Table Views in iOS

Hands-On Challenges

Table Views: Beginning to Advanced Hands-On Challenges

Challenge #1: Scary Bugs

A long time ago it was late at night, and I couldn't sleep. So as usual, I was on my favorite web site (other than rw.com, of course!): reddit.

For some reason I came across this post where this guy took a picture of a bug he found in his back yard that looked like this:

It's hard to tell from the picture, but this thing is **HUGE**! It's called a Potato Bug, and I had never heard of such a thing before.

So of course, next I did what I always do and started obsessing, and before I knew it I was up till 4AM looking at scary bug pictures.

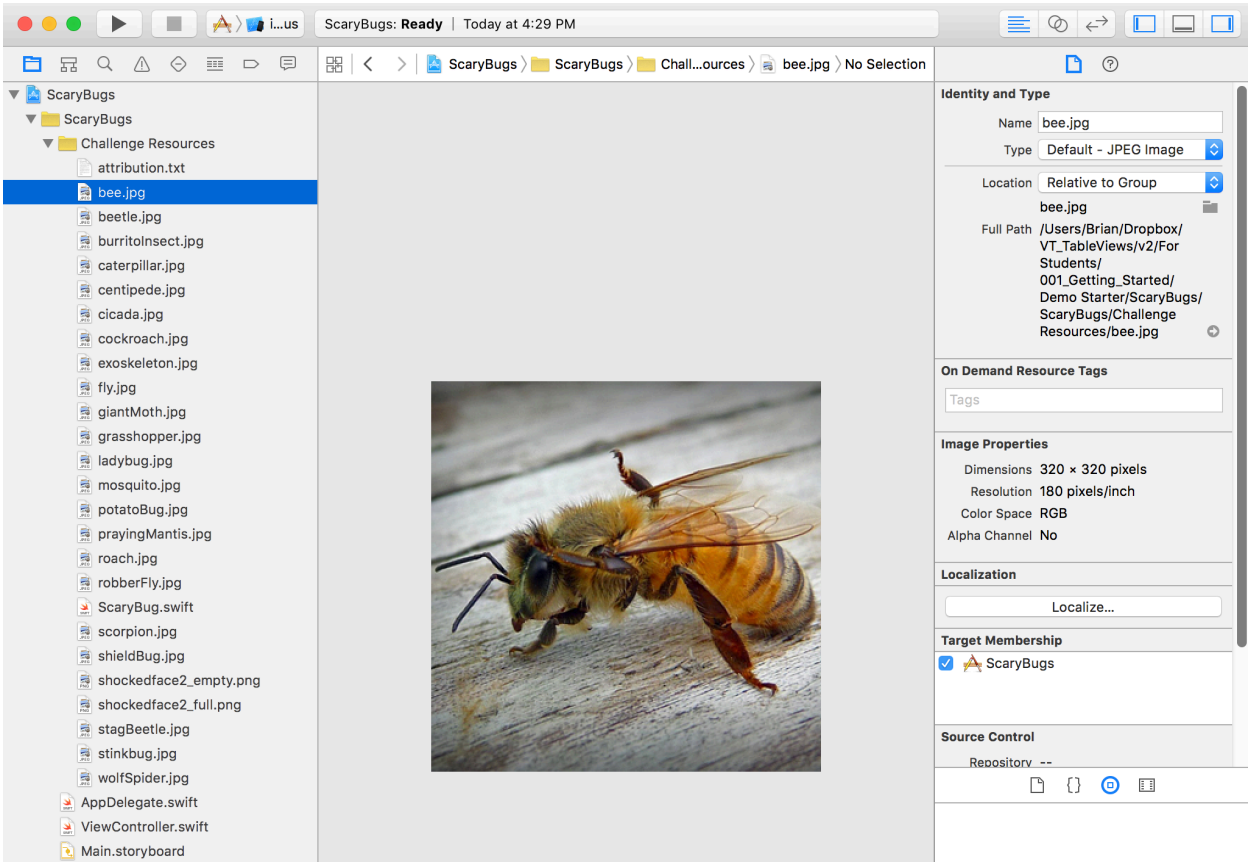
So in commemoration of this strange obsession, your first challenge in this Table Views in iOS series is to create a simple app that shows a list of scary bugs. This will demonstrate some basic usage of table views, and you will expand upon this in each challenge.

