

Flashcards

Lesson 7

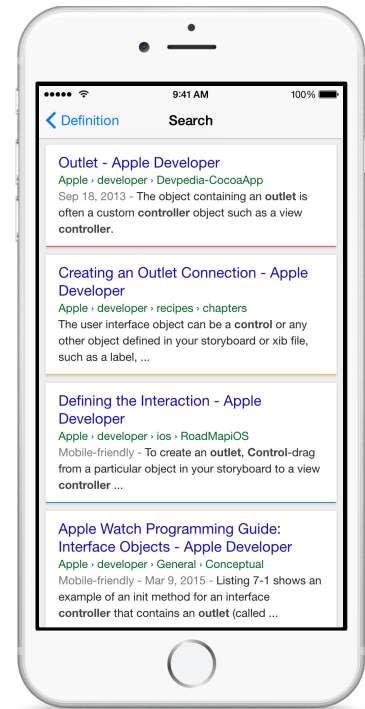


Description

Add a third view controller to the navigation hierarchy, and display web search results for the `Flashcard` term.

Learning Outcomes

- Practice adding a view controller to a storyboard, and binding it to a Swift class.
- Practice defining Swift classes and properties.
- Practice using segues to pass data between view controllers.
- Practice using a web view to display web content within a view.
- Define URL, and explain how URLs and requests are used to obtain web data.
- Practice overriding view controller methods, and using Swift optional binding.



Vocabulary

view controller	segue	web view
property	controller outlet	UIStoryboardSegue
type casting	optional binding	URL
NSURL	http request	NSURLRequest

Materials

- **Flashcards Lesson 7** Xcode project

Opening

How can we add a third view controller to the navigation hierarchy?

Agenda

- Discuss the desire of having an additional view controller that will display web search results for the particular `Flashcard` term.
- Using Interface Builder and the Object Library (⌘L), add a new Bar Button Item to the right side of the definition view controller navigation bar.
- Select the new bar button item, open the Attributes Inspector (⌘4), and set the **Identifier** attribute to **Search**.
- Using Interface Builder and the Object Library (⌘L), add a new View Controller to the storyboard.
- Using the Document Outline (⌘O), rename the new View Controller to **Search Controller**.
- Select the Search bar button item, Control-drag a connection from the button to the Search Controller, select a **show** segue, and observe that Interface Builder adds a navigation bar to the new view controller.
- Using Interface Builder and the Object Library (⌘L), drag a Navigation Item onto the new Search Controller view and set the title to **Search**.
- Using the Object Library (⌘L), add a Web View to the new view controller.
- Adjust the size of the web view so that it is smaller than the main view. Use Control-dragging to add four constraints for the top, bottom, leading and trailing edges of the web view.
- Adjust the size of the web view so that it is flush with the bottom edge of the navigation bar, and the left, right and bottom edges of the view. Use the menu item *Editor > Resolve Auto Layout Issues > Update Constraints* (⇧⌘=) to update the constraints to match what is seen in the canvas.
- Explain how a web view component allows one to embed web content within a view.
- Run the app (⌘R), and tap the Definition, Search and back buttons to navigate up and down the view hierarchy.
- Create a new (⌘N) `SearchController` class that extends `UIViewController`.




```
import UIKit

class SearchController: UIViewController {

}
```

- Add a `Flashcard` property to the `SearchController` class.

```
var flashcard: Flashcard?
```

- Using Interface Builder and the Document Outline () , select the Search Controller, and use the Identity Inspector () to set the **Class** to `SearchController`.
- Using the Assistant Editor () , Control-drag a connection from the web view to the `SearchController` class to create a new outlet.

```
@IBOutlet weak var webView: UIWebView!
```

- In the `DefinitionController` class, add an implementation of `prepareForSegue:sender:`.

```
override func prepareForSegue(segue: UIStoryboardSegue,  
    sender: AnyObject?) {  
    if let searchController =  
        segue.destinationViewController as? SearchController {  
        searchController.flashcard = flashcard  
    }  
}
```

- Discuss how both the `TermController` and `DefinitionController` both use the same approach to assigning the current `Flashcard` object to their respective segue destination view controller.
- Discuss the work necessary for the `SearchController` to display search results: obtaining the `Flashcard` object from the `flashcard` property, using the term to build a URL string, escaping the URL string, creating an `NSURL`, creating an `NSURLRequest`, and finally updating the web view.
- In the `SearchController`, add a verbose implementation of `viewDidLoad`.

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    if let card = flashcard {  
        let urlString =  
            "http://google.com/search?q=apple developer \(preparedTerm)"  
        if let escapedURLString = urlString  
            .stringByAddingPercentEncodingWithAllowedCharacters(.URLQueryA  
llowedCharacterSet()) {  
            if let url = NSURL(string: escapedURLString) {  
                let request = NSURLRequest(URL: url)  
                webView.loadRequest(request)  
            }  
        }  
    }  
}
```

- Explain the optional binding of the `flashcard` property, the construction of the `urlString`, and how optional binding of `escapedURLString` is necessary because `stringByAddingPercentEncodingWithAllowedCharacters:` returns an optional.
- Explain how, because terms may include spaces and punctuation, the method `stringByAddingPercentEncodingWithAllowedCharacters:` is used to properly format the `urlString`.
- Run the app (⌘R), navigate down into the search view, observe the search results displayed, and navigate back up the interface hierarchy.

Closing

What do you think about our implementation of `viewDidLoad`? How would you rate its readability? What about its brevity? Are there things you see that we can improve?

Modifications and Extensions

- Extract the similarities of the three view controllers into one parent `FlashcardController` class, and extend this class with the three individual view controllers.

Resources

View Controller Programming Guide for iOS <http://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/Introduction/Introduction.html>

Cocoa Application Competencies for iOS: Storyboard <http://developer.apple.com/library/ios/documentation/general/conceptual/Devpedia-CocoaApp/Storyboard.html>

Xcode Overview: Build a User Interface https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/edit_user_interface.html

UIKit User Interface Catalog: Web Views <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/UIWebView.html>

UIViewController Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIViewController_Class/index.html

UINavigationController Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UINavigationController_Class/index.html

The Swift Programming Language: Strings and Characters https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/StringsAndCharacters.html

NSURL Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/index.html

NSURLRequest Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSURLRequest_Class/index.html

UIWebView Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIWebView_Class/index.html

NSString Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSString_Class/