Surpass Manual

Ana Nelson

March 22, 2009

Contents

To compile this documentation requires LaTeX, Idiopidae, and my branch of webby, which is available from a bzr repository at .

This documentation refers to Surpass version 0.0.4.

Chapter 1

Installation and Hello World

1.1 Dependencies

Surpass only needs basic Ruby. It has been tested using Ruby 1.8.6 and JRuby 1.1.5.

For development, you will want to have access to something that can open Microsoft Excel files. This could be Microsoft Excel, Open Office, Google Docs or even a gmail account.

1.2 Gem Installation

```
sudo gem install surpass
```

1.3 Source Installation

```
bzr branch http://ananelson.com/code/surpass
cd surpass
sudo rake gem:install
```

1.4 Hello World

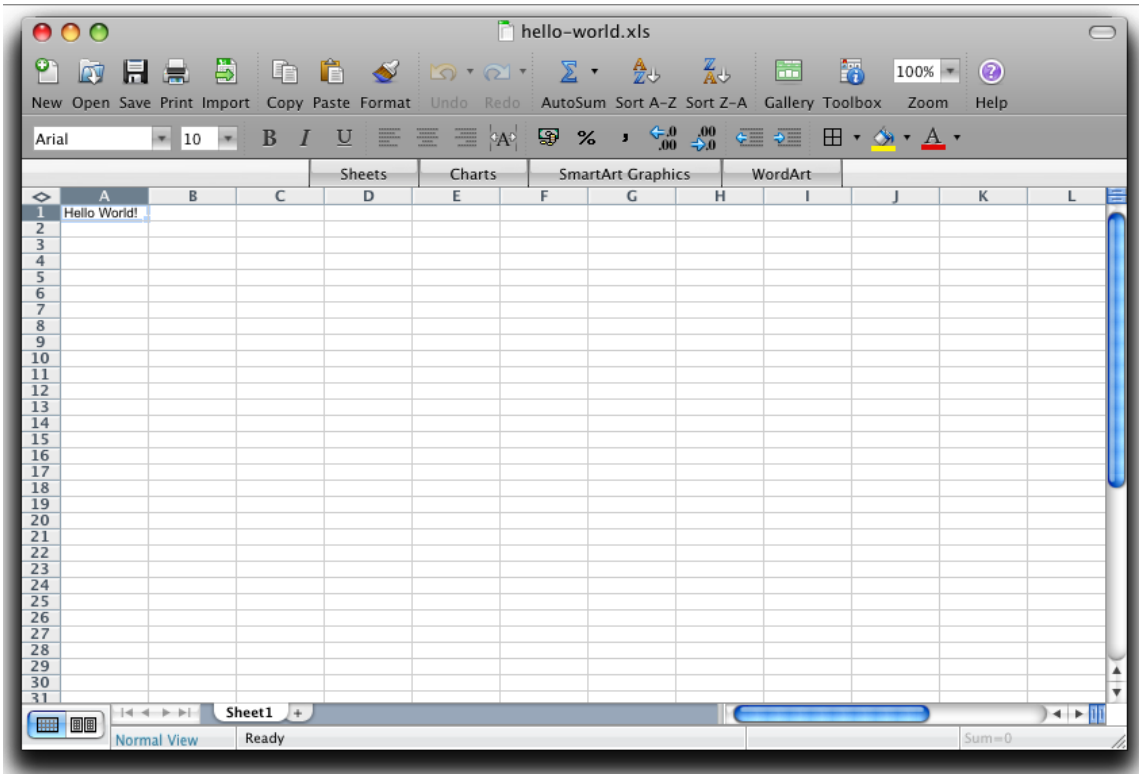
Let's do a minimal "Hello World" script. We'll need to take care of any imports, initialize a Workbook object, create a Worksheet within the workbook, then write some text. Here's how.

1.4.1 Surpass

```
1  require 'rubygems'
2  require 'surpass'
3
4  book = Workbook.new
5  sheet = book.add_sheet
6
7  sheet.write(0, 0, "Hello World!")
8
9  book.save("content/examples/hello-world.xls")
10
```

1.4.2 Result

And, here's how it looks.


hello-world.xls
New
Open
Save
Print
Import
Copy
Paste
Format
Undo
Redo
AutoSum
Sort A–Z
Sort Z–A
Gallery
Toolbox
100%
Zoom
Help
Arial
10
Sheets
Charts
SmartArt Graphics
WordArt
Hello World!
Sheet1
Normal View
Ready
Sum=0

Chapter 2

Writing Data

2.1 Autoformatting

Autoformats are number formats which are automatically applied to Dates, Floats and similar classes. To have autoformats applied, then pass true as the style parameter to the write function.

