Homework Assignment #3: USA Table View

Summary

You will create an application which lists the 50 United States and provides more detail about each. The user will also have the ability to search within the list of states.

This assignment is meant to demonstrate your understanding of:

- Various aspects of displaying data in UITableViews.
- Searching within a list and presenting the results to the user.
- Miscellaneous data loading and display techniques.

Functional Requirements

Start with a new Single View Application project.

In addition to a simulator build of the application, the downloadable archive “hw3_files.zip” contains the states.plist file you will use, as well as flag images in two sizes. You are NOT to rename or otherwise edit these files: Part of the homework is to figure out how to map your available data source to the assets provided and maintained by your “graphics department.”

For the table view controllers, you may choose to use either a UITableViewController subclass or your own UIViewController subclass containing a UITableView. There is no penalty for choosing one over the other.

While handling autorotation is part of iOS craftsmanship, no points are specific to these behaviors in this assignment.

Do not worry about the fact that many of the small flags “disappear” due to white-on-white backgrounds. Hopefully you’ll eventually work with better graphics designers than I happen to be.

States List View

The initial view the user encounters is a list of US states and their capitals, with a thumbnail of the respective state flag. When needed, load the states data from states.plist and transform it to an in-memory data model object instead of using the loaded dictionaries directly.

- The title in the navigation bar should read "States".
- The table view should be plain style.
- Each row will show the state's small flag image and the state name on the left, with the state capital on the right.
  - The row should be tall enough to not need to resize the small 50x50 flag image.
- When any state row is tapped, it should highlight blue and push a detail view for the selected state.
- Present each state in a section based on its starting initial.
  - Within each section, states should be alphabetized.
- Provide a functioning section index along the right side of the view.
  - All letters of the alphabet should be present.
  - Touching a letter in the section index should jump the table to the corresponding section.
  - According to Apple's guidelines, rows should not have a disclosure indicator when a section index is present.

State Detail View

The main detail view consists of a grouped table view. None of these rows are selectable by the user, and should not show a highlight when tapped.

- The title should be the selected state's name, followed by its abbreviation in parentheses.
- The back button should read "States".
  - Pressing the back button should take the user back to the states list, where the selected row should then deselect.

- The table header is a UIImageView displaying the large (200px wide) version of the state flag.
    - The images should not be resized.
- The table view has three sections.
    - The first table section displays the Capital and Largest cities. It has a section header reading “Cities”.
    - The second section displays the date of statehood in an abbreviated yet readable format, like “Jan 3, 1959.”
    - The third section displays the population and area in square miles (the value provided is already in square miles, so no need to convert). Format the number with thousands separators.
- A table footer should read “Population and Largest City based on 2010 Census Data.”

Search

Add a functioning search bar to the top of the main list table view.

- When searching, the table view needs to change some of its decorations to maximize the display space left by the search bar and keyboard.
    - No search index should be displayed.
    - Each row should display a disclosure indicator.
    - The results should be displayed in a single section, with no section header.
- Add a “search” magnifying glass icon to the top of the search index.
    - Tapping this should expose the search bar if not currently visible.
- While searching, the table should update with alphabetical matching records as the user types. A state matches if it contains the search string anywhere within its name.
- Selecting any row should push the detail view for the selected state.

General Requirements

- All your custom view controllers must load their views from XIBs.
- The app must not crash.
- You must not use any private API.
- You must follow Apple's Coding Guidelines for Cocoa.
- The app must not leak memory or other resources (such as sockets, file handles, ports, etc.).
- The app must compile with zero compiler warnings and zero warnings from the static analyzer.

BONUS Opportunities

- 10 points: In the states list, remove any sections which do not have any rows. These entries should still be present in the section index, though. Tapping a "missing" section in the section index should take the user to the alphabetically preceding section which *does* have an entry (i.e., "B" would move you to "A"; "E" to "D"; and "X", "Y", and "Z" to "W").
- 10 points: Add scope bar capabilities to the search display controller, providing the user with the ability to search within "Name", "Abbrev", "Capital" or "All" scopes. When selected, each of the first three scopes should only search results within the state name, abbreviation or capital city's name, respectively. When "All" is selected, the search results should contain states matching the search string in any of those three fields.

Submitting Homework and Due Date

You must submit your project source code, including the Xcode project file, all nibs, and all other resources, in a single zip archive by Wednesday, February 13 at 11:55 PM Pacific Time. Homework should be submitted to the CollectIt Dropbox linked from the course page.