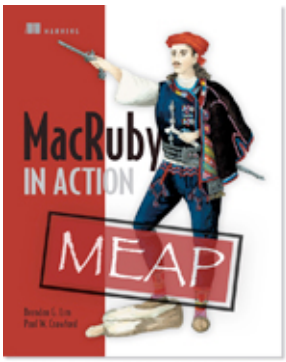
MANNING PUBLICATIONS



MacRuby in Action
By Brendan Lim and Paul Crawford

In this article, based on chapter 3 of MacRuby in Action, the authors Brendan Lim and Paul Crawford show you how to build a nice user interface without writing a single line of code using Interface Builder.

Constructing Your User Interface in Interface Builder

Once you create your project in Xcode, look for the MainMenu.xib file and open it up in Interface Builder. Think about what you would need in your user interface. You're going to need to use a table to list all of your upcoming todo items. You're also going to need a text field for someone to type in what they want to add to their todo list. Then you're going to need a few buttons that allow you to create and mark a todo item as completed. Let's get started!

Building your interface

First, let's bring our main application window into view. Then, let's look in the Library tool and search for the table view. Once you've located it, drag it over to your application window and size the table view to fit the layout. When sizing interface elements, Interface Builder will help us guide and snap them into place. Feel free to size it similar to the way we have it in figure 1. We want to leave a little space at the bottom to be able to add in a text field and a button to add in new todo items.

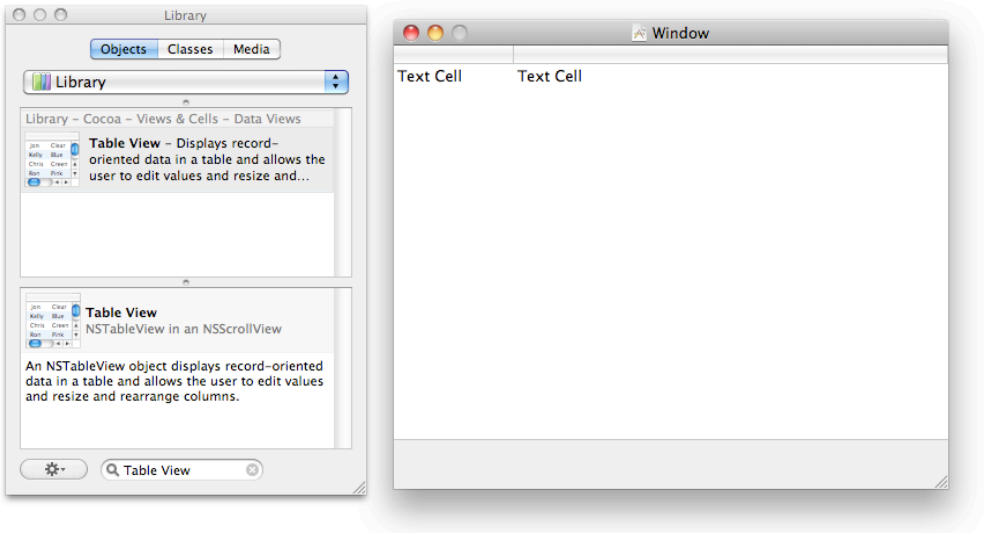


Figure 1 Table view within your window

Now that you have your table view in place, let's make sure it's configured properly. If you bring up your document window and expand the tree item for your Window to expose all of the subviews, you can find two `NSTableColumn` objects for your table view (see figure 2). Click on the first one and view the attributes in the Inspector tool. Within here you can change the title to Status and the identifier to just status. You'll want to do the same thing for the last `NSTableColumn` object and give it the title Title with identifier title.

