Ruby Reports

Beyond 1.0

RubyEast Conference
Brown & Milner 2007.09.28

```ruby
def get(name)
  case name
  when String,Symbol
    self[name] || send(name)
  when Fixnum
    self[name]
  else
    raise ArgumentError, "Whatchu Talkin' Bout, Willis?"
  end
end
```

```ruby
def add_columns(names,options={})
  raise "Greg isn't smart enough to figure this out.\n"+
        "Send ideas in at http://list.rubyreports.org" if block_given?
  need_reverse = !!(options[:after] || options[:position])
  names = names.reverse if need_reverse
  names.each { |n| add_column(n,options) }
  self
end
```

Reporting Sucks

Ruby Reports
makes reporting
suck less.

No massive API to master

name	mailing_list_posts	date	user_id
Gregory Brown	3	2007.09.12	990
Gregory Brown	2	2007.09.13	990
Brad Ediger	1	2007.09.12	1001
Mike Milner	7	2007.09.13	999
Mike Milner	2	2007.09.14	990
Sam Jackson	4	2007.09.17	102

```ruby
table = Table("foo.csv") do |t,row|
  t.transform { |r| r.user_id = Integer(r.user_id) }
  t.filter { |r| r.user_id < 1000 }
  t << row
end

g = Grouping(table, :by => "name", :order => :name)
g.to_pdf(:file => "grouped.pdf")
```

Gregory Brown

mailing_list_posts	date	user_id
3	2007.09.12	990
2	2007.09.13	990

Mike Milner

mailing_list_posts	date	user_id
7	2007.09.13	999
2	2007.09.14	990

Sam Jackson

mailing_list_posts	date	user_id
4	2007.09.17	102

Obsessively Extensible

```ruby
class EnglishFormatter < Ruport::Formatter
  renders :english, :for => [Ruport::Renderer::Group,
                             Ruport::Renderer::Grouping]

  def build_group_header
    output << "#{data.name} had:\n"
  end

  def build_group_body
    data.each do |e|
      output << "- #{e.mailing_list_posts} Mailing List Posts on " <<
                "#{e.date}.\n"
    end
  end

  def build_grouping_body
    data.each do |n,g|
      render_group(g)
      output << "\n"
    end
  end
end
```

Gregory Brown had:
- 3 Mailing List Posts on 2007.09.12.
- 2 Mailing List Posts on 2007.09.13.

Mike Milner had:
- 7 Mailing List Posts on 2007.09.13.
- 2 Mailing List Posts on 2007.09.14.

Sam Jackson had:
- 4 Mailing List Posts on 2007.09.17.

```
g.to_pdf(:file => “grouped.pdf”)
```

becomes

```
g.to_english(:file => “grouped.txt”)
```

Plays Well With Others

```ruby
class PostTally < ActiveRecord::Base
  acts_as_reportable
end

table = PostTally.report_table(:all,
            :only => %w[name mailing_list_posts date user_id],
            :conditions => ["user_id < ?",1000])
```

Ruport isn’t a framework, it’s a foundation

Ruport and Rails

acts_as_reportable

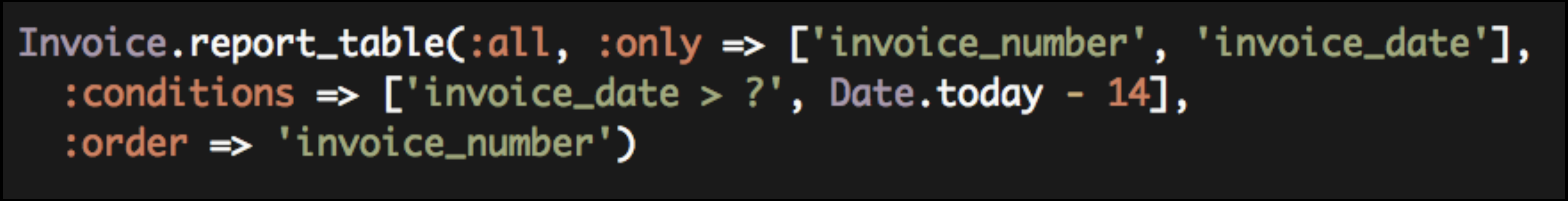
Ruport + ActiveRecord for data collection

```ruby
class Invoice < ActiveRecord::Base
  acts_as_reportable
  has_many :line_items

  def total
    ...
  end
end

class LineItem < ActiveRecord::Base
  acts_as_reportable
  belongs_to :invoice
end
```

```ruby
Invoice.report_table(:all,
  :only => ['invoice_number', 'invoice_date', 'line_items.amount'],
  :methods => :total,
  :include => { :line_items => { :only => 'amount' } })
```

invoice_number	invoice_date	line_items.amount
55411	2007-09-24	5000.26
55412	2007-09-24	800.34
55412	2007-09-24	736.22
55413	2007-09-25	2334.67
55414	2007-09-25	34355.66
55415	2007-09-25	2323.88

ActiveRecord Options



```ruby
Invoice.report_table(:all, :only => ['invoice_number', 'invoice_date'],
  :conditions => ['invoice_date > ?', Date.today - 14],
  :order => 'invoice_number')
```

Filters and Transformations

```ruby
Invoice.report_table(:all, :except => 'id',
  :filters => lambda {|r| r.invoice_date > (Date.today - 14) },
  :transforms => lambda {|r| r.amount += 100.0 })
```

