ShoppingListVC Class Description

Draft 0.1

### Inherits from…

NSObject : UIResponder : UIViewController : ShoppingListVC

# Overview:

The ShoppingListVC Class provides core view controller functionality for a scene including a sectioned UITableView in either Grouped or Plain mode and other views outside of the UITableView of the developer’s choosing. By default, the scene is expected to be built using StoryBoard but this can be changed if ShoppingListVC is sub-classed. The UITableView may utilize any of the standard, built-in UITableCell styles or a custom style that is defined using the prototype cell in StoryBoard.

If provided with a managed object context, ShoppingListVC will display a complete table of ProductItem objects from the database in cells defined by the prototype in Storyboard and with cell.title.text=ItemName.

The major behaviors implemented by ShoppingListVC are:

* Standard behaviors of UITableViewController for the embedded table view:
  + If a nib file is specified via the initWithNibName:bundle: method (which is declared by the superclass [UIViewController](http://developer.apple.com/library/ios/documentation/uikit/reference/UIViewController_Class/Reference/Reference.html#//apple_ref/occ/cl/UIViewController)), UITableViewController loads the table view archived in the nib file. Otherwise, it creates an unconfigured [UITableView](http://developer.apple.com/library/ios/documentation/uikit/reference/UITableView_Class/Reference/Reference.html#//apple_ref/occ/cl/UITableView) object with the correct dimensions and autoresize mask. You can access this view through the tableView property.
  + If a nib file containing the table view is loaded, the data source and delegate become those objects defined in the nib file (if any). If no nib file is specified or if the nib file defines no data source or delegate, UITableViewController sets the data source and the delegate of the table view to self.
  + When the table view is about to appear the first time it’s loaded, the table-view controller reloads the table view’s data. It also clears its selection (with or without animation, depending on the request) every time the table view is displayed. The UITableViewController class implements this in the superclass method [viewWillAppear:](http://developer.apple.com/library/ios/documentation/uikit/reference/UIViewController_Class/Reference/Reference.html#//apple_ref/occ/instm/UIViewController/viewWillAppear:). You can disable this behavior by changing the value in the clearsSelectionOnViewWillAppear property.
  + When the table view has appeared, the controller flashes the table view’s scroll indicators. The UITableViewController class implements this in the superclass method [viewDidAppear:](http://developer.apple.com/library/ios/documentation/uikit/reference/UIViewController_Class/Reference/Reference.html#//apple_ref/occ/instm/UIViewController/viewDidAppear:). (This behavior is recommended in Apple’s UI Guidelines)
  + It implements the superclass method [setEditing:animated:](http://developer.apple.com/library/ios/documentation/uikit/reference/UIViewController_Class/Reference/Reference.html#//apple_ref/occ/instm/UIViewController/setEditing:animated:) so that if a user taps an Edit|Done button in the navigation bar, the controller toggles the edit mode of the table.
* Other behaviors:
  + (Needed?) Automatically scrolls the currently selected cell out of the way if the keyboard comes into view. (Uses keyboard notifications for this, see [Moving Content That Is Located Under the Keyboard](http://developer.apple.com/library/ios/#documentation/StringsTextFonts/Conceptual/TextAndWebiPhoneOS/KeyboardManagement/KeyboardManagement.html%23//apple_ref/doc/uid/TP40009542-CH5-SW7))
  + Out of the box – provides all table view data source and delegate methods necessary to *display* a sectioned table with title=ItemName and cells in whatever style is in the Storyboard prototype. Table content will be based on a database identified by the developer.
    - Sufficent UITableViewDataSource methods are implemented to perform this display function
    - No UITableViewDelegate methods are implemented (e.g., no tap actions are implemented)
    - If the developer wishes to add other behaviors that depend on unimplemented Data Source or Delegate methods, these must be added when sub-classing ShoppingListVC.

# Design Notes

## A word on the ProductItem class

This class may contain additional attributes but will contain *at least* ItemName, Section, and SectionIndex (an ordinal number that allows us to control the order of sections regardless of their alphabetical order).

This structure will allow us to use SectionIndex as the section title attribute for NSFetchedResultsController (FRC). FRC will thus interpret the list of numbers as the section titles and will put this list of numbers into its FRC.sections array.

We will then provide a second table with SectionIndex and SectionTitle so that we can translate the index number into the title as needed (see tableView:titleForHeaderInSection).

## Table View Data Source methods

### – tableView:cellForRowAtIndexPath:

* Returns a cell as defined in the prototype from StoryBoard
* Calls an internal initCellAtIndexPath: method with two parameters – indexPath and cell
  + Method gets data from fetchedResultsController and populates cell.title.text with ItemName
  + This method should be overridden when subclassing

### – numberOfSectionsInTableView:

* Uses NSFetchedResultsController; see class reference for example code for this method.

### – tableView:numberOfRowsInSection:  required method

* Uses NSFetchedResultsController; see class reference for example code for this method

### – sectionIndexTitlesForTableView:

* Not used (section index titles are used for the right hand nav, e.g., alpha references on Contacts)

### – tableView:sectionForSectionIndexTitle:atIndex:

* Not used (section index titles are used for the right hand nav, e.g., alpha references on Contacts)

### – tableView:titleForHeaderInSection:

* See A Word On The ProductItem Class for background on the pertinent attributes of ProductItem objects and why they are designed as they are.
* We will provide a table that matches SectionIndexes with SectionTitles.
* That table will be read into an in-memory array, SectionTitles, the first time this method is called (will check to see if the array has already been instantiated so we don’t recreate it constantly).
* When this method is called we will simply return [SectionTitles objectAtIndex: section]

### – tableView:titleForFooterInSection:

* Not implemented.

# Notes on Sub-classing

When sub-classing ProductItemVC, the following methods should be overridden:

* initCellAtIndexPath:
  + This method is called by ProductItemVC’s cellForRowAtIndexPath: method. By calling initCellAtIndexPath, it allows the developer to initialize a cell of arbitrary design and complexity without having to override the basic, static plumbing of cellForRowAtIndexPath.