Here is the relevant code from row.rb:

```
111       when TrueClass # Automatically apply a nice numeric format.
112         case label
113         when DateTime, Time
114           style = @parent_wb.styles.default_datetime_style
115         when Date
116           style = @parent_wb.styles.default_date_style
117         when Float
118           style = @parent_wb.styles.default_float_style
119         else
120           style = @parent_wb.styles.default_style
121         end
122
```

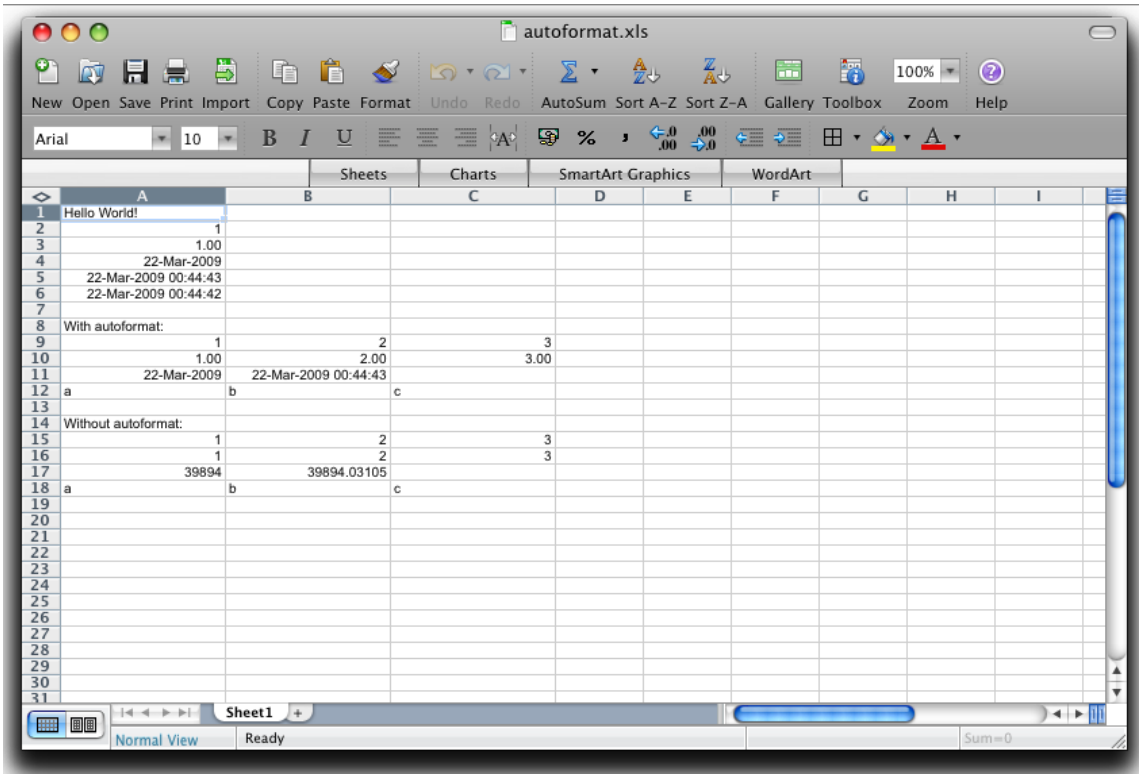
And here are the default formats being defined in style.rb:

```
100     def default_date_style
101       @default_date_style ||= StyleFormat.new(:number_format_string => 'dd-mmm-yyyy')
102     end
103
104     def default_datetime_style
105       @default_datetime_style ||= StyleFormat.new(:number_format_string => 'dd-mmm-yyyy hh:mm:ss')
106     end
107
108     def default_float_style
109       @default_float_style ||= StyleFormat.new(:number_format_string => '#,##0.00')
110     end
111
```

