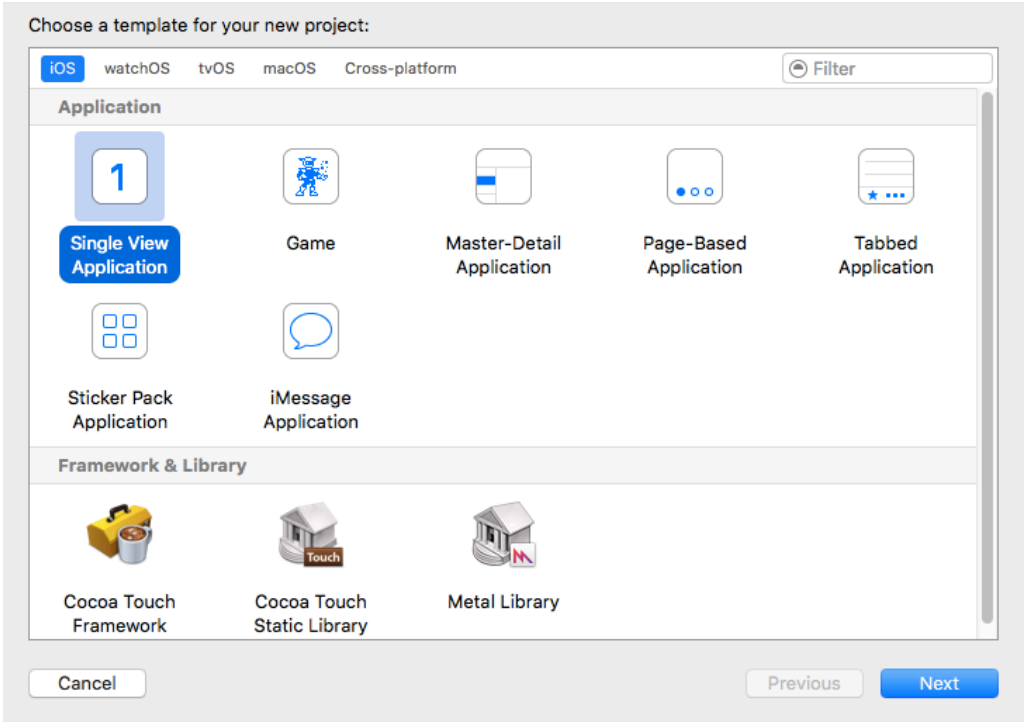
Practical 04: Creating Table View Application

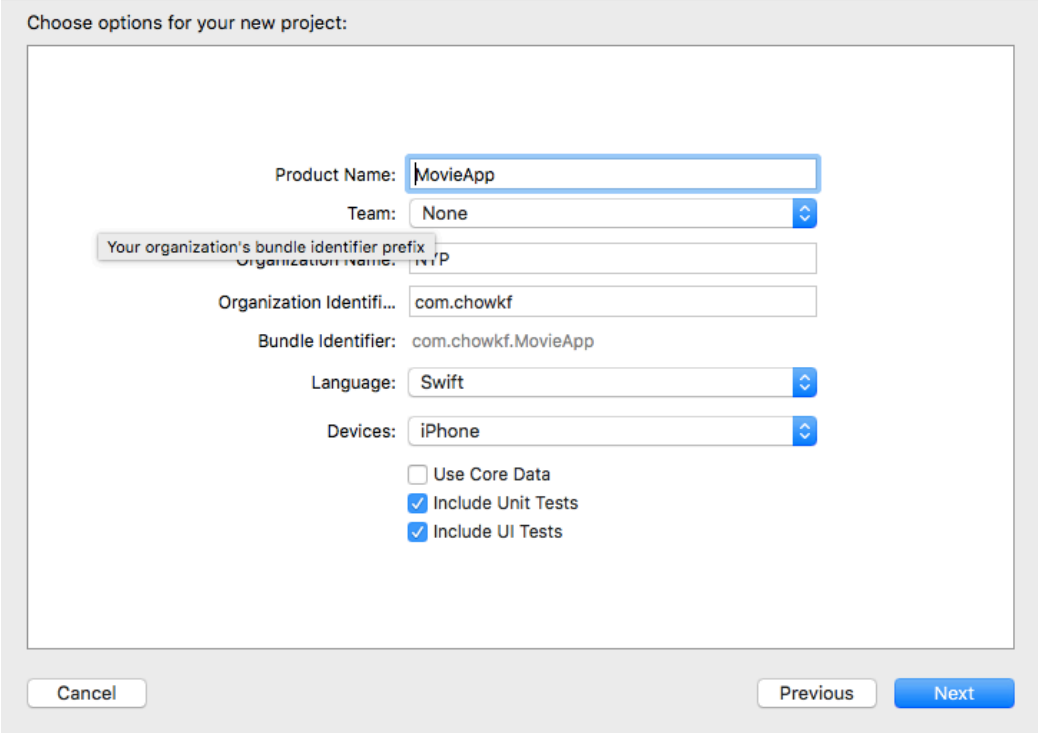
In this lab, you will practice the basics of creating a table view application using TableViewDelegates and TableViewDatasource.

Section 1: Creating a Empty Application Template

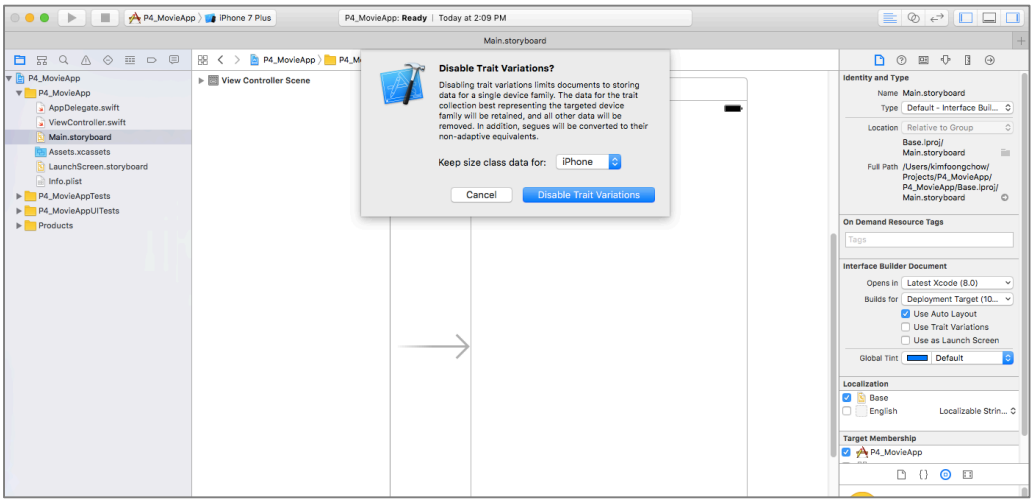
1. Using Xcode, choose a **Single View Application** template for this project.