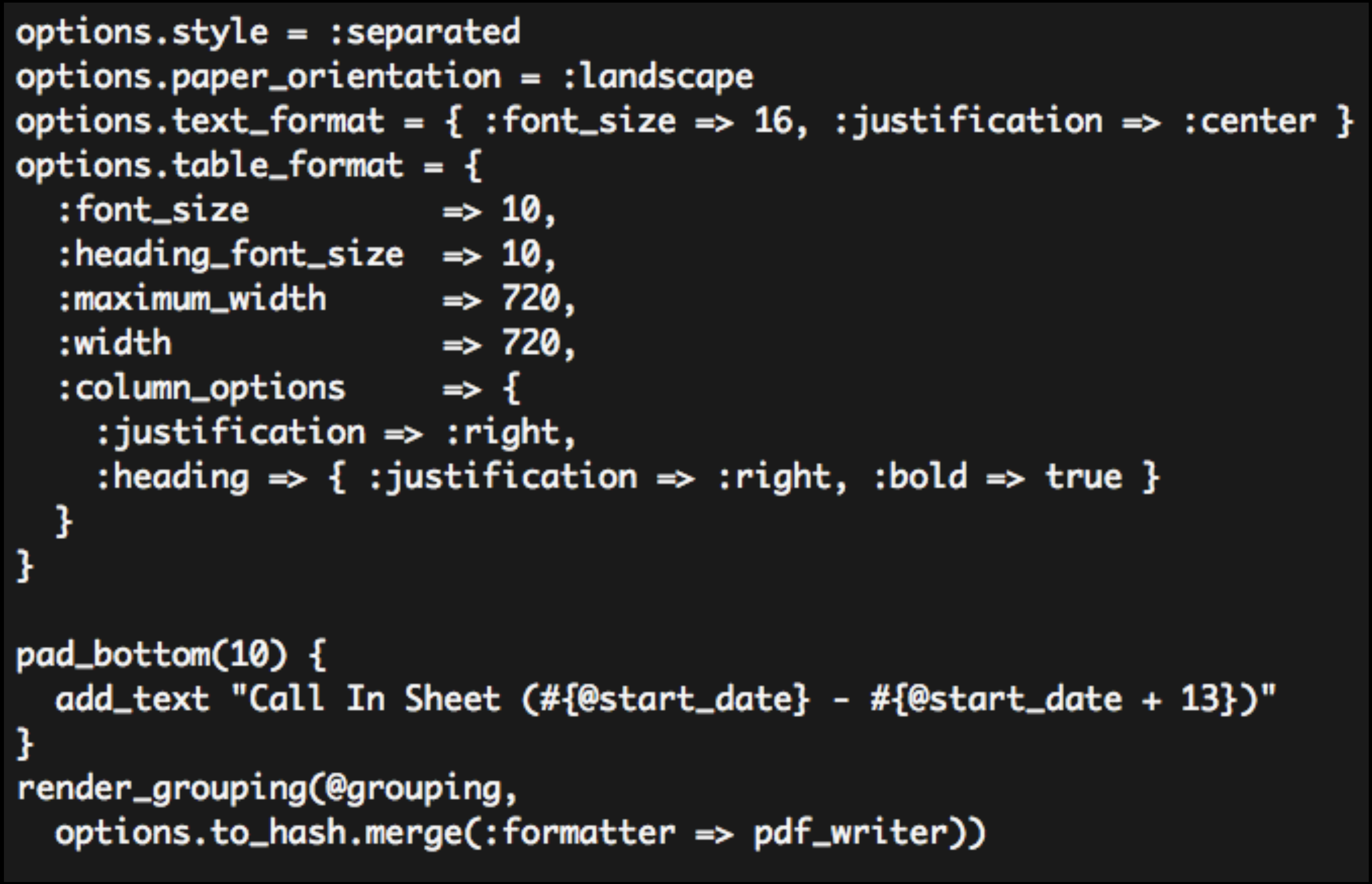
Ruport/Rails Plugin

Still Experimental
(but I’m using it in a production app)

```
class Timesheet < ActiveRecord::Base
  acts_as_reportable

  def regular_hours
    week1 + week2
  end

  def overtime
    (s = week1 + week2 - 80) > 0 ? s : 0.0
  end
end
```

```ruby
class TimesheetsController < ApplicationController

  def callin
    table = Timesheet.report_table(:all,
              :methods  => [:regular_hours, :overtime],
              :only     => %w[employee_type employee week1 week2 regular_hours
                                 lunch overtime personal sick vacation],
              :order    => 'employee')
    table.rename_columns {|c| c.titleize }
    table.rename_columns("Employee Type" => "Type",
      "Week1" => "Week 1 - Regular",
      "Week2" => "Week 2 - Regular")
    @grouping = Grouping(table, :by => 'Type', :order => :name)
    @start_date = Date.today

    render :template => "timesheets/callin.report"
  end

end
```

callin.report



```
options.style = :separated
options.paper_orientation = :landscape
options.text_format = { :font_size => 16, :justification => :center }
options.table_format = {
  :font_size          => 10,
  :heading_font_size  => 10,
  :maximum_width      => 720,
  :width              => 720,
  :column_options     => {
    :justification => :right,
    :heading => { :justification => :right, :bold => true }
  }
}

pad_bottom(10) {
  add_text "Call In Sheet (#{@start_date} - #{@start_date + 13})"
}
render_grouping(@grouping,
  options.to_hash.merge(:formatter => pdf_writer))
```

