November 7th, 2011

To BitLove,

I am writing to you to apply for the position of Rails & Javascript developer. I have over three years of writing Ruby on Rails websites, progressing from a fairly small presentation upload site through an employment application site, an online browser-based game to a full forums and marketing application. I believe my background, consisting of work at a theme park, on a cruise ship, for an audiovisual company and an online gaming company, would make an interesting addition to the BitLove family.

I first began seriously writing sites in RoR while working for VAE Corporation as the I3S, creating several web applications that they still use today. While writing those first sites, my knowledge and comfort with Ruby, Rails, HTML and Javascript expanded quite a bit, but I was limited in that no one else at my company knew anything about the technologies I was using. I was pretty much entirely self-taught and self-driving while at VAE; while this made for an enjoyable work experience on some levels, I had no one to fall back on or ask advice of.

In April of 2010, I was presented with the opportunity to work for an online gaming company, Simutronics. I had worked for Simutronics as an offsite contractor in one of their games (with a proprietary language, GSL) for over six years at that point, and I had wanted to work onsite for most of that time. I moved from Maryland (where last I had worked for VAE) to Missouri in order to work onsite for Simutronics as a Developer. Although initially hired for my Ruby on Rails experience, I was able to expand my knowledge of Ruby, Rails, Javascript, SQL, Sinatra, HTML, CSS, the Linux command system, and networking, as well as being exposed to Objective-C, Flash and C#.

I believe that, while my experience at Simutronics has been beneficial, I am limited in what I can learn from my coworkers in certain areas. I am seen as one of two database leads and the leading Ruby and Rails programmer in the office. Despite that, I know I have more to learn with high-throughput applications and wish to pursue an opportunity that would allow me to expand my knowledge more in that direction. BitLove seems to be the perfect fit in that regard. From what I have seen, BitLove has a similar work atmosphere as Simutronics, while also challenging me in regards to technologies I have not mastered yet.

I hope that my application piques your interest and I look forward to hearing from you in the future.

Michael Madison

# Michael Neal Madison, Jr.

1172 Schulte Hill Dr, Apt B
Maryland Heights, MO 63043
512.773.2960 (cell)
cadetstar@hotmail.com

## Technical Skills

Knowledge of multiple programming, scripting and query languages including GSL, HSL, Visual Basic, AJAX, HTML, Javascript, SQL and extensive experience with Ruby and Rails
Experience with Microsoft Office, Crimson Editor, RubyMine

## Work Experience

### Developer and OnSite GameMaster

*Simutronics Corporation – April 2010 to present*

Worked with a team of seven developers on *Fantasy University*, developed the first new forums system for Simutronics in seven years and expanded it to host the tinyheroes.com website. Supported the IT staff for both software and hardware maintenance and monitoring. Developed several small applications and continued development on Quest systems for multiple Simutronics games.

### Information Systems Support Specialist

*Visual Aids Electronics (VAE) - April 2008 to April 2010*

Provided support company-wide for internal office equipment, including personal computers, copiers, printers, scanners and smartphones. Managed the email and spam-scanning servers, firewalls, web servers and application servers. Designed and installed custom internet access control configurations for various offices. Developed custom websites using Ruby on Rails, HTML and Javascript (see Websites Developed below). Created and maintained reports for inventory and billing system using direct SQL access to an Oracle database and Crystal Reports.

### Manager of Audiovisual Services

*Visual Aids Electronics (VAE) - September 2006 to April 2008*

Managed audiovisual operations at a hotel in the Houston Metropolitan Area. Duties included selling of audiovisual equipment, setup and strike of equipment, billing and personnel management.