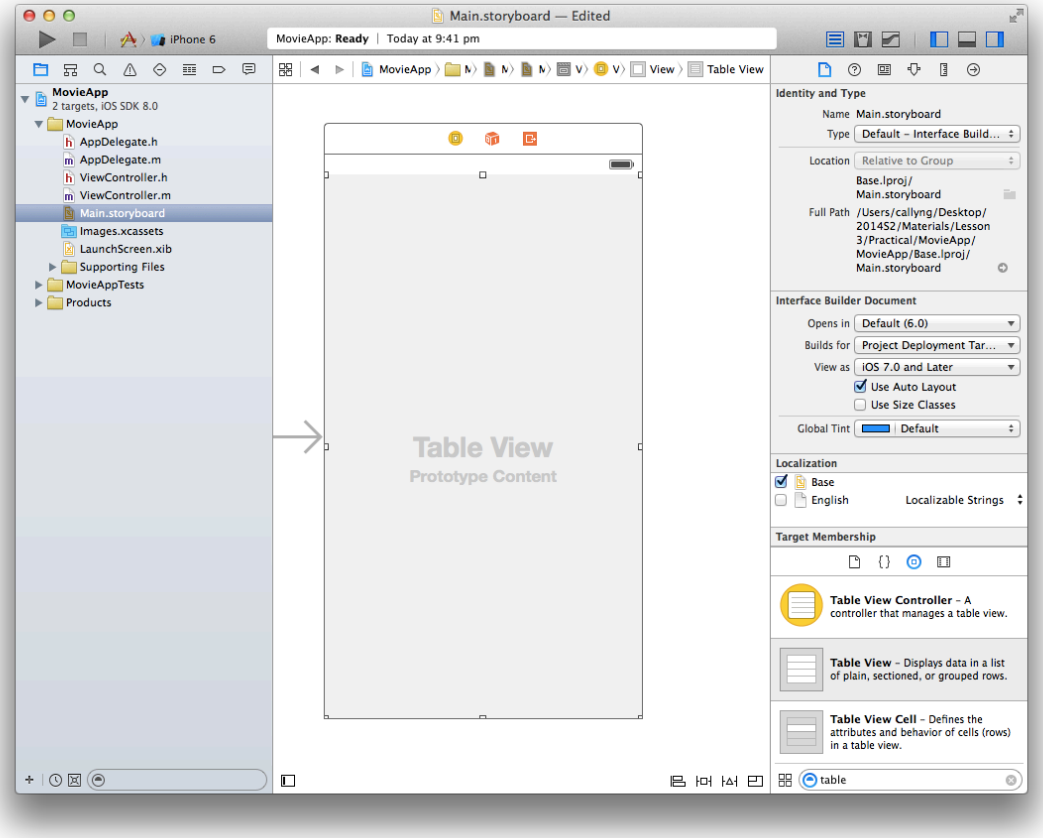
2. Next, configure this project and name is as **MovieApp**.



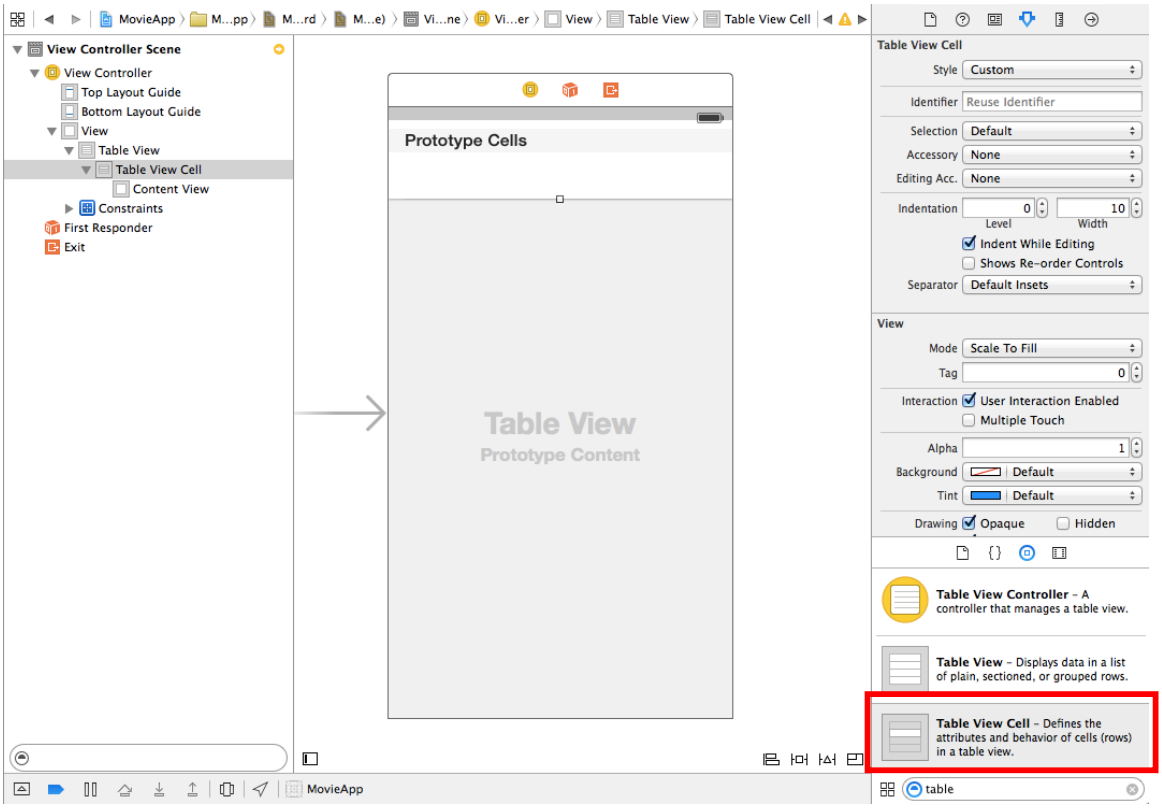
3. Click on the **Main.Storyboard** and at the Utility navigator pane, under the **File Inspector**, make sure the **Use Trait Variations** is uncheck. You will be prompt to **Disable Trait Variations** and Keep size class data for **iPhone**.



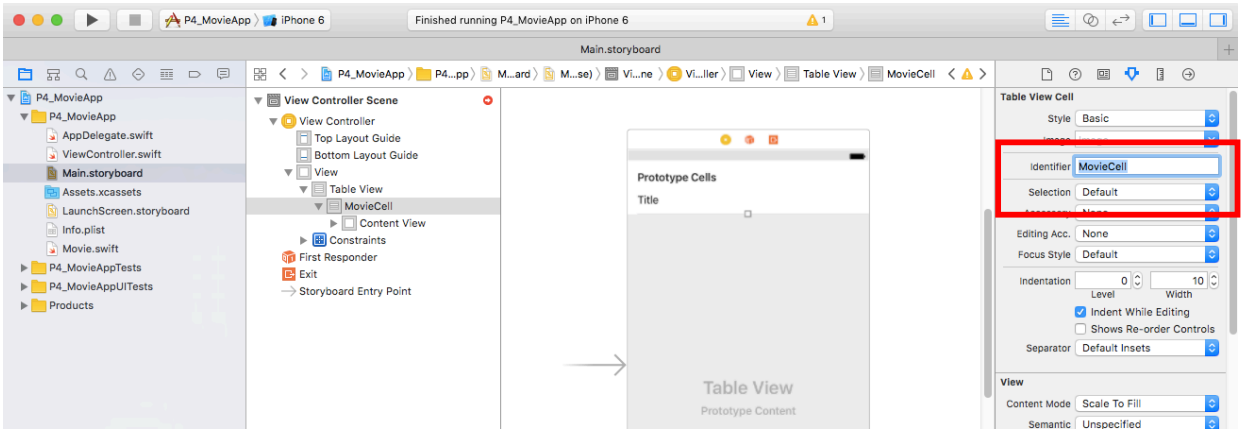
4. In the Object Library, select the **Table View** object and drag it into the view. Resize the Table View height to fit nicely in the View Controller and make sure it does not cover the status bar. Your screen should look like below after inserting the table view.



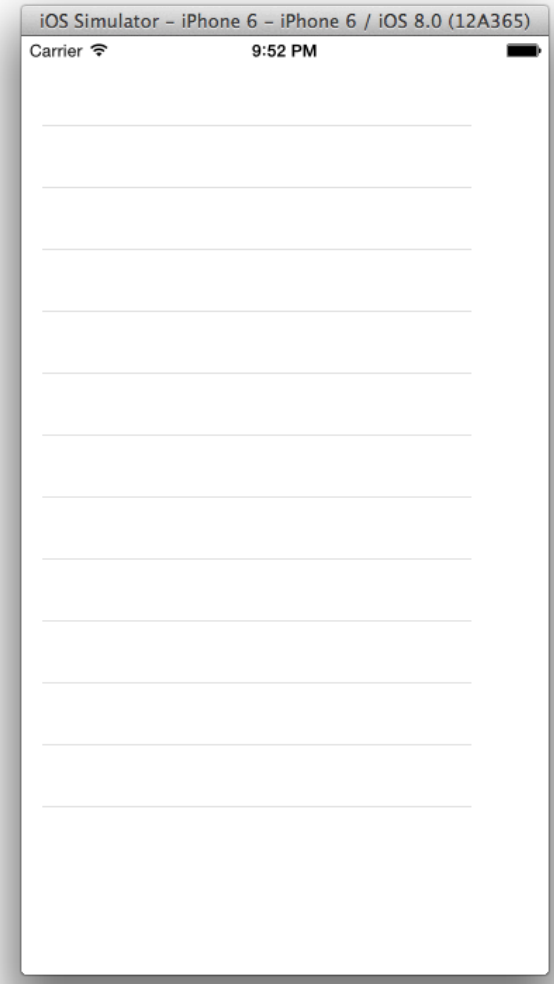
5. Click on the **Main.storyboard**. Under the Object Library, drag and drop the **Table View Cell** object onto the Table View.



