402513af8101b1dfd2d8'

# Repeated applications of the hash make brute force (even with a compromised
# database and site key) harder, and scale with Moore's law.
#
#   bq. "To squeeze the most security out of a limited-entropy password or
#   passphrase, we can use two techniques [salting and stretching]... that are
#   so simple and obvious that they should be used in every password system.
#   There is really no excuse not to use them." http://tinyurl.com/37lb73
#   Practical Security (Ferguson & Scheier) p350
#
```

```
# A modest 10 foldings (the default here) adds 3ms.  This makes brute forcing 10
# times harder, while reducing an app that otherwise serves 100 reqs/s to 78
signin
# reqs/s, an app that does 10reqs/s to 9.7 reqs/s
#
# More:
# * http://www.owasp.org/index.php/Hashing_Java
# * "An Illustrated Guide to Cryptographic
Hashes":http://www.unixwiz.net/techtips/iguide-crypto-hashes.html

REST_AUTH_DIGEST_STRETCHES = 10
require "smtp_tls"

mailer_config = File.open("#{RAILS_ROOT}/config/mailer.yml")
mailer_options = YAML.load(mailer_config)
ActionMailer::Base.smtp_settings = mailer_options
require 'libxml'

class LibXML::XML::Parser
  class << self
    def parse(string)
      string(string).parse
    end
  end
endActionController::Routing::Routes.draw do |map|
  map.logout '/logout', :controller => 'sessions', :action => 'destroy'
  map.login '/login', :controller => 'sessions', :action => 'new'
  map.register '/register', :controller => 'users', :action => 'create'
  map.signup '/signup', :controller => 'users', :action => 'new'
  map.activate '/activate/:id', :controller => 'users', :action => 'activate',
:activation_code => nil
  map.learn '/learn', :controller => 'home', :action => 'learn'
  map.api '/api', :controller => 'home', :action => 'api'
  map.about '/about', :controller => 'home', :action => 'about'
  map.resources :users, :member => { :suspend   => :get,
                                     :unsuspend => :get,
                                     :purge     => :delete }
  map.resource :session
  map.resources(:items,
    :member => { :activate => [:post, :get], :suspend => [:post, :get], :delete
=> :post },
    :collection => {
      :add => :post,
      :list => :get
  })
  map.connect('items/list/:question_id/:rank_algorithm/:limit/:offset/:order/',
    :controller => 'items',
    :action => 'list',
    :defaults => { :question_id => 0, :rank_algorithm => 0, :limit => 0, :offset
=> 0, :order => 0 })
  map.resources :prompts, :collection => { :list => :get }, :member => { :view
=> :get }
  map.connect('prompts/list/:question_id/:item_id', :controller => 'prompts',
:action => 'list',
    :defaults => { :question_id => 0, :item_id => 'A' })
```

```
  map.connect('prompts/create/:question_id/:voter_id/:n/:prime', :controller =>
'prompts', :action => 'create',
    :conditions => { :method => [:post, :get] }, :defaults => { :n => 1, :prime
=> 0, :voter_id => 0 })
  map.resources :questions, :member => { :delete => :post }, :collection => {
:list => :get, :add => :post }
  map.resources(:votes, :collection => { :list => :get, :add => [:post, :get] },
    :conditions => { :method => [:post, :get] })
  map.connect('votes/list/:question_id/:item_id', :controller => 'votes',
:action => 'list',
    :defaults => { :question_id => 0, :item_id => 0 })
  map.connect('votes/add/:prompt_id/:voter_id/:response_time/:item_id/',
:controller => 'votes', :action => 'add',
    :conditions => { :method => [:post, :get] }, :defaults => { :response_time
=> 0, :item_id => 0, :voter_id => 0 })
  map.resources :voters, :collection => { :list => :get, :add => :post }
  map.resources :rank_algorithms, :collection => { :build_stats => :get, :list
=> :get }
  map.resources :prompt_algorithms, :collection => { :list => :get }

  map.root :controller => 'home'
  map.connect ':controller/:action/:id'
  map.connect ':controller/:action/:id.:format'
end
class CreateQuestions < ActiveRecord::Migration
  def self.up
    create_table :questions do |t|
      t.integer :user_id, :null => false
      t.string :name, :null => false
      t.timestamps
    end

    add_index :questions, :user_id
  end

  def self.down
    drop_table :questions
  end
end
class CreateItems < ActiveRecord::Migration
  def self.up
    create_table :items do |t|
      t.integer :user_id, :null => false
      t.text :data
      t.boolean :active, :default => false
      t.timestamps
    end

    add_index :items, :user_id
  end

  def self.down
    drop_table :items
  end
endclass CreatePrompts < ActiveRecord::Migration
```

```
  def self.up
    create_table :prompts do |t|
      t.integer :question_id, :null => false
      t.integer :prompt_algorithm_id, :null => false
      t.integer :voter_id, :null => false
      t.boolean :active, :default => true
      t.timestamps
    end

    create_table :items_prompts, :id => false do |t|
      t.integer :item_id, :null => false
      t.integer :prompt_id, :null => false
    end

    add_index :prompts, :question_id
    add_index :prompts, :voter_id

    add_index :items_prompts, :item_id
    add_index :items_prompts, :prompt_id
  end

  def self.down
    drop_table :prompts
    drop_table :items_prompts
  end
endclass CreateVotes < ActiveRecord::Migration
  def self.up
    create_table :votes do |t|
      t.integer :prompt_id, :null => false
      t.integer :voter_id, :null => false
      t.integer :response_time
      t.timestamps
    end

    create_table :items_votes, :id => false do |t|
      t.integer :item_id, :null => false
      t.integer :vote_id, :null => false
    end

    add_index :votes, :prompt_id
    add_index :votes, :voter_id

    add_index :items_votes, :item_id
    add_index :items_votes, :vote_id
  end

  def self.down
    drop_table :votes
    drop_table :items_votes
  end
endclass CreatePromptAlgorithms < ActiveRecord::Migration
  def self.up
    create_table :prompt_algorithms do |t|
      t.string :name, :null => false
      t.string :data, :null => false
```

```
      t.timestamps
    end
  end

  def self.down
    drop_table :prompt_algorithms
  end
endclass CreateRankAlgorithms < ActiveRecord::Migration
  def self.up
    create_table :rank_algorithms do |t|
      t.string :name, :null => false
      t.string :data, :null => false
      t.timestamps
    end
  end

  def self.down
    drop_table :rank_algorithms
  end
endclass CreateSessions < ActiveRecord::Migration
  def self.up
    create_table :sessions do |t|
      t.string :session_id, :null => false
      t.text :data
      t.timestamps
    end

    add_index :sessions, :session_id
    add_index :sessions, :updated_at
  end

  def self.down
    drop_table :sessions
  end
end
class CreateUsers < ActiveRecord::Migration
  def self.up
    create_table "users", :force => true do |t|
      t.column :login,                     :string, :limit => 40
      t.column :name,                      :string, :limit => 100, :default =>
'', :null => true
      t.column :email,                     :string, :limit => 100
      t.column :crypted_password,          :string, :limit => 40
      t.column :salt,                      :string, :limit => 40
      t.column :created_at,                :datetime
      t.column :updated_at,                :datetime
      t.column :remember_token,            :string, :limit => 40
      t.column :remember_token_expires_at, :datetime
      t.column :activation_code,           :string, :limit => 40
      t.column :activated_at,              :datetime
      t.column :state,                     :string, :null => :no, :default =>
'passive'
      t.column :deleted_at,                :datetime
    end
```

```
    add_index :users, :login, :unique => true
  end

  def self.down
    drop_table "users"
  end
end
class CreateVoters < ActiveRecord::Migration
  def self.up
    create_table :voters do |t|
      t.integer :user_id, :null => false
      t.timestamps
    end

    add_index :voters, :user_id
  end

  def self.down
    drop_table :voters
  end
end
class CreateFeatures < ActiveRecord::Migration
  def self.up
    create_table :features do |t|
      t.integer :voter_id, :null => false
      t.string :name, :null => false
      t.integer :value, :null => false

      t.timestamps
    end

    add_index :features, :voter_id
    add_index :features, :name
  end

  def self.down
    drop_table :features
  end
end
class CreatePromptRequests < ActiveRecord::Migration
  def self.up
    create_table :prompt_requests do |t|
      t.integer :question_id, :null => false
      t.integer :voter_id, :null => false
      t.integer :count, :default => 1
      t.text :item_ids
      t.timestamps
    end

    add_index :prompt_requests, :question_id
    add_index :prompt_requests, :voter_id
  end

  def self.down
    drop_table :prompt_requests
```

```
  end
end
class CreateItemsQuestions < ActiveRecord::Migration
  def self.up
    create_table :items_questions do |t|
      t.integer :item_id, :null => false
      t.integer :question_id, :null => false
      t.integer :position, :default => 1400, :null => false
      t.integer :wins, :default => 0, :null => false
      t.integer :ratings, :default => 0, :null => false
      t.timestamps
    end

    add_index :items_questions, :item_id
    add_index :items_questions, :question_id
  end

  def self.down
    drop_table :items_questions
  end
end
class CreateStats < ActiveRecord::Migration
  def self.up
    create_table :stats do |t|
      t.integer :question_id, :null => false
      t.integer :rank_algorithm_id
      t.integer :views, :null => false, :default => 0
      t.integer :votes, :null => false, :default => 0
      t.float :score, :null => false, :default => 0
      t.timestamps
    end

    create_table :items_stats, :id => false do |t|
      t.integer :item_id, :null => :false
      t.integer :stat_id, :null => :false
    end

    add_index :stats, :question_id
    add_index :items_stats, :item_id
    add_index :items_stats, :stat_id
  end

  def self.down
    drop_table :stats
    drop_table :items_stats
  end
end
class AddItemsPromptRequests < ActiveRecord::Migration
  def self.up
    remove_column :prompt_requests, :item_ids

    create_table :items_prompt_requests, :id => false do |t|
      t.integer :item_id, :null => false
      t.integer :prompt_request_id, :null => false
    end
```

```
    add_index :items_prompt_requests, :item_id
    add_index :items_prompt_requests, :prompt_request_id
  end

  def self.down
    add_column :prompt_requests, :item_ids, :text

    drop_table :items_prompt_requests
  end
end
class AddTrackingToVotes < ActiveRecord::Migration
  def self.up
    add_column :votes, :tracking, :text
  end

  def self.down
    remove_column :votes, :tracking
  end
end
class AddActiveIndexes < ActiveRecord::Migration
  def self.up
    add_index :items, :active
  end

  def self.down
    remove_index :items, :active
  end
end
class AddItemsQuestionsLosses < ActiveRecord::Migration
  def self.up
    add_column :items_questions, :losses, :integer, :default => 0
  end

  def self.down
    remove_column :items_questions, :losses
  end
end
class AddTrackingToItems < ActiveRecord::Migration
  def self.up
    add_column :items, :tracking, :text
  end

  def self.down
    remove_column :items, :tracking
  end
end
# This file is auto-generated from the current state of the database. Instead of
editing this file,
# please use the migrations feature of Active Record to incrementally modify
your database, and
# then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
database schema. If you need
```

```
# to create the application database on another system, you should be using
db:schema:load, not running
# all the migrations from scratch. The latter is a flawed and unsustainable
approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20090527113348) do

  create_table "features", :force => true do |t|
    t.integer  "voter_id",   :null => false
    t.string   "name",       :null => false
    t.integer  "value",      :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  add_index "features", ["name"], :name => "index_features_on_name"
  add_index "features", ["voter_id"], :name => "index_features_on_voter_id"

  create_table "items", :force => true do |t|
    t.integer  "user_id",                        :null => false
    t.boolean  "active",     :default => false
    t.datetime "created_at"
    t.datetime "updated_at"
    t.text     "data"
    t.text     "tracking"
  end

  add_index "items", ["active"], :name => "index_items_on_active"
  add_index "items", ["user_id"], :name => "index_items_on_user_id"

  create_table "items_prompt_requests", :id => false, :force => true do |t|
    t.integer "item_id",           :null => false
    t.integer "prompt_request_id", :null => false
  end

  add_index "items_prompt_requests", ["item_id"], :name =>
"index_items_prompt_requests_on_item_id"
  add_index "items_prompt_requests", ["prompt_request_id"], :name =>
"index_items_prompt_requests_on_prompt_request_id"

  create_table "items_prompts", :id => false, :force => true do |t|
    t.integer "item_id",   :null => false
    t.integer "prompt_id", :null => false
  end

  add_index "items_prompts", ["item_id"], :name =>
"index_items_prompts_on_item_id"
  add_index "items_prompts", ["prompt_id"], :name =>
"index_items_prompts_on_prompt_id"

  create_table "items_questions", :force => true do |t|
    t.integer  "item_id",                        :null => false
```

```
    t.integer  "question_id",                   :null => false
    t.integer  "wins",        :default => 0,    :null => false
    t.integer  "ratings",     :default => 0,    :null => false
    t.integer  "position",    :default => 1400, :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
    t.integer  "losses",      :default => 0
  end

  add_index "items_questions", ["item_id"], :name =>
"index_items_questions_on_item_id"
  add_index "items_questions", ["question_id"], :name =>
"index_items_questions_on_question_id"

  create_table "items_stats", :id => false, :force => true do |t|
    t.integer "item_id"
    t.integer "stat_id"
  end

  add_index "items_stats", ["item_id"], :name => "index_items_stats_on_item_id"
  add_index "items_stats", ["stat_id"], :name => "index_items_stats_on_stat_id"

  create_table "items_votes", :id => false, :force => true do |t|
    t.integer "item_id", :null => false
    t.integer "vote_id", :null => false
  end

  add_index "items_votes", ["item_id"], :name => "index_items_votes_on_item_id"
  add_index "items_votes", ["vote_id"], :name => "index_items_votes_on_vote_id"

  create_table "prompt_algorithms", :force => true do |t|
    t.string   "name",       :null => false
    t.string   "data",       :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "prompt_requests", :force => true do |t|
    t.integer  "question_id",                :null => false
    t.integer  "voter_id",                   :null => false
    t.integer  "count",       :default => 1
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  add_index "prompt_requests", ["question_id"], :name =>
"index_prompt_requests_on_question_id"
  add_index "prompt_requests", ["voter_id"], :name =>
"index_prompt_requests_on_voter_id"

  create_table "prompts", :force => true do |t|
    t.integer  "question_id",                          :null => false
    t.integer  "prompt_algorithm_id",                  :null => false
    t.integer  "voter_id",                             :null => false
    t.boolean  "active",             :default => true
```

```
  t.datetime "created_at"
  t.datetime "updated_at"
end

add_index "prompts", ["question_id"], :name => "index_prompts_on_question_id"
add_index "prompts", ["voter_id"], :name => "index_prompts_on_voter_id"

create_table "questions", :force => true do |t|
  t.integer  "user_id",    :null => false
  t.string   "name",       :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end

add_index "questions", ["user_id"], :name => "index_questions_on_user_id"

create_table "rank_algorithms", :force => true do |t|
  t.string   "name",       :null => false
  t.string   "data",       :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end

create_table "sessions", :force => true do |t|
  t.string   "session_id", :null => false
  t.text     "data"
  t.datetime "created_at"
  t.datetime "updated_at"
end

add_index "sessions", ["session_id"], :name => "index_sessions_on_session_id"
add_index "sessions", ["updated_at"], :name => "index_sessions_on_updated_at"

create_table "stats", :force => true do |t|
  t.integer  "question_id",                      :null => false
  t.integer  "rank_algorithm_id"
  t.integer  "views",             :default => 0,   :null => false
  t.integer  "votes",             :default => 0,   :null => false
  t.float    "score",             :default => 0.0, :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end

add_index "stats", ["question_id"], :name => "index_stats_on_question_id"

create_table "users", :force => true do |t|
  t.string   "login",                     :limit => 40
  t.string   "name",                      :limit => 100, :default => ""
  t.string   "email",                     :limit => 100
  t.string   "crypted_password",          :limit => 40
  t.string   "salt",                      :limit => 40
  t.datetime "created_at"
  t.datetime "updated_at"
  t.string   "remember_token",            :limit => 40
  t.datetime "remember_token_expires_at"
```

```
    t.string   "activation_code",           :limit => 40
    t.datetime "activated_at"
    t.string   "state",                                  :default => "passive"
    t.datetime "deleted_at"
  end

  add_index "users", ["login"], :name => "index_users_on_login", :unique => true

  create_table "voters", :force => true do |t|
    t.integer  "user_id",    :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  add_index "voters", ["user_id"], :name => "index_voters_on_user_id"

  create_table "votes", :force => true do |t|
    t.integer  "prompt_id",     :null => false
    t.integer  "voter_id",      :null => false
    t.integer  "response_time"
    t.datetime "created_at"
    t.datetime "updated_at"
    t.text     "tracking"
  end

  add_index "votes", ["prompt_id"], :name => "index_votes_on_prompt_id"
  add_index "votes", ["voter_id"], :name => "index_votes_on_voter_id"

end
module Algorithms::Prompt::Popular
  class << self
    include Algorithms::Prompt

    # algorithm ID
    ID = 3

    # Generate count number of primed prompts based on question ID, voter ID
    # and stats.
    # ==== Return
    # Hash with prompt IDs as keys and item IDs for that prompt as values.
    # ==== Parameters
    # question_id<int>:: Generate prompts for this question ID
    # voter_id<int>:: Generate prompts for this voter ID
    # count<int>:: Generate this number of prompts
    # result<Mysql::Result>:: Stats values.
    def prompts(question_id, voter_id, count, result)
      # make all stats positive by adding |min(stats)| + 1 to all stats
      prompt_item_ids = {}
      norm = cur = 0
      stats = []
      stat = result.fetch_hash
      min = stat['score'].to_f.abs + 1
      while !stat.nil?
        norm += (adj = stat['score'].to_f + min)
        stats << [stat['id'].to_i, [cur, cur += adj]]
```

```
        stat = result.fetch_hash
      end
      result.free
      Prompt.transaction do
        count.times do |i|
          prompt = prompt_for_request(question_id, voter_id, ID)
          # choose the stat [0] <= r < [1]
          r = rand(norm)
          # detect treats hash as [key, value] array
          stat_id = stats.detect { |stat| stat[1][0] <= r && r < stat[1][1] }[0]
          item_ids = Stat.find(stat_id).item_ids
          prompt.item_ids = item_ids
          redo if bad_prompt?(prompt)
          prompt_item_ids[prompt.id] = item_ids
        end
      end
      prompt_item_ids
    end
  end
end
    module Algorithms::Prompt::Random
  class << self
    include Algorithms::Prompt

    # algorithm ID
    ID = 2

    # Generate count number of random prompts based on question ID and voter ID.
    # ==== Return
    # An array of prompts
    # ==== Parameters
    # question_id<int>:: Generate prompts for this question ID
    # voter_id<int>:: Generate prompts for this voter ID
    # count<int>:: Generate this number of prompts
    def prompts(question_id, voter_id, count)
      all_items = items_for_request(question_id)
      items = []
      prompt_item_ids = {}
      # leaks with create/find in req loop
      Prompt.transaction do
        count.times do |i|
          items = all_items.dup if items.length < 2 # ensure we still have items
          prompt = prompt_for_request(question_id, voter_id, ID)
          item = items.delete_at(rand(items.length))
          prompt.items << item << items.delete_at(rand(items.length))
          redo if bad_prompt?(prompt)
          prompt_item_ids[prompt.id] = prompt.item_ids
        end
      end
      prompt_item_ids
    end
  end
endmodule Algorithms::Prompt
  ID = 1
```

```
  # Generate count number of prompts based on question and voter. Choose
  # generation algorithm based on prime.
  # ==== Return
  # An array of prompts
  # ==== Parameters
  # question_id<int>:: Prompts for this question ID.
  # voter_id<int>:: Prompts for this voter ID.
  # count<int>:: This number of prompts.
  # prompt<bool>:: If true primed prompts are generated, otherwise random
  # prompts are generated.
  def create_prompts(question_id, voter_id, count, prime)
    srand if PRODUCTION
    result = ActiveRecord::Base.connection.execute(
      "SELECT id,score FROM stats WHERE question_id=#{question_id} ORDER BY
score;"
    ) if prime
    if prime && result.num_rows > 0
      Algorithms::Prompt::Popular.prompts(question_id, voter_id, count, result)
    else
      Algorithms::Prompt::Random.prompts(question_id, voter_id, count)
    end
  end

  # Create default prompt
  def prompt_for_request(question_id, voter_id, algorithm_id)
    Prompt.create(
      :question_id => question_id,
      :voter_id => voter_id,
      :prompt_algorithm_id => algorithm_id,
      :active => false
    )
  end

  # Test if prompt is bad.  Prompt is bad if it does not have 2 items or if
  # any of its items are nil.
  def bad_prompt?(prompt)
    return (prompt.items.length != 2 || prompt.items.any?(&:nil?))
  end

  def conditions(question_id)
    {
      :joins => "INNER JOIN items_questions ON
(items_questions.question_id=#{question_id} AND
items_questions.item_id=items.id)",
      :conditions => { :active => true },
    }
  end

  private
    def items_for_request(question_id)
      Item.all(conditions(question_id).merge(:group => "items.id"))
    end
endmodule Algorithms::Rank::Elo
  START_RATING = 1400
  K_VALUE = 32
```

```
  DRAW_SCORE = 0.5
  LOSS_SCORE = 0
  WIN_SCORE = 1

  class << self

    # Calculate Elo score for all items given a user and question.
    # ==== Return
    # Array of item IDs, followed by an array of Elo scores matching to the
    # item with the same array index.
    # ==== Parameters
    # user_id<int>:: Get Elo scores for user with this ID.
    # order<string>:: If 'asc' return Elo scores in ascending order, otherwise
    # return Elo scores in descending order.
    # question_id<int>:: Get Elo scores for items in this question.
    # limit<int>:: Truncate scores returned to this number of items.
    # adj<bool>:: Use an adjusted Elo algorithm. Default true.
    def score(user_id, order, question_id, limit, adj=true)
      joins = "INNER JOIN users ON items.user_id=#{user_id}"
      joins += " INNER JOIN items_questions ON
(items_questions.question_id=#{question_id} AND
items_questions.item_id=items.id)" if question_id.to_i > 0
      conditions = {
        :joins => joins,
        :group => 'items.id'
      }
      items = Item.find(:all, conditions)

      conditions = {
        :conditions => { :active => false },
        :order => 'prompts.created_at',
        :include => [:items, :votes],
        :group => 'prompts.id'
      }
      conditions[:conditions].merge!(:question_id => question_id) if question_id
> 0
      prompts = Prompt.all(conditions)

      elo = {}
      items.each { |item| elo[item] = START_RATING }
      @adj = adj
      prompts.each do |prompt|
        prompt.votes.each do |vote|
          # elo values static during update
          old_elo = elo.clone
          # if vote has items it has winner(s)
          unless vote.items.empty?
            vote.items.each do |item|
              lost_items = prompt.items - vote.items
              lost_items.each do |loser|
                elo[item] += adjust_elo(WIN_SCORE, old_elo[item], elo[loser])
                elo[loser] += adjust_elo(LOSS_SCORE, elo[loser], old_elo[item])
              end
            end
          else
```

```
            # otherwise consider as draw between all prompt items
            prompt.items.each do |item|
              (prompt.items - [item]).each do |other|
                elo[item] += adjust_elo(DRAW_SCORE, elo[item], old_elo[other])
              end
            end # item_ids each
          end # else
        end # votes each
      end # prompts each
      elo = (order == "asc") ? elo.sort_by { |k, v| v } : elo.sort_by { |k, v| -
v }
      elo = elo.first(limit) if limit > 0
      elo.transpose
    end

    # This value is added to the previous Elo score to obtain the current score.
    # ==== Return
    # Integer which is the amount by which to adjust the Elo score
    # ==== Parameters
    # score<int>:: The adjustment value for a win, loss, or draw.
    # r_A<int>:: The Elo score of the item whose score to be adjusted.
    # r_B<int>:: The Elo score of the item whhose score is not being adjusted.
    def adjust_elo(score, r_A, r_B)
      k_value(r_A) * (score - expected_score(r_A, r_B))
    end

    # The score if A plays B.
    # ==== Return
    # Integer of the expected score.
    # ==== Parameters
    # r_A<int>:: The Elo score of item A.
    # r_B<int>:: The Elo score of item B.
    def expected_score(r_A, r_B)
      1 / (1 + 10**((r_B - r_A) / 400.0))
    end

    # Calculate the K value for an item based on if an adjusted algorithm is
    # being used and the score of the item
    # ==== Return
    # The K value for an item.
    # ==== Parameters
    # r_A<int>:: The score of the item whose K value is to be generated.
    def k_value(r_A)
      if !@adj || r_A < 2100
        K_VALUE
      elsif r_A > 2400
        16
      else
        24
      end
    end
  end
endmodule AuthenticatedSystem
  protected
    # Returns true or false if the user is logged in.
```

```
    # Preloads @current_user with the user model if they're logged in.
    def logged_in?
      !!current_user
    end

    # Accesses the current user from the session.
    # Future calls avoid the database because nil is not equal to false.
    def current_user
      @current_user ||= (login_from_session || login_from_basic_auth ||
login_from_cookie) unless @current_user == false
    end

    # Store the given user id in the session.
    def current_user=(new_user)
      session[:user_id] = new_user ? new_user.id : nil
      @current_user = new_user || false
    end

    # Check if the user is authorized
    #
    # Override this method in your controllers if you want to restrict access
    # to only a few actions or if you want to check if the user
    # has the correct rights.
    #
    # Example:
    #
    #  # only allow nonbobs
    #  def authorized?
    #    current_user.login != "bob"
    #  end
    #
    def authorized?(action = action_name, resource = nil)
      logged_in?
    end

    # Filter method to enforce a login requirement.
    #
    # To require logins for all actions, use this in your controllers:
    #
    #   before_filter :login_required
    #
    # To require logins for specific actions, use this in your controllers:
    #
    #   before_filter :login_required, :only => [ :edit, :update ]
    #
    # To skip this in a subclassed controller:
    #
    #   skip_before_filter :login_required
    #
    def login_required
      authorized? || access_denied
    end

    # Redirect as appropriate when an access request fails.
    #
```

```
    # The default action is to redirect to the login screen.
    #
    # Override this method in your controllers if you want to have special
    # behavior in case the user is not authorized
    # to access the requested action.  For example, a popup window might
    # simply close itself.
    def access_denied
      respond_to do |format|
        format.html do
          store_location
          redirect_to new_session_path
        end
        # format.any doesn't work in rails version <
http://dev.rubyonrails.org/changeset/8987
        # Add any other API formats here.  (Some browsers, notably IE6, send
Accept: */* and trigger
        # the 'format.any' block incorrectly. See http://bit.ly/ie6_borken or
http://bit.ly/ie6_borken2
        # for a workaround.)
        format.any(:json, :xml) do
          request_http_basic_authentication 'Web Password'
        end
      end
    end

    # Store the URI of the current request in the session.
    #
    # We can return to this location by calling #redirect_back_or_default.
    def store_location
      session[:return_to] = request.request_uri
    end

    # Redirect to the URI stored by the most recent store_location call or
    # to the passed default.  Set an appropriately modified
    #   after_filter :store_location, :only => [:index, :new, :show, :edit]
    # for any controller you want to be bounce-backable.
    def redirect_back_or_default(default)
      redirect_to(session[:return_to] || default)
      session[:return_to] = nil
    end

    # Inclusion hook to make #current_user and #logged_in?
    # available as ActionView helper methods.
    def self.included(base)
      base.send :helper_method, :current_user, :logged_in?, :authorized? if
base.respond_to? :helper_method
    end

    #
    # Login
    #

    # Called from #current_user.  First attempt to login by the user id stored
in the session.
    def login_from_session
```

```
      self.current_user = User.find_by_id(session[:user_id]) if
session[:user_id]
    end

    # Called from #current_user.  Now, attempt to login by basic authentication
information.
    def login_from_basic_auth
      authenticate_with_http_basic do |login, password|
        self.current_user = User.authenticate(login, password)
      end
    end

    #
    # Logout
    #

    # Called from #current_user.  Finaly, attempt to login by an expiring token
in the cookie.
    # for the paranoid: we _should_ be storing user_token = hash(cookie_token,
request IP)
    def login_from_cookie
      user = cookies[:auth_token] &&
User.find_by_remember_token(cookies[:auth_token])
      if user && user.remember_token?
        self.current_user = user
        handle_remember_cookie! false # freshen cookie token (keeping date)
        self.current_user
      end
    end

    # This is ususally what you want; resetting the session willy-nilly wreaks
    # havoc with forgery protection, and is only strictly necessary on login.
    # However, **all session state variables should be unset here**.
    def logout_keeping_session!
      # Kill server-side auth cookie
      @current_user.forget_me if @current_user.is_a? User
      @current_user = false     # not logged in, and don't do it for me
      kill_remember_cookie!     # Kill client-side auth cookie
      session[:user_id] = nil   # keeps the session but kill our variable
      # explicitly kill any other session variables you set
    end

    # The session should only be reset at the tail end of a form POST --
    # otherwise the request forgery protection fails. It's only really necessary
    # when you cross quarantine (logged-out to logged-in).
    def logout_killing_session!
      logout_keeping_session!
      reset_session
    end

    #
    # Remember_me Tokens
    #
    # Cookies shouldn't be allowed to persist past their freshness date,
    # and they should be changed at each login
```

```
    # Cookies shouldn't be allowed to persist past their freshness date,
    # and they should be changed at each login

    def valid_remember_cookie?
      return nil unless @current_user
      (@current_user.remember_token?) &&
        (cookies[:auth_token] == @current_user.remember_token)
    end

    # Refresh the cookie auth token if it exists, create it otherwise
    def handle_remember_cookie!(new_cookie_flag)
      return unless @current_user
      case
      when valid_remember_cookie? then @current_user.refresh_token # keeping
same expiry date
      when new_cookie_flag        then @current_user.remember_me
      else                             @current_user.forget_me
      end
      send_remember_cookie!
    end

    def kill_remember_cookie!
      cookies.delete :auth_token
    end

    def send_remember_cookie!
      cookies[:auth_token] = {
        :value   => @current_user.remember_token,
        :expires => @current_user.remember_token_expires_at }
    end

end
module AuthenticatedTestHelper
  # Sets the current user in the session from the user fixtures.
  def login_as(user)
    @request.session[:user_id] = user ? users(user).id : nil
  end

  def authorize_as(user)
    @request.env["HTTP_AUTHORIZATION"] = user ?
ActionController::HttpAuthentication::Basic.encode_credentials(users(user).login
, 'monkey') : nil
  end

  # rspec
  def mock_user
    user = mock_model(User, :id => 1,
      :login  => 'user_name',
      :name   => 'U. Surname',
      :to_xml => "User-in-XML", :to_json => "User-in-JSON",
      :errors => [])
    user
  end
end
```

```
module Constants
  CAN_SELECT_PROMPT_ALGORITHM = false
  DEFAULT_PROMPT_ALGORITHM_ID = 1
  MAX_BATCH_PROMPTS = 100
  BASE_URL = "http://pairwise.photocracy.org/"
  ADMIN_EMAILS = "peter@photocracy.org"
  USER_KEY = 'cetuwes9rewruRec6k7hUz4CRabaD7musa3es9muzeda5'

  module Stat
    DEFAULT_RANK_ALGO = 2
  end
endrequire 'logger'
require 'English'
# Jan  2 03:38:05 topfunky postfix/postqueue[2947]: warning blah blah blah

##
# A logger for use with pl_analyze and other tools that expect syslog-style log
output.

class Hodel3000CompliantLogger < Logger

  ##
  # Note: If you are using FastCGI you may need to hard-code the hostname here
instead of using Socket.gethostname

  def format_message(severity, timestamp, progname, msg)
    "#{timestamp.strftime("%b %d %H:%M:%S")} #{hostname} rails[#{$PID}]:
#{msg2str(msg).gsub(/\n/, '').lstrip}\n"
  end

  # original method, pre-patch for Exception handling:
  #
  # def format_message(severity, timestamp, msg, progname)
  #   "#{timestamp.strftime("%b %d %H:%M:%S")}
#{Socket.gethostname.split('.').first} rails[#{$PID}]: #{progname.gsub(/\n/,
'').lstrip}\n"
  # end

  private

  def hostname
    @parsed_hostname ||= Socket.gethostname.split('.').first
  end

  def msg2str(msg)
    case msg
    when ::String
      msg
    when ::Exception
      "#{ msg.message } (#{ msg.class }): " <<
      (msg.backtrace || []).join(" | ")
    else
      msg.inspect
    end
  end
```

```
end
#!/opt/local/bin/ruby

require File.dirname(__FILE__) + "/../config/environment" unless
defined?(RAILS_ROOT)

# If you're using RubyGems and mod_ruby, this require should be changed to an
absolute path one, like:
# "/usr/local/lib/ruby/gems/1.8/gems/rails-0.8.0/lib/dispatcher" -- otherwise
performance is severely impaired
require "dispatcher"

ADDITIONAL_LOAD_PATHS.reverse.each { |dir| $:.unshift(dir) if
File.directory?(dir) } if defined?(Apache::RubyRun)
Dispatcher.dispatchall_wins = Prompt.all(
  :select => "items_votes.item_id, prompts.question_id, COUNT(*) AS count",
  :joins => "INNER JOIN votes ON (votes.prompt_id=prompts.id) INNER JOIN
items_votes ON (votes.id=items_votes.vote_id)",
  :group => "items_votes.item_id, prompts.question_id"
)
all_ratings = Prompt.all(
  :select => "items_prompts.item_id, prompts.question_id, COUNT(*) AS count",
  :joins => "INNER JOIN items_prompts ON (prompts.id=items_prompts.prompt_id)
INNER JOIN votes ON (votes.prompt_id=prompts.id)",
  :group => "items_prompts.item_id, prompts.question_id"
)
all_ratings_no_skips = Prompt.all(
  :select => "items_prompts.item_id, prompts.question_id, COUNT(*) AS count",
  :joins => "INNER JOIN items_prompts ON (prompts.id=items_prompts.prompt_id)
INNER JOIN votes ON (votes.prompt_id=prompts.id) INNER JOIN items_votes ON
(votes.id=items_votes.vote_id)",
  :group => "items_prompts.item_id, prompts.question_id"
)
ItemsQuestion.transaction do
  for iq in ItemsQuestion.all
    wins = all_wins.find { |w| w.item_id.to_i == iq.item_id &&
w.question_id.to_i == iq.question_id }
    ratings = all_ratings.find { |w| w.item_id.to_i == iq.item_id &&
w.question_id.to_i == iq.question_id }
    ratings_no_skips = all_ratings_no_skips.find { |w| w.item_id.to_i ==
iq.item_id && w.question_id.to_i == iq.question_id }
    if ratings
      wins = wins ? wins.count.to_i : 0
      ratings_no_skips = ratings_no_skips ? ratings_no_skips.count.to_i : 0
#      puts "r:#{ratings.count} w:#{wins} l:#{ratings_no_skips - wins}"
      iq.update_attributes(
        :ratings => ratings.count,
        :wins => wins,
        :losses => ratings_no_skips - wins
      )
    end
  end
endrequire File.dirname(__FILE__) + '/../spec_helper'
  # Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead
```

```
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper

#
# A test controller with and without access controls
#
class AccessControlTestController < ApplicationController
  before_filter :login_required, :only => :login_is_required
  def login_is_required
    respond_to do |format|
      @foo = { 'success' => params[:format]||'no fmt given'}
      format.html do render :text => "success"             end
      format.xml  do render :xml  => @foo, :status => :ok  end
      format.json do render :json => @foo, :status => :ok  end
    end
  end
  def login_not_required
    respond_to do |format|
      @foo = { 'success' => params[:format]||'no fmt given'}
      format.html do render :text => "success"             end
      format.xml  do render :xml  => @foo, :status => :ok  end
      format.json do render :json => @foo, :status => :ok  end
    end
  end
end

#
# Access Control
#

ACCESS_CONTROL_FORMATS = [
  ['',     "success"],
  ['xml',  "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<hash>\n
<success>xml</success>\n</hash>\n"],
  ['json', "{\"success\": \"json\"}"],]
ACCESS_CONTROL_AM_I_LOGGED_IN = [
  [:i_am_logged_in,     :quentin],
  [:i_am_not_logged_in, nil],]
ACCESS_CONTROL_IS_LOGIN_REQD = [
  :login_not_required,
  :login_is_required,]

describe AccessControlTestController do
  fixtures        :users
  before do
    # is there a better way to do this?
    ActionController::Routing::Routes.add_route '/login_is_required',
:controller => 'access_control_test',   :action => 'login_is_required'
    ActionController::Routing::Routes.add_route '/login_not_required',
:controller => 'access_control_test',   :action => 'login_not_required'
  end

  ACCESS_CONTROL_FORMATS.each do |format, success_text|
    ACCESS_CONTROL_AM_I_LOGGED_IN.each do |logged_in_status, user_login|
      ACCESS_CONTROL_IS_LOGIN_REQD.each do |login_reqd_status|
```

```
        describe "requesting #{format.blank? ? 'html' : format};
#{logged_in_status.to_s.humanize} and #{login_reqd_status.to_s.humanize}" do
          before do
            logout_keeping_session!
            @user = format.blank? ? login_as(user_login) :
authorize_as(user_login)
            get login_reqd_status.to_s, :format => format
          end

          if ((login_reqd_status == :login_not_required) ||
              (login_reqd_status == :login_is_required && logged_in_status ==
:i_am_logged_in))
            it "succeeds" do
              response.should have_text(success_text)
              response.code.to_s.should == '200'
            end

          elsif (login_reqd_status == :login_is_required && logged_in_status ==
:i_am_not_logged_in)
            if ['html', ''].include? format
              it "redirects me to the log in page" do
                response.should redirect_to('/session/new')
              end
            else
              it "returns 'Access denied' and a 406 (Access Denied) status code"
do
                response.should have_text("HTTP Basic: Access denied.\n")
                response.code.to_s.should == '401'
              end
            end

          else
            warn "Oops no case for #{format} and
#{logged_in_status.to_s.humanize} and #{login_reqd_status.to_s.humanize}"
          end
        end # describe

      end
    end
  end # cases

end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe ApplicationController do
  describe 'force xml' do
    before do
      allow_message_expectations_on_nil
    end

    it 'should merge xml format into params' do

@controller.params.should_receive(:include?).with(:format).and_return(false)
      @controller.params.should_receive(:merge!).with(:format => :xml)
      @controller.force_xml
```

```
    end

    it 'should not merge xml format into params with format' do

@controller.params.should_receive(:include?).with(:format).and_return(true)
      @controller.params.should_not_receive(:merge!)
      @controller.force_xml
    end
  end

  describe 'admin required' do
    fixtures :users

    it 'should redirect if not logged in' do

@controller.should_receive(:redirect_back_or_default).with('/').and_return(true)
      @controller.current_user = false
      @controller.admin_required
    end

    describe 'when logged in' do
      it 'should redirect if not admin' do
        login_as :quentin

@controller.should_receive(:redirect_back_or_default).with('/').and_return(true)
        @controller.admin_required
      end

      it 'should not redirect if admin' do
        login_as :smurf
        @controller.should_not_receive(:redirect_back_or_default)
      end
    end
  end
end
require File.dirname(__FILE__) + '/../spec_helper'

# Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper
include AuthenticatedSystem
def action_name() end

describe SessionsController do
  fixtures :users

  before do
    # FIXME -- sessions controller not testing xml logins
    stub!(:authenticate_with_http_basic).and_return nil
  end
  describe "logout_killing_session!" do
    before do
      login_as :quentin
      stub!(:reset_session)
    end
```

```
    it 'resets the session'         do should_receive(:reset_session);
logout_killing_session! end
    it 'kills my auth_token cookie' do should_receive(:kill_remember_cookie!);
logout_killing_session! end
    it 'nils the current user'      do logout_killing_session!;
current_user.should be_nil end
    it 'kills :user_id session' do
      session.stub!(:[]=)
      session.should_receive(:[]=).with(:user_id, nil).at_least(:once)
      logout_killing_session!
    end
    it 'forgets me' do
      current_user.remember_me
      current_user.remember_token.should_not be_nil;
current_user.remember_token_expires_at.should_not be_nil
      User.find(1).remember_token.should_not be_nil;
User.find(1).remember_token_expires_at.should_not be_nil
      logout_killing_session!
      User.find(1).remember_token.should     be_nil;
User.find(1).remember_token_expires_at.should     be_nil
    end
  end

  describe "logout_keeping_session!" do
    before do
      login_as :quentin
      stub!(:reset_session)
    end
    it 'does not reset the session' do should_not_receive(:reset_session);
logout_keeping_session! end
    it 'kills my auth_token cookie' do should_receive(:kill_remember_cookie!);
logout_keeping_session! end
    it 'nils the current user'      do logout_keeping_session!;
current_user.should be_nil end
    it 'kills :user_id session' do
      session.stub!(:[]=)
      session.should_receive(:[]=).with(:user_id, nil).at_least(:once)
      logout_keeping_session!
    end
    it 'forgets me' do
      current_user.remember_me
      current_user.remember_token.should_not be_nil;
current_user.remember_token_expires_at.should_not be_nil
      User.find(1).remember_token.should_not be_nil;
User.find(1).remember_token_expires_at.should_not be_nil
      logout_keeping_session!
      User.find(1).remember_token.should     be_nil;
User.find(1).remember_token_expires_at.should     be_nil
    end
  end

  describe 'When logged out' do
    it "should not be authorized?" do
      authorized?().should be_false
    end
```

