

**060010815:**  
**iOS Application Development**

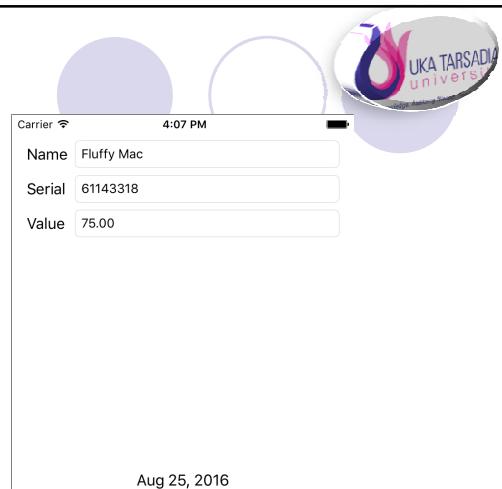
**Navigation, Touch and Gesture**

Dharmendra Bhatti

1

**Homepwner**

- Homepwner with stack views



Dharme

2

## Homepwner



- Main.storyboard => View Controller from the object library onto the canvas.
- Drag a Vertical Stack View from the object library onto the view for the View Controller.
- Add constraints to the stack view to pin it to the leading and trailing margins, and pin the top and bottom edges to be 8 points from the top and bottom layout guides.

Dharmendra Bhatti

3

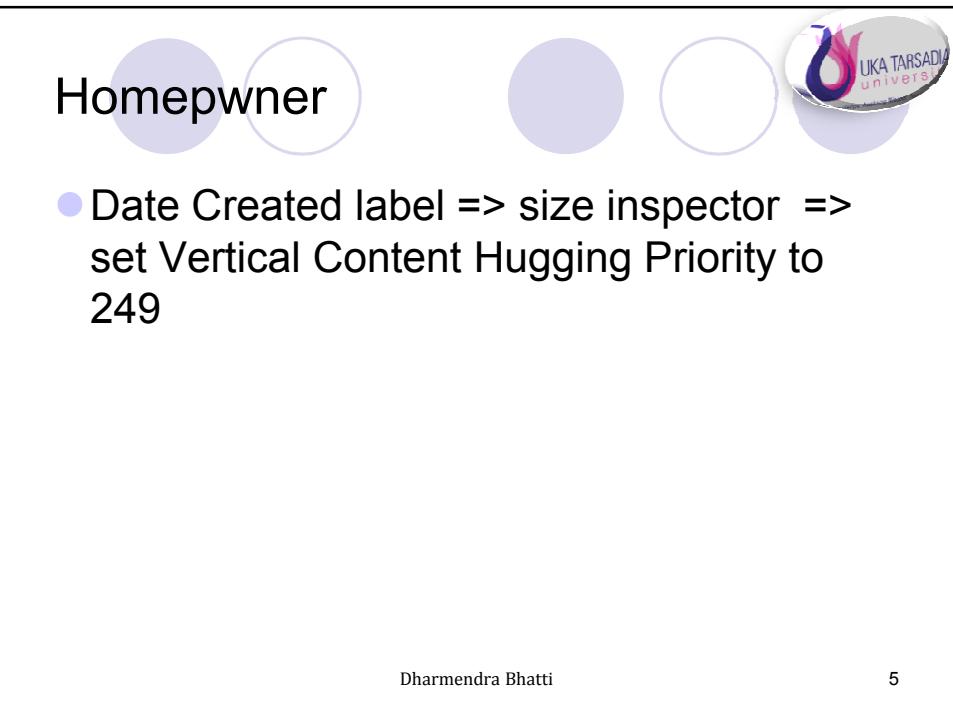
## Homepwner

- Labels added to the stack view

Dharmendra Bhatti

Serial  
Value  
Date Created

Name

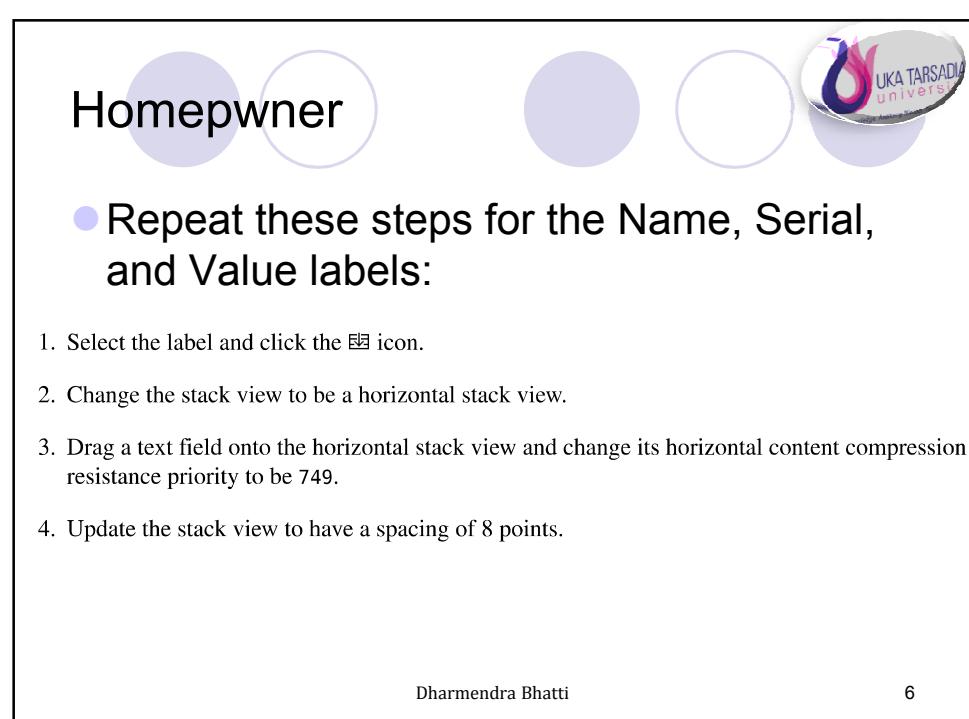


Homepwner

- Date Created label => size inspector => set Vertical Content Hugging Priority to 249

Dharmendra Bhatti

5



Homepwner

- Repeat these steps for the Name, Serial, and Value labels:

1. Select the label and click the  icon.
2. Change the stack view to be a horizontal stack view.
3. Drag a text field onto the horizontal stack view and change its horizontal content compression resistance priority to be 749.
4. Update the stack view to have a spacing of 8 points.