## Additional Work Experience

| | |
|---|---|
| Stagehand (DC Area) | April 2008 to April 2010 |
| Audiovisual Technician for VAE | May 2006 to September 2006 |
| Sound Technician for Royal Caribbean Int'l | July 2005 to January 2006 |
| Stage Manager for Busch Gardens Williamsburg | August 2004 to November 2004 |
| Development GameMaster for *DragonRealms* | December 2003 to present |
| Stage Manager (various productions in Austin, TX) | January 2002 to May 2005 |
| TicketSeller and TicketSeller Supervisor | September 2000 to May 2005 |

## Websites Developed (* - indicates websites developed exclusively)

| | |
|---|---|
| careers.vaecorp.com* | conferencesystems.com* |
| vaepresentations.com* | survey.vaecorp.com* |
| www.fantasyuniversity.net | forums.play.net |
| simu360.heroku.com | vae-opinion.heroku.com* |

## Education

Bachelor of Arts in Theatre and Dance from The University of Texas at Austin (May 2004)
Bachelor of Science in Mathematics from The University of Texas at Austin (May 2004)

# References

Eric Latham
Producer, Simutronics Corporation
Professional Reference
ericl@simutronics.com
314.324.4297

Andy Kirk
IT Director, Visual Aids Electronics
Professional Reference
akirk@vaecorp.com
859.525.4823

Brett Sweeney
Developer, Simutronics Corporation
Professional Reference
ubiquitous42@gmail.com
573.201.3757

Sara Luebbers
Personal Reference
dtrsec@gmail.com
314.620.8556

These are pieces of Ruby on Rails code, some rake tasks and some in the MVC area, indicated for each snippet, that I have developed for tasks in the past. I have washed the code of anything sensitive to the project that the code was originally for.

---

Snippet #1 (Rake Take to Send Queued Emails)
(Note: In Fall of 2009, VAE decided to send virtual holiday cards instead of paper cards in order to progress in our green initiative. We wanted to personalize each card, so I wrote the code to dynamically generate a JPG from client information provided by our managers to send to the client's email address embedded in an email. I cannot include the PDF generation code here without removing most of it since the content is proprietary to VAE. This rake task was called every 10 minutes to process and send emails for seasons which had been cleared but not already sent. I am also in the process of rewriting this very application for VAE, and will probably be changing much of this code due to what I have learned since.)

---

```
require 'net/pop'

namespace :utils do
  desc "Send Emails for All Seasons"
  task(:send_emails_for_seasons) do
    Season.find_all_by_season_enabled(true).each do |season|
      thankyous = season.thankyous
      thankyous.delete_if {|a| !a.emailsent.nil? or a.updated_at >
      season.season_when_email or a.client.client_email == ""}
      thankyous.delete_if {|a| !a.client.client_email.include?("@") or
      a.client.client_email.include?("[") or a.client.client_email.include?
      ("]")}

      listofpeople = []

      season_ctr = 0

      thankyous.each do |i|
        listofpeople << i.client.name_std
        UserMailer.deliver_thankyou_email(i.property.manager, i)
        i.update_attributes(:emailsent => Time.now)
        season_ctr += 1
        break if season_ctr >= 10

      end

      if season_ctr > 0
        for k in Role.find_by_rolename('emailadmin').users
          UserMailer.deliver_seasonsent(k, season.season_name, season_ctr,
      listofpeople)
        end
      end
      print "\nI ran for season #{season.season_name} at
      #{Time.now.to_s(:date_time12)}.\n"
      #logger.info "\nI ran for season #{season.season_name} at
      #{Time.now.to_s(:date_time12)}.\n"
    end
  end
end
```