```
  end

  #
  # Cookie Login
  #
  describe "Logging in by cookie" do
    def set_remember_token token, time
      @user[:remember_token]            = token;
      @user[:remember_token_expires_at] = time
      @user.save!
    end
    before do
      @user = User.find(:first);
      set_remember_token 'hello!', 5.minutes.from_now
    end
    it 'logs in with cookie' do
      stub!(:cookies).and_return({ :auth_token => 'hello!' })
      logged_in?.should be_true
    end

    it 'fails cookie login with bad cookie' do
      should_receive(:cookies).at_least(:once).and_return({ :auth_token =>
'i_haxxor_joo' })
      logged_in?.should_not be_true
    end

    it 'fails cookie login with no cookie' do
      set_remember_token nil, nil
      should_receive(:cookies).at_least(:once).and_return({ })
      logged_in?.should_not be_true
    end

    it 'fails expired cookie login' do
      set_remember_token 'hello!', 5.minutes.ago
      stub!(:cookies).and_return({ :auth_token => 'hello!' })
      logged_in?.should_not be_true
    end
  end

end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe HomeController do
  fixtures :users, :items, :questions, :items_questions, :rank_algorithms,
:voters, :features, :prompt_requests, :prompts, :stats, :items_stats

  before(:each) do
    @user_id = 1
    login_as('quentin')
  end

  describe 'delete' do
    describe 'get' do
      it 'should require post' do
        # get :delete
```

```
        # flash[:message].should == nil
      end

      it 'should redirect' do
        get :delete
        response.should be_redirect
      end
    end

    describe 'post' do
      before do
        @user = User.find(@user_id)
        @voters = @user.voters
        @items = @user.items
        @questions = @user.questions
        @features = @voters.map(&:features).flatten
        @prompt_requests = @questions.map(&:prompt_requests).flatten |
@items.map(&:prompt_requests).flatten
        @stats = @items.map(&:stats).flatten
        @iqs = @questions.map(&:items_questions).flatten |
@items.map(&:items_questions).flatten
        @prompts = @items.map(&:prompts).flatten
        @votes = @items.map(&:votes).flatten | @prompts.map(&:vote).flatten
        post :delete
      end

      it 'should set flash' do
        # flash[:message].should_not == nil
      end

      it 'should redirect' do
        response.should be_redirect
      end

#      it 'should delete voters' do
#        Voter.all(:conditions => { :id => @voters.map(&:id) }).empty?.should ==
true
#        @user.reload.voters.empty?.should == true
#      end

      it 'should delete items' do
        Item.all(:conditions => { :id => @items.map(&:id) }).empty?.should ==
true
        @user.reload.items.empty?.should == true
      end

      it 'should delete questions' do
        Question.all(:conditions => { :id => @questions.map(&:id)
}).empty?.should == true
        @user.reload.questions.empty?.should == true
      end

#      it 'should delete features' do
#        Feature.all(:conditions => { :id => @features.map(&:id)
}).empty?.should == true
```

```
#      end

      it 'should delete prompt_requests' do
        PromptRequest.all(:conditions => { :id => @prompt_requests.map(&:id)
}).empty?.should == true
      end

      it 'should delete items questions' do
        ItemsQuestion.all(:conditions => { :id => @iqs.map(&:id)
}).empty?.should == true
      end

      it 'should delete votes' do
        Vote.all(:conditions => { :id => @votes.map(&:id) }).empty?.should ==
true
      end

      it 'should delete prompts' do
        Prompt.all(:conditions => { :id => @prompts.map(&:id) }).empty?.should
== true
      end
    end
  end
endrequire File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe ItemsController do
  include Algorithms::Rank::Elo
  fixtures :users, :items, :questions, :items_questions, :rank_algorithms,
:votes, :items_votes

  def mock_item(stubs={})
    @mock_item ||= mock_model(Item, stubs)
  end

  before(:each) do
    @user_id = 1
    login_as('quentin')
  end

  def all_user_items
    Item.find(:all, :conditions => { :user_id => @user_id }, :order =>
'created_at asc')
  end

  def items_by_rank_algo(user_id, order, question_id = 0, limit = 0)
    Algorithms::Rank::Elo.score(user_id, order, question_id, limit).first
  end

  describe "responding to GET list" do
    it "should expose all user items as @items" do
      get :list
      assigns[:items].should_not == Item.find(:all)
      assigns[:items].each do |item|
        item.user_id.should == @user_id
      end
```

```
    end

    it "should limit items to n" do
      get :list, :limit => 2
      assigns[:items].should == Item.find(:all, :conditions => { :user_id =>
@user_id }, :order => "items.created_at desc")[0,2]
    end

    it "should order items by order" do
      get :list, :order => 'asc'
      assigns[:items].should == all_user_items
    end

    it "should order items by order and limit" do
      get :list, :order => 'asc', :limit => 2
      assigns[:items].should == all_user_items[0,2]
    end

    it "should get items for question" do
      get :list, :question_id => 1
      assigns[:items].each do |item|
        item.questions.first.id.should == 1
      end
    end

    it "should order by rank algo passing id" do
      get :list, :rank_algorithm => 1
      assigns[:items].should == Item.all(
        :conditions => { :user_id => current_user.id },
        :order => 'items_questions.position desc',
        :include => 'items_questions'
      )
    end

    it "should order by rank algo passing name" do
      get :list, :rank_algorithm => 'elo'
      assigns[:items].should == Item.all(
        :conditions => { :user_id => current_user.id },
        :order => 'items_questions.position desc',
        :include => 'items_questions'
      )
    end

    it 'should order by rank algo asc' do
      get :list, :rank_algorithm => 1, :order => 'ASC'
      assigns[:items].should == Item.all(
        :conditions => { :user_id => current_user.id },
        :order => 'items_questions.position asc',
        :include => 'items_questions'
      )
    end

    it 'should order by rank algo ewp and limit' do
      get :list, :rank_algorithm => 'ewp', :order => 'ASC', :limit => 5
      assigns[:items].length.should == 5
```

```
  end
end

describe "responding to POST add" do
  it "should return unless post" do
    get :add
    assigns[:items].should == nil
  end

  it "should create items" do
    xml_post :add, 'items_good'
    item = Item.find_by_data('new')
    item2 = Item.find_by_data('new2')
    assigns[:items].should == [ item, item2 ]
    assigns[:items].first.questions.first.id.should == 1
    assigns[:items].first.questions.last.id.should == 2
  end

  it "should store tracking with items if passed" do
    track = "track"
    xml_post :add, 'items_good', :tracking => track
    for item in assigns[:items]
      item.tracking.should == track
    end
  end

  it "should error if invalid question_id" do
    xml_post :add, 'items_bad'
    response.status.should =~ /400/
  end
end

describe "responding to GET activate" do
  it "should activate" do
    get :activate, :id => 1
    assigns[:item].should == Item.find(1)
    assigns[:item].active.should == true
  end

  it "should not activate not owned items" do
    get :activate, :id => 11
    response.status.should =~ /400/
  end
end

describe "responding to GET suspend" do
  it "should suspend" do
    get :suspend, :id => 1
    assigns[:item].should == Item.find(1)
    assigns[:item].active.should == false
  end

  it "should not suspend not owned items" do
    get :suspend, :id => 11
    response.status.should =~ /400/
```

```
    end
  end

  describe 'responding to GET show' do
    it 'should expose item' do
      get :show, :id => 1
      assigns[:item].should == Item.find(1)
    end

    it 'should not expose non-user items' do
      get :show, :id => 11
      assigns[:item].should == nil
    end
  end

  describe 'responding to POST delete' do
    fixtures :prompt_requests, :stats, :items_stats

    before do
      @item = Item.first(:conditions => { :user_id => @user_id })
      @iqs = @item.items_questions
      @prompts = @item.prompts
      @stats = @item.stats
      @votes = @item.votes
      @prs = @item.prompt_requests
    end

    it 'should return if not post' do
      get :delete
      assigns[:success].should == nil
    end

    it 'should delete item' do
      post :delete, :id => @item.id
      Item.exists?(@item.id).should == false
    end

    it 'should delete item questions' do
      post :delete, :id => @item.id
      ItemsQuestion.all(:conditions => { :id => @iqs.map(&:id) }).empty?.should
== true
    end

    it 'should delete prompts' do
      post :delete, :id => @item.id
      Prompt.all(:conditions => { :id => @prompts.map(&:id) }).empty?.should ==
true
    end

    it 'should delete votes' do
      post :delete, :id => @item.id
      Vote.all(:conditions => { :id => @votes.map(&:id) }).empty?.should == true
    end

    it 'should delete stats' do
```

```
      post :delete, :id => @item.id
      Stat.all(:conditions => { :id => @stats.map(&:id) }).empty?.should == true
    end

    it 'should delete prompt requests' do
      post :delete, :id => @item.id
      PromptRequest.all(:conditions => { :id => @prs.map(&:id) }).empty?.should
== true
    end


    it 'should assign id' do
      post :delete, :id => @item.id
      assigns[:id].should == @item.id.to_s
    end

    it 'should assign success true' do
      post :delete, :id => @item.id
      assigns[:success].should == true
    end

    it 'should assign success false' do
      post :delete, :id => @item.id
      post :delete, :id => @item.id
      response.status.should =~ /400/
    end
  end
endrequire File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe PromptAlgorithmsController do
  fixtures :users, :prompt_algorithms

  before(:each) do
    login_as('quentin')
  end

  describe "responding to GET list" do
    it "should list rank algos" do
      get :list
      assigns[:algorithms].should == PromptAlgorithm.find(:all)
    end
  end
endrequire File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe PromptsController do
  fixtures :users, :questions, :prompt_algorithms, :items, :items_questions,
:voters

  before(:each) do
    login_as('quentin')
  end

  describe "responding to GET create" do
    it "should not create prompts without question_id" do
      get :create
```

```
      response.status.should =~ /417/
    end

    it "should not create prompts if user doesn't own voter" do
      get :create, :question_id => 1, :voter_id => 2
      response.status.should =~ /417/
    end

    it "should not create prompts if user doesn't own question_id" do
      get :create, :question_id => 3, :voter_id => 1
      response.status.should =~ /417/
    end

    it "should create prompts for question and voter" do
      get :create, :question_id => 1, :voter_id => 1
      assigns[:prompt_item_ids].should_not == nil
      assigns[:prompt_item_ids].should_not == {}
      prompt = Prompt.find(assigns[:prompt_item_ids].keys.first)
      prompt.question.id.should == 1
      prompt.voter_id.should == 1
      prompt.reload.active.should == false
    end

    it "should create prompts without voter_id" do
      get :create, :question_id => 1
      Prompt.find(assigns[:prompt_item_ids].keys.first).voter_id.should == 0
    end

    it "should create N prompts for question" do
      get :create, :question_id => 1, :n => 4, :voter_id => 1
      assigns[:prompt_item_ids].should_not == nil
      assigns[:prompt_item_ids].keys.length.should == 4
      assigns[:prompt_item_ids].keys.each do |prompt|
        prompt = Prompt.find(prompt)
        prompt.question.id.should == 1
        prompt.voter.id.should == 1
        prompt.reload.active.should == false
      end
    end

#    it "should create prompts restricted to items" do
#      items = Item.all(:limit => 3, :conditions => { "items.active" => true,
"items_questions.question_id" => 1 }, :include => :questions)
#      get :create, :question_id => 1, :n => 2, :item_id =>
items.map(&:id).join(','), :voter_id => 1
#      assigns[:prompt_ids].should_not == nil || []
#      assigns[:prompt_ids].each do |prompt_id|
#        prompt = Prompt.find(prompt_id)
#        prompt.question.id.should == 1
#        prompt.voter.id.should == 1
#        sorted(prompt.items | items).should == sorted(items)
#        prompt.reload.active.should == false
#      end
#    end
```

```
    it 'should not set data if not passed' do
      get :create, :question_id => 1
      assigns[:data].should == false
    end

    it 'should set data if passed' do
      get :create, :question_id => 1, :data => 1
      assigns[:data].should == true
    end

  end

  describe "responding to GET list" do
    fixtures :prompts, :items_prompts

    it "should get only users prompts" do
      get :list
      assigns[:prompts].each do |prompt|
        prompt.question.user_id.should == 1
      end
    end

    it "should get prompts for question" do
      get :list, :question_id => 1
      assigns[:prompts].each do |prompt|
        prompt.question.id.should == 1
      end
    end

    it "should get prompts with items" do
      items = Item.all(:limit => 2, :conditions => { :user_id => 1 })
      get :list, :item_id => items.map { |el| el.id }.join(',')
      assigns[:prompts].each do |prompt|
        (prompt.items & items).should_not be_empty
      end
    end
  end

  describe 'responding to GET view' do
    fixtures :prompts, :items_prompts, :items, :stats, :items_stats

    it 'should incr views for prompt' do
      prompt = Prompt.first
      stat = prompt.items.first.stats.first
      get :view, :id => prompt.id
      prompt.items.first.stats.first.views.should == stat.views + 1
    end
  end
end

def sorted(items)
  items.sort_by &:id
endrequire File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe PromptsController do
```

```
  describe "route recognition" do
    it "should generate params for #vote" do
      params_from(:get, "/prompts/view/1").should == {:controller => "prompts",
:action => "view", :id => '1'}
    end
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe QuestionsController do
  fixtures :users, :questions, :items, :votes, :prompts

  before(:each) do
    login_as('quentin')
  end

  describe "responding to GET list" do
    it "should get only users questions" do
      get :list
      assigns[:questions].each do |q|
        q.user_id.should == 1
      end
    end

    it 'should get items for question' do
      get :list
      assigns[:items_count].should == User.find(1).items.all(:conditions => {
:active => true }).length
    end

    it 'should get all items for question' do
      get :list
      assigns[:all_items_count].should == User.find(1).items.length
    end

    it 'should get all votes for question' do
      get :list
      assigns[:votes_count].should == Vote.count(:conditions => {
'questions.user_id' => 1 }, :include => { :prompt => :question })
    end
  end

  describe "responding to POST add" do
    it "should add questions" do
      xml_post :add, "questions"
      question = Question.find_by_name('new')
      question2 = Question.find_by_name('new2')
      assigns[:questions].should == [question, question2]
    end
  end

  describe 'responding to GET show' do
    it 'should expose question' do
      get :show, :id => 1
      assigns[:question].should == Question.find(1)
```

```
    end


    it 'should not expose non-user items' do
      get :show, :id => 4
      assigns[:question].should == nil
    end

    it 'should get items for question' do
      get :show, :id => 1
      assigns[:items_count].should == Question.find(1).items.all(:conditions =>
{ :active => true }).length
    end

    it 'should get all items for question' do
      get :show, :id => 1
      assigns[:all_items_count].should == Question.find(1).items.length
    end

    it 'should get all votes for question' do
      get :show, :id => 1
      assigns[:votes_count].should == Vote.count(:conditions => {
'prompts.question_id' => 1 }, :include => :prompt)
    end
  end

  describe 'responding to POST delete' do
    fixtures :items, :items_questions, :prompts, :votes
    before do
      @question = Question.find(1)
      @iqs = @question.items_questions
      @prompts = @question.prompts
      @votes = @prompts.map(&:vote).flatten.compact
      @prs = @question.prompt_requests
    end

    it 'should return if not post' do
      get :delete
      assigns[:success].should == nil
    end

    it 'should delete question' do
      post :delete, :id => @question.id
      Question.exists?(@question.id).should == false
    end

    it 'should delete item questions' do
      post :delete, :id => @question.id
      ItemsQuestion.all(:conditions => { :id => @iqs.map(&:id) }).empty?.should
== true
    end

    it 'should delete prompt requests' do
      post :delete, :id => @question.id
```

```
      PromptRequest.all(:conditions => { :id => @prs.map(&:id) }).empty?.should
== true
    end

    it 'should delete prompts' do
      post :delete, :id => @question.id
      Prompt.all(:conditions => { :id => @prompts.map(&:id) }).empty?.should ==
true
    end

    it 'should delete votes' do
      post :delete, :id => @question.id
      Vote.all(:conditions => { :id => @votes.map(&:id) }).empty?.should == true
    end

    it 'should assign id' do
      post :delete, :id => @question.id
      assigns[:id].should == @question.id.to_s
    end

    it 'should assign success true' do
      post :delete, :id => @question.id
      assigns[:success].should == true
    end

    it 'should assign success false' do
      post :delete, :id => @question.id
      post :delete, :id => @question.id
      response.status.should =~ /400/
    end
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe RankAlgorithmsController do
  fixtures :users, :rank_algorithms

  before(:each) do
    login_as('quentin')
  end

  describe "responding to GET list" do
    it "should list rank algos" do
      get :list
      assigns[:algorithms].should == RankAlgorithm.find(:all)
    end
  end
endrequire File.dirname(__FILE__) + '/../spec_helper'

# Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper

describe SessionsController do
  fixtures        :users
```

```
  before do
    @user  = mock_user
    @login_params = { :login => 'quentin', :password => 'test' }
    User.stub!(:authenticate).with(@login_params[:login],
@login_params[:password]).and_return(@user)
  end
  def do_create
    post :create, @login_params
  end
  describe "on successful login," do
    [ [:nil,       nil,            nil],
      [:expired,   'valid_token',  15.minutes.ago],
      [:different, 'i_haxxor_joo', 15.minutes.from_now],
      [:valid,     'valid_token',  15.minutes.from_now]
        ].each do |has_request_token, token_value, token_expiry|
      [ true, false ].each do |want_remember_me|
        describe "my request cookie token is #{has_request_token.to_s}," do
          describe "and ask #{want_remember_me ? 'to' : 'not to'} be remembered"
do
            before do
              @ccookies = mock('cookies')
              controller.stub!(:cookies).and_return(@ccookies)
              @ccookies.stub!(:[]).with(:auth_token).and_return(token_value)
              @ccookies.stub!(:delete).with(:auth_token)
              @ccookies.stub!(:[]=)
              @user.stub!(:remember_me)
              @user.stub!(:refresh_token)
              @user.stub!(:forget_me)
              @user.stub!(:remember_token).and_return(token_value)
              @user.stub!(:remember_token_expires_at).and_return(token_expiry)
              @user.stub!(:remember_token?).and_return(has_request_token ==
:valid)
              if want_remember_me
                @login_params[:remember_me] = '1'
              else
                @login_params[:remember_me] = '0'
              end
            end
            it "kills existing login"        do
controller.should_receive(:logout_keeping_session!); do_create; end
            it "authorizes me"               do do_create;
controller.send(:authorized?).should be_true;   end
            it "logs me in"                  do do_create;
controller.send(:logged_in?).should  be_true  end
            it "greets me nicely"            do do_create;
response.flash[:notice].should =~ /success/i   end
            it "sets/resets/expires cookie"  do
controller.should_receive(:handle_remember_cookie!).with(want_remember_me);
do_create end
            it "sends a cookie"              do
controller.should_receive(:send_remember_cookie!);  do_create end
            it 'redirects to the home page'  do do_create; response.should
redirect_to('/')   end
```

```
            it "does not reset my session"   do
controller.should_not_receive(:reset_session).and_return nil; do_create end #
change if you uncomment the reset_session path
            if (has_request_token == :valid)
              it 'does not make new token'   do
@user.should_not_receive(:remember_me);   do_create end
              it 'does refresh token'        do
@user.should_receive(:refresh_token);     do_create end
              it "sets an auth cookie"       do do_create;  end
            else
              if want_remember_me
                it 'makes a new token'       do
@user.should_receive(:remember_me);       do_create end
                it "does not refresh token"  do
@user.should_not_receive(:refresh_token); do_create end
                it "sets an auth cookie"       do do_create;  end
              else
                it 'does not make new token' do
@user.should_not_receive(:remember_me);   do_create end
                it 'does not refresh token'  do
@user.should_not_receive(:refresh_token); do_create end
                it 'kills user token'        do
@user.should_receive(:forget_me);         do_create end
              end
            end
          end # inner describe
        end
      end
    end
  end

  describe "on failed login" do
    before do
      User.should_receive(:authenticate).with(anything(),
anything()).and_return(nil)
      login_as :quentin
    end
    it 'logs out keeping session'   do
controller.should_receive(:logout_keeping_session!); do_create end
    it 'flashes an error'           do do_create; flash[:error].should =~
/Couldn't log you in as 'quentin'/ end
    it 'renders the log in page'    do do_create; response.should
render_template('new')  end
    it "doesn't log me in"          do do_create;
controller.send(:logged_in?).should == false end
    it "doesn't send password back" do
      @login_params[:password] = 'FROBNOZZ'
      do_create
      response.should_not have_text(/FROBNOZZ/i)
    end
  end

  describe "on signout" do
    def do_destroy
      get :destroy
```

```
    end
    before do
      login_as :quentin
    end
    it 'logs me out'                    do
controller.should_receive(:logout_killing_session!); do_destroy end
    it 'redirects me to the home page' do do_destroy; response.should
be_redirect     end
  end

end

describe SessionsController do
  describe "route generation" do
    it "should route the new sessions action correctly" do
      route_for(:controller => 'sessions', :action => 'new').should == { :path
=> "/login", :method => :get }
    end
    it "should route the create sessions correctly" do
      route_for(:controller => 'sessions', :action => 'create').should == {
:path => "/session", :method => :post }
    end
    it "should route the destroy sessions action correctly" do
      route_for(:controller => 'sessions', :action => 'destroy').should == {
:path => "/logout", :method => :delete }
    end
  end

  describe "route recognition" do
    it "should generate params from GET /login correctly" do
      params_from(:get, '/login').should == {:controller => 'sessions', :action
=> 'new'}
    end
    it "should generate params from POST /session correctly" do
      params_from(:post, '/session').should == {:controller => 'sessions',
:action => 'create'}
    end
    it "should generate params from DELETE /session correctly" do
      params_from(:delete, '/logout').should == {:controller => 'sessions',
:action => 'destroy'}
    end
  end

  describe "named routing" do
    before(:each) do
      get :new
    end
    it "should route session_path() correctly" do
      session_path().should == "/session"
    end
    it "should route new_session_path() correctly" do
      new_session_path().should == "/session/new"
    end
  end
```

```
end
require File.dirname(__FILE__) + '/../spec_helper'

describe UsersController do
  fixtures :users

  it 'allows signup' do
    lambda do
      create_user
      response.should be_redirect
    end.should change(User, :count).by(1)
  end

  it 'signs up user in pending state' do
    create_user
    assigns(:user).reload
    assigns(:user).should be_pending
  end

  it 'signs up user with activation code' do
    create_user
    assigns(:user).reload
    assigns(:user).activation_code.should_not be_nil
  end
  it 'requires login on signup' do
    lambda do
      create_user(:login => nil)
      assigns[:user].errors.on(:login).should_not be_nil
      response.should be_success
    end.should_not change(User, :count)
  end

  it 'requires password on signup' do
    lambda do
      create_user(:password => nil)
      assigns[:user].errors.on(:password).should_not be_nil
      response.should be_success
    end.should_not change(User, :count)
  end

  it 'requires password confirmation on signup' do
    lambda do
      create_user(:password_confirmation => nil)
      assigns[:user].errors.on(:password_confirmation).should_not be_nil
      response.should be_success
    end.should_not change(User, :count)
  end

  it 'requires email on signup' do
    lambda do
      create_user(:email => nil)
      assigns[:user].errors.on(:email).should_not be_nil
      response.should be_success
    end.should_not change(User, :count)
  end
```

```
  it 'activates user' do
    User.authenticate('aaron', 'monkey').should be_nil
    get :activate, :id => users(:aaron).activation_code
    response.should redirect_to('/login')
    flash[:notice].should_not be_nil
    flash[:error ].should     be_nil
    User.authenticate('aaron', 'monkey').should == users(:aaron)
  end

  it 'does not activate user without key' do
    get :activate
    flash[:notice].should     be_nil
    flash[:error ].should_not be_nil
  end

  it 'does not activate user with blank key' do
    get :activate, :activation_code => ''
    flash[:notice].should     be_nil
    flash[:error ].should_not be_nil
  end

  it 'does not activate user with bogus key' do
    get :activate, :activation_code => 'i_haxxor_joo'
    flash[:notice].should     be_nil
    flash[:error ].should_not be_nil
  end

  describe 'xml users' do
    # password == jopass
    before do
      @param = { :key =>
Base64.encode64('cetuwes9rewruRec6k7hUz4CRabaD7musa3es9muzeda5') }
    end

    it 'requires a key' do
      lambda do
        xml_post :add, 'user'
      end.should_not change(User, :count)
    end

    it 'create new user' do
      lambda do
        xml_post :add, 'user', @param
      end.should change(User, :count)
    end

    it 'create new active user' do
      xml_post :add, 'user', @param
      User.last.state.should == 'active'
    end
  end
```

```
  def create_user(options = {})
    post :create, :user => { :login => 'quire', :email => 'quire@example.com',
      :password => 'quire69', :password_confirmation => 'quire69'
}.merge(options)
  end
end

describe UsersController do
  describe "route generation" do
    it "should route users's 'index' action correctly" do
      route_for(:controller => 'users', :action => 'index').should == "/users"
    end

    it "should route users's 'new' action correctly" do
      route_for(:controller => 'users', :action => 'new').should == "/signup"
    end

    it "should route {:controller => 'users', :action => 'create'} correctly" do
      route_for(:controller => 'users', :action => 'create').should ==
"/register"
    end

    it "should route users's 'show' action correctly" do
      route_for(:controller => 'users', :action => 'show', :id => '1').should ==
{ :path => "/users/1", :method => :get }
    end

    it "should route users's 'edit' action correctly" do
      route_for(:controller => 'users', :action => 'edit', :id => '1').should ==
{ :path => "/users/1/edit", :method => :get }
    end

    it "should route users's 'update' action correctly" do
      route_for(:controller => 'users', :action => 'update', :id => '1').should
== { :path => "/users/1", :method => :put }
    end

    it "should route users's 'destroy' action correctly" do
      route_for(:controller => 'users', :action => 'destroy', :id => '1').should
== { :path => "/users/1", :method => :delete }
    end
  end

  describe "route recognition" do
    it "should generate params for users's index action from GET /users" do
      params_from(:get, '/users').should == {:controller => 'users', :action =>
'index'}
      params_from(:get, '/users.xml').should == {:controller => 'users', :action
=> 'index', :format => 'xml'}
      params_from(:get, '/users.json').should == {:controller => 'users',
:action => 'index', :format => 'json'}
    end

    it "should generate params for users's new action from GET /users" do
```

```
      params_from(:get, '/users/new').should == {:controller => 'users', :action
=> 'new'}
      params_from(:get, '/users/new.xml').should == {:controller => 'users',
:action => 'new', :format => 'xml'}
      params_from(:get, '/users/new.json').should == {:controller => 'users',
:action => 'new', :format => 'json'}
    end

    it "should generate params for users's create action from POST /users" do
      params_from(:post, '/users').should == {:controller => 'users', :action =>
'create'}
      params_from(:post, '/users.xml').should == {:controller => 'users',
:action => 'create', :format => 'xml'}
      params_from(:post, '/users.json').should == {:controller => 'users',
:action => 'create', :format => 'json'}
    end

    it "should generate params for users's show action from GET /users/1" do
      params_from(:get , '/users/1').should == {:controller => 'users', :action
=> 'show', :id => '1'}
      params_from(:get , '/users/1.xml').should == {:controller => 'users',
:action => 'show', :id => '1', :format => 'xml'}
      params_from(:get , '/users/1.json').should == {:controller => 'users',
:action => 'show', :id => '1', :format => 'json'}
    end

    it "should generate params for users's edit action from GET /users/1/edit"
do
      params_from(:get , '/users/1/edit').should == {:controller => 'users',
:action => 'edit', :id => '1'}
    end

    it "should generate params {:controller => 'users', :action => update', :id
=> '1'} from PUT /users/1" do
      params_from(:put , '/users/1').should == {:controller => 'users', :action
=> 'update', :id => '1'}
      params_from(:put , '/users/1.xml').should == {:controller => 'users',
:action => 'update', :id => '1', :format => 'xml'}
      params_from(:put , '/users/1.json').should == {:controller => 'users',
:action => 'update', :id => '1', :format => 'json'}
    end

    it "should generate params for users's destroy action from DELETE /users/1"
do
      params_from(:delete, '/users/1').should == {:controller => 'users',
:action => 'destroy', :id => '1'}
      params_from(:delete, '/users/1.xml').should == {:controller => 'users',
:action => 'destroy', :id => '1', :format => 'xml'}
      params_from(:delete, '/users/1.json').should == {:controller => 'users',
:action => 'destroy', :id => '1', :format => 'json'}
    end
  end

  describe "named routing" do
    before(:each) do
```

```
      get :new
    end

    it "should route users_path() to /users" do
      users_path().should == "/users"
#      formatted_users_path(:format => 'xml').should == "/users.xml"
#      formatted_users_path(:format => 'json').should == "/users.json"
    end

    it "should route new_user_path() to /users/new" do
      new_user_path().should == "/users/new"
#      formatted_new_user_path(:format => 'xml').should == "/users/new.xml"
#      formatted_new_user_path(:format => 'json').should == "/users/new.json"
    end

    it "should route user_(:id => '1') to /users/1" do
      user_path(:id => '1').should == "/users/1"
#      formatted_user_path(:id => '1', :format => 'xml').should ==
"/users/1.xml"
#      formatted_user_path(:id => '1', :format => 'json').should ==
"/users/1.json"
    end

    it "should route edit_user_path(:id => '1') to /users/1/edit" do
      edit_user_path(:id => '1').should == "/users/1/edit"
    end
  end

end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe VotersController do
  fixtures :users, :voters

  before(:each) do
    login_as('quentin')
  end

  describe "responding to GET list" do
    it "should return all of the users voters" do
      get :list
      assigns[:voters].should == Voter.find_all_by_user_id(1)
      assigns[:voters].each do |voter|
        voter.user_id.should == 1
      end
    end
  end

  describe "responding to POST add" do
    it "should create new voters" do
      xml_post :add, 'voters'
      voters = Voter.find_all_by_user_id(1)[-2,2]
      assigns[:voters].should == voters
      assigns[:voters].each do |voter|
        voter.features.first.name.should == "gender"
```

```
      end
      assigns[:voters].first.features.first.value.should == 0
      assigns[:voters].last.features.first.value.should == 1
    end
  end

  describe "responding to GET set" do
    it "should require a voter id of the user" do
      get :set, :id => 2
      response.status.should =~ /400/
    end

    it "should update the feature" do
      voter = Voter.find(1)
      voter.features.clear
      get :set, :id => 1, :gender => 1
      voter.reload
      feature = voter.features.find_by_name('gender')
      assigns[:voter].should == voter
      feature.value.should == 1
      voter.features.should == [feature]
    end
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe VotersController do
  describe "route generation" do
    it "should map #index" do
      route_for(:controller => "voters", :action => "index").should == {:path =>
"/voters", :method => :get}
    end

    it "should map #new" do
      route_for(:controller => "voters", :action => "new").should == {:path =>
"/voters/new", :method => :get}
    end

    it "should map #show" do
      route_for(:controller => "voters", :action => "show", :id => "1").should
== {:path => "/voters/1", :method => :get}
    end

    it "should map #edit" do
      route_for(:controller => "voters", :action => "edit", :id => "1").should
== {:path => "/voters/1/edit", :method => :get}
    end

    it "should map #update" do
      route_for(:controller => "voters", :action => "update", :id => "1").should
== {:path => "/voters/1", :method => :put}
    end

    it "should map #destroy" do
```

```
      route_for(:controller => "voters", :action => "destroy", :id =>
"1").should == {:path => "/voters/1", :method => :delete}
    end
  end

  describe "route recognition" do
    it "should generate params for #index" do
      params_from(:get, "/voters").should == {:controller => "voters", :action
=> "index"}
    end

    it "should generate params for #new" do
      params_from(:get, "/voters/new").should == {:controller => "voters",
:action => "new"}
    end

    it "should generate params for #create" do
      params_from(:post, "/voters").should == {:controller => "voters", :action
=> "create"}
    end

    it "should generate params for #show" do
      params_from(:get, "/voters/1").should == {:controller => "voters", :action
=> "show", :id => "1"}
    end

    it "should generate params for #edit" do
      params_from(:get, "/voters/1/edit").should == {:controller => "voters",
:action => "edit", :id => "1"}
    end

    it "should generate params for #update" do
      params_from(:put, "/voters/1").should == {:controller => "voters", :action
=> "update", :id => "1"}
    end

    it "should generate params for #destroy" do
      params_from(:delete, "/voters/1").should == {:controller => "voters",
:action => "destroy", :id => "1"}
    end
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe VotesController do
  fixtures :users, :questions, :prompts, :items_prompts, :prompt_algorithms,
:items, :items_questions, :voters, :rank_algorithms

  before(:each) do
    login_as('quentin')
  end

  describe "responding to GET add" do
    before do
      Vote.delete_all
```

```
  Stat.delete_all
  ActiveRecord::Base.connection.execute("DELETE FROM `items_stats`")
  for iq in ItemsQuestion.all
    iq.update_attributes({:ratings => 0, :wins => 0, :losses => 0})
  end
end

it "should require prompt_id" do
  get :add
  response.status.should =~ /417/
end

it "should require prompt_id with skip" do
  get :add, :skip => 1
  response.status.should =~ /417/
end

it "should require prompt_id with item" do
  get :add, :item_id => 1
  response.status.should =~ /417/
end

it "should require item_id in prompt" do
  get :add, :prompt_id => 1, :item_id => 3, :voter_id => 1
  response.status.should =~ /417/
end

it "should require user voter" do
  get :add, :prompt_id => 1, :item_id => 1, :voter_id => 2
  response.status.should =~ /403/
end

it "should add vote" do
  get :add, :prompt_id => 1, :item_id => 1, :voter_id => 1
  assigns[:vote].items.first.id.should == 1
  assigns[:vote].prompt_id.should == 1
end

it 'should allow skip' do
  get :add, :prompt_id => 1, :voter_id => 1, :skip => 1
  assigns[:vote].prompt_id.should == 1
  assigns[:vote].items.should be_empty
end

it 'should allow 0 voter id' do
  get :add, :prompt_id => 1, :item_id => 1, :voter_id => 0
  assigns[:vote].items.first.id.should == 1
  assigns[:vote].prompt_id.should == 1
end

it "should add vote for multiple items" do
  get :add, :prompt_id => 1, :item_id => '1,2', :voter_id => 0
  assigns[:vote].voter_id.should == 0
end
```

```
    it "should add vote with response time" do
      # 100 == 1s
      get :add, :prompt_id => 1, :item_id => 1, :voter_id => 1, :response_time
=> 100
      assigns[:vote].response_time.should == 100
    end

    it 'should increment ratings' do
      items = Prompt.find(1).items
      get :add, :prompt_id => 1, :voter_id => 1, :skip => 1
      prompt = Prompt.find(1)
      prompt.items.each do |item|
        item.iq(prompt.question_id).ratings.should ==
items.shift.iq(prompt.question_id).ratings
      end
    end

    it 'should keep stored ratings aligned with db records' do
      for prompt in Prompt.all do
        get :add, :prompt_id => prompt.id, :voter_id => 1, :skip => 1
      end
      for prompt in Prompt.all do
        prompt.items.each do |item|
          item.iq(prompt.question_id).ratings.should == Vote.count(
            :joins => "INNER JOIN prompts ON (votes.prompt_id=prompts.id) INNER
JOIN items_prompts ON (prompts.id=items_prompts.prompt_id AND
items_prompts.item_id=#{item.id})",
            :conditions => { 'prompts.question_id' => prompt.question_id }
          )
        end
      end
    end

    it 'should increment wins of winner' do
      prompt = Prompt.find(1)
      wins = Item.find(1).iq(prompt.question_id).wins
      get :add, :prompt_id => 1, :item_id => 1, :voter_id => 1
      Item.find(1).iq(prompt.question_id).wins.should == wins + 1
    end

    it 'should keep stored wins aligned with db records' do
      i = 0
      for prompt in Prompt.all do
        if ((i += 1) % 2).zero?
          get :add, :prompt_id => prompt.id, :voter_id => 1, :item_id => 1
        else
          get :add, :prompt_id => prompt.id, :voter_id => 1, :skip => 1
        end
      end
      item = Item.find(1)
      item.iq(1).wins.should == Vote.count(
        :joins => "INNER JOIN prompts ON (votes.prompt_id=prompts.id) INNER JOIN
items_votes ON (votes.id=items_votes.vote_id AND
items_votes.item_id=#{item.id})",
        :conditions => { 'prompts.question_id' => 1 }
```

```
      )
    end

    it 'should keep store losses aligned with db losses' do
      i = 0
      for prompt in Prompt.all do
        if ((i += 1) % 2).zero?
          get :add, :prompt_id => prompt.id, :voter_id => 1, :item_id => 1
        else
          get :add, :prompt_id => prompt.id, :voter_id => 1, :skip => 1
        end
      end
      for item in Item.all
        item.iq(prompt.question_id).losses.should == Prompt.count(
          :joins => "INNER JOIN items_prompts ON
(prompts.id=items_prompts.prompt_id AND items_prompts.item_id=#{item.id}) INNER
JOIN votes ON (votes.prompt_id=prompts.id) INNER JOIN items_votes ON
(votes.id=items_votes.vote_id AND items_votes.item_id!=items_prompts.item_id)",
          :conditions => { 'prompts.question_id' => prompt.question_id }
        )
      end
    end

    it 'should have total ratings equal to number vote records * 2' do
      i = 0
      for prompt in Prompt.all do
        if ((i += 1) % 2).zero?
          get :add, :prompt_id => prompt.id, :voter_id => 1, :item_id => 1
        else
          get :add, :prompt_id => prompt.id, :voter_id => 1, :skip => 1
        end
      end
      ItemsQuestion.all.sumup(:ratings).should == Vote.count * 2
    end

    it 'should have total wins equal to number vote records with items * 2' do
      i = 0
      for prompt in Prompt.all do
        if ((i += 1) % 2).zero?
          get :add, :prompt_id => prompt.id, :voter_id => 1, :item_id => 1
        else
          get :add, :prompt_id => prompt.id, :voter_id => 1, :skip => 1
        end
      end
      ItemsQuestion.all.sumup(:wins).should == Prompt.all(
        :select => "items_votes.item_id, prompts.question_id, COUNT(*) AS
count",
        :joins => "INNER JOIN votes ON (votes.prompt_id=prompts.id) INNER JOIN
items_votes ON (votes.id=items_votes.vote_id)",
        :group => "items_votes.item_id, prompts.question_id"
      ).sumup(:count)
    end

    it 'should have total skips equal to number vote records without items * 2'
do
```

```
      i = 0
      for prompt in Prompt.all do
        if ((i += 1) % 2).zero?
          get :add, :prompt_id => prompt.id, :voter_id => 1, :item_id => 1
        else
          get :add, :prompt_id => prompt.id, :voter_id => 1, :skip => 1
        end
      end
      ItemsQuestion.all.inject(0) do |sum, iq|
        sum += iq.ratings - iq.wins - iq.losses
      end.should == Prompt.all(
        :select => "items_votes.item_id, prompts.question_id, COUNT(*) AS
count",
        :joins => "INNER JOIN votes ON (votes.prompt_id=prompts.id) LEFT OUTER
JOIN items_votes ON (votes.id=items_votes.vote_id)",
        :conditions => "items_votes.item_id IS NULL",
        :group => "items_votes.item_id, prompts.question_id"
      ).sumup(:count) * 2
    end

    it 'should adjust elo with winner' do
      prompt = Prompt.find(1)
      old_first = prompt.items.first.iq(prompt.question_id).position
      old_last = prompt.items.last.iq(prompt.question_id).position
      get :add, :prompt_id => 1, :item_id => 1, :voter_id => 1
      (prompt.items.first.iq(prompt.question_id).position > old_first).should ==
true
      (prompt.items.last.iq(prompt.question_id).position < old_last).should ==
true
    end

    it 'should adjust elo with skip' do
      prompt = Prompt.find(1)
      old_first = prompt.items.first.iq(prompt.question_id).position
      old_last = prompt.items.last.iq(prompt.question_id).position
      get :add, :prompt_id => 1, :skip => 1, :voter_id => 1
      prompt.items.first.iq(prompt.question_id).position.should == old_first
      prompt.items.last.iq(prompt.question_id).position.should == old_last - 1
    end

    it 'should create stat for items' do
      get :add, :prompt_id => 1, :skip => 1, :voter_id => 1
      Prompt.find(1).items.each do |item|
        item.stats.empty?.should == false
      end
    end

    it 'should set stats for items' do
      get :add, :prompt_id => 1, :skip => 1, :voter_id => 1
      Prompt.find(1).items.each do |item|
        item.stats.each { |stat| stat.votes.should == 1 }
      end
    end

    it 'should add tracking for vote' do
```

```
      tracking = 'DATA STRING'
      get :add, :prompt_id => 1, :skip => 1, :voter_id => 1, :tracking =>
tracking
      Vote.last.tracking.should == tracking
    end
  end

  describe "responding to GET list" do
    it "should only get votes for user" do
      get :list
      assigns[:votes].each do |vote|
        vote.prompt.question.user_id.should == 1
      end
    end
  end
endrequire File.dirname(__FILE__) + '/../spec_helper'
include ApplicationHelper
include UsersHelper
include AuthenticatedTestHelper

describe UsersHelper do
  before do
    @user = mock_user
  end

  describe "if_authorized" do
    it "yields if authorized" do
      should_receive(:authorized?).with('a','r').and_return(true)
      if_authorized?('a','r'){|action,resource| [action,resource,'hi'] }.should
== ['a','r','hi']
    end
    it "does nothing if not authorized" do
      should_receive(:authorized?).with('a','r').and_return(false)
      if_authorized?('a','r'){ 'hi' }.should be_nil
    end
  end

  describe "link_to_user" do
    it "should give an error on a nil user" do
      lambda { link_to_user(nil) }.should raise_error('Invalid user')
    end
    it "should link to the given user" do
      should_receive(:user_path).at_least(:once).and_return('/users/1')
      link_to_user(@user).should have_tag("a[href='/users/1']")
    end
    it "should use given link text if :content_text is specified" do
      link_to_user(@user, :content_text => 'Hello there!').should have_tag("a",
'Hello there!')
    end
    it "should use the login as link text with no :content_method specified" do
      link_to_user(@user).should have_tag("a", 'user_name')
    end
    it "should use the name as link text with :content_method => :name" do
      link_to_user(@user, :content_method => :name).should have_tag("a", 'U.
Surname')
```

