Mastering View Controllers

Vasilica Costescu

vasilica.costescu@gmail.com

What we’ll learn today

- Segues
- Modal presentation
- Navigation controller
- Push/pop transition
- Tab bar controller
- View Controller Life Cycle
- Navigation hierarchy
- Navigation design guidelines

Segues

- Defines a transition from one view controller to another
- Created in Interface Builder by connecting the start and end points
- You can trigger segues programmatically
- Defines the presentation method of the view controller.

Modal presentation

- A new view controller is placed on top of the previous one
- On small screen, it appears full screen
- On large screens, you can customise it to be a popover, form sheet or full screen presentation
- Can use an unwind segue to allow the user to dismiss the new view controller

Demo

Navigation Controllers

- Manage the stack of view controllers
- Provide animations when navigating between related views
- Every navigation controller has a root view controller

Demo

What we learned so far

- Segues
- Modal presentation
- Unwind segue
- Navigation Controller
- Navigation Items
- Passing information between view controllers
- Perform segues programatically

Tab Bar Controllers

- Allow you to arrange your app according to distinct modes or sections
- Have a tab bar view which runs along the bottom of the screen
- Each tab can contain its own independent navigation hierarchy
- Have a property called viewControllers to keep track of all the root view controllers displayed

Download images from here:

https://github.com/costescv/MasteringViewControllers

Demo

The More View Controller

- A special view controller that lists the view controllers that don’t fit on the tab bar
- Can’t be customised or selected
- Doesn’t appear in the viewControllers list of the tab bar controller
- Appears only when needed

What we learned

- Tab Bar Items
- Customising Tab Bar Items (in IB and programmatically)
- The More View Controller

View Controller Life Cycle

View Controller states:

- view not loaded
- view appearing
- view appeared
- view disappearing
- view disappeared

- viewDidLoad()
- viewWillAppear(_ :)
- viewDidAppear(_ :)
- viewWillDisappear(_ :)
- viewDidDisappear(_ :)

viewDidLoad()

- Called after the view controller has finished loading its views

- Here is the place to perform work that depends on the view being loaded and ready

- Type of work suited for this method:

    - update UI (eg: update font of a label)
    - additional initialization of views
    - network requests
    - database access

viewWillAppear(_ :)

- Called right before the view appears on screen.
- Add work that needs to be performed every time the view is displayed to the user.
- Tasks suitable:
    - starting network requests
    - refreshing views
    - updating views
    - adjusting to new screen orientations

viewDidAppear(_ :)

- Called after the view appears on screen
- Use it for:
  - starting an animation
  - fetching data
  - long-running tasks

viewWillDisappear(_ :)

- Called before the view disappears from the screen
- Use it for:
    - saving edits
    - hiding the keyboard
    - canceling network requests

viewDidDisappear(_ :)

- Called after the view disappears from the screen
- Use it for:
    - stop services related to the view (eg stop audio)
    - remove notification observers

Demo

How views are managed

- If the view controller's view is to be added to the view hierarchy, first ensure that the view has been loaded. This triggers viewDidLoad()
- Before adding the view controller's view to the hierarchy, trigger viewWillAppear(_ :)
- Remove views that are no longer displayed, triggering viewWillDisappear(_ :) and viewDidDisappear(_ :)
- Display the new view, triggering viewDidAppear(_ :)

What we learned

- View Controllers state
- viewDidLoad()
- viewWillAppear(_ :)
- viewDidAppear(_ :)
- viewWillDisappear(_ :)
- viewDidDisappear(_ :)