If you use any of the array-writing methods, then autoformatting will be applied by default. To override this behaviour you can pass your own StyleFormat or nil to use the generic default format.

```
1  require 'rubygems'
2  require 'surpass'
3
4  book = Workbook.new(__FILE__.gsub(/rb$/, "xls"))
5  sheet = book.add_sheet
```

```ruby

# Passing true for the style parameter to write will invoke autoformatting.
sheet.write(0, 0, "Hello World!", true)
sheet.write(1, 0, 1, true)
sheet.write(2, 0, 1.0, true)
sheet.write(3, 0, Date.today, true)
sheet.write(4, 0, DateTime.now, true)
sheet.write(5, 0, Time.now, true)

array_of_arrays = [
  [1, 2, 3],
  [1.0, 2.0, 3.0],
  [Date.today, DateTime.now],
  %w{a b c}
]

# Writing arrays will automatically autoformat.
sheet.write(7, 0, "With autoformat:")
sheet.write_arrays(8, 0, array_of_arrays)

# Unless you specify your own format, or nil for a generic default.
sheet.write(13, 0, "Without autoformat:")
sheet.write_arrays(14, 0, array_of_arrays, nil)

sheet.set_column_widths(0..2, 20)

book.save

```



Chapter 3

Formatting

3.1 Reference

There is a command line tool included with Surpass which provides some useful reference data:

```
surpass-info -h

Usage: surpass-info [options]
    -c, --colors                     List available colors
    -h, --help                       Show this message
```

And since you are running this on the command line, you can save or pipe the output to other commands:

```
surpass-info -c | grep green

bright-green
dark-green
green
light-green
olive-green
sea-green
```

3.2 Formatting

The StyleFormat class is a wrapper for the various types of formatting you can apply to a cell. StyleFormat has attributes:

- number_format_string
- font
- alignment
- borders
- pattern
- protection

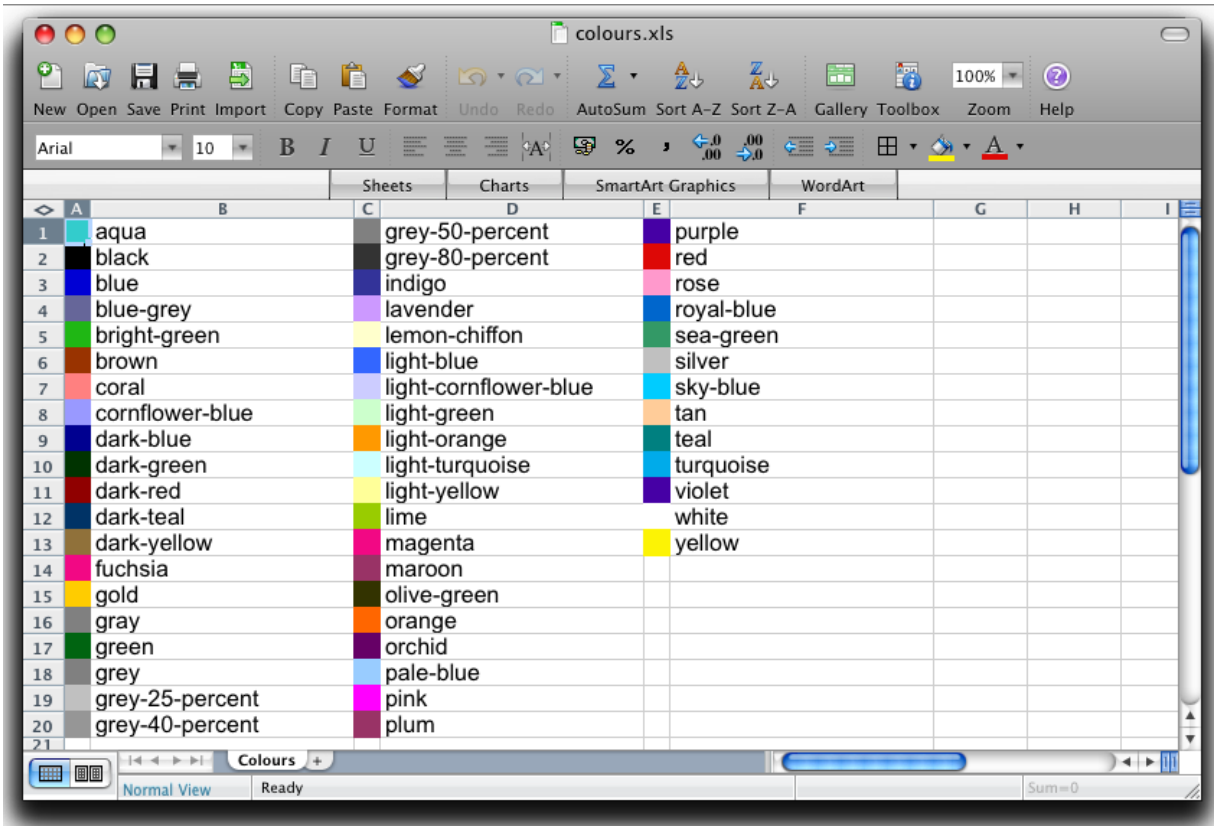
Each of these attributes has a corresponding class, and you can look in lib/formatting.rb for the source.

