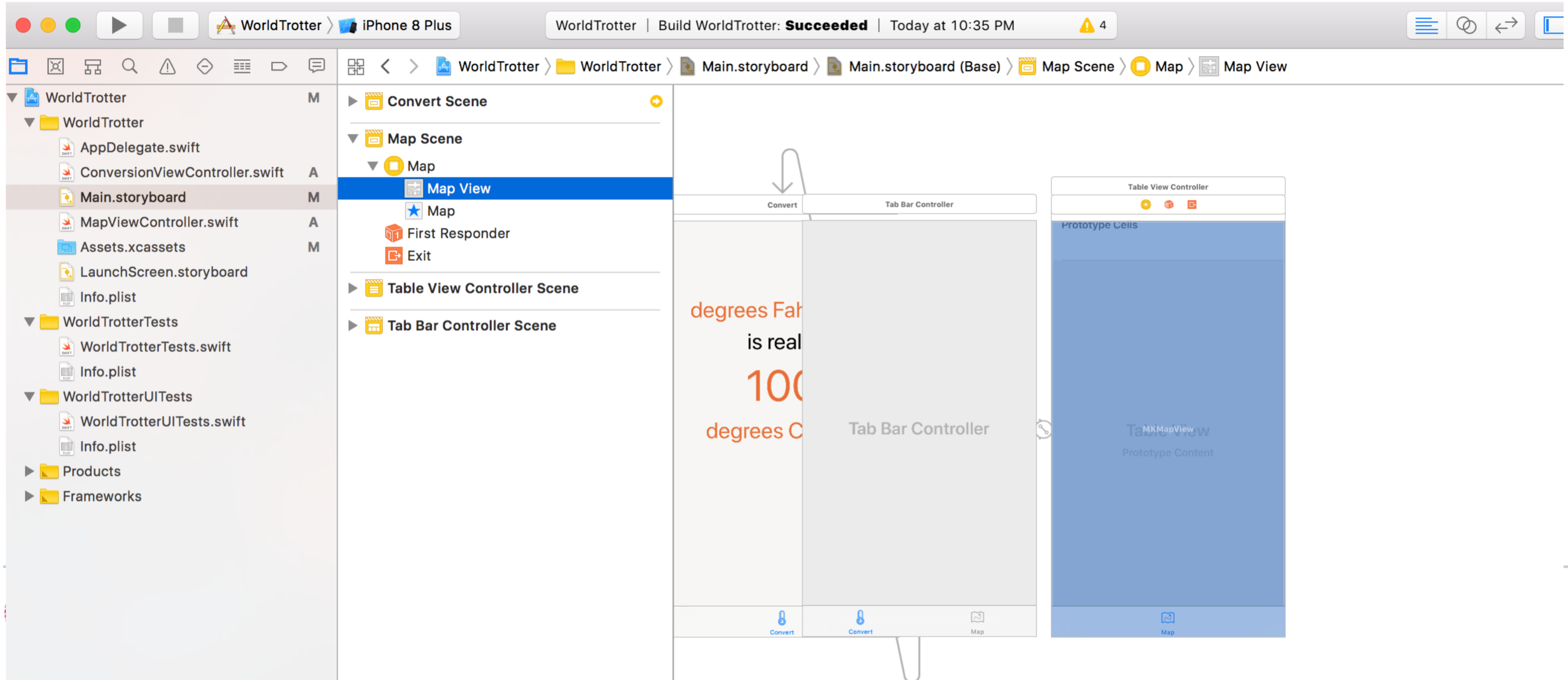
Programming the view for MapViewController

MapViewController view is currently defined in storyboard

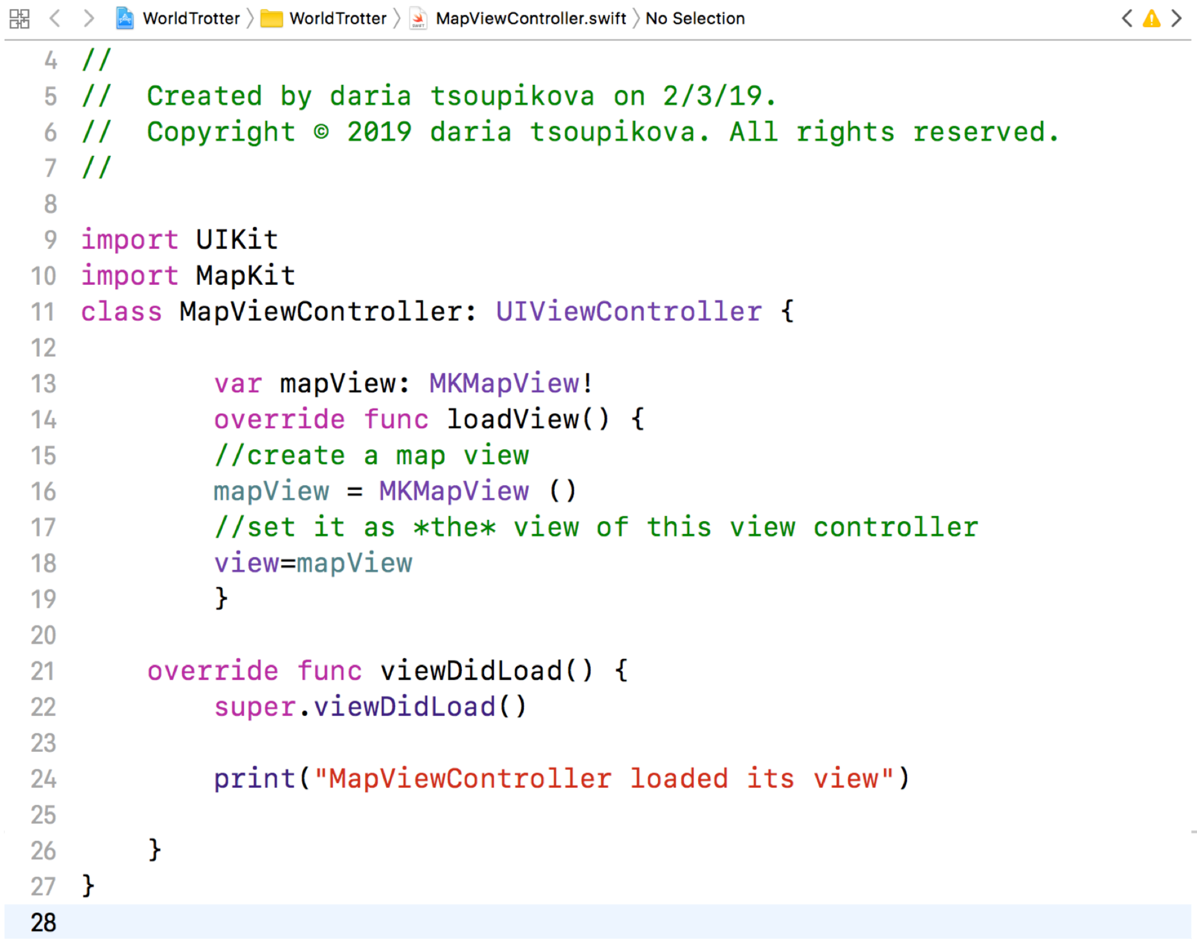
Select map view under Map View Controller and delete it



