Advanced Mobile Application Development
Week 5: Split Views

HarryPotter
File | New Project
Master-Detail Application template
HarryPotter
Device: universal

Initial Setup
Run the app in the simulator to see all that has been done already. Make sure you switch between landscape and portrait mode so you can see the split view.

Look at the Storyboard to see everything that was created.

- A split view controller
- A navigation controller and table view controller for the master view controller (left side)
- A navigation controller and detail view controller (right side)
- Relationship segues from the split view controller to the master and detail view controllers
- The datasource and delegate have also been set for the table view

In AppDelegate.swift a lot of work has been done for us

```
application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool
```

- An instance of the split view controller class is created and made the root view controller of the window
- Assigns the last object in the splitviewcontrollers array of controllers to a navigation controller
- Sets the left bar button
- assigns the split view's delegate

```
splitViewController(splitViewController: UISplitViewController,
collapseSecondaryViewController secondaryViewController:UIViewController,
ontoPrimaryViewController primaryViewController:UIViewController) -> Bool
```

- this method performs the tasks related to the transition to a collapsed interface
- adjusts the primary view controller and incorporate the secondary view controller into the collapsed interface
- a compact horizontal size class (iPhone) will collapse the split view
- When the split view collapses what happens to the second view controller is up to the second view controller (return false)
- To remove the second view controller and have the first view controller as the only child of the split view controller return true. This case is so the master view controller is shown on the iPhone when the app launches
- guard statements are new in Swift 2 and are often used to catching invalid data. If the Boolean condition is false the else is executed
- return values
    - NO tells the split view controller to use its default behavior to try and incorporate the secondary view controller into the collapsed interface
    - YES tells the split view controller not to apply the default behavior, that you do not want the split view controller to do anything with the secondary view controller

This template created a MasterViewController class and a DetailViewController class. These represent, respectively, the views that will appear on the left and right sides of the split view.

MasterViewController.swift
MasterViewController is a subclass of UITableViewController and defines an instance for the detail view controller.

viewDidLoad()
adding the edit and + buttons to the navigation bar.
Setting the detailviewcontroller

viewWillAppear()
The value of `clearsSelectionOnViewWillAppear` is based on whether or not the split view is collapsed.
By default, UITableViewController is set up to deselect all rows each time it's displayed. That may be OK in an iPhone app, where each table view is usually displayed on its own; however, in an iPad app featuring a split view, you probably don't want that selection to disappear.

prepareForSegue(_, sender)
Handles the segue and has all the methods for the table view.

DetailViewController.swift
DetailViewController is a subclass of UIViewController
Defines the outlet detailDescriptionLabel for the label in the storyboard.
Creates a variable called detailItem where the view controller stores its reference to the object that the user selected in the master view controller. The code in the didSet block is called after its value has been changed and calls configureView(), another method that's generated for us. configureView() to update the label with the detail item.

Wow, a lot of the setup was done for us. Now let's get going on our app.

Data
Drag in the harrypotter.plist file we're using and remember to check Copy items if needed.
Look at the plist to understand what's in it and what the key/value pairs look like.

In MasterViewController we need an array that will hold our list of characters. Replace the one in the template. This matches the format of our plist.

```
var characters = [[String : String]]()
```

Then we need to load in our plist data in viewDidLoad
First remove(or comment out) the code that's in there to allow edits since we won't be doing that.

```
        //use a NSBundle object of the directory for our application to
retrieve the pathname of qwordswithoutu1.plist
        let path = NSBundle.mainBundle().pathForResource("harrypotter",
ofType: "plist")
        //load the data of the plist file into the dictionary
        let alldata = NSDictionary(contentsOfFile: path!) as! [String:
[[String : String]]]
```

```
        if alldata.isEmpty != true {
            characters = Array(alldata["characters"]!)
        }
```

We also need to remove some table view delegate and data source methods.
Delete/comment out these whole methods:

```
func insertNewObject(sender: AnyObject)

func tableView(tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath)
```

Now let's update the other table view data source methods (stubs)
We have 1 section so `numberOfSectionsInTableView` is fine

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

Update `func tableView(tableView: UITableView, canEditRowAtIndexPath indexPath: NSIndexPath) -> Bool{`

```
        return false
}
```

And now configure the appearance of the cell

```
    override func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("Cell",
forIndexPath: indexPath)
        let character = characters[indexPath.row]
        cell.textLabel!.text = character["name"]! as String
        return cell
    }
```

Lastly we have to update the prepareForSegue to pass the URL to the detail view controller.

