Custom UITableViewCell from nib in Swift

I'm trying to create a custom table view cell from a nib. I'm referring to this article here. I'm facing two issues.

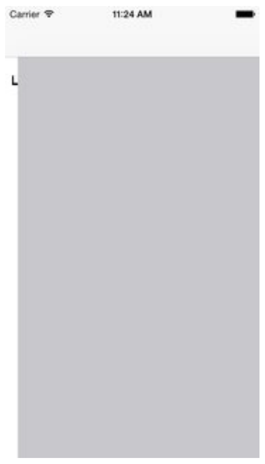
I created a .xib file with a UITableViewCell object dragged on to it. I created a subclass of `UITableViewCell` and set it as the cell's class and *Cell* as the reusable identifier.

```
import UIKit

class CustomOneCell: UITableViewCell {

    @IBOutlet weak var middleLabel: UILabel!
    @IBOutlet weak var leftLabel: UILabel!
    @IBOutlet weak var rightLabel: UILabel!

    required init(coder aDecoder: NSCoder!) {
        super.init(coder: aDecoder)
    }

    override init(style: UITableViewCellStyle, reuseIdentifier: String!) {
        super.init(style: style, reuseIdentifier: reuseIdentifier)
    }

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

}
```

In the UITableViewController I have this code,

```
import UIKit

class ViewController: UITableViewController, UITableViewDataSource,
UITableViewDelegate {

    var items = ["Item 1", "Item2", "Item3", "Item4"]

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    // MARK: - UITableViewDataSource
    override func tableView(tableView: UITableView!, numberOfRowsInSection section:
Int) -> Int {
        return items.count
    }

    override func tableView(tableView: UITableView!, cellForRowAtIndexPath
indexPath: NSIndexPath!) -> UITableViewCell! {
        let identifier = "Cell"
        var cell: CustomOneCell! =
tableView.dequeueReusableCellWithIdentifier(identifier) as? CustomOneCell
        if cell == nil {
            tableView.registerNib(UINib(nibName: "CustomCellOne", bundle: nil),
forCellReuseIdentifier: identifier)
            cell = tableView.dequeueReusableCellWithIdentifier(identifier) as?
CustomOneCell
        }

        return cell
    }
}
```

This code complies with no errors but when I run it in the simulator, it looks like this.



In the UITableViewController in the storyboard I haven't done anything to the cell. Blank identifier and no subclass. I tried adding the *Cell* identifier to the prototype cell and ran it again but I get the same result.

Another error I faced is, when I tried to implement the following method in the UITableViewController.

```
override func tableView(tableView: UITableView!, willDisplayCell cell:
CustomOneCell!, forRowAtIndexPath indexPath: NSIndexPath!) {

    cell.middleLabel.text = items[indexPath.row]
    cell.leftLabel.text = items[indexPath.row]
    cell.rightLabel.text = items[indexPath.row]
}
```

As shown in the article I mentioned I changed the `cell` parameter's type form `UITableViewCell` to `CustomOneCell` which is my subclass of UITableViewCell. But I get the following error,

Overriding method with selector 'tableView:willDisplayCell:forRowAtIndexPath:' has incompatible type '(UITableView!, CustomOneCell!, NSIndexPath!) -> ()'

Anyone have any idea how to resolve these errors? These seemed to work fine in Objective-C.

Thank you.

EDIT: I just noticed if I change the simulator's orientation to landscape and turn it back to portrait, the cells appear! I still couldn't figure out what's going on. I uploaded an Xcode project here demonstrating the problem if you have time for a quick look.

ios uitableview swift ios8

edited Aug 28 '14 at 11:08

asked Aug 28 '14 at 6:05
Isuru
13.3k 39 137 223

4 Answers

You should try the following code for your project (updated for Swift 2):