```
    end
    it "should use the login as title with no :title_method specified" do
      link_to_user(@user).should have_tag("a[title='user_name']")
    end
    it "should use the name as link title with :content_method => :name" do
      link_to_user(@user, :title_method => :name).should have_tag("a[title='U.
Surname']")
    end
    it "should have nickname as a class by default" do
      link_to_user(@user).should have_tag("a.nickname")
    end
    it "should take other classes and no longer have the nickname class" do
      result = link_to_user(@user, :class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
  end

  describe "link_to_login_with_IP" do
    it "should link to the login_path" do
      link_to_login_with_IP().should have_tag("a[href='/login']")
    end
    it "should use given link text if :content_text is specified" do
      link_to_login_with_IP('Hello there!').should have_tag("a", 'Hello there!')
    end
    it "should use the login as link text with no :content_method specified" do
      link_to_login_with_IP().should have_tag("a", '0.0.0.0')
    end
    it "should use the ip address as title" do
      link_to_login_with_IP().should have_tag("a[title='0.0.0.0']")
    end
    it "should by default be like school in summer and have no class" do
      link_to_login_with_IP().should_not have_tag("a.nickname")
    end
    it "should have some class if you tell it to" do
      result = link_to_login_with_IP(nil, :class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
    it "should have some class if you tell it to" do
      result = link_to_login_with_IP(nil, :tag => 'abbr')
      result.should have_tag("abbr[title='0.0.0.0']")
    end
  end

  describe "link_to_current_user, When logged in" do
    before do
      stub!(:current_user).and_return(@user)
    end
    it "should link to the given user" do
      should_receive(:user_path).at_least(:once).and_return('/users/1')
      link_to_current_user().should have_tag("a[href='/users/1']")
    end
    it "should use given link text if :content_text is specified" do
```

```
      link_to_current_user(:content_text => 'Hello there!').should have_tag("a",
'Hello there!')
    end
    it "should use the login as link text with no :content_method specified" do
      link_to_current_user().should have_tag("a", 'user_name')
    end
    it "should use the name as link text with :content_method => :name" do
      link_to_current_user(:content_method => :name).should have_tag("a", 'U.
Surname')
    end
    it "should use the login as title with no :title_method specified" do
      link_to_current_user().should have_tag("a[title='user_name']")
    end
    it "should use the name as link title with :content_method => :name" do
      link_to_current_user(:title_method => :name).should have_tag("a[title='U.
Surname']")
    end
    it "should have nickname as a class" do
      link_to_current_user().should have_tag("a.nickname")
    end
    it "should take other classes and no longer have the nickname class" do
      result = link_to_current_user(:class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
  end

  describe "link_to_current_user, When logged out" do
    before do
      stub!(:current_user).and_return(nil)
    end
    it "should link to the login_path" do
      link_to_current_user().should have_tag("a[href='/login']")
    end
    it "should use given link text if :content_text is specified" do
      link_to_current_user(:content_text => 'Hello there!').should have_tag("a",
'Hello there!')
    end
    it "should use 'not signed in' as link text with no :content_method
specified" do
      link_to_current_user().should have_tag("a", 'not signed in')
    end
    it "should use the ip address as title" do
      link_to_current_user().should have_tag("a[title='0.0.0.0']")
    end
    it "should by default be like school in summer and have no class" do
      link_to_current_user().should_not have_tag("a.nickname")
    end
    it "should have some class if you tell it to" do
      result = link_to_current_user(:class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
  end
```

```
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe VotersHelper do

  #Delete this example and add some real ones or delete this file
  it "should be included in the object returned by #helper" do
    included_modules = (class << helper; self; end).send :included_modules
    included_modules.should include(VotersHelper)
  end

end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Feature do
  before(:each) do
    @valid_attributes = {
      :voter_id => "1",
      :name => "value for name",
      :value => "value for value"
    }
  end

  it "should create a new instance given valid attributes" do
    Feature.create!(@valid_attributes)
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Item do
  before(:each) do
    @valid_attributes = {
      :user_id => "1",
      :data => "value for name"
    }
  end

  it "should create a new instance given valid attributes" do
    Item.create!(@valid_attributes)
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe ItemsQuestion do
  before(:each) do
    @valid_attributes = {
      :item_id => "1",
      :question_id => "1",
      :position => "1",
      :wins => "1",
      :ratings => "1"
    }
  end

  it "should create a new instance given valid attributes" do
```

```
    ItemsQuestion.create!(@valid_attributes)
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe PromptAlgorithm do
  before(:each) do
    @valid_attributes = {
      :name => "1",
      :data => "1"
    }
  end

  it "should create a new instance given valid attributes" do
    PromptAlgorithm.create!(@valid_attributes)
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe PromptRequest do
  before(:each) do
    @valid_attributes = {
      :question_id => "1",
      :voter_id => "1"
    }
  end

  it "should create a new instance given valid attributes" do
    PromptRequest.create!(@valid_attributes)
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Prompt do
  before(:each) do
    @valid_attributes = {
      :question_id => "1",
      :prompt_algorithm_id => "1",
      :voter_id => "1"
    }
  end

  it "should create a new instance given valid attributes" do
    Prompt.create!(@valid_attributes)
  end

  # describe 'find or create' do
  #   before do
  #     @prompt = mock_model(Prompt)
  #   end
  #
  #   it 'should find prompts for args' do
  #     Prompt.should_receive(:find_for).with({}, nil, 1).and_return(@prompt)
  #     @prompt.should_receive(:length).and_return(1)
  #     @prompt.should_receive(:map).and_return([1])
```

```
  #     @prompt.should_receive(:empty?).and_return(false)
  #     Prompt.find_or_create({}, nil, 1).should == @prompt
  #   end
  #
  #   it 'should deactivate prompts' do
  #     Prompt.should_receive(:find_for).with({}, nil, 1).and_return(@prompt)
  #     @prompt.should_receive(:length).and_return(1)
  #     @prompt.should_receive(:empty?).and_return(false)
  #     @prompt.should_receive(:map).and_return([1])
  #     Prompt.should_receive(:update_all).with("active=0", "prompts.id IN (1)")
  #     Prompt.find_or_create({}, nil, 1).should == @prompt
  #   end
  #
  #   it 'should not deactive prompts if prompts are empty' do
  #     Prompt.should_receive(:find_for).with({}, nil, 1).and_return(@prompt)
  #     @prompt.should_receive(:length).and_return(1)
  #     @prompt.should_receive(:empty?).and_return(true)
  #     Prompt.should_not_receive(:update_all)
  #     @prompt.should_not_receive(:map)
  #     Prompt.find_or_create({}, nil, 1).should == @prompt
  #   end
  #
  #   it 'should request missing prompts' do
  #     Prompt.should_receive(:find_for).with({}, nil, 2).and_return([@prompt])
  #     Prompt.should_receive(:request).with({:count => 1},
nil).and_return([@prompt])
  #     @prompt.should_receive(:question_id).twice.and_return([1])
  #     @prompt.should_receive(:items).twice.and_return([])
  #     Prompt.find_or_create({}, nil, 2).should == Array.new(2, @prompt)
  #   end
  #
  #   it 'should create stats for items in prompts' do
  #     items = Array.new(2, mock_model(Item))
  #     Stat.should_receive(:view).with(1, items).twice
  #     Prompt.should_receive(:find_for).with({}, nil, 2).and_return([@prompt])
  #     Prompt.should_receive(:request).with({:count => 1},
nil).and_return([@prompt])
  #     @prompt.should_receive(:question_id).twice.and_return(1)
  #     @prompt.should_receive(:items).twice.and_return(items)
  #     Prompt.find_or_create({}, nil, 2).should == Array.new(2, @prompt)
  #   end
  # end

  describe 'prompt request algo' do
    fixtures :items, :questions, :items_questions, :rank_algorithms

    before do
      @question = Question.first
      @voter_id = 0
      @num = 1
      @items = Item.all(        {
        :joins => "INNER JOIN items_questions ON
(items_questions.question_id=#{@question.id} AND
items_questions.item_id=items.id)",
        :conditions => { :active => true },
```

```
        :group => "items.id"
      })
      srand 1234
      @count = 10
      items = @items.dup
      @prompt_items = @count.times.inject([]) do |arr, i|
        items = @items.dup if items.length < 2 # ensure we still have items
        el = items.delete_at(rand(items.length))
        arr << [el, items.delete_at(rand(items.length))]
      end
      Prompt.delete_all
      PromptRequest.delete_all
      Stat.delete_all
      ActiveRecord::Base.connection.execute("DELETE FROM `items_stats`")
      srand 1234
    end

    it 'should return prompt' do
      Prompt.fetch(@question.id, @voter_id, @num, false).empty?.should == false
    end

    it "should return same as seeded prompts if count < #{@count}" do
      pids = @prompt_items[0].map(&:id).sort
      (Prompt.find((Prompt.fetch(@question.id, @voter_id, @num,
false).first.first)).items.map(&:id) | pids).sort.should == pids
    end

    it 'should set prompts inactive' do
      Prompt.find(Prompt.fetch(@question.id, @voter_id, @num,
false).first.first).reload.active.should == false
    end

    it 'should set view for prompt items' do
      Prompt.find(Prompt.fetch(@question.id, @voter_id, @num,
false).first.first).items.map{ |i| i.reload.stats }.flatten.each do |stat|
        stat.views.should == 1
      end
    end

    it 'should return standard random prompt prime and no stats' do
      prompts = Prompt.find(Prompt.fetch(@question.id, @voter_id, @count,
true).keys)
      @prompt_items.each do |items|
        pids = items.map(&:id).sort
        (prompts.shift.items.map(&:id).sort | pids).sort.should == pids
      end
    end

    describe 'with stats' do
      before do
        Stat.delete_all
        @count = 10
        Prompt.fetch(@question.id, @voter_id, @count, false).keys.each do
|prompt_id|
          prompt = Prompt.find(prompt_id)
```

```
          if (prompt_id % 2).zero?
            2.times { Stat.vote(prompt.question_id, prompt.items) }
          else
            Stat.view(prompt.question_id, prompt.items)
          end
        end
        srand 1234
        stats = Stat.find_all_by_question_id(@question.id, :order => 'score')
        pos = stats.map(&:score).min.abs + 1
        norm = cur = 0
        # the probablity of choose the items in a stat is stat.score /
sum(stats.scores)
        stats = stats.inject({}) do |hash, stat|
          norm += adj = stat.score + pos
          hash[stat] = [cur, cur += adj]
          hash
        end
        @stat_items = @count.times.inject([]) do |arr, i|
          r = rand(norm)
          arr << stats.detect { |stat| stat[1][0] <= r && r < stat[1][1]
}[0].items
        end
        srand 1234
      end

      it 'should return standard random prompt prime false' do
        prompts = Prompt.find(Prompt.fetch(@question.id, @voter_id, @count,
false).keys)
        @prompt_items.first(@count).each do |items|
          pids = items.map(&:id).sort
          (prompts.shift.items.map(&:id).sort | pids).sort.should == pids
        end
      end

      it 'should set rand algo id if prime false' do
        Prompt.find(Prompt.fetch(@question.id, @voter_id, @count,
false).keys).each do |prompt|
          prompt.prompt_algorithm_id.should == 2
        end
      end


      it 'should base prompts on stats if prime true and stats' do
        prompts = Prompt.find(Prompt.fetch(@question.id, @voter_id, @count,
true).keys)
        @stat_items.each do |items|
          pids = items.map(&:id).sort
          (prompts.shift.items.map(&:id) | pids).sort.should == pids
        end
      end

      it 'should set pop algo id if prime true and stats' do
        Prompt.find(Prompt.fetch(@question.id, @voter_id, @count,
true).keys).each do |prompt|
          prompt.prompt_algorithm_id.should == 3
```

```
        end
      end
    end
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Question do
  before(:each) do
    @valid_attributes = {
      :user_id => "1",
      :name => "1"
    }
  end

  it "should create a new instance given valid attributes" do
    Question.create!(@valid_attributes)
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe RankAlgorithm do
  before(:each) do
    @valid_attributes = {
      :name => "1",
      :data => "1"
    }
  end

  it "should create a new instance given valid attributes" do
    RankAlgorithm.create!(@valid_attributes)
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Stat do
  fixtures :items, :rank_algorithms

  before(:each) do
    @valid_attributes = {
      :question_id => 1,
    }
  end

  it 'should create a new instance given valid attributes' do
    Stat.create!(@valid_attributes)
  end

  describe 'view' do
    fixtures :stats, :items_stats

    before do
      @ra = RankAlgorithm.find(Constants::Stat::DEFAULT_RANK_ALGO)
      @qid = 1
      @items = Item.all(:limit => 2)
```

```
  @stat = Stat.view(@qid, @items)
  @items |= @stat.items
end

describe 'if stat does not exist' do
  it 'should create a stat' do
    @stat.should_not == nil
  end

  it 'should create a stat with 0 votes' do
    @stat.votes.should == 0
  end

  it 'should create a stat with 1 view' do
    @stat.views.should == 1
  end

  it 'should create a stat for question' do
    @stat.question_id.should == @qid
  end

  it 'should create a stat for default rank algo' do
    @stat.rank_algorithm.should == @ra
  end

  it 'should create a stat with score based on default rank algo' do
    @stat.score.should == (-1 ** @ra.data.to_i).to_f
  end

  it 'should create a stat with items' do
    @stat.items.sort_by(&:id).should == @items.sort_by(&:id)
  end
end

describe 'if stat does exist' do
  before do
    @old_stat = @stat.dup
    @stat = Stat.view(@qid, @items)
  end

  it 'should return stat' do
    @stat.id.should == @old_stat.id
  end

  it 'should incr stat views' do
    @stat.views.should == @old_stat.views + 1
  end

  it 'should set views to >= votes' do
    (@stat.views >= @stat.votes).should == true
  end

  it 'should not increment votes' do
    @stat.votes.should == @old_stat.votes
  end
```

```
    it 'should update score' do
      n = 2 * @stat.votes - @stat.views
      @stat.score.should == ((n / n.abs) * (n.abs) ** @ra.data.to_i).to_f
    end
  end
end

describe 'vote' do

  fixtures :stats, :items_stats

  before do
    @ra = RankAlgorithm.find(Constants::Stat::DEFAULT_RANK_ALGO)
    @qid = 1
    @items = Item.all(:limit => 2)
    @stat = Stat.vote(@qid, @items)
    @items |= @stat.items
  end

  describe 'if stat does not exist' do
    it 'should create a stat' do
      @stat.should_not == nil
    end

    it 'should create a stat with one vote' do
      @stat.votes.should == 1
    end

    it 'should create a stat with one view' do
      @stat.views.should == 1
    end

    it 'should create a stat for question' do
      @stat.question_id.should == @qid
    end

    it 'should create a stat for default rank algo' do
      @stat.rank_algorithm.should == @ra
    end

    it 'should create a stat with score based on default rank algo' do
      @stat.score.should == 1 ** @ra.data.to_i
    end

    it 'should create a stat with items' do
      @stat.items.sort_by(&:id).should == @items.sort_by(&:id)
    end
  end

  describe 'if stat does exist' do
    before do
      @old_stat = @stat.dup
      @stat = Stat.vote(@qid, @items)
    end
```

```
      it 'should return stat' do
        @stat.id.should == @old_stat.id
      end

      it 'should incr stat votes' do
        @stat.votes.should == @old_stat.votes + 1
      end

      it 'should set views to >= votes' do
        (@stat.views >= @stat.votes).should == true
      end

      it 'should update score' do
        @stat.score.should == (2*@stat.votes - @stat.views) ** @ra.data.to_i
      end

      it 'should have only one stat for pair of items' do
        @stat = Stat.vote(@qid, @items)
        old_lengths = @items.map { |item| item.stats.length }
        1.upto(10) do
          @stat = Stat.vote(@qid, @items)
        end
        for item in @items
          item.stats.length.should == old_lengths.shift
        end
      end
    end
  end
end
# -*- coding: utf-8 -*-
require File.dirname(__FILE__) + '/../spec_helper'

# Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead.
# Then, you can remove it from this and the functional test.
include AuthenticatedTestHelper

describe User do
  fixtures :users

  describe 'being created' do
    before do
      @user = nil
      @creating_user = lambda do
        @user = create_user
        violated "#{@user.errors.full_messages.to_sentence}" if
@user.new_record?
      end
    end

    it 'increments User#count' do
      @creating_user.should change(User, :count).by(1)
    end

    it 'initializes #activation_code' do
```

```
      @creating_user.call
      @user.reload
      @user.activation_code.should_not be_nil
    end

    it 'starts in pending state' do
      @creating_user.call
      @user.reload
      @user.should be_pending
    end
  end

  #
  # Validations
  #

  it 'requires login' do
    lambda do
      u = create_user(:login => nil)
      u.errors.on(:login).should_not be_nil
    end.should_not change(User, :count)
  end

  describe 'allows legitimate logins:' do
    ['123', '1234567890_234567890_234567890_234567890',
     'hello.-_there@funnychar.com'].each do |login_str|
      it "'#{login_str}'" do
        lambda do
          u = create_user(:login => login_str)
          u.errors.on(:login).should     be_nil
        end.should change(User, :count).by(1)
      end
    end
  end
  describe 'disallows illegitimate logins:' do
    ['12', '1234567890_234567890_234567890_234567890_', "tab\t", "newline\n",
     "I√±t√´rn√¢ti√¥n√†liz√¶ti√∏n hasn't happened to ruby 1.8 yet",
     'semicolon;', 'quote"', 'tick\'', 'backtick`', 'percent%', 'plus+', 'space
'].each do |login_str|
      it "'#{login_str}'" do
        lambda do
          u = create_user(:login => login_str)
          u.errors.on(:login).should_not be_nil
        end.should_not change(User, :count)
      end
    end
  end

  it 'requires password' do
    lambda do
      u = create_user(:password => nil)
      u.errors.on(:password).should_not be_nil
    end.should_not change(User, :count)
  end
```

```
  it 'requires password confirmation' do
    lambda do
      u = create_user(:password_confirmation => nil)
      u.errors.on(:password_confirmation).should_not be_nil
    end.should_not change(User, :count)
  end

  it 'requires email' do
    lambda do
      u = create_user(:email => nil)
      u.errors.on(:email).should_not be_nil
    end.should_not change(User, :count)
  end

  describe 'allows legitimate emails:' do
    ['foo@bar.com', 'foo@newskool-tld.museum', 'foo@twoletter-tld.de',
'foo@nonexistant-tld.qq',
     'r@a.wk', '1234567890-234567890-234567890-234567890-234567890-234567890-
234567890-234567890-234567890@gmail.com',
     'hello.-_there@funnychar.com', 'uucp%addr@gmail.com', 'hello+routing-
str@gmail.com',
     'domain@can.haz.many.sub.doma.in', 'student.name@university.edu'
    ].each do |email_str|
      it "'#{email_str}'" do
        lambda do
          u = create_user(:email => email_str)
          u.errors.on(:email).should     be_nil
        end.should change(User, :count).by(1)
      end
    end
  end
  describe 'disallows illegitimate emails' do
    ['!!@nobadchars.com', 'foo@no-rep-dots..com', 'foo@badtld.xxx',
'foo@toolongtld.abcdefg',
     'I√±t√´rn√¢ti√¥n√†liz√¶ti√∏n@hasnt.happened.to.email',
'need.domain.and.tld@de', "tab\t", "newline\n",
     'r@.wk', '1234567890-234567890-234567890-234567890-234567890-234567890-
234567890-234567890-234567890@gmail2.com',
     # these are technically allowed but not seen in practice:
     'uucp!addr@gmail.com', 'semicolon;@gmail.com', 'quote"@gmail.com',
'tick\'@gmail.com', 'backtick`@gmail.com', 'space @gmail.com',
'bracket<@gmail.com', 'bracket>@gmail.com'
    ].each do |email_str|
      it "'#{email_str}'" do
        lambda do
          u = create_user(:email => email_str)
          u.errors.on(:email).should_not be_nil
        end.should_not change(User, :count)
      end
    end
  end

  describe 'allows legitimate names:' do
    ['Andre The Giant (7\'4", 520 lb.) -- has a posse',
```

```
     '',
'1234567890_234567890_234567890_234567890_234567890_234567890_234567890_23456789
0_234567890_234567890',
    ].each do |name_str|
      it "'#{name_str}'" do
        lambda do
          u = create_user(:name => name_str)
          u.errors.on(:name).should     be_nil
        end.should change(User, :count).by(1)
      end
    end
  end
  describe "disallows illegitimate names" do
    ["tab\t", "newline\n",

'1234567890_234567890_234567890_234567890_234567890_234567890_234567890_23456789
0_234567890_234567890_',
     ].each do |name_str|
      it "'#{name_str}'" do
        lambda do
          u = create_user(:name => name_str)
          u.errors.on(:name).should_not be_nil
        end.should_not change(User, :count)
      end
    end
  end

  it 'resets password' do
    users(:quentin).update_attributes(:password => 'new password',
:password_confirmation => 'new password')
    User.authenticate('quentin', 'new password').should == users(:quentin)
  end

  it 'does not rehash password' do
    users(:quentin).update_attributes(:login => 'quentin2')
    User.authenticate('quentin2', 'monkey').should == users(:quentin)
  end

  #
  # Authentication
  #

  it 'authenticates user' do
    User.authenticate('quentin', 'monkey').should == users(:quentin)
  end

  it "doesn't authenticate user with bad password" do
    User.authenticate('quentin', 'invalid_password').should be_nil
  end

 if REST_AUTH_SITE_KEY.blank?
   # old-school passwords
   it "authenticates a user against a hard-coded old-style password" do
     User.authenticate('old_password_holder', 'test').should ==
users(:old_password_holder)
```

```
   end
 else
   it "doesn't authenticate a user against a hard-coded old-style password" do
     User.authenticate('old_password_holder', 'test').should be_nil
   end

   # New installs should bump this up and set REST_AUTH_DIGEST_STRETCHES to give
a 10ms encrypt time or so
   desired_encryption_expensiveness_ms = 0.1
   it "takes longer than #{desired_encryption_expensiveness_ms}ms to encrypt a
password" do
     test_reps = 100
     start_time = Time.now; test_reps.times{ User.authenticate('quentin',
'monkey'+rand.to_s) }; end_time   = Time.now
     auth_time_ms = 1000 * (end_time - start_time)/test_reps
     auth_time_ms.should > desired_encryption_expensiveness_ms
   end
 end

  #
  # Authentication
  #

  it 'sets remember token' do
    users(:quentin).remember_me
    users(:quentin).remember_token.should_not be_nil
    users(:quentin).remember_token_expires_at.should_not be_nil
  end

  it 'unsets remember token' do
    users(:quentin).remember_me
    users(:quentin).remember_token.should_not be_nil
    users(:quentin).forget_me
    users(:quentin).remember_token.should be_nil
  end

  it 'remembers me for one week' do
    before = 1.week.from_now.utc
    users(:quentin).remember_me_for 1.week
    after = 1.week.from_now.utc
    users(:quentin).remember_token.should_not be_nil
    users(:quentin).remember_token_expires_at.should_not be_nil
    users(:quentin).remember_token_expires_at.between?(before, after).should
be_true
  end

  it 'remembers me until one week' do
    time = 1.week.from_now.utc
    users(:quentin).remember_me_until time
    users(:quentin).remember_token.should_not be_nil
    users(:quentin).remember_token_expires_at.should_not be_nil
    users(:quentin).remember_token_expires_at.should == time
  end

  it 'remembers me default two weeks' do
```

```
    before = 2.weeks.from_now.utc
    users(:quentin).remember_me
    after = 2.weeks.from_now.utc
    users(:quentin).remember_token.should_not be_nil
    users(:quentin).remember_token_expires_at.should_not be_nil
    users(:quentin).remember_token_expires_at.between?(before, after).should
be_true
  end

  it 'registers passive user' do
    user = create_user(:password => nil, :password_confirmation => nil)
    user.should be_passive
    user.update_attributes(:password => 'new password', :password_confirmation
=> 'new password')
    user.register!
    user.should be_pending
  end

  it 'suspends user' do
    users(:quentin).suspend!
    users(:quentin).should be_suspended
  end

  it 'does not authenticate suspended user' do
    users(:quentin).suspend!
    User.authenticate('quentin', 'monkey').should_not == users(:quentin)
  end

  it 'deletes user' do
    users(:quentin).deleted_at.should be_nil
    users(:quentin).delete!
    users(:quentin).deleted_at.should_not be_nil
    users(:quentin).should be_deleted
  end

  describe "being unsuspended" do
    fixtures :users

    before do
      @user = users(:quentin)
      @user.suspend!
    end

    it 'reverts to active state' do
      @user.unsuspend!
      @user.should be_active
    end

    it 'reverts to passive state if activation_code and activated_at are nil' do
      User.update_all :activation_code => nil, :activated_at => nil
      @user.reload.unsuspend!
      @user.should be_passive
    end
```

```
    it 'reverts to pending state if activation_code is set and activated_at is
nil' do
      User.update_all :activation_code => 'foo-bar', :activated_at => nil
      @user.reload.unsuspend!
      @user.should be_pending
    end
  end

  describe 'destroy data' do
    fixtures :users, :items, :questions, :items_questions, :rank_algorithms,
:voters, :features, :prompt_requests, :prompts, :stats, :items_stats

    before do
      @user = User.find(1)
      @voters = @user.voters
      @items = @user.items
      @questions = @user.questions
      @features = @voters.map(&:features).flatten
      @prompt_requests = @questions.map(&:prompt_requests).flatten |
@items.map(&:prompt_requests).flatten
      @stats = @items.map(&:stats).flatten
      @iqs = @questions.map(&:items_questions).flatten |
@items.map(&:items_questions).flatten
      @prompts = @items.map(&:prompts).flatten
      @votes = @items.map(&:votes).flatten | @prompts.map(&:vote).flatten
      @user.destroy_data
    end

#    it 'should delete voters' do
#      Voter.all(:conditions => { :id => @voters.map(&:id) }).empty?.should ==
true
#      @user.reload.voters.empty?.should == true
#    end

    it 'should delete items' do
      Item.all(:conditions => { :id => @items.map(&:id) }).empty?.should == true
      @user.reload.items.empty?.should == true
    end

    it 'should delete questions' do
      Question.all(:conditions => { :id => @questions.map(&:id) }).empty?.should
== true
      @user.reload.questions.empty?.should == true
    end

#    it 'should delete features' do
#      Feature.all(:conditions => { :id => @features.map(&:id) }).empty?.should
== true
#    end

    it 'should delete prompt_requests' do
      PromptRequest.all(:conditions => { :id => @prompt_requests.map(&:id)
}).empty?.should == true
    end
```

```
    it 'should delete items questions' do
      ItemsQuestion.all(:conditions => { :id => @iqs.map(&:id) }).empty?.should
== true
    end

    it 'should delete votes' do
      Vote.all(:conditions => { :id => @votes.map(&:id) }).empty?.should == true
    end

    it 'should delete prompts' do
      Prompt.all(:conditions => { :id => @prompts.map(&:id) }).empty?.should ==
true
    end
  end

protected
  def create_user(options = {})
    record = User.new({ :login => 'quire', :email => 'quire@example.com',
:password => 'quire69', :password_confirmation => 'quire69' }.merge(options))
    record.register! if record.valid?
    record
  end
end
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Vote do
  before(:each) do
    @valid_attributes = {
      :prompt_id => 1,
      :voter_id => 1
    }
  end

  it "should create a new instance given valid attributes" do
    Vote.create!(@valid_attributes)
  end
endrequire File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Voter do
  before(:each) do
    @valid_attributes = {
      :user_id => 1
    }
  end

  it "should create a new instance given valid attributes" do
    Voter.create!(@valid_attributes)
  end
end
# This file is copied to ~/spec when you run 'ruby script/generate rspec'
# from the project root directory.
ENV["RAILS_ENV"] = "test"
require File.expand_path(File.dirname(__FILE__) + "/../config/environment")
require 'spec'
require 'spec/rails'
```

```
include AuthenticatedTestHelper
include AuthenticatedSystem

Spec::Runner.configure do |config|
  # If you're not using ActiveRecord you should remove these
  # lines, delete config/database.yml and disable :active_record
  # in your config/boot.rb
  config.use_transactional_fixtures = false
  config.use_instantiated_fixtures  = false
  config.fixture_path = RAILS_ROOT + '/spec/fixtures/'

  # == Fixtures
  #
  # You can declare fixtures for each example_group like this:
  #   describe "...." do
  #     fixtures :table_a, :table_b
  #
  # Alternatively, if you prefer to declare them only once, you can
  # do so right here. Just uncomment the next line and replace the fixture
  # names with your fixtures.
  #
  # config.global_fixtures = :table_a, :table_b
  #
  # If you declare global fixtures, be aware that they will be declared
  # for all of your examples, even those that don't use them.
  #
  # You can also declare which fixtures to use (for example fixtures for
test/fixtures):
  #
  # config.fixture_path = RAILS_ROOT + '/spec/fixtures/'
  #
  # == Mock Framework
  #
  # RSpec uses it's own mocking framework by default. If you prefer to
  # use mocha, flexmock or RR, uncomment the appropriate line:
  #
  # config.mock_with :mocha
  # config.mock_with :flexmock
  # config.mock_with :rr
  #
  # == Notes
  #
  # For more information take a look at Spec::Example::Configuration and
Spec::Runner
end

def xml_post(to, file, params = {})
  request.env['RAW_POST_DATA'] =
File.read("#{RAILS_ROOT}/spec/fixtures/xml/#{file}.xml")
  post to, params.merge(:content_type => 'application/xml')
enddir = File.dirname(__FILE__)
Dir[File.expand_path("#{dir}/**/*.rb")].uniq.each do |file|
  require file
endENV["RAILS_ENV"] = "test"
```

```
require File.expand_path(File.dirname(__FILE__) + "/../config/environment")
require 'spec/rails/story_adapter'#!/usr/bin/env ruby
ENV["RAILS_ENV"] = "test"
require File.expand_path(File.dirname(__FILE__) + "/../config/environment")
require 'spec/rails/story_adapter'
require 'spec/story'
require File.expand_path(File.dirname(__FILE__) +
"/rest_auth_stories_helper.rb")

# Make visible for testing
ApplicationController.send(:public, :logged_in?, :current_user, :authorized?)

this_dir = File.dirname(__FILE__)
Dir[File.join(this_dir, "steps/*.rb")].each do |file|
  puts file.to_s
  require file
end

with_steps_for :ra_navigation, :ra_response, :ra_resource, :user do
  story_files = Dir[File.join(this_dir, "users", '*.story')]
  story_files.each do |file|
    run file, :type => RailsStory
  end
end
# If you have a global stories helper, move this line there:
include AuthenticatedTestHelper

# Most of the below came out of code from Ben Mabey
# http://www.benmabey.com/2008/02/04/rspec-plain-text-stories-webrat-chunky-
bacon/

# These allow exceptions to come through as opposed to being caught and having
non-helpful responses returned.
ActionController::Base.class_eval do
  def perform_action
    perform_action_without_rescue
  end
end
Dispatcher.class_eval do
  def self.failsafe_response(output, status, exception = nil)
    raise exception
  end
end

#
# Sugar for turning a story's attribute list into list, array, etc.
#
module ToFooFromStory
  def ToFooFromStory.fix_key key
    key.downcase.gsub(/\s+/, '_')
  end
  def ToFooFromStory.fix_value value
    return '' if !value
    value.strip!
    case
```

```
    when value =~ /^'(.*)'$/    then value = $1
    when value =~ /^"(.*)"$/    then value = $1
    when value == 'nil!'        then value = nil
    when value == 'non-nil!'    then value = be_nil
    when value =~ /^#\{(.*)\}$/ then value = eval($1)
    end
    value
  end
  # Converts a key: value list found in the steps into a hash.
  # Example:
  #   ISBN: '0967539854' and comment: 'I love this book' and Quality rating: '4'
  #   # => {"quality_rating"=>"4", "isbn"=>"0967539854", "comment"=>"I love this
book"}
  def to_hash_from_story
    hsh = self.split(/,? and |, /).inject({}) do |hash_so_far, key_value|
      key, value = key_value.split(":")
      if !value then warn "Couldn't understand story '#{self}': only understood
up to the part '#{hash_so_far.to_yaml}'" end
      hash_so_far.merge(ToFooFromStory::fix_key(key) =>
ToFooFromStory::fix_value(value))
    end
  end
  # Coverts an attribute list found in the steps into an array
  # Example:
  #   login, email, updated_at, and gravatar
  #   # => ['login', 'email', 'updated_at', 'gravatar']
  def to_array_from_story
    self.split(/,? and |, /).map do |value|
      ToFooFromStory::fix_value(value)
    end
  end
end
class String
  include ToFooFromStory
end

def instantize(string)
  instance_variable_get("@#{string}")
end

#
# Spew response onto screen -- painful but scrolling >> debugger
#
def dump_response
  # note that @request and @template won't to_yaml and that @session includes
@cgi
  response_methods = response.instance_variables         - ['@request',
'@template', '@cgi']
  request_methods  = response.request.instance_variables -
['@session_options_with_string_keys', '@cgi', '@session']
  response_methods.map!{|attr| attr.gsub(/^@/,'')}.sort!
  request_methods.map!{ |attr| attr.gsub(/^@/,'')}.sort!
  puts '', '*' * 75,
    response.instance_values.slice(*response_methods).to_yaml,
    "*" * 75, '',
```

```
    response.request.instance_values.slice(*request_methods).to_yaml,
    "*" * 75, ''
end
#
# Where to go
#
steps_for(:ra_navigation) do
  #
  # GET
  # Go to a given page.
  When "$actor goes to $path" do |actor, path|
    case path
    when 'the home page' then get '/'
    else                      get path
    end
  end

  # POST -- Ex:
  #   When she creates a book with ISBN: '0967539854' and comment: 'I love this
book' and rating: '4'
  #   When she creates a singular session with login: 'reggie' and password:
'i_haxxor_joo'
  # Since I'm not smrt enough to do it right, explicitly specify singular
resources
  When %r{$actor creates an? $resource with $attributes} do |actor, resource,
attributes|
    attributes = attributes.to_hash_from_story
    if resource =~ %r{singular ([\w/]+)}
      resource = $1.downcase.singularize
      post "/#{resource}", attributes
    else
      post "/#{resource.downcase.pluralize}", { resource.downcase.singularize =>
attributes }
    end
  end

  # PUT
  When %r{$actor asks to update '$resource' with $attributes} do |_, resource,
attributes|
    attributes = attributes.to_hash_from_story
    put "#{resource}", attributes
    dump_response
  end

  # DELETE -- Slap together the POST-form-as-fake-HTTP-DELETE submission
  When %r{$actor asks to delete '$resource'} do |_, resource|
    post "/#{resource.downcase.pluralize}", { :_method => :delete }
    dump_response
  end


  # Redirect --
  #   Rather than coding in get/get_via_redirect's and past/p_v_r's,
  #   let's just demand that in the story itself.
  When "$actor follows that redirect!" do |actor|
```

```
    follow_redirect!
  end
end
# The flexible code for resource testing came out of code from Ben Mabey
# http://www.benmabey.com/2008/02/04/rspec-plain-text-stories-webrat-chunky-
bacon/
steps_for(:ra_resource) do
  #
  # Construct resources
  #

  #
  # Build a resource as described, store it as an @instance variable. Ex:
  #   "Given a user with login: 'mojojojo'"
  # produces a User instance stored in @user with 'mojojojo' as its login
  # attribute.
  #
  Given "a $resource instance with $attributes" do |resource, attributes|
    klass, instance, attributes = parse_resource_args resource, attributes
    instance = klass.new(attributes)
    instance.save!
    find_resource(resource, attributes).should_not be_nil
    keep_instance! resource, instance
  end

  #
  # Stuff attributes into a preexisting @resource
  #   "And the user has thac0: 3"
  # takes the earlier-defined @user instance and sets its thac0 to '3'.
  #
  Given "the $resource has $attributes" do |resource, attributes|
    klass, instance, attributes = parse_resource_args resource, attributes
    attributes.each do |attr, val|
      instance.send("#{attr}=", val)
    end
    instance.save!
    find_resource(resource, attributes).should_not be_nil
    keep_instance! resource, instance
  end

  #
  # Destroy all for this resource
  #
  Given "no $resource with $attr: '$val' exists" do |resource, attr, val|
    klass, instance = parse_resource_args resource
    klass.destroy_all(attr.to_sym => val)
    instance = find_resource resource, attr.to_sym => val
    instance.should be_nil
    keep_instance! resource, instance
  end

  #
  # Then's for resources
  #
```

```
  # Resource like this DOES exist
  Then %r{an? $resource with $attributes should exist} do |resource, attributes|
    instance = find_resource resource, attributes
    instance.should_not be_nil
    keep_instance! resource, instance
  end
  # Resource like this DOES NOT exist
  Then %r{no $resource with $attributes should exist} do |resource, attributes|
    instance = find_resource resource, attributes
    instance.should be_nil
  end

  # Resource has attributes with given values
  Then  "the $resource should have $attributes" do |resource, attributes|
    klass, instance, attributes = parse_resource_args resource, attributes
    attributes.each do |attr, val|
      instance.send(attr).should == val
    end
  end
  # Resource attributes should / should not be nil
  Then  "the $resource's $attr should be nil" do |resource, attr|
    klass, instance = parse_resource_args resource
    instance.send(attr).should be_nil
  end
  Then  "the $resource's $attr should not be nil" do |resource, attr|
    klass, instance = parse_resource_args resource
    instance.send(attr).should_not be_nil
  end

  #
  # Bank each of the @resource's listed attributes for later.
  #
  Given "we try hard to remember the $resource's $attributes" do |resource,
attributes|
    attributes = attributes.to_array_from_story
    attributes.each do |attr|
      memorize_resource_value resource, attr
    end
  end
  #
  # Bank each of the @resource's listed attributes for later.
  #
  Given "we don't remember anything about the past" do
    memorize_forget_all!
  end

  #
  # Compare @resource.attr to its earlier-memorized value.
  # Specify ' using method_name' (abs, to_s, &c) to coerce before comparing.
  # For important and mysterious reasons, timestamps want to_i or to_s.
  #
  Then %r{the $resource\'s $attribute should stay the same(?: under $func)?} do
|resource, attr, func|
    klass, instance = parse_resource_args resource
    # Get the values
```

```
    old_value = recall_resource_value(resource, attr)
    new_value = instance.send(attr)
    # Transform each value, maybe, using value.func
    if func then new_value = new_value.send(func); old_value =
old_value.send(func) end
    # Compare
    old_value.should eql(new_value)
  end

  #
  # Look for each for the given attributes in the page's text
  #
  Then "page should have the $resource's $attributes" do |resource, attributes|
    actual_resource = instantize(resource)
    attributes.split(/, and |, /).each do |attribute|
      response.should have_text(/#{actual_resource.send(attribute.strip.gsub("
","_"))}/)
    end
  end

end

#
# Turn a resource name and a to_hash_from_story string like
#   "attr: 'value', attr2: 'value2', ... , and attrN: 'valueN'"
# into
#   * klass      -- the class matching that Resource
#   * instance   -- the possibly-preexisting local instance value @resource
#   * attributes -- a hash matching the given attribute-list string
#
def parse_resource_args resource, attributes=nil
  instance   = instantize resource
  klass      = resource.classify.constantize
  attributes = attributes.to_hash_from_story if attributes
  [klass, instance, attributes]
end

#
# Given a class name 'resource' and a hash of conditsion, find a model
#
def find_resource resource, conditions
  klass, instance = parse_resource_args resource
  conditions = conditions.to_hash_from_story unless (conditions.is_a? Hash)
  klass.find(:first, :conditions => conditions)
end

#
# Simple, brittle, useful: store the given resource's attribute
# so we can compare it later.
#
def memorize_resource_value resource, attr
  klass, instance = parse_resource_args resource
  value = instance.send(attr)
  @_memorized ||= {}
  @_memorized[resource] ||= {}
```