Dharmendra Bhatti

6

## Homepwner

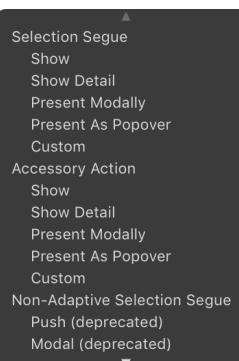
- vertical stack view => attributes inspector => set Spacing to 8 points.
- Date Created label => attributes inspector => change the Alignment to be centered
- Control-drag from the Name text field to the Serial text field and select Leading.
- Then do the same for the Serial text field and the Value text field.

Dharmendra Bhatti

7

## Homepwner

- Main.storyboard => **ItemCell** prototype cell => Control-drag from the cell to the new view controller => Selection Segue => Show



Dharmendra Bhatti

8

## Homepwner

- Create a new Swift file and name it DetailViewController

```
import Foundation
import UIKit

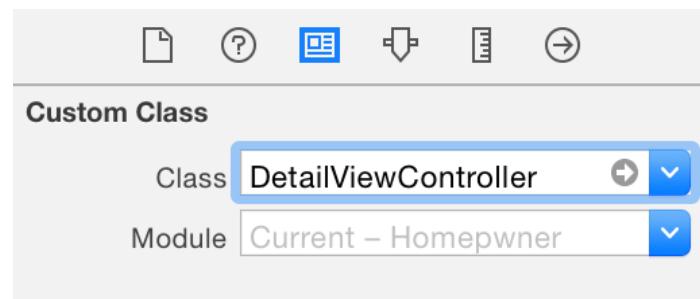
class DetailViewController: UIViewController {
```

Dharmendra Bhatti

9

## Homepwner

- Select the View Controller on the canvas and open its identity inspector.
- Change the Class to be DetailViewController



10

## Homepwner

- Toggle the assistant editor by clicking the middle button from the Editor control at the top of the workspace.
- Command-Option-Return
  - The shortcut to display the assistant editor
- Command-Return
  - The shortcut to return to the standard editor

Dharmendra Bhatti

11

## Homepwner

- Dragging from storyboard to source file

```
class DetailViewController: UIViewController {

    @IBOutlet var nameField: UITextField!
    @IBOutlet var serialNumberField: UITextField!
    @IBOutlet var valueField: UITextField!
    @IBOutlet var dateLabel: UILabel!

}
```

12

# Homepwner

```
class DetailViewController: UIViewController {

    @IBOutlet var nameField: UITextField!
    @IBOutlet var serialNumberField: UITextField!
    @IBOutlet var valueField: UITextField!
    @IBOutlet var dateLabel: UILabel!

    var item: Item!

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)

        nameField.text = item.name
        serialNumberField.text = item.serialNumber
        valueField.text = "\(item.valueInDollars)"
        dateLabel.text = "\(item.dateCreated)"
    }
}
```

Dharmendra Bhatti

13

# Homepwner

## • Use number formatter

```
let numberFormatter: NumberFormatter = {
    let formatter = NumberFormatter()
    formatter.numberStyle = .decimal
    formatter.minimumFractionDigits = 2
    formatter.maximumFractionDigits = 2
    return formatter
}()

let dateFormatter: DateFormatter = {
    let formatter = DateFormatter()
    formatter.dateStyle = .medium
    formatter.timeStyle = .none
    return formatter
}()

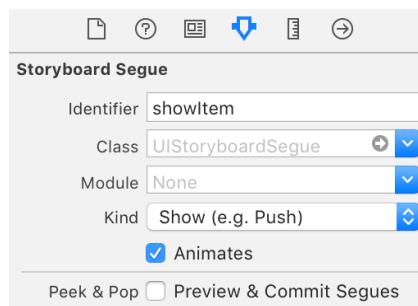
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)

    nameField.text = item.name
    serialNumberField.text = item.serialNumber
    valueField.text = "\(item.valueInDollars)"
    dateLabel.text = "\(item.dateCreated)"
    valueField.text =
        numberFormatter.string(from: NSNumber(value: item.valueInDollars))
    dateLabel.text = dateFormatter.string(from: item.dateCreated)
}
```

Dharmendra Bhatti

## Homepwner

- Main.storyboard again => click on the arrow between the two view controllers => open the attributes inspector => set identifier = showItem



15

## Homepwner

- ItemsViewController.swift

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    // If the triggered segue is the "showItem" segue
    switch segue.identifier {
        case "showItem"?:  

            // Figure out which row was just tapped
            if let row = tableView.indexPathForSelectedRow?.row {  

