Swift NSLayoutConstraint Programmatically

ddas.tech/swift-nslayoutconstraint-programmatically/

September 19, 2022

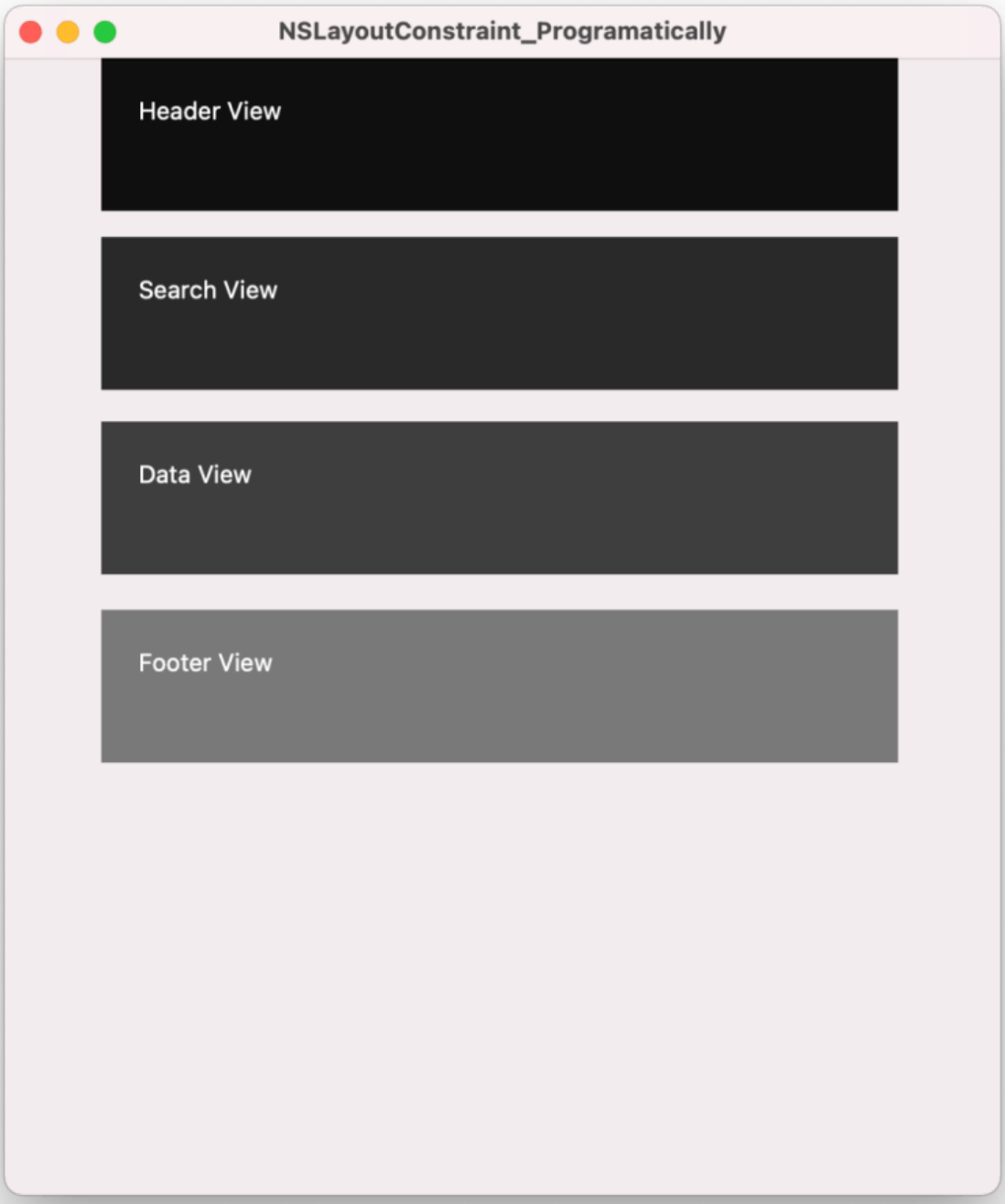
Created By: Debasis Das (Sep 2022)

In this post we will apply constraints to 4 views in a NSWindow using ***swift NSLayoutConstraint*** *programmatically* . This approach is much more tedious and lengthy in comparison to the constraintsWithVisualFormat Approach.