Open MapViewController.swift

Override loadView() to create instance of MKMapView() to set it

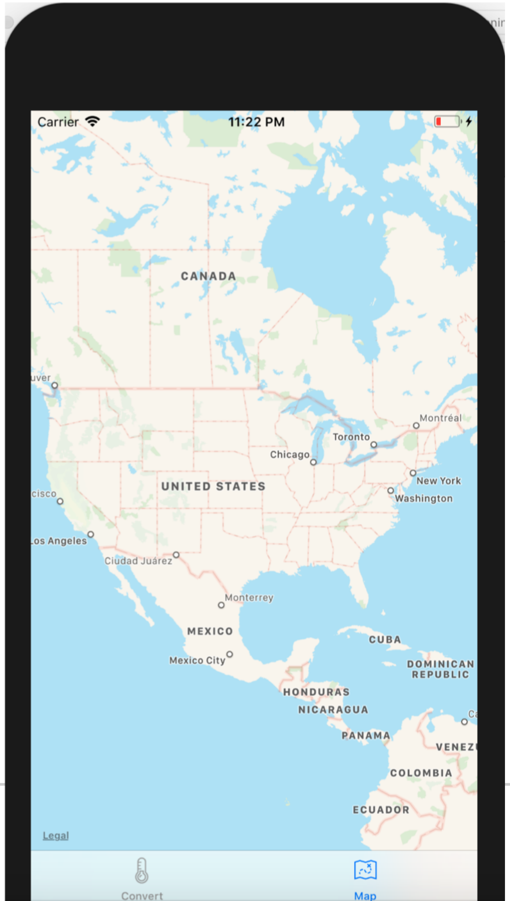
```
import UIKit
import MapKit
class MapViewController: UIViewController {

        var mapView: MKMapView!
        override func loadView() {
        //create a map view
        mapView = MKMapView ()
        //set it as *the* view of this view controller
        view=mapView
        }
    override func viewDidLoad() {
       super.viewDidLoad()
       print("MapViewController loaded its view")
    }
}
```



WorldTrotter 〉 WorldTrotter 〉 MapViewController.swift 〉 No Selection

```
//
//  Created by daria tsoupikova on 2/3/19.
//  Copyright © 2019 daria tsoupikova. All rights reserved.
//

import UIKit
import MapKit
class MapViewController: UIViewController {

        var mapView: MKMapView!
        override func loadView() {
        //create a map view
        mapView = MKMapView ()
        //set it as *the* view of this view controller
        view=mapView
        }

    override func viewDidLoad() {
        super.viewDidLoad()

        print("MapViewController loaded its view")

    }
}

```

Build and run to test. The map is created programmatically at runtime.