                // Get the item associated with this row and pass it along
                let item = itemStore.allItems[row]
                let destinationViewController
                    = segue.destination as! DetailViewController
                destinationViewController.item = item
            }
        default:  

            preconditionFailure("Unexpected segue identifier.")
    }
}
```

Dharmendra Bhatti

16

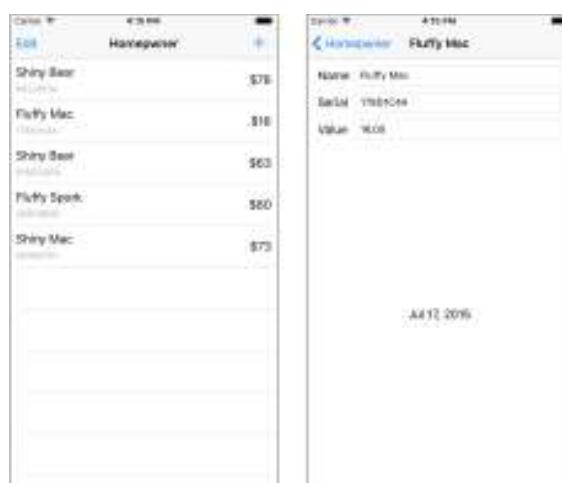
## Drill-down interface in Settings



Dharmendra Bhatti

17

## Homeowner with UINavigationController



Dharmendra Bhatti

18

## UINavigationController

- **UINavigationController**

- is a subclass of **UIViewController**
  - maintains an array of view controllers presenting related information in a stack

Dharmendra Bhatti

19

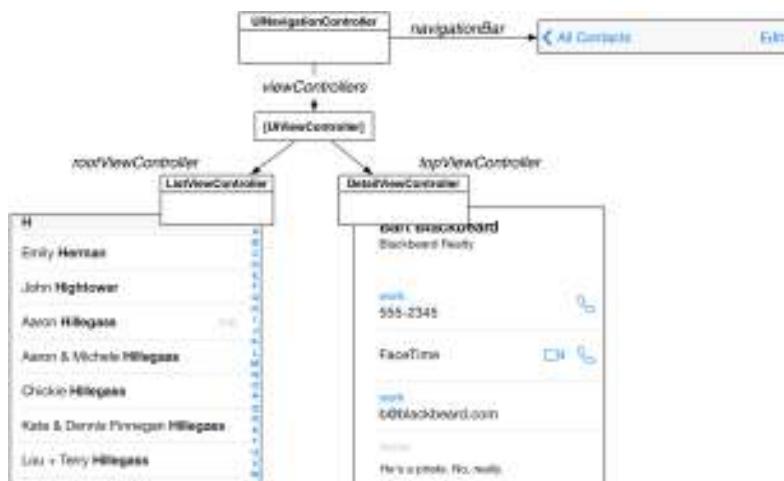
## UINavigationController

- view controller is pushed onto the stack
  - view slides onscreen from the right
- When the stack is popped (i.e., the last item is removed),
  - the top view controller is removed from the stack
  - its view slides off to the right, exposing the view of the next view controller on the stack, which becomes the top view controller

Dharmendra Bhatti

20

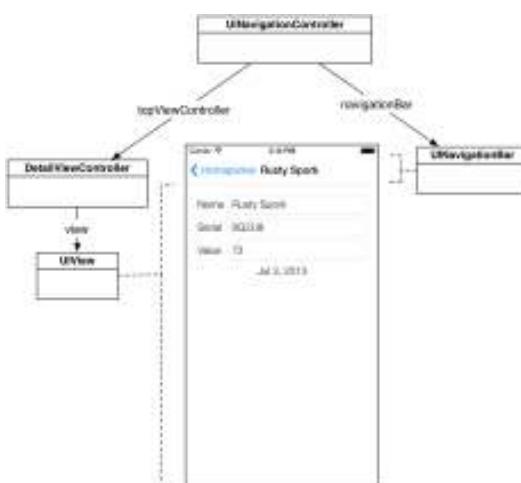
## UINavigationController's stack



Dharmendra Bhatti

21

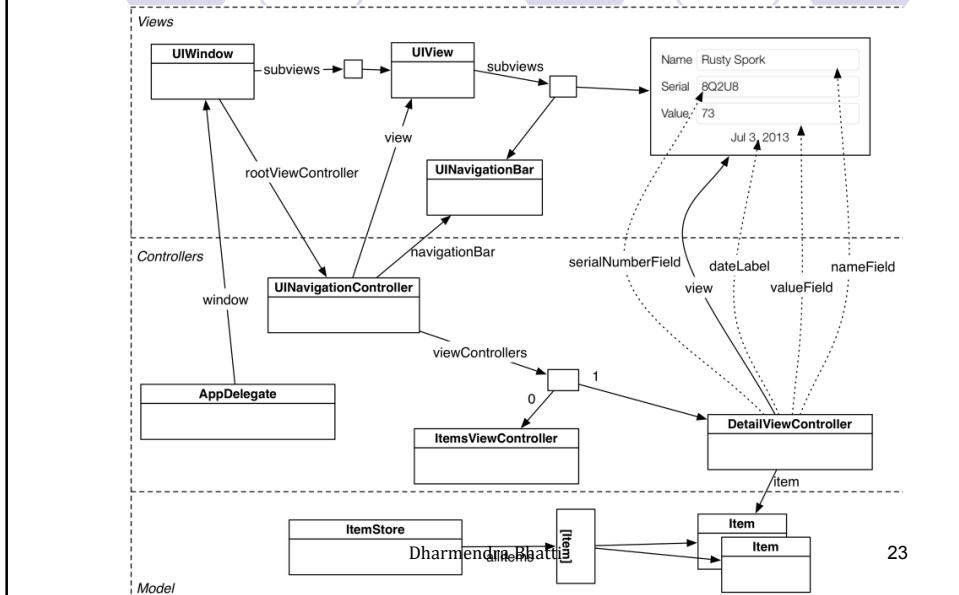
## UINavigationController's view



Dharmendra Bhatti

22

## Homeowner object diagram



23

## Homeowner

- Main.storyboard => select the Items View Controller => Editor menu => Embed In → Navigation Controller

Dharmendra Bhatti

24

# Homepwner

## • AppDelegate.swift

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey : Any]?) -> Bool {
    // Override point for customization after application launch.

    // Create an ItemStore
    let itemStore = ItemStore()

    // Access the ItemsViewController and set its item store
    let itemsController = window!.rootViewController as! ItemsViewController
    let navController = window!.rootViewController as! UINavigationController
    let itemsController = navController.topViewController as! ItemsViewController
    itemsController.itemStore = itemStore

    return true
}
```

