

## 1 Excell controller wrappers

```
"goi/withXL.rb" 1a ≡
# control Excel
<License 1b>
<Required Modules 2b>
<definitions ?>
<classes ?>
module XL
  @@inited = false
  module_function
  <XL definitions 2a, ... >
end
◇
```

```
<License 1b> ≡
# Copyright (C) 2004, Javier Goizueta <javier@goizueta.info>
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
◇
```

Macro referenced in [1ad](#).

```
<constants initialization 1c> ≡
unless @@inited
  WIN32OLE.const_load xl, XL
  @@inited = true
end
◇
```

Macro referenced in [2d](#).

## 2 Tests

```
"withXL_test.rb" 1d ≡

<License 1b>
require 'test/unit/testcase'
require 'test/unit/ui/console/testrunner'
require 'test/unit/testsuite'
<Tests definitions ?>
class Test_withXL < Test::Unit::TestCase
  <Tests ?>
end
Test::Unit::UI::Console::TestRunner.run(Test_withXL)
◇
```

### 3 File names

```

< XL definitions 2a > ≡
  def absFileName(rel_fn)
    dir_base = File.expand_path($0,Dir.pwd)
    dir_base = File.dirname dir_base
    if File::ALT_SEPARATOR
      rel_fn.gsub!(Regexp.new(Regexp.escape(File::ALT_SEPARATOR)), File::SEPARATOR)
    end
    abs_fn = File.expand_path(rel_fn, dir_base)
    if File::ALT_SEPARATOR
      abs_fn.gsub!(Regexp.new(Regexp.escape(File::SEPARATOR)), File::ALT_SEPARATOR)
    end
    abs_fn
  end
  ◇

```

Macro defined by [2acd](#), [3ab](#), [4](#).  
 Macro referenced in [1a](#).

### 4 Control Excel application

```

< Required Modules 2b > ≡
  require 'win32ole';◇

```

Macro referenced in [1a](#).

```

< XL definitions 2c > ≡
  def withApplication
    begin
      xl = openApplication
      yield xl
    rescue StandardError => err
      puts "ERROR: \n_ (#{err.class}) \n_--_ #{err}"
      print err.backtrace.join("\n")
    ensure
      xl.quit if xl
    end
  end
  ◇

```

Macro defined by [2acd](#), [3ab](#), [4](#).  
 Macro referenced in [1a](#).

```

< XL definitions 2d > ≡
  def openApplication
    xl = WIN32OLE.new('Excel.Application')
    < constants initialization 1c >
    xl
  end
  ◇

```

Macro defined by [2acd](#), [3ab](#), [4](#).  
 Macro referenced in [1a](#).

## 5 Access a workbook

```

< XL definitions 3a > ≡
  def withWorkBook(filename, new=false, numsheets=1)
    withApplication do |xl|
      wb = openWorkBook(xl, filename, new, numsheets)
      yield xl, wb
      wb.save if new
      wb.close
    end
  end
  ◇

```

Macro defined by [2acd](#), [3ab](#), [4](#).  
 Macro referenced in [1a](#).

```

< XL definitions 3b > ≡
  def openWorkBook(xl, filename, new=:existing, numsheets=1)
    wb = nil

    # compatibility with old version
    if new==false
      new = :existing
    elsif new==true
      new = :new
    end

    if xl
      case new
      when :new, :unnamed
        orig_numsh = xl.sheetsInNewWorkbook
        xl.sheetsInNewWorkbook = numsheets if numsheets!=orig_numsh
        wb = xl.workbooks.add
        if new!=:unnamed
          filename = xl.getSaveAsFilename unless filename
          wb.saveAs filename
        end
        xl.sheetsInNewWorkbook = orig_numsh
      when :existing
        wb = xl.workbooks.add filename
      end
    end
    wb
  end
  ◇

```

Macro defined by [2acd](#), [3ab](#), [4](#).  
 Macro referenced in [1a](#).

```

⟨XL definitions 4⟩ ≡
  @@col_digits = "ABCDEFGHJKLMNOPQRSTUVWXYZ"
  @@col_base = @@col_digits.size
  def col_ref(i, mode=:rel) # i =0,...
    rf = ''

    # rf = 'A'; i.times{ rf.succ!}; rf
    begin
      rf = @@col_digits[(i % @@col_base), 1] + rf
      i /=_@@col_base
      cnt = i>0
      i -= 1 if cnt
    end while cnt

    rf = '$'+rf if mode==:abs
    rf
  end
  def row_ref(i, mode=:rel) # i =0,...
    rf = (i+1).to_s
    rf = '$'+rf if mode==:abs
    rf
  end
  def ref(cl,rw,mode_cl=:rel,mode_rw=:rel)
    col_ref(cl,mode_cl)+row_ref(rw,mode_rw)
  end
  def rref(rf1,rf2)
    "#{rf1}:#{rf2}"
  end
  def col_index(rf)
    rf = rf[1..-1] if rf[0,1]== '$'
    i = 0
    while rf && rf.size>0
      i *= @@col_base
      i += @@col_digits.index(rf[0,1])+1
      rf = rf[1..-1]
    end
    i-1
  end
  def row_index(rf)
    rf = rf[1..-1] if rf[0,1]== '$'
    i = 0
    if rf
      i = rf.to_i-1
    end
    i
  end
  def parse_ref(rf)
    sheet = nil
    row1 = col1 = nil
    row1_mode = col1_mode = nil
    row2 = col2 = nil
    row2_mode = col2_mode = nil
    md = /\A(?: (.+) !)? (\$?) ([A-Z]+) (\$?) (\d+) (?: : (\$?) ([A-Z]+) (\$?) (\d+)) ?\Z/.match(rf)
    if md
      sheet = md[1]
      col1_mode = md[2]== '$' ? :abs : :rel
      col1 = col_index(md[3])
      row1_mode = md[4]== '$' ? :abs : :rel
      row1 = row_index(md[5])
      if md[7]
        col2_mode = md[6]== '$' ? :abs : :rel
        col2 = col_index(md[7])
        row2_mode = md[8]== '$' ? :abs : :rel
        row2 = row_index(md[9])
      end
    end
  end
  def
    [sheet, col1, row1, col1_mode, row1_mode, col2, row2, col2_mode, row2_mode]
  end
  ◇

```

## 6 Índices

### 6.1 Archivos

"goi/withXL.rb" Defined by [1a](#).

"withXL\_test.rb" Defined by [1d](#).

### 6.2 Fragmentos

⟨ License [1b](#) ⟩ Referenced in [1ad](#).

⟨ Required Modules [2b](#) ⟩ Referenced in [1a](#).

⟨ Tests definitions ? ⟩ Referenced in [1d](#).

⟨ Tests ? ⟩ Referenced in [1d](#).

⟨ XL definitions [2acd](#), [3ab](#), [4](#) ⟩ Referenced in [1a](#).

⟨ classes ? ⟩ Referenced in [1a](#).

⟨ constants initialization [1c](#) ⟩ Referenced in [2d](#).

⟨ definitions ? ⟩ Referenced in [1a](#).

### 6.3 Identificadores