6. With the Table View Cell selected, under the **Attribute Inspector**, make sure the style of the Table View Cell is **Basic** and set the **Identifier** to **MovieCell**.

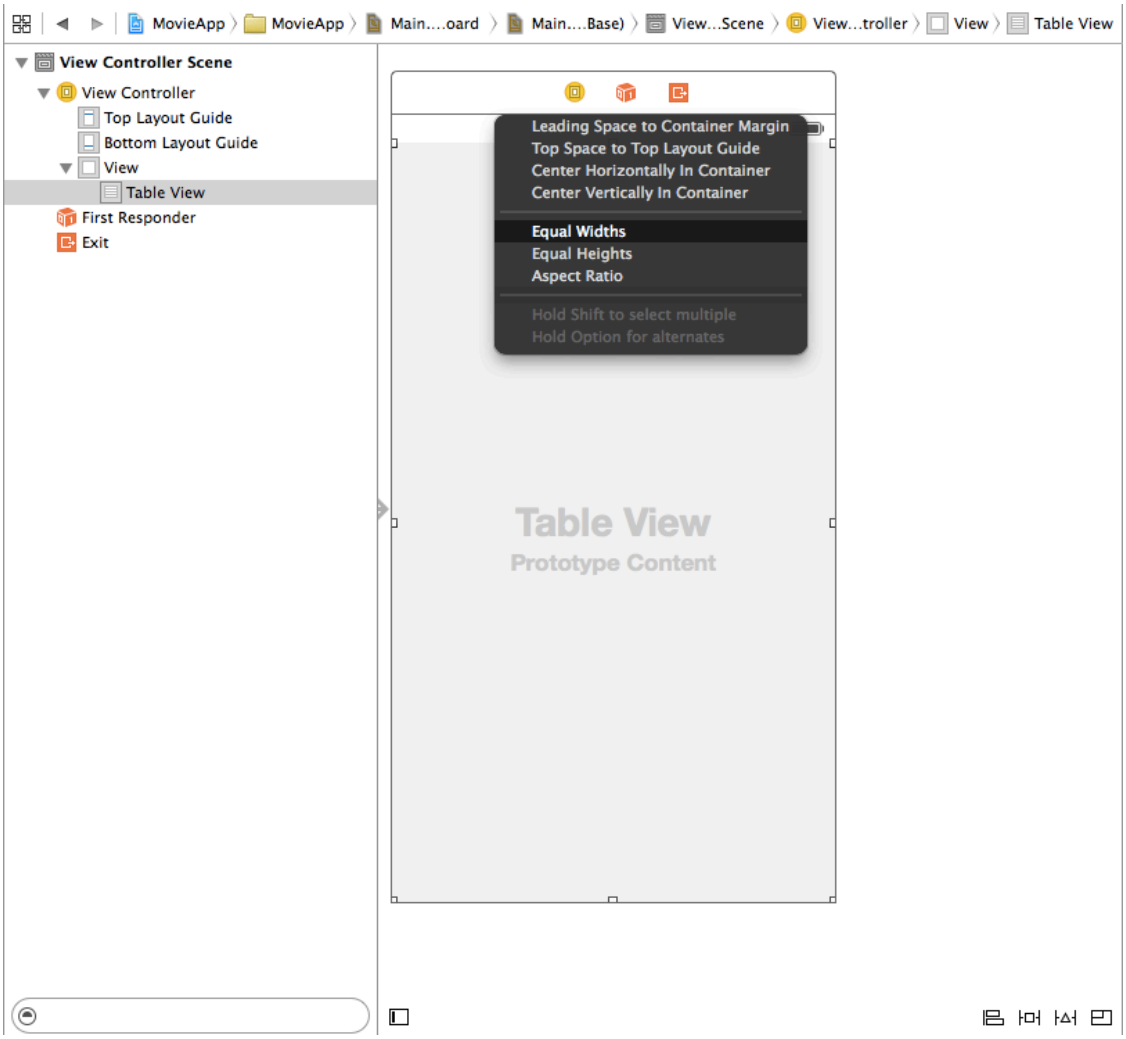


7. Build and run the application. You will see the default appearance of a plain UITableView with no contents.

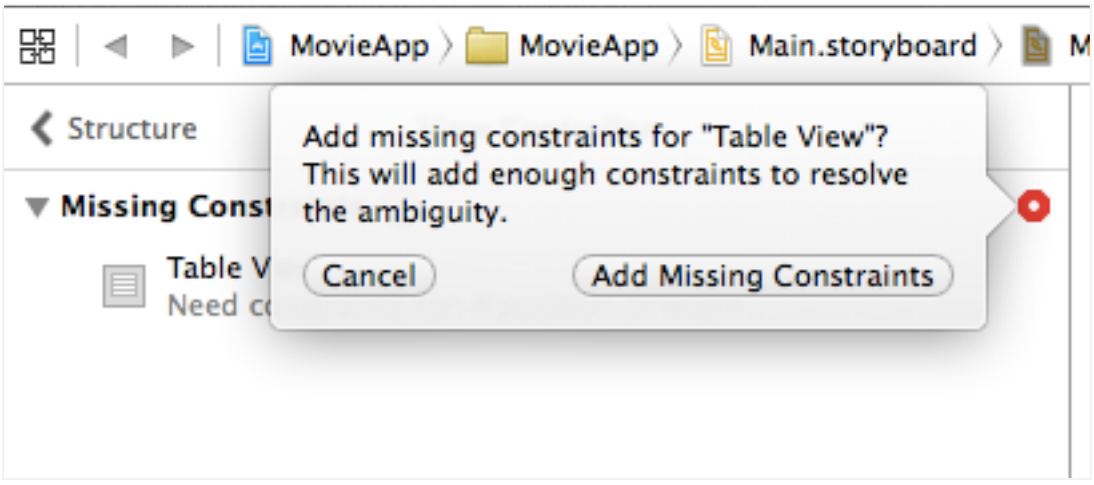


8. You will noticed that the table view does not fit nicely in the View. Let's add a few constraints for the table view so that it will take up the whole width and height of the view.

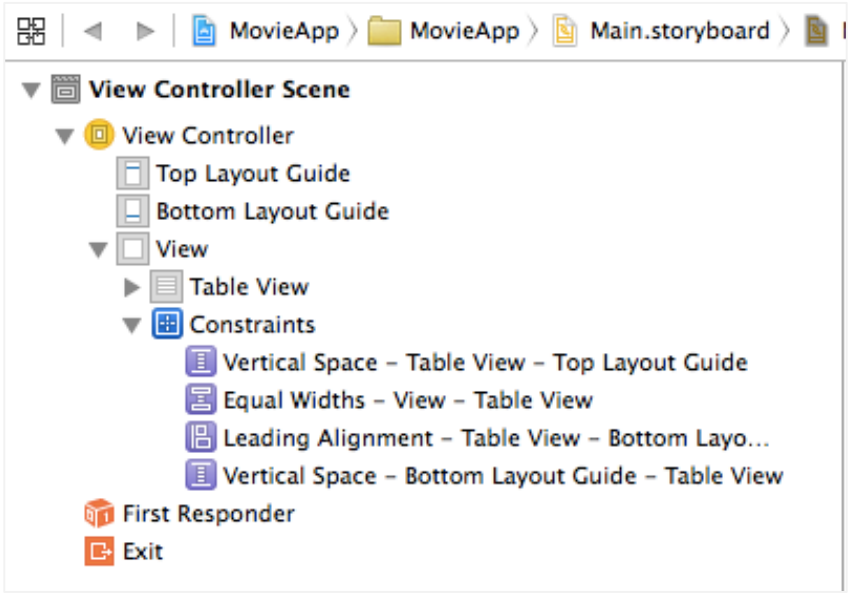
9. At the interface builder, select the **table view**. **Ctrl drag** the **Table View** to the **View** and add the constraints **Equal Widths**.



