-*- mode: org -*-

April 27, 2013

Contents

1 DONE Display current goal

1.1 DONE goal_{now} in User model

```
def self.goal_now(user_id)
  u = User.find(user_id)
  elapsed_time = Time.now - u.goal_start_time
  lbs_per_second = ( u.goal_loss_rate / 86400.0 / 3500.0 )
  return ( u.goal_start_weight - lbs_per_second * elapsed_time )
end
```

1.2 DONE @goal_{now} in user controller <users_controller.rb> (C-c C-o)

```
u = User.find(session[:user_id])
@goal_now = User.goal_now(u.id)
```

1.3 DONE View

<%= $\text{number}_{\text{withprecision}}$(@$\text{goal}_{\text{now}}$, :precision => 3)%>

2 TODO [7/7] Weight as a function of time

1. ☒ In Reading model, $\text{weight}_{\text{attime}}$ function

   ```
   def self.weight_at_time(user_id, time)
     time_initial = Reading.time_initial(user_id)
     weight_initial = Reading.weight_initial(user_id)
     if ( time < time_initial )
       return weight_initial
   ```

```
    end
    max_gain_rate = User.filter_rate_gain(user_id)
    max_loss_rate = User.filter_rate_loss(user_id)
    readings = Reading.get_readings_after( user_id, time_initial, time )
    for reading in readings
      w = apply_filter(max_gain_rate, max_loss_rate, time_initial, weight_initial, r
      time_initial = reading.reading_time
      weight_initial = w
    end
    next_reading = Reading.get_next_reading_after(user_id, time)
    if next_reading
      weight = interpolate( max_gain_rate, max_loss_rate, time_initial, weight_initi
                            ng.weight, time )
    else
      weight = apply_filter(max_gain_rate, max_loss_rate, time_initial, weight_initi
    end
    #    return number_with_precision(weight, :precision => 5 )
    return weight
  end
```

2. ☒ In Reading model, $\text{time}_{\text{initial}}$ function

```
  def self.time_initial( user_id )
    return Reading.order('reading_time ASC').where(:user_id => user_id).first.readin
  end
```

3. ☒ In Reading model, $\text{weight}_{\text{initial}}$ function

```
  def self.weight_initial( user_id )
    return Reading.order('reading_time ASC').where(:user_id => user_id).first.weight
  end
```

4. ☒ In User model, functions $\text{filter}_{\text{rategain}}$, $\text{filter}_{\text{rateloss}}$, $\text{goal}_{\text{lossrate}}$

```
  def self.filter_rate_gain(user_id)
    return User.where(:id => user_id).first.filter_rate_gain
  end
  def self.filter_rate_loss(user_id)
    return User.where(:id => user_id).first.filter_rate_loss
```

```
end
def self.goal_loss_rate(user_id)
  cals_per_day = User.where(:id => user_id).first.goal_loss_rate
  lbs_per_second = cals_per_day / 3500.0 / 86400.0
  return  lbs_per_second
end
```

5. ☒ In Reading model, self.get$_{readingsafter}$, self.get$_{nextreadingafter}$(user$_{id}$, time)

```
def self.get_readings_after( user_id, start_time, end_time )
  return Reading.order('reading_time ASC').where(:user_id => user_id).where('readi
end
def self.get_next_reading_after( user_id, time )
    return Reading.order('reading_time ASC').where(:user_id => user_id).where('rea
end
```

6. ☒ In Reading model, apply$_{filter}$(max$_{gainrate}$, max$_{lossrate}$, initial$_{time}$, initial$_{weight}$, time, weight)

```
def self.apply_filter( max_gain_rate, max_loss_rate, initial_time, initial_weight,
  if ( weight == initial_time )
    return weight
  else
    delta_time = ( time - initial_time ).to_i
    cals_day_pounds_second = 1.0 / 86400.0 / 3500.0
    max_allowable_weight = initial_weight + ( max_gain_rate * cals_day_pounds_seco
    min_allowable_weight = initial_weight - ( max_loss_rate * cals_day_pounds_seco
    if ( weight > max_allowable_weight )
      return max_allowable_weight
    end
    if ( weight < min_allowable_weight )
      return min_allowable_weight
    end
  end
  return  weight
end
```

