Surpass and XLWT Manual

Ana Nelson

March 18, 2009

1 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.

1.1 Surpass

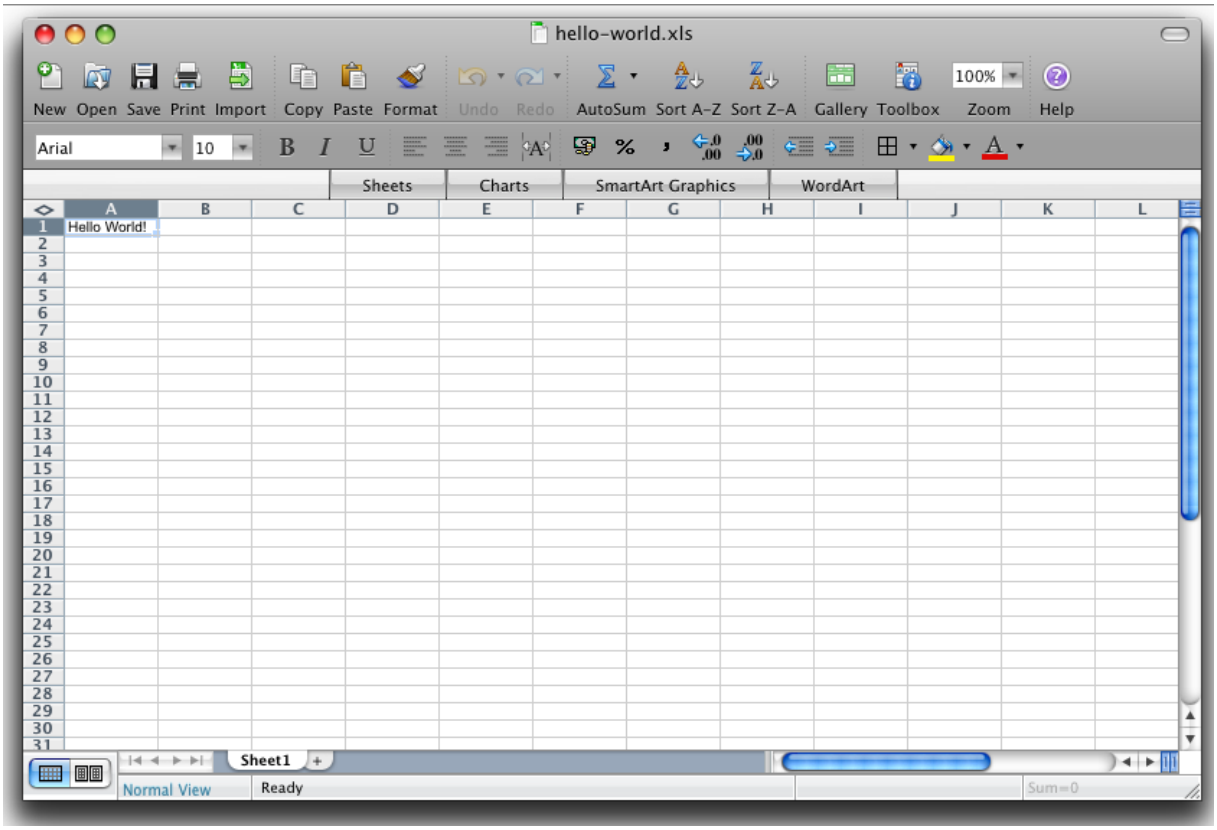
```ruby
require 'rubygems'
require 'surpass'

book = Workbook.new
sheet = book.add_sheet

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

book.save("output/examples/hello-world.xls")

```

1.2 XLWT

```python
from xlwt import *

book = Workbook()
sheet = book.add_sheet("Sheet1")

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

book.save("output/examples/hello-world-python.xls")

```

1.3 Result

And, here's how it looks.



2 A Demo

If you are the sort of person who prefers to learn by example, here is a rather comprehensive demo showing most of the basics you might want to do.