---

Snippet #2 (Rake Task to Send Reminder Emails)
(Note: When an administrator finds that a client's information is not in order, they can flag the survey and send a message to the manager in charge with the problems that were indicated. This flag lasts for seven days before it automatically expires. This task runs every morning at 12:05am to send reminder emails every two days if the flag has not been handled.)

```
require 'net/pop'

namespace :utils do
  desc "Send Flag Reminders"
  task(:send_flags) do
    Cif.find(:all, :conditions => ['flagged_until between ? and ? or flagged_until
      between ? and ? or flagged_until between ? and ?', 1.days.from_now,
      2.days.from_now, 3.days.from_now, 4.days.from_now, 5.days.from_now,
      6.days.from_now]).each do |cif|
      users = []
      users << cif.property.manager
      users << cif.property.users
      users.flatten!.uniq!

      users.each do |user|
        begin
          unless user.receive_flagged
            UserMailer.deliver_flagged_survey(cif, user, cif.flag_comment.to_s)
          end
        rescue Net::SMTPFatalError => e
          # Not actually doing anything
        rescue Net::SMTPServerBusy, Net::SMTPUnknownError, Net::SMTPSyntaxError,
            TimeoutError => e
          # Not actually doing anything
        end
      end
    end
  end
end
```

---

Snippet #3 (Convert Posts from old Oracle DB)
(Note: I recently rewrote the forums system at Simutronics Corporation to run on a new backend, as the database structure had not been updated in over 20 years and we wanted some additional functionality. We were also transitioning off of an Oracle DB to a PostgreSQL DB. This function could be called with certain restrictions so that the conversion could be done in chunks. Also, I implemented a shim in the .query method so that the remote server could contact an intermediary to fetch info from the old Oracle DB as the oracle DB was behind a firewall. See attached file: converter.rb

---

Snippet #4 (Generate Excel Summary Document of Response Scores)
(Note: This one is quite length as it involves specific logic to handle quarters and annual surveys separately. It is also set to *not* generate an entire year at once, but only Year-To-Date. This is included in the controller and is initiated from a hyperlink. This code turned a job which used to take eight hours to perform manually into 20 seconds of code execution and about 10 minutes of formatting. This code will also be rewritten

in the next few days as I'm rewriting this entire site for VAE.)

