iOS: Table Views and Navigation Controllers

Tommy MacWilliam

Harvard University

April 5, 2011

Announcements

iOS: Table
Views and
Navigation
Controllers

Tommy
MacWilliam

- Lectures: http://cs76.net/Lectures
- Sections: http://cs76.net/Sections

- Property Lists
- Table Views
- Navigation Controllers
- Advanced TableViews

- ▶ key-value pairs, used to store settings
- ▶ XCode plist editor
- ▶ stored as XML

plist

iOS: Table
Views and
Navigation
Controllers

Tommy
MacWilliam

Property Lists

Table Views

Navigation
Controllers

Advanced
TableViews

- `<key>name</key>`
- `NSString: <string>`
- `NSNumber: <integer>, <real>`
- `NSDate: <date>`
- `NSData: <data>`
- `NSArray: <array>`
- `NSDictionary: <dict>`

- `NSDictionary` also contains key/value pairs
- `initWithContentsOfFile` parses plist into an `NSDictionary`
- `writeToFile` writes plist from `NSDictionary`

plist

- `NSBundle`: location of resources in filesystem
- `mainBundle`: bundle for current app
- `pathForResource:ofType:` `NSString` of path to file of given type

- single-column tables, just like `ListAdapter`s
- bind data source (model) to UI (view)
- `UITableViewDataSource` protocol
    - or, just use the `UITableViewController`

TableViews

- ▶ `numberOfSectionsInTableView:` define how many sections the table has
- ▶ `tableView:numberOfRowsInSection:` define how many rows the table has
  - ▶ `count` of the model
  - ▶ effectively ignore section if you just have one
- ▶ `tableView:cellForRowAtIndexPath:` how to render a given cell
- ▶ `tableView:didSelectRowAtIndexPath:` handle tap on a cell

cellForRowAtIndexPath

iOS: Table
Views and
Navigation
Controllers

Tommy
MacWilliam

Property Lists

Table Views

Navigation
Controllers

Advanced
TableViews

- just as with `Adapter`s, we want to recycle/cache cells to improve performance
  - cell off-screen: we can re-use it

- cell identifier: unique string identifying the type of cell we want
  - more pertinent when using different types of cells in a table

- `dequeueReusableCellWithIdentifier`: try to retrieve cell from the cache
  - nothing in the cache until user scrolls!
  - if `nil`, nothing in cache, so create new cell

NSIndexPath

iOS: Table
Views and
Navigation
Controllers

Tommy
MacWilliam

Property Lists

Table Views

Navigation
Controllers

Advanced
TableViews

- contains property for `section` and `row`
  - integer, 0-indexed
- does NOT contain cell itself!
  - `[model objectAtIndex:indexPath.row];`

Cell Properties

iOS: Table
Views and
Navigation
Controllers

Tommy
MacWilliam

Property Lists

Table Views

Navigation
Controllers

Advanced
TableViews

- `cell.textLabel.text`: text displayed in cell
- `cell.accessoryType`: image at right of row, describes cell behavior
  - `UITableViewCellAccessory DisclosureIndicator`: chevron, designates drill-down
  - `UITableViewCellAccessory DetailDisclosureButton`: blue arrow button, designates detail view
  - `UITableViewCellAccessory Checkmark`: checkmark when row is selected

- ► style attribute
  - ► Plain: edge to edge rows
  - ► Grouped: centered with rounded corners
- ► outlets
  - ► `dataSource/delegate`: where methods defining table behavior are defined

- example time!

UINavigationController

- maintains a stack of view controllers
- `pushViewController:animated:` push a new controller onto the stack and switch to it
- `popViewControlerAnimated:` pop a view controller off the stack and switch to parent

UIViewController Properties

- `self.navigationController`: navigation controller managing the current stack
- `self.navigationItem`: navigation bar for current view controller
  - `title`: text in navigation bar
  - `leftBarButton`, `rightBarButton`: buttons on navigation bar

- don't forget about `@class` (interface) and `#import` (implementation)!
  - can't access classes you've created without them

Pushing a new ViewController

- from Interface Builder
    - create outlet in interface
    - add a generic `View Controller` object
    - set class in Identity Inspector
    - set nib name in Attributes Inspector
    - connect `View Controller` to `File's Owner`

- from Objective-C
  - `alloc` and `initWithNibName:` new instance of the controller
    - no file extension on the nib
  - don't forget to `release`!

UIImageView

- ▶ `image` property of `UIImageView` changes `UIImage` displayed
  - ▶ `UIImage imageNamed:` create `UIImage` from filename
- ▶ `contentMode` property of `UIImage` controls how image is scaled
  - ▶ `UIViewContentModeScaleToFill`: scale to fit, changing aspect ratio
  - ▶ `UIViewContentModeScaleAspectFit`: scale to fit, maintaining aspect ratio
  - ▶ `UIViewContentModeScaleAspectFill`: scale to fit, cropping if necessary

- `loadRequest`: load web page from `NSURLRequest`
    - `NSURLRequest requestWithURL`: create `NSURLRequest` from `NSURL`
    - `NSURL URLWithString`: create `NSURL` from `NSString`

- example time!

TableView Sections

- ▶ can have different number of rows per section
    - ▶ remember, `numberInRowsInSection:` gives the section as a parameter
- ▶ `titleForHeaderInSection:` NSString of section header
- ▶ `titleForFooterInSection:` NSString of section footer

TableView Sections

- example time!

Editable TableViews

- ► `self.editItemButton`: built-in button handling delete/move
- ► `canEditRowAtIndexPath`: set whether a given row is editable
  - ► `return YES` makes entire table editable
- ► `canMoveRowAtIndexPath`: set whether a given row is movable

▶ `commitEditingStyle:forRowAtIndexPath`: edit operation performed on given row
▶ `moveRowAtIndexPath:toIndexPath`: move operation performed on rows
▶ update both model and view!
  ▶ invalidate state can lead to app crash

- example time!

Cell Styles

- ▶ notice the constructor for cells is `initWithStyle:`
  - ▶ `UITableViewCellStyleDefault`: no detail label
  - ▶ `UITableViewCellStyleSubtitle`: detail label below
  - ▶ `UITableViewCellStyleValue1`: detail label to the right
  - ▶ `UITableViewCellStyleValue2`: equally-sized text and detail labels

Cell Styles

- example time!