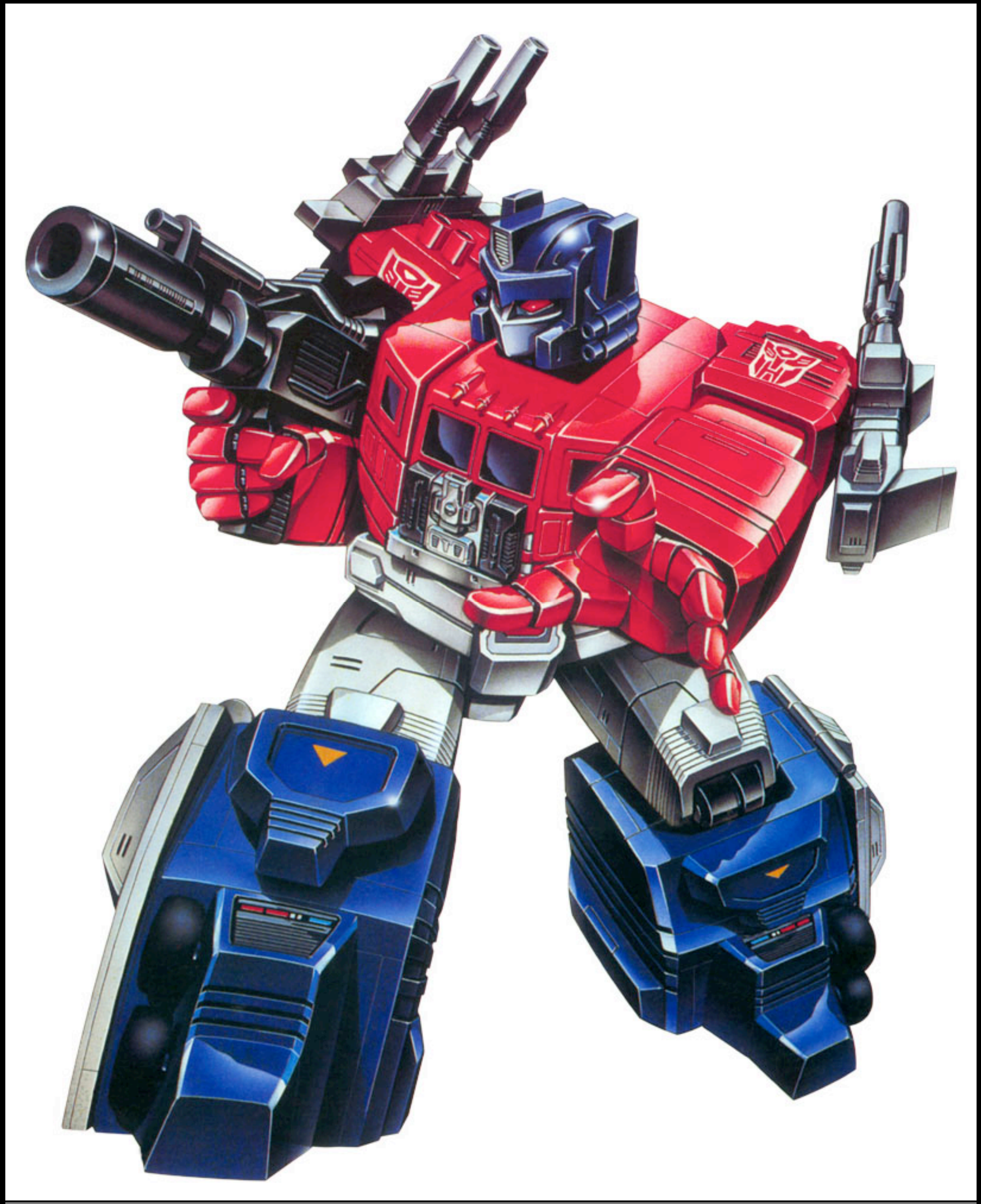
Limitations

- Simple Reports
- Only PDF

Suggestions are Welcome

New in 1.2

Data Feeders



TRANSFORM

```ruby
table = Table(%w[name num]) << %w[greg 1] << %w[joe 2] << %w[jim 3]

table.replace_column("name") { |r| r.name.capitalize }
table.replace_column("num") { |r| Integer(r.num) }
table.reduce { |r| (r.num % 2).nonzero? }
```

```ruby
table = Table(%w[name num]) do |feeder|
  feeder.transform do |r|
    r.name.capitalize!
    r.num = Integer(r.num)
  end

  feeder.filter { |r| (r.num % 2).nonzero? }

  feeder << %w[greg 1] << %w[joe 2] << %w[jim 3]
end
```

name	num
Greg	1
Jim	3

```ruby
class Condition
  attr_accessor :field, :relation, :value

  def initialize(options = {})
    options.symbolize_keys!
    @field, @relation, @value =  options[:field],
                                 options[:relation],
                                 options[:value]
  end

  def to_condition
    case @relation.to_sym
      when :starts_with: ["#{@field} like ?", "#{value}%"]
      when :ends_with: ["#{@field} like ?", "%#{value}"]
      when :contains: ["#{@field} like ?", "%#{value}%"]
      else ["#{@field} #{@relation} ?", value]
    end
  end

  def to_proc
    meth = @field.sub(/^_/,"")
    case @relation.to_sym
    when :starts_with
      lambda { |r| r.get(meth) =~ Regexp.new("^#{@value}") }
    when :ends_with
      lambda { |r| r.get(meth) =~ Regexp.new("#{@value}$") }
    when :contains
      lambda { |r| r.get(meth).to_s.include?(@value) }
    when :"="
      lambda { |r| r.get(meth).to_s == @value }
    else
      lambda { |r|
        res = r.get(meth)
        if res =~ /\d/
          Float(res).send(@relation,Float(@value))
        else
          (res.to_s).send(@relation,@value.to_s)
        end
        }
    end
  end

end
```

```ruby
name_contains_joe = Condition.new(:field => "_name",
                                  :relation => :contains,
                                  :value => "joe" )

zipcode_is_06510 = Condition.new(:field => "_zipcode",
                                 :relation => "=",
                                 :value => "06510" )

filters = [name_contains_joe, zipcode_is_06510].map { |c| c.to_proc }

MyModel.report_table(:all, :methods => ["name","zipcode"],
                           :filters => filters)
```

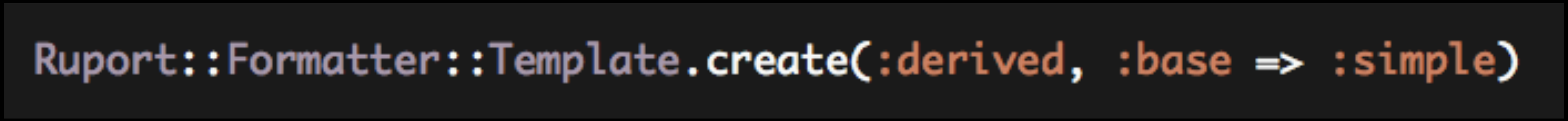
```ruby
class Array
  def feed_element(element)
    element
  end
end

int_array = []
feeder = Ruport::Data::Feeder.new(int_array)
feeder.filter { |r| r.kind_of?(Integer) }

feeder << 1 << "5" << 4.7 << "kitten" << 4
int_array #=> [1, 4]
```

Templates and Formatting

Create a Template

```ruby
Ruport::Formatter::Template.create(:simple) do |t|
  t.page_format = {
    :size   => "LETTER",
    :layout => :landscape
  }
end
```

Or Derive One

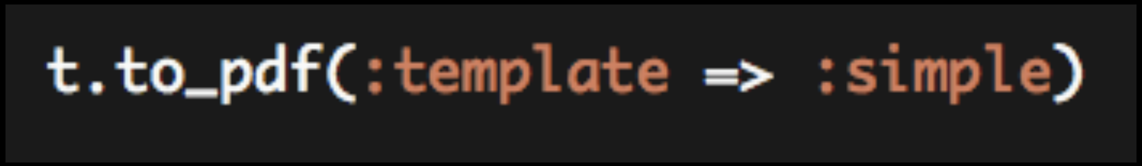


```ruby
Ruport::Formatter::Template.create(:derived, :base => :simple)
```

