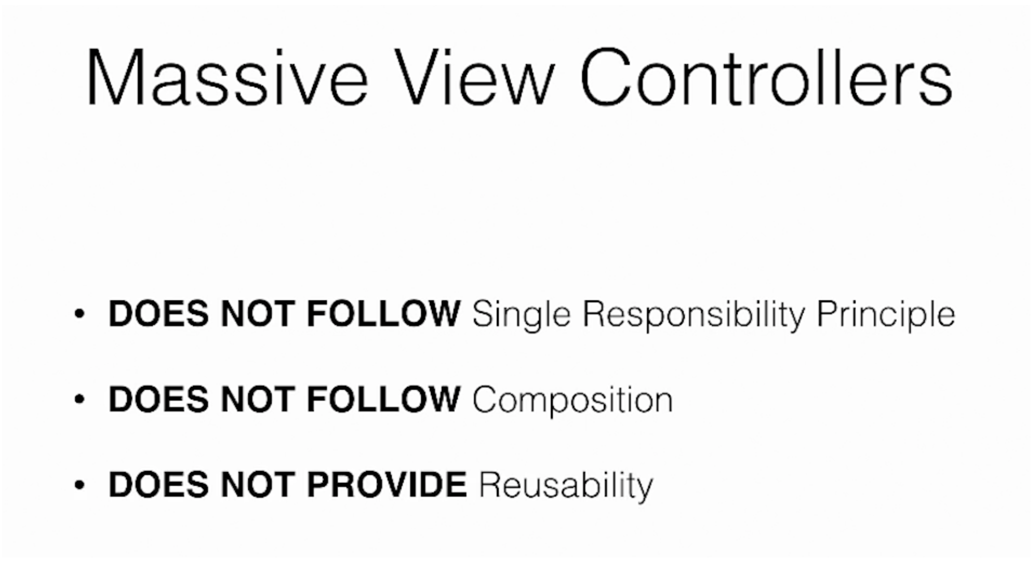
iOS Lean Controllers: 1 Setup, Persistent Data, and Implementation

16-Nov-2021

Course Link - LinkedIn

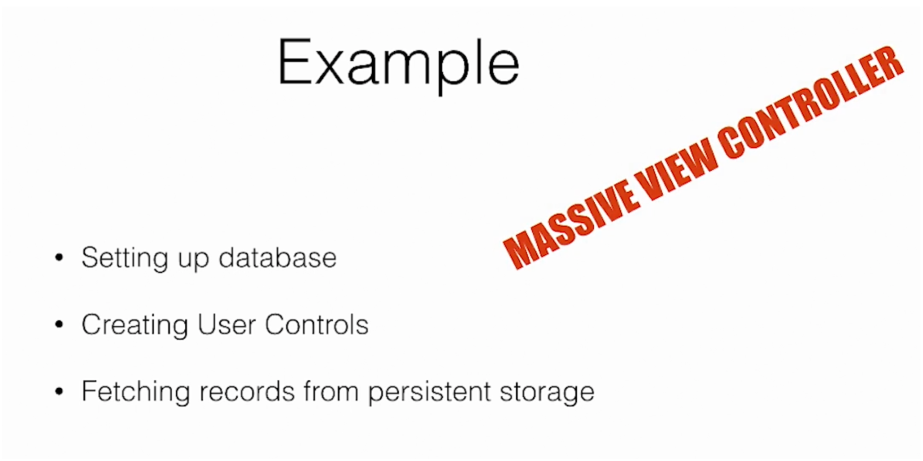
1. **Getting Started**
   - **Massive view controllers**



Massive View Controllers

- **DOES NOT FOLLOW** Single Responsibility Principle
- **DOES NOT FOLLOW** Composition
- **DOES NOT PROVIDE** Reusability