```
  @_memorized[resource][attr] = value
  value
end
def recall_resource_value resource, attr
  @_memorized[resource][attr]
end
def memorize_forget_all!
  @_memorized = {}
end

#
# Keep the object around in a local instance variable @resource.
#
# So, for instance,
#   klass, instance = parse_resource_args 'user'
#   instance = klass.new({login => 'me', password => 'monkey', ...})
#   keep_instance! resource, instance
# keeps the just-constructed User model in the @user instance variable.
#
def keep_instance! resource, object
  instance_variable_set("@#{resource}", object)
end
#
# What you should see when you get there
#

steps_for(:ra_response) do
  #
  # Destinations.  Ex:
  #   She should be at the new kids page
  #   Tarkin should be at the destroy alderaan page
  #   The visitor should be at the '/lolcats/download' form
  #   The visitor should be redirected to '/hi/mom'
  #
  # It doesn't know anything about actual routes -- it just
  # feeds its output to render_template or redirect_to
  #
  Then "$actor should be at $path" do |_, path|
    response.should render_template(grok_path(path))
  end

  Then "$actor should be redirected to $path" do |_, path|
    response.should redirect_to(grok_path(path))
  end

  Then "the page should look AWESOME" do
    response.should have_tag('head>title')
    response.should have_tag('h1')
    # response.should be_valid_xhtml
  end

  #
  # Tags
  #
```

```
  Then "the page should contain '$text'" do |_, text|
    response.should have_text(/#{text}/)
  end

  # please note: this enforces the use of a <label> field
  Then "$actor should see a <$container> containing a $attributes" do |_,
container, attributes|
    attributes = attributes.to_hash_from_story
    response.should have_tag(container) do
      attributes.each do |tag, label|
        case tag
        when "textfield" then with_tag "input[type='text']";
with_tag("label", label)
        when "password"  then with_tag "input[type='password']";
with_tag("label", label)
        when "submit"    then with_tag "input[type='submit'][value='#{label}']"
        else with_tag tag, label
        end
      end
    end
  end

  #
  # Session, cookie variables
  #
  Then "$actor $token cookie should include $attrlist" do |_, token, attrlist|
    attrlist = attrlist.to_array_from_story
    cookies.include?(token).should be_true
    attrlist.each do |val|
      cookies[token].include?(val).should be_true
    end
  end

  Then "$actor $token cookie should exist but not include $attrlist" do |_,
token, attrlist|
    attrlist = attrlist.to_array_from_story
    cookies.include?(token).should be_true
    puts [cookies, attrlist, token].to_yaml
    attrlist.each do |val|
      cookies[token].include?(val).should_not be_true
    end
  end

  Then "$actor should have $an $token cookie" do |_, _, token|
    cookies[token].should_not be_blank
  end
  Then "$actor should not have $an $token cookie" do |_, _, token|
    cookies[token].should be_blank
  end

  Given "$actor has $an cookie jar with $attributes" do |_, _, attributes|
    attributes = attributes.to_hash_from_story
    attributes.each do |attr, val|
      cookies[attr] = val
    end
```

```
  end
  Given "$actor session store has no $attrlist" do |_, attrlist|
    attrlist = attrlist.to_array_from_story
    attrlist.each do |attr|
      # Note that the comparison passes through 'to_s'
      session[attr.to_sym] = nil
    end
  end

  Then "$actor session store should have $attributes" do |_, attributes|
    attributes = attributes.to_hash_from_story
    attributes.each do |attr, val|
      # Note that the comparison passes through 'to_s'
      session[attr.to_sym].to_s.should eql(val)
    end
  end

  Then "$actor session store should not have $attrlist" do |_, attrlist|
    attrlist = attrlist.to_array_from_story
    attrlist.each do |attr|
      session[attr.to_sym].blank?.should be_true
    end
  end

  #
  # Flash messages
  #

  Then "$actor should see $an $notice message '$message'" do |_, _, notice,
message|
    response.should have_flash(notice, %r{#{message}})
  end

  Then "$actor should not see $an $notice message '$message'" do |_, _, notice,
message|
    response.should_not have_flash(notice, %r{#{message}})
  end

  Then "$actor should see no messages" do |_|
    ['error', 'warning', 'notice'].each do |notice|
      response.should_not have_flash(notice)
    end
  end

  RE_POLITENESS = /(?:please|sorry|thank(?:s| you))/i
  Then %r{we should be polite about it} do
    response.should have_tag("div.error,div.notice", RE_POLITENESS)
  end
  Then %r{we should not even be polite about it} do
    response.should_not have_tag("div.error,div.notice", RE_POLITENESS)
  end

  #
  # Resource's attributes
  #
```

```
  # "Then page should have the $resource's $attributes" is in resource_steps

  # helpful debug step
  Then "we dump the response" do
    dump_response
  end
end


def have_flash notice, *args
  have_tag("div.#{notice}", *args)
end

RE_PRETTY_RESOURCE = /the (index|show|new|create|edit|update|destroy) (\w+)
(page|form)/i
RE_THE_FOO_PAGE    = /the '?([^']*)'? (page|form)/i
RE_QUOTED_PATH     = /^'([^']*)'$/i
def grok_path path
  path.gsub(/\s+again$/,'') # strip trailing ' again'
  case
  when path == 'the home page'    then dest = '/'
  when path =~ RE_PRETTY_RESOURCE then dest = template_for $1, $2
  when path =~ RE_THE_FOO_PAGE    then dest = $1
  when path =~ RE_QUOTED_PATH     then dest = $1
  else                                 dest = path
  end
  dest
end

# turns 'new', 'road bikes' into 'road_bikes/new'
# note that it's "action resource"
def template_for(action, resource)
  "#{resource.gsub(" ","_")}/#{action}"
end

require File.dirname(__FILE__) + '/../helper'

RE_User      = %r{(?:(?:the )? *(\w+) *)}
RE_User_TYPE = %r{(?: *(\w+)? *)}
steps_for(:user) do

  #
  # Setting
  #

  Given "an anonymous user" do
    log_out!
  end

  Given "$an $user_type user with $attributes" do |_, user_type, attributes|
    create_user! user_type, attributes.to_hash_from_story
  end

  Given "$an $user_type user named '$login'" do |_, user_type, login|
    create_user! user_type, named_user(login)
```

```
  end

  Given "$an $user_type user logged in as '$login'" do |_, user_type, login|
    create_user! user_type, named_user(login)
    log_in_user!
  end

  Given "$actor is logged in" do |_, login|
    log_in_user! @user_params || named_user(login)
  end

  Given "there is no $user_type user named '$login'" do |_, login|
    @user = User.find_by_login(login)
    @user.destroy! if @user
    @user.should be_nil
  end

  #
  # Actions
  #
  When "$actor logs out" do
    log_out
  end

  When "$actor registers an account as the preloaded '$login'" do |_, login|
    user = named_user(login)
    user['password_confirmation'] = user['password']
    create_user user
  end

  When "$actor registers an account with $attributes" do |_, attributes|
    create_user attributes.to_hash_from_story
  end

  When "$actor activates with activation code $attributes" do |_,
activation_code|
    activation_code = '' if activation_code == 'that is blank'
    activate
  end

  When "$actor logs in with $attributes" do |_, attributes|
    log_in_user attributes.to_hash_from_story
  end

  #
  # Result
  #
  Then "$actor should be invited to sign in" do |_|
    response.should render_template('/sessions/new')
  end

  Then "$actor should not be logged in" do |_|
    controller.logged_in?.should_not be_true
  end
```

```
  Then "$login should be logged in" do |login|
    controller.logged_in?.should be_true
    controller.current_user.should === @user
    controller.current_user.login.should == login
  end

end

def named_user login
  user_params = {
    'admin'   => {'id' => 1, 'login' => 'addie', 'password' => '1234addie',
'email' => 'admin@example.com',       },
    'oona'    => {          'login' => 'oona',   'password' => '1234oona',
'email' => 'unactivated@example.com'},
    'reggie'  => {          'login' => 'reggie', 'password' => 'monkey',
'email' => 'registered@example.com' },
    }
  user_params[login.downcase]
end

#
# User account actions.
#
# The ! methods are 'just get the job done'.  It's true, they do some testing of
# their own -- thus un-DRY'ing tests that do and should live in the user account
# stories -- but the repetition is ultimately important so that a faulty test
setup
# fails early.
#

def log_out
  get '/sessions/destroy'
end

def log_out!
  log_out
  response.should redirect_to('/')
  follow_redirect!
end

def create_user(user_params={})
  @user_params       ||= user_params
  post "/users", :user => user_params
  @user = User.find_by_login(user_params['login'])
end

def create_user!(user_type, user_params)
  user_params['password_confirmation'] ||= user_params['password'] ||=
user_params['password']
  create_user user_params
  response.should redirect_to('/')
  follow_redirect!

  # fix the user's activation status
  activate_user! if user_type == 'activated'
```

```
end


def activate_user activation_code=nil
  activation_code = @user.activation_code if activation_code.nil?
  get "/activate/#{activation_code}"
end

def activate_user! *args
  activate_user *args
  response.should redirect_to('/login')
  follow_redirect!
  response.should have_flash("notice", /Signup complete!/)
end

def log_in_user user_params=nil
  @user_params ||= user_params
  user_params  ||= @user_params
  post "/session", user_params
  @user = User.find_by_login(user_params['login'])
  controller.current_user
end

def log_in_user! *args
  log_in_user *args
  response.should redirect_to('/')
  follow_redirect!
  response.should have_flash("notice", /Logged in successfully/)
end
ENV["RAILS_ENV"] = "test"
require File.expand_path(File.dirname(__FILE__) + "/../config/environment")
require 'test_help'

class Test::Unit::TestCase
  # Transactional fixtures accelerate your tests by wrapping each test method
  # in a transaction that's rolled back on completion.  This ensures that the
  # test database remains unchanged so your fixtures don't have to be reloaded
  # between every test method.  Fewer database queries means faster tests.
  #
  # Read Mike Clark's excellent walkthrough at
  #   http://clarkware.com/cgi/blosxom/2005/10/24#Rails10FastTesting
  #
  # Every Active Record database supports transactions except MyISAM tables
  # in MySQL.  Turn off transactional fixtures in this case; however, if you
  # don't care one way or the other, switching from MyISAM to InnoDB tables
  # is recommended.
  #
  # The only drawback to using transactional fixtures is when you actually
  # need to test transactions.  Since your test is bracketed by a transaction,
  # any transactions started in your code will be automatically rolled back.
  self.use_transactional_fixtures = true

  # Instantiated fixtures are slow, but give you @david where otherwise you
  # would need people(:david).  If you don't want to migrate your existing
  # test cases which use the @david style and don't mind the speed hit (each
```

```
  # instantiated fixtures translates to a database query per test method),
  # then set this back to true.
  self.use_instantiated_fixtures  = false

  # Setup all fixtures in test/fixtures/*.(yml|csv) for all tests in
alphabetical order.
  #
  # Note: You'll currently still have to declare fixtures explicitly in
integration tests
  # -- they do not yet inherit this setting
  fixtures :all

  # Add more helper methods to be used by all tests here...
end
require 'test_helper'

class SystemNotifierTest < ActionMailer::TestCase
  # replace this with your real tests
  test "the truth" do
    assert true
  end
end
require_dependency 'smtp_tls'
require "openssl"
require "net/smtp"

Net::SMTP.class_eval do
  private
  def do_start(helodomain, user, secret, authtype)
    raise IOError, 'SMTP session already started' if @started
    check_auth_args user, secret, authtype if user or secret

    sock = timeout(@open_timeout) { TCPSocket.open(@address, @port) }
    @socket = Net::InternetMessageIO.new(sock)
    @socket.read_timeout = 60 #@read_timeout

    check_response(critical { recv_response() })
    do_helo(helodomain)

    if starttls
      raise 'openssl library not installed' unless defined?(OpenSSL)
      ssl = OpenSSL::SSL::SSLSocket.new(sock)
      ssl.sync_close = true
      ssl.connect
      @socket = Net::InternetMessageIO.new(ssl)
      @socket.read_timeout = 60 #@read_timeout
      do_helo(helodomain)
    end

    authenticate user, secret, authtype if user
    @started = true
  ensure
    unless @started
      # authentication failed, cancel connection.
      @socket.close if not @started and @socket and not @socket.closed?
```

```
      @socket = nil
    end
  end

  def do_helo(helodomain)
    begin
      if @esmtp
        ehlo helodomain
      else
        helo helodomain
      end
    rescue Net::ProtocolError
      if @esmtp
        @esmtp = false
        @error_occured = false
        retry
      end
      raise
    end
  end

  def starttls
    getok('STARTTLS') rescue return false
    return true
  end

  def quit
    begin
      getok('QUIT')
    rescue EOFError
    end
  end
end#
# For Rails 2.x:
#   A copy of this file should be placed in RAILS_ROOT/initializers/
#   A file named mailer.yml should be placed in RAILS_ROOT/config/
#     See mailer.yml.sample
#

require "smtp_tls"

mailer_config = File.open("#{RAILS_ROOT}/config/mailer.yml")
mailer_options = YAML.load(mailer_config)
ActionMailer::Base.smtp_settings = mailer_options
begin
  require File.join(File.dirname(__FILE__), 'lib', 'haml') # From here
rescue LoadError
  require 'haml' # From gem
end

# Load Haml and Sass
Haml.init_rails(binding)
#!/usr/bin/env ruby

require 'benchmark'
```

```
require File.dirname(__FILE__) + "/../lib/oink.rb"

Benchmark.bmbm(15) do |x|
  x.report("Running Oink") {
    f = File.open(File.expand_path(File.dirname(__FILE__) +
"/../logs/production.log"))
    Oink::MemoryUsageReporter.new([f], 75*1024).print(STDOUT)
    f.close
  }
end

require 'activerecord'

ActiveRecord::Base.establish_connection(
  :adapter => 'sqlite3',
  :dbfile => ':memory:'
)ActiveRecord::Schema.define(:version => 1) do

  create_table "pigs", :force => true do |t|
    t.integer "pen_id"
    t.string "name"
    t.boolean "smells"
  end

  create_table "pens", :force => true do |t|
    t.string "location"
  end

endrequire "oink/rails/memory_usage_logger"
require "oink/rails/instance_type_counter"#!/usr/bin/env ruby
require 'ftools'

template_executable_file = File.join("application_files", "script", "oink")
executable_file =
File.expand_path("#{File.dirname(__FILE__)}/../../../script/oink")

File.copy template_executable_file, executable_file
File.chmod 0755, executable_file
require "date"
require "oink/base"
require "oink/oinked_request/oinked_ar_request"
require "oink/priority_queue"

module Oink

  class ActiveRecordInstantiationReporter < Base

    def print(output)
      output.puts "---- OINK FOR ACTIVERECORD ----"
      output.puts "THRESHOLD: #{@threshold} Active Record objects per request\n"

      output.puts "\n-- REQUESTS --\n" if @format == :verbose

      @inputs.each do |input|
```

```
        input.each_line do |line|
          line = line.strip

          if line =~ /rails\[(\d+)\]/
            pid = $1
            @pids[pid] ||= { :buffer => [], :ar_count => -1, :action => "",
:request_finished => true }
            @pids[pid][:buffer] << line
          end

          if line =~ /Processing ((\w+)#(\w+)) /

            @pids[pid][:action] = $1
            unless @pids[pid][:request_finished]
              @pids[pid][:buffer] = [line]
            end
            @pids[pid][:request_finished] = false

          elsif line =~ /Instantiation Breakdown: Total: (\d+)/

            @pids[pid][:ar_count] = $1.to_i

          elsif line =~ /Completed in/

            if @pids[pid][:ar_count] > @threshold
              @bad_actions[@pids[pid][:action]] ||= 0
              @bad_actions[@pids[pid][:action]] =
@bad_actions[@pids[pid][:action]] + 1
              date = /^(\w+ \d{2} \d{2}:\d{2}:\d{2})/.match(line).captures[0]
              @bad_requests.push(OinkedARRequest.new(@pids[pid][:action], date,
@pids[pid][:buffer], @pids[pid][:ar_count]))
              if @format == :verbose
                @pids[pid][:buffer].each { |b| output.puts b }
                output.puts "-------------------------------------------------
------------------"
              end
            end

            @pids[pid][:request_finished] = true
            @pids[pid][:buffer] = []
            @pids[pid][:ar_count] = -1

          end # end elsif
        end # end each_line
      end # end each input

      print_summary(output)

    end

  end

endmodule Oink

  class Base
```

```
    VERSION = '0.1.0'
    FORMATS = %w[verbose short-summary summary]
    FORMAT_ALIASES = { "v" => "verbose", "ss" => "short-summary", "s" =>
"summary" }

    def initialize(input, threshold, options = {})
      @inputs = Array(input)
      @threshold = threshold
      @format = options[:format] || :short_summary

      @pids = {}
      @bad_actions = {}
      @bad_requests = PriorityQueue.new(10)
    end

  protected

    def print_summary(output)
      output.puts "\n-- SUMMARY --\n"
      output.puts "Worst Requests:"
      @bad_requests.each_with_index do |offender, index|
        output.puts "#{index + 1}. #{offender.datetime},
#{offender.display_oink_number}, #{offender.action}"
        if @format == :summary
          offender.log_lines.each { |b| output.puts b }
          output.puts "-------------------------------------------------------
------------"
        end
      end
      output.puts "\nWorst Actions:"
      @bad_actions.sort{|a,b| b[1]<=>a[1]}.each { |elem|
        output.puts "#{elem[1]}, #{elem[0]}"
      }
    end

  end

endrequire 'optparse'

class Cli

  def initialize(args)
    @args = args
  end

  def process
    options = { :format => :short_summary, :type => :memory }

    op = OptionParser.new do |opts|
      opts.banner = "Usage: oink [options] files"

      opts.on("-t", "--threshold [INTEGER]", Integer,
              "Memory threshold in MB") do |threshold|
        options[:threshold] = threshold
```

```
      end

      opts.on("-f", "--file filepath", "Output to file") do |filename|
        options[:output_file] = filename
      end

      format_list = (Oink::MemoryUsageReporter::FORMAT_ALIASES.keys +
Oink::MemoryUsageReporter::FORMATS).join(',')
      opts.on("--format FORMAT", Oink::MemoryUsageReporter::FORMATS,
Oink::MemoryUsageReporter::FORMAT_ALIASES, "Select format",
              "  (#{format_list})") do |format|
        options[:format] = format.to_sym
      end

      opts.on("-m", "--memory", "Check for Memory Threshold (default)") do |v|
         options[:type] = :memory
      end

      opts.on("-r", "--active-record", "Check for Active Record Threshold") do
|v|
         options[:type] = :active_record
      end

    end

    op.parse!(@args)

    if @args.empty?
      puts op
      exit
    end

    output = nil

    if options[:output_file]
      output = File.open(options[:output_file], 'w')
    else
      output = STDOUT
    end

    files = get_file_listing(@args)

    handles = files.map { |f| File.open(f) }

    if options[:type] == :memory

      options[:threshold] ||= 75
      options[:threshold] *= 1024

      Oink::MemoryUsageReporter.new(handles, options[:threshold], :format =>
options[:format]).print(output)

    elsif options[:type] == :active_record

      options[:threshold] ||= 500
```

```
      Oink::ActiveRecordInstantiationReporter.new(handles, options[:threshold],
:format => options[:format]).print(output)

    end

    output.close
    handles.each { |h| h.close }
  end

protected

  def get_file_listing(args)
    listing = []
    args.each do |file|
      unless File.exist?(file)
        raise "Could not find \"#{file}\""
      end
      if File.directory?(file)
        listing += Dir.glob("#{file}/**")
      else
        listing << file
      end
    end
    listing
  end

end


require "date"
require "oink/base"
require "oink/oinked_request/oinked_memory_request"
require "oink/priority_queue"

module Oink

  class MemoryUsageReporter < Base

    def print(output)
      output.puts "---- MEMORY THRESHOLD ----"
      output.puts "THRESHOLD: #{@threshold/1024} MB\n"

      output.puts "\n-- REQUESTS --\n" if @format == :verbose

      @inputs.each do |input|
        input.each_line do |line|
          line = line.strip

          if line =~ /rails\[(\d+)\]/
            pid = $1
            @pids[pid] ||= { :buffer => [], :last_memory_reading => -1,
:current_memory_reading => -1, :action => "", :request_finished => true }
            @pids[pid][:buffer] << line
          end
```

```
          if line =~ /Processing ((\w+)#(\w+)) /

            unless @pids[pid][:request_finished]
              @pids[pid][:last_memory_reading] = -1
            end
            @pids[pid][:action] = $1
            @pids[pid][:request_finished] = false

          elsif line =~ /Memory usage: (\d+) /

            memory_reading = $1.to_i
            @pids[pid][:current_memory_reading] = memory_reading

          elsif line =~ /Completed in/

            @pids[pid][:request_finished] = true
            unless @pids[pid][:current_memory_reading] == -1 ||
@pids[pid][:last_memory_reading] == -1
              memory_diff = @pids[pid][:current_memory_reading] -
@pids[pid][:last_memory_reading]
              if memory_diff > @threshold
                @bad_actions[@pids[pid][:action]] ||= 0
                @bad_actions[@pids[pid][:action]] =
@bad_actions[@pids[pid][:action]] + 1
                date = /^(\w+ \d{2} \d{2}:\d{2}:\d{2})/.match(line).captures[0]
                @bad_requests.push(OinkedMemoryRequest.new(@pids[pid][:action],
date, @pids[pid][:buffer], memory_diff))
                if @format == :verbose
                  @pids[pid][:buffer].each { |b| output.puts b }
                  output.puts "----------------------------------------------------
--------------------"
                end
              end
            end

            @pids[pid][:buffer] = []
            @pids[pid][:last_memory_reading] =
@pids[pid][:current_memory_reading]
            @pids[pid][:current_memory_reading] = -1

          end # end elsif
        end # end each_line
      end # end each input

      print_summary(output)

    end

  end

endrequire "oink/oinked_request/oinked_request"

class OinkedARRequest < OinkedRequest
```

```
  def display_oink_number
    @oink_number
  end

endrequire "oink/oinked_request/oinked_request"

class OinkedMemoryRequest < OinkedRequest

  def display_oink_number
    "#{@oink_number} KB"
  end

endclass OinkedRequest

  attr_accessor :action, :datetime, :log_lines, :oink_number

  def initialize(action, datetime, log_lines, oink_number)
    @action = action
    @datetime = datetime
    @log_lines = log_lines
    @oink_number = oink_number
  end

  def <=>(other)
    self.oink_number <=> other.oink_number
  end

  def >(other)
    self.oink_number > other.oink_number
  end

endclass PriorityQueue

  include Enumerable

  def initialize(size)
    @size = size
    @queue = []
  end

  def push(item)
    if @queue.size < @size
      @queue << item
    elsif item > @queue.last
      @queue[@size - 1] = item
    end
    prioritize
  end

  def to_a
    @queue
  end

  def size
    @queue.size
```

```
  end

  def each
    @queue.each { |i| yield i }
  end

protected

  def prioritize
    @queue.sort! { |a, b| b <=> a }
  end

endmodule Oink
  module InstanceTypeCounter
    def self.included(klass)
      ActiveRecord::Base.send(:include, OinkInstanceTypeCounterInstanceMethods)

      klass.class_eval do
        after_filter :report_instance_type_count
      end
    end

    def before_report_active_record_count(instantiation_data)
    end

    private

      def report_instance_type_count
        report_hash = ActiveRecord::Base.instantiated_hash.merge("Total" =>
ActiveRecord::Base.total_objects_instantiated)
        breakdown = report_hash.sort{|a,b| b[1]<=>a[1]}.collect {|k,v| "#{k}:
#{v}" }.join(" | ")
        before_report_active_record_count(breakdown)
        if logger
          logger.info("Instantiation Breakdown: #{breakdown}")
        end
        ActiveRecord::Base.reset_instance_type_count
      end

  end

  module OinkInstanceTypeCounterInstanceMethods

    def self.included(klass)
      klass.class_eval do

        @@instantiated = {}
        @@total = nil

        def self.reset_instance_type_count
          @@instantiated = {}
          @@total = nil
        end

        def self.instantiated_hash
```

```
          @@instantiated
        end

        def self.total_objects_instantiated
          @@total ||= @@instantiated.inject(0) { |i, j| i + j.last }
        end

      end
    end

    def after_initialize
      @@instantiated[self.class.base_class.name] ||= 0
      @@instantiated[self.class.base_class.name] =
@@instantiated[self.class.base_class.name] + 1
    end

    def after_initialize_with_instance_type_count
      after_initialize_without_instance_type_count
      _instance_counter_after_initialize
    end
  end
endmodule Oink
  module MemoryUsageLogger
    def self.included(klass)
      klass.class_eval do
        after_filter :log_memory_usage
      end
    end

    private
      def log_memory_usage
        if logger
          memory_usage = `ps -o rss= -p #{$$}`.to_i
          logger.info("Memory usage: #{memory_usage} | PID: #{$$}")
        end
      end
  end
end$:.unshift(File.dirname(__FILE__ + '.rb') + '/../lib') unless
$:.include?(File.dirname(__FILE__ + '.rb') + '/../lib')

require "oink/memory_usage_reporter"
require "oink/active_record_instantiation_reporter"
require "oink/cli"require File.expand_path(File.dirname(__FILE__) +
"/../spec_helper")

describe Oink::ActiveRecordInstantiationReporter do

  describe "short summary with frequent offenders" do

    it "should report actions which exceed the threshold once" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
```

```
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output.should include("1, Users#show")
    end

    it "should not report actions which do not exceed the threshold" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
50 | User: 50
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output.should_not include("1, Users#show")
    end

    it "should report actions which exceed the threshold multiple times" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output.should include("2, Users#show")
    end

    it "should order actions by most exceeded" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Media#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
```

```
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Media#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output[-2].should == "2, Media#show"
      output[-1].should == "1, Users#show"
    end

    it "should not be bothered by incomplete requests" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Media#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
24 | User: 24
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Media#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output.should include("1, Users#show")
    end

  end

  describe "summary with top 10 offenses" do

    it "should only report requests over threshold" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
```

```
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output.should include("1. Feb 01 01:58:31, 51, Users#show")
    end

    it "should not include requests which are not over threshold" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
50 | User: 50
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output.should_not include("1. Feb 01 01:58:31, 50, Users#show")
    end

    it "should order offenses from biggest to smallest" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing DetailsController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
75 | User: 75
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:32 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:33 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
100 | User: 100
      Feb 01 01:58:34 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50).print(output)
      output[4].should == "1. Feb 01 01:58:34, 100, MediaController#show"
      output[5].should == "2. Feb 01 01:58:31, 75, DetailsController#show"
    end

  end

  describe "verbose format" do
    it "should print the full lines of actions exceeding the threshold" do
```

```
      str = <<-STR
      Feb 01 01:58:32 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:33 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
100 | User: 100
      Feb 01 01:58:34 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR
      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50, :format =>
:verbose).print(output)
      output[3..5].should == str.split("\n")[0..2].map { |o| o.strip }
    end

    it "should handle actions which do not complete properly" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Media#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
24 | User: 24
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Media#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
51 | User: 51
      Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new(io, 50, :format =>
:verbose).print(output)
      output[3..5].should == str.split("\n")[4..6].map { |o| o.strip }
    end
  end

  describe "multiple io streams" do
    it "should accept multiple files" do

      str1 = <<-STR
      Feb 01 01:58:32 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:33 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
100 | User: 100
      Feb 01 01:58:34 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      str2 = <<-STR
```

```
      Feb 01 01:58:32 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:33 ey04-s00297 rails[4413]: Instantiation Breakdown: Total:
100 | User: 100
      Feb 01 01:58:34 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io1 = StringIO.new(str1)
      io2 = StringIO.new(str2)
      output = PsuedoOutput.new
      Oink::ActiveRecordInstantiationReporter.new([io1, io2], 50).print(output)
      output.should include("2, MediaController#show")
    end

  end


end
require File.expand_path(File.dirname(__FILE__) + "/../spec_helper")

describe Oink::MemoryUsageReporter do

  TEN_MEGS = 10 * 1024

  describe "short summary with frequent offenders" do

    it "should report actions which exceed the threshold once" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
      output.should include("1, MediaController#show")
    end

    it "should not report actions which do not exceed the threshold" do
      threshold = 10

      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
```

```
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS} | PID:
4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
      output.should_not include("1, MediaController#show")
    end

    it "should report actions which exceed the threshold multiple times" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{(TEN_MEGS * 2) +
2} | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
      output.should include("2, MediaController#show")
    end

    it "should order actions by most exceeded" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
```

```
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{(TEN_MEGS * 2) +
2} | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{(TEN_MEGS * 3) +
3} | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
      output[-2].should == "2, MediaController#show"
      output[-1].should == "1, Users#show"
    end

    it "should not report actions which do not complete properly" do
      threshold = 10

      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
      output.should_not include("1, MediaController#show")
    end

    it "should not report actions from different pids" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[5513]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
```

```
      Feb 01 01:58:30 ey04-s00297 rails[5513]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[5513]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
      output.should_not include("1, MediaController#show")
    end

    describe "summary with top 10 offenses" do

      it "should only report requests over threshold" do
        str = <<-STR
        Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
        Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
        Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
        Feb 01 01:58:32 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
        Feb 01 01:58:33 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
        Feb 01 01:58:34 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
        STR

        io = StringIO.new(str)
        output = PsuedoOutput.new
        Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
        output.should include("1. Feb 01 01:58:34, #{TEN_MEGS + 1} KB,
MediaController#show")
      end

      it "should not include requests which are not over the threshold" do
        str = <<-STR
        Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
        Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
        Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
        Feb 01 01:58:32 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
        Feb 01 01:58:33 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS} |
PID: 4413
        Feb 01 01:58:34 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
        STR

        io = StringIO.new(str)
        output = PsuedoOutput.new
        Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
```

```
        output.should_not include("1. Feb 01 01:58:34, #{TEN_MEGS + 1} KB,
MediaController#show")
      end

      it "should order offenses from biggest to smallest" do
        str = <<-STR
        Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
        Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
        Feb 01 01:58:31 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
        Feb 01 01:58:32 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
        Feb 01 01:58:33 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
        Feb 01 01:58:34 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
        Feb 01 01:58:35 ey04-s00297 rails[4413]: Processing
DetailsController#show (for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
        Feb 01 01:58:36 ey04-s00297 rails[4413]: Memory usage: #{(TEN_MEGS * 2)
+ 2} | PID: 4413
        Feb 01 01:58:37 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
        STR

        io = StringIO.new(str)
        output = PsuedoOutput.new
        Oink::MemoryUsageReporter.new(io, TEN_MEGS).print(output)
        output[4].should == "1. Feb 01 01:58:34, #{TEN_MEGS + 1} KB,
MediaController#show"
        output[5].should == "2. Feb 01 01:58:37, #{TEN_MEGS + 1} KB,
DetailsController#show"
      end

    end

    # it "should report the time span" do
    #   str = <<-STR
    #   Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
    #   Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
    #   Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
    #   Mar 13 01:58:29 ey04-s00297 rails[5513]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
    #   Mar 13 01:58:30 ey04-s00297 rails[5513]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
    #   Mar 13 03:58:30 ey04-s00297 rails[5513]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
    #   STR
    #
    #   io = StringIO.new(str)
    #   output = PsuedoOutput.new
    #   Oink::MemoryUsageReporter.new(io, TEN_MEGS).each_line do |line|
    #     output << line
```

```
    #   end
    #   output.first.should == "Feb 01 01:58:29 - Mar 13 03:58:30"
    # end

  end

  describe "verbose format" do
    it "should print the full lines of actions exceeding the threshold" do
      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Parameters: {"id"=>"2332",
"controller"=>"users"}
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Parameters: {"id"=>"22900",
"controller"=>"media"}
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR
      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS, :format =>
:verbose).print(output)
      output[3..6].should == str.split("\n")[4..7].map { |o| o.strip }
    end

    it "should handle actions which do not complete properly" do
      threshold = 10

      str = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing ActorController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{(TEN_MEGS * 2) +
2} | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR
```

```
      io = StringIO.new(str)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new(io, TEN_MEGS, :format =>
:verbose).print(output)
      output[3..5].should == str.split("\n")[6..8].map { |o| o.strip }
    end
  end

  describe "multiple io streams" do
    it "should accept multiple files" do

      str1 = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      str2 = <<-STR
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing Users#show (for
92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: 0 | PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      Feb 01 01:58:29 ey04-s00297 rails[4413]: Processing MediaController#show
(for 92.84.151.171 at 2009-02-01 01:58:29) [GET]
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Memory usage: #{TEN_MEGS + 1} |
PID: 4413
      Feb 01 01:58:30 ey04-s00297 rails[4413]: Completed in 984ms (View: 840,
DB: 4) | 200 OK
      STR

      io1 = StringIO.new(str1)
      io2 = StringIO.new(str2)
      output = PsuedoOutput.new
      Oink::MemoryUsageReporter.new([io1, io2], TEN_MEGS).print(output)
      output.should include("2, MediaController#show")
    end

  end

end
require File.expand_path(File.dirname(__FILE__) + "/../spec_helper")

describe OinkedRequest do

  describe "sort" do
    it "should sort by memory used" do
```

```
      lr1 = OinkedRequest.new("Controller#Action", "February 1 10:20", [], 10)
      lr2 = OinkedRequest.new("Controller#Action", "February 1 10:20", [], 5)
      lr3 = OinkedRequest.new("Controller#Action", "February 1 10:20", [], 30)

      [lr1, lr2, lr3].sort.should == [lr2, lr1, lr3]
    end
  end


endrequire File.expand_path(File.dirname(__FILE__) + "/../spec_helper")

describe PriorityQueue do

  describe "size" do

    it "should report the right size" do
      pq = PriorityQueue.new(5)
      pq.push(1)
      pq.size.should == 1
      pq.push(2)
      pq.size.should == 2
    end

    it "should be limited to the size initialized with" do
      pq = PriorityQueue.new(5)
      pq.push(1)
      pq.push(2)
      pq.push(3)
      pq.push(4)
      pq.push(5)
      pq.push(6)
      pq.size.should == 5
    end

  end

  describe "order" do

    it "should be in order from highest to lowest" do
      pq = PriorityQueue.new(5)
      pq.push(1)
      pq.push(2)
      pq.push(3)
      pq.to_a.should == [3,2,1]
    end

    it "should throw out the lower value when adding a new value" do
      pq = PriorityQueue.new(3)
      pq.push(1)
      pq.push(2)
      pq.push(3)
      pq.push(4)
      pq.to_a.should == [4,3,2]
    end
```

```
    it "should not make it into the queue if it's smaller than the items in the
queue" do
      pq = PriorityQueue.new(3)
      pq.push(2)
      pq.push(3)
      pq.push(4)
      pq.push(1)
      pq.to_a.should == [4,3,2]
    end

  end

  describe "each" do
    it "should return each item in turn" do
      arr = []
      pq = PriorityQueue.new(5)
      pq.push(2)
      pq.push(3)
      pq.push(4)
      pq.push(1)
      pq.push(5)
      pq.each do |i|
        arr << i
      end
      arr.should == [5,4,3,2,1]
    end
  end


endrequire File.expand_path(File.dirname(__FILE__) + "/../spec_helper")

describe Oink::OinkInstanceTypeCounterInstanceMethods do
  before :each do
    ActiveRecord::Base.reset_instance_type_count
  end

  describe "hash" do
    it "should not count objects not instantiated" do
      ActiveRecord::Base.instantiated_hash["Pig"].should == nil
    end

    it "should include the objects instantiated" do
      Pig.create(:name => "Babe")
      Pig.first
      ActiveRecord::Base.instantiated_hash["Pig"].should == 2
    end

    it "should count instantiations for multiple classes" do
      Pig.create(:name => "Babe")
      Pen.create(:location => "Backyard")
      Pig.first
      ActiveRecord::Base.instantiated_hash["Pen"].should == 1
    end

    it "should report the total number of objects instantiated" do
```

```
      Pig.create(:name => "Babe")
      Pen.create(:location => "Backyard")
      Pig.first
      ActiveRecord::Base.total_objects_instantiated.should == 3
    end
  end

  describe "reset" do
    it "should reset the total count" do
      Pig.create(:name => "Babe")
      ActiveRecord::Base.instantiated_hash["Pig"].should == 1
      ActiveRecord::Base.reset_instance_type_count
      ActiveRecord::Base.total_objects_instantiated.should == 0
      Pig.create(:name => "Babe")
      ActiveRecord::Base.total_objects_instantiated.should == 0
    end
  end

endrequire "rubygems"
require "spec"
require 'ostruct'

dir = File.dirname(__FILE__)
require File.join(dir, "/../lib/oink.rb")
require "oink/rails/instance_type_counter"
require File.join(dir, '../config/environment')

class PsuedoOutput < Array

  def puts(line)
    self << line
  end

end

Spec::Runner.configure do |config|

  config.before :suite do
    load File.join(dir, "../db/schema.rb")
  end

  config.before :each do
    Pig.delete_all
    Pen.delete_all
  end

  config.before :suite do
    ActiveRecord::Base.send(:include,
Oink::OinkInstanceTypeCounterInstanceMethods)
    Pig = Class.new(ActiveRecord::Base)
    Pen = Class.new(ActiveRecord::Base)
    Pig.belongs_to :pen
  end

end#!/usr/bin/env ruby
```

```
executable_file =
File.expand_path("#{File.dirname(__FILE__)}/../../../script/oink")
File.delete(executable_file)require File.expand_path(File.dirname(__FILE__) +
"/lib/insert_routes.rb")
require 'digest/sha1'
class AuthenticatedGenerator < Rails::Generator::NamedBase
  default_options :skip_migration => false,
                  :skip_routes    => false,
                  :old_passwords  => false,
                  :include_activation => false

  attr_reader   :controller_name,
                :controller_class_path,
                :controller_file_path,
                :controller_class_nesting,
                :controller_class_nesting_depth,
                :controller_class_name,
                :controller_singular_name,
                :controller_plural_name,
                :controller_routing_name,                 # new_session_path
                :controller_routing_path,                 # /session/new
                :controller_controller_name,              # sessions
                :controller_file_name
  alias_method  :controller_table_name, :controller_plural_name
  attr_reader   :model_controller_name,
                :model_controller_class_path,
                :model_controller_file_path,
                :model_controller_class_nesting,
                :model_controller_class_nesting_depth,
                :model_controller_class_name,
                :model_controller_singular_name,
                :model_controller_plural_name,
                :model_controller_routing_name,           # new_user_path
                :model_controller_routing_path,           # /users/new
                :model_controller_controller_name         # users
  alias_method  :model_controller_file_name,  :model_controller_singular_name
  alias_method  :model_controller_table_name, :model_controller_plural_name

  def initialize(runtime_args, runtime_options = {})
    super

    @rspec = has_rspec?