There are two basic ways to set formatting options. You can pass a hash with formatting options when you initialize a new StyleFormat instance, or you can set individual attributes of the formatting classes. You can combine both approaches. Both of these are demonstrated in the examples in this section.

3.2.1 Specifying Colours

Here is a list of available colours:

```
aqua                    grey-25-percent         orchid
black                   grey-40-percent         pale-blue
blue                    grey-50-percent         pink
blue-grey               grey-80-percent         plum
bright-green            indigo                  purple
brown                   lavender                red
coral                   lemon-chiffon           rose
cornflower-blue         light-blue              royal-blue
dark-blue               light-cornflower-blue   sea-green
dark-green              light-green             silver
dark-red                light-orange            sky-blue
dark-teal               light-turquoise         tan
dark-yellow             light-yellow            teal
fuchsia                 lime                    turquoise
gold                    magenta                 violet
gray                    maroon                  white
green                   olive-green             yellow
grey                    orange
```



You can refer to any of these colours by name.

3.2.2 Border Formats

Here is a list of available border line types:

```
none
thin
```

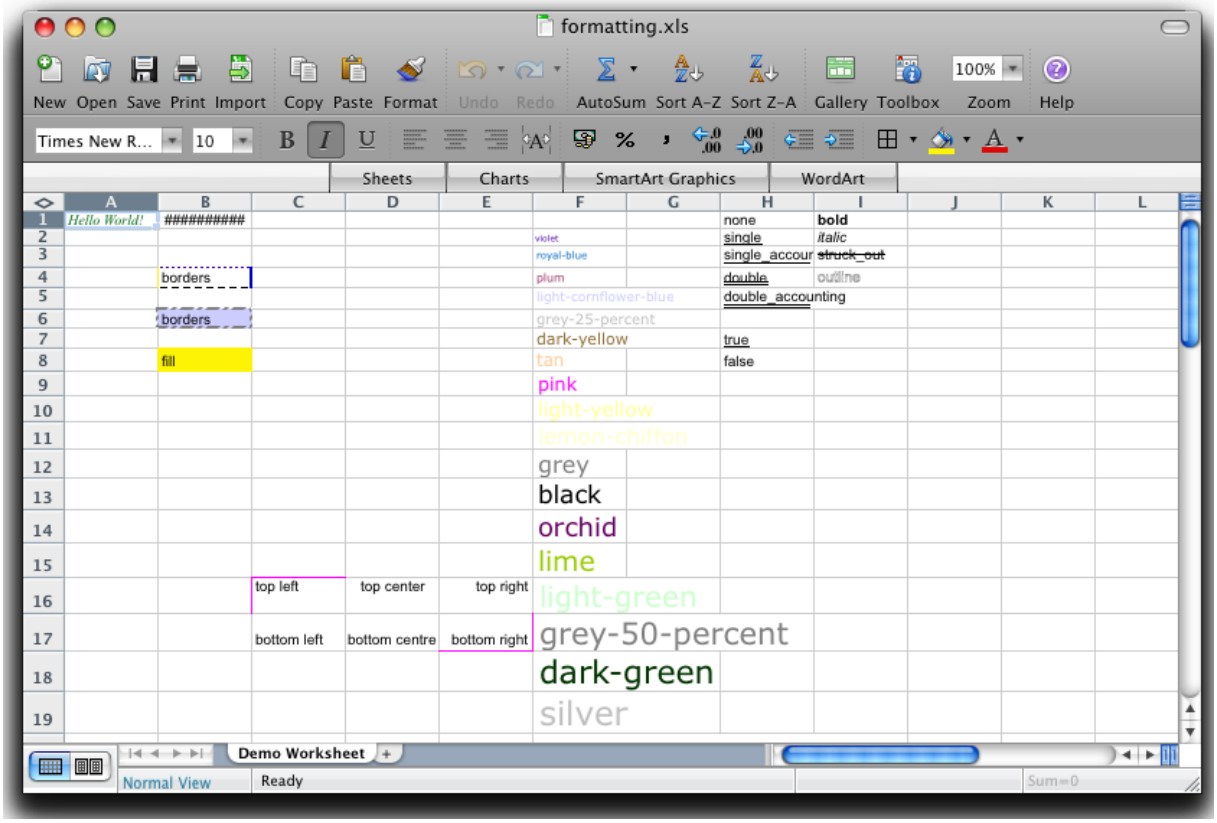
```
medium
dashed
dotted
thick
double
hair
medium-dashed
thin-dash-dotted
medium-dash-dotted
thin-dash-dot-dotted
medium-dash-dot-dotted
slanted-medium-dash-dotted
```