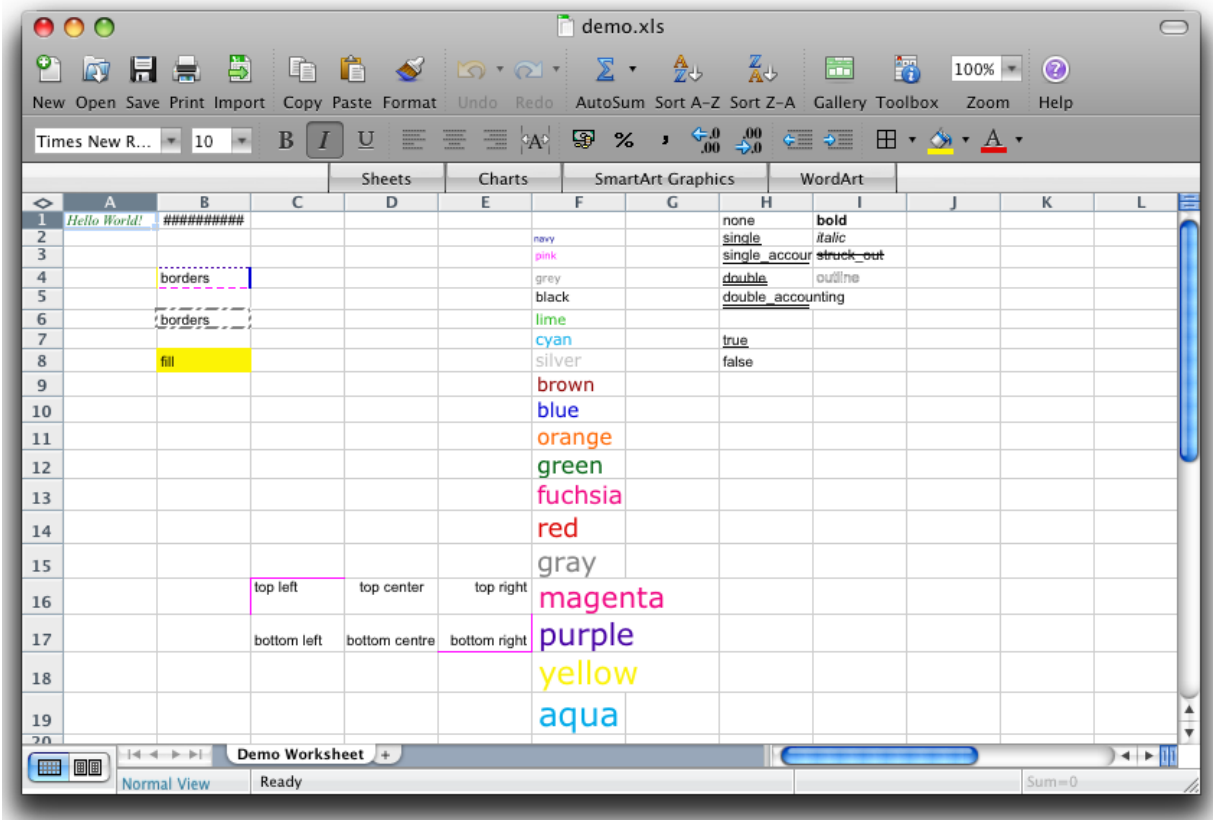
2.1 Surpass

```
require 'rubygems'
require 'surpass'

book = Workbook.new("output/examples/demo.xls") # You can pass a filename here too.
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
```

```ruby

# 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.
# You can pass a hash of style attributes directly to sheet.write, but
# remember this creates a new StyleFormat object each time, so don't
# do this if you are going to re-use a style for multiple cells. If you
# are going to use a format more than once, then create a StyleFormat and
# pass a reference to that.
sheet.write(15, 2, "top left", :text_align => 'top left', :border_top => 'pink', :border_left => 'pi
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_


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

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

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

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

book.save
```

78

And, here's how it looks.



3 Formats

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.