    @controller_name = (args.shift || 'sessions').pluralize
    @model_controller_name = @name.pluralize

    # sessions controller
    base_name, @controller_class_path, @controller_file_path,
@controller_class_nesting, @controller_class_nesting_depth =
extract_modules(@controller_name)
    @controller_class_name_without_nesting, @controller_file_name,
@controller_plural_name = inflect_names(base_name)
    @controller_singular_name = @controller_file_name.singularize
    if @controller_class_nesting.empty?
```

```
      @controller_class_name = @controller_class_name_without_nesting
    else
      @controller_class_name =
"#{@controller_class_nesting}::#{@controller_class_name_without_nesting}"
    end
    @controller_routing_name  = @controller_singular_name
    @controller_routing_path  = @controller_file_path.singularize
    @controller_controller_name = @controller_plural_name

    # model controller
    base_name, @model_controller_class_path, @model_controller_file_path,
@model_controller_class_nesting, @model_controller_class_nesting_depth =
extract_modules(@model_controller_name)
    @model_controller_class_name_without_nesting,
@model_controller_singular_name, @model_controller_plural_name =
inflect_names(base_name)

    if @model_controller_class_nesting.empty?
      @model_controller_class_name =
@model_controller_class_name_without_nesting
    else
      @model_controller_class_name =
"#{@model_controller_class_nesting}::#{@model_controller_class_name_without_nest
ing}"
    end
    @model_controller_routing_name    = @table_name
    @model_controller_routing_path    = @model_controller_file_path
    @model_controller_controller_name = @model_controller_plural_name

    load_or_initialize_site_keys()

    if options[:dump_generator_attribute_names]
      dump_generator_attribute_names
    end
  end

  def manifest
    recorded_session = record do |m|
      # Check for class naming collisions.
      m.class_collisions controller_class_path,
"#{controller_class_name}Controller", # Sessions Controller

"#{controller_class_name}Helper"
      m.class_collisions model_controller_class_path,
"#{model_controller_class_name}Controller", # Model Controller

"#{model_controller_class_name}Helper"
      m.class_collisions class_path,                  "#{class_name}",
"#{class_name}Mailer", "#{class_name}MailerTest", "#{class_name}Observer"
      m.class_collisions [], 'AuthenticatedSystem', 'AuthenticatedTestHelper'

      # Controller, helper, views, and test directories.
      m.directory File.join('app/models', class_path)
      m.directory File.join('app/controllers', controller_class_path)
      m.directory File.join('app/controllers', model_controller_class_path)
```

```
      m.directory File.join('app/helpers', controller_class_path)
      m.directory File.join('app/views', controller_class_path,
controller_file_name)
      m.directory File.join('app/views', class_path, "#{file_name}_mailer") if
options[:include_activation]

      m.directory File.join('app/controllers', model_controller_class_path)
      m.directory File.join('app/helpers', model_controller_class_path)
      m.directory File.join('app/views', model_controller_class_path,
model_controller_file_name)
      m.directory File.join('config/initializers')

      if @rspec
        m.directory File.join('spec/controllers', controller_class_path)
        m.directory File.join('spec/controllers', model_controller_class_path)
        m.directory File.join('spec/models', class_path)
        m.directory File.join('spec/helpers', model_controller_class_path)
        m.directory File.join('spec/fixtures', class_path)
        m.directory File.join('stories', model_controller_file_path)
        m.directory File.join('stories', 'steps')
      else
        m.directory File.join('test/functional', controller_class_path)
        m.directory File.join('test/functional', model_controller_class_path)
        m.directory File.join('test/unit', class_path)
        m.directory File.join('test/fixtures', class_path)
      end

      m.template 'model.rb',
                  File.join('app/models',
                            class_path,
                            "#{file_name}.rb")

      if options[:include_activation]
        %w( mailer observer ).each do |model_type|
          m.template "#{model_type}.rb", File.join('app/models',
                                                   class_path,
                                                   "#{file_name}_#{model_type}.rb")
        end
      end

      m.template 'controller.rb',
                  File.join('app/controllers',
                            controller_class_path,
                            "#{controller_file_name}_controller.rb")

      m.template 'model_controller.rb',
                  File.join('app/controllers',
                            model_controller_class_path,
                            "#{model_controller_file_name}_controller.rb")

      m.template 'authenticated_system.rb',
                  File.join('lib', 'authenticated_system.rb')

      m.template 'authenticated_test_helper.rb',
                  File.join('lib', 'authenticated_test_helper.rb')
```

```
      m.template 'site_keys.rb', site_keys_file

      if @rspec
        # RSpec Specs
        m.template  'spec/controllers/users_controller_spec.rb',
                    File.join('spec/controllers',
                              model_controller_class_path,

"#{model_controller_file_name}_controller_spec.rb")
        m.template  'spec/controllers/sessions_controller_spec.rb',
                    File.join('spec/controllers',
                              controller_class_path,
                              "#{controller_file_name}_controller_spec.rb")
        m.template  'spec/controllers/access_control_spec.rb',
                    File.join('spec/controllers',
                              controller_class_path,
                              "access_control_spec.rb")
        m.template  'spec/controllers/authenticated_system_spec.rb',
                    File.join('spec/controllers',
                              controller_class_path,
                              "authenticated_system_spec.rb")
        m.template  'spec/helpers/users_helper_spec.rb',
                    File.join('spec/helpers',
                              model_controller_class_path,
                              "#{table_name}_helper_spec.rb")
        m.template  'spec/models/user_spec.rb',
                    File.join('spec/models',
                              class_path,
                              "#{file_name}_spec.rb")
        m.template 'spec/fixtures/users.yml',
                    File.join('spec/fixtures',
                               class_path,
                              "#{table_name}.yml")

        # RSpec Stories
        m.template  'stories/steps/ra_navigation_steps.rb',
         File.join('stories/steps/ra_navigation_steps.rb')
        m.template  'stories/steps/ra_response_steps.rb',
         File.join('stories/steps/ra_response_steps.rb')
        m.template  'stories/steps/ra_resource_steps.rb',
         File.join('stories/steps/ra_resource_steps.rb')
        m.template  'stories/steps/user_steps.rb',
         File.join('stories/steps/', "#{file_name}_steps.rb")
        m.template  'stories/users/accounts.story',
         File.join('stories', model_controller_file_path, 'accounts.story')
        m.template  'stories/users/sessions.story',
         File.join('stories', model_controller_file_path, 'sessions.story')
        m.template  'stories/rest_auth_stories_helper.rb',
         File.join('stories', 'rest_auth_stories_helper.rb')
        m.template  'stories/rest_auth_stories.rb',
         File.join('stories', 'rest_auth_stories.rb')

      else
        m.template 'test/functional_test.rb',
```

```
                    File.join('test/functional',
                              controller_class_path,
                              "#{controller_file_name}_controller_test.rb")
        m.template 'test/model_functional_test.rb',
                    File.join('test/functional',
                              model_controller_class_path,

"#{model_controller_file_name}_controller_test.rb")
        m.template 'test/unit_test.rb',
                    File.join('test/unit',
                              class_path,
                              "#{file_name}_test.rb")
        if options[:include_activation]
          m.template 'test/mailer_test.rb', File.join('test/unit', class_path,
"#{file_name}_mailer_test.rb")
        end
        m.template 'spec/fixtures/users.yml',
                    File.join('test/fixtures',
                              class_path,
                              "#{table_name}.yml")
      end

      m.template 'helper.rb',
                  File.join('app/helpers',
                            controller_class_path,
                            "#{controller_file_name}_helper.rb")

      m.template 'model_helper.rb',
                  File.join('app/helpers',
                            model_controller_class_path,
                            "#{model_controller_file_name}_helper.rb")


      # Controller templates
      m.template 'login.html.erb',  File.join('app/views',
controller_class_path, controller_file_name, "new.html.erb")
      m.template 'signup.html.erb', File.join('app/views',
model_controller_class_path, model_controller_file_name, "new.html.erb")
      m.template '_model_partial.html.erb', File.join('app/views',
model_controller_class_path, model_controller_file_name,
"_#{file_name}_bar.html.erb")

      if options[:include_activation]
        # Mailer templates
        %w( activation signup_notification ).each do |action|
          m.template "#{action}.erb",
                      File.join('app/views', "#{file_name}_mailer",
"#{action}.erb")
        end
      end

      unless options[:skip_migration]
        m.migration_template 'migration.rb', 'db/migrate', :assigns => {
          :migration_name => "Create#{class_name.pluralize.gsub(/::/, '')}"
```

```
        }, :migration_file_name => "create_#{file_path.gsub(/\//,
'_').pluralize}"
      end
      unless options[:skip_routes]
        # Note that this fails for nested classes -- you're on your own with
setting up the routes.
        m.route_resource  controller_singular_name
        m.route_resources model_controller_plural_name
        m.route_name('signup',   '/signup',   {:controller =>
model_controller_plural_name, :action => 'new'})
        m.route_name('register', '/register', {:controller =>
model_controller_plural_name, :action => 'create'})
        m.route_name('login',    '/login',    {:controller =>
controller_controller_name, :action => 'new'})
        m.route_name('logout',   '/logout',   {:controller =>
controller_controller_name, :action => 'destroy'})
      end
    end

    #
    # Post-install notes
    #
    action = File.basename($0) # grok the action from './script/generate' or
whatever
    case action
    when "generate"
      puts "Ready to generate."
      puts ("-" * 70)
      puts "Once finished, don't forget to:"
      puts
      if options[:include_activation]
        puts "- Add an observer to config/environment.rb"
        puts "    config.active_record.observers = :#{file_name}_observer"
      end
      if options[:aasm]
        puts "- Install the acts_as_state_machine gem:"
        puts "    sudo gem sources -a http://gems.github.com (If you haven't
already)"
        puts "    sudo gem install rubyist-aasm"
      elsif options[:stateful]
        puts "- Install the acts_as_state_machine plugin:"
        puts "    svn export
http://elitists.textdriven.com/svn/plugins/acts_as_state_machine/trunk
vendor/plugins/acts_as_state_machine"
      end
      puts "- Add routes to these resources. In config/routes.rb, insert routes
like:"
      puts %(    map.signup '/signup', :controller =>
'#{model_controller_file_name}', :action => 'new')
      puts %(    map.login  '/login',  :controller => '#{controller_file_name}',
:action => 'new')
      puts %(    map.logout '/logout', :controller => '#{controller_file_name}',
:action => 'destroy')
      if options[:include_activation]
```

```
        puts %(    map.activate '/activate/:activation_code', :controller =>
'#{model_controller_file_name}', :action => 'activate', :activation_code => nil)
      end
      if options[:stateful]
        puts  "  and modify the map.resources :#{model_controller_file_name}
line to include these actions:"
        puts  "    map.resources :#{model_controller_file_name}, :member => {
:suspend => :put, :unsuspend => :put, :purge => :delete }"
      end
      puts
      puts ("-" * 70)
      puts
      if $rest_auth_site_key_from_generator.blank?
        puts "You've set a nil site key. This preserves existing users'
passwords,"
        puts "but allows dictionary attacks in the unlikely event your database
is"
        puts "compromised and your site code is not.  See the README for more."
      elsif $rest_auth_keys_are_new
        puts "We've create a new site key in #{site_keys_file}.  If you have
existing"
        puts "user accounts their passwords will no longer work (see README). As
always,"
        puts "keep this file safe but don't post it in public."
      else
        puts "We've reused the existing site key in #{site_keys_file}.  As
always,"
        puts "keep this file safe but don't post it in public."
      end
      puts
      puts ("-" * 70)
    when "destroy"
      puts
      puts ("-" * 70)
      puts
      puts "Thanks for using restful_authentication"
      puts
      puts "Don't forget to comment out the observer line in environment.rb"
      puts "  (This was optional so it may not even be there)"
      puts "  # config.active_record.observers = :#{file_name}_observer"
      puts
      puts ("-" * 70)
      puts
    else
      puts "Didn't understand the action '#{action}' -- you might have missed
the 'after running me' instructions."
    end

    #
    # Do the thing
    #
    recorded_session
  end

  def has_rspec?
```

```
    spec_dir = File.join(RAILS_ROOT, 'spec')
    options[:rspec] ||= (File.exist?(spec_dir) && File.directory?(spec_dir))
unless (options[:rspec] == false)
  end

  #
  # !! These must match the corresponding routines in by_password.rb !!
  #
  def secure_digest(*args)
    Digest::SHA1.hexdigest(args.flatten.join('--'))
  end
  def make_token
    secure_digest(Time.now, (1..10).map{ rand.to_s })
  end
  def password_digest(password, salt)
    digest = $rest_auth_site_key_from_generator
    $rest_auth_digest_stretches_from_generator.times do
      digest = secure_digest(digest, salt, password,
$rest_auth_site_key_from_generator)
    end
    digest
  end

  #
  # Try to be idempotent:
  # pull in the existing site key if any,
  # seed it with reasonable defaults otherwise
  #
  def load_or_initialize_site_keys
    case
    when defined? REST_AUTH_SITE_KEY
      if (options[:old_passwords]) && ((! REST_AUTH_SITE_KEY.blank?) ||
(REST_AUTH_DIGEST_STRETCHES != 1))
        raise "You have a site key, but --old-passwords will overwrite it.  If
this is really what you want, move the file #{site_keys_file} and re-run."
      end
      $rest_auth_site_key_from_generator         = REST_AUTH_SITE_KEY
      $rest_auth_digest_stretches_from_generator = REST_AUTH_DIGEST_STRETCHES
    when options[:old_passwords]
      $rest_auth_site_key_from_generator         = nil
      $rest_auth_digest_stretches_from_generator = 1
      $rest_auth_keys_are_new                    = true
    else
      $rest_auth_site_key_from_generator         = make_token
      $rest_auth_digest_stretches_from_generator = 10
      $rest_auth_keys_are_new                    = true
    end
  end
  def site_keys_file
    File.join("config", "initializers", "site_keys.rb")
  end

protected
  # Override with your own usage banner.
  def banner
```

```
    "Usage: #{$0} authenticated ModelName [ControllerName]"
  end

  def add_options!(opt)
    opt.separator ''
    opt.separator 'Options:'
    opt.on("--skip-migration",
      "Don't generate a migration file for this model")           { |v|
options[:skip_migration] = v }
    opt.on("--include-activation",
      "Generate signup 'activation code' confirmation via email") { |v|
options[:include_activation] = true }
    opt.on("--stateful",
      "Use acts_as_state_machine.  Assumes --include-activation") { |v|
options[:include_activation] = options[:stateful] = true }
    opt.on("--aasm",
      "Use (gem) aasm.  Assumes --include-activation")            { |v|
options[:include_activation] = options[:stateful] = options[:aasm] = true }
    opt.on("--rspec",
      "Force rspec mode (checks for RAILS_ROOT/spec by default)") { |v|
options[:rspec] = true }
    opt.on("--no-rspec",
      "Force test (not RSpec mode)")                              { |v|
options[:rspec] = false }
    opt.on("--skip-routes",
      "Don't generate a resource line in config/routes.rb")       { |v|
options[:skip_routes] = v }
    opt.on("--old-passwords",
      "Use the older password encryption scheme (see README)")    { |v|
options[:old_passwords] = v }
    opt.on("--dump-generator-attrs",
      "(generator debug helper)")                                 { |v|
options[:dump_generator_attribute_names] = v }
  end

  def dump_generator_attribute_names
    generator_attribute_names = [
      :table_name,
      :file_name,
      :class_name,
      :controller_name,
      :controller_class_path,
      :controller_file_path,
      :controller_class_nesting,
      :controller_class_nesting_depth,
      :controller_class_name,
      :controller_singular_name,
      :controller_plural_name,
      :controller_routing_name,                 # new_session_path
      :controller_routing_path,                 # /session/new
      :controller_controller_name,              # sessions
      :controller_file_name,
      :controller_table_name, :controller_plural_name,
      :model_controller_name,
      :model_controller_class_path,
```

```
      :model_controller_file_path,
      :model_controller_class_nesting,
      :model_controller_class_nesting_depth,
      :model_controller_class_name,
      :model_controller_singular_name,
      :model_controller_plural_name,
      :model_controller_routing_name,           # new_user_path
      :model_controller_routing_path,           # /users/new
      :model_controller_controller_name,        # users
      :model_controller_file_name,  :model_controller_singular_name,
      :model_controller_table_name, :model_controller_plural_name,
    ]
    generator_attribute_names.each do |attr|
      puts "%-40s %s" % ["#{attr}:", self.send(attr)]  #
instance_variable_get("@#{attr.to_s}"
    end

  end
end

# ./script/generate authenticated FoonParent::Foon SporkParent::Spork -p --force
--rspec --dump-generator-attrs
# table_name:                           foon_parent_foons
# file_name:                            foon
# class_name:                           FoonParent::Foon
# controller_name:                      SporkParent::Sporks
# controller_class_path:                spork_parent
# controller_file_path:                 spork_parent/sporks
# controller_class_nesting:             SporkParent
# controller_class_nesting_depth:       1
# controller_class_name:                SporkParent::Sporks
# controller_singular_name:             spork
# controller_plural_name:               sporks
# controller_routing_name:              spork
# controller_routing_path:              spork_parent/spork
# controller_controller_name:           sporks
# controller_file_name:                 sporks
# controller_table_name:                sporks
# controller_plural_name:               sporks
# model_controller_name:                FoonParent::Foons
# model_controller_class_path:          foon_parent
# model_controller_file_path:           foon_parent/foons
# model_controller_class_nesting:       FoonParent
# model_controller_class_nesting_depth: 1
# model_controller_class_name:          FoonParent::Foons
# model_controller_singular_name:       foons
# model_controller_plural_name:         foons
# model_controller_routing_name:        foon_parent_foons
# model_controller_routing_path:        foon_parent/foons
# model_controller_controller_name:     foons
# model_controller_file_name:           foons
# model_controller_singular_name:       foons
# model_controller_table_name:          foons
# model_controller_plural_name:         foons
Rails::Generator::Commands::Create.class_eval do
```

```
  def route_resource(*resources)
    resource_list = resources.map { |r| r.to_sym.inspect }.join(', ')
    sentinel = 'ActionController::Routing::Routes.draw do |map|'

    logger.route "map.resource #{resource_list}"
    unless options[:pretend]
      gsub_file 'config/routes.rb', /(#{Regexp.escape(sentinel)})/mi do |match|
        "#{match}\n  map.resource #{resource_list}\n"
      end
    end
  end

  def route_name(name, path, route_options = {})
    sentinel = 'ActionController::Routing::Routes.draw do |map|'

    logger.route "map.#{name} '#{path}', :controller =>
'#{route_options[:controller]}', :action => '#{route_options[:action]}'"
    unless options[:pretend]
      gsub_file 'config/routes.rb', /(#{Regexp.escape(sentinel)})/mi do |match|
        "#{match}\n  map.#{name} '#{path}', :controller =>
'#{route_options[:controller]}', :action => '#{route_options[:action]}'"
      end
    end
  end
end

Rails::Generator::Commands::Destroy.class_eval do
  def route_resource(*resources)
    resource_list = resources.map { |r| r.to_sym.inspect }.join(', ')
    look_for = "\n  map.resource #{resource_list}\n"
    logger.route "map.resource #{resource_list}"
    unless options[:pretend]
      gsub_file 'config/routes.rb', /(#{look_for})/mi, ''
    end
  end

  def route_name(name, path, route_options = {})
    look_for =   "\n  map.#{name} '#{path}', :controller =>
'#{route_options[:controller]}', :action => '#{route_options[:action]}'"
    logger.route "map.#{name} '#{path}',     :controller =>
'#{route_options[:controller]}', :action => '#{route_options[:action]}'"
    unless options[:pretend]
      gsub_file    'config/routes.rb', /(#{look_for})/mi, ''
    end
  end
end

Rails::Generator::Commands::List.class_eval do
  def route_resource(*resources)
    resource_list = resources.map { |r| r.to_sym.inspect }.join(', ')
    logger.route "map.resource #{resource_list}"
  end

  def route_name(name, path, options = {})
```

```
    logger.route "map.#{name} '#{path}', :controller =>
'{options[:controller]}', :action => '#{options[:action]}'"
  end
end
module AuthenticatedSystem
  protected
    # Returns true or false if the <%= file_name %> is logged in.
    # Preloads @current_<%= file_name %> with the <%= file_name %> model if
they're logged in.
    def logged_in?
      !!current_<%= file_name %>
    end

    # Accesses the current <%= file_name %> from the session.
    # Future calls avoid the database because nil is not equal to false.
    def current_<%= file_name %>
      @current_<%= file_name %> ||= (login_from_session || login_from_basic_auth
|| login_from_cookie) unless @current_<%= file_name %> == false
    end

    # Store the given <%= file_name %> id in the session.
    def current_<%= file_name %>=(new_<%= file_name %>)
      session[:<%= file_name %>_id] = new_<%= file_name %> ? new_<%= file_name
%>.id : nil
      @current_<%= file_name %> = new_<%= file_name %> || false
    end

    # Check if the <%= file_name %> is authorized
    #
    # Override this method in your controllers if you want to restrict access
    # to only a few actions or if you want to check if the <%= file_name %>
    # has the correct rights.
    #
    # Example:
    #
    #  # only allow nonbobs
    #  def authorized?
    #    current_<%= file_name %>.login != "bob"
    #  end
    #
    def authorized?(action = action_name, resource = nil)
      logged_in?
    end

    # Filter method to enforce a login requirement.
    #
    # To require logins for all actions, use this in your controllers:
    #
    #   before_filter :login_required
    #
    # To require logins for specific actions, use this in your controllers:
    #
    #   before_filter :login_required, :only => [ :edit, :update ]
    #
    # To skip this in a subclassed controller:
```

```
    #
    #   skip_before_filter :login_required
    #
    def login_required
      authorized? || access_denied
    end

    # Redirect as appropriate when an access request fails.
    #
    # The default action is to redirect to the login screen.
    #
    # Override this method in your controllers if you want to have special
    # behavior in case the <%= file_name %> is not authorized
    # to access the requested action.  For example, a popup window might
    # simply close itself.
    def access_denied
      respond_to do |format|
        format.html do
          store_location
          redirect_to new_<%= controller_routing_name %>_path
        end
        # format.any doesn't work in rails version <
http://dev.rubyonrails.org/changeset/8987
        # Add any other API formats here.  (Some browsers, notably IE6, send
Accept: */* and trigger
        # the 'format.any' block incorrectly. See http://bit.ly/ie6_borken or
http://bit.ly/ie6_borken2
        # for a workaround.)
        format.any(:json, :xml) do
          request_http_basic_authentication 'Web Password'
        end
      end
    end

    # Store the URI of the current request in the session.
    #
    # We can return to this location by calling #redirect_back_or_default.
    def store_location
      session[:return_to] = request.request_uri
    end

    # Redirect to the URI stored by the most recent store_location call or
    # to the passed default.  Set an appropriately modified
    #   after_filter :store_location, :only => [:index, :new, :show, :edit]
    # for any controller you want to be bounce-backable.
    def redirect_back_or_default(default)
      redirect_to(session[:return_to] || default)
      session[:return_to] = nil
    end

    # Inclusion hook to make #current_<%= file_name %> and #logged_in?
    # available as ActionView helper methods.
    def self.included(base)
      base.send :helper_method, :current_<%= file_name %>, :logged_in?,
:authorized? if base.respond_to? :helper_method
```

```
    end

    #
    # Login
    #

    # Called from #current_<%= file_name %>.  First attempt to login by the <%=
file_name %> id stored in the session.
    def login_from_session
      self.current_<%= file_name %> = <%= class_name %>.find_by_id(session[:<%=
file_name %>_id]) if session[:<%= file_name %>_id]
    end

    # Called from #current_<%= file_name %>.  Now, attempt to login by basic
authentication information.
    def login_from_basic_auth
      authenticate_with_http_basic do |login, password|
        self.current_<%= file_name %> = <%= class_name %>.authenticate(login,
password)
      end
    end

    #
    # Logout
    #

    # Called from #current_<%= file_name %>.  Finaly, attempt to login by an
expiring token in the cookie.
    # for the paranoid: we _should_ be storing <%= file_name %>_token =
hash(cookie_token, request IP)
    def login_from_cookie
      <%= file_name %> = cookies[:auth_token] && <%= class_name
%>.find_by_remember_token(cookies[:auth_token])
      if <%= file_name %> && <%= file_name %>.remember_token?
        self.current_<%= file_name %> = <%= file_name %>
        handle_remember_cookie! false # freshen cookie token (keeping date)
        self.current_<%= file_name %>
      end
    end

    # This is ususally what you want; resetting the session willy-nilly wreaks
    # havoc with forgery protection, and is only strictly necessary on login.
    # However, **all session state variables should be unset here**.
    def logout_keeping_session!
      # Kill server-side auth cookie
      @current_<%= file_name %>.forget_me if @current_<%= file_name %>.is_a? <%=
class_name %>
      @current_<%= file_name %> = false     # not logged in, and don't do it for
me
      kill_remember_cookie!     # Kill client-side auth cookie
      session[:<%= file_name %>_id] = nil   # keeps the session but kill our
variable
      # explicitly kill any other session variables you set
    end
```

```
    # The session should only be reset at the tail end of a form POST --
    # otherwise the request forgery protection fails. It's only really necessary
    # when you cross quarantine (logged-out to logged-in).
    def logout_killing_session!
      logout_keeping_session!
      reset_session
    end

    #
    # Remember_me Tokens
    #
    # Cookies shouldn't be allowed to persist past their freshness date,
    # and they should be changed at each login

    # Cookies shouldn't be allowed to persist past their freshness date,
    # and they should be changed at each login

    def valid_remember_cookie?
      return nil unless @current_<%= file_name %>
      (@current_<%= file_name %>.remember_token?) &&
        (cookies[:auth_token] == @current_<%= file_name %>.remember_token)
    end

    # Refresh the cookie auth token if it exists, create it otherwise
    def handle_remember_cookie!(new_cookie_flag)
      return unless @current_<%= file_name %>
      case
      when valid_remember_cookie? then @current_<%= file_name %>.refresh_token #
keeping same expiry date
      when new_cookie_flag        then @current_<%= file_name %>.remember_me
      else                             @current_<%= file_name %>.forget_me
      end
      send_remember_cookie!
    end

    def kill_remember_cookie!
      cookies.delete :auth_token
    end

    def send_remember_cookie!
      cookies[:auth_token] = {
        :value   => @current_<%= file_name %>.remember_token,
        :expires => @current_<%= file_name %>.remember_token_expires_at }
    end

end
module AuthenticatedTestHelper
  # Sets the current <%= file_name %> in the session from the <%= file_name %>
fixtures.
  def login_as(<%= file_name %>)
    @request.session[:<%= file_name %>_id] = <%= file_name %> ? <%= table_name
%>(<%= file_name %>).id : nil
  end

  def authorize_as(<%= file_name %>)
```

```
    @request.env["HTTP_AUTHORIZATION"] = <%= file_name %> ?
ActionController::HttpAuthentication::Basic.encode_credentials(<%= table_name
%>(<%= file_name %>).login, 'monkey') : nil
  end

<% if options[:rspec] -%>
  # rspec
  def mock_<%= file_name %>
    <%= file_name %> = mock_model(<%= class_name %>, :id => 1,
      :login  => 'user_name',
      :name   => 'U. Surname',
      :to_xml => "<%= class_name %>-in-XML", :to_json => "<%= class_name %>-in-
JSON",
      :errors => [])
    <%= file_name %>
  end
<% end -%>
end
# This controller handles the login/logout function of the site.
class <%= controller_class_name %>Controller < ApplicationController
  # Be sure to include AuthenticationSystem in Application Controller instead
  include AuthenticatedSystem

  # render new.rhtml
  def new
  end

  def create
    logout_keeping_session!
    <%= file_name %> = <%= class_name %>.authenticate(params[:login],
params[:password])
    if <%= file_name %>
      # Protects against session fixation attacks, causes request forgery
      # protection if user resubmits an earlier form using back
      # button. Uncomment if you understand the tradeoffs.
      # reset_session
      self.current_<%= file_name %> = <%= file_name %>
      new_cookie_flag = (params[:remember_me] == "1")
      handle_remember_cookie! new_cookie_flag
      redirect_back_or_default('/')
      flash[:notice] = "Logged in successfully"
    else
      note_failed_signin
      @login       = params[:login]
      @remember_me = params[:remember_me]
      render :action => 'new'
    end
  end

  def destroy
    logout_killing_session!
    flash[:notice] = "You have been logged out."
    redirect_back_or_default('/')
  end
```

```
protected
  # Track failed login attempts
  def note_failed_signin
    flash[:error] = "Couldn't log you in as '#{params[:login]}'"
    logger.warn "Failed login for '#{params[:login]}' from #{request.remote_ip}
at #{Time.now.utc}"
  end
end
module <%= controller_class_name %>Helper
endclass <%= class_name %>Mailer < ActionMailer::Base
  def signup_notification(<%= file_name %>)
    setup_email(<%= file_name %>)
    @subject    += 'Please activate your new account'
  <% if options[:include_activation] %>
    @body[:url]  = "http://YOURSITE/activate/#{<%= file_name
%>.activation_code}"
  <% else %>
    @body[:url]  = "http://YOURSITE/login/" <% end %>
  end

  def activation(<%= file_name %>)
    setup_email(<%= file_name %>)
    @subject    += 'Your account has been activated!'
    @body[:url]  = "http://YOURSITE/"
  end

  protected
    def setup_email(<%= file_name %>)
      @recipients  = "#{<%= file_name %>.email}"
      @from        = "ADMINEMAIL"
      @subject     = "[YOURSITE] "
      @sent_on     = Time.now
      @body[:<%= file_name %>] = <%= file_name %>
    end
end
class <%= migration_name %> < ActiveRecord::Migration
  def self.up
    create_table "<%= table_name %>", :force => true do |t|
      t.column :login,                     :string, :limit => 40
      t.column :name,                      :string, :limit => 100, :default =>
'', :null => true
      t.column :email,                     :string, :limit => 100
      t.column :crypted_password,          :string, :limit => 40
      t.column :salt,                      :string, :limit => 40
      t.column :created_at,                :datetime
      t.column :updated_at,                :datetime
      t.column :remember_token,            :string, :limit => 40
      t.column :remember_token_expires_at, :datetime
<% if options[:include_activation] -%>
      t.column :activation_code,           :string, :limit => 40
      t.column :activated_at,              :datetime<% end %>
<% if options[:stateful] -%>
      t.column :state,                     :string, :null => :no, :default =>
'passive'
      t.column :deleted_at,                :datetime<% end %>
```

```
    end
    add_index :<%= table_name %>, :login, :unique => true
  end

  def self.down
    drop_table "<%= table_name %>"
  end
end
require 'digest/sha1'

class <%= class_name %> < ActiveRecord::Base
  include Authentication
  include Authentication::ByPassword
  include Authentication::ByCookieToken
<% if options[:aasm] -%>
  include Authorization::AasmRoles
<% elsif options[:stateful] -%>
  include Authorization::StatefulRoles<% end %>
  validates_presence_of     :login
  validates_length_of       :login,    :within => 3..40
  validates_uniqueness_of   :login
  validates_format_of       :login,    :with => Authentication.login_regex,
:message => Authentication.bad_login_message

  validates_format_of       :name,     :with => Authentication.name_regex,
:message => Authentication.bad_name_message, :allow_nil => true
  validates_length_of       :name,     :maximum => 100

  validates_presence_of     :email
  validates_length_of       :email,    :within => 6..100 #r@a.wk
  validates_uniqueness_of   :email
  validates_format_of       :email,    :with => Authentication.email_regex,
:message => Authentication.bad_email_message

  <% if options[:include_activation] && !options[:stateful] %>before_create
:make_activation_code <% end %>

  # HACK HACK HACK -- how to do attr_accessible from here?
  # prevents a user from submitting a crafted form that bypasses activation
  # anything else you want your user to change should be added here.
  attr_accessible :login, :email, :name, :password, :password_confirmation

<% if options[:include_activation] && !options[:stateful] %>
  # Activates the user in the database.
  def activate!
    @activated = true
    self.activated_at = Time.now.utc
    self.activation_code = nil
    save(false)
  end

  # Returns true if the user has just been activated.
  def recently_activated?
    @activated
  end
```

```
  def active?
    # the existence of an activation code means they have not activated yet
    activation_code.nil?
  end<% end %>

  # Authenticates a user by their login name and unencrypted password.  Returns
the user or nil.
  #
  # uff.  this is really an authorization, not authentication routine.
  # We really need a Dispatch Chain here or something.
  # This will also let us return a human error message.
  #
  def self.authenticate(login, password)
    return nil if login.blank? || password.blank?
    u = <% if    options[:stateful]           %>find_in_state :first, :active,
:conditions => {:login => login}<%
          elsif options[:include_activation] %>find :first, :conditions =>
['login = ? and activated_at IS NOT NULL', login]<%
          else %>find_by_login(login)<% end %> # need to get the salt
    u && u.authenticated?(password) ? u : nil
  end

  def login=(value)
    write_attribute :login, (value ? value.downcase : nil)
  end

  def email=(value)
    write_attribute :email, (value ? value.downcase : nil)
  end

  protected

<% if options[:include_activation] -%>
    def make_activation_code
  <% if options[:stateful] -%>
      self.deleted_at = nil
  <% end -%>
      self.activation_code = self.class.make_token
    end
<% end %>

end
class <%= model_controller_class_name %>Controller < ApplicationController
  # Be sure to include AuthenticationSystem in Application Controller instead
  include AuthenticatedSystem
  <% if options[:stateful] %>
  # Protect these actions behind an admin login
  # before_filter :admin_required, :only => [:suspend, :unsuspend, :destroy,
:purge]
  before_filter :find_<%= file_name %>, :only => [:suspend, :unsuspend,
:destroy, :purge]
  <% end %>

  # render new.rhtml
```

```
  def new
    @<%= file_name %> = <%= class_name %>.new
  end

  def create
    logout_keeping_session!
    @<%= file_name %> = <%= class_name %>.new(params[:<%= file_name %>])
<% if options[:stateful] -%>
    @<%= file_name %>.register! if @<%= file_name %> && @<%= file_name %>.valid?
    success = @<%= file_name %> && @<%= file_name %>.valid?
<% else -%>
    success = @<%= file_name %> && @<%= file_name %>.save
<% end -%>
    if success && @<%= file_name %>.errors.empty?
      <% if !options[:include_activation] -%>
      # Protects against session fixation attacks, causes request forgery
      # protection if visitor resubmits an earlier form using back
      # button. Uncomment if you understand the tradeoffs.
      # reset session
      self.current_<%= file_name %> = @<%= file_name %> # !! now logged in
      <% end -%>redirect_back_or_default('/')
      flash[:notice] = "Thanks for signing up!  We're sending you an email with
your activation code."
    else
      flash[:error]  = "We couldn't set up that account, sorry.  Please try
again, or contact an admin (link is above)."
      render :action => 'new'
    end
  end
<% if options[:include_activation] %>
  def activate
    logout_keeping_session!
    <%= file_name %> = <%= class_name
%>.find_by_activation_code(params[:activation_code]) unless
params[:activation_code].blank?
    case
    when (!params[:activation_code].blank?) && <%= file_name %> && !<%=
file_name %>.active?
      <%= file_name %>.activate!
      flash[:notice] = "Signup complete! Please sign in to continue."
      redirect_to '/login'
    when params[:activation_code].blank?
      flash[:error] = "The activation code was missing.  Please follow the URL
from your email."
      redirect_back_or_default('/')
    else
      flash[:error]  = "We couldn't find a <%= file_name %> with that activation
code -- check your email? Or maybe you've already activated -- try signing in."
      redirect_back_or_default('/')
    end
  end
<% end %><% if options[:stateful] %>
  def suspend
    @<%= file_name %>.suspend!
    redirect_to <%= model_controller_routing_name %>_path
```

```
  end

  def unsuspend
    @<%= file_name %>.unsuspend!
    redirect_to <%= model_controller_routing_name %>_path
  end

  def destroy
    @<%= file_name %>.delete!
    redirect_to <%= model_controller_routing_name %>_path
  end

  def purge
    @<%= file_name %>.destroy
    redirect_to <%= model_controller_routing_name %>_path
  end

  # There's no page here to update or destroy a <%= file_name %>.  If you add
those, be
  # smart -- make sure you check that the visitor is authorized to do so, that
they
  # supply their old password along with a new one to update it, etc.

protected
  def find_<%= file_name %>
    @<%= file_name %> = <%= class_name %>.find(params[:id])
  end
<% end -%>
end
module <%= model_controller_class_name %>Helper

  #
  # Use this to wrap view elements that the user can't access.
  # !! Note: this is an *interface*, not *security* feature !!
  # You need to do all access control at the controller level.
  #
  # Example:
  # <%%= if_authorized?(:index,   User)  do link_to('List all users',
users_path) end %> |
  # <%%= if_authorized?(:edit,    @user) do link_to('Edit this user',
edit_user_path) end %> |
  # <%%= if_authorized?(:destroy, @user) do link_to 'Destroy', @user, :confirm
=> 'Are you sure?', :method => :delete end %>
  #
  #
  def if_authorized?(action, resource, &block)
    if authorized?(action, resource)
      yield action, resource
    end
  end

  #
  # Link to user's page ('<%= table_name %>/1')
  #
  # By default, their login is used as link text and link title (tooltip)
```

```
  #
  # Takes options
  # * :content_text => 'Content text in place of <%= file_name %>.login',
escaped with
  #   the standard h() function.
  # * :content_method => :<%= file_name
%>_instance_method_to_call_for_content_text
  # * :title_method => :<%= file_name
%>_instance_method_to_call_for_title_attribute
  # * as well as link_to()'s standard options
  #
  # Examples:
  #   link_to_<%= file_name %> @<%= file_name %>
  #   # => <a href="/<%= table_name %>/3" title="barmy">barmy</a>
  #
  #   # if you've added a .name attribute:
  #  content_tag :span, :class => :vcard do
  #    (link_to_<%= file_name %> <%= file_name %>, :class => 'fn n',
:title_method => :login, :content_method => :name) +
  #          ': ' + (content_tag :span, <%= file_name %>.email, :class =>
'email')
  #   end
  #   # => <span class="vcard"><a href="/<%= table_name %>/3" title="barmy"
class="fn n">Cyril Fotheringay-Phipps</a>: <span
class="email">barmy@blandings.com</span></span>
  #
  #   link_to_<%= file_name %> @<%= file_name %>, :content_text => 'Your user
page'
  #   # => <a href="/<%= table_name %>/3" title="barmy" class="nickname">Your
user page</a>
  #
  def link_to_<%= file_name %>(<%= file_name %>, options={})
    raise "Invalid <%= file_name %>" unless <%= file_name %>
    options.reverse_merge! :content_method => :login, :title_method => :login,
:class => :nickname
    content_text      = options.delete(:content_text)
    content_text    ||= <%= file_name %>.send(options.delete(:content_method))
    options[:title] ||= <%= file_name %>.send(options.delete(:title_method))
    link_to h(content_text), <%= model_controller_routing_name.singularize
%>_path(<%= file_name %>), options
  end

  #
  # Link to login page using remote ip address as link content
  #
  # The :title (and thus, tooltip) is set to the IP address
  #
  # Examples:
  #   link_to_login_with_IP
  #   # => <a href="/login" title="169.69.69.69">169.69.69.69</a>
  #
  #   link_to_login_with_IP :content_text => 'not signed in'
  #   # => <a href="/login" title="169.69.69.69">not signed in</a>
  #
  def link_to_login_with_IP content_text=nil, options={}
```

```
    ip_addr           = request.remote_ip
    content_text    ||= ip_addr
    options.reverse_merge! :title => ip_addr
    if tag = options.delete(:tag)
      content_tag tag, h(content_text), options
    else
      link_to h(content_text), login_path, options
    end
  end

  #
  # Link to the current user's page (using link_to_<%= file_name %>) or to the
login page
  # (using link_to_login_with_IP).
  #
  def link_to_current_<%= file_name %>(options={})
    if current_<%= file_name %>
      link_to_<%= file_name %> current_<%= file_name %>, options
    else
      content_text = options.delete(:content_text) || 'not signed in'
      # kill ignored options from link_to_<%= file_name %>
      [:content_method, :title_method].each{|opt| options.delete(opt)}
      link_to_login_with_IP content_text, options
    end
  end

end
require File.dirname(__FILE__) + '/../spec_helper'
include ApplicationHelper
include <%= model_controller_class_name %>Helper

describe "<%= model_controller_class_name %>Helper.link_to_<%= file_name %>" do
  before do
    @<%= file_name %> = <%= class_name %>.new({
        :name  => '<%= class_name %> Name',
        :login => '<%= file_name %>_name',
      })
    @<%= file_name %>.id = 1 # set non-attr_accessible specifically
  end

  it "should give an error on a nil <%= file_name %>" do
    lambda { link_to_<%= file_name %>(nil) }.should raise_error('Invalid <%=
file_name %>')
  end

  it "should link to the given <%= file_name %>" do
    link_to_<%= file_name %>(@<%= file_name %>).should have_tag("a[href='/<%=
table_name %>/1']")
  end

  it "should use given link text if :content_text is specified" do
    link_to_<%= file_name %>(@<%= file_name %>, :content_text => 'Hello
there!').should have_tag("a", 'Hello there!')
  end
```

```
  it "should use the login as link text with no :content_method specified" do
    link_to_<%= file_name %>(@<%= file_name %>).should have_tag("a", '<%=
file_name %>_name')
  end

  it "should use the name as link text with :content_method => :name" do
    link_to_<%= file_name %>(@<%= file_name %>, :content_method => :name).should
have_tag("a", '<%= class_name %> Name')
  end

  it "should use the login as title with no :title_method specified" do
    link_to_<%= file_name %>(@<%= file_name %>).should have_tag("a[title='<%=
file_name %>_name']")
  end

  it "should use the name as link title with :content_method => :name" do
    link_to_<%= file_name %>(@<%= file_name %>, :title_method => :name).should
have_tag("a[title='<%= class_name %> Name']")
  end

  it "should have nickname as a class by default" do
    link_to_<%= file_name %>(@<%= file_name %>).should have_tag("a.nickname")
  end

  it "should take other classes and no longer have the nickname class" do
    result = link_to_<%= file_name %>(@<%= file_name %>, :class => 'foo bar')
    result.should have_tag("a.foo")
    result.should have_tag("a.bar")
  end
end

describe "<%= model_controller_class_name %>Helper.link_to_signin_with_IP" do
  before do
  end

  it "should link to the signin_path" do
    link_to_signin_with_IP().should have_tag("a[href='/signin']")
  end

  it "should use given link text if :content_text is specified" do
    link_to_signin_with_IP(:content_text => 'Hello there!').should have_tag("a",
'Hello there!')
  end

  it "should use the login as link text with no :content_method specified" do
    link_to_signin_with_IP().should have_tag("a", '0.0.0.0')
  end

  it "should use the ip address as title" do
    link_to_signin_with_IP().should have_tag("a[title='0.0.0.0']")
  end

  it "should by default be like school in summer and have no class" do
    link_to_signin_with_IP().should_not have_tag("a.nickname")
  end
```