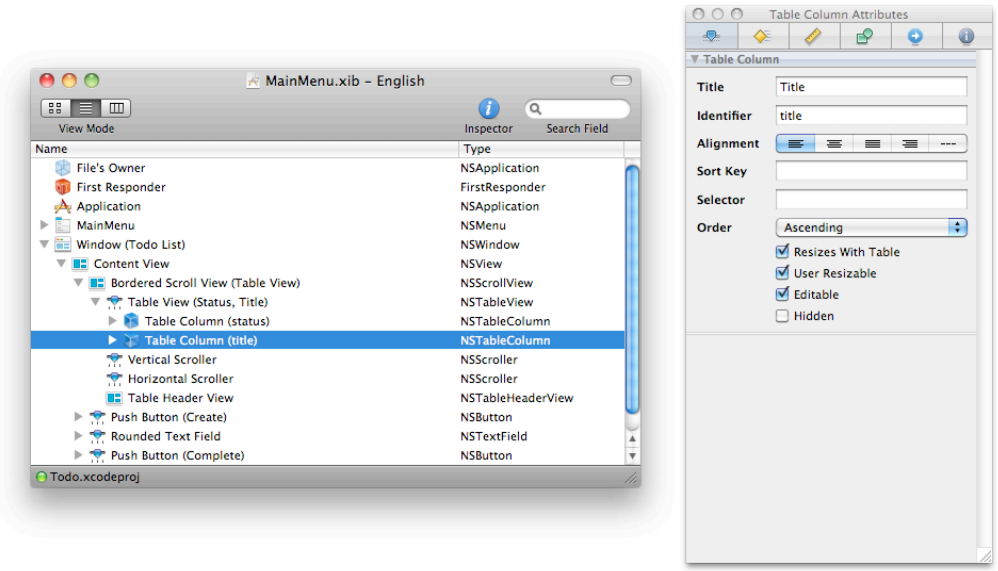


Figure 2 Changing attributes of NSTableColumn

What you should do next is look for a push button (`NSButton`) and a text field (`NSTextField`) in the Library tool. Once you've located these items, you can drag them into the application window. You'll need two push buttons, one to create new tasks and another to mark them as completed. You may resize and position them to your liking.

For the first button, you can click on it and take a look at the attributes within the Inspector. Change the title from Button to Create in the Inspector tool to rename the button. For the second button change the title to Complete.

Now, let's move onto your text field and change some of its attributes to make it look a little better. Click on the text field and bring up the Attributes tab within the Inspector. From here, let's change the border attribute to rounded corners for a little added flair. Next, let's add some text to the placeholder attribute so that you can have some helper text within your text field when it's not focused. Figure 3 shows your application in its current state with buttons and a styled text field in place.

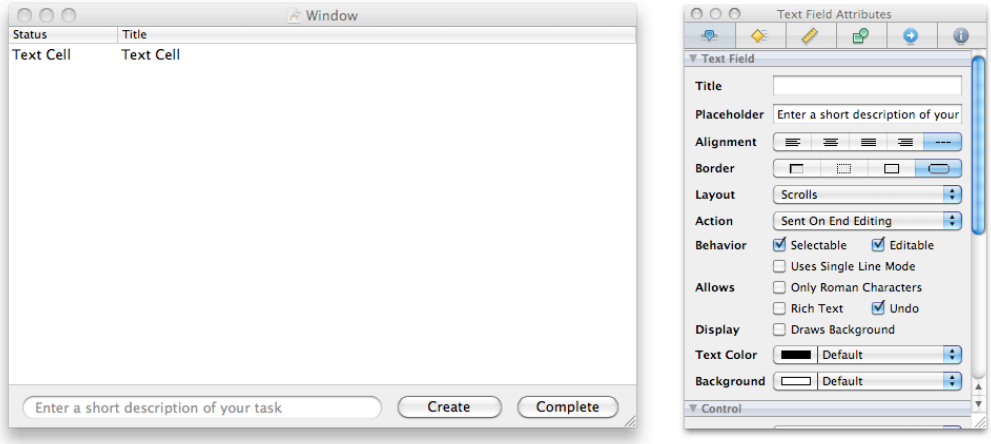


Figure 3 Styling your text field

Next, let's change the title of your actual window. Right now, the application just says "Window" in the title bar. You wouldn't want to ship your application like this. Find your NSWindow object within the document window and change the title attribute to Todo List.