Define apply_template

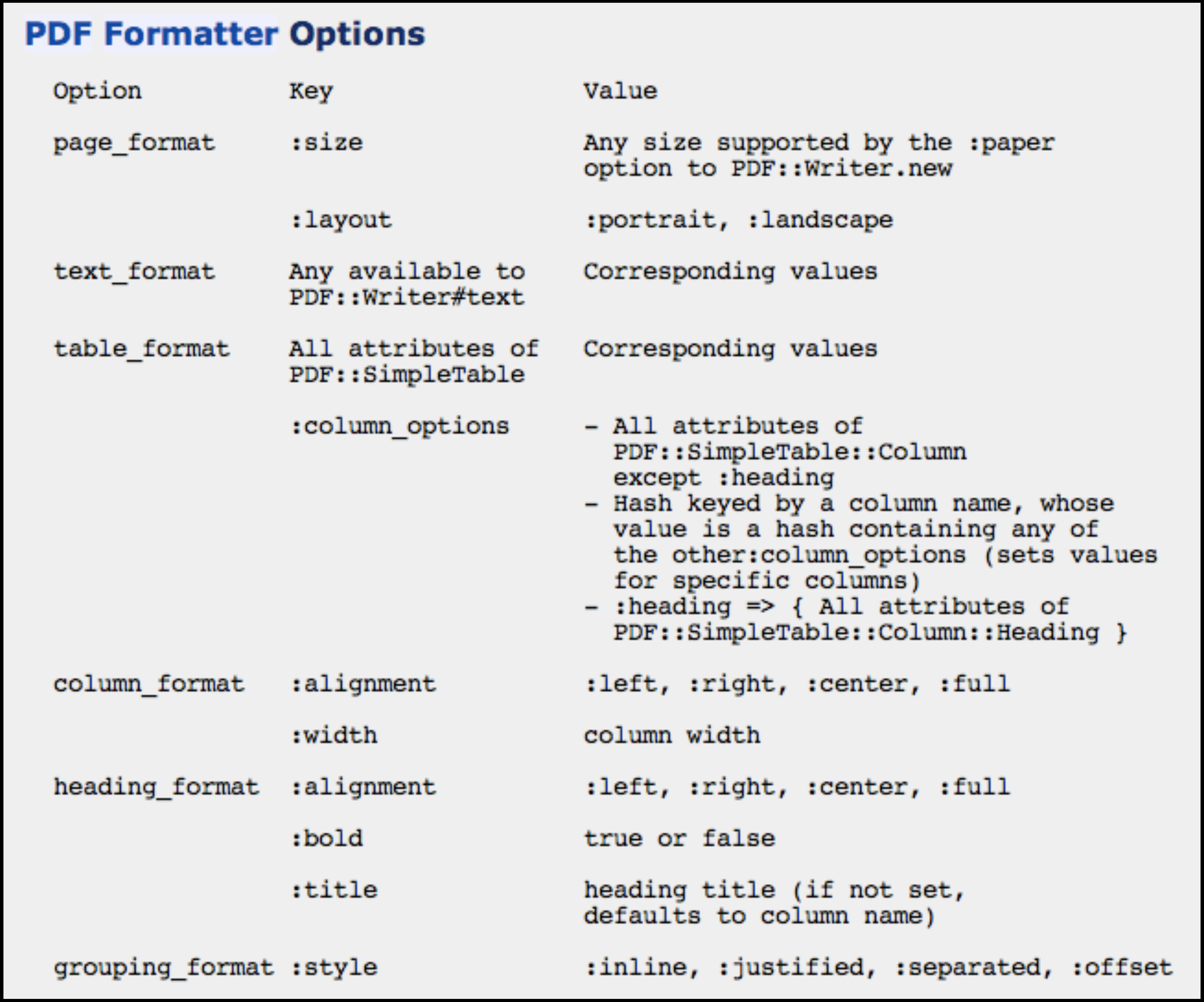
```ruby
class Ruport::Formatter::PDF

  def apply_template
    options.paper_size = template.page_format[:size]
    options.paper_orientation = template.page_format[:layout]
  end

end
```