```
  it "should have some class if you tell it to" do
    result = link_to_signin_with_IP(:class => 'foo bar')
    result.should have_tag("a.foo")
    result.should have_tag("a.bar")
  end
end

describe "<%= model_controller_class_name %>Helper.link_to_current_<%= file_name
%>, When logged in" do
  fixtures :<%= table_name %>
  include AuthenticatedTestHelper
  before do
    login_as(:quentin)
  end

  it "should link to the given <%= file_name %>" do
    link_to_current_<%= file_name %>().should have_tag("a[href='/<%= table_name
%>/1']")
  end

  it "should use given link text if :content_text is specified" do
    link_to_current_user(:content_text => 'Hello there!').should have_tag("a",
'Hello there!')
  end

  it "should use the login as link text with no :content_method specified" do
    link_to_current_user().should have_tag("a", 'quentin')
  end

  it "should use the name as link text with :content_method => :name" do
    link_to_current_user(:content_method => :name).should have_tag("a",
'Quentin')
  end

  it "should use the login as title with no :title_method specified" do
    link_to_current_user().should have_tag("a[title='quentin']")
  end

  it "should use the name as link title with :content_method => :name" do
    link_to_current_user(:title_method => :name).should
have_tag("a[title='Quentin']")
  end

  it "should have nickname as a class" do
    link_to_current_user().should have_tag("a.nickname")
  end

  it "should take other classes and no longer have the nickname class" do
    result = link_to_current_user(:class => 'foo bar')
    result.should have_tag("a.foo")
    result.should have_tag("a.bar")
  end
end
```

```
describe "<%= model_controller_class_name %>Helper.link_to_current_user, When
logged out" do
  include AuthenticatedTestHelper
  before do
  end

  it "should link to the signin_path" do
    link_to_current_user().should have_tag("a[href='/signin']")
  end

  it "should use given link text if :content_text is specified" do
    link_to_current_user(:content_text => 'Hello there!').should have_tag("a",
'Hello there!')
  end

  it "should use the IP address as link text with no :content_method specified"
do
    link_to_current_user().should have_tag("a", '0.0.0.0')
  end

  it "should use the ip address as title" do
    link_to_current_user().should have_tag("a[title='0.0.0.0']")
  end

  it "should by default be like school in summer and have no class" do
    link_to_current_user().should_not have_tag("a.nickname")
  end

  it "should have some class if you tell it to" do
    result = link_to_current_user(:class => 'foo bar')
    result.should have_tag("a.foo")
    result.should have_tag("a.bar")
  end
end
class <%= class_name %>Observer < ActiveRecord::Observer
  def after_create(<%= file_name %>)
    <%= class_name %>Mailer.deliver_signup_notification(<%= file_name %>)
  end

  def after_save(<%= file_name %>)
  <% if options[:include_activation] %>
    <%= class_name %>Mailer.deliver_activation(<%= file_name %>) if <%=
file_name %>.recently_activated?
  <% end %>
  end
end

# A Site key gives additional protection against a dictionary attack if your
# DB is ever compromised.  With no site key, we store
#   DB_password = hash(user_password, DB_user_salt)
# If your database were to be compromised you'd be vulnerable to a dictionary
# attack on all your stupid users' passwords.  With a site key, we store
#   DB_password = hash(user_password, DB_user_salt, Code_site_key)
```

```
# That means an attacker needs access to both your site's code *and* its
# database to mount an "offline dictionary
attack.":http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/web-
authentication.html
#
# It's probably of minor importance, but recommended by best practices: 'defense
# in depth'.  Needless to say, if you upload this to github or the youtubes or
# otherwise place it in public view you'll kinda defeat the point.  Your users'
# passwords are still secure, and the world won't end, but defense_in_depth -=
1.
#
# Please note: if you change this, all the passwords will be invalidated, so DO
# keep it someplace secure.  Use the random value given or type in the lyrics to
# your favorite Jay-Z song or something; any moderately long, unpredictable
text.
REST_AUTH_SITE_KEY         = '<%= $rest_auth_site_key_from_generator %>'

# Repeated applications of the hash make brute force (even with a compromised
# database and site key) harder, and scale with Moore's law.
#
#   bq. "To squeeze the most security out of a limited-entropy password or
#   passphrase, we can use two techniques [salting and stretching]... that are
#   so simple and obvious that they should be used in every password system.
#   There is really no excuse not to use them." http://tinyurl.com/37lb73
#   Practical Security (Ferguson & Scheier) p350
#
# A modest 10 foldings (the default here) adds 3ms.  This makes brute forcing 10
# times harder, while reducing an app that otherwise serves 100 reqs/s to 78
signin
# reqs/s, an app that does 10reqs/s to 9.7 reqs/s
#
# More:
# * http://www.owasp.org/index.php/Hashing_Java
# * "An Illustrated Guide to Cryptographic
Hashes":http://www.unixwiz.net/techtips/iguide-crypto-hashes.html

REST_AUTH_DIGEST_STRETCHES = <%= $rest_auth_digest_stretches_from_generator %>
require File.dirname(__FILE__) + '<%= ('/..'*controller_class_nesting_depth) +
'/../spec_helper' %>'
  # Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper

#
# A test controller with and without access controls
#
class AccessControlTestController < ApplicationController
  before_filter :login_required, :only => :login_is_required
  def login_is_required
    respond_to do |format|
      @foo = { 'success' => params[:format]||'no fmt given'}
      format.html do render :text => "success"             end
      format.xml  do render :xml  => @foo, :status => :ok  end
      format.json do render :json => @foo, :status => :ok  end
    end
```

```
  end
  def login_not_required
    respond_to do |format|
      @foo = { 'success' => params[:format]||'no fmt given'}
      format.html do render :text => "success"             end
      format.xml  do render :xml  => @foo, :status => :ok  end
      format.json do render :json => @foo, :status => :ok  end
    end
  end
end

#
# Access Control
#

ACCESS_CONTROL_FORMATS = [
  ['',     "success"],
  ['xml',  "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<hash>\n
<success>xml</success>\n</hash>\n"],
  ['json', "{\"success\": \"json\"}"],]
ACCESS_CONTROL_AM_I_LOGGED_IN = [
  [:i_am_logged_in,     :quentin],
  [:i_am_not_logged_in, nil],]
ACCESS_CONTROL_IS_LOGIN_REQD = [
  :login_not_required,
  :login_is_required,]

describe AccessControlTestController do
  fixtures        :<%= table_name %>
  before do
    # is there a better way to do this?
    ActionController::Routing::Routes.add_route '/login_is_required',
:controller => 'access_control_test',   :action => 'login_is_required'
    ActionController::Routing::Routes.add_route '/login_not_required',
:controller => 'access_control_test',   :action => 'login_not_required'
  end

  ACCESS_CONTROL_FORMATS.each do |format, success_text|
    ACCESS_CONTROL_AM_I_LOGGED_IN.each do |logged_in_status, <%= file_name
%>_login|
      ACCESS_CONTROL_IS_LOGIN_REQD.each do |login_reqd_status|
        describe "requesting #{format.blank? ? 'html' : format};
#{logged_in_status.to_s.humanize} and #{login_reqd_status.to_s.humanize}" do
          before do
            logout_keeping_session!
            @<%= file_name %> = format.blank? ? login_as(<%= file_name %>_login)
: authorize_as(<%= file_name %>_login)
            get login_reqd_status.to_s, :format => format
          end

          if ((login_reqd_status == :login_not_required) ||
              (login_reqd_status == :login_is_required && logged_in_status ==
:i_am_logged_in))
            it "succeeds" do
              response.should have_text(success_text)
```

```
            response.code.to_s.should == '200'
          end

        elsif (login_reqd_status == :login_is_required && logged_in_status ==
:i_am_not_logged_in)
          if ['html', ''].include? format
            it "redirects me to the log in page" do
              response.should redirect_to('/<%= controller_routing_path
%>/new')
            end
          else
            it "returns 'Access denied' and a 406 (Access Denied) status code"
do
              response.should have_text("HTTP Basic: Access denied.\n")
              response.code.to_s.should == '401'
            end
          end

        else
          warn "Oops no case for #{format} and
#{logged_in_status.to_s.humanize} and #{login_reqd_status.to_s.humanize}"
        end
      end # describe

    end
  end
end # cases

end
require File.dirname(__FILE__) + '<%= ('/..'*controller_class_nesting_depth) +
'/../spec_helper' %>'

# Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper
include AuthenticatedSystem
def action_name() end

describe <%= controller_class_name %>Controller do
  fixtures :<%= table_name %>

  before do
    # FIXME -- <%= controller_file_name %> controller not testing xml logins
    stub!(:authenticate_with_http_basic).and_return nil
  end
  describe "logout_killing_session!" do
    before do
      login_as :quentin
      stub!(:reset_session)
    end
    it 'resets the session'         do should_receive(:reset_session);
logout_killing_session! end
    it 'kills my auth_token cookie' do should_receive(:kill_remember_cookie!);
logout_killing_session! end
```

```
    it 'nils the current <%= file_name %>'      do logout_killing_session!;
current_<%= file_name %>.should be_nil end
    it 'kills :<%= file_name %>_id session' do
      session.stub!(:[]=)
      session.should_receive(:[]=).with(:<%= file_name %>_id,
nil).at_least(:once)
      logout_killing_session!
    end
    it 'forgets me' do
      current_<%= file_name %>.remember_me
      current_<%= file_name %>.remember_token.should_not be_nil; current_<%=
file_name %>.remember_token_expires_at.should_not be_nil
      <%= class_name %>.find(1).remember_token.should_not be_nil; <%= class_name
%>.find(1).remember_token_expires_at.should_not be_nil
      logout_killing_session!
      <%= class_name %>.find(1).remember_token.should     be_nil; <%= class_name
%>.find(1).remember_token_expires_at.should     be_nil
    end
  end

  describe "logout_keeping_session!" do
    before do
      login_as :quentin
      stub!(:reset_session)
    end
    it 'does not reset the session' do should_not_receive(:reset_session);
logout_keeping_session! end
    it 'kills my auth_token cookie' do should_receive(:kill_remember_cookie!);
logout_keeping_session! end
    it 'nils the current <%= file_name %>'      do logout_keeping_session!;
current_<%= file_name %>.should be_nil end
    it 'kills :<%= file_name %>_id session' do
      session.stub!(:[]=)
      session.should_receive(:[]=).with(:<%= file_name %>_id,
nil).at_least(:once)
      logout_keeping_session!
    end
    it 'forgets me' do
      current_<%= file_name %>.remember_me
      current_<%= file_name %>.remember_token.should_not be_nil; current_<%=
file_name %>.remember_token_expires_at.should_not be_nil
      <%= class_name %>.find(1).remember_token.should_not be_nil; <%= class_name
%>.find(1).remember_token_expires_at.should_not be_nil
      logout_keeping_session!
      <%= class_name %>.find(1).remember_token.should     be_nil; <%= class_name
%>.find(1).remember_token_expires_at.should     be_nil
    end
  end

  describe 'When logged out' do
    it "should not be authorized?" do
      authorized?().should be_false
    end
  end
```

```
  #
  # Cookie Login
  #
  describe "Logging in by cookie" do
    def set_remember_token token, time
      @<%= file_name %>[:remember_token]            = token;
      @<%= file_name %>[:remember_token_expires_at] = time
      @<%= file_name %>.save!
    end
    before do
      @<%= file_name %> = <%= class_name %>.find(:first);
      set_remember_token 'hello!', 5.minutes.from_now
    end
    it 'logs in with cookie' do
      stub!(:cookies).and_return({ :auth_token => 'hello!' })
      logged_in?.should be_true
    end

    it 'fails cookie login with bad cookie' do
      should_receive(:cookies).at_least(:once).and_return({ :auth_token =>
'i_haxxor_joo' })
      logged_in?.should_not be_true
    end

    it 'fails cookie login with no cookie' do
      set_remember_token nil, nil
      should_receive(:cookies).at_least(:once).and_return({ })
      logged_in?.should_not be_true
    end

    it 'fails expired cookie login' do
      set_remember_token 'hello!', 5.minutes.ago
      stub!(:cookies).and_return({ :auth_token => 'hello!' })
      logged_in?.should_not be_true
    end
  end

end
require File.dirname(__FILE__) + '<%= ('/..'*controller_class_nesting_depth) +
'/../spec_helper' %>'

# Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper

describe <%= controller_class_name %>Controller do
  fixtures        :<%= table_name %>
  before do
    @<%= file_name %>  = mock_<%= file_name %>
    @login_params = { :login => 'quentin', :password => 'test' }
    <%= class_name %>.stub!(:authenticate).with(@login_params[:login],
@login_params[:password]).and_return(@<%= file_name %>)
  end
  def do_create
    post :create, @login_params
```

```
  end
  describe "on successful login," do
    [ [:nil,       nil,            nil],
      [:expired,   'valid_token',  15.minutes.ago],
      [:different, 'i_haxxor_joo', 15.minutes.from_now],
      [:valid,     'valid_token',  15.minutes.from_now]
        ].each do |has_request_token, token_value, token_expiry|
      [ true, false ].each do |want_remember_me|
        describe "my request cookie token is #{has_request_token.to_s}," do
          describe "and ask #{want_remember_me ? 'to' : 'not to'} be remembered"
do
            before do
              @ccookies = mock('cookies')
              controller.stub!(:cookies).and_return(@ccookies)
              @ccookies.stub!(:[]).with(:auth_token).and_return(token_value)
              @ccookies.stub!(:delete).with(:auth_token)
              @ccookies.stub!(:[]=)
              @<%= file_name %>.stub!(:remember_me)
              @<%= file_name %>.stub!(:refresh_token)
              @<%= file_name %>.stub!(:forget_me)
              @<%= file_name %>.stub!(:remember_token).and_return(token_value)
              @<%= file_name
%>.stub!(:remember_token_expires_at).and_return(token_expiry)
              @<%= file_name
%>.stub!(:remember_token?).and_return(has_request_token == :valid)
              if want_remember_me
                @login_params[:remember_me] = '1'
              else
                @login_params[:remember_me] = '0'
              end
            end
            it "kills existing login"        do
controller.should_receive(:logout_keeping_session!); do_create; end
            it "authorizes me"               do do_create;
controller.send(:authorized?).should be_true;   end
            it "logs me in"                  do do_create;
controller.send(:logged_in?).should  be_true  end
            it "greets me nicely"            do do_create;
response.flash[:notice].should =~ /success/i   end
            it "sets/resets/expires cookie"  do
controller.should_receive(:handle_remember_cookie!).with(want_remember_me);
do_create end
            it "sends a cookie"              do
controller.should_receive(:send_remember_cookie!);  do_create end
            it 'redirects to the home page'  do do_create; response.should
redirect_to('/')   end
            it "does not reset my session"   do
controller.should_not_receive(:reset_session).and_return nil; do_create end #
change if you uncomment the reset_session path
            if (has_request_token == :valid)
              it 'does not make new token'   do @<%= file_name
%>.should_not_receive(:remember_me);   do_create end
              it 'does refresh token'        do @<%= file_name
%>.should_receive(:refresh_token);     do_create end
              it "sets an auth cookie"       do do_create;  end
```

```
            else
              if want_remember_me
                it 'makes a new token'       do @<%= file_name
%>.should_receive(:remember_me);       do_create end
                it "does not refresh token"  do @<%= file_name
%>.should_not_receive(:refresh_token); do_create end
                it "sets an auth cookie"       do do_create;  end
              else
                it 'does not make new token' do @<%= file_name
%>.should_not_receive(:remember_me);   do_create end
                it 'does not refresh token'  do @<%= file_name
%>.should_not_receive(:refresh_token); do_create end
                it 'kills user token'        do @<%= file_name
%>.should_receive(:forget_me);         do_create end
              end
            end
          end # inner describe
        end
      end
    end
  end

  describe "on failed login" do
    before do
      <%= class_name %>.should_receive(:authenticate).with(anything(),
anything()).and_return(nil)
      login_as :quentin
    end
    it 'logs out keeping session'   do
controller.should_receive(:logout_keeping_session!); do_create end
    it 'flashes an error'           do do_create; flash[:error].should =~
/Couldn't log you in as 'quentin'/ end
    it 'renders the log in page'    do do_create; response.should
render_template('new')  end
    it "doesn't log me in"          do do_create;
controller.send(:logged_in?).should == false end
    it "doesn't send password back" do
      @login_params[:password] = 'FROBNOZZ'
      do_create
      response.should_not have_text(/FROBNOZZ/i)
    end
  end

  describe "on signout" do
    def do_destroy
      get :destroy
    end
    before do
      login_as :quentin
    end
    it 'logs me out'                    do
controller.should_receive(:logout_killing_session!); do_destroy end
    it 'redirects me to the home page' do do_destroy; response.should
be_redirect     end
  end
```

```
end

describe <%= controller_class_name %>Controller do
  describe "route generation" do
    it "should route the new <%= controller_controller_name %> action correctly"
do
      route_for(:controller => '<%= controller_controller_name %>', :action =>
'new').should == "/login"
    end
    it "should route the create <%= controller_controller_name %> correctly" do
      route_for(:controller => '<%= controller_controller_name %>', :action =>
'create').should == "/<%= controller_routing_path %>"
    end
    it "should route the destroy <%= controller_controller_name %> action
correctly" do
      route_for(:controller => '<%= controller_controller_name %>', :action =>
'destroy').should == "/logout"
    end
  end

  describe "route recognition" do
    it "should generate params from GET /login correctly" do
      params_from(:get, '/login').should == {:controller => '<%=
controller_controller_name %>', :action => 'new'}
    end
    it "should generate params from POST /<%= controller_routing_path %>
correctly" do
      params_from(:post, '/<%= controller_routing_path %>').should ==
{:controller => '<%= controller_controller_name %>', :action => 'create'}
    end
    it "should generate params from DELETE /<%= controller_routing_path %>
correctly" do
      params_from(:delete, '/logout').should == {:controller => '<%=
controller_controller_name %>', :action => 'destroy'}
    end
  end

  describe "named routing" do
    before(:each) do
      get :new
    end
    it "should route <%= controller_routing_name %>_path() correctly" do
      <%= controller_routing_name %>_path().should == "/<%=
controller_routing_path %>"
    end
    it "should route new_<%= controller_routing_name %>_path() correctly" do
      new_<%= controller_routing_name %>_path().should == "/<%=
controller_routing_path %>/new"
    end
  end

end
require File.dirname(__FILE__) + '<%=
('/..'*model_controller_class_nesting_depth) + '/../spec_helper' %>'
```

```
# Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper

describe <%= model_controller_class_name %>Controller do
  fixtures :<%= table_name %>

  it 'allows signup' do
    lambda do
      create_<%= file_name %>
      response.should be_redirect
    end.should change(<%= class_name %>, :count).by(1)
  end

  <% if options[:stateful] %>
  it 'signs up user in pending state' do
    create_<%= file_name %>
    assigns(:<%= file_name %>).reload
    assigns(:<%= file_name %>).should be_pending
  end<% end %>

<% if options[:include_activation] -%>
  it 'signs up user with activation code' do
    create_<%= file_name %>
    assigns(:<%= file_name %>).reload
    assigns(:<%= file_name %>).activation_code.should_not be_nil
  end<% end -%>

  it 'requires login on signup' do
    lambda do
      create_<%= file_name %>(:login => nil)
      assigns[:<%= file_name %>].errors.on(:login).should_not be_nil
      response.should be_success
    end.should_not change(<%= class_name %>, :count)
  end

  it 'requires password on signup' do
    lambda do
      create_<%= file_name %>(:password => nil)
      assigns[:<%= file_name %>].errors.on(:password).should_not be_nil
      response.should be_success
    end.should_not change(<%= class_name %>, :count)
  end

  it 'requires password confirmation on signup' do
    lambda do
      create_<%= file_name %>(:password_confirmation => nil)
      assigns[:<%= file_name %>].errors.on(:password_confirmation).should_not
be_nil
      response.should be_success
    end.should_not change(<%= class_name %>, :count)
  end

  it 'requires email on signup' do
```

```
    lambda do
      create_<%= file_name %>(:email => nil)
      assigns[:<%= file_name %>].errors.on(:email).should_not be_nil
      response.should be_success
    end.should_not change(<%= class_name %>, :count)
  end

  <% if options[:include_activation] %>
  it 'activates user' do
    <%= class_name %>.authenticate('aaron', 'monkey').should be_nil
    get :activate, :activation_code => <%= table_name %>(:aaron).activation_code
    response.should redirect_to('/login')
    flash[:notice].should_not be_nil
    flash[:error ].should     be_nil
    <%= class_name %>.authenticate('aaron', 'monkey').should == <%= table_name
%>(:aaron)
  end

  it 'does not activate user without key' do
    get :activate
    flash[:notice].should     be_nil
    flash[:error ].should_not be_nil
  end

  it 'does not activate user with blank key' do
    get :activate, :activation_code => ''
    flash[:notice].should     be_nil
    flash[:error ].should_not be_nil
  end

  it 'does not activate user with bogus key' do
    get :activate, :activation_code => 'i_haxxor_joo'
    flash[:notice].should     be_nil
    flash[:error ].should_not be_nil
  end<% end %>

  def create_<%= file_name %>(options = {})
    post :create, :<%= file_name %> => { :login => 'quire', :email =>
'quire@example.com',
      :password => 'quire69', :password_confirmation => 'quire69'
}.merge(options)
  end
end

describe <%= model_controller_class_name %>Controller do
  describe "route generation" do
    it "should route <%= model_controller_controller_name %>'s 'index' action
correctly" do
      route_for(:controller => '<%= model_controller_controller_name %>',
:action => 'index').should == "/<%= model_controller_routing_path %>"
    end

    it "should route <%= model_controller_controller_name %>'s 'new' action
correctly" do
```

```
      route_for(:controller => '<%= model_controller_controller_name %>',
:action => 'new').should == "/signup"
    end

    it "should route {:controller => '<%= model_controller_controller_name %>',
:action => 'create'} correctly" do
      route_for(:controller => '<%= model_controller_controller_name %>',
:action => 'create').should == "/register"
    end

    it "should route <%= model_controller_controller_name %>'s 'show' action
correctly" do
      route_for(:controller => '<%= model_controller_controller_name %>',
:action => 'show', :id => '1').should == "/<%= model_controller_routing_path
%>/1"
    end

    it "should route <%= model_controller_controller_name %>'s 'edit' action
correctly" do
      route_for(:controller => '<%= model_controller_controller_name %>',
:action => 'edit', :id => '1').should == "/<%= model_controller_routing_path
%>/1/edit"
    end

    it "should route <%= model_controller_controller_name %>'s 'update' action
correctly" do
      route_for(:controller => '<%= model_controller_controller_name %>',
:action => 'update', :id => '1').should == "/<%= model_controller_routing_path
%>/1"
    end

    it "should route <%= model_controller_controller_name %>'s 'destroy' action
correctly" do
      route_for(:controller => '<%= model_controller_controller_name %>',
:action => 'destroy', :id => '1').should == "/<%= model_controller_routing_path
%>/1"
    end
  end

  describe "route recognition" do
    it "should generate params for <%= model_controller_controller_name %>'s
index action from GET /<%= model_controller_routing_path %>" do
      params_from(:get, '/<%= model_controller_routing_path %>').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'index'}
      params_from(:get, '/<%= model_controller_routing_path %>.xml').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'index',
:format => 'xml'}
      params_from(:get, '/<%= model_controller_routing_path %>.json').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'index',
:format => 'json'}
    end

    it "should generate params for <%= model_controller_controller_name %>'s new
action from GET /<%= model_controller_routing_path %>" do
```

```
      params_from(:get, '/<%= model_controller_routing_path %>/new').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'new'}
      params_from(:get, '/<%= model_controller_routing_path %>/new.xml').should
== {:controller => '<%= model_controller_controller_name %>', :action => 'new',
:format => 'xml'}
      params_from(:get, '/<%= model_controller_routing_path %>/new.json').should
== {:controller => '<%= model_controller_controller_name %>', :action => 'new',
:format => 'json'}
    end

    it "should generate params for <%= model_controller_controller_name %>'s
create action from POST /<%= model_controller_routing_path %>" do
      params_from(:post, '/<%= model_controller_routing_path %>').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'create'}
      params_from(:post, '/<%= model_controller_routing_path %>.xml').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'create',
:format => 'xml'}
      params_from(:post, '/<%= model_controller_routing_path %>.json').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'create',
:format => 'json'}
    end

    it "should generate params for <%= model_controller_controller_name %>'s
show action from GET /<%= model_controller_routing_path %>/1" do
      params_from(:get , '/<%= model_controller_routing_path %>/1').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'show',
:id => '1'}
      params_from(:get , '/<%= model_controller_routing_path %>/1.xml').should
== {:controller => '<%= model_controller_controller_name %>', :action => 'show',
:id => '1', :format => 'xml'}
      params_from(:get , '/<%= model_controller_routing_path %>/1.json').should
== {:controller => '<%= model_controller_controller_name %>', :action => 'show',
:id => '1', :format => 'json'}
    end

    it "should generate params for <%= model_controller_controller_name %>'s
edit action from GET /<%= model_controller_routing_path %>/1/edit" do
      params_from(:get , '/<%= model_controller_routing_path %>/1/edit').should
== {:controller => '<%= model_controller_controller_name %>', :action => 'edit',
:id => '1'}
    end

    it "should generate params {:controller => '<%=
model_controller_controller_name %>', :action => update', :id => '1'} from PUT
/<%= model_controller_routing_path %>/1" do
      params_from(:put , '/<%= model_controller_routing_path %>/1').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'update',
:id => '1'}
      params_from(:put , '/<%= model_controller_routing_path %>/1.xml').should
== {:controller => '<%= model_controller_controller_name %>', :action =>
'update', :id => '1', :format => 'xml'}
      params_from(:put , '/<%= model_controller_routing_path %>/1.json').should
== {:controller => '<%= model_controller_controller_name %>', :action =>
'update', :id => '1', :format => 'json'}
    end
```

```
    it "should generate params for <%= model_controller_controller_name %>'s
destroy action from DELETE /<%= model_controller_routing_path %>/1" do
      params_from(:delete, '/<%= model_controller_routing_path %>/1').should ==
{:controller => '<%= model_controller_controller_name %>', :action => 'destroy',
:id => '1'}
      params_from(:delete, '/<%= model_controller_routing_path %>/1.xml').should
== {:controller => '<%= model_controller_controller_name %>', :action =>
'destroy', :id => '1', :format => 'xml'}
      params_from(:delete, '/<%= model_controller_routing_path
%>/1.json').should == {:controller => '<%= model_controller_controller_name %>',
:action => 'destroy', :id => '1', :format => 'json'}
    end
  end

  describe "named routing" do
    before(:each) do
      get :new
    end

    it "should route <%= model_controller_routing_name %>_path() to /<%=
model_controller_routing_path %>" do
      <%= model_controller_routing_name %>_path().should == "/<%=
model_controller_routing_path %>"
      formatted_<%= model_controller_routing_name %>_path(:format =>
'xml').should == "/<%= model_controller_routing_path %>.xml"
      formatted_<%= model_controller_routing_name %>_path(:format =>
'json').should == "/<%= model_controller_routing_path %>.json"
    end

    it "should route new_<%= model_controller_routing_name.singularize %>_path()
to /<%= model_controller_routing_path %>/new" do
      new_<%= model_controller_routing_name.singularize %>_path().should ==
"/<%= model_controller_routing_path %>/new"
      formatted_new_<%= model_controller_routing_name.singularize
%>_path(:format => 'xml').should == "/<%= model_controller_routing_path
%>/new.xml"
      formatted_new_<%= model_controller_routing_name.singularize
%>_path(:format => 'json').should == "/<%= model_controller_routing_path
%>/new.json"
    end

    it "should route <%= model_controller_routing_name.singularize %>_(:id =>
'1') to /<%= model_controller_routing_path %>/1" do
      <%= model_controller_routing_name.singularize %>_path(:id => '1').should
== "/<%= model_controller_routing_path %>/1"
      formatted_<%= model_controller_routing_name.singularize %>_path(:id =>
'1', :format => 'xml').should == "/<%= model_controller_routing_path %>/1.xml"
      formatted_<%= model_controller_routing_name.singularize %>_path(:id =>
'1', :format => 'json').should == "/<%= model_controller_routing_path %>/1.json"
    end

    it "should route edit_<%= model_controller_routing_name.singularize
%>_path(:id => '1') to /<%= model_controller_routing_path %>/1/edit" do
```

```
      edit_<%= model_controller_routing_name.singularize %>_path(:id =>
'1').should == "/<%= model_controller_routing_path %>/1/edit"
    end
  end

end
require File.dirname(__FILE__) + '<%=
('/..'*model_controller_class_nesting_depth) + '/../spec_helper' %>'
include ApplicationHelper
include <%= model_controller_class_name %>Helper
include AuthenticatedTestHelper

describe <%= model_controller_class_name %>Helper do
  before do
    @<%= file_name %> = mock_<%= file_name %>
  end

  describe "if_authorized" do
    it "yields if authorized" do
      should_receive(:authorized?).with('a','r').and_return(true)
      if_authorized?('a','r'){|action,resource| [action,resource,'hi'] }.should
== ['a','r','hi']
    end
    it "does nothing if not authorized" do
      should_receive(:authorized?).with('a','r').and_return(false)
      if_authorized?('a','r'){ 'hi' }.should be_nil
    end
  end

  describe "link_to_<%= file_name %>" do
    it "should give an error on a nil <%= file_name %>" do
      lambda { link_to_<%= file_name %>(nil) }.should raise_error('Invalid <%=
file_name %>')
    end
    it "should link to the given <%= file_name %>" do
      should_receive(:<%= model_controller_routing_name.singularize
%>_path).at_least(:once).and_return('/<%= model_controller_file_path %>/1')
      link_to_<%= file_name %>(@<%= file_name %>).should have_tag("a[href='/<%=
model_controller_file_path %>/1']")
    end
    it "should use given link text if :content_text is specified" do
      link_to_<%= file_name %>(@<%= file_name %>, :content_text => 'Hello
there!').should have_tag("a", 'Hello there!')
    end
    it "should use the login as link text with no :content_method specified" do
      link_to_<%= file_name %>(@<%= file_name %>).should have_tag("a",
'user_name')
    end
    it "should use the name as link text with :content_method => :name" do
      link_to_<%= file_name %>(@<%= file_name %>, :content_method =>
:name).should have_tag("a", 'U. Surname')
    end
    it "should use the login as title with no :title_method specified" do
      link_to_<%= file_name %>(@<%= file_name %>).should
have_tag("a[title='user_name']")
```

```
    end
    it "should use the name as link title with :content_method => :name" do
      link_to_<%= file_name %>(@<%= file_name %>, :title_method => :name).should
have_tag("a[title='U. Surname']")
    end
    it "should have nickname as a class by default" do
      link_to_<%= file_name %>(@<%= file_name %>).should have_tag("a.nickname")
    end
    it "should take other classes and no longer have the nickname class" do
      result = link_to_<%= file_name %>(@<%= file_name %>, :class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
  end

  describe "link_to_login_with_IP" do
    it "should link to the login_path" do
      link_to_login_with_IP().should have_tag("a[href='/login']")
    end
    it "should use given link text if :content_text is specified" do
      link_to_login_with_IP('Hello there!').should have_tag("a", 'Hello there!')
    end
    it "should use the login as link text with no :content_method specified" do
      link_to_login_with_IP().should have_tag("a", '0.0.0.0')
    end
    it "should use the ip address as title" do
      link_to_login_with_IP().should have_tag("a[title='0.0.0.0']")
    end
    it "should by default be like school in summer and have no class" do
      link_to_login_with_IP().should_not have_tag("a.nickname")
    end
    it "should have some class if you tell it to" do
      result = link_to_login_with_IP(nil, :class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
    it "should have some class if you tell it to" do
      result = link_to_login_with_IP(nil, :tag => 'abbr')
      result.should have_tag("abbr[title='0.0.0.0']")
    end
  end

  describe "link_to_current_<%= file_name %>, When logged in" do
    before do
      stub!(:current_<%= file_name %>).and_return(@<%= file_name %>)
    end
    it "should link to the given <%= file_name %>" do
      should_receive(:<%= model_controller_routing_name.singularize
%>_path).at_least(:once).and_return('/<%= model_controller_file_path %>/1')
      link_to_current_<%= file_name %>().should have_tag("a[href='/<%=
model_controller_file_path %>/1']")
    end
    it "should use given link text if :content_text is specified" do
      link_to_current_<%= file_name %>(:content_text => 'Hello there!').should
have_tag("a", 'Hello there!')
```

```
    end
    it "should use the login as link text with no :content_method specified" do
      link_to_current_<%= file_name %>().should have_tag("a", 'user_name')
    end
    it "should use the name as link text with :content_method => :name" do
      link_to_current_<%= file_name %>(:content_method => :name).should
have_tag("a", 'U. Surname')
    end
    it "should use the login as title with no :title_method specified" do
      link_to_current_<%= file_name %>().should have_tag("a[title='user_name']")
    end
    it "should use the name as link title with :content_method => :name" do
      link_to_current_<%= file_name %>(:title_method => :name).should
have_tag("a[title='U. Surname']")
    end
    it "should have nickname as a class" do
      link_to_current_<%= file_name %>().should have_tag("a.nickname")
    end
    it "should take other classes and no longer have the nickname class" do
      result = link_to_current_<%= file_name %>(:class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
  end

  describe "link_to_current_<%= file_name %>, When logged out" do
    before do
      stub!(:current_<%= file_name %>).and_return(nil)
    end
    it "should link to the login_path" do
      link_to_current_<%= file_name %>().should have_tag("a[href='/login']")
    end
    it "should use given link text if :content_text is specified" do
      link_to_current_<%= file_name %>(:content_text => 'Hello there!').should
have_tag("a", 'Hello there!')
    end
    it "should use 'not signed in' as link text with no :content_method
specified" do
      link_to_current_<%= file_name %>().should have_tag("a", 'not signed in')
    end
    it "should use the ip address as title" do
      link_to_current_<%= file_name %>().should have_tag("a[title='0.0.0.0']")
    end
    it "should by default be like school in summer and have no class" do
      link_to_current_<%= file_name %>().should_not have_tag("a.nickname")
    end
    it "should have some class if you tell it to" do
      result = link_to_current_<%= file_name %>(:class => 'foo bar')
      result.should have_tag("a.foo")
      result.should have_tag("a.bar")
    end
  end

end
# -*- coding: utf-8 -*-
```

```
require File.dirname(__FILE__) + '<%=
('/..'*model_controller_class_nesting_depth) + '/../spec_helper' %>'

# Be sure to include AuthenticatedTestHelper in spec/spec_helper.rb instead.
# Then, you can remove it from this and the functional test.
include AuthenticatedTestHelper

describe <%= class_name %> do
  fixtures :<%= table_name %>

  describe 'being created' do
    before do
      @<%= file_name %> = nil
      @creating_<%= file_name %> = lambda do
        @<%= file_name %> = create_<%= file_name %>
        violated "#{@<%= file_name %>.errors.full_messages.to_sentence}" if @<%=
file_name %>.new_record?
      end
    end

    it 'increments <%= class_name %>#count' do
      @creating_<%= file_name %>.should change(<%= class_name %>, :count).by(1)
    end
<% if options[:include_activation] %>
    it 'initializes #activation_code' do
      @creating_<%= file_name %>.call
      @<%= file_name %>.reload
      @<%= file_name %>.activation_code.should_not be_nil
    end
<% end %><% if options[:stateful] %>
    it 'starts in pending state' do
      @creating_<%= file_name %>.call
      @<%= file_name %>.reload
      @<%= file_name %>.should be_pending
    end
<% end %>  end

  #
  # Validations
  #

  it 'requires login' do
    lambda do
      u = create_<%= file_name %>(:login => nil)
      u.errors.on(:login).should_not be_nil
    end.should_not change(<%= class_name %>, :count)
  end

  describe 'allows legitimate logins:' do
    ['123', '1234567890_234567890_234567890_234567890',
     'hello.-_there@funnychar.com'].each do |login_str|
      it "'#{login_str}'" do
        lambda do
          u = create_<%= file_name %>(:login => login_str)
          u.errors.on(:login).should     be_nil
```

```
        end.should change(<%= class_name %>, :count).by(1)
      end
    end
  end
  describe 'disallows illegitimate logins:' do
    ['12', '1234567890_234567890_234567890_234567890_', "tab\t", "newline\n",
     "I√±t√´rn√¢ti√¥n√†liz√¶ti√∏n hasn't happened to ruby 1.8 yet",
     'semicolon;', 'quote"', 'tick\'', 'backtick`', 'percent%', 'plus+', 'space
'].each do |login_str|
      it "'#{login_str}'" do
        lambda do
          u = create_<%= file_name %>(:login => login_str)
          u.errors.on(:login).should_not be_nil
        end.should_not change(<%= class_name %>, :count)
      end
    end
  end

  it 'requires password' do
    lambda do
      u = create_<%= file_name %>(:password => nil)
      u.errors.on(:password).should_not be_nil
    end.should_not change(<%= class_name %>, :count)
  end

  it 'requires password confirmation' do
    lambda do
      u = create_<%= file_name %>(:password_confirmation => nil)
      u.errors.on(:password_confirmation).should_not be_nil
    end.should_not change(<%= class_name %>, :count)
  end

  it 'requires email' do
    lambda do
      u = create_<%= file_name %>(:email => nil)
      u.errors.on(:email).should_not be_nil
    end.should_not change(<%= class_name %>, :count)
  end

  describe 'allows legitimate emails:' do
    ['foo@bar.com', 'foo@newskool-tld.museum', 'foo@twoletter-tld.de',
'foo@nonexistant-tld.qq',
     'r@a.wk', '1234567890-234567890-234567890-234567890-234567890-234567890-
234567890-234567890-234567890@gmail.com',
     'hello.-_there@funnychar.com', 'uucp%addr@gmail.com', 'hello+routing-
str@gmail.com',
     'domain@can.haz.many.sub.doma.in', 'student.name@university.edu'
    ].each do |email_str|
      it "'#{email_str}'" do
        lambda do
          u = create_<%= file_name %>(:email => email_str)
          u.errors.on(:email).should     be_nil
        end.should change(<%= class_name %>, :count).by(1)
      end
    end
```

```
  end
  describe 'disallows illegitimate emails' do
    ['!!@nobadchars.com', 'foo@no-rep-dots..com', 'foo@badtld.xxx',
'foo@toolongtld.abcdefg',
     'I√±t√´rn√¢ti√¥n√†liz√¶ti√∏n@hasnt.happened.to.email',
'need.domain.and.tld@de', "tab\t", "newline\n",
     'r@.wk', '1234567890-234567890-234567890-234567890-234567890-234567890-
234567890-234567890-234567890@gmail2.com',
     # these are technically allowed but not seen in practice:
     'uucp!addr@gmail.com', 'semicolon;@gmail.com', 'quote"@gmail.com',
'tick\'@gmail.com', 'backtick`@gmail.com', 'space @gmail.com',
'bracket<@gmail.com', 'bracket>@gmail.com'
    ].each do |email_str|
      it "'#{email_str}'" do
        lambda do
          u = create_<%= file_name %>(:email => email_str)
          u.errors.on(:email).should_not be_nil
        end.should_not change(<%= class_name %>, :count)
      end
    end
  end

  describe 'allows legitimate names:' do
    ['Andre The Giant (7\'4", 520 lb.) -- has a posse',
     '',
'1234567890_234567890_234567890_234567890_234567890_234567890_234567890_23456789
0_234567890_234567890',
    ].each do |name_str|
      it "'#{name_str}'" do
        lambda do
          u = create_<%= file_name %>(:name => name_str)
          u.errors.on(:name).should     be_nil
        end.should change(<%= class_name %>, :count).by(1)
      end
    end
  end
  describe "disallows illegitimate names" do
    ["tab\t", "newline\n",

'1234567890_234567890_234567890_234567890_234567890_234567890_234567890_23456789
0_234567890_234567890_',
     ].each do |name_str|
      it "'#{name_str}'" do
        lambda do
          u = create_<%= file_name %>(:name => name_str)
          u.errors.on(:name).should_not be_nil
        end.should_not change(<%= class_name %>, :count)
      end
    end
  end

  it 'resets password' do
    <%= table_name %>(:quentin).update_attributes(:password => 'new password',
:password_confirmation => 'new password')
```

```
    <%= class_name %>.authenticate('quentin', 'new password').should == <%=
table_name %>(:quentin)
  end

  it 'does not rehash password' do
    <%= table_name %>(:quentin).update_attributes(:login => 'quentin2')
    <%= class_name %>.authenticate('quentin2', 'monkey').should == <%=
table_name %>(:quentin)
  end

  #
  # Authentication
  #

  it 'authenticates <%= file_name %>' do
    <%= class_name %>.authenticate('quentin', 'monkey').should == <%= table_name
%>(:quentin)
  end

  it "doesn't authenticate <%= file_name %> with bad password" do
    <%= class_name %>.authenticate('quentin', 'invalid_password').should be_nil
  end

 if REST_AUTH_SITE_KEY.blank?
   # old-school passwords
   it "authenticates a user against a hard-coded old-style password" do
     <%= class_name %>.authenticate('old_password_holder', 'test').should == <%=
table_name %>(:old_password_holder)
   end
 else
   it "doesn't authenticate a user against a hard-coded old-style password" do
     <%= class_name %>.authenticate('old_password_holder', 'test').should be_nil
   end

   # New installs should bump this up and set REST_AUTH_DIGEST_STRETCHES to give
a 10ms encrypt time or so
   desired_encryption_expensiveness_ms = 0.1
   it "takes longer than #{desired_encryption_expensiveness_ms}ms to encrypt a
password" do
     test_reps = 100
     start_time = Time.now; test_reps.times{ <%= class_name
%>.authenticate('quentin', 'monkey'+rand.to_s) }; end_time   = Time.now
     auth_time_ms = 1000 * (end_time - start_time)/test_reps
     auth_time_ms.should > desired_encryption_expensiveness_ms
   end
 end