Navigation hierarchy

- Hierarchical navigation
- Flat navigation
- Content-driven navigation

Navigation Design Guidelines

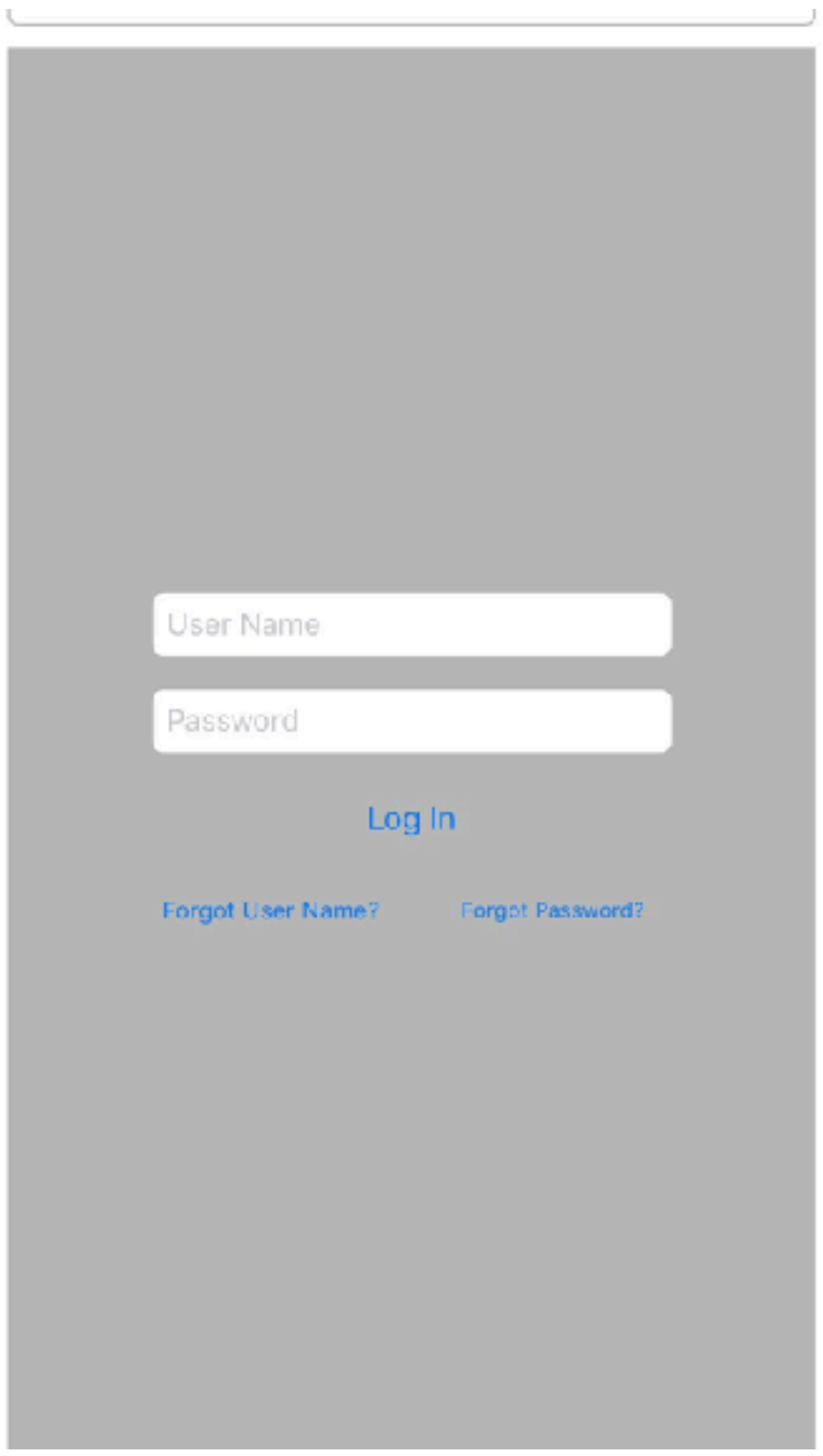
- Design an information structure that makes it fast and easy to get to content
- Use standard navigation components
- Use a navigation bar to traverse a hierarchy of data
- Use a tab bar to present peer categories of content
- Use the proper transition style

Challenge

Create a login screen that will pass a user name between view controllers.

You'll have a text field for username and one for password and a login button. If the user pressed login, a new view controller will be displayed and its title will be the username entered on the previous screen.

You also have two buttons for forgot username and forgot password. When tapped, the title of the view controller that is displayed should be "Forgot Username" or "Forgot Password"


User Name
Password
Log In
Forgot User Name?
Forgot Password?

Resources

- View Controller Programming Guide for iOS
- iOS Human Interface Guidelines
- Github project

Thank you!