Use it when Rendering



```
t.to_pdf(:template => :simple)
```

Built-in Formatters

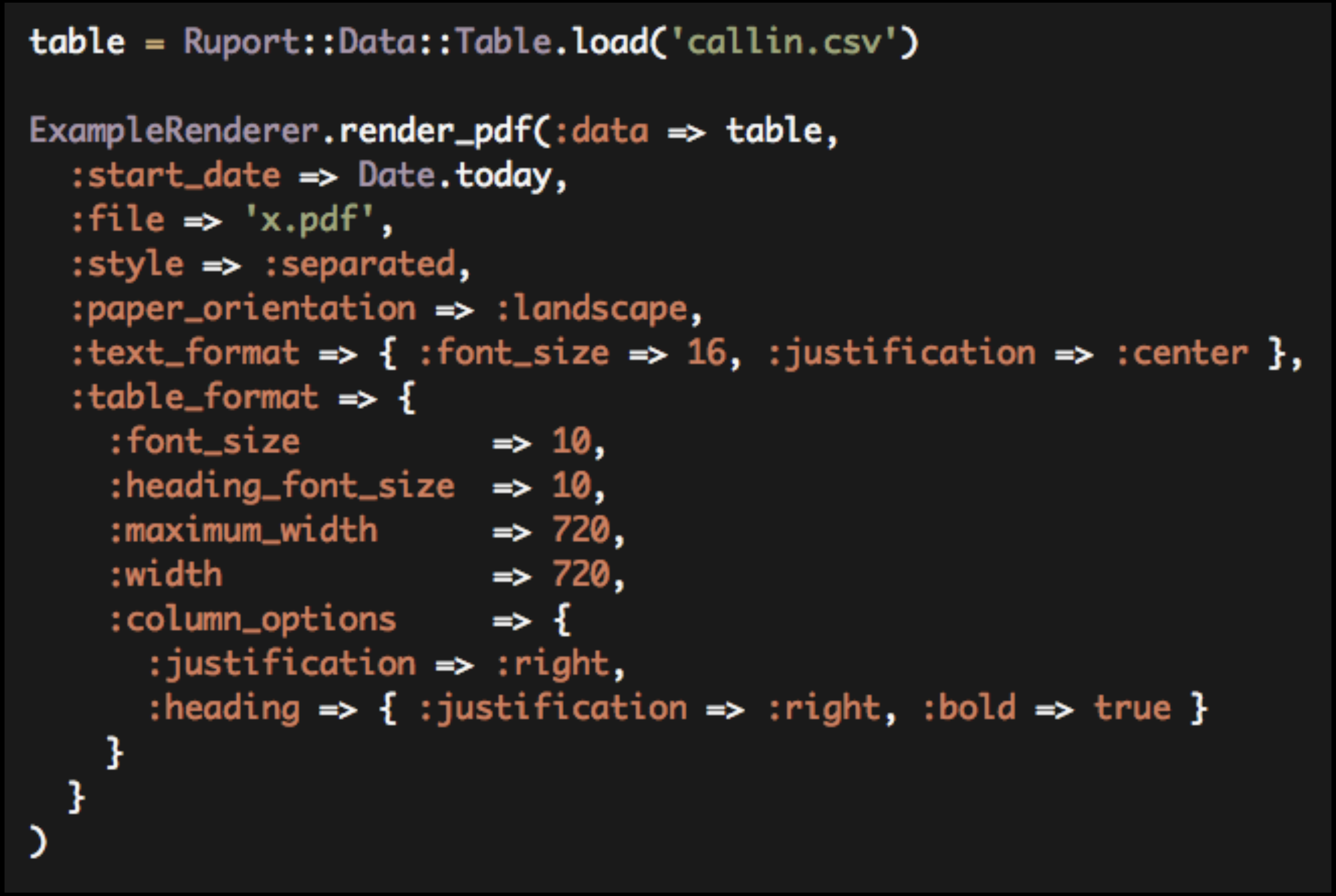


PDF Formatter Options

Option	Key	Value
page_format	:size	Any size supported by the :paper option to PDF::Writer.new
	:layout	:portrait, :landscape
text_format	Any available to PDF::Writer#text	Corresponding values
table_format	All attributes of PDF::SimpleTable	Corresponding values
	:column_options	- All attributes of PDF::SimpleTable::Column except :heading - Hash keyed by a column name, whose value is a hash containing any of the other:column_options (sets values for specific columns) - :heading => { All attributes of PDF::SimpleTable::Column::Heading }
column_format	:alignment	:left, :right, :center, :full
	:width	column width
heading_format	:alignment	:left, :right, :center, :full
	:bold	true or false
	:title	heading title (if not set, defaults to column name)
grouping_format	:style	:inline, :justified, :separated, :offset

```ruby
class ExampleRenderer < Ruport::Renderer
  stage :report

  def setup
    data.sort_rows_by!('employee')
    data.add_column('Regular Hours', :after => "week2") {|t|
      t.week1.to_f + t.week2.to_f
    }
    data.add_column('Overtime', :after => "lunch") {|t|
      (s = t.week1.to_f + t.week2.to_f - 80) > 0 ? s : 0.0
    }
    data.rename_columns {|c| c.titleize }
    data.rename_columns("Employee Type" => "Type",
      "Week1" => "Week 1 - Regular",
      "Week2" => "Week 2 - Regular")
    options.grouping = Grouping(data, :by => 'Type', :order => :name)
  end

  class MyPDF < Ruport::Formatter::PDF
    renders :pdf, :for => ExampleRenderer

    def build_report
      pad_bottom(10) {
        add_text "Call In Sheet (#{options.start_date} " +
          "- #{options.start_date + 13})"
      }
      render_grouping options.grouping,
        options.to_hash.merge(:formatter => pdf_writer)
    end
  end
end
```

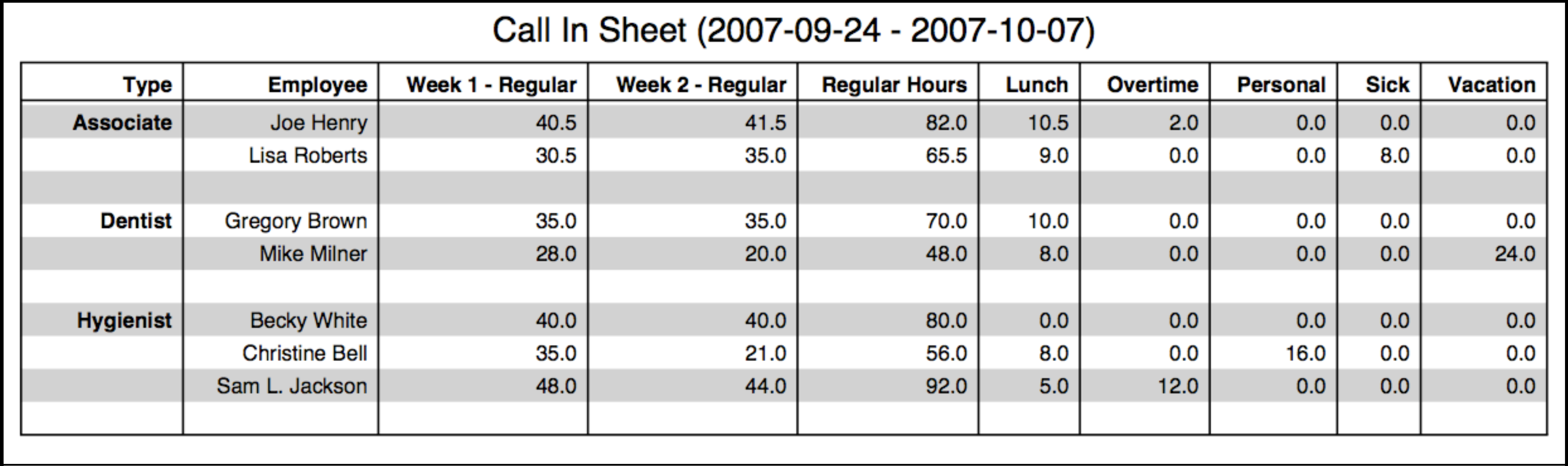
Without Templates



```
table = Ruport::Data::Table.load('callin.csv')

ExampleRenderer.render_pdf(:data => table,
  :start_date => Date.today,
  :file => 'x.pdf',
  :style => :separated,
  :paper_orientation => :landscape,
  :text_format => { :font_size => 16, :justification => :center },
  :table_format => {
    :font_size          => 10,
    :heading_font_size  => 10,
    :maximum_width      => 720,
    :width              => 720,
    :column_options     => {
      :justification => :right,
      :heading => { :justification => :right, :bold => true }
    }
  }
)
```

