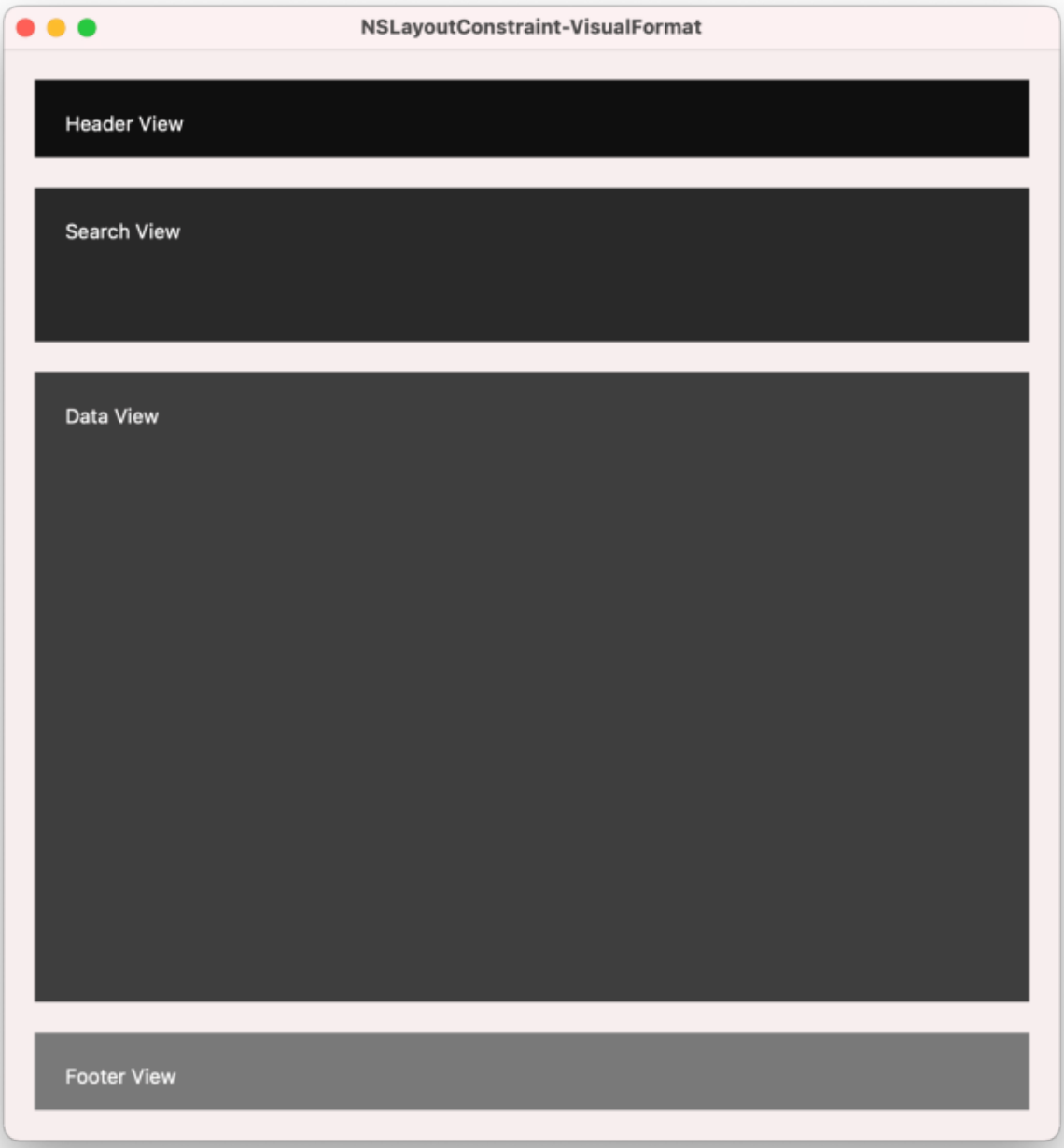
# Swift NSLayoutConstraint constraintsWithVisualFormat – Sample Code