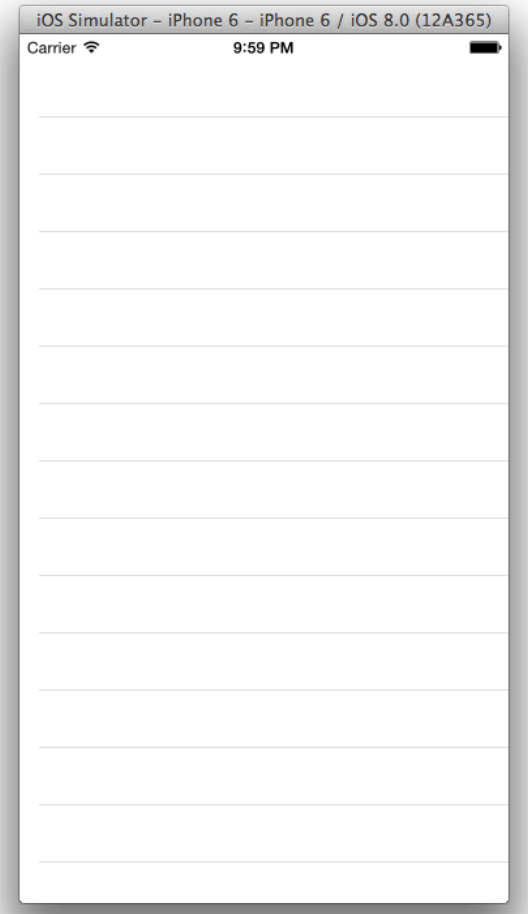
10. Next, Ctrl+Drag the **Table view** to the **Top Layout Guide** and select **Vertical Spacing** constraints.
11. Ctrl+Drag the **Table view** to the **Bottom Layout Guide** and select **Vertical Spacing** constraints.
12. Lastly, you will see the warning at the outline pane as show below. Click on the Add Missing Constraints.



13. Ensure that the constraints are as show below:



14. Try and run the application.



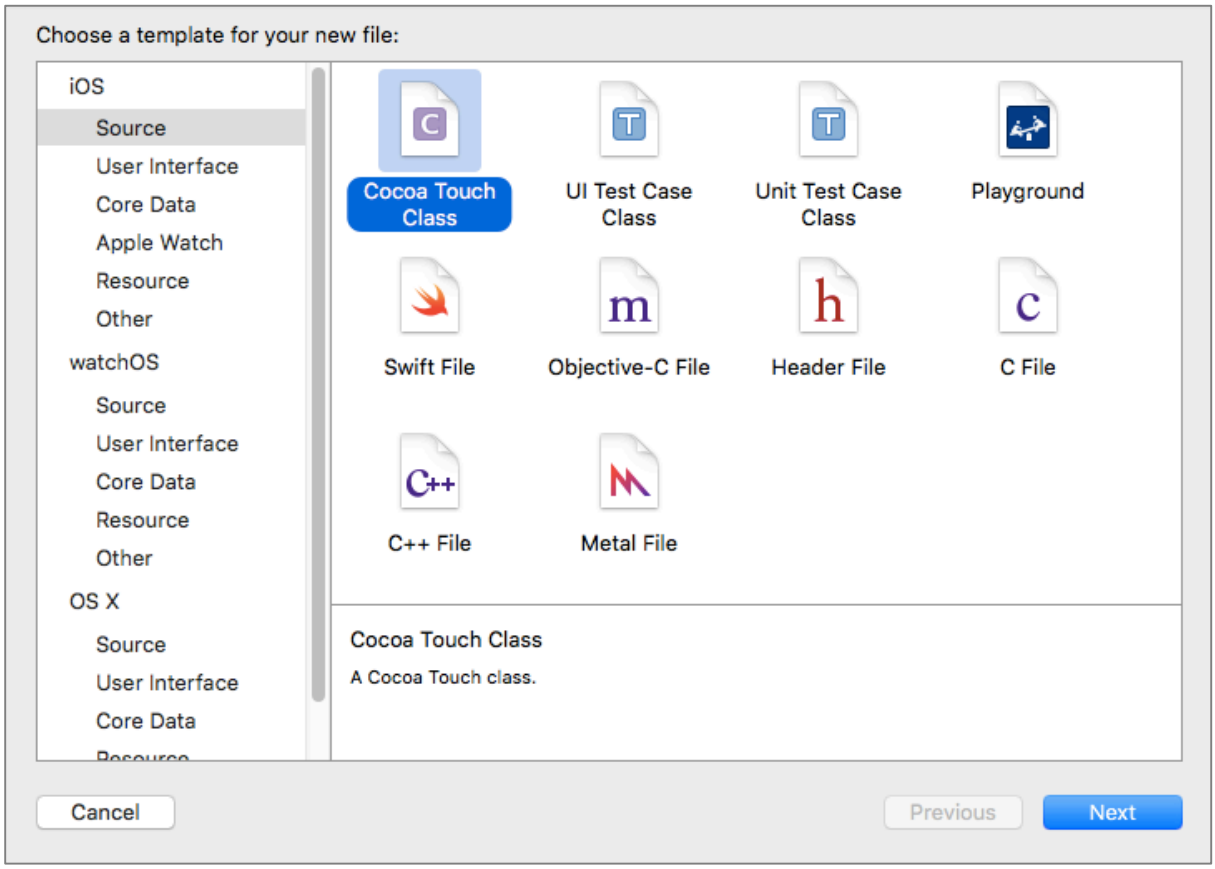
Section 2: Binding Data to Table View

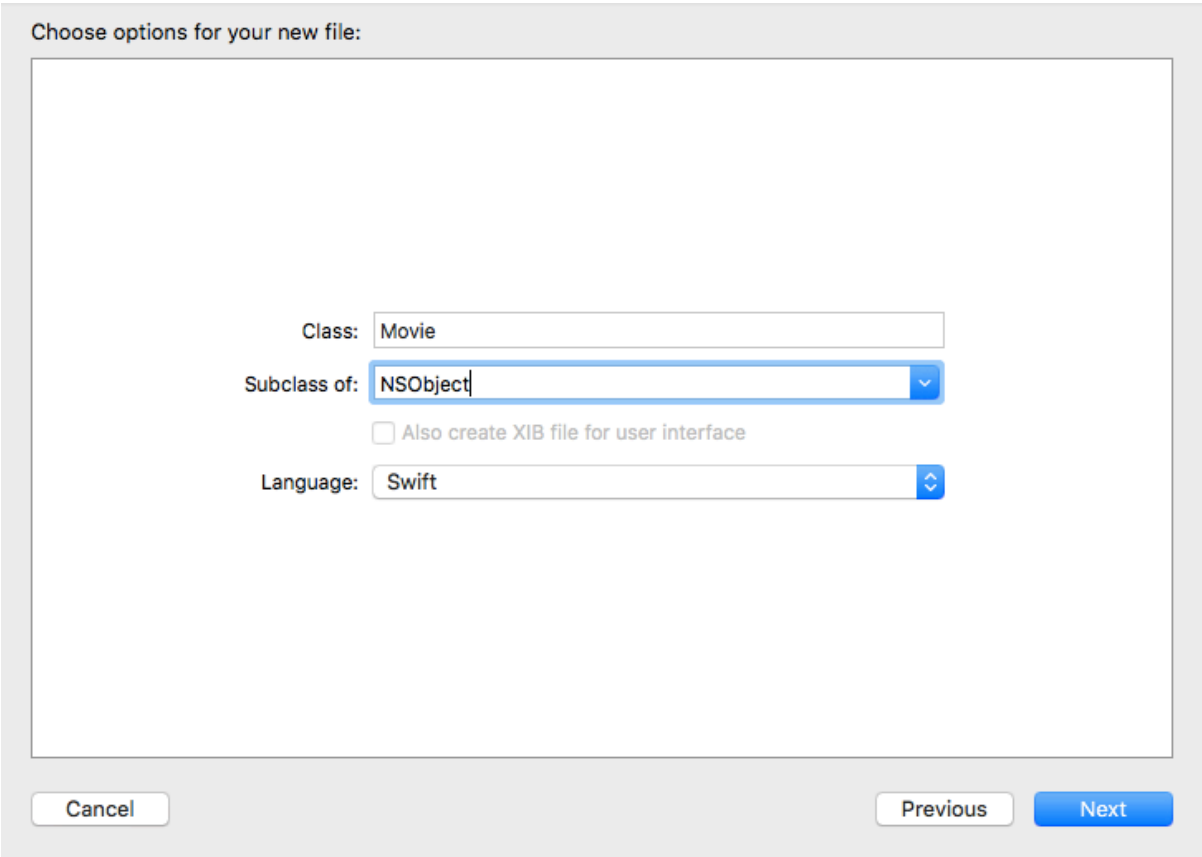
*In this section, we will be populating data to the Table View using **UITableViewDataSource**. A table's data source provides the tables with its data.*