I've put together some images of scary bugs for you and a model class – if you're feeling brave, go ahead and try to make a simple table view app that lists the bugs based on what you learned in the lecture. But if you get stuck (or scared), don't worry – feel free to follow along with the step-by-step instructions instead!

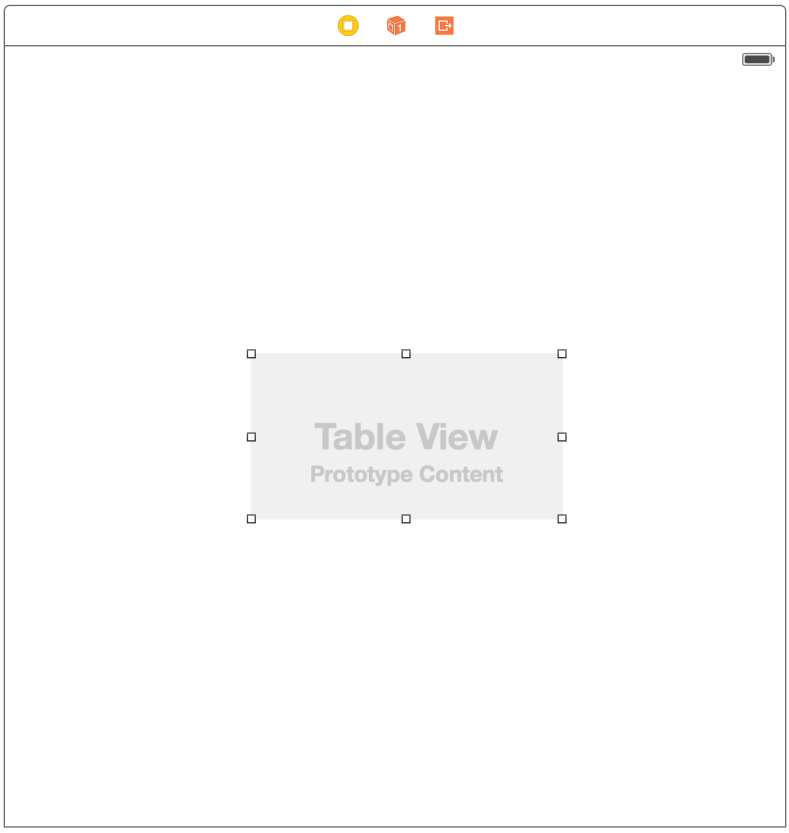
Full Walkthrough

Create a new project in Xcode with the **iOS\Application\Single View Application** template. Name the project **ScaryBugs**, and save it somewhere on your hard drive.

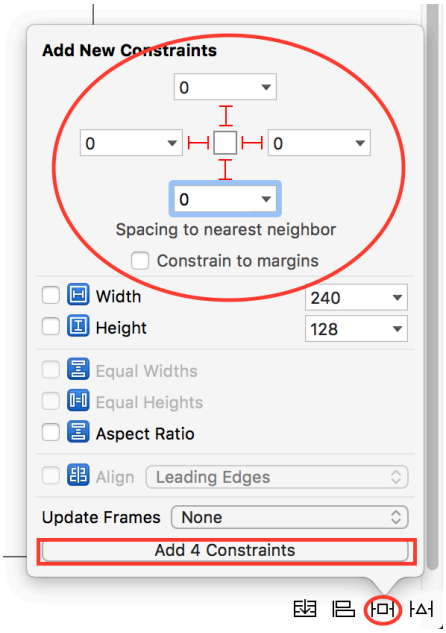
Then find the resources for this challenge and drag it into your Xcode project. You should see a list of files like this:



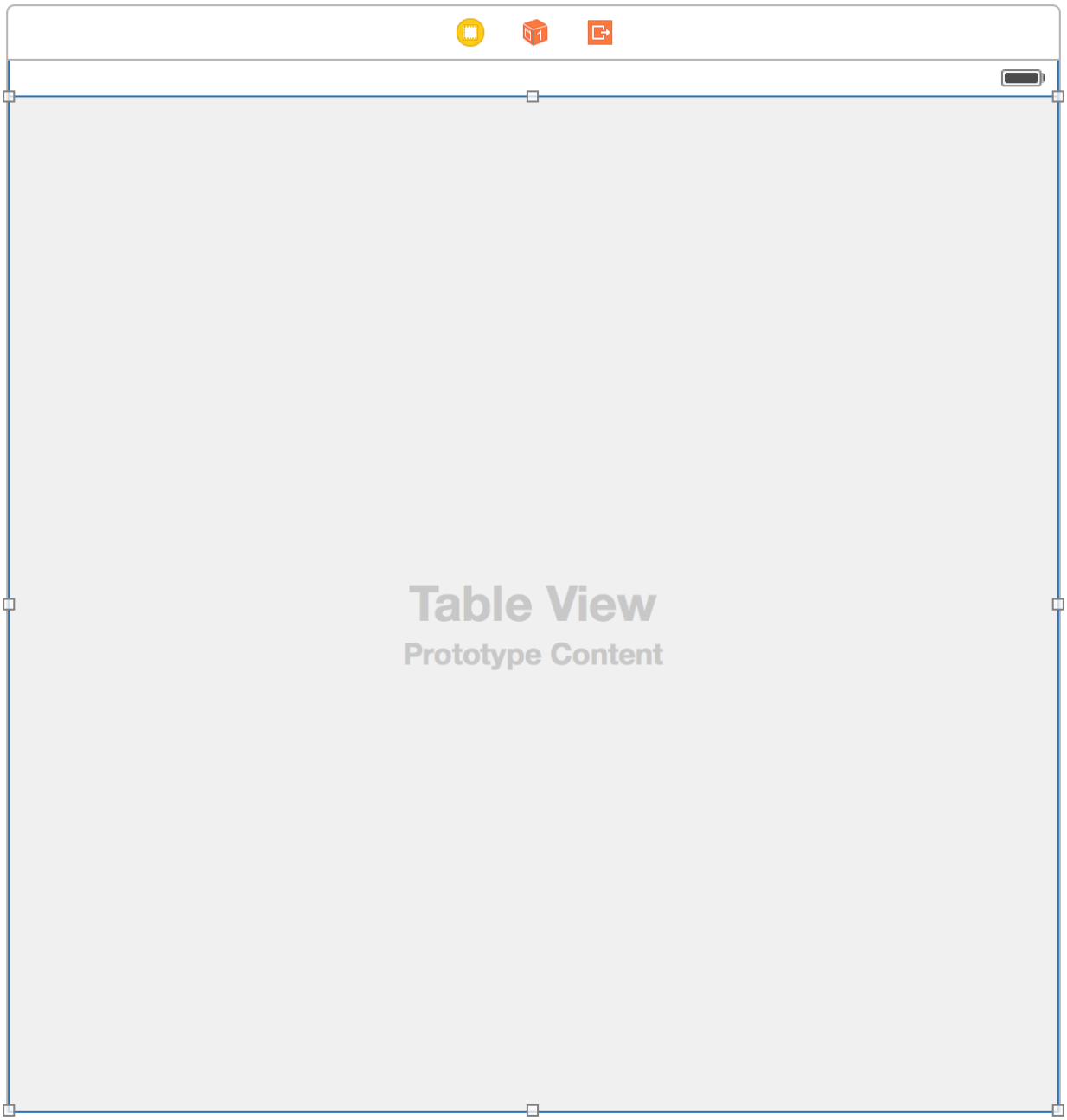
Open **Main.storyboard.** Drag a **table view** from the object library into the center of the **view controller**.