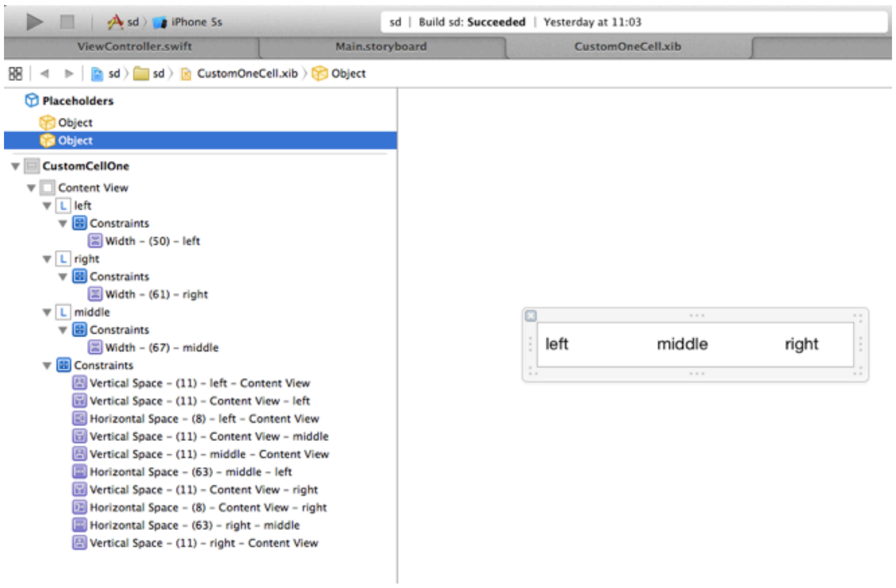
CustomOneCell.swift

```
import UIKit

class CustomOneCell: UITableViewCell {

    // Link those IBOutlets with the UILabels in your .XIB file
    @IBOutlet weak var middleLabel: UILabel!
    @IBOutlet weak var leftLabel: UILabel!
    @IBOutlet weak var rightLabel: UILabel!

}
```

TableViewController.swift

```
import UIKit

class TableViewController: UITableViewController {

    let items = ["Item 1", "Item2", "Item3", "Item4"]

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.registerNib(UINib(nibName: "CustomOneCell", bundle: nil),
forCellReuseIdentifier: "CustomCellOne")
    }

    // MARK: - UITableViewDataSource

    override func tableView(tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        return items.count
    }

    override func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("CustomCellOne",
forIndexPath: indexPath) as! CustomOneCell

        cell.middleLabel.text = items[indexPath.row]
        cell.leftLabel.text = items[indexPath.row]
        cell.rightLabel.text = items[indexPath.row]

        return cell
    }

}
```

The image below shows a set of constraints that work with the provided code without any constraints ambiguity message from Xcode.



sd | Build sd: **Succeeded** | Yesterday at 11:03

ViewController.swift | Main.storyboard | CustomOneCell.xib

sd › sd › CustomOneCell.xib › Object

```
Placeholders
  Object
  Object
▼ CustomCellOne
  ▼ Content View
    ▼ left
      ▼ Constraints
          Width - (50) - left
    ▼ right
      ▼ Constraints
          Width - (61) - right
    ▼ middle
      ▼ Constraints
          Width - (67) - middle
    ▼ Constraints
        Vertical Space - (11) - left - Content View
        Vertical Space - (11) - Content View - left
        Horizontal Space - (8) - left - Content View
        Vertical Space - (11) - Content View - middle
        Vertical Space - (11) - middle - Content View
        Horizontal Space - (63) - middle - left
        Vertical Space - (11) - Content View - right
        Horizontal Space - (8) - Content View - right
        Horizontal Space - (63) - right - middle
        Vertical Space - (11) - right - Content View
```

edited Dec 28 '15 at 21:04

answered Aug 28 '14 at 9:19

Imanou Petit
32.7k 7 118 106

Thanks for the response. But that didn't work either. Do I need to change anything in the table view controller? Because Its still set to prototype cells. – Isuru Aug 28 '14 at 10:23