  #
  # Authentication
  #

  it 'sets remember token' do
    <%= table_name %>(:quentin).remember_me
    <%= table_name %>(:quentin).remember_token.should_not be_nil
    <%= table_name %>(:quentin).remember_token_expires_at.should_not be_nil
```

```
  end

  it 'unsets remember token' do
    <%= table_name %>(:quentin).remember_me
    <%= table_name %>(:quentin).remember_token.should_not be_nil
    <%= table_name %>(:quentin).forget_me
    <%= table_name %>(:quentin).remember_token.should be_nil
  end

  it 'remembers me for one week' do
    before = 1.week.from_now.utc
    <%= table_name %>(:quentin).remember_me_for 1.week
    after = 1.week.from_now.utc
    <%= table_name %>(:quentin).remember_token.should_not be_nil
    <%= table_name %>(:quentin).remember_token_expires_at.should_not be_nil
    <%= table_name %>(:quentin).remember_token_expires_at.between?(before,
after).should be_true
  end

  it 'remembers me until one week' do
    time = 1.week.from_now.utc
    <%= table_name %>(:quentin).remember_me_until time
    <%= table_name %>(:quentin).remember_token.should_not be_nil
    <%= table_name %>(:quentin).remember_token_expires_at.should_not be_nil
    <%= table_name %>(:quentin).remember_token_expires_at.should == time
  end

  it 'remembers me default two weeks' do
    before = 2.weeks.from_now.utc
    <%= table_name %>(:quentin).remember_me
    after = 2.weeks.from_now.utc
    <%= table_name %>(:quentin).remember_token.should_not be_nil
    <%= table_name %>(:quentin).remember_token_expires_at.should_not be_nil
    <%= table_name %>(:quentin).remember_token_expires_at.between?(before,
after).should be_true
  end
<% if options[:stateful] %>
  it 'registers passive <%= file_name %>' do
    <%= file_name %> = create_<%= file_name %>(:password => nil,
:password_confirmation => nil)
    <%= file_name %>.should be_passive
    <%= file_name %>.update_attributes(:password => 'new password',
:password_confirmation => 'new password')
    <%= file_name %>.register!
    <%= file_name %>.should be_pending
  end

  it 'suspends <%= file_name %>' do
    <%= table_name %>(:quentin).suspend!
    <%= table_name %>(:quentin).should be_suspended
  end

  it 'does not authenticate suspended <%= file_name %>' do
    <%= table_name %>(:quentin).suspend!
```

```
    <%= class_name %>.authenticate('quentin', 'monkey').should_not == <%=
table_name %>(:quentin)
  end

  it 'deletes <%= file_name %>' do
    <%= table_name %>(:quentin).deleted_at.should be_nil
    <%= table_name %>(:quentin).delete!
    <%= table_name %>(:quentin).deleted_at.should_not be_nil
    <%= table_name %>(:quentin).should be_deleted
  end

  describe "being unsuspended" do
    fixtures :<%= table_name %>

    before do
      @<%= file_name %> = <%= table_name %>(:quentin)
      @<%= file_name %>.suspend!
    end

    it 'reverts to active state' do
      @<%= file_name %>.unsuspend!
      @<%= file_name %>.should be_active
    end

    it 'reverts to passive state if activation_code and activated_at are nil' do
      <%= class_name %>.update_all :activation_code => nil, :activated_at => nil
      @<%= file_name %>.reload.unsuspend!
      @<%= file_name %>.should be_passive
    end

    it 'reverts to pending state if activation_code is set and activated_at is
nil' do
      <%= class_name %>.update_all :activation_code => 'foo-bar', :activated_at
=> nil
      @<%= file_name %>.reload.unsuspend!
      @<%= file_name %>.should be_pending
    end
  end
<% end %>
protected
  def create_<%= file_name %>(options = {})
    record = <%= class_name %>.new({ :login => 'quire', :email =>
'quire@example.com', :password => 'quire69', :password_confirmation => 'quire69'
}.merge(options))
    record.<% if options[:stateful] %>register! if record.valid?<% else %>save<%
end %>
    record
  end
end
#!/usr/bin/env ruby
ENV["RAILS_ENV"] = "test"
require File.expand_path(File.dirname(__FILE__) + "/../config/environment")
require 'spec/rails/story_adapter'
require 'spec/story'
```

```
require File.expand_path(File.dirname(__FILE__) +
"/rest_auth_stories_helper.rb")

# Make visible for testing
ApplicationController.send(:public, :logged_in?, :current_user, :authorized?)

this_dir = File.dirname(__FILE__)
Dir[File.join(this_dir, "steps/*.rb")].each do |file|
  puts file.to_s
  require file
end

with_steps_for :ra_navigation, :ra_response, :ra_resource, :<%= file_name %> do
  story_files = Dir[File.join(this_dir, "<%= table_name %>", '*.story')]
  story_files.each do |file|
    run file, :type => RailsStory
  end
end
# If you have a global stories helper, move this line there:
include AuthenticatedTestHelper

# Most of the below came out of code from Ben Mabey
# http://www.benmabey.com/2008/02/04/rspec-plain-text-stories-webrat-chunky-
bacon/

# These allow exceptions to come through as opposed to being caught and having
non-helpful responses returned.
ActionController::Base.class_eval do
  def perform_action
    perform_action_without_rescue
  end
end
Dispatcher.class_eval do
  def self.failsafe_response(output, status, exception = nil)
    raise exception
  end
end

#
# Sugar for turning a story's attribute list into list, array, etc.
#
module ToFooFromStory
  def ToFooFromStory.fix_key key
    key.downcase.gsub(/\s+/, '_')
  end
  def ToFooFromStory.fix_value value
    return '' if !value
    value.strip!
    case
    when value =~ /^'(.*)'$/    then value = $1
    when value =~ /^"(.*)"$/    then value = $1
    when value == 'nil!'        then value = nil
    when value == 'non-nil!'    then value = be_nil
    when value =~ /^#\{(.*)\}$/ then value = eval($1)
    end
```

```
    value
  end
  # Converts a key: value list found in the steps into a hash.
  # Example:
  #   ISBN: '0967539854' and comment: 'I love this book' and Quality rating: '4'
  #   # => {"quality_rating"=>"4", "isbn"=>"0967539854", "comment"=>"I love this
book"}
  def to_hash_from_story
    hsh = self.split(/,? and |, /).inject({}) do |hash_so_far, key_value|
      key, value = key_value.split(":")
      if !value then warn "Couldn't understand story '#{self}': only understood
up to the part '#{hash_so_far.to_yaml}'" end
      hash_so_far.merge(ToFooFromStory::fix_key(key) =>
ToFooFromStory::fix_value(value))
    end
  end
  # Coverts an attribute list found in the steps into an array
  # Example:
  #   login, email, updated_at, and gravatar
  #   # => ['login', 'email', 'updated_at', 'gravatar']
  def to_array_from_story
    self.split(/,? and |, /).map do |value|
      ToFooFromStory::fix_value(value)
    end
  end
end
class String
  include ToFooFromStory
end

def instantize(string)
  instance_variable_get("@#{string}")
end

#
# Spew response onto screen -- painful but scrolling >> debugger
#
def dump_response
  # note that @request and @template won't to_yaml and that @session includes
@cgi
  response_methods = response.instance_variables         - ['@request',
'@template', '@cgi']
  request_methods  = response.request.instance_variables -
['@session_options_with_string_keys', '@cgi', '@session']
  response_methods.map!{|attr| attr.gsub(/^@/,'')}.sort!
  request_methods.map!{ |attr| attr.gsub(/^@/,'')}.sort!
  puts '', '*' * 75,
    response.instance_values.slice(*response_methods).to_yaml,
    "*" * 75, '',
    response.request.instance_values.slice(*request_methods).to_yaml,
    "*" * 75, ''
end
#
# Where to go
#
```

```
steps_for(:ra_navigation) do
  #
  # GET
  # Go to a given page.
  When "$actor goes to $path" do |actor, path|
    case path
    when 'the home page' then get '/'
    else                      get path
    end
  end

  # POST -- Ex:
  #   When she creates a book with ISBN: '0967539854' and comment: 'I love this
book' and rating: '4'
  #   When she creates a singular session with login: 'reggie' and password:
'i_haxxor_joo'
  # Since I'm not smrt enough to do it right, explicitly specify singular
resources
  When %r{$actor creates an? $resource with $attributes} do |actor, resource,
attributes|
    attributes = attributes.to_hash_from_story
    if resource =~ %r{singular ([\w/]+)}
      resource = $1.downcase.singularize
      post "/#{resource}", attributes
    else
      post "/#{resource.downcase.pluralize}", { resource.downcase.singularize =>
attributes }
    end
  end

  # PUT
  When %r{$actor asks to update '$resource' with $attributes} do |_, resource,
attributes|
    attributes = attributes.to_hash_from_story
    put "#{resource}", attributes
    dump_response
  end

  # DELETE -- Slap together the POST-form-as-fake-HTTP-DELETE submission
  When %r{$actor asks to delete '$resource'} do |_, resource|
    post "/#{resource.downcase.pluralize}", { :_method => :delete }
    dump_response
  end


  # Redirect --
  #   Rather than coding in get/get_via_redirect's and past/p_v_r's,
  #   let's just demand that in the story itself.
  When "$actor follows that redirect!" do |actor|
    follow_redirect!
  end
end
# The flexible code for resource testing came out of code from Ben Mabey
# http://www.benmabey.com/2008/02/04/rspec-plain-text-stories-webrat-chunky-
bacon/
```

```
steps_for(:ra_resource) do
  #
  # Construct resources
  #

  #
  # Build a resource as described, store it as an @instance variable. Ex:
  #   "Given a <%= file_name %> with login: 'mojojojo'"
  # produces a <%= class_name %> instance stored in @<%= file_name %> with
'mojojojo' as its login
  # attribute.
  #
  Given "a $resource instance with $attributes" do |resource, attributes|
    klass, instance, attributes = parse_resource_args resource, attributes
    instance = klass.new(attributes)
    instance.save!
    find_resource(resource, attributes).should_not be_nil
    keep_instance! resource, instance
  end

  #
  # Stuff attributes into a preexisting @resource
  #   "And the <%= file_name %> has thac0: 3"
  # takes the earlier-defined @<%= file_name %> instance and sets its thac0 to
'3'.
  #
  Given "the $resource has $attributes" do |resource, attributes|
    klass, instance, attributes = parse_resource_args resource, attributes
    attributes.each do |attr, val|
      instance.send("#{attr}=", val)
    end
    instance.save!
    find_resource(resource, attributes).should_not be_nil
    keep_instance! resource, instance
  end

  #
  # Destroy all for this resource
  #
  Given "no $resource with $attr: '$val' exists" do |resource, attr, val|
    klass, instance = parse_resource_args resource
    klass.destroy_all(attr.to_sym => val)
    instance = find_resource resource, attr.to_sym => val
    instance.should be_nil
    keep_instance! resource, instance
  end

  #
  # Then's for resources
  #

  # Resource like this DOES exist
  Then %r{an? $resource with $attributes should exist} do |resource, attributes|
    instance = find_resource resource, attributes
    instance.should_not be_nil
```

```
    keep_instance! resource, instance
  end
  # Resource like this DOES NOT exist
  Then %r{no $resource with $attributes should exist} do |resource, attributes|
    instance = find_resource resource, attributes
    instance.should be_nil
  end

  # Resource has attributes with given values
  Then  "the $resource should have $attributes" do |resource, attributes|
    klass, instance, attributes = parse_resource_args resource, attributes
    attributes.each do |attr, val|
      instance.send(attr).should == val
    end
  end
  # Resource attributes should / should not be nil
  Then  "the $resource's $attr should be nil" do |resource, attr|
    klass, instance = parse_resource_args resource
    instance.send(attr).should be_nil
  end
  Then  "the $resource's $attr should not be nil" do |resource, attr|
    klass, instance = parse_resource_args resource
    instance.send(attr).should_not be_nil
  end

  #
  # Bank each of the @resource's listed attributes for later.
  #
  Given "we try hard to remember the $resource's $attributes" do |resource,
attributes|
    attributes = attributes.to_array_from_story
    attributes.each do |attr|
      memorize_resource_value resource, attr
    end
  end
  #
  # Bank each of the @resource's listed attributes for later.
  #
  Given "we don't remember anything about the past" do
    memorize_forget_all!
  end

  #
  # Compare @resource.attr to its earlier-memorized value.
  # Specify ' using method_name' (abs, to_s, &c) to coerce before comparing.
  # For important and mysterious reasons, timestamps want to_i or to_s.
  #
  Then %r{the $resource\'s $attribute should stay the same(?: under $func)?} do
|resource, attr, func|
    klass, instance = parse_resource_args resource
    # Get the values
    old_value = recall_resource_value(resource, attr)
    new_value = instance.send(attr)
    # Transform each value, maybe, using value.func
```

```
    if func then new_value = new_value.send(func); old_value =
old_value.send(func) end
    # Compare
    old_value.should eql(new_value)
  end

  #
  # Look for each for the given attributes in the page's text
  #
  Then "page should have the $resource's $attributes" do |resource, attributes|
    actual_resource = instantize(resource)
    attributes.split(/, and |, /).each do |attribute|
      response.should have_text(/#{actual_resource.send(attribute.strip.gsub("
","_"))}/)
    end
  end

end

#
# Turn a resource name and a to_hash_from_story string like
#   "attr: 'value', attr2: 'value2', ... , and attrN: 'valueN'"
# into
#   * klass      -- the class matching that Resource
#   * instance   -- the possibly-preexisting local instance value @resource
#   * attributes -- a hash matching the given attribute-list string
#
def parse_resource_args resource, attributes=nil
  instance   = instantize resource
  klass      = resource.classify.constantize
  attributes = attributes.to_hash_from_story if attributes
  [klass, instance, attributes]
end

#
# Given a class name 'resource' and a hash of conditsion, find a model
#
def find_resource resource, conditions
  klass, instance = parse_resource_args resource
  conditions = conditions.to_hash_from_story unless (conditions.is_a? Hash)
  klass.find(:first, :conditions => conditions)
end

#
# Simple, brittle, useful: store the given resource's attribute
# so we can compare it later.
#
def memorize_resource_value resource, attr
  klass, instance = parse_resource_args resource
  value = instance.send(attr)
  @_memorized ||= {}
  @_memorized[resource] ||= {}
  @_memorized[resource][attr] = value
  value
end
```

```
def recall_resource_value resource, attr
  @_memorized[resource][attr]
end
def memorize_forget_all!
  @_memorized = {}
end

#
# Keep the object around in a local instance variable @resource.
#
# So, for instance,
#   klass, instance = parse_resource_args '<%= file_name %>'
#   instance = klass.new({login => 'me', password => 'monkey', ...})
#   keep_instance! resource, instance
# keeps the just-constructed <%= class_name %> model in the @<%= file_name %>
instance variable.
#
def keep_instance! resource, object
  instance_variable_set("@#{resource}", object)
end
#
# What you should see when you get there
#

steps_for(:ra_response) do
  #
  # Destinations.  Ex:
  #   She should be at the new kids page
  #   Tarkin should be at the destroy alderaan page
  #   The visitor should be at the '/lolcats/download' form
  #   The visitor should be redirected to '/hi/mom'
  #
  # It doesn't know anything about actual routes -- it just
  # feeds its output to render_template or redirect_to
  #
  Then "$actor should be at $path" do |_, path|
    response.should render_template(grok_path(path))
  end

  Then "$actor should be redirected to $path" do |_, path|
    response.should redirect_to(grok_path(path))
  end

  Then "the page should look AWESOME" do
    response.should have_tag('head>title')
    response.should have_tag('h1')
    # response.should be_valid_xhtml
  end

  #
  # Tags
  #

  Then "the page should contain '$text'" do |_, text|
    response.should have_text(/#{text}/)
```

```
  end

  # please note: this enforces the use of a <label> field
  Then "$actor should see a <$container> containing a $attributes" do |_,
container, attributes|
    attributes = attributes.to_hash_from_story
    response.should have_tag(container) do
      attributes.each do |tag, label|
        case tag
        when "textfield" then with_tag "input[type='text']";
with_tag("label", label)
        when "password"  then with_tag "input[type='password']";
with_tag("label", label)
        when "submit"    then with_tag "input[type='submit'][value='#{label}']"
        else with_tag tag, label
        end
      end
    end
  end

  #
  # Session, cookie variables
  #
  Then "$actor $token cookie should include $attrlist" do |_, token, attrlist|
    attrlist = attrlist.to_array_from_story
    cookies.include?(token).should be_true
    attrlist.each do |val|
      cookies[token].include?(val).should be_true
    end
  end

  Then "$actor $token cookie should exist but not include $attrlist" do |_,
token, attrlist|
    attrlist = attrlist.to_array_from_story
    cookies.include?(token).should be_true
    puts [cookies, attrlist, token].to_yaml
    attrlist.each do |val|
      cookies[token].include?(val).should_not be_true
    end
  end

  Then "$actor should have $an $token cookie" do |_, _, token|
    cookies[token].should_not be_blank
  end
  Then "$actor should not have $an $token cookie" do |_, _, token|
    cookies[token].should be_blank
  end

  Given "$actor has $an cookie jar with $attributes" do |_, _, attributes|
    attributes = attributes.to_hash_from_story
    attributes.each do |attr, val|
      cookies[attr] = val
    end
  end
  Given "$actor session store has no $attrlist" do |_, attrlist|
```

```
    attrlist = attrlist.to_array_from_story
    attrlist.each do |attr|
      # Note that the comparison passes through 'to_s'
      session[attr.to_sym] = nil
    end
  end

  Then "$actor session store should have $attributes" do |_, attributes|
    attributes = attributes.to_hash_from_story
    attributes.each do |attr, val|
      # Note that the comparison passes through 'to_s'
      session[attr.to_sym].to_s.should eql(val)
    end
  end

  Then "$actor session store should not have $attrlist" do |_, attrlist|
    attrlist = attrlist.to_array_from_story
    attrlist.each do |attr|
      session[attr.to_sym].blank?.should be_true
    end
  end

  #
  # Flash messages
  #

  Then "$actor should see $an $notice message '$message'" do |_, _, notice,
message|
    response.should have_flash(notice, %r{#{message}})
  end

  Then "$actor should not see $an $notice message '$message'" do |_, _, notice,
message|
    response.should_not have_flash(notice, %r{#{message}})
  end

  Then "$actor should see no messages" do |_|
    ['error', 'warning', 'notice'].each do |notice|
      response.should_not have_flash(notice)
    end
  end

  RE_POLITENESS = /(?:please|sorry|thank(?:s| you))/i
  Then %r{we should be polite about it} do
    response.should have_tag("div.error,div.notice", RE_POLITENESS)
  end
  Then %r{we should not even be polite about it} do
    response.should_not have_tag("div.error,div.notice", RE_POLITENESS)
  end

  #
  # Resource's attributes
  #
  # "Then page should have the $resource's $attributes" is in resource_steps
```

```
  # helpful debug step
  Then "we dump the response" do
    dump_response
  end
end


def have_flash notice, *args
  have_tag("div.#{notice}", *args)
end

RE_PRETTY_RESOURCE = /the (index|show|new|create|edit|update|destroy) (\w+)
(page|form)/i
RE_THE_FOO_PAGE    = /the '?([^']*)'? (page|form)/i
RE_QUOTED_PATH     = /^'([^']*)'$/i
def grok_path path
  path.gsub(/\s+again$/,'') # strip trailing ' again'
  case
  when path == 'the home page'    then dest = '/'
  when path =~ RE_PRETTY_RESOURCE then dest = template_for $1, $2
  when path =~ RE_THE_FOO_PAGE    then dest = $1
  when path =~ RE_QUOTED_PATH     then dest = $1
  else                                 dest = path
  end
  dest
end

# turns 'new', 'road bikes' into 'road_bikes/new'
# note that it's "action resource"
def template_for(action, resource)
  "#{resource.gsub(" ","_")}/#{action}"
end

require File.dirname(__FILE__) + '/../helper'

RE_<%= file_name.capitalize %>      = %r{(?:(?:the )? *(\w+) *)}
RE_<%= file_name.capitalize %>_TYPE = %r{(?: *(\w+)? *)}
steps_for(:<%= file_name %>) do

  #
  # Setting
  #

  Given "an anonymous <%= file_name %>" do
    log_out!
  end

  Given "$an $<%= file_name %>_type <%= file_name %> with $attributes" do |_,
<%= file_name %>_type, attributes|
    create_<%= file_name %>! <%= file_name %>_type,
attributes.to_hash_from_story
  end

  Given "$an $<%= file_name %>_type <%= file_name %> named '$login'" do |_, <%=
file_name %>_type, login|
```

```
    create_<%= file_name %>! <%= file_name %>_type, named_<%= file_name
%>(login)
  end

  Given "$an $<%= file_name %>_type <%= file_name %> logged in as '$login'" do
|_, <%= file_name %>_type, login|
    create_<%= file_name %>! <%= file_name %>_type, named_<%= file_name
%>(login)
    log_in_<%= file_name %>!
  end

  Given "$actor is logged in" do |_, login|
    log_in_<%= file_name %>! @<%= file_name %>_params || named_<%= file_name
%>(login)
  end

  Given "there is no $<%= file_name %>_type <%= file_name %> named '$login'" do
|_, login|
    @<%= file_name %> = <%= class_name %>.find_by_login(login)
    @<%= file_name %>.destroy! if @<%= file_name %>
    @<%= file_name %>.should be_nil
  end

  #
  # Actions
  #
  When "$actor logs out" do
    log_out
  end

  When "$actor registers an account as the preloaded '$login'" do |_, login|
    <%= file_name %> = named_<%= file_name %>(login)
    <%= file_name %>['password_confirmation'] = <%= file_name %>['password']
    create_<%= file_name %> <%= file_name %>
  end

  When "$actor registers an account with $attributes" do |_, attributes|
    create_<%= file_name %> attributes.to_hash_from_story
  end
<% if options[:include_activation] %>
  When "$actor activates with activation code $attributes" do |_,
activation_code|
    activation_code = '' if activation_code == 'that is blank'
    activate
  end<% end %>

  When "$actor logs in with $attributes" do |_, attributes|
    log_in_<%= file_name %> attributes.to_hash_from_story
  end

  #
  # Result
  #
  Then "$actor should be invited to sign in" do |_|
    response.should render_template('/<%= controller_file_path %>/new')
```

```
  end

  Then "$actor should not be logged in" do |_|
    controller.logged_in?.should_not be_true
  end

  Then "$login should be logged in" do |login|
    controller.logged_in?.should be_true
    controller.current_<%= file_name %>.should === @<%= file_name %>
    controller.current_<%= file_name %>.login.should == login
  end

end

def named_<%= file_name %> login
  <%= file_name %>_params = {
    'admin'   => {'id' => 1, 'login' => 'addie', 'password' => '1234addie',
'email' => 'admin@example.com',       },
    'oona'    => {          'login' => 'oona',   'password' => '1234oona',
'email' => 'unactivated@example.com'},
    'reggie'  => {          'login' => 'reggie', 'password' => 'monkey',
'email' => 'registered@example.com' },
    }
  <%= file_name %>_params[login.downcase]
end

#
# <%= class_name %> account actions.
#
# The ! methods are 'just get the job done'.  It's true, they do some testing of
# their own -- thus un-DRY'ing tests that do and should live in the <%=
file_name %> account
# stories -- but the repetition is ultimately important so that a faulty test
setup
# fails early.
#

def log_out
  get '/<%= controller_file_path %>/destroy'
end

def log_out!
  log_out
  response.should redirect_to('/')
  follow_redirect!
end

def create_<%= file_name %>(<%= file_name %>_params={})
  @<%= file_name %>_params       ||= <%= file_name %>_params
  post "/<%= model_controller_file_path %>", :<%= file_name %> => <%= file_name
%>_params
  @<%= file_name %> = <%= class_name %>.find_by_login(<%= file_name
%>_params['login'])
end
```

```
def create_<%= file_name %>!(<%= file_name %>_type, <%= file_name %>_params)
  <%= file_name %>_params['password_confirmation'] ||= <%= file_name
%>_params['password'] ||= <%= file_name %>_params['password']
  create_<%= file_name %> <%= file_name %>_params
  response.should redirect_to('/')
  follow_redirect!
<% if options[:include_activation] %>
  # fix the <%= file_name %>'s activation status
  activate_<%= file_name %>! if <%= file_name %>_type == 'activated'<% end %>
end

<% if options[:include_activation] %>
def activate_<%= file_name %> activation_code=nil
  activation_code = @<%= file_name %>.activation_code if activation_code.nil?
  get "/activate/#{activation_code}"
end

def activate_<%= file_name %>! *args
  activate_<%= file_name %> *args
  response.should redirect_to('/login')
  follow_redirect!
  response.should have_flash("notice", /Signup complete!/)
end<% end %>

def log_in_<%= file_name %> <%= file_name %>_params=nil
  @<%= file_name %>_params ||= <%= file_name %>_params
  <%= file_name %>_params  ||= @<%= file_name %>_params
  post "/<%= controller_routing_path %>", <%= file_name %>_params
  @<%= file_name %> = <%= class_name %>.find_by_login(<%= file_name
%>_params['login'])
  controller.current_<%= file_name %>
end

def log_in_<%= file_name %>! *args
  log_in_<%= file_name %> *args
  response.should redirect_to('/')
  follow_redirect!
  response.should have_flash("notice", /Logged in successfully/)
end
require File.dirname(__FILE__) + '/../test_helper'
require '<%= controller_file_name %>_controller'

# Re-raise errors caught by the controller.
class <%= controller_class_name %>Controller; def rescue_action(e) raise e end;
end

class <%= controller_class_name %>ControllerTest < ActionController::TestCase
  # Be sure to include AuthenticatedTestHelper in test/test_helper.rb instead
  # Then, you can remove it from this and the units test.
  include AuthenticatedTestHelper

  fixtures :<%= table_name %>

  def test_should_login_and_redirect
    post :create, :login => 'quentin', :password => 'monkey'
```

```
    assert session[:<%= file_name %>_id]
    assert_response :redirect
  end

  def test_should_fail_login_and_not_redirect
    post :create, :login => 'quentin', :password => 'bad password'
    assert_nil session[:<%= file_name %>_id]
    assert_response :success
  end

  def test_should_logout
    login_as :quentin
    get :destroy
    assert_nil session[:<%= file_name %>_id]
    assert_response :redirect
  end

  def test_should_remember_me
    @request.cookies["auth_token"] = nil
    post :create, :login => 'quentin', :password => 'monkey', :remember_me =>
"1"
    assert_not_nil @response.cookies["auth_token"]
  end

  def test_should_not_remember_me
    @request.cookies["auth_token"] = nil
    post :create, :login => 'quentin', :password => 'monkey', :remember_me =>
"0"
    puts @response.cookies["auth_token"]
    assert @response.cookies["auth_token"].blank?
  end

  def test_should_delete_token_on_logout
    login_as :quentin
    get :destroy
    assert @response.cookies["auth_token"].blank?
  end

  def test_should_login_with_cookie
    <%= table_name %>(:quentin).remember_me
    @request.cookies["auth_token"] = cookie_for(:quentin)
    get :new
    assert @controller.send(:logged_in?)
  end

  def test_should_fail_expired_cookie_login
    <%= table_name %>(:quentin).remember_me
    <%= table_name %>(:quentin).update_attribute :remember_token_expires_at,
5.minutes.ago
    @request.cookies["auth_token"] = cookie_for(:quentin)
    get :new
    assert !@controller.send(:logged_in?)
  end

  def test_should_fail_cookie_login
```

```
    <%= table_name %>(:quentin).remember_me
    @request.cookies["auth_token"] = auth_token('invalid_auth_token')
    get :new
    assert !@controller.send(:logged_in?)
  end

  protected
    def auth_token(token)
      CGI::Cookie.new('name' => 'auth_token', 'value' => token)
    end

    def cookie_for(<%= file_name %>)
      auth_token <%= table_name %>(<%= file_name %>).remember_token
    end
end
require File.dirname(__FILE__) + '/../test_helper'
require '<%= file_name %>_mailer'

class <%= class_name %>MailerTest < Test::Unit::TestCase
  FIXTURES_PATH = File.dirname(__FILE__) + '/../fixtures'
  CHARSET = "utf-8"

  include ActionMailer::Quoting

  def setup
    ActionMailer::Base.delivery_method = :test
    ActionMailer::Base.perform_deliveries = true
    ActionMailer::Base.deliveries = []

    @expected = TMail::Mail.new
    @expected.set_content_type "text", "plain", { "charset" => CHARSET }
  end

  def test_dummy_test
    #do nothing
  end

  private
    def read_fixture(action)
      IO.readlines("#{FIXTURES_PATH}/<%= file_name %>_mailer/#{action}")
    end

    def encode(subject)
      quoted_printable(subject, CHARSET)
    end
end
require File.dirname(__FILE__) + '/../test_helper'
require '<%= model_controller_file_name %>_controller'

# Re-raise errors caught by the controller.
class <%= model_controller_class_name %>Controller; def rescue_action(e) raise e
end; end

class <%= model_controller_class_name %>ControllerTest <
ActionController::TestCase
```

```
# Be sure to include AuthenticatedTestHelper in test/test_helper.rb instead
# Then, you can remove it from this and the units test.
include AuthenticatedTestHelper

fixtures :<%= table_name %>

def test_should_allow_signup
  assert_difference '<%= class_name %>.count' do
    create_<%= file_name %>
    assert_response :redirect
  end
end

def test_should_require_login_on_signup
  assert_no_difference '<%= class_name %>.count' do
    create_<%= file_name %>(:login => nil)
    assert assigns(:<%= file_name %>).errors.on(:login)
    assert_response :success
  end
end

def test_should_require_password_on_signup
  assert_no_difference '<%= class_name %>.count' do
    create_<%= file_name %>(:password => nil)
    assert assigns(:<%= file_name %>).errors.on(:password)
    assert_response :success
  end
end

def test_should_require_password_confirmation_on_signup
  assert_no_difference '<%= class_name %>.count' do
    create_<%= file_name %>(:password_confirmation => nil)
    assert assigns(:<%= file_name %>).errors.on(:password_confirmation)
    assert_response :success
  end
end

def test_should_require_email_on_signup
  assert_no_difference '<%= class_name %>.count' do
    create_<%= file_name %>(:email => nil)
    assert assigns(:<%= file_name %>).errors.on(:email)
    assert_response :success
  end
end
<% if options[:stateful] %>
def test_should_sign_up_user_in_pending_state
  create_<%= file_name %>
  assigns(:<%= file_name %>).reload
  assert assigns(:<%= file_name %>).pending?
end<% end %>

<% if options[:include_activation] %>
def test_should_sign_up_user_with_activation_code
  create_<%= file_name %>
  assigns(:<%= file_name %>).reload
```

```
    assert_not_nil assigns(:<%= file_name %>).activation_code
  end

  def test_should_activate_user
    assert_nil <%= class_name %>.authenticate('aaron', 'test')
    get :activate, :activation_code => <%= table_name %>(:aaron).activation_code
    assert_redirected_to '/<%= controller_routing_path %>/new'
    assert_not_nil flash[:notice]
    assert_equal <%= table_name %>(:aaron), <%= class_name
%>.authenticate('aaron', 'monkey')
  end

  def test_should_not_activate_user_without_key
    get :activate
    assert_nil flash[:notice]
  rescue ActionController::RoutingError
    # in the event your routes deny this, we'll just bow out gracefully.
  end

  def test_should_not_activate_user_with_blank_key
    get :activate, :activation_code => ''
    assert_nil flash[:notice]
  rescue ActionController::RoutingError
    # well played, sir
  end<% end %>

  protected
    def create_<%= file_name %>(options = {})
      post :create, :<%= file_name %> => { :login => 'quire', :email =>
'quire@example.com',
        :password => 'quire69', :password_confirmation => 'quire69'
}.merge(options)
    end
end
require File.dirname(__FILE__) + '/../test_helper'

class <%= class_name %>Test < ActiveSupport::TestCase
  # Be sure to include AuthenticatedTestHelper in test/test_helper.rb instead.
  # Then, you can remove it from this and the functional test.
  include AuthenticatedTestHelper
  fixtures :<%= table_name %>

  def test_should_create_<%= file_name %>
    assert_difference '<%= class_name %>.count' do
      <%= file_name %> = create_<%= file_name %>
      assert !<%= file_name %>.new_record?, "#{<%= file_name
%>.errors.full_messages.to_sentence}"
    end
  end
<% if options[:include_activation] %>
  def test_should_initialize_activation_code_upon_creation
    <%= file_name %> = create_<%= file_name %>
    <%= file_name %>.reload
    assert_not_nil <%= file_name %>.activation_code
  end
```

```
<% end %><% if options[:stateful] %>
  def test_should_create_and_start_in_pending_state
    <%= file_name %> = create_<%= file_name %>
    <%= file_name %>.reload
    assert <%= file_name %>.pending?
  end

<% end %>
  def test_should_require_login
    assert_no_difference '<%= class_name %>.count' do
      u = create_<%= file_name %>(:login => nil)
      assert u.errors.on(:login)
    end
  end

  def test_should_require_password
    assert_no_difference '<%= class_name %>.count' do
      u = create_<%= file_name %>(:password => nil)
      assert u.errors.on(:password)
    end
  end

  def test_should_require_password_confirmation
    assert_no_difference '<%= class_name %>.count' do
      u = create_<%= file_name %>(:password_confirmation => nil)
      assert u.errors.on(:password_confirmation)
    end
  end

  def test_should_require_email
    assert_no_difference '<%= class_name %>.count' do
      u = create_<%= file_name %>(:email => nil)
      assert u.errors.on(:email)
    end
  end

  def test_should_reset_password
    <%= table_name %>(:quentin).update_attributes(:password => 'new password',
:password_confirmation => 'new password')
    assert_equal <%= table_name %>(:quentin), <%= class_name
%>.authenticate('quentin', 'new password')
  end

  def test_should_not_rehash_password
    <%= table_name %>(:quentin).update_attributes(:login => 'quentin2')
    assert_equal <%= table_name %>(:quentin), <%= class_name
%>.authenticate('quentin2', 'monkey')
  end

  def test_should_authenticate_<%= file_name %>
    assert_equal <%= table_name %>(:quentin), <%= class_name
%>.authenticate('quentin', 'monkey')
  end

  def test_should_set_remember_token
```

```
    <%= table_name %>(:quentin).remember_me
    assert_not_nil <%= table_name %>(:quentin).remember_token
    assert_not_nil <%= table_name %>(:quentin).remember_token_expires_at
  end

  def test_should_unset_remember_token
    <%= table_name %>(:quentin).remember_me
    assert_not_nil <%= table_name %>(:quentin).remember_token
    <%= table_name %>(:quentin).forget_me
    assert_nil <%= table_name %>(:quentin).remember_token
  end

  def test_should_remember_me_for_one_week
    before = 1.week.from_now.utc
    <%= table_name %>(:quentin).remember_me_for 1.week
    after = 1.week.from_now.utc
    assert_not_nil <%= table_name %>(:quentin).remember_token
    assert_not_nil <%= table_name %>(:quentin).remember_token_expires_at
    assert <%= table_name
%>(:quentin).remember_token_expires_at.between?(before, after)
  end

  def test_should_remember_me_until_one_week
    time = 1.week.from_now.utc
    <%= table_name %>(:quentin).remember_me_until time
    assert_not_nil <%= table_name %>(:quentin).remember_token
    assert_not_nil <%= table_name %>(:quentin).remember_token_expires_at
    assert_equal <%= table_name %>(:quentin).remember_token_expires_at, time
  end

  def test_should_remember_me_default_two_weeks
    before = 2.weeks.from_now.utc
    <%= table_name %>(:quentin).remember_me
    after = 2.weeks.from_now.utc
    assert_not_nil <%= table_name %>(:quentin).remember_token
    assert_not_nil <%= table_name %>(:quentin).remember_token_expires_at
    assert <%= table_name
%>(:quentin).remember_token_expires_at.between?(before, after)
  end
<% if options[:stateful] %>
  def test_should_register_passive_<%= file_name %>
    <%= file_name %> = create_<%= file_name %>(:password => nil,
:password_confirmation => nil)
    assert <%= file_name %>.passive?
    <%= file_name %>.update_attributes(:password => 'new password',
:password_confirmation => 'new password')
    <%= file_name %>.register!
    assert <%= file_name %>.pending?
  end

  def test_should_suspend_<%= file_name %>
    <%= table_name %>(:quentin).suspend!
    assert <%= table_name %>(:quentin).suspended?
  end
```

```
  def test_suspended_<%= file_name %>_should_not_authenticate
    <%= table_name %>(:quentin).suspend!
    assert_not_equal <%= table_name %>(:quentin), <%= class_name
%>.authenticate('quentin', 'test')
  end

  def test_should_unsuspend_<%= file_name %>_to_active_state
    <%= table_name %>(:quentin).suspend!
    assert <%= table_name %>(:quentin).suspended?
    <%= table_name %>(:quentin).unsuspend!
    assert <%= table_name %>(:quentin).active?
  end

  def test_should_unsuspend_<%= file_name
%>_with_nil_activation_code_and_activated_at_to_passive_state
    <%= table_name %>(:quentin).suspend!
    <%= class_name %>.update_all :activation_code => nil, :activated_at => nil
    assert <%= table_name %>(:quentin).suspended?
    <%= table_name %>(:quentin).reload.unsuspend!
    assert <%= table_name %>(:quentin).passive?
  end

  def test_should_unsuspend_<%= file_name
%>_with_activation_code_and_nil_activated_at_to_pending_state
    <%= table_name %>(:quentin).suspend!
    <%= class_name %>.update_all :activation_code => 'foo-bar', :activated_at =>
nil
    assert <%= table_name %>(:quentin).suspended?
    <%= table_name %>(:quentin).reload.unsuspend!
    assert <%= table_name %>(:quentin).pending?
  end

  def test_should_delete_<%= file_name %>
    assert_nil <%= table_name %>(:quentin).deleted_at
    <%= table_name %>(:quentin).delete!
    assert_not_nil <%= table_name %>(:quentin).deleted_at
    assert <%= table_name %>(:quentin).deleted?
  end
<% end %>
protected
  def create_<%= file_name %>(options = {})
    record = <%= class_name %>.new({ :login => 'quire', :email =>
'quire@example.com', :password => 'quire69', :password_confirmation => 'quire69'
}.merge(options))
    record.<% if options[:stateful] %>register! if record.valid?<% else %>save<%
end %>
    record
  end
end
require 'aasm'
require File.dirname(__FILE__) + '/lib/authentication'
require File.dirname(__FILE__) + '/lib/authentication/by_password'
require File.dirname(__FILE__) + '/lib/authentication/by_cookie_token'
puts IO.read(File.join(File.dirname(__FILE__), 'README'))# -*- coding: mule-utf-
8 -*-
```

```
module Authentication
  module ByCookieToken
    # Stuff directives into including module
    def self.included(recipient)
      recipient.extend(ModelClassMethods)
      recipient.class_eval do
        include ModelInstanceMethods
      end
    end

    #
    # Class Methods
    #
    module ModelClassMethods
    end # class methods

    #
    # Instance Methods
    #
    module ModelInstanceMethods
      def remember_token?
        (!remember_token.blank?) &&
          remember_token_expires_at && (Time.now.utc <
remember_token_expires_at.utc)
      end

      # These create and unset the fields required for remembering users between
browser closes
      def remember_me
        remember_me_for 2.weeks
      end

      def remember_me_for(time)
        remember_me_until time.from_now.utc
      end

      def remember_me_until(time)
        self.remember_token_expires_at = time
        self.remember_token            = self.class.make_token
        save(false)
      end

      # refresh token (keeping same expires_at) if it exists
      def refresh_token
        if remember_token?
          self.remember_token = self.class.make_token
          save(false)
        end
      end

      #
      # Deletes the server-side record of the authentication token.  The
      # client-side (browser cookie) and server-side (this remember_token) must
      # always be deleted together.
      #
```

```
      def forget_me
        self.remember_token_expires_at = nil
        self.remember_token            = nil
        save(false)
      end
    end # instance methods
  end

  module ByCookieTokenController
    # Stuff directives into including module
    def self.included( recipient )
      recipient.extend( ControllerClassMethods )
      recipient.class_eval do
        include ControllerInstanceMethods
      end
    end

    #
    # Class Methods
    #
    module ControllerClassMethods
    end # class methods

    module ControllerInstanceMethods
    end # instance methods
  end
end

module Authentication
  module ByPassword
    # Stuff directives into including module
    def self.included(recipient)
      recipient.extend(ModelClassMethods)
      recipient.class_eval do
        include ModelInstanceMethods

        # Virtual attribute for the unencrypted password
        attr_accessor :password
        validates_presence_of     :password,                   :if =>
:password_required?
        validates_presence_of     :password_confirmation,      :if =>
:password_required?
        validates_confirmation_of :password,                   :if =>
:password_required?
        validates_length_of       :password, :within => 6..40, :if =>
:password_required?
        before_save :encrypt_password
      end
    end # #included directives

    #
    # Class Methods
    #
    module ModelClassMethods
      # This provides a modest increased defense against a dictionary attack if
```