---

```
  def generatereport
    book = Spreadsheet::Workbook.new

    sheet1 = book.create_worksheet :name => 'CSP Results'

    sheet1[0,0] = 'Property'

    # Okay, let's pull our groups and properties first
    if current_user.administrate
      @agggroups = CifAggregate.find(:all).select{|a| a.any_active }
      @agggroups = @agggroups.sort_by{|a| a.properties.first.property_code }
    else
      @agggroups = []
      for i in current_user.allvalidprops
        if i.cif_include
          @agggroups << i.cifaggregate
        end
      end
      @agggroups.flatten!
      @agggroups.uniq!
    end

    rowscan = 2

    @agggroups.each do |agggroup|
      sheet1[rowscan,1] = agggroup.report_prop_codes
      sheet1[rowscan,0] = agggroup.agg_name
      rowscan += 1
    end

    sheet1[rowscan + 1,0] = 'TOTAL'

    timeend = Time.parse("#{params[:rep_month]}/1/#{params[:rep_year]}").at_end_of_month
    timeperiod = timeend.at_beginning_of_year

    columnscan = 2
    quarter = '1st'
    headformat = Spreadsheet::Format.new(:rotation => 80, :horizontal_align =>
      :center, :vertical_align => :center, :weight => :bold)
    qtrheadformat = Spreadsheet::Format.new(:rotation => 80, :horizontal_align
      => :center, :vertical_align => :center, :pattern_bg_color => :cyan,
      :pattern_fg_color => :cyan, :pattern => 1, :weight => :bold)
    month1headformat = Spreadsheet::Format.new(:rotation => 80, :horizontal_align
      => :center, :vertical_align => :center, :pattern_bg_color =>
      :yellow, :pattern_fg_color => :yellow, :pattern => 1, :weight => :bold)
    month2headformat = Spreadsheet::Format.new(:rotation => 80, :horizontal_align
      => :center, :vertical_align => :center, :pattern_bg_color => :lime,
      :pattern_fg_color => :lime, :pattern => 1, :weight => :bold)
    month3headformat = Spreadsheet::Format.new(:rotation => 80, :horizontal_align
      => :center, :vertical_align => :center, :pattern_bg_color =>
      :silver, :pattern_fg_color => :silver, :pattern => 1, :weight => :bold)
    yearheadformat = Spreadsheet::Format.new(:rotation => 80, :horizontal_align
      => :center, :vertical_align => :center, :weight => :bold)

    month1color = Spreadsheet::Format.new(:pattern_bg_color => :yellow,
      :pattern_fg_color => :yellow, :pattern => 1, :horizontal_align =>
      :center)
    month2color = Spreadsheet::Format.new(:pattern_bg_color => :lime,
      :pattern_fg_color => :lime, :pattern => 1, :horizontal_align =>
```

```
      :center)
    month3color = Spreadsheet::Format.new(:pattern_bg_color => :silver,
      :pattern_fg_color => :silver, :pattern => 1, :horizontal_align =>
      :center)
    yearcolor = Spreadsheet::Format.new(:horizontal_align => :center)
    qtrcolor = Spreadsheet::Format.new(:pattern_bg_color => :cyan,
      :pattern_fg_color => :cyan, :pattern => 1, :horizontal_align =>
      :center)
    defformat = Spreadsheet::Format.new(:horizontal_align => :center)
    leftformat = Spreadsheet::Format.new(:horizontal_align => :left)

    for i in 0..100 do
      sheet1.column(i).width = 6
    end

    sheet1.column(0).default_format = leftformat
    sheet1.column(0).width = 30
    sheet1.column(1).default_format = defformat
    sheet1.column(1).width = 20

    end_period = timeend

    while timeperiod < end_period do
      rowscan = 2
      thisformat = case timeperiod.month % 3
        when 0
          month1headformat
        when 1
          month2headformat
        when 2
          month3headformat
      end
      sheet1[0,columnscan] = "#{timeperiod.strftime("%b %y")} # CIF Received"
      sheet1.row(0).set_format(columnscan, thisformat)
      sheet1[0,columnscan + 1] = "#{timeperiod.strftime("%b %y")} Response Rate"