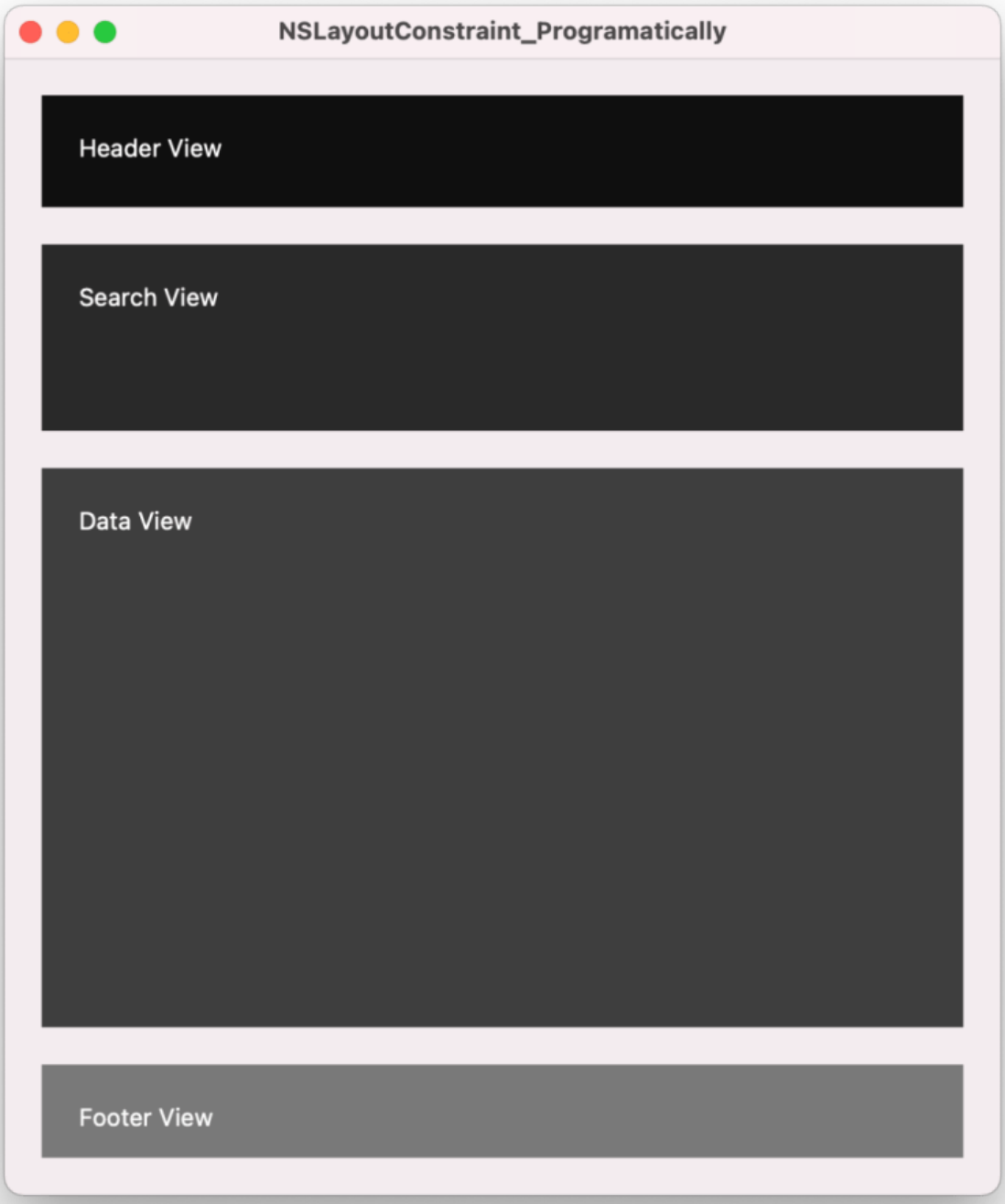
This is a continuation to our previous post for adding NSLayoutConstraint using constraintWithVisualFormats. https://ddas.tech/swift-nslayoutconstraint-constraintswithvisualformat-sample-code/

Its upto the developers discretion to either use the visual format or constraint initializer approach.

Alternatively one can also use interface builder to add constraints to objects in a view.