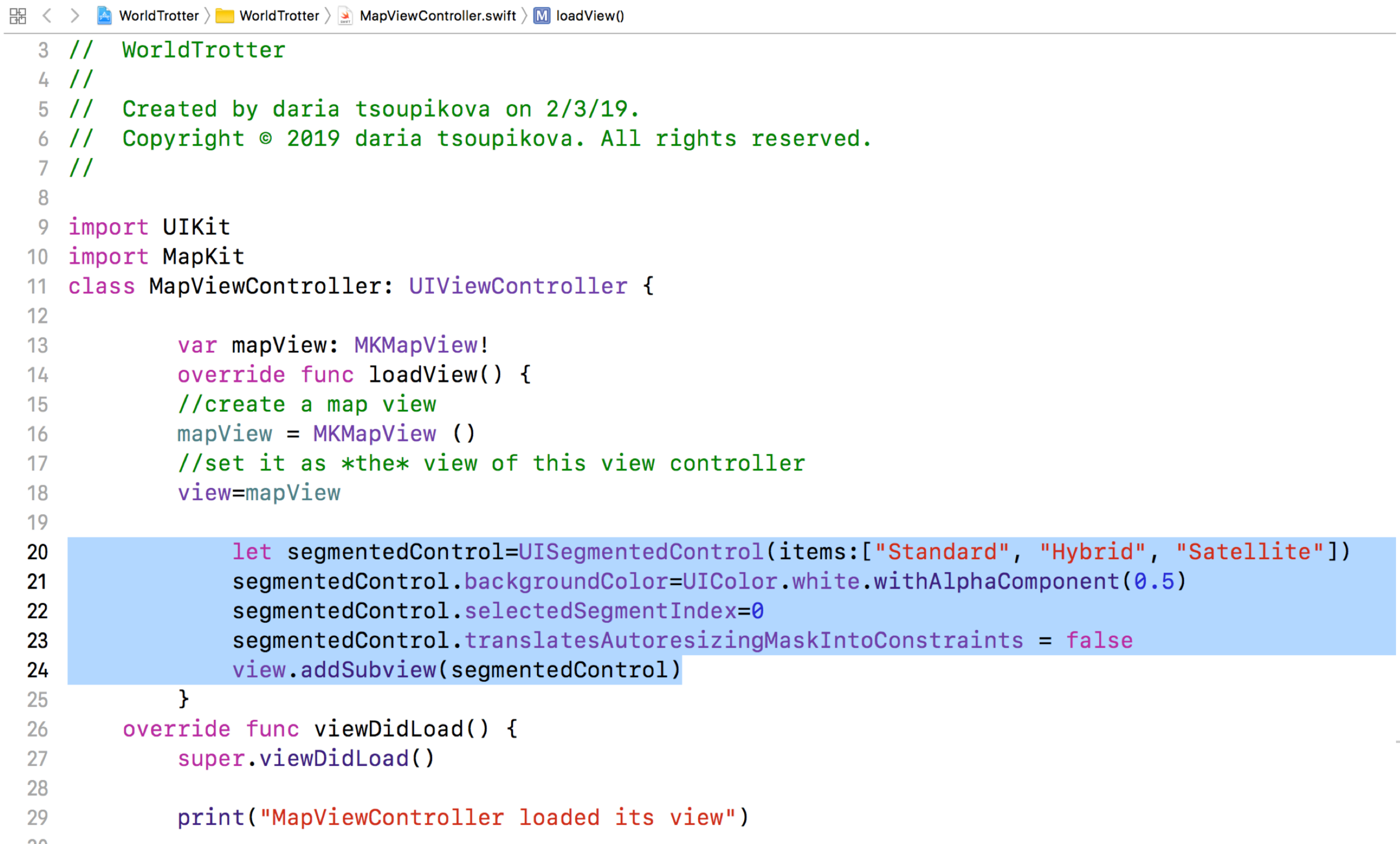


If your views are created in Swift, by programming, you need to constrain them programmatically. MapViewController is created by code.

Add UISegmentControl to MapViewController
It allows the user to choose between a discrete set of options. To allow the user switch between map types: standard, hybrid, and satellite.

Open MapViewController.swift add the following code:

```
view=mapView
 let segmentedControl=UISegmentedControl(items:["Standard", "Hybrid", "Satellite"])
 segmentedControl.backgroundColor=UIColor.white.withAlphaComponent(0.5)
 segmentedControl.selectedSegmentIndex=0
 segmentedControl.translatesAutoresizingMaskIntoConstraints = false
 view.addSubview(segmentedControl)
 }
```



WorldTrotter 〉 WorldTrotter 〉 MapViewController.swift 〉 loadView()

```
//  WorldTrotter
//
//  Created by daria tsoupikova on 2/3/19.
//  Copyright © 2019 daria tsoupikova. All rights reserved.
//

import UIKit
import MapKit
class MapViewController: UIViewController {

        var mapView: MKMapView!
        override func loadView() {
        //create a map view
        mapView = MKMapView ()
        //set it as *the* view of this view controller
        view=mapView

            let segmentedControl=UISegmentedControl(items:["Standard", "Hybrid", "Satellite"])
            segmentedControl.backgroundColor=UIColor.white.withAlphaComponent(0.5)
            segmentedControl.selectedSegmentIndex=0
            segmentedControl.translatesAutoresizingMaskIntoConstraints = false
            view.addSubview(segmentedControl)
        }
    override func viewDidLoad() {
        super.viewDidLoad()

        print("MapViewController loaded its view")
```

Every view has autoresizing mask

Constraints are created by default and added to the view
Can conflict with with IB layout constraints

segmentedControl.translatesAutoresizingMaskIntoConstraints = false

The above command turns off default constraints

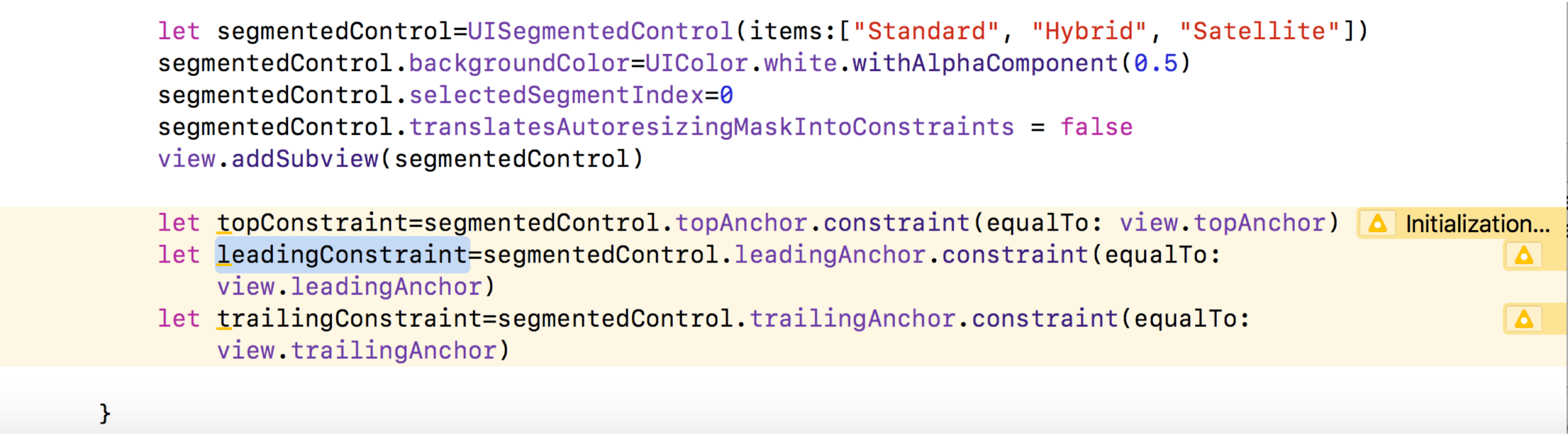
To use AutoLayout in code, use anchors to create constraints
Anchors are properties of the view that correspond to to attributes you constrain to anchor on another view