ddas.tech/swift-nslayoutconstraint-constraintswithvisualformat-sample-code/

September 5, 2022

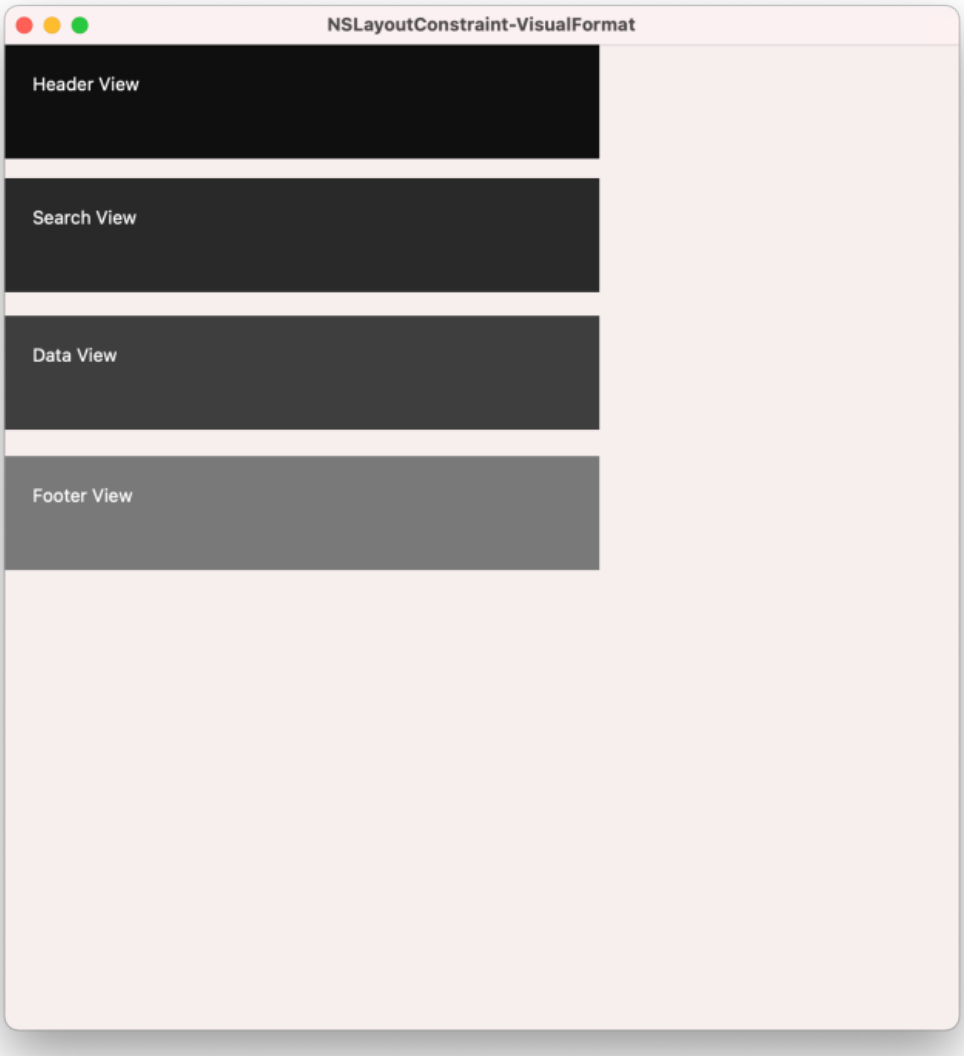



In this post we will apply constraints to 4 views in a NSWindow using **constraintsWithVisualFormat programmatically.**