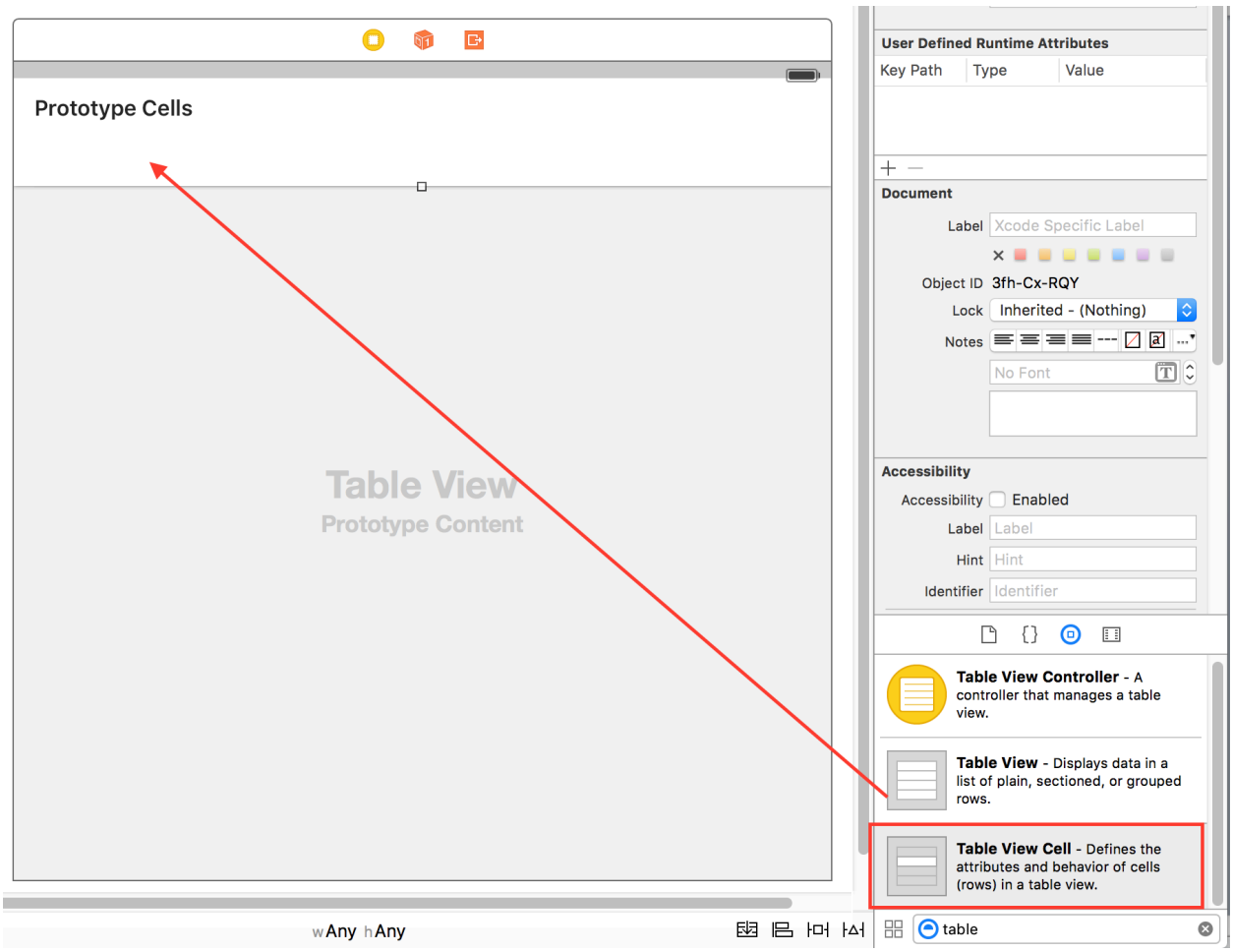
With the table selected, click the pin button and click the pin button. In the pin flyout, **uncheck Constrain to margins**. Click each I-beam at the **top**, **right**, **bottom**, and **left** views. Make sure to set the value to **0.** Finally, click the **Add 4 Constraints** button.



Finally, click the table view in the View Controller Scene, and from the menu bar, select **Editor \ Resolve Auto Layout Issues \ All Views \ Update Frames**. Now, your table view will be the size of the view controller.