-



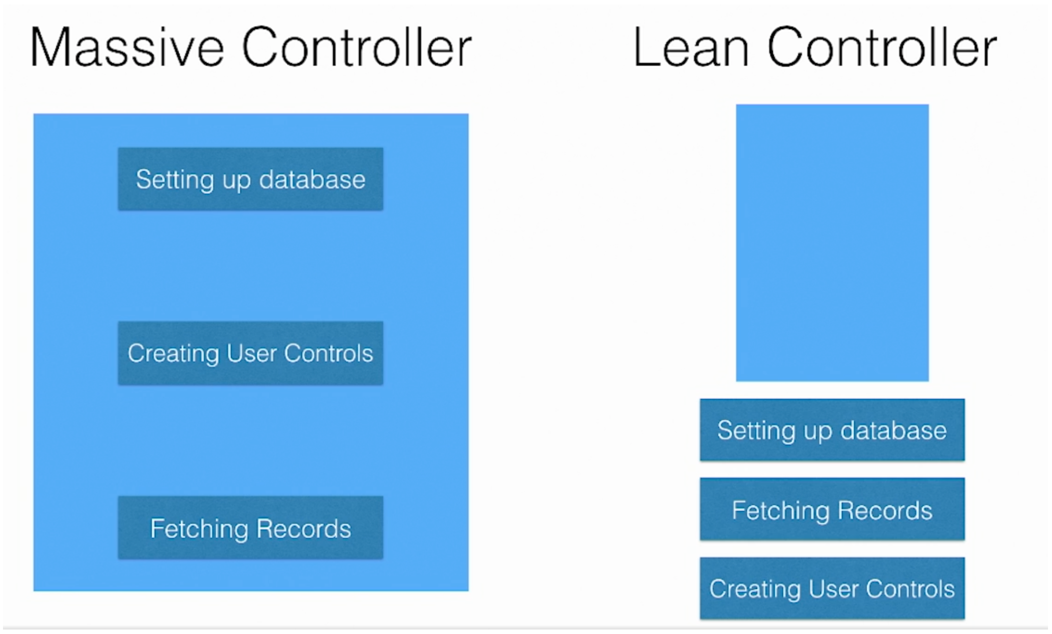
Example

- Setting up database
- Creating User Controls
- Fetching records from persistent storage