Autosizing interface elements

If you try to run your application right now, it'll look pretty good given its current window size. Although, if you try to maximize or resize the window, you'll notice that all of your elements don't resize with it. Here's where you can take advantage of the autosizing tools that Interface Builder gives us for free.

Let's first set the minimum width and height of your window. Autosizing can only do so much and, if somebody makes your window extremely small, it can definitely have a pretty bad effect. Go to the document window and select your NSWindow and then go to the Size tab within the Inspector tool. Scroll down until you see "Minimum Size" and check that box. Let's click on the button labeled Use Current. This way your window cannot be sized any smaller than its current size.

Let's take a look at the table view that will house all of your todo items. If you look within your document window, you'll notice that this table view is actually housed within an NSScrollView. Let's click on that parent NSScrollView and go to the size tab within the Inspector tool. You should see the autosizing section. So, what exactly do these lines within the autosizing box actually do? The lines outside the inner gray box determine the positioning and spacing. Activating the lines within the inner gray box tell Interface Builder which direction this view should be stretched or shrunk when its parent window's size changes.

Within the autosizing box, you want to make sure to enable autosizing for all directions. This view is already anchored to the top left of your window. You want to make sure that it stretches proportionally in all directions when the parent window is resized. Figure 4 shows exactly how I've set the auto-sizing for the NSScrollView.

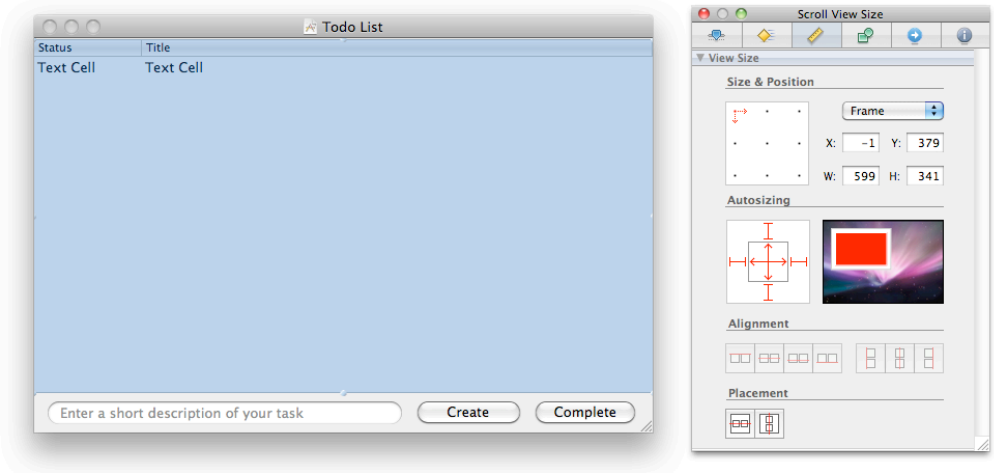


Figure 4 Autosizing the NSScrollView

Next, let's setup the autosizing for your text field. You don't want it to stretch vertically but horizontally. For positioning, let's enable the directions on the left, right, and bottom outside of the inner gray box. Within the inner gray box, let's enable the arrows that point to the left and right since you only want it to stretch horizontally.

Lastly, for your two buttons, you just want to retain its positioning when your main window is resized. You don't want your buttons to get any bigger, in any direction. All you need to do is enable the positioning outside of the inner gray box. Let's choose the bottom and right direction. This will ensure that your buttons stay on the bottom right position of your window at all times. You can see what I set for the autosizing for these buttons in figure 5.