15. in order to display data in Table View, we have to conform to the requirements defined in the protocols and implement all the mandatory methods.
    In the **ViewController.swift**, add the following codes to conform to the **UITableViewDelegate** and **UITableViewDatasource**.

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

16. Before we create the array to store the list of movies, we must first create a subclass of **NSObject** named **Movie**. An instance of Movie class represents an item with a set of attributes. To create a new class in Xcode, right-click on the project folder at the project navigator panel, choose **New File…**. Then select **Cocoa Touch Class** class, click **Next** and name the class **Movie** with subclass **NSObject**.





17. Next, declare some properties for **Movie.swift** class, and an initializer for the properties.

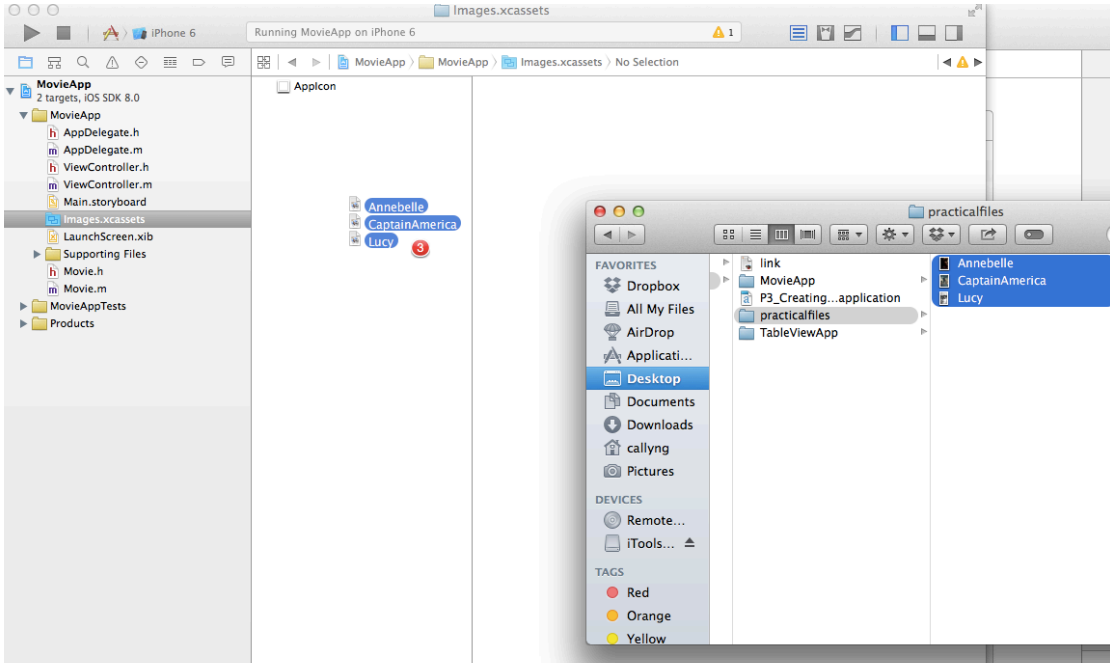
```
import UIKit

class Movie: NSObject {
    var movieName: String!
    var movieDesc: String!
    var runtime: Int
    var imageName: String!

    init(name: String,
         description desc: String,
         runTime time:Int,
         imageName img: String )
    {

         self.movieName = name
         self.movieDesc = desc
         self.runtime = time
         self.imageName = img
         super.init()
    }

}
```

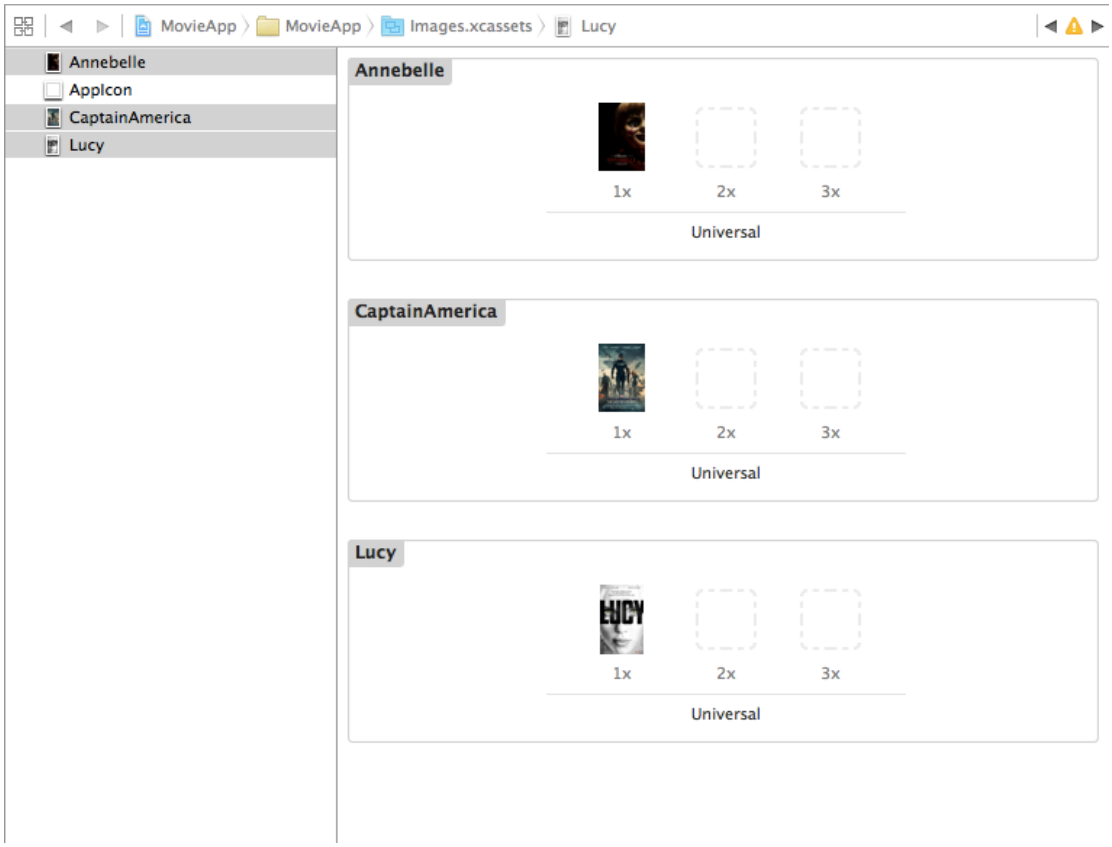
18. In the **ViewController.swift**, create an **array** to store the list of movies and create the **tableView**.

```
#
```

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

    // Declare an array of Movie objects
    var movieList : [Movie] = []

    @IBOutlet weak var tableView : UITableView!
}
```

19. Before we add objects to the array list, we first have to add images to the project. Images will be used later in this practical. Click on the **Assets.xcassets**.
From the Finder, select the images found in **practicalfiles.zip** in Blackboard and drag to the right side of editor.





20. Next, we going to create a few Movie objects and store in the array that we have created earlier. In the **ViewController.swift**, import the **Movie** header file and modify the viewDidLoad method.