      sheet1.row(0).set_format(columnscan + 1, thisformat)
      sheet1[0,columnscan + 2] = "#{timeperiod.strftime("%b %y")} # CIF Sent"
      sheet1.row(0).set_format(columnscan + 2, thisformat)
      sheet1[0,columnscan + 3] = "#{timeperiod.strftime("%b %y")} Overall
      Satisfaction"
      sheet1.row(0).set_format(columnscan + 3, thisformat)
      sheet1[0,columnscan + 4] = "#{timeperiod.strftime("%b %y")} Average Score"
      sheet1.row(0).set_format(columnscan + 4, thisformat)

      thisformat = case timeperiod.month % 3
        when 0
          month1color
        when 1
          month2color
        when 2
          month3color
      end

      gtcifs = []

      @agggroups.each do |agggroup|
        totalcifs = []
        agggroup.properties.select{|a| a.cif_include }.each do |property|
          totalcifs << property.cifs.find(:all, :conditions => ['sent_at is not NULL and
count_survey = 1 and cif_captured != 1 and created_at between ? and ?', timeperiod,
timeperiod.at_end_of_month])
        end
        totalcifs.flatten!
```

```
  gtcifs << totalcifs

  cifs = totalcifs.select { |a| !a.completed_at.nil? }

  totaloverall = 0.0
  overallcounter = 0

  totalhold = 0.0
  counter = 0

  for j in cifs
    if j.average_score > 0
      totalhold += j.average_score
      counter += 1
    end
    if j.key_score > 0
      totaloverall += j.key_score
      overallcounter += 1
    end
  end

  if counter > 0
    totalhold = totalhold / counter
  end
  if overallcounter > 0
    totaloverall = totaloverall / overallcounter
  end

  sheet1[rowscan, columnscan] = cifs.size == 0 ? '-' : cifs.size
  sheet1[rowscan, columnscan + 1] = totalcifs.size == 0 ? '-' :
"#{sprintf('%.0f', 100 * cifs.size / totalcifs.size)}%"
  sheet1[rowscan, columnscan + 2] = totalcifs.size == 0 ? '-' :
totalcifs.size
  sheet1[rowscan, columnscan + 3] = cifs.size == 0 ? '-' : sprintf('%.2f',
totaloverall)
  sheet1[rowscan, columnscan + 4] = cifs.size == 0 ? '-' : sprintf('%.2f',
totalhold)
  rowscan += 1
end

gtcifs.flatten!
cifs = gtcifs.select { |a| !a.completed_at.nil? }

rowscan += 1

totaloverall = 0.0
overallcounter = 0

totalhold = 0.0
counter = 0
for j in cifs
  if j.average_score > 0
    totalhold += j.average_score
    counter += 1
  end
  if j.key_score > 0
    totaloverall += j.key_score
    overallcounter += 1
  end
end

if counter > 0
  totalhold = totalhold / counter
```

```
      end
      if overallcounter > 0
        totaloverall = totaloverall / overallcounter
      end

      sheet1[rowscan, columnscan] = cifs.size == 0 ? '-' : cifs.size
      sheet1[rowscan, columnscan + 1] = gtcifs.size == 0 ? '-' : "#{sprintf('%.0f',
      100 * cifs.size / gtcifs.size)}%"
      sheet1[rowscan, columnscan + 2] = gtcifs.size == 0 ? '-' : gtcifs.size
      sheet1[rowscan, columnscan + 3] = cifs.size == 0 ? '-' : sprintf('%.2f',
      totaloverall)
      sheet1[rowscan, columnscan + 4] = cifs.size == 0 ? '-' : sprintf('%.2f',
      totalhold)

      for i in 1..rowscan
        sheet1.row(i).set_format(columnscan, thisformat)
        sheet1.row(i).set_format(columnscan + 1, thisformat)
        sheet1.row(i).set_format(columnscan + 2, thisformat)
        sheet1.row(i).set_format(columnscan + 3, thisformat)
        sheet1.row(i).set_format(columnscan + 4, thisformat)
      end

      columnscan += 5
      timeperiod = timeperiod.next_month

      if timeperiod == timeperiod.at_beginning_of_quarter # We need to do the
      quarterly summaries first.
        rowscan = 2

        sheet1[0,columnscan] = "#{quarter} Qtr # CIF Received"
        sheet1.row(0).set_format(columnscan, qtrheadformat)
        sheet1[0,columnscan + 1] = "#{quarter} Qtr Response Rate"
        sheet1.row(0).set_format(columnscan + 1, qtrheadformat)