Dharmendra Bhatti

25

# Homepwner

## • In DetailViewController.swift

```
override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)

    // "Save" changes to item
    item.name = nameField.text ?? ""
    item.serialNumber = serialNumberField.text

    if let valueText = valueField.text,
       let value = numberFormatter.number(from: valueText) {
        item.valueInDollars = value.intValue
    } else {
        item.valueInDollars = 0
    }
}
```

Dharmendra Bhatti

26

## Homepwner



- values of the **Item** will be updated when the user taps the Back button on the **UINavigationBar**

- In **ItemsViewController.swift**

```
override func viewDidAppear(_ animated: Bool) {  
    super.viewDidAppear(animated)  
  
    tableView.reloadData()  
}
```

Dharmendra Bhatti

27

## Homepwner



- Build and run the application.

Dharmendra Bhatti

28

## Homepwner

- Dismissing the Keyboard
  - Dismissing by pressing the Return key
- class DetailViewController:  
UIViewController, **UITextFieldDelegate** {

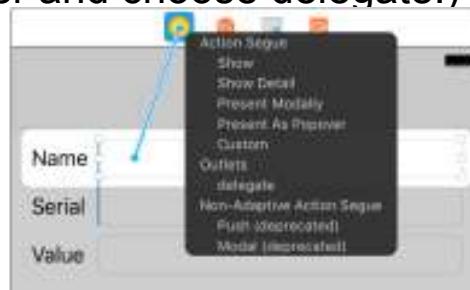
```
func textFieldShouldReturn(_ textField: UITextField) -> Bool {  
    textField.resignFirstResponder()  
    return true  
}
```

Dharmendra Bhatti

29

## Homepwner

- open Main.storyboard and connect the delegate property of each text field to the Detail View Controller (Control-drag from each **UITextField** to the Detail View Controller and choose delegate.)



30

## Homepwner

- Build and run the application.

Dharmendra Bhatti

31

## Homepwner

- Dismissing the Keyboard
  - Dismissing by tapping elsewhere
- Main.storyboard and find Tap Gesture Recognizer in the object library.
- Drag this object onto the background view for the Detail View Controller.

Dharmendra Bhatti

32

## Homepwner

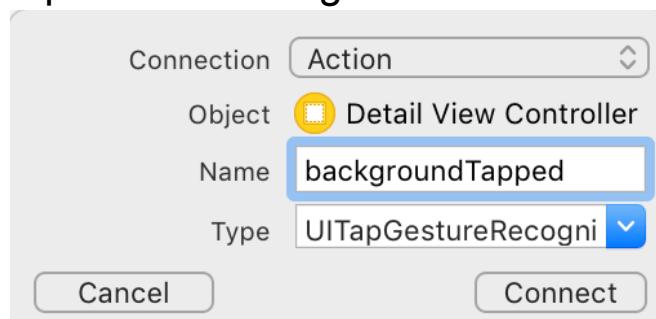
- In the project navigator, Option-click DetailViewController.swift to open it in the assistant editor.
- Control-drag from the tap gesture recognizer in the storyboard to the implementation of **DetailViewController**.

Dharmendra Bhatti

33

## Homepwner

- select Action from the Connection menu.
- Name the action **backgroundTapped**.
- For the Type, choose UITapGestureRecognizer



34

## Homepwner

- DetailViewController.swift
- @IBAction func backgroundTapped(\_ sender: UITapGestureRecognizer) {
  - **view.endEditing(true)**
  - }

Dharmendra Bhatti

35

## Homepwner

- Build and run the application.

Dharmendra Bhatti

36

## Homepwner

- Dismiss the keyboard, when the user taps the Back button

```
override func viewWillDisappear(_ animated: Bool) {  
    super.viewWillDisappear(animated)  
  
    // Clear first responder  
    view.endEditing(true)  
  
    ...  
    ...
```

Dharmendra Bhatti

37

## Homepwner

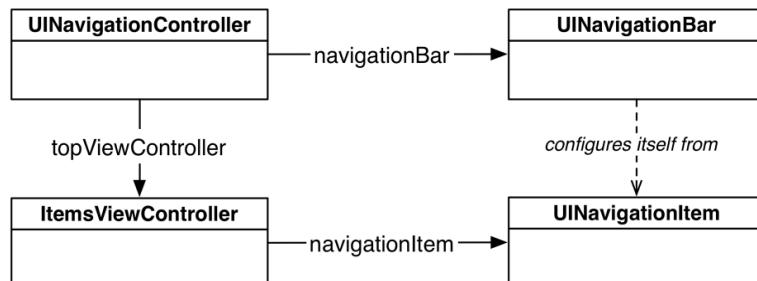
- Build and run the application.

Dharmendra Bhatti

38

## UINavigationItem

- Every **UIViewController** has a `navigationItem` property of type **UINavigationItem**

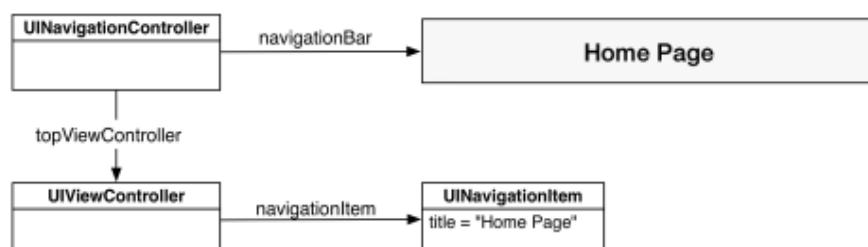


Dharmendra Bhatti

39

## UINavigationItem

- UINavigationItem** with title

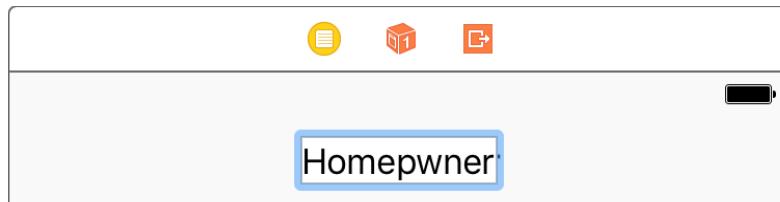


Dharmendra Bhatti

40

## UINavigationItem

- Open Main.storyboard.
- Double-click on the center of the navigation bar above the Items View Controller to edit its title.
- Give it a title of “Homepwner”