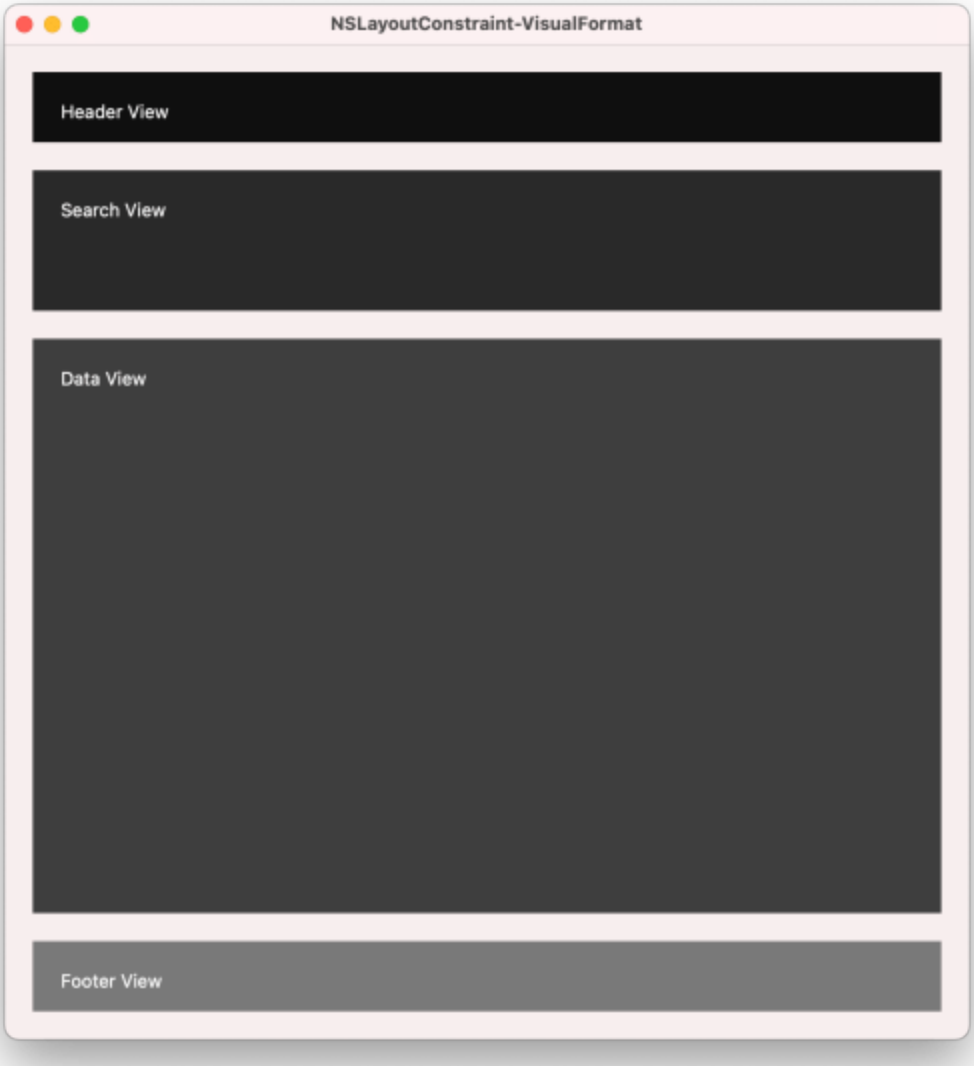
Note: The autoLayOut checkbox is switched off in the XIB and all the constraints will be added programmatically

In the below sample the following behaviour is desired

- Lead Colored View – Header and is of constant height- width is resizable as the window resizes
- Tungsten Colored View – Search Section is of constant height – width of the green view increases as the window resizes
- Iron Colored View – Data View is of dynamic width and height and increases as the screen resizes.
- Steel Colored View – Footer section is of fixed height and dynamic width
- The screen has a minimum width and height that is derived from the sections




**Before applying the constraints**




**After applying the constraints**

We have intentionally kept the views at vertical & horizontal spacing to understand the behavior

Lets start with defining our custom colors (this step is not mandatory) and one can use default colors available in the NSColor Class

```
public extension NSColor{
    static let lead = NSColor(calibratedRed: 13.0/255.0, green: 13.0/255.0, blue:
13.0/255.0, alpha: 1.0)
    static let tungsten = NSColor(calibratedRed: 31.0/255.0, green: 31.0/255.0, blue:
31.0/255.0, alpha: 1.0)
    static let iron = NSColor(calibratedRed: 47.0/255.0, green: 47.0/255.0, blue:
47.0/255.0, alpha: 1.0)
    static let steel = NSColor(calibratedRed: 102.0/255.0, green: 102.0/255.0, blue:
102.0/255.0, alpha: 1.0)
}
```

Now lets define 4 view classes corresponding to HeaderView, Search View, Data View and Footer View

```
//Custom views for the 4 views depicted in 4 different colors
class HeaderView: NSView {
    override func draw(_ dirtyRect: NSRect) {
        super.draw(dirtyRect)
        NSColor.lead.setFill()
        dirtyRect.fill()
    }
}

class SearchView: NSView {
    override func draw(_ dirtyRect: NSRect) {
        super.draw(dirtyRect)
        NSColor.tungsten.setFill()
        dirtyRect.fill()
    }
}

class DataView: NSView {
    override func draw(_ dirtyRect: NSRect) {
        super.draw(dirtyRect)
        NSColor.iron.setFill()
        dirtyRect.fill()
    }
}

class FooterView: NSView {
    override func draw(_ dirtyRect: NSRect) {
        super.draw(dirtyRect)
        NSColor.steel.setFill()
        dirtyRect.fill()
    }
}
```