        sheet1[0,columnscan + 2] = "#{quarter} Qtr # CIF Sent"
        sheet1.row(0).set_format(columnscan + 2, qtrheadformat)
        sheet1[0,columnscan + 3] = "#{quarter} Qtr Avg. Overall Satisfaction"
        sheet1.row(0).set_format(columnscan + 3, qtrheadformat)
        sheet1[0,columnscan + 4] = "#{quarter} Qtr Average Score"
        sheet1.row(0).set_format(columnscan + 4, qtrheadformat)

        gtcifs = []

        @agggroups.each do |agggroup|
          totalcifs = []
          agggroup.properties.select{|a| a.cif_include }.each do |property|
            totalcifs << property.cifs.find(:all, :conditions => ['sent_at is not NULL
and count_survey = 1 and cif_captured != 1 and created_at between ? and ?',
timeperiod.last_month.at_beginning_of_quarter, timeperiod.last_month.at_end_of_quarter])
          end
          totalcifs.flatten!
          gtcifs << totalcifs

          cifs = totalcifs.select { |a| !a.completed_at.nil? }

          totaloverall = 0.0
          overallcounter = 0

          totalhold = 0.0
          counter = 0
          for j in cifs
            if j.average_score > 0
              totalhold += j.average_score
              counter += 1
```

```
    end
    if j.key_score > 0
      totaloverall += j.key_score
      overallcounter += 1
    end
  end

  if counter > 0
    totalhold = totalhold / counter
  end
  if overallcounter > 0
    totaloverall = totaloverall / overallcounter
  end

  sheet1[rowscan, columnscan] = cifs.size == 0 ? '-' : cifs.size
  sheet1[rowscan, columnscan + 1] = totalcifs.size == 0 ? '-' :
    "#{sprintf('%.0f', 100 * cifs.size / totalcifs.size)}%"
  sheet1[rowscan, columnscan + 2] = totalcifs.size == 0 ? '-' :
    totalcifs.size
  sheet1[rowscan, columnscan + 3] = cifs.size == 0 ? '-' : sprintf('%.2f',
    totaloverall)
  sheet1[rowscan, columnscan + 4] = cifs.size == 0 ? '-' : sprintf('%.2f',
    totalhold)
  rowscan += 1
end

gtcifs.flatten!
cifs = gtcifs.select { |a| !a.completed_at.nil? }

rowscan += 1

  totaloverall = 0.0
  overallcounter = 0

  totalhold = 0.0
  counter = 0
  for j in cifs
    if j.average_score > 0
      totalhold += j.average_score
      counter += 1
    end
    if j.key_score > 0
      totaloverall += j.key_score
      overallcounter += 1
    end
  end

  if counter > 0
    totalhold = totalhold / counter
  end
  if overallcounter > 0
    totaloverall = totaloverall / overallcounter
  end

sheet1[rowscan, columnscan] = cifs.size == 0 ? '-' : cifs.size
sheet1[rowscan, columnscan + 1] = gtcifs.size == 0 ? '-' :
    "#{sprintf('%.0f', 100 * cifs.size / gtcifs.size)}%"
sheet1[rowscan, columnscan + 2] = gtcifs.size == 0 ? '-' : gtcifs.size
sheet1[rowscan, columnscan + 3] = cifs.size == 0 ? '-' : sprintf('%.2f',
    totaloverall)
sheet1[rowscan, columnscan + 4] = cifs.size == 0 ? '-' : sprintf('%.2f',
    totalhold)
```

```
        for i in 1..rowscan
          sheet1.row(i).set_format(columnscan, qtrcolor)
          sheet1.row(i).set_format(columnscan + 1, qtrcolor)
          sheet1.row(i).set_format(columnscan + 2, qtrcolor)
          sheet1.row(i).set_format(columnscan + 3, qtrcolor)
          sheet1.row(i).set_format(columnscan + 4, qtrcolor)
        end

        columnscan += 5
        quarter = case quarter
          when '1st'
            '2nd'
          when '2nd'
            '3rd'
          when '3rd'
            '4th'
        end
      end
    end

    # Now do the end of year stuff

    rowscan = 2

    timeperiod = timeend.at_beginning_of_year

    sheet1[0,columnscan] = "#{timeperiod.strftime("%Y")} # CIF Received YTD"
    sheet1.row(0).set_format(columnscan, yearheadformat)
    sheet1[0,columnscan + 1] = "#{timeperiod.strftime("%Y")} Response Rate YTD"
    sheet1.row(0).set_format(columnscan + 1, yearheadformat)
