

# [blog.elevenworks.com](http://blog.elevenworks.com)

Jon Lipsky's Weblog

« [RelativeLayout](#)

Aug 02

## [Cocoa calculator example using MacRuby](#)

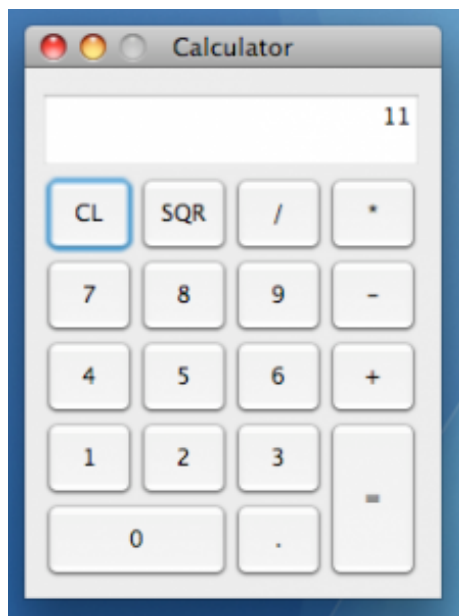
[Cocoa](#), [Mac](#), [MacRuby](#), [Ruby](#)

I intended to write about the Java Swing layout manager that I now use as my next post; however I got sidetracked on a small experiment that I wanted to post about first.

Almost 5 years ago to the day, I had posted on my previous blog about how to create a Cocoa based calculator using the now defunct Cocoa-Java bridge. At that time, I was working on a commercial tool called XMLFace that allowed you to define your GUI (view) using XML, write your controller code in Java and then run your application (assuming you used standard controls) using either Swing or SWT. The Cocoa/Java calculator example was my first experiments into researching the viability of adding Cocoa bindings to XMLFace. Luckily, due to the lifespan and robustness (or lack thereof) of the Cocoa/Java bridge, I didn't spend much more time on it.

Recently, I've been intrigued with the idea that I could use Ruby to write cross platform applications instead of Java; where the controller logic is in Ruby, and the view, while implemented in Ruby, is platform specific so that the application feels (or actually is) native. Also, this would give me an opportunity to learn a new language after working almost exclusively with Java and C# in the recent years.

I decided last week, waiting in an airport for a delayed flight to arrive, to try to implement that Calculator example that I had written using the Cocoa/Java bridge using [MacRuby](#). It took me the duration of the flight, and a few hours at home afterwards to get it up and running (mostly due to my lack of Ruby experience and due to an initial lack of understand of how MacRuby wraps the Cocoa framework); however I did get it working rather painlessly. Here is the obligatory screenshot:



If your curious, you can download the source code to the original Java source code here: [Calculator.java](#).

The source code to the ruby version is below; however if you want to download it, you can download it

here: [Calculator.rb](#).

**Note:** I've only run this on the version of MacRuby available in their SVN trunk. I'm not sure if it works with the downloadable version (0.2) available on the website.