- The top anchor of segmented control should be equal to the top anchor of its superview
- The leading anchor of segmented control should be equal to the leading anchor of its superview
- The training anchor of the segmented control should be equal to the trailing anchor of its superview

In MapViewControler.swift:
Add method **constraint(equalTo:)** to create a constraint between the two anchors.

```
view.addSubview(segmentedControl)

let topConstraint=segmentedControl.topAnchor.constraint(equalTo: view.topAnchor)

let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo:
view.leadingAnchor)

let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo:
view.trailingAnchor)

        }
```

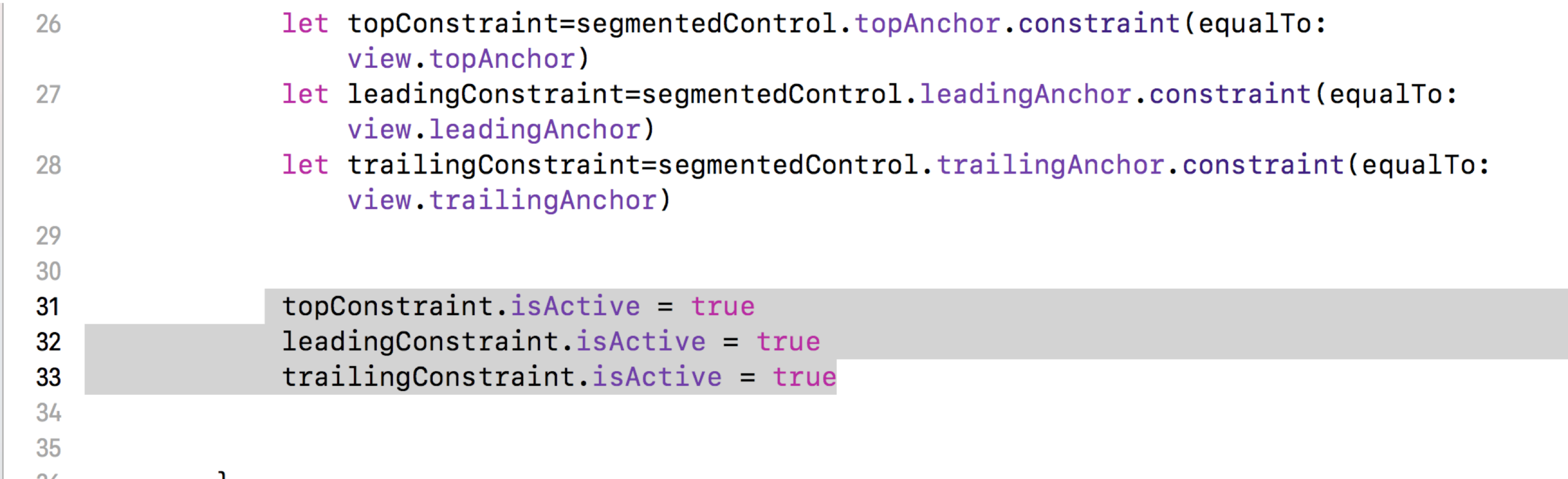
Add method **constraint(equalTo:)** to create a constraint between the two anchors.



```
    let segmentedControl=UISegmentedControl(items:["Standard", "Hybrid", "Satellite"])
    segmentedControl.backgroundColor=UIColor.white.withAlphaComponent(0.5)
    segmentedControl.selectedSegmentIndex=0
    segmentedControl.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(segmentedControl)

    let topConstraint=segmentedControl.topAnchor.constraint(equalTo: view.topAnchor)  ⚠ Initialization...
    let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo:          ⚠
        view.leadingAnchor)
    let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo:        ⚠
        view.trailingAnchor)

}
```