    sheet1[0,columnscan + 2] = "#{timeperiod.strftime("%Y")} # CIF Sent YTD"
    sheet1.row(0).set_format(columnscan + 2, yearheadformat)
    sheet1[0,columnscan + 3] = "#{timeperiod.strftime("%Y")} Avg. Overall YTD"
    sheet1.row(0).set_format(columnscan + 3, yearheadformat)
    sheet1[0,columnscan + 4] = "#{timeperiod.strftime("%Y")} Average Score YTD"
    sheet1.row(0).set_format(columnscan + 4, yearheadformat)

    gtcifs = []
    @agggroups.each do |agggroup|
      totalcifs = []
      agggroup.properties.select{|a| a.cif_include }.each do |property|
        totalcifs << property.cifs.find(:all, :conditions => ['sent_at is not NULL and
count_survey = 1 and cif_captured != 1 and created_at between ? and ?',
timeperiod.at_beginning_of_year, timeend])
      end
      totalcifs.flatten!
      gtcifs << totalcifs

      cifs = totalcifs.select { |a| !a.completed_at.nil? }

      totaloverall = 0.0
      overallcounter = 0

      totalhold = 0.0
      counter = 0
      for j in cifs
        if j.average_score > 0
          totalhold += j.average_score
          counter += 1
        end
        if j.key_score > 0
          totaloverall += j.key_score
          overallcounter += 1
```

```
    end
  end

  if counter > 0
    totalhold = totalhold / counter
  end
  if overallcounter > 0
    totaloverall = totaloverall / overallcounter
  end


  sheet1[rowscan, columnscan] = cifs.size == 0 ? '-' : cifs.size
  sheet1[rowscan, columnscan + 1] = totalcifs.size == 0 ? '-' :
        "#{sprintf('%.0f', 100 * cifs.size / totalcifs.size)}%"
  sheet1[rowscan, columnscan + 2] = totalcifs.size == 0 ? '-' : totalcifs.size
  sheet1[rowscan, columnscan + 3] = cifs.size == 0 ? '-' : sprintf('%.2f',
        totaloverall)
  sheet1[rowscan, columnscan + 4] = cifs.size == 0 ? '-' : sprintf('%.2f',
        totalhold)
  rowscan += 1
end

gtcifs.flatten!
cifs = gtcifs.select { |a| !a.completed_at.nil? }

rowscan += 1

totaloverall = 0.0
overallcounter = 0

totalhold = 0.0
counter = 0
for j in cifs
  if j.average_score > 0
    totalhold += j.average_score
    counter += 1
  end
  if j.key_score > 0
    totaloverall += j.key_score
    overallcounter += 1
  end
end

if counter > 0
  totalhold = totalhold / counter
end
if overallcounter > 0
  totaloverall = totaloverall / overallcounter
end

sheet1[rowscan, columnscan] = cifs.size == 0 ? '-' : cifs.size
sheet1[rowscan, columnscan + 1] = gtcifs.size == 0 ? '-' : "#{sprintf('%.0f',
  100 * cifs.size / gtcifs.size)}%"
sheet1[rowscan, columnscan + 2] = gtcifs.size == 0 ? '-' : gtcifs.size
sheet1[rowscan, columnscan + 3] = cifs.size == 0 ? '-' : sprintf('%.2f',
  totaloverall)
sheet1[rowscan, columnscan + 4] = cifs.size == 0 ? '-' : sprintf('%.2f',
  totalhold)

for i in 1..rowscan
  sheet1.row(i).set_format(columnscan, yearcolor)
  sheet1.row(i).set_format(columnscan + 1, yearcolor)
  sheet1.row(i).set_format(columnscan + 2, yearcolor)
```

```
    sheet1.row(i).set_format(columnscan + 3, yearcolor)
    sheet1.row(i).set_format(columnscan + 4, yearcolor)
  end

  columnscan += 5

  rowscan = 2
  @agggroups.each do |agggroup|
    sheet1[rowscan,columnscan] = agggroup.report_prop_codes
    sheet1[rowscan,columnscan + 1] = agggroup.agg_name
    rowscan += 1
  end

  sheet1[1,columnscan] = ""
  sheet1[rowscan,columnscan] = ""

  sheet1.column(columnscan + 1).default_format = leftformat
  sheet1.column(columnscan + 1).width = 30
  sheet1.column(columnscan).default_format = defformat
  sheet1.column(columnscan).width = 20

  sheet1[rowscan + 1, columnscan + 1] = 'TOTAL'

  filename = "#{Time.now.to_s(:file_date)}.xls"

  book.write "#{RAILS_ROOT}/excel/#{filename}"

  send_file "#{RAILS_ROOT}/excel/#{filename}", :filename => filename
end
```

---