```
    // When the view is loaded, do some initialization.
    override func viewDidLoad() {
        super.viewDidLoad()

        // Create some Movie objects and insert it into
        // the array.
        //
        movieList.append(Movie(
            name: "Annabelle",
            description: "A couple begin to experience terrifying
supernatural occurrences involving a vintage doll shortly after their
home is invaded by satanic cultists.",
            runTime: 115,
            imageName:"Annebelle.jpg"))

        movieList.append(Movie(
            name: "The Winter Soldier",
            description: "Steve Rogers, now finding difficult to fit in
to the era of today then leads an assault against a friend turned rival
from World War II, a Soviet emissary known as The Winter Soldier and
his lead of a precarious uprising.",
            runTime: 140,
            imageName:"CaptainAmerica.jpg"))

        movieList.append(Movie(
            name: "Lucy",
            description: "A woman, accidentally caught in a dark deal,
turns the tables on her captors and transforms into a merciless warrior
evolved beyond human logic.",
            runTime: 112,
            imageName:"Lucy.jpg"))
```

```
}
```

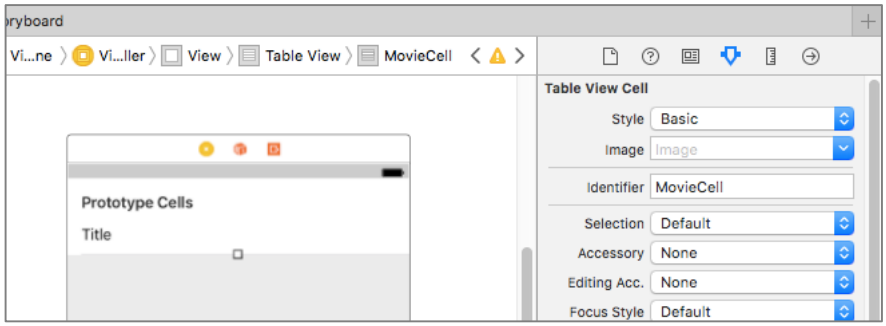
21. In order to populate the data using the UITableViewDataSource, we need to implement the two required methods: **tableView(tableView, numberOfRowsInSection)** and **tableView(tableView, cellForRowAt)**. These methods tell the table view how many rows to display and what content to be displayed.

    In the **ViewController.swift**, add the following codes:

```
// This is a function that the UITableViewDataSource
// should implement. It tells the UITableView how many
// sections there will be.
//
func tableView(_ tableView: UITableView,
    numberOfRowsInSection section: Int) -> Int
{
    return movieList.count
}

// This is a function that the UITableViewDataSource
// must implement. It needs to create / reuse a UITableViewCell
// and return it to the UITableView to draw on the screen.
//
func tableView(_ tableView: UITableView,
    cellForRowAt indexPath: IndexPath)
    -> UITableViewCell
{
    // First we query the table view to see if there are
    // any UITableViewCells that can be reused. iOS will
    // create a new one if there aren't any.
    //
    let cell =
        tableView.dequeueReusableCell(withIdentifier:
        "MovieCell", for: indexPath)

    // Using the re-used cell, or the newly created
    // cell, we update the text label's text property.
    //
    let p = movieList[indexPath.row]
    cell.textLabel?.text = "\(p.movieName!) (\(p.runtime) mins)"