41

## UINavigationItem

- In DetailViewController.swift, add a property observer to the item property that updates the title of the navigationItem.

```
var item: Item! {
    didSet {
        navigationItem.title = item.name
    }
}
```

Dharmendra Bhatti

42

## UINavigationItem

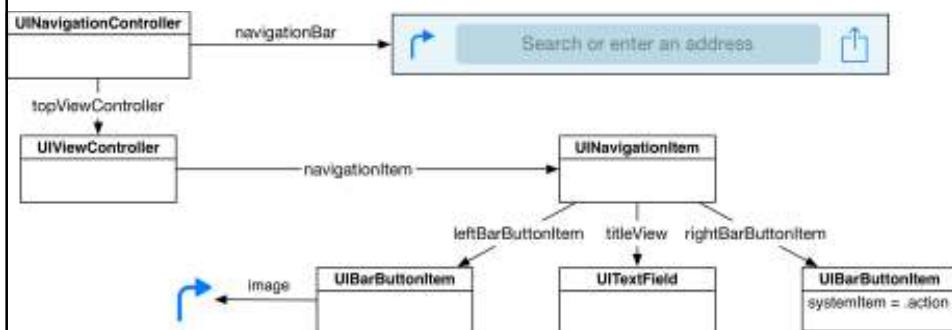
- Build and run the application.

Dharmendra Bhatti

43

## UINavigationItem

- UINavigationItem** with everything



Dharmendra Bhatti

44

## Adding buttons to the navigation bar

- In ItemsViewController.swift, update the method signature for **addNewItem(\_ :)**.

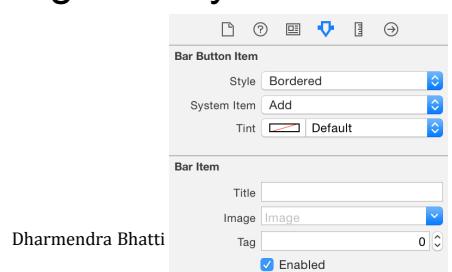
```
@IBAction func addNewItem(_ sender: UIButton) {  
@IBAction func addNewItem(_ sender: UIBarButtonItem) {  
    ...  
}
```

Dharmendra Bhatti

45

## Adding buttons to the navigation bar

- Main.storyboard => object library => Drag a Bar Button Item to the right side of Items View Controller's navigation bar.
- Select this bar button item => attributes inspector => Change the System Item to Add

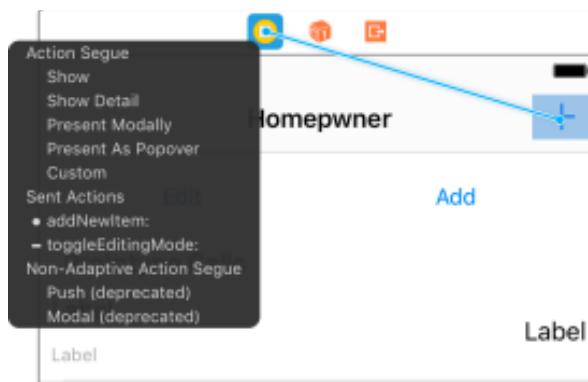


Dharmendra Bhatti

46

## Connecting the addNewItem: action

- Control-drag from this bar button item to the Items View Controller and select addNewItem:



47

## UINavigationItem

- Build and run the application.
- Tap the + button and a new row will appear in the table.

Dharmendra Bhatti

48

## Add “Edit” Bar Button Item Programmatically

- In ItemsViewController.swift, override the **init(coder:)** method to set the left bar button item.

```
required init?(coder aDecoder: NSCoder) {  
    super.init(coder: aDecoder)  
  
    navigationItem.leftBarButtonItem = editButtonItem  
}
```

Dharmendra Bhatti

49

## UINavigationItem

- Build and run the application.

Dharmendra Bhatti

50

## UINavigationItem

- The editButtonItem property creates a **UIBarButtonItem** with the title Edit.
- Even better, this button comes with a target-action pair: It calls the method **setEditing(\_:animated:)** on its **UIViewController** when tapped.

Dharmendra Bhatti

51

## Remove the header view and the associated code

- Select the header view on the table view and press Delete.
- In ItemsViewController.swift, delete the following code

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Get the height of the status bar
    let statusBarHeight = UIApplication.shared.statusBarFrame.height

    let insets = UIEdgeInsets(top: statusBarHeight, left: 0, bottom: 0, right: 0)
    tableView.contentInset = insets
    tableView.scrollIndicatorInsets = insets

    tableView.rowHeight = UITableViewAutomaticDimension
    tableView.estimatedRowHeight = 44
}
```

52

## Remove the header view and the associated code

- Finally, remove the **toggleEditingStyle(\_:)** method.