With Templates

```ruby
Ruport::Formatter::Template.create(:simple) do |t|
  t.page_format = {
    :layout => :landscape
  }
  t.grouping_format = {
    :style => :separated
  }
  t.text_format = {
    :font_size => 16,
    :justification => :center
  }
  t.table_format = {
    :font_size          => 10,
    :heading_font_size  => 10,
    :maximum_width      => 720,
    :width              => 720
  }
  t.column_format = {
    :alignment => :right
  }
  t.heading_format = {
    :alignment => :right,
    :bold => true
  }
end

table = Ruport::Data::Table.load('callin.csv')

ExampleRenderer.render_pdf(:data => table,
  :start_date => Date.today,
  :file => 'x.pdf',
  :template => :simple
)
```



Call In Sheet (2007-09-24 - 2007-10-07)

Type	Employee	Week 1 - Regular	Week 2 - Regular	Regular Hours	Lunch	Overtime	Personal	Sick	Vacation
Associate	Joe Henry	40.5	41.5	82.0	10.5	2.0	0.0	0.0	0.0
	Lisa Roberts	30.5	35.0	65.5	9.0	0.0	0.0	8.0	0.0
Dentist	Gregory Brown	35.0	35.0	70.0	10.0	0.0	0.0	0.0	0.0
	Mike Milner	28.0	20.0	48.0	8.0	0.0	0.0	0.0	24.0
Hygienist	Becky White	40.0	40.0	80.0	0.0	0.0	0.0	0.0	0.0
	Christine Bell	35.0	21.0	56.0	8.0	0.0	16.0	0.0	0.0
	Sam L. Jackson	48.0	44.0	92.0	5.0	12.0	0.0	0.0	0.0

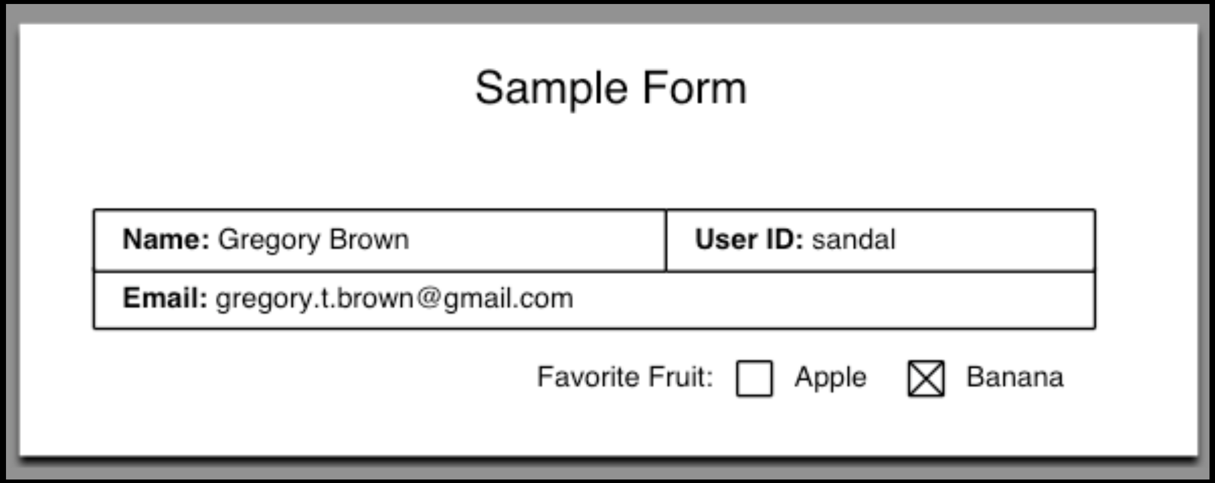
ruport-util

(a work in progress)

- Code generation via rope
- PDF form helpers
- Simple SVG and XML/SWF graphs
- Basic ODS output
- csv2ods util
- Invoice generation
- PDF Form drawing primitives

```ruby
require "ruport"
require "ruport/util"
class MyReport < Ruport::Report

  renders_as_table

  def prepare
    add_mailer :default,
      :host => "mail.adelphia.net",
      :address => "gregory.t.brown@gmail.com"
   end

  def renderable_data(format)
    Table(%w[a b c]) { |t| t << [1,2,3] << [4,5,6] }
  end

end

MyReport.generate do |report|
 report.save_as "foo.pdf"
 report.send_to("gregory.t.brown@gmail.com") do |mail|
   mail.subject = "Sample report"
   mail.attach "foo.pdf"
   mail.text = <<-EOS
     this is a sample of sending an emailed report from within Ruport.
   EOS
 end
end
```

```ruby
class SimpleForm < Ruport::Renderer

  stage :form_header, :form_body
  finalize :form

  required_option :name, :userid, :fruit, :email

  class PDF < Ruport::Formatter::PDF
    renders :pdf, :for => SimpleForm
    include Ruport::Util::FormHelpers

    def build_form_header
      add_text "Sample Form", :font_size => 16, :justification => :center
    end

    def build_form_body
      options.text_format = { :font_size => 10 }

      form_field "Name:", options.name,
        :x => 125, :y => 650, :width => 200, :border => [1,0,1,1]
      form_field "User ID:", options.userid,
        :x => 325, :y => 650, :width => 150
      form_field "Email:", options.email,
        :x => 125, :y => 630, :width => 350, :border => [0,1,1]

      draw_text "Favorite Fruit: ", :left => 280, :y => 600

      option_box :apple, options.fruit,
       :label => "Apple", :y => 340, :x => 600
      option_box :banana, options.fruit,
       :label => "Banana", :x => 400, :y => 600
    end

    def finalize_form
      render_pdf
    end

  end

end

puts SimpleForm.render_pdf { |r|
  r.name = "Gregory Brown"
  r.userid = "sandal"
  r.fruit = :banana
  r.email = "gregory.t.brown@gmail.com"
}
```



Sample Form

Name: Gregory Brown	**User ID:** sandal
Email: gregory.t.brown@gmail.com	

Favorite Fruit: ☐ Apple ☒ Banana

It’s a mess.