#### PLAIN TEXT

```
1. # This is an example of how to use MacRuby to programatica
2. # build a Cocoa application without using InterfaceBuilder
3. #
4. # Author:: Jon Lipsky (jon.lipsky [at] elevenworks.com)
5. # Copyright:: Copyright (c) 2008, Jon Lipsky
6. # License:: Apache Software License
7.
8. framework 'Cocoa'
9.
10. class Calculator
11.
12.   # Define the constants for the operations
13.   NONE = -1
14.   ADDITION = 0
15.   SUBTRACTION = 1
16.   MULTIPLICATION = 2
17.   DIVISION = 3
18.
19.   # Define the constants for the window and buttons sizes
20.   WINDOW_WIDTH = 220
21.   WINDOW_HEIGHT = 280
22.   BUTTON_COLUMNS = 4
23.   BUTTON_ROWS = 6
24.   BUTTON_SPACING = 2
25.   WINDOW_MARGIN = 8
26.   BUTTON_WIDTH = (WINDOW_WIDTH - (WINDOW_MARGIN * 2) - ((B
27.   BUTTON_HEIGHT = (WINDOW_HEIGHT - (WINDOW_MARGIN * 2) - (
28.
29.   # Define other constants
30.   MAX_PRECISION = 10
31.
32.   def initialize
33.     @editing = false
34.     @result = 0.0
35.     @entered_number = 0.0
36.     @fractional_digits = 0
37.     @operation = 0
38.     @decimal_separator = false
39.
40.     @application = NSApplication.sharedApplication()
41.     self.setup
42.     @application.run
43.   end
44.
45.   def setup
46.     # Create the text field to display the current value
47.     x = BUTTON_SPACING + WINDOW_MARGIN
48.     y = BUTTON_SPACING * 6 + (BUTTON_HEIGHT * 5) + WINDOW_
49.
50.     width = BUTTON_WIDTH * 4 + (BUTTON_SPACING * 3)
51.     height = BUTTON_HEIGHT - (BUTTON_SPACING * 2)
52.
53.     @display = NSTextField.alloc.initWithFrame [x, y, width,
54.     @display.setEditable(false)
55.     @display.setBezeled(true)
56.     @display.setBezelStyle(NSBezelBorder)
```

```
57.     @display.setBackground(true)
58.     @display.setAlignment(NSRightTextAlignment)
59.
60.     # Create the buttons
61.     buttons = Array.new
62.
63.     buttons <<create_digit_button(0, 0, 0)
64.     buttons <<create_button(2, 0, 1, 1, "handle_decimal:",
65.     buttons <<create_button(3, 0, 1, 2, "handle_equals:",
66.
67.     buttons <<create_digit_button(0, 1, 1)
68.     buttons <<create_digit_button(1, 1, 2)
69.     buttons <<create_digit_button(2, 1, 3)
70.
71.     buttons <<create_digit_button(0, 2, 4)
72.     buttons <<create_digit_button(1, 2, 5)
73.     buttons <<create_digit_button(2, 2, 6)
74.     buttons <<create_button(3, 2, 1, 1, "handle_operation:",
75.
76.     buttons <<create_digit_button(0, 3, 7)
77.     buttons <<create_digit_button(1, 3, 8)
78.     buttons <<create_digit_button(2, 3, 9)
79.     buttons <<create_button(3, 3, 1, 1, "handle_operation:",
80.
81.     buttons <<create_button(0, 4, 1, 1, "handle_clear:", "
82.     buttons <<create_button(1, 4, 1, 1, "handle_square_roo
83.     buttons <<create_button(2, 4, 1, 1, "handle_operation:",
84.     buttons <<create_button(3, 4, 1, 1, "handle_operation:",
85.
86.     # Create the window
87.     win = NSWindow.alloc.initWithContentRect [100,100,WIND
88.     win.setTitle("Calculator")
89.     win.setDelegate(self)
90.
91.     # Add the controls to the window
92.     buttons.each do |button|
93.         win.contentView.addSubview(button)
94.     end
95.     win.contentView.addSubview(@display)
96.
97.     win.center
98.     win.makeKeyAndOrderFront(win)
99. end
100.
101. # Create a digit button for the calculator panel
102. #
103. # aCellX - The X position of the button in the grid
104. # aCellY - The Y position of the button in the grid
105. def create_digit_button(aCellX, aCellY, aDigit)
106.     create_button(aCellX, aCellY, aDigit==0?2:1, 1, "handl
107. end
108.
109. # Create a a NSButton positioned in the grid. The grid
110. # lower left and extends right and up. (0,0) is in the
111. # hand corner.
112. #
113. # aCellX           - The X position of the button in the
114. # aCellY           - The Y position of the button in the
115. # aColumns         - The number of columns the button sho
116. # aRows            - The number of rows the button should
117. # aMethod          - The method name to call when the act
118. # aTitle           - The title of the button
119. # aKeyEquivalent  - The key equivalent of the button.
120. # aTag             - The tag of the button
```

```
121.     def create_button(aCellX, aCellY, aColumns, aRows, aMeth
122.
123.         x = (BUTTON_SPACING * (aCellX + 1)) + (BUTTON_WIDTH *
124.         y = (BUTTON_SPACING * (aCellY + 1)) + (BUTTON_HEIGHT *
125.
126.         button_width = BUTTON_WIDTH * aColumns + (BUTTON_SPACI
127.         button_height = BUTTON_HEIGHT * aRows + (BUTTON_SPACIN
128.
129.         vButton = NSButton.alloc.initWithFrame [x, y, button_w
130.
131.         vButton.setTarget(self)
132.         vButton.setAction(aMethod)
133.         vButton.setTitle(aTitle)
134.         vButton.setKeyEquivalent(aKeyEquivalent)
135.         vButton.setTag(aTag)