```
@IBAction func toggleEditingStyle(_ sender: UIButton) {  
    // If you are currently in editing mode...  
    if isEditing {  
        // Change text of button to inform user of state  
        sender.setTitle("Edit", for: .normal)  
  
        // Turn off editing mode  
        setEditing(false, animated: true)  
    } else {  
        // Change text of button to inform user of state  
        sender.setTitle("Done", for: .normal)  
  
        // Enter editing mode  
        setEditing(true, animated: true)
```

Dharmendra Bhatti

53

## Remove the header view and the associated code

- Build and run the application.

Carrier	4:10 PM	
Edit	Homepwner	+
Shiny Mac		\$21
157A5D98		
Shiny Spork		\$0
2850CFFB		
Fluffy Mac		\$96
0D18981B		

54

# CAMERA

Dharmendra Bhatti

55

Homeowner with camera addition



Dharmendra Bhatti

56

## Homepwner-UIImageView

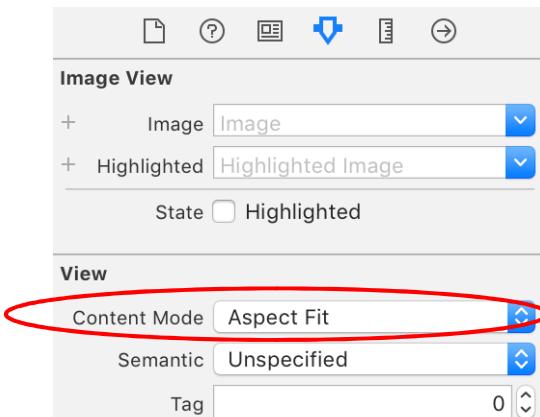
- Homepwner.xcodeproj => Main.storyboard => drag an instance of **UIImageView** onto the view at the bottom of the stack view
- Select image view => size inspector => lower Vertical Content Hugging Priority to be 248 and the Vertical Content Compression Resistance Priority to be 749

Dharmendra Bhatti

57

## Homepwner-UIImageView

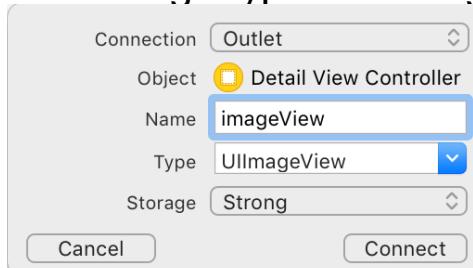
- Select UIImageView => attribute inspector => change “Content Mode” to “Aspect Fit”



58

## Homepwner-UIImageView

- Option-click DetailViewController.swift
- Control-drag from the **UIImageView** to the top of DetailViewController.swift.
- Name the outlet imageView and make sure the storage type is Strong.



59

## Homepwner-UIImageView

- DetailViewController.swift

```
class DetailViewController: UIViewController, UITextFieldDelegate {  
    @IBOutlet var nameField: UITextField!  
    @IBOutlet var serialNumberField: UITextField!  
    @IBOutlet var valueField: UITextField!  
    @IBOutlet var dateLabel: UILabel!  
    @IBOutlet var imageView: UIImageView!
```

Dharmendra Bhatti

60

## Adding a camera button

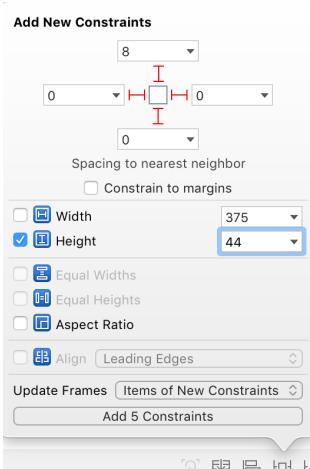
- Main.storyboard => Command-Return
- Select the bottom constraint for the stack view and press Delete to remove it.
- Drag the stack view up a bit

Dharmendra Bhatti



## Adding a camera button

- Drag a toolbar from the object library onto the bottom of the view.
- Select the toolbar => Add New Constraints



62

## Adding a camera button

- Select UIToolBar => UIBarButtonItem => attribute inspector => Change the System Item to Camera



63

## Adding a camera button

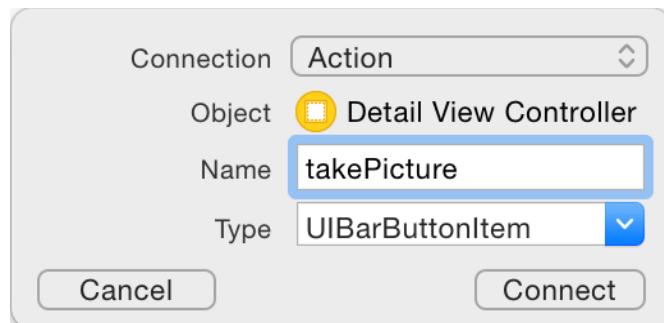
- Build and run the application

Dharmendra Bhatti

64

## Adding a camera button

- Option-click DetailViewController.swift
- Control-drag from the camera button to DetailViewController.swift



65

## Taking Pictures and UIImagePickerController

- Setting the image picker's sourceType
  - `UIImagePickerControllerSourceType.camera`
    - Allows the user to take a new photo.
  - `UIImagePickerControllerSourceType.photoLibrary`
    - Prompts the user to select an album and then a photo from that album.
  - `UIImagePickerControllerSourceType.savedPhotosAlbum`
    - Prompts the user to choose from the most recently taken photos.

Dharmendra Bhatti

66

# Taking Pictures and UIImagePickerController

- UIImagePickerControllerSourceType



# Taking Pictures and UIImagePickerController

- DetailViewController.swift

```
@IBAction func takePicture(_ sender: UIBarButtonItem) {  
    let imagePicker = UIImagePickerController()  
  
    // If the device has a camera, take a picture; otherwise,  
    // just pick from photo library  
    if UIImagePickerController.isSourceTypeAvailable(.camera) {  
        imagePicker.sourceType = .camera  
    } else {  
        imagePicker.sourceType = .photoLibrary  
    }  
}
```

## Setting the image picker's delegate

- class DetailViewController:  
UIViewController, UITextFieldDelegate,  
**UINavigationControllerDelegate,**  
**UIImagePickerControllerDelegate {**

Dharmendra Bhatti

69

## Setting the image picker's delegate

- In DetailViewController.swift