- Lots of tools still need work
- Officially supported but not part of core
- Haven’t quite decided on scope yet
- No promise of API stability (for now)

Challenges on the horizon

PDF::Writer
is a dead project

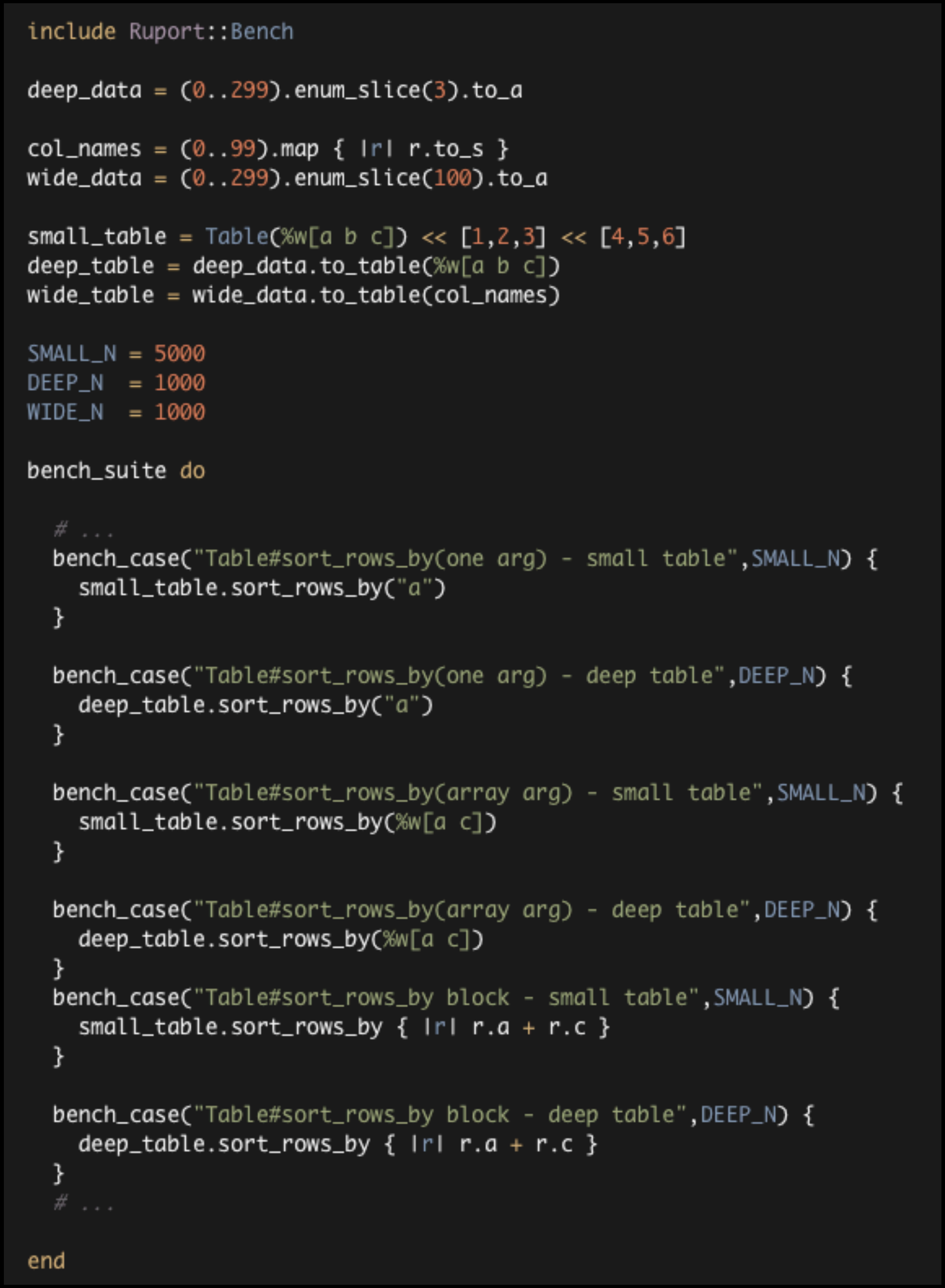
Abstract PDF adapters

- basic geometry tools (line / box drawing)
- table drawing with substantial formatting flexibility
- absolute text positioning
- automatic flow control
- control of page sizes, layouts, margins, etc.