136.         vButton.setButtonType(NSMomentaryPushInButton)
137.         vButton.setImagePosition(NSNoImage)
138.         vButton.setBezelStyle(NSRegularSquareBezelStyle)
139.
140.         return vButton
141.     end
142.
143.     def windowWillClose(a_notification)
144.         Process.exit
145.     end
146.
147.     # Handle that the decimal button was pressed
148.     #
149.     # @param aButton The button that was pressed
150.     def handle_decimal(sender)
151.         if !@editing
152.             @entered_number = 0
153.             @decimal_separator = false
154.             @fractional_digits = 0
155.             @editing = true
156.         end
157.         if !@decimal_separator
158.             @decimal_separator = true
159.             update_display(@fractional_digits)
160.         end
161.     end
162.
163.     def handle_equals(sender)
164.         case @operation
165.             when NONE
166.                 @result = @entered_number
167.                 @entered_number = 0
168.                 @decimal_separator = false
169.                 @fractional_digits = 0
170.                 return
171.             when ADDITION
172.                 @result = @result + @entered_number
173.             when SUBTRACTION
174.                 @result = @result - @entered_number
175.             when MULTIPLICATION
176.                 @result = @result * @entered_number
177.             when DIVISION
178.                 if @entered_number == 0
179.                     report_error("Error")
180.                 else
181.                     @result = @result / @entered_number
182.                 end
183.         end
184.     end
```

```
185.     @entered_number = @result
186.     update_display(MAX_PRECISION)
187.
188.     @operation = NONE
189.     @editing = false
190. end
191.
192. # Handle that an operation button was pressed
193. #
194. # sender The button that was pressed
195. def handle_operation(sender)
196.     if (@operation == NONE)
197.         @result = @entered_number
198.         @entered_number = 0
199.         @decimal_separator = false
200.         @fractional_digits = 0
201.         @operation = sender.tag
202.     else
203.         handle_equals(nil)
204.         handle_operation(sender)
205.     end
206. end
207.
208. #Handle that the square root button was pressed
209. #
210. #sender The button that was pressed
211. def handle_square_root(sender)
212.     if (@operation == NONE)
213.         @result = Math.sqrt(@entered_number)
214.         @entered_number = @result
215.
216.         @editing = true
217.         update_display(MAX_PRECISION)
218.         @editing = false
219.         @operation = NONE
220.     else
221.         handle_equals(nil)
222.         handle_square_root(sender)
223.     end
224. end
225.
226. # Handle that a digit button was pressed
227. #
228. # sender The button that was pressed
229. def handle_digit(sender)
230.     digit = sender.tag.to_i
231.     puts("digit pressed: #{digit}")
232.
233.     if !@editing
234.         @entered_number = 0
235.         @decimal_separator = false
236.         @fractional_digits = 0
237.         @editing = true
238.     end
239.
240.     if @decimal_separator
241.         @entered_number = @entered_number + digit * (0.1**(1
242.         @fractional_digits+=1
243.     else
244.         @entered_number = @entered_number * 10 + digit
245.
246.         # Check overflow
247.         if (@entered_number > 10**15)
248.             report_error("Overflow Error")
```

```
249.         return
250.     end
251. end
252.
253.     update_display(@fractional_digits)
254. end
255.
256. # Handle that the clear button was pressed
257. #
258. # sender The button that was pressed
259. def handle_clear(sender)
260.
261.     @result = 0
262.     @enteredNumber = 0
263.     @operation = NONE
264.     @fractionalDigits = 0
265.     @decimalSeparator = false
266.     @editing = false
267.
268.     update_display(@fractional_digits)
269. end
270.
271. #Update the number displayed in the text field
272. def update_display(a_fractional_digits)
273.     if !@editing
274.         value=""
275.     elsif (@decimal_separator) || (a_fractional_digits>0)
276.         if a_fractional_digits == 0
277.             value = "#{@entered_number}."
278.         else
279.             value = @entered_number.to_s
280.         end
281.     else
282.         value = @entered_number.to_i.to_s
283.     end
284.
285.     puts("display=#{value}")
286.     @display.setStringValue(value)
287. end
288.
289. # Report an error to the user
290. #
291. # a_message The message to display in the text input fi
292. def report_error(a_message)
293.     @display.setStringValue(a_message)
294. end
295. end
296.
297. ## Create a new calculator when this ruby script is run
298. Calculator.new
```