    return cell
}
```

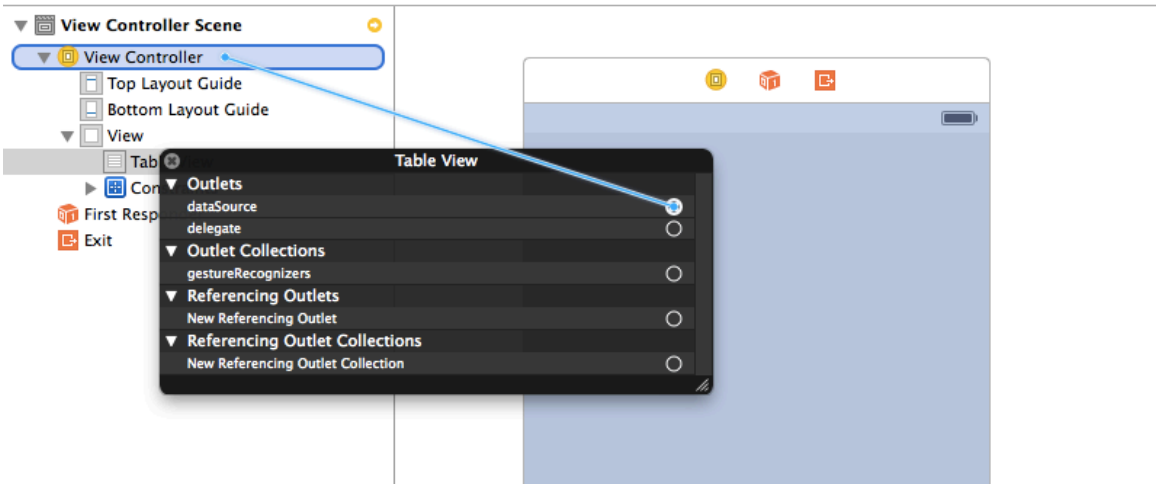
Note: *iOS devices have limited memory, so it is important to reuse cell instances by using the resueIdentifier property. When user scrolls the table, some cells move offscreen. For those offscreen cells, it will become unused cells and the data source will configure it with new data and returns to the table view.*

Note: *The dequeueReusableCell requires an identifier. In this case we passed in "MovieCell". This name has to be the same name as the one we entered into our Storyboard designer earlier when we set the identifier for the table view cell.*



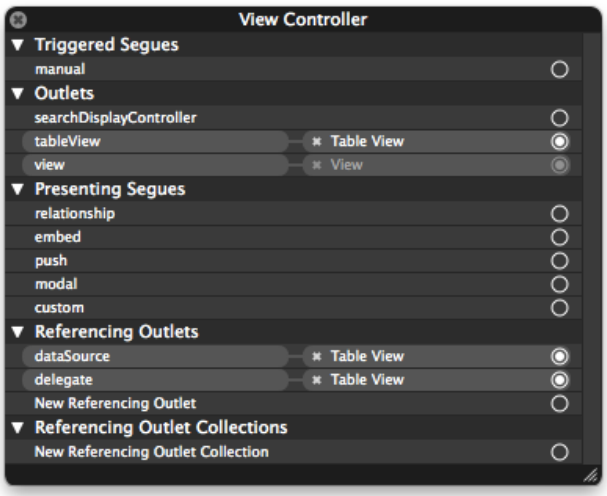
22. Try running the application and you will notice that the table is still empty. That's because we have not connect the delegate and datasource to the ViewController.

23. Go back to the **Main.storyboard** and select the table view. Right-click on the TableView. Notice that the **datasource** and **delegate** have yet to be connected to any outlet. Connect the outlet by clicking the small circle on the right side of the **datasource**. Keep your mouse button pressed and drag to the **View Controller** located at the outline view. Repeat it for delegate.

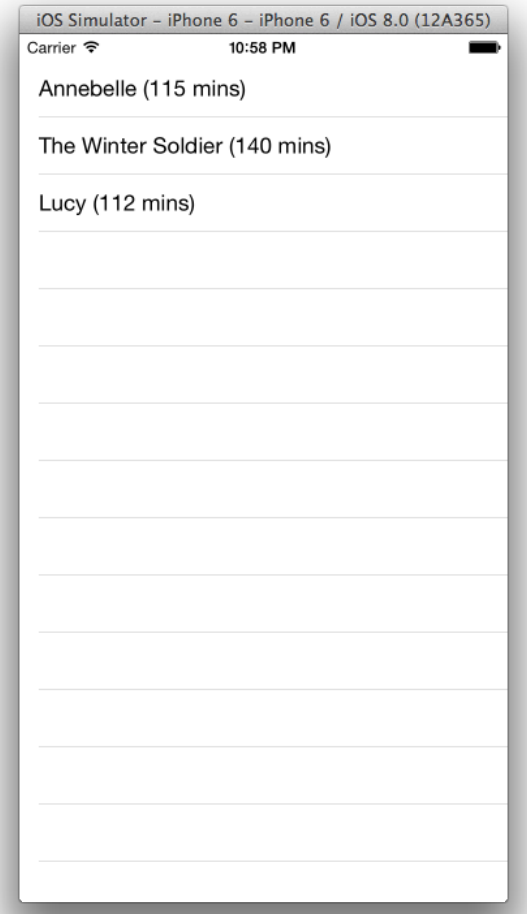


24. Next, connect the outlet **tableView** to the Table View in the View.

25. Ensure that the Connections Inspector window for the **View Controller** window looks as shown:



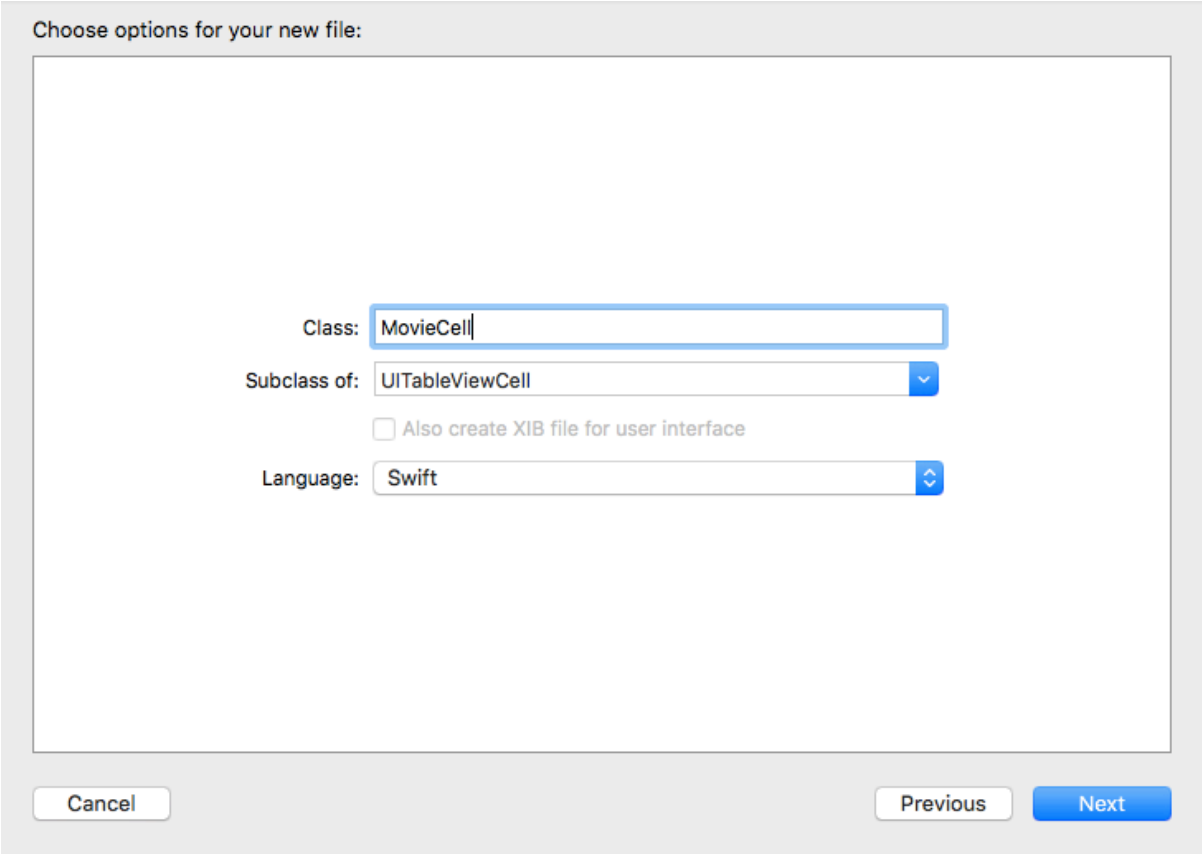
26. Build and run the application. You will see your table view populated with a list of movies.



Section 3: Customizing a Table Cell

In many iOS application available in iTunes, many do not use the standard table cell to display their contents. Their table views are customized. In this section, we are going to customize our table cell and create its graphical layout using Interface builder.

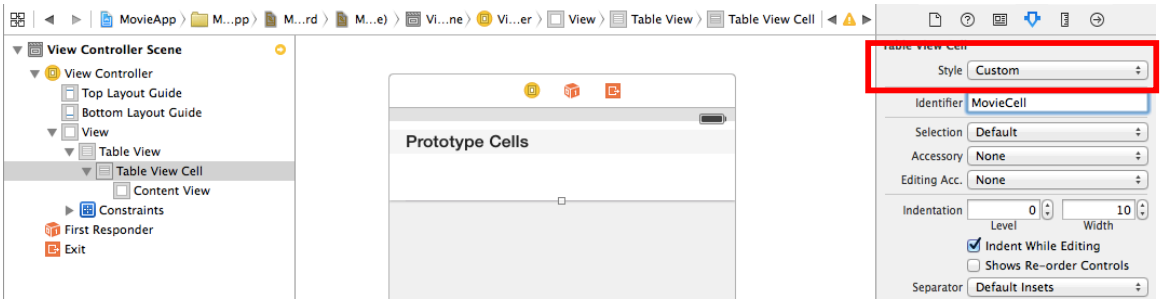
27. Right-click on the project in the project navigator and select **New File…**. Select **Cocoa Touch Class** class, click **Next** and name the class **MovieCell** with subclass **UITableViewCell**.



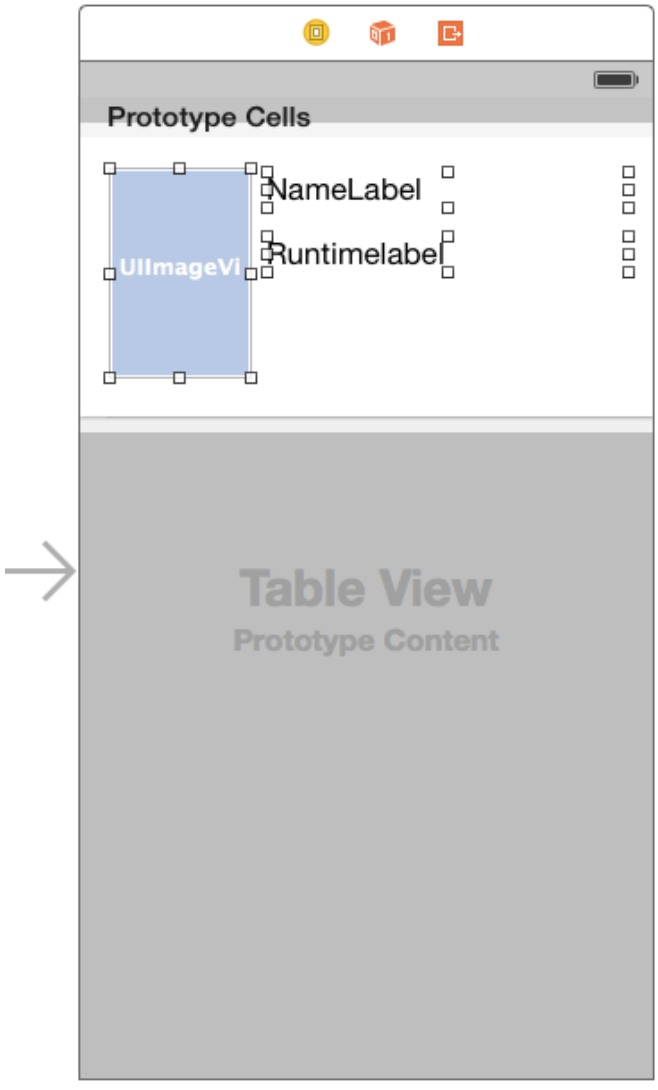
28. In the **MovieCell.swift,** create 2 new **UILabel** variables named **nameLabel** and **runTimeLabel** and a new **UIImageView** variable named **movieImageView**.

