Roman Arriaga
Final Test.

Friday, October 11, 2019

When implementing MVVM, the UI would have to be able to update its values accordingly in every window automatically, without having to resort to delegates or notification center observers, but this binding problem is not handled by swift natively.

MVVM is an effective architecture which is used by a vast majority of programmers, however it has certain disadvantages when it comes to data binding, one of them is that it does not update the UI accordingly to any value changed unless it's told to do so. This can create performance issues among programmers, as well as messy and cluttered code, unless third party frameworks are used, it's not an issue that affects small and simple apps, but fixing this issue will not only improve the development time in the long run, but it will also improve scalability of the code.

Because data binding, is pretty much an observer/listener pattern, there will be a swift file for the observer, and another one for the listener, no third party frameworks will be used, instead it will be accomplished by 2-way binding.

The UI

There will be 3 view controllers.

The first view controller will contain a text input field at the top, under it will be a table view container, which will take over most of the view, and under it a label with multiple lines, and the more lines it has, it will shrink the table view above it to make space for itself. And at the very bottom, a button.

The text input field will be used to contain text that will be updated on the second view controller. The table view is mostly to show arrays, or dictionaries, tapping on a cell in the table view will load the third view controller, the label under the table view will be gathering data from the second view controller, and the button will load the second view controller. The button will NOT set the text input field's text to the next controller, it will just load the second view.

The second view controller will have a label to display the text modified from the first view controller, including a numerical value, and a boolean value, a slider used to modify a numeric value, a switch used to modify a boolean value, and a button used to go back to the first screen.

The third view controller contains a table view, which will take most of the screen, and a button at the bottom. The table view contains the values of the array or the dictionary that was selected on the first table view. If the row contains another array or another dictionary, selecting it will load a duplicate of the third view controller and on its table, it will display the values of the selected array or dictionary. The button at the bottom will be used to navigate back to the previous screen.

The Model

The two main aspects of two-way-binding is are the observers, and the bound data. The observer's function will be to notify all other observers whenever its value has been set, and the bound data is what will allow us to have our two-way setup.

The roadmap

Friday 11, October 2019
-Start of the project
-Set up the UI
-Created navigation controls to move to the different UI windows.

Sunday 13, October 2019
-Created the observer class.

Monday 14, October 2019
-got stuck on the binding class. Will attempt to finish.