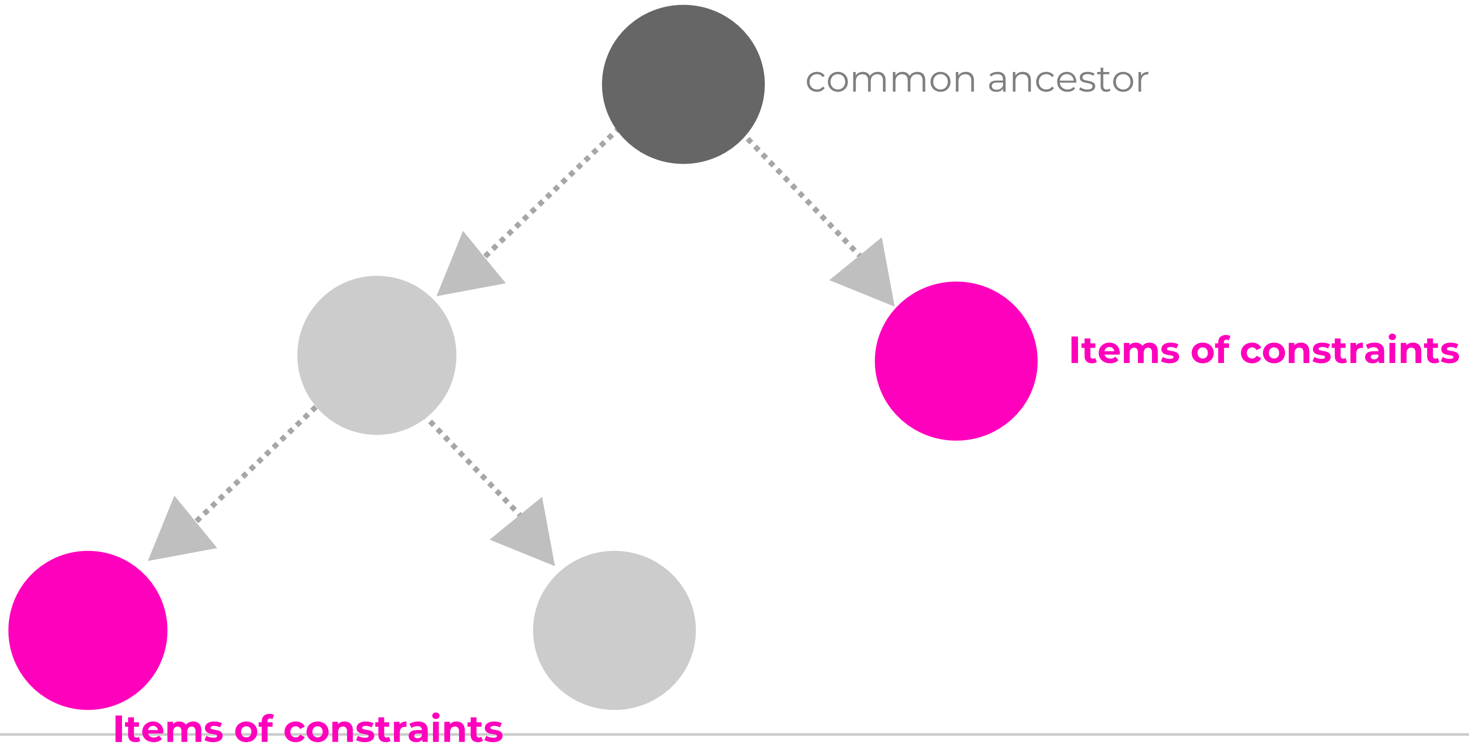
Because constraints are not active, the Xcode issues yellow warnings.
To activate the constraints add:

topConstraint.isActive = true
leadingConstraint.isActive = true
trailingConstraint.isActive = true

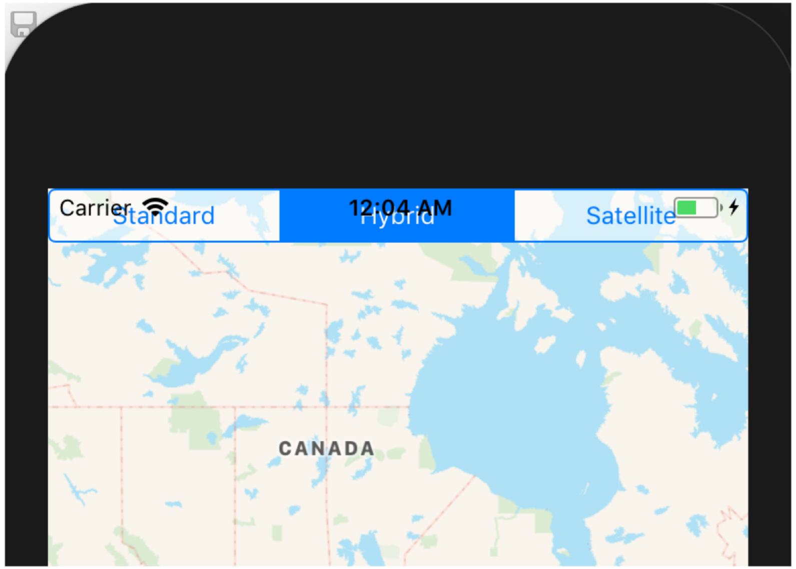


```
        let topConstraint=segmentedControl.topAnchor.constraint(equalTo:
            view.topAnchor)
        let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo:
            view.leadingAnchor)
        let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo:
            view.trailingAnchor)


        topConstraint.isActive = true
        leadingConstraint.isActive = true
        trailingConstraint.isActive = true


    }
```

Constraints are added to the most recent common ancestor for the views associated with the constraint.



The segmented control is overlapping the status bar.