```
import UIKit

class MovieCell: UITableViewCell {
    @IBOutlet weak var nameLabel : UILabel!
    @IBOutlet weak var runTimeLabel : UILabel!
    @IBOutlet weak var movieImageView: UIImageView!


}
```

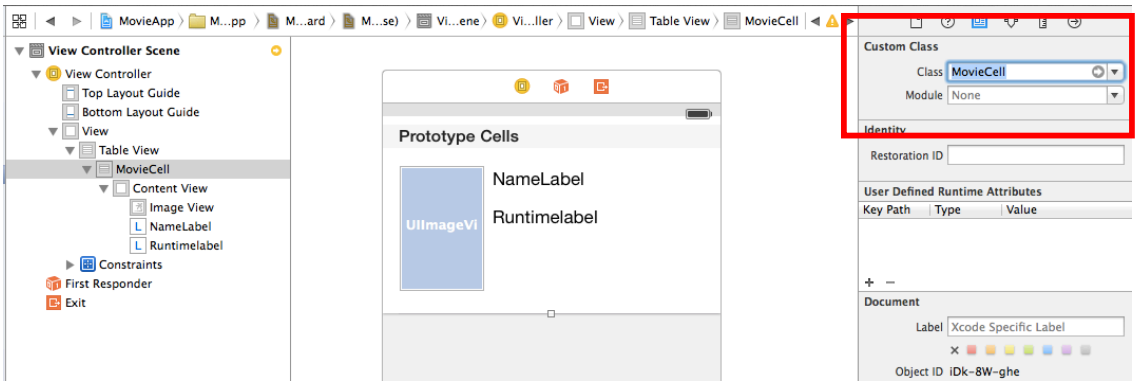
29. With the Table View Cell selected, under the **Attribute Inspector**, make sure the style of the Table View Cell is changed to **Custom.**



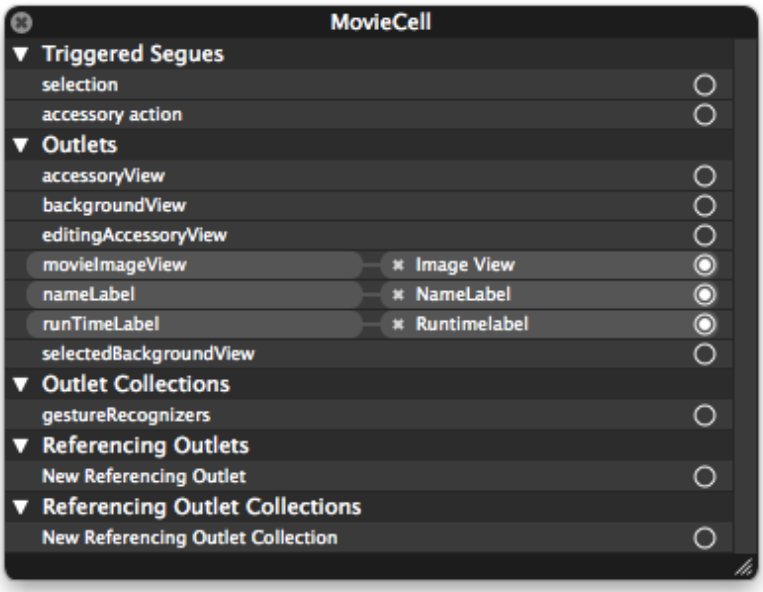
30. Design the table view cell with 2 **Labels** and 1 **Image View** as shown below. Note that the **width** of the table view is set to **320** and **height** of the table view cell is set to **140**. The dimension of the image view is **80 x 118**. You can change the size of the table view under the size inspector.



31. With the Movie Cell Selected, under the **identity inspector**, change the cell's class to **MovieCell**.



32. **Right-click** the movie cell on the **Outline View** and connect the outlets to the respective IBOutlet created in MovieCell. Ensure the **connections Inspector window** for Movie Cell looks as shown:



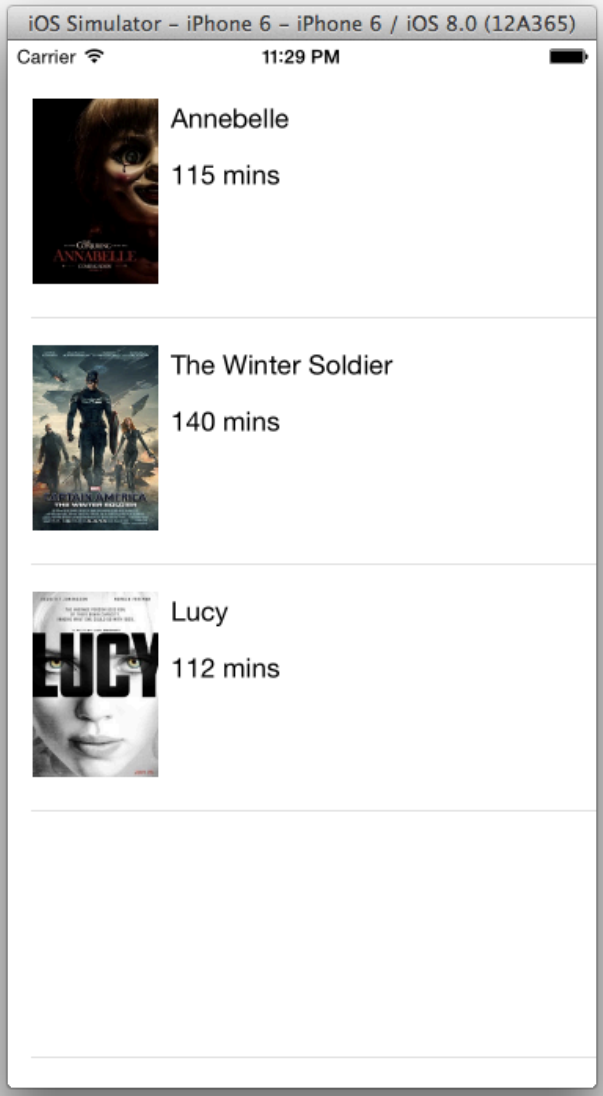
33. Save the changes.

34. Modify the **ViewController.swift** to use this newly created cell.

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


    func tableView(_ tableView: UITableView,
        cellForRowAt indexPath: NSIndexPath)
        -> UITableViewCell
    {
        // Dequeue a reusable cell.
        //
        var cell : UITableViewCell! =
            tableView
            .dequeueReusableCell (withIdentifier: "UITableViewCell")
        let cell : MovieCell = tableView
            .dequeueReusableCell (withIdentifier: "MovieCell",
```

```
            for: indexPath)
            as! MovieCell

        // Using the re-used cell, or the newly created
        // cell, we update the text label's text property.
        //
        let p = movieList[indexPath.row]
        cell.textLabel?.text = "\(p.movieName) (\(p.runtime) mins)"

        let p = movieList[indexPath.row]
        cell.nameLabel.text = p.movieName
        cell.runTimeLabel.text = "\(p.runtime) mins"
        cell.movieImageView.image = UIImage(named: p.imageName)

        return cell

    }
```

35. Build and run the program, we will notice that the table cells have been customized.



Section 4: Challenge

1. Modify the application to allow deletion of the row by the user.
   (Hint: Use the tableView(tableView: commitEditingStyle: forRowAtIndexPath:) function)
2. Add a Edit button at the top of the TableView to allow user to set the TableView into Edit mode.
3. Modify the application to allow the user to reorder the rows in Edit mode.
   (Hint: Use the tableView(tableView: moveAtIndexPath: toIndexPath:)
4. Modify the application to display the movies in two sections – one section with movies of runtime more than 120 mins, another for the rest.
   (Hint: numberOfSectionsInTableView(tableView:), tableView(tableView: numOfRowsInSection:), tableView(tableView: titleForHeaderInSection:))