```
      # your db were ever compromised, but will invalidate existing passwords.
      # See the README and the file config/initializers/site_keys.rb
      #
      # It may not be obvious, but if you set REST_AUTH_SITE_KEY to nil and
      # REST_AUTH_DIGEST_STRETCHES to 1 you'll have backwards compatibility with
      # older versions of restful-authentication.
      def password_digest(password, salt)
        digest = REST_AUTH_SITE_KEY
        REST_AUTH_DIGEST_STRETCHES.times do
          digest = secure_digest(digest, salt, password, REST_AUTH_SITE_KEY)
        end
        digest
      end
    end # class methods

    #
    # Instance Methods
    #
    module ModelInstanceMethods

      # Encrypts the password with the user salt
      def encrypt(password)
        self.class.password_digest(password, salt)
      end

      def authenticated?(password)
        crypted_password == encrypt(password)
      end

      # before filter
      def encrypt_password
        return if password.blank?
        self.salt = self.class.make_token if new_record?
        self.crypted_password = encrypt(password)
      end
      def password_required?
        crypted_password.blank? || !password.blank?
      end
    end # instance methods
  end
end
module Authentication
  mattr_accessor :login_regex, :bad_login_message,
    :name_regex, :bad_name_message,
    :email_name_regex, :domain_head_regex, :domain_tld_regex, :email_regex,
:bad_email_message

  self.login_regex       = /\A\w[\w\.\-_@]+\z/                     # ASCII,
strict
  # self.login_regex       = /\A[[:alnum:]][[:alnum:]\.\-_@]+\z/     # Unicode,
strict
  # self.login_regex       = /\A[^[:cntrl:]\\<>\/&]*\z/              # Unicode,
permissive

  self.bad_login_message = "use only letters, numbers, and .-_@ please.".freeze
```

```
  self.name_regex        = /\A[^[:cntrl:]\\<>\/&]*\z/              # Unicode,
permissive
  self.bad_name_message  = "avoid non-printing characters and \\&gt;&lt;&amp;/
please.".freeze

  self.email_name_regex  = '[\w\.%\+\-]+'.freeze
  self.domain_head_regex = '(?:[A-Z0-9\-]+\.)+'.freeze
  self.domain_tld_regex  = '(?:[A-
Z]{2}|com|org|net|edu|gov|mil|biz|info|mobi|name|aero|jobs|museum)'.freeze
  self.email_regex       =
/\A#{email_name_regex}@#{domain_head_regex}#{domain_tld_regex}\z/i
  self.bad_email_message = "should look like an email address.".freeze

  def self.included(recipient)
    recipient.extend(ModelClassMethods)
    recipient.class_eval do
      include ModelInstanceMethods
    end
  end

  module ModelClassMethods
    def secure_digest(*args)
      Digest::SHA1.hexdigest(args.flatten.join('--'))
    end

    def make_token
      secure_digest(Time.now, (1..10).map{ rand.to_s })
    end
  end # class methods

  module ModelInstanceMethods
  end # instance methods
end
module Authorization
  module AasmRoles
    unless Object.constants.include? "STATEFUL_ROLES_CONSTANTS_DEFINED"
      STATEFUL_ROLES_CONSTANTS_DEFINED = true # sorry for the C idiom
    end

    def self.included( recipient )
      recipient.extend( StatefulRolesClassMethods )
      recipient.class_eval do
        include StatefulRolesInstanceMethods
        include AASM
        aasm_column :state
        aasm_initial_state :initial => :pending
        aasm_state :passive
        aasm_state :pending, :enter => :make_activation_code
        aasm_state :active,  :enter => :do_activate
        aasm_state :suspended
        aasm_state :deleted, :enter => :do_delete
        aasm_state :admin

        aasm_event :register do
```

```
          transitions :from => :passive, :to => :pending, :guard => Proc.new
{|u| !(u.crypted_password.blank? && u.password.blank?) }
        end

        aasm_event :activate do
          transitions :from => :pending, :to => :active
        end

        aasm_event :suspend do
          transitions :from => [:passive, :pending, :active], :to => :suspended
        end

        aasm_event :delete do
          transitions :from => [:passive, :pending, :active, :suspended], :to =>
:deleted
        end

        aasm_event :unsuspend do
          transitions :from => :suspended, :to => :active,  :guard => Proc.new
{|u| !u.activated_at.blank? }
          transitions :from => :suspended, :to => :pending, :guard => Proc.new
{|u| !u.activation_code.blank? }
          transitions :from => :suspended, :to => :passive
        end
      end
    end

    module StatefulRolesClassMethods
    end # class methods

    module StatefulRolesInstanceMethods
      # Returns true if the user has just been activated.
      def recently_activated?
        @activated
      end
      def do_delete
        self.deleted_at = Time.now.utc
      end

      def do_activate
        @activated = true
        self.activated_at = Time.now.utc
        self.deleted_at = self.activation_code = nil
      end
    end # instance methods
  end
end
module Authorization
  module StatefulRoles
    unless Object.constants.include? "STATEFUL_ROLES_CONSTANTS_DEFINED"
      STATEFUL_ROLES_CONSTANTS_DEFINED = true # sorry for the C idiom
    end

    def self.included( recipient )
      recipient.extend( StatefulRolesClassMethods )
```

```
      recipient.class_eval do
        include StatefulRolesInstanceMethods

        acts_as_state_machine :initial => :pending
        state :passive
        state :pending, :enter => :make_activation_code
        state :active,  :enter => :do_activate
        state :suspended
        state :deleted, :enter => :do_delete

        event :register do
          transitions :from => :passive, :to => :pending, :guard => Proc.new
{|u| !(u.crypted_password.blank? && u.password.blank?) }
        end

        event :activate do
          transitions :from => :pending, :to => :active
        end

        event :suspend do
          transitions :from => [:passive, :pending, :active], :to => :suspended
        end

        event :delete do
          transitions :from => [:passive, :pending, :active, :suspended], :to =>
:deleted
        end

        event :unsuspend do
          transitions :from => :suspended, :to => :active,  :guard => Proc.new
{|u| !u.activated_at.blank? }
          transitions :from => :suspended, :to => :pending, :guard => Proc.new
{|u| !u.activation_code.blank? }
          transitions :from => :suspended, :to => :passive
        end
      end
    end

    module StatefulRolesClassMethods
    end # class methods

    module StatefulRolesInstanceMethods
      # Returns true if the user has just been activated.
      def recently_activated?
        @activated
      end
      def do_delete
        self.deleted_at = Time.now.utc
      end

      def do_activate
        @activated = true
        self.activated_at = Time.now.utc
        self.deleted_at = self.activation_code = nil
      end
```

```
    end # instance methods
  end
end
module Authorization
  def self.included(recipient)
    recipient.extend(ModelClassMethods)
    recipient.class_eval do
      include ModelInstanceMethods
    end
  end

  module ModelClassMethods
  end # class methods

  module ModelInstanceMethods
  end # instance methods
end
module Trustification
  module EmailValidation
    unless Object.constants.include? "CONSTANTS_DEFINED"
      CONSTANTS_DEFINED = true # sorry for the C idiom
    end

    def self.included(recipient)
      recipient.extend(ClassMethods)
      recipient.class_eval do
        include InstanceMethods
      end
    end

    module ClassMethods
    end # class methods

    module InstanceMethods
    end # instance methods
  end
end
module Trustification
  def self.included(recipient)
    recipient.extend(ModelClassMethods)
    recipient.class_eval do
      include ModelInstanceMethods
    end
  end

  module ModelClassMethods
  end # class methods

  module ModelInstanceMethods
  end # instance methods
end
%h2 About

%p
  %strong Contact Us
```

```
  %p= link_to('pairwise@photocracy.org', 'mailto:pairwise@photocracy.org')%h2
Pairwise Web Service Developer API Documentation

%h3 Info

%p This is an API to the pairwise web service allowing users to add questions,
items, and voters then from these generate prompts for a question (based on it's
items) and a voter (or for an anonymous voter).  Prompts are generated by a
specific prompt algorithm and votes can be added to a prompt referencing winning
items or indicating a skip.  All entities can be listed.  Items can be listed
according to a ranking algorithm.  Key (string) to value (integer) pairs can be
set per voter.

%p All data is opaque, only relationships will be known to the web service.
Authentication is performed using a HTTP request header with basic authorization
dependent upon the user's assigned login and password.  SSL can be used over
HTTPS.

%h3 Format and Response Codes

%p Arguments can be passed both via REST or through a query string.  Because
some servers do not support PUT POST request are used in place of PUT requests.

%p All methods return an HTTP response code. In event of an error a specific
error code will be returned.

%ul
  %li
    %strong 200
    returned on success
  %li
    %strong 400
    returned on record not found, record invalid, or XML invalid
  %li
    %strong 403
    returned on access to items that you do not have permission to access
  %li
    %strong 417
    returned on error with passed arguments
  %li
    %strong 500
    returned on server error or unexpected result

%p All input POST data must match XML format below.  All output data is returned
as XML.

%h3 Actions

%ul
  %li= link_to('Questions', :anchor => 'questions')
  %li= link_to('Items', :anchor => 'items')
  %li= link_to('Prompts', :anchor => 'prompts')
  %li= link_to('Votes', :anchor => 'votes')
  %li= link_to('Voters', :anchor => 'voters')
  %li= link_to('Prompt Algorithms', :anchor => 'prompt_algorithms')
```

```
  %li= link_to('Ranking Algorithms', :anchor => 'ranking_algorithms')
  %li= link_to('Users', :anchor => 'users')

%ul
  %li#questions
    %h4 Questions
    %h4 Methods
    .methods
      %dl
        %dt url
        %dd questions/add/
        %dt description
        %dd Add a question
        %dt parameters
        %dd none
        %dt request
        %dd
          POST: Array of questions to add
          %p
            &lt;questions&gt;
            %br
                &lt;question&gt;question 1&lt;/question&gt;
            %br
                &lt;question&gt;question 2&lt;/question&gt;
            %br
                &lt;question&gt;question 3&lt;/question&gt;
            %br
                ...
            %br
            &lt;/questions&gt;
        %dt response
        %dd
          Array of added questions
          %p
            &lt;questions&gt;
            %br
                &lt;question id="[system_question_id]"/&gt;
            %br
                &lt;question id="[system_question_id]"/&gt;
            %br
                &lt;question id="[system_question_id]"/&gt;
            %br
                ...
            %br
            &lt;/questions&gt;
    .methods
      %dl
        %dt url
        %dd questions/list
        %dt description
        %dd List all questions
        %dt parameters
        %dd none
        %dt request
        %dd none
```

```
        %dt response
        %dd
          Array of questions
          %p
            &lt;questions items="[items_for_user]"
active_and_inactive_items="[all_items_for_user]" votes="[votes_for_user]"&gt;
            %br
                &lt;question
id="[system_question_id]"&gt;question 1&lt;/question&gt;
            %br
                &lt;question
id="[system_question_id]"&gt;question 2&lt;/question&gt;
            %br
                &lt;question
id="[system_question_id]"&gt;question 3&lt;/question&gt;
            %br
                ...
            %br
            &lt;/questions&gt;
    .methods
      %dl
        %dt url
        %dd questions/[system_question_id]
        %dt description
        %dd The question's unique URL
        %dt parameters
        %dd none
        %dt request
        %dd none
        %dt response
        %dd
          Information about a specific question
          %p
            &lt;question id="[system_question_id]" items="[items_for_question]"
active_and_inactive_items="[all_items_for_user]"
votes="[votes_for_question]"&gt;question1&lt;/question&gt;
    .methods
      %dl
        %dt url
        %dd questions/delete (questions/[id]/delete)
        %dt description
        %dd Delete a question
        %dt parameters
        %dd id -- id of the question to delete
        %dt request
        %dd none
        %dt response
        %dd
          Status of deletion
          %p
            &lt;question
id="[system_question_id]"&gt;[true|false]&lt;/question&gt;
  %li#items
    %h4 Items
    %h4 Methods
```

```
    .methods
      %dl
        %dt url
        %dd items/add
        %dt description
        %dd Adds some number of items to some number of questions
        %dt parameters
        %dd active [optional] -- If present items are automatically
activate.  Defaults to false.
        %dt request
        %dd
          POST: Array of items to add
          %p
            &lt;items&gt;
            %br
                &lt;item&gt;
            %br
                    &lt;data&gt;my
item&lt;/data&gt;
            %br
                    &lt;questions&gt;
            %br

            
&lt;question id="[system_question_id]"/&gt;
            %br

            
&lt;question id="[system_question_id]"/&gt;
            %br
                    &lt;/questions&gt;
            %br
                 &lt;item&gt;
            %br
                 ...
            %br
            &lt;/items&gt;
        %dt response
        %dd
          Array of added items
          %p
            &lt;items&gt;
            %br
                &lt;item id="[system_item_id]"/&gt;
            %br
                &lt;item id="[system_item_id]"/&gt;
            %br
                &lt;item id="[system_item_id]"/&gt;
            %br
                ...
            %br
            &lt;/items&gt;
    .methods
      %dl
        %dt url
```

```
        %dd items/list
(items/list/[question_id]/[rank_algorithm]/[limit]/[offset]/[order])
        %dt description
        %dd List all items uploaded by a user
        %dt parameters
        %dd question_id [optional]
        %dd rank_algorithm [optional] -- name or ID of rank algorithm. Default
order is by created at date.
        %dd limit [optional] -- number of items to return
        %dd offset [optional] -- start from this item
        %dd order [optional] -- asc or desc (default desc), orders by created
date, overrode by position if rank algorithm id is passed<br>
        %dd data [optional] -- If present item is passed back as '...&lt;item
id="[system_item_id]"&gt;[item data]&lt;/item&gt;'
        %dt request
        %dd none
        %dt response
        %dd
          Array of items (if rank_algorithm_id is valid)
          %p
            &lt;items&gt;
            %br
                &lt;item id="[system_item_id]"
added="[item_created_at_text_date]" active="[item_active_value]"
rank="[system_rank_value]" score="[score_given_algo]" ratings="[item_ratings]"
losses="[item_losses]" wins="[item_wins]" skips="[item_skips]"/&gt;
            %br
                &lt;item id="[system_item_id]"
added="[item_created_at_text_date]" active="[item_active_value]"
rank="[system_rank_value]" score="[score_given_algo]" ratings="[item_ratings]"
losses="[item_losses]" wins="[item_wins]" skips="[item_skips]"/&gt;
            %br
                &lt;item id="[system_item_id]"
added="[item_created_at_text_date]" active="[item_active_value]"
rank="[system_rank_value]" score="[score_given_algo]" ratings="[item_ratings]"
losses="[item_losses]" wins="[item_wins]" skips="[item_skips]"/&gt;
            %br
                ...
            %br
            &lt;/items&gt;
    .methods
      %dl
        %dt url
        %dd items/[id]
        %dt description
        %dd The item's unique url
        %dt parameters
        %dd id -- id of the item
        %dt request
        %dd none
        %dt response
        %dd
          Information about an item
          %p
```

```
            &lt;item id="[system_item_id]" active="[item_active_value]"
added="[item_created_at_text_date]"&gt;
            %br
                &lt;data&gt;item data&lt;/data&gt;
            %br
                &lt;questions&gt;
            %br
                    &lt;question
id="[system_question_id]" rank="[system_rank_value]" ratings="[item_ratings]"
losses="[item_losses]" wins="[item_wins]" skips="[item_skips]"/&gt;
            %br
                    &lt;question
id="[system_question_id]" rank="[system_rank_value]" ratings="[item_ratings]"
losses="[item_losses]" wins="[item_wins]" skips="[item_skips]"/&gt;
            %br
                    ...
            %br
                &lt;/question&gt;
            %br
            &lt;/item&gt;
    .methods
      %dl
        %dt url
        %dd items/activate (items/[id]/activate)
        %dt description
        %dd Activate the item
        %dt parameters
        %dd id -- id of the item to activate
        %dt request
        %dd none
        %dt response
        %dd
          Information about an item
    .methods
      %dl
        %dt url
        %dd items/suspend (items/[id]/suspend)
        %dt description
        %dd Suspend the item
        %dt parameters
        %dd id -- id of the item to suspend
        %dt request
        %dd none
        %dt response
        %dd
          Information about an item
    .methods
      %dl
        %dt url
        %dd items/delete (/items/[id]/delete)
        %dt description
        %dd Delete an item
        %dt parameters
        %dd id -- id of the question to delete
        %dt request
```

```
        %dd none
        %dt response
        %dd
          Status of deletion
          %p
            &lt;item id="[system_item_id]"&gt;[true|false]&lt;/item&gt;<br>
  %li#prompts
    %h4 Prompts
    %h4 Methods
    .methods
      %dl
        %dt url
        %dd prompts/list (prompts/list/[question_id]/[item_id])
        %dt description
        %dd The question's unique URL
        %dt parameters
        %dd question_id [optional]
        %dd item_id [optional] -- If given, only prompts including
item[s].  Multiple ids should be comma separated
        %dd data [optional] -- If present item is passed back as '...&lt;item
id="[system_item_id]"&gt;[item data]&lt;/item&gt;'
        %dt request
        %dd none
        %dt response
        %dd
          Array of prompts
          %p
            &lt;prompts&gt;
            %br
                 &lt;prompt id="[system_prompt_id]"&gt;
            %br
                      &lt;items&gt;
            %br

             &n
bsp; &lt;item id="[system_item_id]"/&gt;
            %br

             &n
bsp; ...
            %br
                     
&lt;/items&gt;
            %br
                     
&lt;prompt_algorithm id="[system_prompt_algorithm_id]"&gt;
            %br
                 &lt;/prompt&gt;
            %br
                 ...
            %br
            &lt;/prompts&gt;
    .methods
      %dl
        %dt url
```

```
        %dd prompts/create (prompts/create/[question_id]/[voter_id]/[n]/[prime])
        %dt description
        %dd Create a new prompt
        %dt parameters
        %dd question_id
        %dd voter_id [optional] -- defaults to anonymous voter id of 0
        %dd n [optional] -- number of prompts to generate, defaults to 1
        %dd prime [optional] -- if passed the probability of a prompt being
generated will be proportional to the number of times this prompt has been voted
on in the past (note: you must submit prompt view requests on prompt viewing for
this proportion to be accurate)
        %dd data [optional] -- If present item is passed back as '...&lt;item
id="[system_item_id]"&gt;[item data]&lt;/item&gt;'
        %dt request
        %dd none
        %dt response
        %dd
          A new prompt
          %p
            &lt;prompts&gt;
            %br
                 &lt;algorithm id="[system_alg_id]"/&gt;
            %br
                 &lt;question id="[system_question_id]"/&gt;
            %br
                 &lt;prompt id="[system_prompt_id]"&gt;
            %br
                      &lt;items&gt;
            %br
                     
     &lt;item id="[system_item_id]"/&gt;
            %br
                 
         ...
            %br
                      &lt;/items&gt;
            %br
                 &lt;/prompt&gt;
            %br
                 ...
            %br
            &lt;/prompts&gt;
    .methods
      %dl
        %dt url
        %dd prompts/[id]
        %dt description
        %dd The prompt's unique url
        %dt parameters
        %dd id -- id of the prompt
        %dt request
        %dd none
        %dt response
        %dd
          Information about a prompt
```

```
          %p
            &lt;prompt id="[system_prompt_id]"&gt;
            %br
                &lt;question id="[system_question_id]"/&gt;
            %br
                &lt;items&gt;
            %br
                    &lt;item
id="[system_item_id]"/&gt;
            %br
                    &lt;item
id="[system_item_id]"/&gt;
            %br
                    ...
            %br
                &lt;/items&gt;
            %br
            &lt;/prompt&gt;
    .methods
      %dl
        %dt url
        %dd prompts/view (prompts/view/[id])
        %dt description
        %dd The question's unique URL
        %dt parameters
        %dd id -- the id of the prompt voted on
        %dt request
        %dd none
        %dt response
        %dd
          Head OK or error status.
  %li#votes
    %h4 Votes
    %h4 Methods
    .methods
      %dl
        %dt url
        %dd votes/list (votes/list/[question_id]/[item_id])
        %dt description
        %dd list all votes and the item(s) voted on
        %dt parameters
        %dd item_id [optional] -- restrict to specific item_id
        %dd question_id [optional] -- restrict to specific question
        %dt request
        %dd none
        %dt response
        %dd
          Array of votes
          %p
            &lt;votes&gt;
            %br
                &lt;vote id="[system_vote_id]"
tracking="[system_tracking_value]" response_time="[system_response_time]"&gt;
            %br
```

```
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;prompt
id="[system_prompt_id]"/&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;voter
id="[system_voter_id]"/&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;items&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;
&amp;lt;item id="[system_item_id]"/&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;
...
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;/items&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;/vote&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp; ...
            %br
            &amp;lt;/votes&amp;gt;
    .methods
      %dl
        %dt url
        %dd votes/add
(votes/add/[prompt_id]/[voter_id]/[response_time]/[item_id])
        %dt description
        %dd Add new vote to system
        %dt parameters
        %dd prompt_id
        %dd voter_id [optional] -- Defaults to anonymous voter id of 0
        %dd response_time [optional] -- Note that if using REST you must pass a
response_time if you wish to pass an item_id.
        %dd item_id (the winning item id) -- If given, vote is for included
item[s].&amp;nbsp; Multiple ids should be comma separated.&amp;nbsp; If no item_id is
passed or the item_id is 0 a skip is assumed.&lt;br&gt;
        %dd skip (1 or null) [optional] -- If given and is 1 item_id is ignored
        %dd tracking [optional] -- Additional tracking data to be stored with
the vote&lt;br&gt;
        %dt request
        %dd none
        %dt response
        %dd
          The vote
          %p
            &amp;lt;vote id="[system_vote_id]"&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;prompt id="[system_prompt_id]"/&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;items&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;lt;item
id="[system_item_id]"/&amp;gt;
            %br
            &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; ...
```

```
            %br
                &lt;/items&gt;
            %br
            &lt;/vote&gt;
  %li#voters
    %h4 Voters
    %h4 Methods
    .methods
      %dl
        %dt url
        %dd voters/add
        %dt description
        %dd Delete a question
        %dt parameters
        %dd id -- id of the question to delete
        %dt request
        %dd
          Array of voters to be added
          %p
            &lt;voters&gt;
            %br
                &lt;voter&gt;
            %br
                    &lt;features&gt;
            %br
                       
&lt;feature name="gender"&gt;1&lt;/feature&gt;
            %br
                        ...
            %br
                    &lt;/features&gt;
            %br
                &lt;/voter&gt;
            %br
                ...
            %br
            &lt;/voters&gt;
        %dt response
        %dd
          Array of added voters
          %p
            &lt;voters&gt;
            %br
                &lt;voter id="[system_voter_id]"/&gt;
            %br
                ...
            %br
            &lt;/voters&gt;
    .methods
      %dl
        %dt url
        %dd voters/list
        %dt description
        %dd Lists all voters
        %dt parameters
```

```
        %dd id -- id of the question to delete
        %dt request
        %dd none
        %dt response
        %dd
          List of voters
          %p
            &lt;voters&gt;
            %br
                &lt;voter id="[system_voter_id]"&gt;
            %br
                    &lt;features&gt;
            %br
                        &lt;feature
name="gender"&gt;1&lt;/feature&gt;
            %br
                        ...
            %br
                    &lt;/features&gt;
            %br
                &lt;/voter&gt;
            %br
                ...
            %br
            &lt;/voters&gt;
    .methods
      %dl
        %dt url
        %dd voters/set (voters/set/[id])
        %dt description
        %dd Set the value of a voter's feature
        %dt parameters
        %dd id -- id of the voter to set attributes for
        %dd {} -- [name of feature]=[value of feature]
        %dt request
        %dd none
        %dt response
        %dd
          Voter id
          %p
            &lt;voter id="[system_voter_id]"/&gt;
  %li#prompt_algorithms
    %h4 Prompt Algorithms
    .methods
      %dl
        %dt url
        %dd prompt_algorithms/list
        %dt description
        %dd List all available prompting algorithms
        %dt parameters
        %dd id -- id of the question to delete
        %dt request
        %dd none
        %dt response
        %dd
```

```
          Array of algorithms
          %p
            &lt;prompt_algorithms&gt;
            %br
                &lt;prompt_algorithm
id="[system_prompt_algorithm_id]"&gt;&lt;description&gt;Random&lt;/description&g
t;&lt;/prompt_algorithm&gt;
            %br
                &lt;prompt_algorithm
id="[system_prompt_algorithm_id]"&gt;&lt;description&gt;Simple&lt;/description&g
t;&lt;/prompt_algorithm&gt;
            %br
                &lt;prompt_algorithm
id="[system_prompt_algorithm_id]"&gt;&lt;description&gt;Whateva'&lt;/description
&gt;&lt;/prompt_algorithm&gt;
            %br
            &lt;/prompt_algorithms&gt;
  %li#ranking_algorithms
    %h4 Ranking Algorithms
    %h4 Methods
    .methods
      %dl
        %dt url
        %dd rank_algorithms/list
        %dt description
        %dd List all available ranking algorithms
        %dt parameters
        %dd id -- id of the question to delete
        %dt request
        %dd none
        %dt response
        %dd
          Array of algorithms
          %p
            &lt;rank_algorithms&gt;
            %br
                &lt;rank_algorithm
id="[system_rank_algorithm_id]"&gt;&lt;description&gt;Random&lt;/description&gt;
&lt;/rank_algorithm&gt;
            %br
                &lt;rank_algorithm
id="[system_rank_algorithm_id]"&gt;&lt;description&gt;Simple&lt;/description&gt;
&lt;/rank_algorithm&gt;
            %br
                &lt;rank_algorithm
id="[system_rank_algorithm_id]"&gt;&lt;description&gt;Whateva'&lt;/description&g
t;&lt;/rank_algorithm&gt;
            %br
            &lt;/rank_algorithms&gt;
  %li#users
    %h4 Users
    %h4 Methods
    .methods
      %dl
        %dt url
```

```
        %dd users/add/
        %dt description
        %dd Add a user
        %dt parameters
        %dd key -- user add access key
        %dt request
        %dd
          POST: User to add
          %p
            &lt;user&gt;
            %br
                &lt;login&gt;kurt&lt;/login&gt;
            %br
                &lt;email&gt;kurt@godel.at&lt;/email&gt;
            %br
                &lt;password&gt;encoded_password&lt;/password&gt;
            %br
            &lt;/user&gt;
        %dt response
        %dd
          Added user
          %p
            &lt;user&gt;
            %br
                &lt;login&gt;kurt&lt;/login&gt;
            %br
            &lt;/user&gt;%h2.error
  = @error
  =h @requested_page%error{:page => @requested_page}
  = cdata(@error)%h2 API Debug Interface
%table
  %tr
    %td
      %label url
    %td= text_field_tag 'url', nil, :size => 100
  %tr
    %td
      %label post
    %td= text_area_tag 'post', nil, :size => "60x10"
  %tr
    %td
    %td
      = submit_tag 'send', :onclick => "postTest($F('url'), $F('post'),
'response')"
  %tr
    %td
      %label response
    %td= text_area_tag 'response', nil, :size => "60x10"
%p= link_to('Clear All Data', { :action => 'delete' }, :confirm => 'This will
destroy ALL data associated with your account. It is irreversible. Are you
sure?', :method => :post)
%p= link_to('Clear All Item Data', { :action => 'delete_items' }, :confirm =>
'This will destroy ALL data associated with your account. It is irreversible.
Are you sure?', :method => :post)
%h2 Pairwise Web Service API
```

```
%p The Pairwise Web Service allows anyone to create pairwise comparison voting
systems.  To participate you send us:
%ol
  %li questions to create pairs for (at least one)
  %li items to vote on (at least two, but the more the better)
  %li [optional] voters to track who votes on what
  %li [optional] key value attributes for your voters

%p Then you can send us API requests and we'll send back:
%ol
  %li pairs of items to vote on (prompts)
  %li== rankings of the items based on their votes (by percent wins or
#{link_to('Elo Score', 'http://en.wikipedia.org/wiki/Elo_rating_system')})
  %li the number of votes, wins, losses, and skips for items, questions, and for
all your questions

%p To record a vote send us the item your user clicks on, or skips.  You can
also send us when a user has viewed a prompt and generate statistics on which
prompts are more popular.

%p All data is opaque so we will only know relationships between objects.  A
REST interface is used with POST replacing PUT as some servers do not support
PUT.  Although, if you prefer, you can use query strings.

%p Authentication is performed using an HTTP request header with basic
authorization dependent upon a login and password you'll choose when signing up.
If you wish, SSL can be used by interacting over HTTPS.

%p== For details on using the API refer to the #{link_to('Developer API
Documentation', api_path)}.

%h3 Additional Resources

%p Ruby on Rails Pairwise Plugin
%ul
  %li= link_to('Git Project Page', 'http://github.com/pld/ror-
pairwise/tree/master')
  %li= link_to('RDoc Documentation', 'http://code.helioid.com/ror-pairwise/')
%p Pairwise Website
%ul
  %li= link_to('Git Project Page', 'http://github.com/pld/pairwise/tree/master')
  %li= link_to('RDoc Documentation', 'http://code.helioid.com/pairwise/')%items
  -for item in @items
    %item{:id => item.id}%item{:id => @id}= @success
  %items{:count => @items.length}
  -if @data
    -for i in @items
      -iq = i.items_questions.first
      %item{:id => i.id, :active => active?(i), :rank => iq.position, :score =>
@score.call(i), :ratings => iq.ratings, :wins => iq.wins, :losses => iq.losses,
:skips => iq.ratings - (iq.wins + iq.losses) ,:added =>
i.created_at.to_s(:long)}= cdata(i.data)
  -else
    -for i in @items
```

```
      -iq = i.items_questions.first
      %item{:id => i.id, :active => active?(i), :rank => iq.position, :score =>
@score.call(i), :ratings => iq.ratings, :wins => iq.wins, :losses => iq.losses,
:skips => iq.ratings - (iq.wins + iq.losses), :added =>
i.created_at.to_s(:long)}/
-if @item
  %item{:id => @item.id, :active => active?(@item), :added =>
@item.created_at.to_s(:long)}
    %data= cdata(@item.data.to_s)
    -if @item.questions
      %questions
        -for question in @item.questions
          -iq = ItemsQuestion.first(:conditions => { :item_id => @item.id,
:question_id => question.id })
          %question{:id => question.id, :rank => iq.position, :ratings =>
iq.ratings, :wins => iq.wins, :losses => iq.losses, :skips => iq.ratings -
(iq.wins + iq.losses) }!!! 1.0
%html{:xmlns => "http://www.w3.org/1999/xhtml", :'xml:lang' => "en", :lang =>
"en"}
  %head
    %meta{:'http-equiv' => "content-type", :content => "text/html;charset=UTF-
8"}
    %link{:rel => "shortcut icon", :href => "/favicon.ico"}
    %title Pairwise
    = stylesheet_link_tag 'screen'
    = javascript_include_tag :defaults
  %body
    = render :partial => "shared/header"
    %p.notice= flash[:notice]
    = yield
    = analytics
    = render :partial => "shared/footer"!!! XML
%pairwise{:version=>"0.1.1"}
  = yield=f.error_messages

%p
  = f.label :name
  = f.text_field :name
%p
  = f.label :data
  = f.text_field :data%h1 Editing prompt_algorithm

-form_for(@prompt_algorithm) do |f|
  = render :partial => 'form', :locals => { :f => f }
  %p
    = f.submit "Update"
= link_to 'Show', @prompt_algorithm
|
= link_to 'Back', prompt_algorithms_path
%h1 Listing prompt_algorithms

%table
  %tr
    %th Name
    %th Data
```

```
  -for prompt_algorithm in @prompt_algorithms
    %tr
      %td=h prompt_algorithm.name
      %td=h prompt_algorithm.data
      %td= link_to 'Show', prompt_algorithm
      %td= link_to 'Edit', edit_prompt_algorithm_path(prompt_algorithm)
      %td= link_to 'Destroy', prompt_algorithm, :confirm => 'Are you sure?',
:method => :delete
%br
= link_to 'New prompt_algorithm', new_prompt_algorithm_path
%prompt_algorithms
  -for algorithm in @algorithms
    %prompt_algorithm{:id => algorithm.id}
      %description= cdata(algorithm.name)%h1 New prompt_algorithm

-form_for(@prompt_algorithm) do |f|
  = render :partial => 'form', :locals => { :f => f }
  %p
    = f.submit "Create"
= link_to 'Back', prompt_algorithms_path
%p
  %strong Name:
  =h @prompt_algorithm.name
%p
  %strong Data:
  =h @prompt_algorithm.data
= link_to 'Edit', edit_prompt_algorithm_path(@prompt_algorithm)
|
= link_to 'Back', prompt_algorithms_path%prompts
  %algorithm{:id => @algorithm_id}/
  %question{:id => @question_id}/
  -if @data
    -for prompt_id, prompt_item_ids in @prompt_item_ids
      %prompt{:id => prompt_id}
        %items
          -for id in prompt_item_ids
            %item{:id => id}= cdata(Item.find(id).data)
  -else
    -for prompt_id, prompt_item_ids in @prompt_item_ids
      %prompt{:id => prompt_id}
        %items
          -for id in prompt_item_ids
            %item{:id => id}/%prompts
  -if @data
    -for p in @prompts
      %prompt{:id => p.id}
        %items
          -for i in p.items
            %item{:id => i.id}= cdata(i.data)
      %algorithm{:id => p.prompt_algorithm_id}/
  -else
    -for p in @prompts
      %prompt{:id => p.id}
        %items
```

```
          -for i in p.items
            %item{:id => i.id}/
      %algorithm{:id => p.prompt_algorithm_id}/-if @prompt
  %prompt{:id => @prompt.id, :active => @prompt.active}
    %question{:id => @prompt.question.id}/
    %items
      -for item in @prompt.items
        %item{:id => item.id}/%questions
  -for question in @questions
    %question{:id => question.id}%question{:id => @id}=
@success%questions{:items => @items_count, :active_and_inactive_items =>
@all_items_count, :votes => @votes_count}
  -for question in @questions
    %question{:id => question.id}= cdata(question.name)-if @question
  %question{:id => @question.id, :items => @items_count,
:active_and_inactive_items => @all_items_count, :votes => @votes_count}=
cdata(@question.name)=f.error_messages

%p
  = f.label :name
  = f.text_field :name
%p
  = f.label :data
  = f.text_field :data%h1 Editing rank_algorithm

-form_for(@rank_algorithm) do |f|
  = render :partial => 'form', :locals => { :f => f }
  %p= f.submit "Update"
= link_to 'Show', @rank_algorithm
|
= link_to 'Back', rank_algorithms_path
%h1 Listing rank_algorithms

%table
  %tr
    %th ID
    %th Name
    %th Data

  -for rank_algorithm in @rank_algorithms
    %tr
      %td=h rank_algorithm.id
      %td=h rank_algorithm.name
      %td=h rank_algorithm.data
      %td= link_to 'Show', rank_algorithm
      %td= link_to 'Edit', edit_rank_algorithm_path(rank_algorithm)
      %td= link_to 'Destroy', rank_algorithm, :confirm => 'Are you sure?',
:method => :delete
%br
= link_to 'New rank_algorithm', new_rank_algorithm_path
%br
= link_to 'Build Stats', build_stats_rank_algorithms_path
%rank_algorithms
  -for algorithm in @algorithms
    %rank_algorithm{:id => algorithm.id}
```

```
      %description= cdata(algorithm.name)%h1 New rank_algorithm

-form_for(@rank_algorithm) do |f|
  = render :partial => 'form', :locals => { :f => f }
  %p= f.submit "Create"
= link_to 'Back', rank_algorithms_path
%p
  %strong Name:
  =h @rank_algorithm.name
%p
  %strong Data:
  =h @rank_algorithm.data
= link_to 'Edit', edit_rank_algorithm_path(@rank_algorithm)
|
= link_to 'Back', rank_algorithms_path
%h2 Pairwise Web Service API
%table.info
  %tr
    %td
      %p  With the Pairwise Web Service anyone can build community-generated and
community-sorted websites. These websites let the best ideas in your group or
organization "bubble to the top" all in a democratic, transparent, and bottom-up
process.
      %p
        For example, the
        = link_to('Princeton Undergraduate Student Government',
'http://www.dailyprincetonian.com/2009/01/12/22505/')
        used a site like this to learn about the best ways to improve campus
life. In a matter of weeks, more than 2,000 students contributed more than
40,000 votes on more than 300 suggestions, about 100 of which were uploaded by
the students themselves.
      %p
        In addition to text, community-generated and community-sorted
information websites can be used for to handle images as was done in
        == #{link_to('Photocracy', 'http://www.photocracy.org')},
        a project to study national identity and cross-national perceptions.
    %td.image= image_tag('which_do_you_want_more_schematic_wranking.png', :width
=> 400)

%h3 Learn More about using Pairwise
%p= link_to('Learn More', learn_path)
%p= link_to('API Documentation', api_path)

%h3 Sites already using the Pairwise Webservice
%p= link_to("All Our Ideas", 'http://www.allourideas.org')
%p= link_to("Photocracy", 'http://www.photocracy.org')
%p= link_to("Princeton Art of Science 2009",
'http://www.princeton.edu/artofscience/2009/vote/')

%br
%br

-form_tag session_path do
  %table
    %tr
```

```
      %td= label_tag 'login'
      %td= text_field_tag 'login', @email
    %tr
      %td= label_tag 'password'
      %td= password_field_tag 'password', nil
    %tr
      %td
      %td
        = submit_tag 'Log in'
        = link_to 'Sign up', signup_path%script{:type =>"text/javascript"}
  Google.gaSSDSLoad("UA-7217573-2");%ul.footer.horizontal
  %li= link_to 'home', root_path
  %li= link_to 'learn more', learn_path
  %li= link_to 'api', api_path
  %li= link_to 'about', about_path
%ul.horizontal
  %li
    This research is support by a grant from the
    = link_to('Center for Information Technology Policy',
'http://citp.princeton.edu/')
    at
    = link_to('Princeton University', 'http://www.princeton.edu/')%h1= link_to
'Pairwise', root_path
-if logged_in?
  %p
    logged in as
    = current_user.login
    == (#{link_to "log out", logout_path, { :title => "Log out" }})-require 'pp'
%p
  A
  %strong=h @exception.class
  occured in
  %strong=h @controller.controller_name
  \#
  %strong=h @controller.action_name
  \:
  %pre=h @exception.message
  %pre=h @backtrace.first
%hr
%p Request information:
%hr
%ul
  %li
    URL:
    %pre=h @request.protocol
    %pre=h @host
    %pre=h @request.request_uri
  %li
    Parameters:
    %pre=h @request.parameters.inspect
  %li
    Rails root:
    %pre=h @rails_root
%hr
%p Session dump:
```

```
%hr
%ul
-for variable in @request.session.instance_variables
  -next if variable =~ /^@db/
  %li
    %strong=h variable
    %pre=h
escape_once(@request.session.instance_variable_get(variable).inspect.gsub(/\n/,
"\n   ").strip)
%hr
%p Visit:
%hr
%ul
-for key, value in @visit.attributes
  %li
    %strong=h key
    \:
    %pre=h value
%hr
%p Environment:
%hr
%ul
-for key, value in @request.env
  %li
    %strong=h key
    \:
    %pre=h value.to_s.strip
%hr
%p Full execution backtrace:
%hr
  %pre= @backtrace.join "<br/>  "

==#{h @user.login}, your account has been activated.  Welcome aboard!

=h @url
An account has been created.

Username:
=h @user.login

Name:
=h @user.name

Email:
=h @user.email

Created:
=h @user.created_at

State:
=h @user.stateYour account has been created.

Username:
=h @user.login
```

```
Visit this url to activate your account:
=h @url
%table
  %tr
    %td Name:
    %td= user.login
  %tr
    %td Email:
    %td= user.email
  %tr
    %td State:
    %td= user.state
  -if user.state == "passive"
    %tr
      %td Activation Code:
      %td= user.activation_code
%ul
  %li= link_to 'Suspend', suspend_user_path(user)
  %li= link_to 'Unsuspend', unsuspend_user_path(user)
  %li= link_to 'Delete', users_path, :method => :delete, :id => user-if @user &&
!@user.new_record?
  %user{:id => @user.id}
    %login= @user.login.content
  = render :partial => "user", :collection => @users%h1 Sign up as a new user

= error_messages_for :user
-form_for :user, :url => users_path do |f|
  %table.form
    %tr
      %td= label_tag 'name'
      %td= f.text_field :login
    %tr
      %td= label_tag 'email'
      %td= f.text_field :email
    %tr
      %td= label_tag 'password'
      %td= f.password_field :password
    %tr
      %td= label_tag 'password_confirmation', 'Confirm Password'
      %td= f.password_field :password_confirmation
    %tr
      %td
      %td= submit_tag 'Sign up'
%voters
  -for voter in @voters
    %voter{:id => voter.id}%voters
  -for voter in @voters
    %voter{:id => voter.id}
      %features
        -for feature in voter.features
          %feature{:name => feature.name}= cdata(feature.value)%voter{:id =>
@voter.id}%vote{:id => @vote.id}
  %prompt{:id => @vote.prompt_id}
  %items
    -for item in @vote.items
```

```
      %item{:id => item.id}%votes
  -for vote in @votes
    %vote{:id => vote.id, :tracking => vote.tracking, :response_time =>
vote.response_time}
      %prompt{:id => vote.prompt_id}
      %voter{:id => vote.voter_id}
      %items
        -if @votes_items.nil? || @votes_items.include?(vote)
          -for item in vote.items
            %item{:id => item.id}
```