Advanced Mobile Application Development
Week 3: Navigation Controllers

Countries
File | New Project
Single View app
iPhone
countries

Root view controller
Go into the MainStoryboard document outline and delete the view.
Drag a Table View out into the controller.
Go into the connections inspector and connect the dataSource and delegate to the View Controller icon.
Drag out a table view cell onto the view.
Select the Table View Cell and in the attributes inspector make the identifier "CellIdentifier".
Select the View Controller and go into the identity inspector and make sure the class is our ViewController class.

(Or you can do the same thing we did last time, replace the view controller with a table view controller.)

Now for this table view controller to be controlled by a navigation controller, with the controller selected go to Editor | Embed in | Navigation Controller.
This creates a navigation controller as the root view controller.
It also created a relationship segue from the navigation controller to the table view controller.

We want our class to be the controller so go into ViewController.swift and change its super class to `UITableViewController.`

Run it and you should see an empty table(and space for a navigation bar at the top). Although this looks the same as last time, we're going to be able to navigate from the cells to another view controller (once we add that).

Drag continents.plist into your app and make sure you have Copy items if needed checked.
It's a Dictionary of [String : [String]]

Before we get our table set up we're going to create a class for our data model.
File | New File
iOS Source
Swift File
Continents
Make sure it's saving to your project folder and the target is checked.

Create a class for our data model.
```
class Continents {
    var continentData = [String : [String]]()
    var continents = [String]()
}
```

Now let's get our table set up. This process will be similar to our last app.
In ViewController.swift create an instance of the Continents class to store our data
```
var continentList=Continents()
```

Now let's load our plist of countries into our app. This is the same code as last time.
```
    override func viewDidLoad() {
        super.viewDidLoad()
        //use a NSBundle object of the directory for our application to 
retrieve the pathname of qwordswithoutu1.plist
        let path = NSBundle.mainBundle().pathForResource("continents", 
ofType: "plist")
        //load the data of the plist file into the dictionary
        continentList.continentData = NSDictionary(contentsOfFile: path!) 
as! [String : [String]]
        //puts all the continents in an array
        continentList.continents = Array(continentList.continentData.keys)
    }
```

Now we'll implement the two required methods for the UITableViewDataSource protocol.
```
    //Number of rows in the section
    override func tableView(tableView: UITableView, numberOfRowsInSection 
section: Int) -> Int {
        return continentList.continentData.count
    }

    // Displays table view cells
    override func tableView(tableView: UITableView, cellForRowAtIndexPath 
indexPath: NSIndexPath) -> UITableViewCell {
        //configure the cell
        let cell = 
tableView.dequeueReusableCellWithIdentifier("CellIdentifier", forIndexPath: 
indexPath)
        cell.textLabel?.text = continentList.continents[indexPath.row]
        return cell
    }
```

Run it and you should see the continent data listed in the table view.

Detail view controller
Now we want to be able to select a continent and see a list of countries.
Go into MainStoryboard and drag a table view controller onto the canvas to the right to be the detail view controller.
Now we need a class to control it.
File | New | File
iOS | Cocoa touch class
DetailViewController subclass of UITableViewController
Once created double check that DetailViewController is a subclass of UITableViewController.
Also look to see that it's added methods stubs for all the table view methods we'll need.

Go back into the storyboard and make that the class for your new table view controller.

Before we forget, select the table view cell in the detail view controller and give it the identifier CellIdentifier.
Now let's make the segue from the master to detail view controller.
Cntrl-click from the master prototype cell and drag to the detail view controller.
When you release the mouse you will get a popup and must choose a show segue.
Select your new segue and in the attributes inspector give it the identifier countrysegue.
In the master view controller select the table view cell and in the attributes inspector change accessory to disclosure indicator. This indicates that selecting that row will bring up related data.
We're not going to set the title for this view in the storyboard but we're going to do it programmatically so it will say whatever continents' countries we're looking at.

If you run it at this point the controller will navigate, we just have to load the data.