7. ☒ In Reading model, interpolate

```
def self.interpolate( max_gain_rate, max_loss_rate, last_time, last_weight, next_t
  filtered_next_weight = apply_filter(max_gain_rate, max_loss_rate, last_time, las
  delta_time = next_time - last_time
  delta_weight = ( filtered_next_weight - last_weight )
  percent = ( time - last_time ) / delta_time.to_f
  interpolated_weight = last_weight + percent * delta_weight
end
```

3 TODO Display weight now

```
<%= Reading.weight_at_time(2, Time.now) %>
```

4 Add New Reading to Welcome Page

1. ☐ Add a $_{\text{form}}$ partial by linking to reading/$_form$ undefined method model$_{\text{name}}$'
for NilClass:Class Extracted source (around line #1):
1: <%= form$_{\text{for}}$(@reading) do |f| %> 2: <% if @reading.errors.any? %>
3: <div id="error$_{\text{explanation}}$"> 4: <h2><%= pluralize(@reading.errors.count,
"error") %> prohibited this reading from being saved:</h2>

   1. ☐ Add @reading = Reading.new to welcome/index

   2. ☐ Add hidden field (see http://api.rubyonrails.org/classes/ActionView/Helpers/FormHelper.html
)

   3. ☐ Delete <%= f.label :user$_{\text{id}}$ %><br />

   4. ☐ Add @reading.user$_{\text{id}}$ = session[:user$_{\text{id}}$] in create method in readings
   controller

5 Mail

5.1 Chapter 13: Task H: Sending Mail

1. ☐ environment.rb

```
 config.action_mailer.delivery_method = :smtp | :sendmail | :test
```

```
Depot::Application.configure do
  config.action_mailer.delivery_method = :smtp
  config.action_mailer.smtp_settings = {
    address: "smtp.gmail.com",
    port: 587,
    domain: "domain.of.sender.net",
    authentication: "plain",
    user_name: "dave",
    password: "secret",
    enable_starttls_auto: true
  }
end
```

1. ☐ restart server
2. ☐ rails generate mailer GoalReminder goal calculation create app/mailers/goal$_{\text{reminder}}$.rb invoke erb create app/views/goal$_{\text{reminder}}$ create app/views/goal$_{\text{reminder}}$/goal.text.erb create app/views/goal$_{\text{reminder}}$/calculation.text.erb invoke test$_{\text{unit}}$ create test/functional/goal$_{\text{remindertest}}$.rb
3. ☐ Change into app/mailers and edit goal$_{\text{reminder}}$.rb
4. ☐ In console => GoalReminder.goal.deliver
5. ☐ 24.1 A Stand-Alone Application Using Active Record

```
require "config/environment.rb"
order = Order.find(1)
order.name = "Dave Thomas"
order.save
```

5.2 stand alone application

6 Display readings table on Welcome Page

1. ☐ @readings = Readings.all won't work because would get other user's Readings
2. ☐ Controller: @readings = Reading.by$_{\text{user}}$(session[:user$_{\text{id}}$]).order('reading$_{\text{time}}$ DESC')
3. ☐ Model: scope :by$_{\text{user}}$, lambda { |user$_{\text{id}}$| where('user$_{\text{id}}$ = ?', user$_{\text{id}}$) }
4. ☐ See http://asciicasts.com/episodes/215-advanced-queries-in-rails-3
5. ☐ See Agile book, active record

7 Weight loss/gain over the last 28 days

8 Draw a graph

1. ☐ http://nubyonrails.com/pages/gruff
2. ☐ Build and Install RMagick
   (a) ☐ Download http://rubyforge.org/frs/download.php/70067/RMagick-2.13.1.tar.bz2 or from https://github.com/rmagick/rmagick
   (b) ☐ Run "ruby setup.rb"
   (c) ☐ Run "sudo ruby setup.rb install"
3. ☐ sudo gem install gruff
4. ☐ cd into plugins and run gem unpack gruff
5. ☐ rails generate controller WeightGraph week month year
6. ☐ In config/environment.rb add require 'gruff' after the ::Application.initialize! line
7. ☐ See http://www.igvita.com/2007/01/05/dynamic-stat-graphs-in-rails/
8. ☐ See http://api.rubyonrails.org/classes/ActionController/DataStreaming.html
9. ☐ In weight$_{graphcontroller}$.rb:

```
def month
  g = Gruff::Line.new
  # Next line is transient bug fix; see http://stackoverflow.com/questions/10881173/gru
  g.marker_count = 4 #explicitly assign value to @marker_count
  g.title = "My Graph"
  g.data("Apples", [1, 2, 3, 4, 4, 3])
  g.data("Oranges", [4, 8, 7, 9, 8, 9])
  g.data("Watermelon", [2, 3, 1, 5, 6, 8])
  g.data("Peaches", [9, 9, 10, 8, 7, 9])
  g.labels = {0 => '2003', 2 => '2004', 4 => '2012'}
  send_data(g.to_blob, :disposition => 'inline', :type => 'image/png', :filename => "1w
end
```