NSLayoutConstraint – Before applying the constraints



NSLayoutConstraint – After applying the constraints

```
//  AppDelegate.swift
//  NSLayoutConstraint-Programatically
//  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() {
        self.autoLayoutUsingConstraints()
    }

    func autoLayoutUsingConstraints(){
        //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

        if let mainView = self.window.contentView{
            //HeaderView
             //Header = 20 from left edge of screen
            let cn1 = NSLayoutConstraint(item: headerView, attribute: .leading,
relatedBy: .equal, toItem: mainView, attribute: .leading, multiplier: 1.0, constant:
20)
             //Header view trailing end is 20 px from right edge of the screen
             let cn2 = NSLayoutConstraint(item: headerView, attribute: .trailing,
relatedBy: .equal, toItem: mainView, attribute: .trailing, multiplier: 1.0, constant:
-20)
     //Header view height = constant 60
             let cn3 = NSLayoutConstraint(item: headerView, attribute: .height,
relatedBy: .equal, toItem: nil, attribute: .notAnAttribute, multiplier: 1.0,
constant: 60)
     //Header view width greater than or equal to 400
             let cn4 = NSLayoutConstraint(item: headerView, attribute: .width,
relatedBy: .greaterThanOrEqual, toItem: nil, attribute: .notAnAttribute, multiplier:
1.0, constant: 400)
```

```
     //Header view vertical padding from the top edge of the screen = 20
              let cn5 = NSLayoutConstraint(item: headerView, attribute: .top,
relatedBy: .equal, toItem: mainView, attribute: .top, multiplier: 1.0, constant: 20)

              //Search Section
              //search section 20px from left edge of screen
              let cn6 = NSLayoutConstraint(item: searchView, attribute: .leading,
relatedBy: .equal, toItem: mainView, attribute: .leading, multiplier: 1.0, constant:
20)
              //search section 20px from the right edge of the screen
              let cn7 = NSLayoutConstraint(item: searchView, attribute: .trailing,
relatedBy: .equal, toItem: mainView, attribute: .trailing, multiplier: 1.0, constant:
-20)
              //search section height = constant 100
              let cn8 = NSLayoutConstraint(item: searchView, attribute: .height,
relatedBy: .equal, toItem: nil, attribute: .notAnAttribute, multiplier: 1.0,
constant: 100)
              //search section width >=400
              let cn9 = NSLayoutConstraint(item: searchView, attribute: .width,
relatedBy: .greaterThanOrEqual, toItem: nil, attribute: .notAnAttribute, multiplier:
1.0, constant: 400)
              let cn10 = NSLayoutConstraint(item: searchView, attribute: .top,
relatedBy: .equal, toItem: headerView, attribute: .bottom, multiplier: 1.0, constant:
20)


              //Data Section
              let cn11 = NSLayoutConstraint(item: dataView, attribute: .leading,
relatedBy: .equal, toItem: mainView, attribute: .leading, multiplier: 1.0, constant:
20)
              let cn12 = NSLayoutConstraint(item: dataView, attribute: .trailing,
relatedBy: .equal, toItem: mainView, attribute: .trailing, multiplier: 1.0, constant:
-20)
               //Data section height is >=300 and width is >=400
               let cn13 = NSLayoutConstraint(item: dataView, attribute: .height,
relatedBy: .greaterThanOrEqual, toItem: nil, attribute: .notAnAttribute, multiplier:
1.0, constant: 300)
              let cn14 = NSLayoutConstraint(item: dataView, attribute: .width,
relatedBy: .greaterThanOrEqual, toItem: nil, attribute: .notAnAttribute, multiplier:
1.0, constant: 400)
              let cn15 = NSLayoutConstraint(item: dataView, attribute: .top,
relatedBy: .equal, toItem: searchView, attribute: .bottom, multiplier: 1.0, constant:
20)


              //Footer Section
              let cn16 = NSLayoutConstraint(item: footerView, attribute: .leading,
relatedBy: .equal, toItem: mainView, attribute: .leading, multiplier: 1.0, constant:
20)
              let cn17 = NSLayoutConstraint(item: footerView, attribute: .trailing,
relatedBy: .equal, toItem: mainView, attribute: .trailing, multiplier: 1.0, constant:
-20)
```

```
            let cn18 = NSLayoutConstraint(item: footerView, attribute: .height,
relatedBy: .equal, toItem: nil, attribute: .notAnAttribute, multiplier: 1.0,
constant: 50)
            let cn19 = NSLayoutConstraint(item: footerView, attribute: .width,
relatedBy: .greaterThanOrEqual, toItem: nil, attribute: .notAnAttribute, multiplier:
1.0, constant: 400)
            let cn20 = NSLayoutConstraint(item: footerView, attribute: .top,
relatedBy: .equal, toItem: dataView, attribute: .bottom, multiplier: 1.0, constant:
20)

            let cn21 = NSLayoutConstraint(item: footerView, attribute: .bottom,
relatedBy: .equal, toItem: mainView, attribute: .bottom, multiplier: 1.0, constant:
-20)


mainView.addConstraints([cn1,cn2,cn3,cn4,cn5,cn6,cn7,cn8,cn9,cn10,cn11,cn12,cn13,cn14
,cn15,cn16,cn17,cn18,cn19,cn20,cn21])
        }
    }
}

//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()
    }
```

```
}

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)
}
```