Since this process was rather painless, I'm thinking about attempting to port an existing Java Swing application that I wrote for my personal use over to Cocoa + MacRuby. In the end, I think the hardest part will be giving up the nice code completion and language support I get from IntelliJ when doing Java development. For this experiment I used Netbeans as my Ruby editor; however I might give a few others a try. (I did try XCode shortly, but discarded it since I didn't want/need to use Interface Builder.)

I'm sure I'll have more to post on this subject as time goes on.

This entry was posted on Saturday, August 2nd, 2008 at 12:08 am and is filed under [Cocoa](#), [Mac](#), [MacRuby](#), [Ruby](#). You can follow

any responses to this entry through the [RSS 2.0](#) feed. You can [leave a response](#), or [trackback](#) from your own site.

## 2 Responses to “Cocoa calculator example using MacRuby”

1. [Laurent Sansonetti](#) Says:  
[August 2nd, 2008 at 2:10 am](#)

Hi,

# My name is Laurent, I am the MacRuby maintainer.

Very cool example 😊

One thing that came into my mind when reading the code, is that constants may be used instead of class variables, which results in more beautiful code.

Example,

```
NONE = -1  
ADDITION = 0
```

Instead of

```
@@NONE = -1  
@@ADDITION = 0
```

Also, I think that this sample would be better rewritten using HotCocoa. HotCocoa is a sublayer of MacRuby, its aim is to make Cocoa development in Ruby easier, by providing nicer and more powerful APIs designed specifically for Ruby. This project is still under development, though we have a functional implementation and some samples already. You can read more about it here: <http://www.macruby.org/trac/wiki/HotCocoa>

Would you be interested to port it to HotCocoa, for fun? Otherwise I would like to try 😊 Let me know what you think about it.

2. [Jon Lipsky](#) Says:  
[August 2nd, 2008 at 8:08 am](#)

Hi Laurent,

I’ ve updated the example to use constants instead of class variables. Thanks for the suggestion, and you’ re right, it’ s much more readable (For the most part, I ported the code to Ruby as closely as it was written in Java so I could learn the Ruby language features.)

I actually looked into the source of HotCocoa to figure out how to do a few things, so I’ m already (lightly) familiar with it. I’ ll try porting the example to HotCocoa over the weekend, and post the results. (If I get time of course.) I don’ t think it should take long.

Jon...

### Leave a Reply

Name (required)

Mail (will not be published) (required)

Website

•

## • Archives

- [August 2008](#)
- [July 2008](#)
- [March 2006](#)
- [February 2006](#)
- [January 2006](#)
- [September 2005](#)
- [August 2005](#)
- [July 2005](#)
- [June 2005](#)
- [May 2005](#)

## • Categories

- [General](#) (2)
- [Java](#) (17)
  - [Swing](#) (15)
- [Mac](#) (1)
  - [Cocoa](#) (1)
- [Ruby](#) (1)
  - [MacRuby](#) (1)

Promoted by SiteGround [web hosting](#) [Wordpress Themes](#) & [WP templates](#) by [Natty WP Entries \(RSS\)](#) and [Comments \(RSS\)](#).