Keep using prototype cells. But make sure that you have set the good Auto layout constraints (if you use Auto layout). – Imanou Petit Aug 28 '14 at 10:59

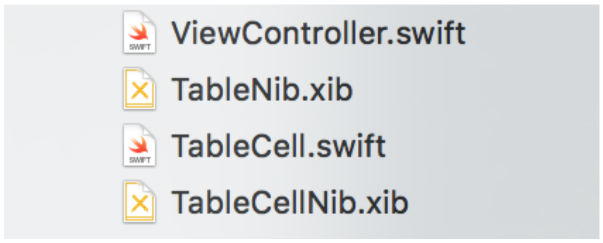
I uploaded a test project here demonstrating the issue I'm having. Can you please have a look at it if you have time? – Isuru Aug 28 '14 at 11:09

Your test project confirms it: I was able to make your app work fine after I set some auto layout constraints to your custom cell in your .xib file. Have a look at this video if you need to know more about Auto layout. – Imanou Petit Aug 28 '14 at 12:29

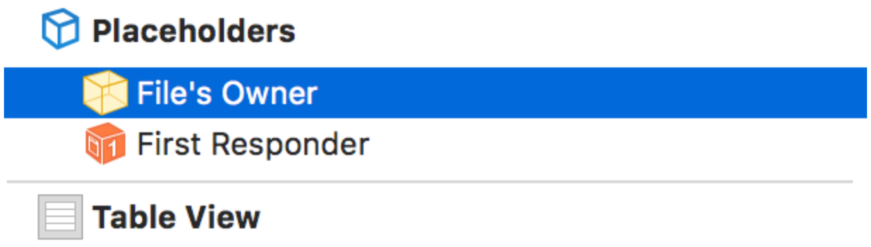
I'm aware of Auto Layout but I just can't figure out what kinda constraints I should use? Can you please tell me what constraints I should add to the custom cell. Or attach the Xcode project you got working? – Isuru Aug 28 '14 at 17:34

|

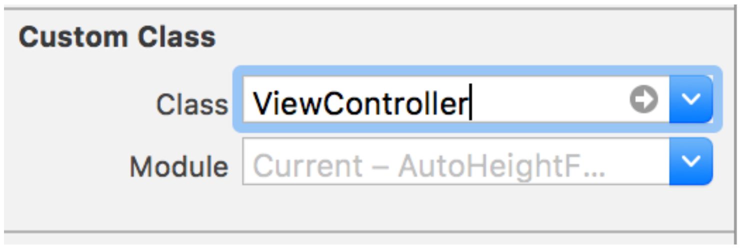
Here's my approach using Swift 2 and Xcode 7.3. This example will use a single ViewController to load two .xib files -- one for a UITableView and one for the UITableCellView.


ViewController.swift
TableNib.xib
TableCell.swift
TableCellNib.xib

For this example you can drop a UITableView right into an empty **TableNib**.xib file. Inside, set the **file's owner** to your ViewController class and use an outlet to reference the tableView.



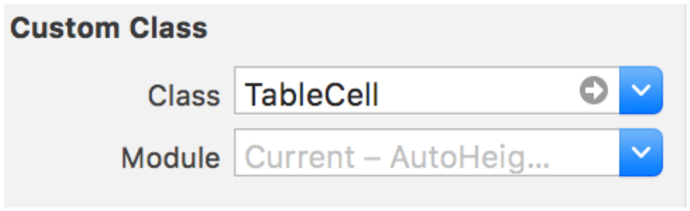
and


Custom Class
Class: ViewController
Module: Current – AutoHeightF...

Now, in your view controller, you can delegate the tableView as you normally would, like so

```
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource
{

    @IBOutlet weak var tableView: UITableView!

    ...

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

        // Table view delegate
        self.tableView.delegate = self
        self.tableView.dataSource = self

        ...
```

To create your Custom cell, again, drop a Table View Cell object into an empty **TableCellNib**.xib file. This time, in the cell .xib file you don't have to specify an "owner" but you do need to specify a **Custom Class** and an **identifier** like "TableCellId"


Custom Class
Class: TableCell
Module: Current – AutoHeig...