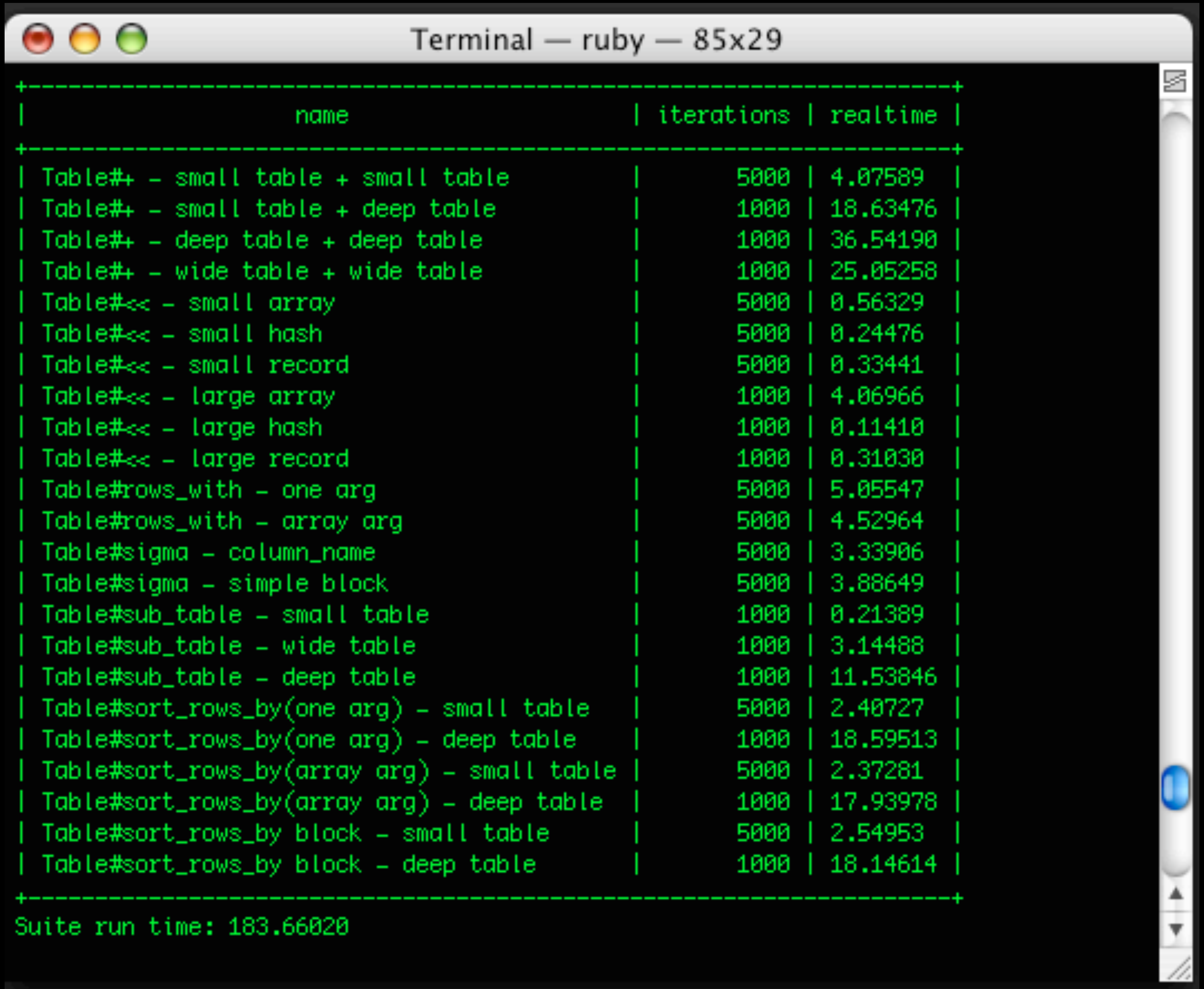
```ruby
module Ruport::PrintableDocumentAdapter

  def pdf_writer
    unless @pdf_writer
      @pdf_writer = ::FPDF.new
      @pdf_writer.AddPage
      @pdf_writer.SetFont(options[:font_style] || 'Arial','B',
                          options[:font_size]  || 16)
    end
    return @pdf_writer
  end

  def draw_text(text,options={})
    pdf_writer.Cell(options[:x],options[:y],text)
  end

  def render_pdf
    output << pdf_writer.Output
  end

end
```

We don’t want to read the PDF spec

Performance & Code Quality



```ruby
include Ruport::Bench

deep_data = (0..299).enum_slice(3).to_a

col_names = (0..99).map { |r| r.to_s }
wide_data = (0..299).enum_slice(100).to_a

small_table = Table(%w[a b c]) << [1,2,3] << [4,5,6]
deep_table = deep_data.to_table(%w[a b c])
wide_table = wide_data.to_table(col_names)

SMALL_N = 5000
DEEP_N  = 1000
WIDE_N  = 1000

bench_suite do

  # ...
  bench_case("Table#sort_rows_by(one arg) - small table",SMALL_N) {
    small_table.sort_rows_by("a")
  }

  bench_case("Table#sort_rows_by(one arg) - deep table",DEEP_N) {
    deep_table.sort_rows_by("a")
  }

  bench_case("Table#sort_rows_by(array arg) - small table",SMALL_N) {
    small_table.sort_rows_by(%w[a c])
  }

  bench_case("Table#sort_rows_by(array arg) - deep table",DEEP_N) {
    deep_table.sort_rows_by(%w[a c])
  }
  bench_case("Table#sort_rows_by block - small table",SMALL_N) {
    small_table.sort_rows_by { |r| r.a + r.c }
  }

  bench_case("Table#sort_rows_by block - deep table",DEEP_N) {
    deep_table.sort_rows_by { |r| r.a + r.c }
  }
  # ...

end
```



```
+-----------------------------------------------------------------------+
|                         name                  | iterations | realtime |
+-----------------------------------------------------------------------+
| Table#+ - small table + small table           |       5000 | 4.07589  |
| Table#+ - small table + deep table            |       1000 | 18.63476 |
| Table#+ - deep table + deep table             |       1000 | 36.54190 |
| Table#+ - wide table + wide table             |       1000 | 25.05258 |
| Table#<< - small array                        |       5000 | 0.56329  |
| Table#<< - small hash                         |       5000 | 0.24476  |
| Table#<< - small record                       |       5000 | 0.33441  |
| Table#<< - large array                        |       1000 | 4.06966  |
| Table#<< - large hash                         |       1000 | 0.11410  |
| Table#<< - large record                       |       1000 | 0.31030  |
| Table#rows_with - one arg                     |       5000 | 5.05547  |
| Table#rows_with - array arg                   |       5000 | 4.52964  |
| Table#sigma - column_name                     |       5000 | 3.33906  |
| Table#sigma - simple block                    |       5000 | 3.88649  |
| Table#sub_table - small table                 |       1000 | 0.21389  |
| Table#sub_table - wide table                  |       1000 | 3.14488  |
| Table#sub_table - deep table                  |       1000 | 11.53846 |
| Table#sort_rows_by(one arg) - small table     |       5000 | 2.40727  |
| Table#sort_rows_by(one arg) - deep table      |       1000 | 18.59513 |
| Table#sort_rows_by(array arg) - small table   |       5000 | 2.37281  |
| Table#sort_rows_by(array arg) - deep table    |       1000 | 17.93978 |
| Table#sort_rows_by block - small table        |       5000 | 2.54953  |
| Table#sort_rows_by block - deep table         |       1000 | 18.14614 |
+-----------------------------------------------------------------------+
Suite run time: 183.66020
```

- Write more benchmarks for Ruport
- Improve the quality of existing benchmarks
- Refactor ugly code that is creeping in
- Tighten up the API where needed

The Biggest Challenge

Ruport Book 2007.12.15

(we hope)

Questions?

GREGORY BROWN

MICHAEL MILNER

rubyreports.org

ruportbook.com