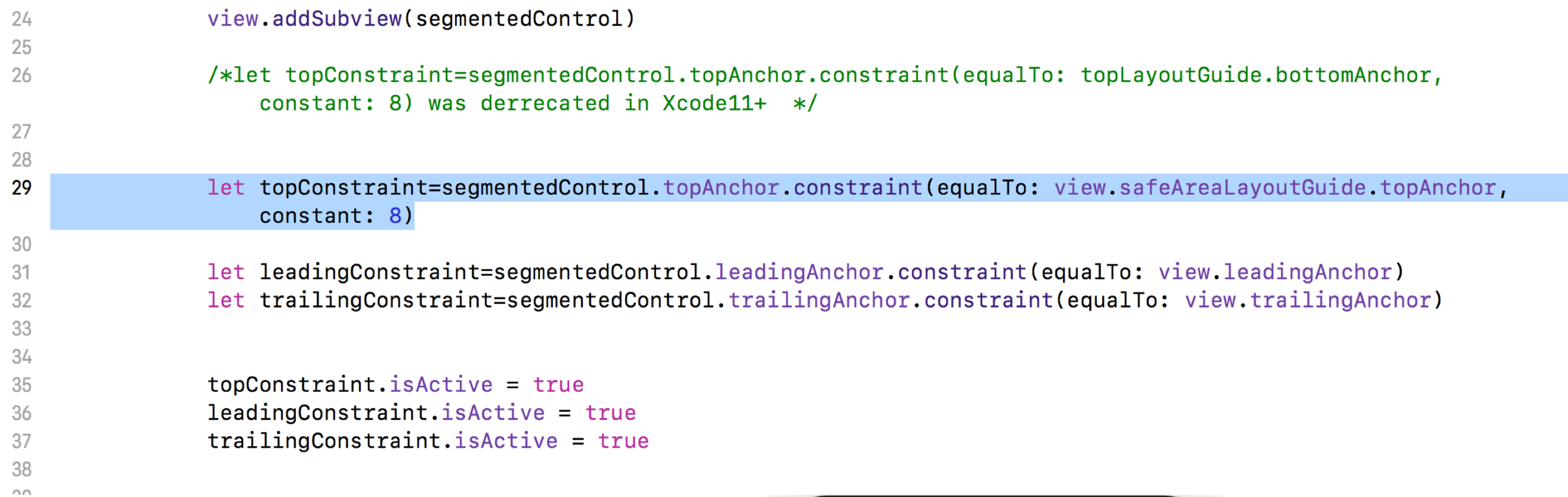


To assist with Layout content use tow methods:
topLayoutGuide and bottomLayoutGuide

topLayoutguide allows the content not to underlap the status bar

bottomLayoutGuide allows not to overlap the bottom of the screen

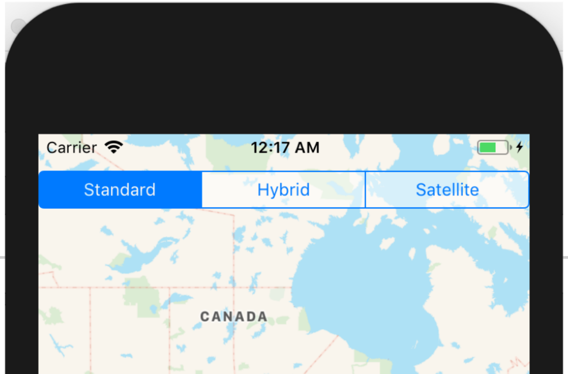
```
let topConstraint=segmentedControl.topAnchor.constraint(equalTo:
view.safeAreaLayoutGuide.topAnchor, constant: 8)
```



```
        view.addSubview(segmentedControl)

        /*let topConstraint=segmentedControl.topAnchor.constraint(equalTo: topLayoutGuide.bottomAnchor,
            constant: 8) was derrecated in Xcode11+  */


        let topConstraint=segmentedControl.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor,
            constant: 8)

        let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo: view.leadingAnchor)
        let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo: view.trailingAnchor)


        topConstraint.isActive = true
        leadingConstraint.isActive = true
        trailingConstraint.isActive = true
```

Build and run

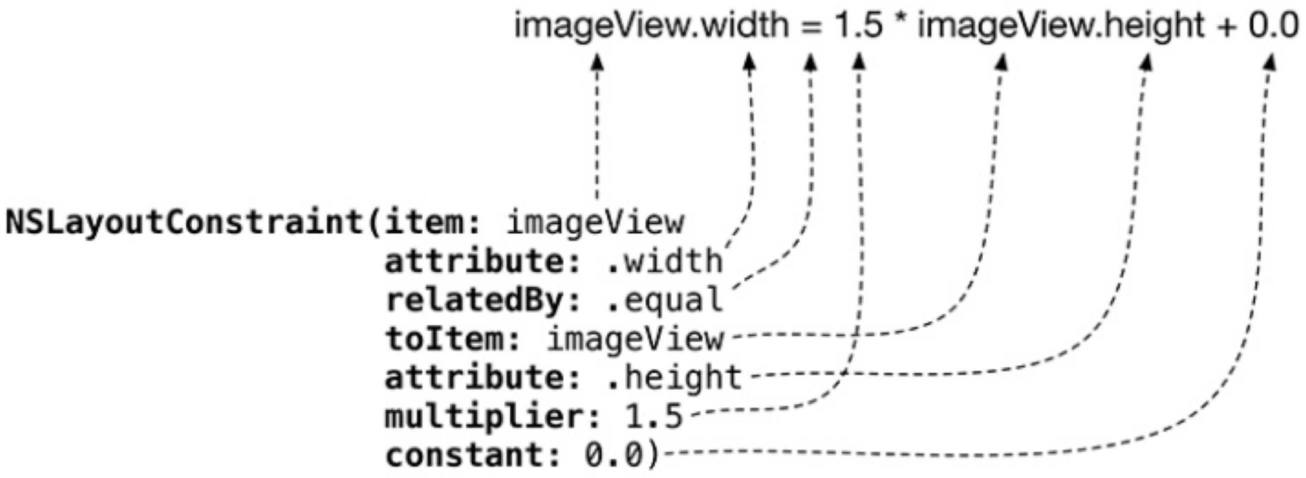
The views adapt to show status bar



The layout attributes are defined as constants in the NSLayoutConstraint class:

- NSLayoutAttribute.left
- NSLayoutAttribute.right
- NSLayoutAttribute.top
- NSLayoutAttribute.bottom
- NSLayoutAttribute.width
- NSLayoutAttribute.height
- NSLayoutAttribute.baseline
- NSLayoutAttribute.centerX
- NSLayoutAttribute.centerY
- NSLayoutAttribute.leading
- NSLayoutAttribute.trailing
  NSLayoutAttribute.lastBaseLine

```
let aspectConstraint  = NSLayoutConstraint (item:imageView,
                         attribute:.width,
                         relatedBy:.equal,
                         toItem:.imageView,
                         attribute: .height,
                        multiplier:1.5,
                        constant:0.0);
```



Common control events:

UIControlEvents: touchDown – a touch down on the control

UIControlEvents: touchUpInside – a touch down followed by touch up within boundaries

UIControlEvents: valueChanged – a touch that changes the value

UIControlEvents: editingChanged – a touch that causes an editing change for UITextFiald

Change the map type when the user taps on a segment.
In MapViewcontroller.swift update loadView() to include .valueChanged event:

```
@objc func mapTypeChanged(_ segControl: UISegmentedControl) {
      switch segControl.selectedSegmentIndex {
      case 0:
         mapView.mapType = .standard
      case 1:
         mapView.mapType = .hybrid
      case 2:
         mapView.mapType = .satellite
      default:
         break
      }
   }
```

Change the map type when the user taps on a segment.
In MapViewcontroller.swift update loadView() to include .valueChanged event:

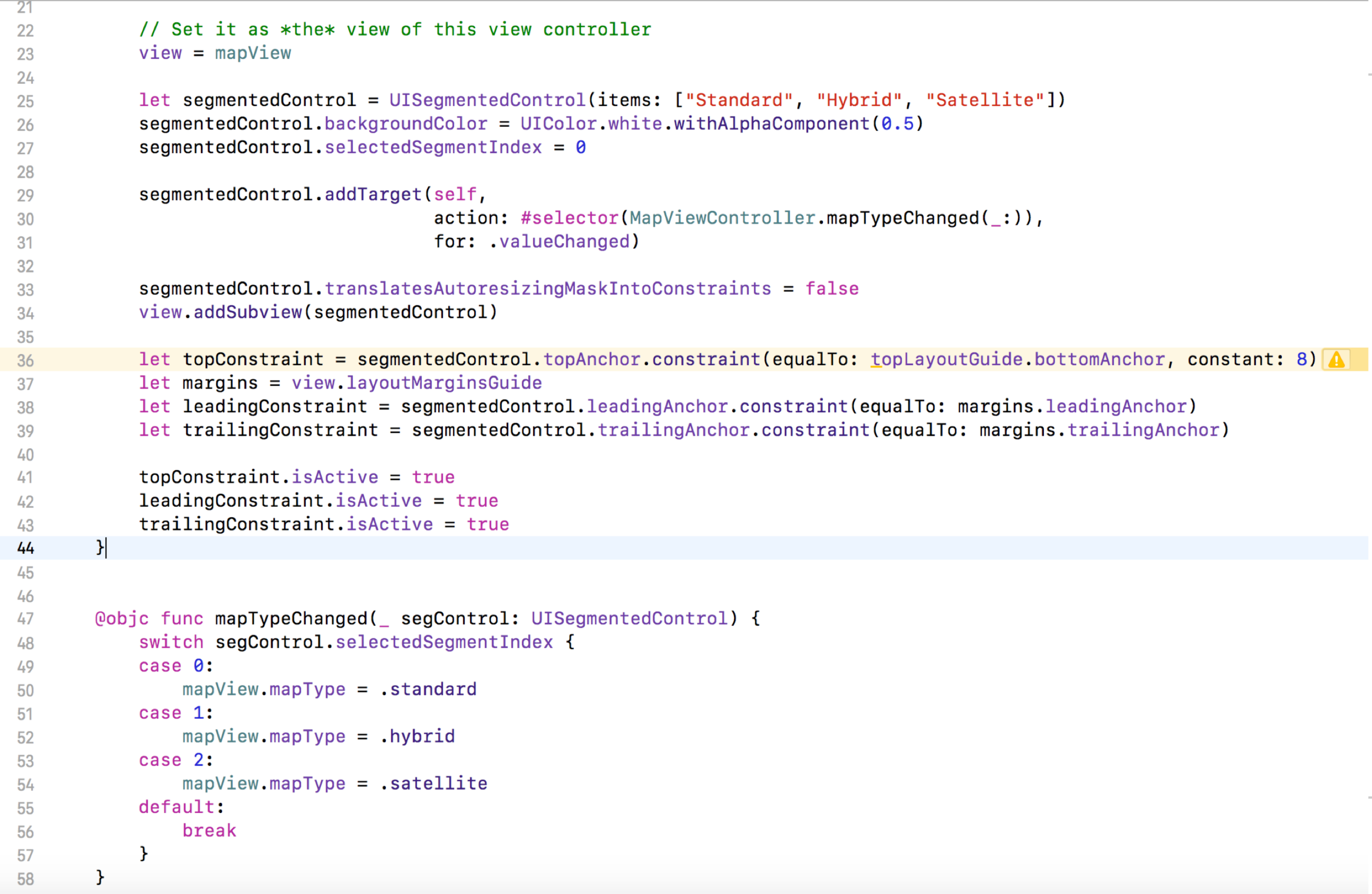
```
@objc func mapTypeChanged(_ segControl: UISegmentedControl) {
```