3.2.3 Surpass

```ruby
require 'rubygems'
require 'surpass'

book = Workbook.new(__FILE__.gsub(/rb$/, "xls"))
sheet = book.add_sheet("Demo Worksheet") # You can name your worksheets.

# Let's set up some formatting.

# Remember to use Excel-style formatting directives, not sprintf.
date_format = StyleFormat.new(:number_format_string => "DDDD DD MMM YYYY")

fancy_format = StyleFormat.new(
  :font_name => 'Times New Roman',
  :font_colour => 'green',
  :font_italic => true
)

sheet.write(0, 0, "Hello World!", fancy_format)
sheet.write(0, 1, Date.today, date_format)

# You can also set up formatting by passing attributes directly to the constituents of StyleFormat

# Font colours.
Formatting::COLOURS.keys.each_with_index do |c, i|
  format = StyleFormat.new
  format.font.name = 'Verdana'
  format.font.color = c
  format.font.size = i + 5
  sheet.write(i, 5, c, format)
end

# Font underlining.

[:none, :single, :single_accounting, :double, :double_accounting, nil, true, false].each_with_index
  format = StyleFormat.new
  format.font.underline = u
  sheet.write(i, 7, u.to_s, format)
end

# Font bold, italic, strikethrough, outline are simple booleans.
[:bold, :italic, :struck_out, :outline].each_with_index do |s, i|
```

```
  attribute = ("font_" + s.to_s).to_sym
  sheet.write(i, 8, s.to_s, StyleFormat.new(attribute => true))
end

# Cell alignment.
sheet.write(15, 2, "top left", :text_align => 'top left',
  :border_top => 'pink',
  :border_left => 'pink'
)
sheet.write(15, 3, "top center", :text_align => 'top center')
sheet.write(15, 4, "top right", :text_align => 'top right')
sheet.write(16, 2, "bottom left", :text_align => 'bottom left')
sheet.write(16, 3, "bottom centre", :text_align => 'bottom centre')
sheet.write(16, 4, "bottom right", :text_align => 'bottom right',
  :border_bottom => 'pink',
  :border_right => 'pink'
)


# Borders
sheet.write(3, 1, "borders",
  :border_right => 'medium blue',
  :border_left => 'yellow', # thin by default
  :border_top => 'dotted purple',
  :border_bottom => 'dashed' # black by default
)

# Or the hash-free option.
crazy_border_format = StyleFormat.new
crazy_border_format.borders.all = 'slanted-medium-dash-dotted grey'
crazy_border_format.pattern.fill = 'light-cornflower-blue'

sheet.write(5, 1, "borders", crazy_border_format)

sheet.write(7, 1, "fill", :fill_color => 'yellow')

book.save

```

And, here's how it looks.


formatting.xls
New Open Save Print Import Copy Paste Format Undo Redo AutoSum Sort A-Z Sort Z-A Gallery Toolbox Zoom Help
Times New R... 10
Sheets Charts SmartArt Graphics WordArt
Hello World!
##########
borders
borders
fill
top left
top center
top right
bottom left
bottom centre
bottom right
violet
royal-blue
plum
light-cornflower-blue
grey-25-percent
dark-yellow
tan
pink
light-yellow
lemon-chiffon
grey
black
orchid
lime
light-green
grey-50-percent
dark-green
silver
none
single
single_accour
double
double_accounting
bold
italic
struck_out
outline
true
false
Demo Worksheet
Normal View
Ready
Sum=0

Chapter 4

Saving

Typically, you will call the workbook's save() method to write that workbook to a file. You can pass the filename as an argument to save(), or as an argument to new() when you first instantiate a workbook object.

However, you can also call a workbook's data() method, which gives you direct access to a workbook's binary data. You can write this to a file manually, as in this example:

```
require 'rubygems'
require 'surpass'

book = Workbook.new
sheet = book.add_sheet

sheet.write(0, 0, "Hello World!")

File.open(__FILE__.gsub(/rb$/, "xls"), "w") do |f|
  f.write book.data
end

```

Or, you could use this data as an argument to Rails' send_data method.