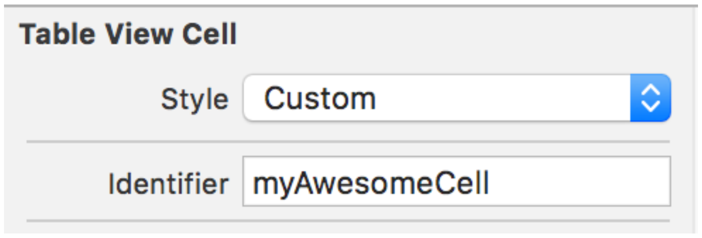


Table View Cell

Style: Custom

Identifier: myAwesomeCell

Create your subclass with whatever outlets you need like so

```
class TableCell: UITableViewCell {

    @IBOutlet weak var nameLabel: UILabel!

}
```

Finally... back in your View Controller, you can load and display the entire thing like so

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.

    // First load table nib
    let bundle = NSBundle(forClass: self.dynamicType)
    let tableNib = UINib(nibName: "TableNib", bundle: bundle)
    let tableNibView = tableNib.instantiateWithOwner(self, options: nil)[0] as!
UIView

    // Then delegate the TableView
    self.tableView.delegate = self
    self.tableView.dataSource = self

    // Set resizable table bounds
    self.tableView.frame = self.view.bounds
    self.tableView.autoresizingMask = [.FlexibleWidth, .FlexibleHeight]

    // Register table cell class from nib
    let cellNib = UINib(nibName: "TableCellNib", bundle: bundle)
    self.tableView.registerNib(cellNib, forCellReuseIdentifier: self.tableCellId)

    // Display table with custom cells
    self.view.addSubview(tableNibView)

}
```

The code shows how you can simply load and display a nib file (the table), and second how to register a nib for cell use.

Hope this helps!!!

edited Oct 3 '16 at 21:57

answered Jun 3 '16 at 10:40
internet-nico
1,200 13 14

1 can you explain what's the "tableCellId" in this line.... self.tableView.registerNib(cellNib, forCellReuseIdentifier: self.tableCellId).... because you have not defined whats that. and you cant manually define the identifier in xib .. no option is there to define it – Pradip Kumar Jul 27 '16 at 9:13

In the interface builder, when you create the tableCell, In the "attributes inspector" you define an identifier. The same identifier is what you use in your controller to reference the object. `let tableCellId = "myAwesomeCell"` . I added another image to help you. – internet-nico Oct 3 '16 at 21:56

Another method that may work for you (it's how I do it) is registering a class.

Assume you create a custom tableView like the following:

```
class UICustomTableViewCell: UITableViewCell {...}
```

You can then register this cell in whatever UITableViewController you will be displaying it in with "registerClass":

```
override func viewDidLoad() {
    super.viewDidLoad()
    tableView.registerClass(UICustomTableViewCell.self, forCellReuseIdentifier:
"UICustomTableViewCellIdentifier")
}
```

And you can call it as you would expect in the cell for row method:

```
override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath:
NSIndexPath) -> UITableViewCell {
    let cell =
tableView.dequeueReusableCellWithIdentifier("UICustomTableViewCellIdentifier",
forIndexPath: indexPath) as! UICustomTableViewCell
    return cell
}
```

answered Jun 17 '16 at 17:20

Ethan Kay
175 1 12

For fix the *"Overriding method... has incompatible type..."* error I've changed the function declaration to

```
override func tableView(tableView: (UITableView!),
                        cellForRowAtIndexPath indexPath: (NSIndexPath!))
    -> UITableViewCell {...}
```

(was `-> UITableViewCell!` -- with exclamation mark at the end)

answered Sep 21 '14 at 19:14

tse
1,415 17 34

this solved my problem with similar situation. Thanks!! – lusketeer Sep 25 '14 at 4:53