Compatibility with Objective C – newer versions of Xcode

In the textbook this code is omitted

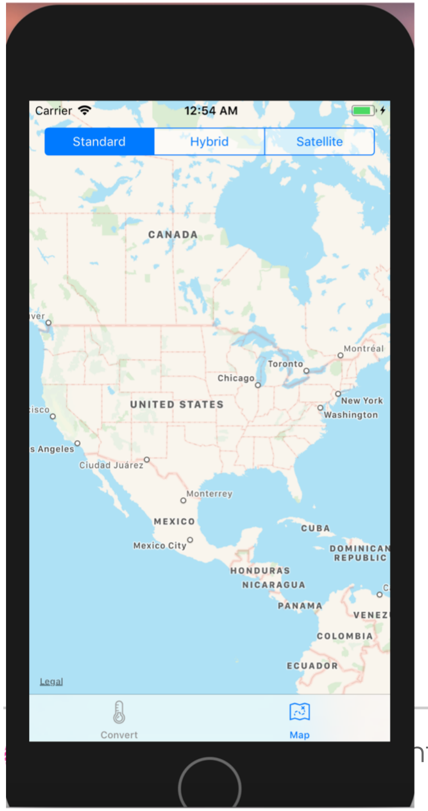
Change the map type when the user taps on a segment.
In MapViewcontroller.swift update loadView() to include .valueChanged event:

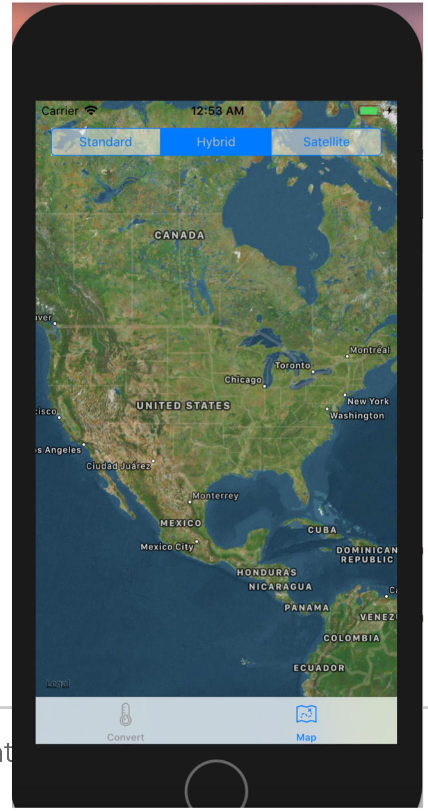
```
segmentedControl.selectedSegmentIndex = 0

    segmentedControl.addTarget(self,
                    action: #selector(MapViewController.mapTypeChanged(_:)),
                    for: .valueChanged)

    segmentedControl.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(segmentedControl)
```

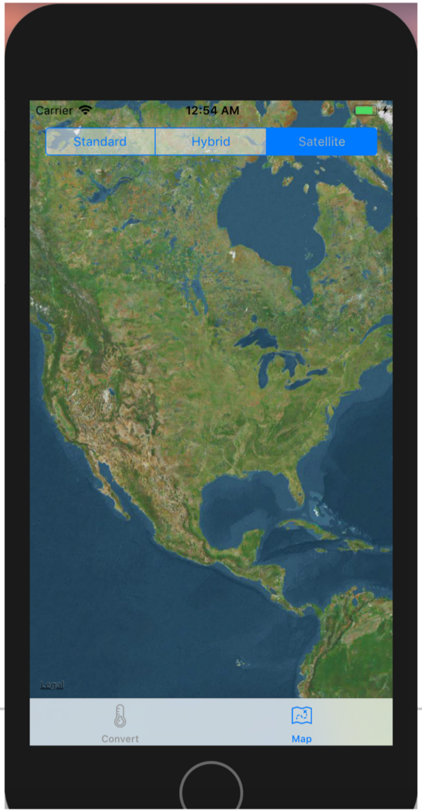


```swift
        // Set it as *the* view of this view controller
        view = mapView

        let segmentedControl = UISegmentedControl(items: ["Standard", "Hybrid", "Satellite"])
        segmentedControl.backgroundColor = UIColor.white.withAlphaComponent(0.5)
        segmentedControl.selectedSegmentIndex = 0

        segmentedControl.addTarget(self,
                                   action: #selector(MapViewController.mapTypeChanged(_:)),
                                   for: .valueChanged)

        segmentedControl.translatesAutoresizingMaskIntoConstraints = false
        view.addSubview(segmentedControl)

        let topConstraint = segmentedControl.topAnchor.constraint(equalTo: topLayoutGuide.bottomAnchor, constant: 8)
        let margins = view.layoutMarginsGuide
        let leadingConstraint = segmentedControl.leadingAnchor.constraint(equalTo: margins.leadingAnchor)
        let trailingConstraint = segmentedControl.trailingAnchor.constraint(equalTo: margins.trailingAnchor)

        topConstraint.isActive = true
        leadingConstraint.isActive = true
        trailingConstraint.isActive = true
    }


    @objc func mapTypeChanged(_ segControl: UISegmentedControl) {
        switch segControl.selectedSegmentIndex {
        case 0:
            mapView.mapType = .standard
        case 1:
            mapView.mapType = .hybrid
        case 2:
            mapView.mapType = .satellite
        default:
            break
        }
    }
```

Compile, run and test


Carrier
12:54 AM
Standard
Hybrid
Satellite
CANADA
UNITED STATES
MEXICO
Legal
Convert
Map


Carrier
12:53 AM
Standard
Hybrid
Satellite
CANADA
UNITED STATES
MEXICO
Legal
Convert
Map


Carrier
12:54 AM
Standard
Hybrid
Satellite
Legal
Convert
Map