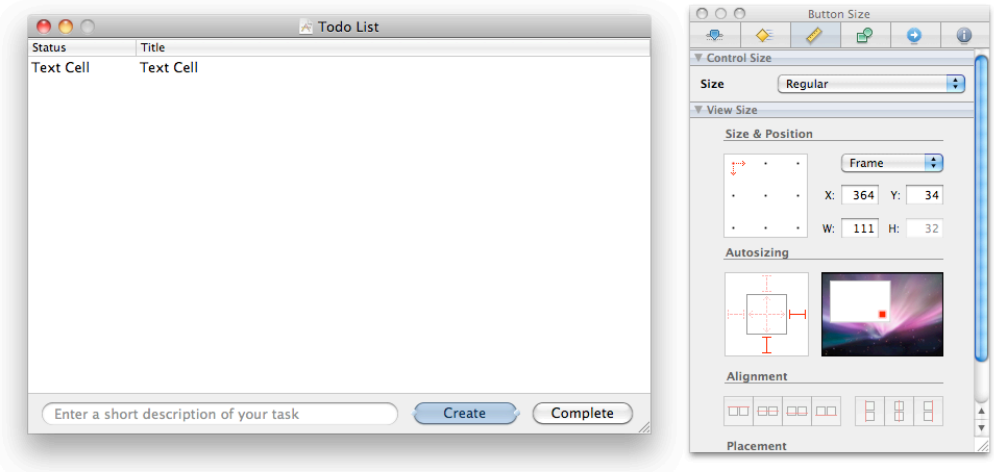


Figure 5 Autosize positioning of your buttons

Your main application window is now ready to go. Launch the simulator within Interface Builder to test out your autosizing changes. The interface elements should now resize properly when you resize your main window. Pretty slick how it does this automatically for us, right? The best part about this is that you've written zero lines of code!

Customizing the menu

The last piece of the user interface you need to change is your menu bar. When you have the application running, on the very top where it has File, Edit, and so on, it says NewApplication. Let's go back into Interface Builder and

double-click on your MainMenu (NSMenu) within your document window. It's rather easy to change these items. Click on the top-level menu titled NewApplication and change the title to Todo List within the Inspector. You can also double-click on these items to rename them.

You'll also notice that, when you clicked on this menu item, it expanded its drop-down menu. Let's do the same and rename all menu items that have NewApplication within it. The items you need to change are the About, Hide, and Quit menu items. There's also a menu item within the Help section that you should change as well. Figure 6 shows what your menu should end up looking like.

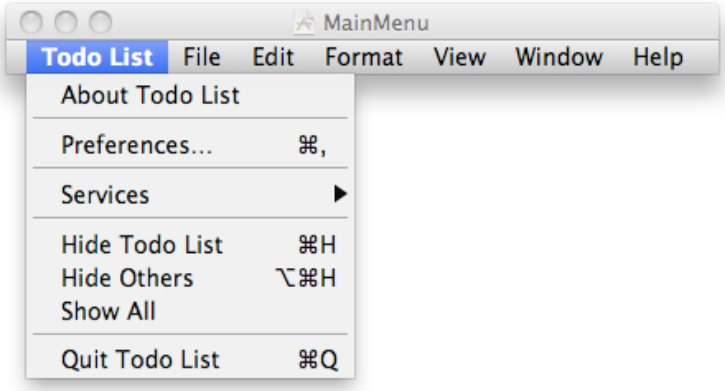


Figure 6 Todo List's menu bar

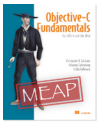
Summary

You've just created a pretty nice user interface without writing a single line of code using Interface Builder! Interface Builder makes this all so easy for us.

Here are some other Manning titles you might be interested in:

Ruby in Practice
Jeremy McAnally and Assaf Arkin

The Well-Grounded Rubyist
Covering Ruby 1.9
David Black



Objective-C Fundamentals
For iOS 4 and the iPad
Christopher K. Fairbairn, Johannes Fahrenkrug, and Collin Ruffenach

Last updated: February 26, 2011