**Assignment 3 - Delegates, Data Sources, and Table Views (Lists)**

**Due Week 7**

**Chao Zhao 6555985**

**Task 1 - Separate Your Concerns**

Delegation is very important design patterns in iOS development. Delegation separates the connection amount different components. When an object runs, it does not know what data to display and how users operate on it. It needs other object help to perform the action. This can provide weak coupling between software components. Each part is easy to test and modify. The performance will increase significantly.

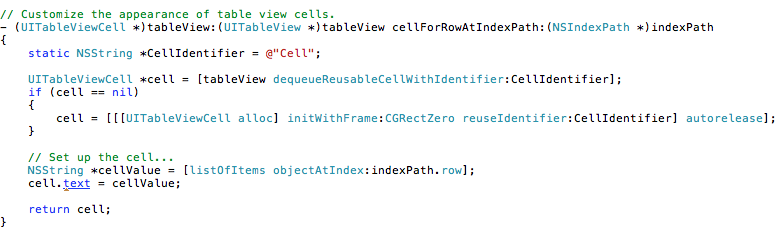
In this case, a UITableView object’s data source is switch to a different objective during runtime of application. A UITableView object does not know the data source formats. In the cell of the UITableView object, it may contain labels, checkboxes, sliders, images. With UITableViewDelegate and UITableViewDataSource helps, the UITableView object property can be configured and respond right feedback to the users.

The UITableViewDataSource protocol is used to configuring a table view, inserting or deleting table rows and recording table rows. There are two instance method is required to implemented: tableView:cellForRowAtIndexPath: and tableView:numberOfRowsInSection:

tableView:numberOfRowsInSection: instance method tells the UITableView object the number of rows to display. If the new data source is NSArray, we can use count instance method to specific the rows in the UITableView. tableView:cellForRowAtIndexPath: instance method returns a TableViewCell object. This method can configure and modified the cells in the UITableView. If the data source is containing some image, we can easily use tableView:cellForRowAtIndexPath: update the cell’s contents depends on the data source.

In the UITableViewDelegete protocol, there is no required instance method need to implement. This protocol is used to configuring rows for the table view, managing accessory views, managing selections, modifying the header and footer, editing table rows and recording table rows.

**Task 2.1 - Recycle, Reuse**



source: http://www.iphonesdkarticles.com/2009/01/uitableview-creating-simple-table-view.html

According to Apple’s Documentations, above is the cellForRowAtIndexPath implementation. This method will return a UITableViewCell object for each row. This method is used to draw the rows contents depends on the different data source. The dequeueResuableCellWithIdentifier method returns a UITableViewCell object. If the system has more resource, it can save some UITableViewCell objects in a queue. The UITableViewCell object with same identifier will be loaded, when an application need it. This method is to avoid build same style of UITableViewCell object. For instance, a user scroll the UITableView up and down, the system will load cell with same identifier from the queue and configure it with right property and display to the screen. Developer needs check the returning object a nil object. Sometimes, system has limited resource, it does not save these cell objects in the memory, so developer need check the returning object is available. If not, developer need use alloc and init to create a new tables. This can save lots of memory for a limited memory on a mobile device.

If developer uses alloc and init to create a new cell each time, it will slow down system.

**Task 2.2**

My algorithm is read the file to a NSArray object. And sort the array by sortedArrayByUsingFunction:context:. The first parameter is a c style function. It returns an NSInteger value. We can pass nil to second parameter. It can make a customize comparison between objects.

We will need to set up a *compare* function which takes two objects (of type id, since we can use cast to convert to NSString) and a *context* parameter.

This function returns an NSComparisonResult:

* It will return NSOrderedAscending, if lastName1 > lastNam2.
* It will return NSOrderedDescending, if lastName1 < lastNam2.
* It will return NSOrderedSame, if lastName1 == lastNam2.

I noticed that in the provided file, somebody has three names, like “Michael Robert Nyman”. So I use if statement to make sure that the middle name is not use to compare with others.

When the device is rotated, we need a method’s help:

-(void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromInterfaceOrientation.

This method is called automatically when the device is finished rotation. To detect the device’s orientation, we can use if statement to check property: interfaceOrientation. It has four values, they are:

* UIInterfaceOrientationPortrait
* UIInterfaceOrientationPortraitUpsideDown
* UIInterfaceOrientationLandscapeLeft
* UIInterfaceOrientationLandscapeRight

If the condition is matched, we can sort the array by our requirements. Finally, the [self.tableView reloadData]; should be call to reload data after sort array.