MASSIVE VIEW CONTROLLER

-

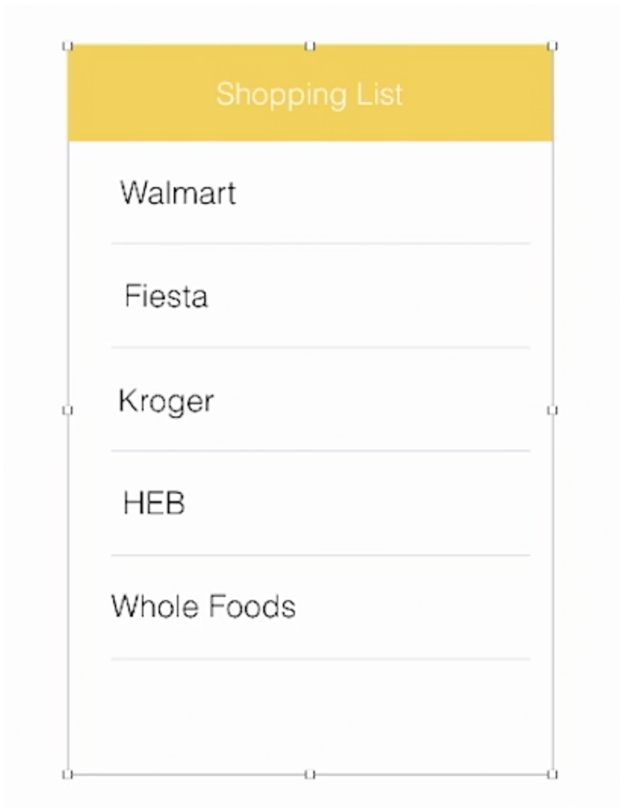
- **Lean controller**



-

2. **Setting Up the Grocery Application**

- **Designing wireframes using Keynote**

- 

- **Implementing the user interface in storyboards**
  - Added a tableViewController in main.storyboard and embedded in a Navigation controller
  - Created ShoppingListsTableViewController
- **Integrating with Core Data**
  - Set up Core Data with ShoppingListsTableViewController
  - Created MyGroceryDataModel.xcdatamodelId
- **Creating a custom view to add a new shopping list**
  - Added header section to add items to the list
  - Used tableView delegates methods viewForHeaderInSection and heightForHeaderinSection

3. **Persisting data using Core Data**

- **Saving New Record**
  - textFieldShouldReturn delegate for TextView

    - Save entered values to core data
- **Fetching and displaying records in UITableView**
    - Set fetchResultController and confirm with NSFetchResultControllerDelegate
    - Set table view rows and cellForRow
- **Deleting records**
    - Add tableView delegate method commit editingStyle and add delete action from core data
    - Handle fetchResultController didChange delegate for inserting and deleting

4. Implementing Data managers and Providers

- **Implementing Core Data manager**
    - Add CoreDataManager class and move initializeCoreDataStack() and managedObjectContext into it
    - Initialize Core Data and pass managedObjectContext from AppDelegate to ShoppingListsVC
    - Handle ShoppingListsVC
- **Implementing shopping list data providers**
    - Create ShoppingListDataProvider class and move fetchResultController logic from ShoppingListsVC

5. Implementing and Configuring Data sources

- **Implementing shopping list data sources**
    - Create ShoppingListDataSource class and move tableView data source methods from ShoppingListsVC
    - Handle ShoppingListDataSource with ShoppingListDataProvider

- **Communicating between the data provider and data source**
    - Pass tableView into ShoppingListDataSource class
    - Create ShoppingListDataProviderDelegate protocol for communication
- **Deleting shopping lists using the data source and provider**
    - Add delete method in ShoppingListDataProviderDelegate protocol
    - Handle delete object in ShoppingListDataSource and ShoppingListDataProvider classes

iOS Lean Controllers: 2 Controls, Views, Extensions, and Networking

17-Nov-2021

Course Link - LinkedIn

1. **Creating Custom Controls**

- **Creating a custom add new item control**
    - Create AddNewItemView and move header code from ShoppingListsVC
- **Adding the custom initializer to configure placeholder text**
    - New Initializer for AddNewItemView with controller and placeholder as parameters
- **Passing data from AddNewItemView using Delegates**
    - Create AddNewItemViewDelegate and handle save new entered shopping list item
- **Passing data from AddNewItemView using closures**
    - Pass an escaping closure with initializer and handle save new entered shopping list item

2. **Generic Data Providers and Data Sources**

- **Creating a generic data provider**
    - Create FetchedResultsDataProvider and FetchedResultsDataProviderDelegate
- **Implementing a generic TableView data source: Part 1**
    - Create TableViewDataSource class
- **Implementing a generic TableView data source: Part 2**
    - Set up TableViewDataSource with ShoppingListsVC with closure
- **Saving records using generic providers and data sources**
    - Handle insert and delete actions

*****Finished MyGrocey App*****

Additional Topics

3. **Building Better View Controller Segues**

- **Default segues**
  - Using prepare(for segue: UIStoryboardSegue, sender: Any?) delegate method
- **Modern segues using extensions**
  - Set a protocol segueHandler for handling segues with enum

4. **Secure TabBar items using protocol extensions**

- Create a LoginHandler protocol
- Create BaseTabBarController and confirm with UITabBarControllerDelegate and handle with LoginHandler
- Confirm Tabs controllers with LoginHandler to secure for login cases

5. **Building UIControl Extensions**

- Move button creation into UIButton extensions
- Add UIView extension to add Layout constraints
-