```
@IBAction func takePicture(_ sender: UIBarButtonItem) {  
    let imagePicker = UIImagePickerController()  
  
    // If the device has a camera, take a picture; otherwise,  
    // just pick from photo library  
    if UIImagePickerController.isSourceTypeAvailable(.camera) {  
        imagePicker.sourceType = .camera  
    } else {  
        imagePicker.sourceType = .photoLibrary  
    }  
  
    imagePicker.delegate = self  
}
```

Dharmendra Bhatti

70

## Presenting the image picker modally

- In DetailViewController.swift, add code to the end of **takePicture(\_:)**

```
imagePickerController.delegate = self  
  
// Place image picker on the screen  
present(imagePickerController, animated: true, completion: nil)  
}
```

Dharmendra Bhatti

71

## Permissions

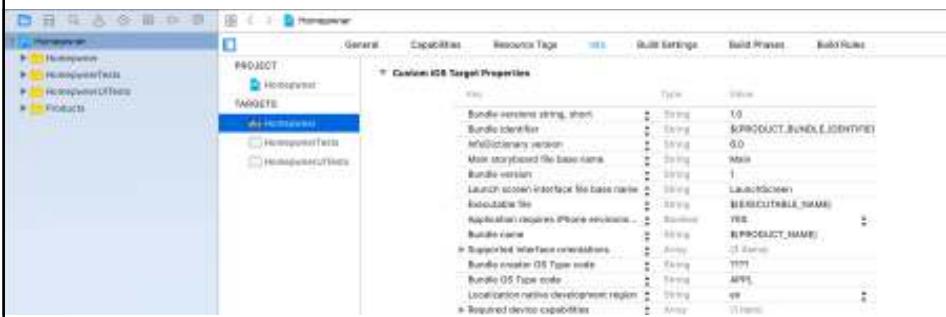
- Application must supply a *usage description* that specifies the reason that your application wants to access particular information
  - Camera and photos
  - Location
  - Microphone
  - HealthKit data
  - Calendar
  - Reminders

Dharmendra Bhatti

72

## Permissions

- In the project navigator, select the project at the top => open the Info tab along the top



Dharmendra Bhatti

73

## Permissions

- Hover over the last entry in this list of Custom iOS Target Properties and click the + button
- Set the Key to be NSCameraUsageDescription
- Set value = “This app uses the camera to associate photos with items.”

Dharmendra Bhatti

74

## Permissions



- Hover over the last entry in this list of Custom iOS Target Properties and click the + button
- Set the Key to be NSPhotoLibraryUsageDescription
- Set value = “This app uses the Photos library to associate photos with items.”

Dharmendra Bhatti

75

## Permissions



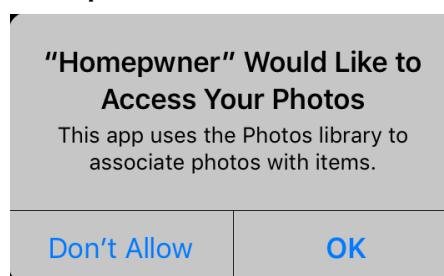
- Adding in the new keys

### Custom iOS Target Properties

Key	Type	Value
Required device capabilities	Array	[Required]
Bundle identifier	String	\$(PRODUCT.BUNDLE.IDENTIFIER)
InfoDictionary version	String	6.0
Main storyboard file base name	String	Main
Bundle version	String	1
Launch screen interface file base name	String	LaunchScreen
Executable file	String	\$(EXECUTABLE_NAME)
Application requires iPhone environment	Boolean	YES
Bundle versions string, short	String	1.0
Supported interface orientations	Array	[Portrait]
Privacy - Photo Library Usage Description	String	This app uses the Photos library to associate photos with items.
Bundle OS Type code	String	APPL
Privacy - Camera Usage Description	String	This app uses the camera to associate photos with items.
Localization native development region	String	en
Supported interface orientations (iPad)	Array	[Portrait]
Bundle name	String	\$(PRODUCT.NAME)

## Permissions

- Build and run the application and navigate to an item.
- Tap the camera button and you will see the permission dialog presented with the usage description



77

## Saving the image

- In DetailViewController.swift,

```
func imagePickerController(_ picker: UIImagePickerController,  
    didFinishPickingMediaWithInfo info: [String: Any]) {  
  
    // Get picked image from info dictionary  
    let image = info[UIImagePickerControllerOriginalImage] as! UIImage  
  
    // Put that image on the screen in the image view  
    imageView.image = image  
  
    // Take image picker off the screen -  
    // you must call this dismiss method  
    dismiss(animated: true, completion: nil)  
}
```

Dharmendra Bhatti

78

## Saving the image

- Build and run the application again.
- Detail view => Select a photo.
- The image picker is dismissed, and you are returned to the **DetailViewController**'s view, where you will see the selected photo.

Dharmendra Bhatti

79

## Creating ImageStore

- Create a new Swift file named ImageStore.