In DetailViewController.swift create the array that will hold the countries and a variable to hold the selected continent.

```
var countries = [String]()
var selectedContinent = 0
```

Create an instance of our data model

```
var continentCountries = Continents()
changed to
var continentListDetail = Continents()
```

Now we need to set up the countries for the selected continent. We're going to do this in viewWillAppear instead of viewDidLoad because we need to do this every time the view appears. viewDidLoad will only be called the first time the view is loaded.

```
    override func viewWillAppear(animated: Bool) {
        continentListDetail.continents = 
Array(continentListDetail.continentData.keys)
        let chosenContinent = 
continentListDetail.continents[selectedContinent]
        countries = continentListDetail.continentData[chosenContinent]! as 
[String]
    }
```

Now let's update the delegate protocol methods. These should look familiar.

```
    override func numberOfSectionsInTableView(tableView: UITableView) -> Int 
{
        return 1
    }

    override func tableView(tableView: UITableView, numberOfRowsInSection 
section: Int) -> Int {
        // Return the number of rows in the section.
        return countries.count
    }
```

Uncomment this method

```
    override func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell
```
In the first line replace '`reuseIdentifier`' with '`CellIdentifier`'.

Where it says Configure cell replace it with:
```
        cell.textLabel?.text=countries[indexPath.row]
```

The last part is for the master view controller to send to the detail view controller which row was selected and which countries to show.
Go into ViewController.swift

We need it to tell the detail view controller which continents' countries it should show. The place to do this is prepareForSegue, as it's about to transition to the detail view controller.

```
    override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
        if segue.identifier == "countrysegue" {
            let detailVC = segue.destinationViewController as!
DetailViewController
            let indexPath = tableView.indexPathForCell(sender as!
UITableViewCell)!
            //sets the data for the destination controller
            detailVC.title = continentList.continents[indexPath.row]
            detailVC.continentListDetail=continentList
            detailVC.selectedContinent = indexPath.row
        }
    }
```

<u>Delete rows</u>
Now let's add the ability to delete countries.
In DetailViewController.swift in viewDidLoad uncomment the following line
```
self.navigationItem.rightBarButtonItem = self.editButtonItem()
```
(although you can add a bar button item called Edit in the storyboard, it will not automatically call the methods needed, so you should do it programmatically.)

Uncomment the following:
```
    override func tableView(tableView: UITableView, canEditRowAtIndexPath
indexPath: NSIndexPath) -> Bool {
        // Return NO if you do not want the specified item to be editable.
        return true
    }
```

Uncomment and implement
```
    override func tableView(tableView: UITableView, commitEditingStyle
editingStyle: UITableViewCellEditingStyle, forRowAtIndexPath indexPath:
NSIndexPath) {
        if editingStyle == .Delete {
            //Delete the country from the array
            countries.removeAtIndex(indexPath.row)
            let chosenContinent =
continentListDetail.continents[selectedContinent]
            //Delete from the data model
continentListDetail.continentData[chosenContinent]?.removeAtIndex(indexPath.
row)
            // Delete the row from the table
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation:
.Fade)
        } else if editingStyle == .Insert {
            // Create a new instance of the appropriate class, insert it
into the array, and add a new row to the table view
        }
    }
```

Move Rows

Now let's make the rows moveable. In DetailViewController.swift find the stub for this method and un-comment it.

```
    override func tableView(tableView: UITableView, canMoveRowAtIndexPath
indexPath: NSIndexPath) -> Bool {
        // Return NO if you do not want the item to be re-orderable.
        return true
    }
```

Now un-comment and implement the movement of the rows.

```
    override func tableView(tableView: UITableView, moveRowAtIndexPath
fromIndexPath: NSIndexPath, toIndexPath: NSIndexPath) {
        let fromRow = fromIndexPath.row //row being moved from
        let toRow = toIndexPath.row //row being moved to
        let moveCountry = countries[fromRow] //country being moved
        //move in array
        countries.removeAtIndex(fromRow)
        countries.insert(moveCountry, atIndex: toRow)
        //move in data model
        let chosenContinent =
continentListDetail.continents[selectedContinent]

continentListDetail.continentData[chosenContinent]?.removeAtIndex(fromRow)

continentListDetail.continentData[chosenContinent]?.insert(moveCountry,
atIndex: toRow)
    }
```

To move rows in the simulator click on the Edit button and then grab the 3 lines to move a row.