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?){
        if segue.identifier == "showDetail" {
            if let indexPath = self.tableView.indexPathForSelectedRow {
                let character = characters[indexPath.row]
                let url = character["url"]! as String
                let name = character["name"]! as String
                let controller = (segue.destinationViewController as!
UINavigationController).topViewController as! DetailViewController
                controller.detailItem = url
                controller.title = name
                controller.navigationItem.leftBarButtonItem =
self.splitViewController?.displayModeButtonItem()
                controller.navigationItem.leftItemsSupplementBackButton =
true
            }
```

```
        }
    }
```

Save and Run. Tap the Master button in the upper-left corner to bring up a popover with a list of characters. Tap a character's name to display their Wikipedia page URL in the detail view.

Rotate to landscape and you'll see the characters in the table view in the master view.

Detail View

In the Master view controller change its title to "Characters" instead of it saying Master.
Now let's get the detail view working.
Remove the title 'Detail' so that doesn't show when the app first starts.
Go into the storyboard and move the existing label to the top just under the nav bar and update its constraints.(replace center Y constraint with top space to top layout 0)
Change it to say "Select a Harry Potter character". We will use this label to show the URL but before the user selects anything it will provide instructions.
Look in the connection inspector to see that it's already set up as an outlet called detailDescriptionLabel.
Then add a web view that fills up the rest of the view and make an outlet connection called webView.
In the attributes inspector for Scaling check Scales Page to Fit.
Add constraints so leading and trailing are to the edges (-16 to superview), and bottom (0 to superview). Top space is to the label(16 to top layout).

In DetailViewController.swift add a method to load the web page.

```
    func loadWebPage(urlString: String){
        //the urlString should be a propery formed url
        //creates a NSURL object
        let url = NSURL(string: urlString)
        //create a NSURLRequest object
        let request = NSURLRequest(URL: url!)
        //load the NSURLRequest object in our web view
        webView.loadRequest(request)
    }
```

Call loadWebPage() from configureView

```
    func configureView() {
        // Update the user interface for the detail item.
        if let detail: AnyObject = self.detailItem {
            if let label = self.detailDescriptionLabel {
                label.text = detail.description
                loadWebPage(detail.description)
            }
       }
    }
```

You should now be able to select a character and see their Wikipedia page in the detail.
To change the title of the back button for the popover in portrait mode change the Title for the master view controller in the Storyboard in its navigation controller.

Activity Indicator

Add an activity indicator in the middle and connect it as an outlet called webSpinner

---

Make it gray and check Hide When Stopped. (look at other styles)
For the activity indicator align horizontal and vertical centers with the web view.
Make sure in the document outline that the activity spinner is below the web view.
(look at the class reference for `UIWebView)`
Go into DetailViewController .swift and add the `UIWebViewDelegate` protocol. All the methods for this protocol are optional (look at class reference).

Then add the two methods for the activity indicator

```
//UIWebViewDelegate method that is called when a web page begins to load
    func webViewDidStartLoad(webView: UIWebView) {
        musicSpinner.startAnimating()
    }

//UIWebViewDelegate method that is called when a web page loads successfully
    func webViewDidFinishLoad(webView: UIWebView) {
        musicSpinner.stopAnimating()
    }
```

Another useful delegate method is `webView(webView, didFailLoadWithError)` which is called if the web page doesn't load (such as no internet connection).
We won't be implementing it in this app.

Don't forget to add app icons and a launch screen.
Note that for universal you need iPhone and iPad app icons. iPad are 76 1x and 2x, and 83.5 2x for iPad Pro.