```
import Foundation
import UIKit

class ImageStore {

    let cache = NSCache<NSString, UIImage>()

}
```

Dharmendra Bhatti

80

## Creating ImageStore

- Implement three methods in ImageStore class for adding, retrieving, and deleting an image from the dictionary

```
func setImage(_ image: UIImage, forKey key: String) {  
    cache.setObject(image, forKey: key as NSString)  
}  
  
func image(forKey key: String) -> UIImage? {  
    return cache.object(forKey: key as NSString)  
}  
  
func deleteImage(forKey key: String) {  
    cache.removeObject(forKey: key as NSString)  
}
```

81

## Giving View Controllers Access to the Image Store

- In DetailViewController.swift,
  - `var imageStore: ImageStore!`
- In ItemsViewController.swift
  - `var imageStore: ImageStore!`

## Giving View Controllers Access to the Image Store

- In `ItemsViewController.swift`,

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    // If the triggered segue is the "showItem" segue  
    switch segue.identifier {  
    case "showItem"?:  
        // Figure out which row was just tapped  
        if let row = tableView.indexPathForSelectedRow?.row {  
  
            // Get the item associated with this row and pass it along  
            let item = itemStore.allItems[row]  
            let detailViewController  
                = segue.destination as! DetailViewController  
            detailViewController.item = item  
            detailViewController.imageStore = imageStore  
        }  
    default:  
        preconditionFailure("Unexpected segue identifier.")  
    }  
}
```

83

## Giving View Controllers Access to the Image Store

- In `AppDelegate.swift`,

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions  
    launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {  
    // Override point for customization after application launch.  
  
    // Create an ItemStore  
    let itemStore = ItemStore()  
  
    // Create an ImageStore  
    let imageStore = ImageStore()  
  
    // Access the ItemsViewController and set its item store and image store  
    let navController = window!.rootViewController as! UINavigationController  
    let itemsController = navController.topViewController as! ItemsViewController  
    itemsController.itemStore = itemStore  
    itemsController.imageStore = imageStore
```

Dharmendra Bhatti

84

## Creating and Using Keys

- Add a property to Item.swift to store the key.
- **let itemKey: String**

Dharmendra Bhatti

85

## Creating and Using Keys

- In Item.swift, generate a UUID and set it as the itemKey.

```
init(name: String, serialNumber: String?, valueInDollars: Int) {  
    self.name = name  
    self.valueInDollars = valueInDollars  
    self.serialNumber = serialNumber  
    self.dateCreated = Date()  
    self.itemKey = UUID().uuidString  
  
    super.init()  
}
```

Dharmendra Bhatti

86

# Creating and Using Keys

- In DetailViewController.swift,

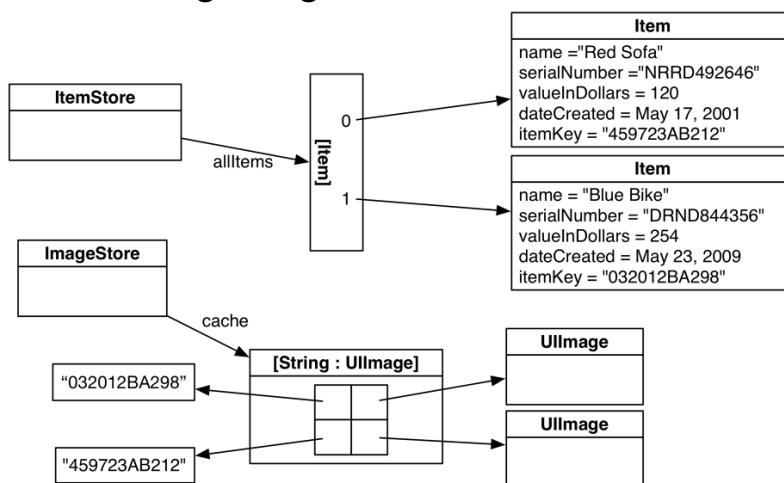
```
func imagePickerController(_ picker: UIImagePickerController,  
    didFinishPickingMediaWithInfo info: [String : Any]) {  
  
    // Get picked image from info dictionary  
    let image = info[UIImagePickerControllerOriginalImage] as! UIImage  
  
    // Store the image in the ImageStore for the item's key  
    imageStore.setImage(image, forKey: item.itemKey)  
  
    // Put that image on the screen in the image view  
    imageView.image = image  
  
    // Take image picker off the screen -  
    // you must call this dismiss method  
    dismiss(animated: true, completion: nil)  
}
```

Dharmendra Bhatti

87

# Creating and Using Keys

- Accessing images from the cache



88

## Creating and Using Keys

- When an item is deleted, delete its image from the image store

```
override func tableView(_ tableView: UITableView,  
                      commit editingStyle: UITableViewCellEditingStyle,  
                      forRowAt indexPath: IndexPath) {  
    // If the table view is asking to commit a delete command...  
    if editingStyle == .delete {  
        let item = itemStore.allItems[indexPath.row]  
  
        let title = "Delete \(item.name)?"  
        let message = "Are you sure you want to delete this item?"  
  
        let ac = UIAlertController(title: title,  
                                 message: message,  
                                 preferredStyle: .actionSheet)  
  
        let cancelAction = UIAlertAction(title: "Cancel",  
                                         style: .cancel,  
                                         handler: nil)  
        ac.addAction(cancelAction)  
  
        let deleteAction = UIAlertAction(title: "Delete", style: .destructive,  
                                         handler: { (action) -> Void in  
        // Remove the item from the store  
        self.itemStore.removeItem(item)  
  
        // Remove the item's image from the image store  
        self.imageStore.deleteImage(forKey: item.itemKey)          89
```

## Wrapping Up ImageStore

- In DetailViewController.swift,

```
override func viewDidAppear(_ animated: Bool) {  
    super.viewDidAppear(animated)  
  
    nameField.text = item.name  
    serialNumberField.text = item.serialNumber  
    valueField.text =  
        numberFormatter.string(from: NSNumber(value: item.valueInDollars))  
    dateLabel.text = dateFormatter.string(from: item.dateCreated)  
  
    // Get the item key  
    let key = item.itemKey  
  
    // If there is an associated image with the item  
    // display it on the image view  
    let imageToDisplay = imageStore.image(forKey: key)  
    imageView.image = imageToDisplay  
}
```

Dharmendra Bhatti

90

## Wrapping Up ImageStore

- Build and run the application.

Dharmendra Bhatti

91

## Questions ???

Dharmendra Bhatti

92