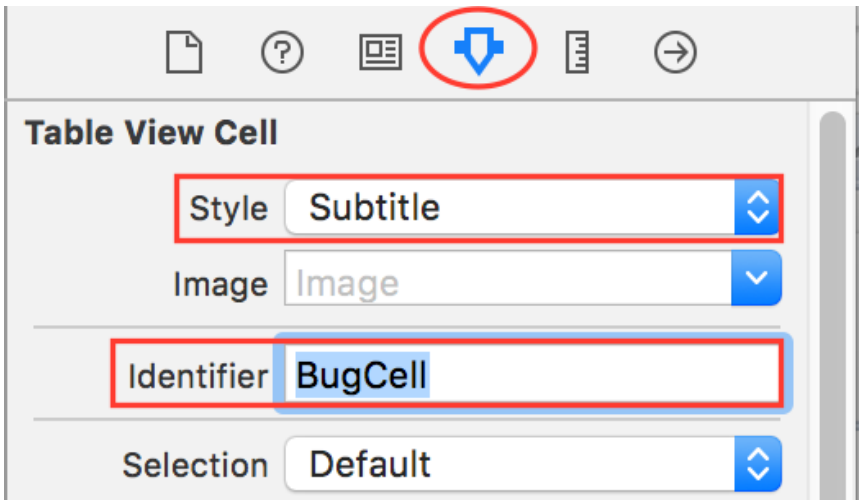


Next, drag a table view cell from the object library and drop it into the table view.



Next select **Editor\Embed In\Navigation Controller** to put it inside a navigation controller. You will see that the Navigtation Controller is set as the initial view controller. Select the **Navigation Item** inside the table view controller and set the **Title** to **Scary Bugs**.

Select the table view cell inside the table view. In the Attributes Inspector, set its **Style** to **Subtitle**, and set the **Identifier** to **BugCell**.



Next, you need to populate your table in code. Open ViewController.swift, and replace the contents with the following:

```
import UIKit

class ViewController: UIViewController {

  var bugs = [ScaryBug]()

  override func viewDidLoad() {
    bugs = ScaryBug.bugs()
    automaticallyAdjustsScrollViewInsets = false
  }

}

extension ViewController : UITableViewDataSource {

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

  func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell {

    let cell =
tableView.dequeueReusableCellWithIdentifier("BugCell",
forIndexPath: indexPath)
    let bug = bugs[indexPath.row]
    cell.textLabel?.text = bug.name
```

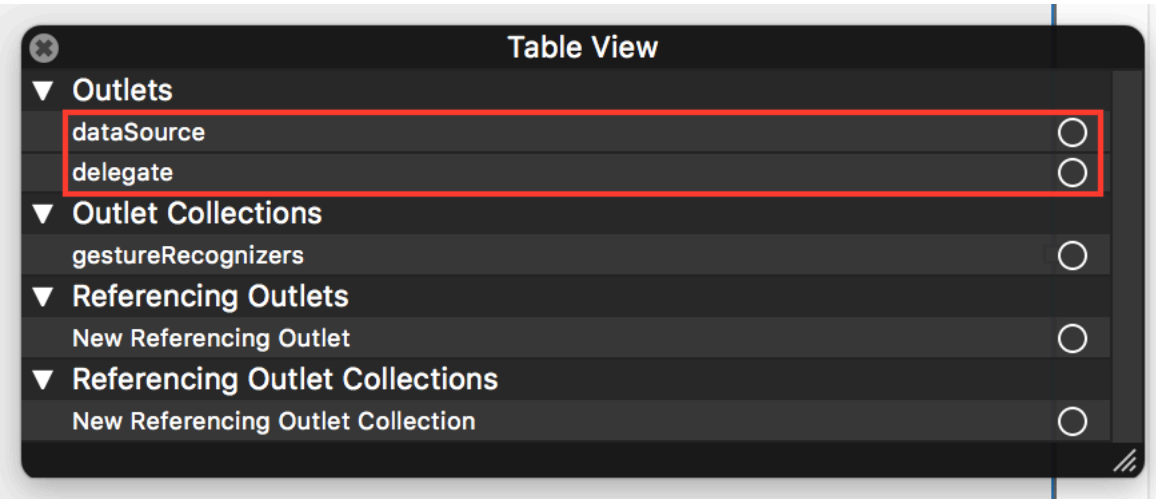
```
        cell.detailTextLabel?.text =
    ScaryBug.scaryFactorToString(bug.howScary)

        if let imageView = cell.imageView, bugImage = bug.image {
            imageView.image = bugImage
        }

        return cell

    }

}
```

After you write the above code, see if you can answer these questions to yourself:

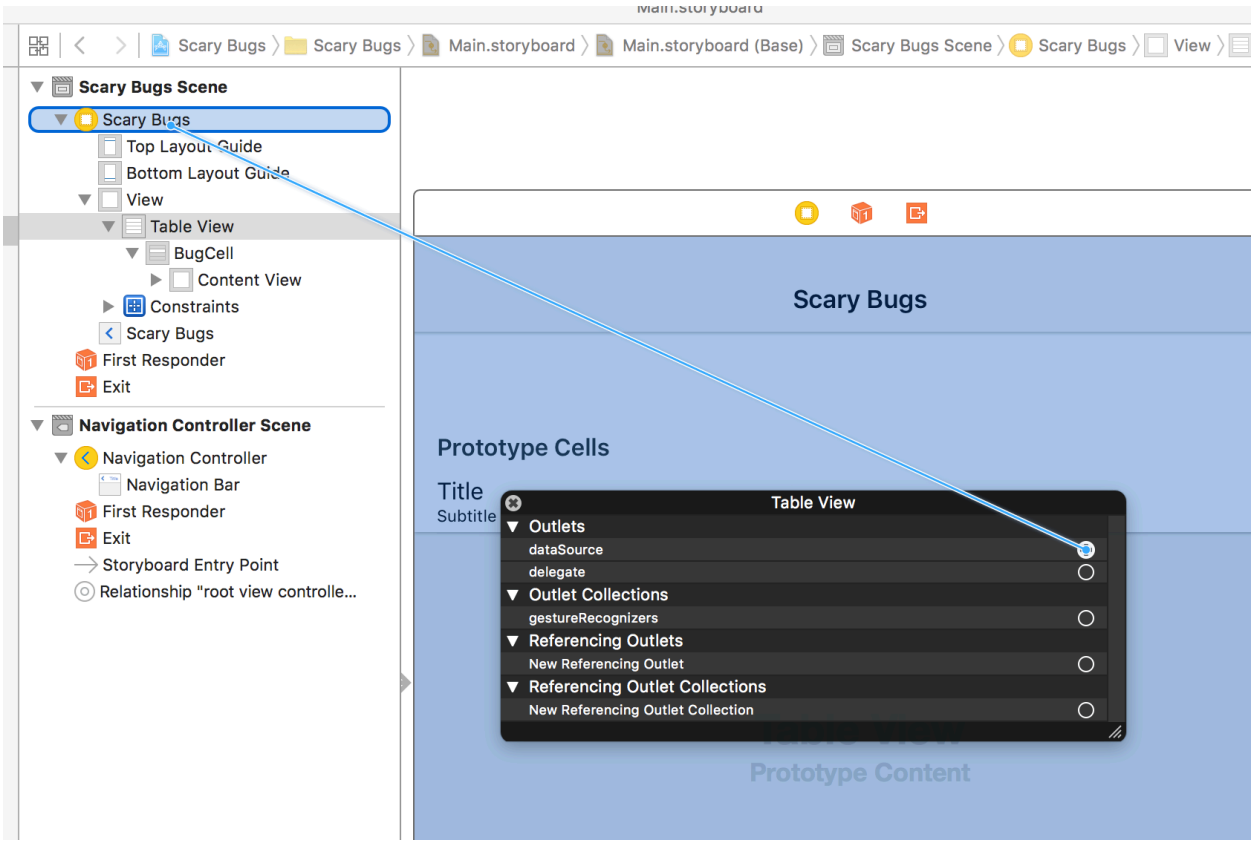
- There is a protocol that has the **tableView(_:numberOfRowsInSection:)** and **tableView(_:cellForRowAtIndexPath:)** methods listed here as required methods. What is the name of this protocol?
- What does **dequeueReusableCellWithIdentifier(_:forIndexPath:)** do?
- What are the four built-in cell styles you can choose from?

If you are unsure about anything, refer back to the lecture.

There's only one last thing to do. Switch back Main.Storyboard, and right click or control click your table view. You'll see a black dialog with two properties: **dataSource** and **delegate**.



Connect both the delegate and the dataSource to the parent view controller.



Build and run, and you should see a table view filled with scary bugs!