Finally lets implement the function in our AppDelegate class. You could replicate this class in your view controller or window controller

```
//  AppDelegate.swift
//  NSLayoutConstraint-VisualFormat
//  Created by Debasis Das on 6/17/22.

import Cocoa

@main
class AppDelegate: NSObject, NSApplicationDelegate {

    @IBOutlet var window: NSWindow!
    @IBOutlet weak var headerView: HeaderView!
    @IBOutlet weak var searchView: SearchView!
    @IBOutlet weak var dataView: DataView!
    @IBOutlet weak var footerView: FooterView!

    func applicationDidFinishLaunching(_ aNotification: Notification) {
        // Insert code here to initialize your application
    }

    func applicationWillTerminate(_ aNotification: Notification) {
        // Insert code here to tear down your application
    }

    override func awakeFromNib() {
        autolayoutUsingVisualFormat()
    }

    func autolayoutUsingVisualFormat(){

        //translatesAutoresizingMaskIntoConstraints are set to false for the 4 views
        self.headerView.translatesAutoresizingMaskIntoConstraints = false
        self.searchView.translatesAutoresizingMaskIntoConstraints = false
        self.dataView.translatesAutoresizingMaskIntoConstraints = false
        self.footerView.translatesAutoresizingMaskIntoConstraints = false

        let variableBindings =
["headerView":self.headerView,"searchView":self.searchView,"dataView":self.dataView,"
footerView":self.footerView]

        //the header is at a horizontal spacing of 20 from both the edges of the
screen. The minimum width is greater than equal to 300
        let cn1 = NSLayoutConstraint.constraints(withVisualFormat: "H:|-20-
[headerView(>=300)]-20-|", options:.alignAllLastBaseline, metrics: nil, views:
variableBindings as [String : Any])

        //The height of the header is set to a constant 50 and vertical spacing from
the top edge of the screen is 20
        let cn2 = NSLayoutConstraint.constraints(withVisualFormat: "V:|-20-
[headerView(50)]", options:.alignAllLastBaseline, metrics: nil, views:
variableBindings as [String : Any])

        //The search section view has padding of 20 from both the sides of the screen
```

```
        let cn3 = NSLayoutConstraint.constraints(withVisualFormat: "H:|-20-
[searchView]-20-|", options:.alignAllLastBaseline, metrics: nil, views:
variableBindings as [String : Any])

        //there is a gap of 20 vertical spacing between the header view and the
search section and the search section is of fixed height of 100
        let cn4 = NSLayoutConstraint.constraints(withVisualFormat: "V:
[headerView]-20-[searchView(100)]",
options:NSLayoutConstraint.FormatOptions(rawValue: 0), metrics: nil, views:
variableBindings as [String : Any])

        //Data view horizontal spacing is 20 from either sides of the screen
        let cn5 = NSLayoutConstraint.constraints(withVisualFormat: "H:|-20-
[dataView]-20-|", options:.alignAllLastBaseline, metrics: nil, views:
variableBindings as [String : Any])

        //The vertical spacing between the search view and the data view is 20 and
the minimum height of the data section is 300
        let cn6 = NSLayoutConstraint.constraints(withVisualFormat: "V:
[searchView]-20-[dataView(>=300)]",
options:NSLayoutConstraint.FormatOptions(rawValue: 0), metrics: nil, views:
variableBindings as [String : Any])

        //Footer view horizontal spacing is 20 from either sides of the screen
        let cn7 = NSLayoutConstraint.constraints(withVisualFormat: "H:|-20-
[footerView]-20-|", options:.alignAllLastBaseline, metrics: nil, views:
variableBindings as [String : Any])

        //Vertical spacing of 20 between the dataView and the footerView.
        //Footer view is of constant height of 50
        //Footer 20 vertical spacing from the bottom edge of the screen
        let cn8 = NSLayoutConstraint.constraints(withVisualFormat: "V:[dataView]-20-
[footerView(50)]-20-|", options:NSLayoutConstraint.FormatOptions(rawValue: 0),
metrics: nil, views: variableBindings as [String : Any])

        self.window.contentView?.addConstraints(cn1)
        self.window.contentView?.addConstraints(cn2)
        self.window.contentView?.addConstraints(cn3)
        self.window.contentView?.addConstraints(cn4)
        self.window.contentView?.addConstraints(cn5)
        self.window.contentView?.addConstraints(cn6)
        self.window.contentView?.addConstraints(cn7)
        self.window.contentView?.addConstraints(cn8)

    }
}
```

You can download the code from https://github.com/debasisdas1/ddas.tech/tree/main/Swift/NSLayoutConstraint-VisualFormat