1. ☐ In View:

```
<img src="<%= url_for :controller => "weight_graph", :action=> "month" %>" style="borde
```

9 Graph last 28 days

```
def month
  g = Gruff::Line.new
  weight = 0
  time_at_point_in_past = 0
  user_id = session[:user_id]
  time_first_reading = Reading.time_initial(user_id)
  weight_first_reading = Reading.weight_initial(user_id).to_f
  # Get weight values for last 28 days
  weight_array = Array.new
  number_of_periods = 28
  (0..number_of_periods).each do |period_num|
    time_at_point_in_past = Time.now-(number_of_periods-period_num).day

    if ( time_at_point_in_past < time_first_reading )
      weight = weight_first_reading
    else
      weight = Reading.weight_at_time(user_id, time_at_point_in_past)
    end
    # Three significant digits to stop Gruff graph library from acting strangely
    weight = ((weight * 10000).to_i)/10000.0
    weight_array.push(weight)
  end

  g.data "28 days", weight_array
  send_data(g.to_blob, :type => 'image/png', :filename => "28days.png")

end
```

10 Make pretty layout

1. ☒ Run CSS application (See Github)
2. ☒ Create welcome/graph.html.erb view
3. ☒ Create graph method in welcome controller
4. ☒ Add route
5. ☒ Add link to graph view in layout

11 Revisit analysis

1. ☐ Link welcome.html.erb

12 Add last weight reading as words helper

1. ☐ add method to welcome controller

```
def self.get_last_reading( user_id )
  return Reading.order('reading_time ASC').where(:user_id => user_id).last
end
```

13 Figure out when we can achieve goal

```
# welcome_helper.rb
user_id = session[:user_id]
goal_loss_rate = User.goal_loss_rate(user_id)
lbs_per_second = goal_loss_rate / 3500 / 86400
```

14 Graph last two years

```
  def month
    g = Gruff::Line.new
    weight = 0
    time_at_point_in_past = 0
    user_id = session[:user_id]
    time_first_reading = Reading.time_initial(user_id)
    weight_first_reading = Reading.weight_initial(user_id).to_f
    # Get weight values for last 28 days
    weight_array = Array.new
    number_of_periods = 28
    (0..number_of_periods).each do |period_num|
      time_at_point_in_past = Time.now-(number_of_periods-period_num).day

      if ( time_at_point_in_past < time_first_reading )
        weight = weight_first_reading
      else
        weight = Reading.weight_at_time(user_id, time_at_point_in_past)
      end
      weight_array.push(weight)
    end

    g.data "28 days", weight_array
    send_data(g.to_blob, :type => 'image/png', :filename => "28days.png")

  end

  def year
  end
end
```

1. ☐ Add view
2. ☐ Add route

15 Footer

1. ☒ Put function to find goal difference in the Reading model

```
def self.goal_difference( user_id )
  goal_now = User.goal_now(user_id)
  weight_now = Reading.weight_at_time(user_id, Time.now)
  return goal_now - weight_now
end
```

1. ☒ in application helper, footer method

```
def footer
  if session[:user_id]
    user_id = session[:user_id]
    lbs = number_with_precision(@diff, :precision => 1, :significant => true)
    goal_difference = Reading.goal_difference(user_id)
    # cals = @diff * 3500
    # cals = number_with_precision(cals, :precision => 2, :significant => true)
    #      return "#{lbs} lbs (#{cals} cal)"
    return "#{lbs} lbs"
  else
    return "nil"
  end
end
```

16 About your last reading

1. ☐ Refactor $\text{last}_{\text{reading}}$ = $\text{Reading.get}_{\text{lastreading}}(\text{user}_{\text{id}})$ helper to $@\text{last}_{\text{reading}}$ in controller