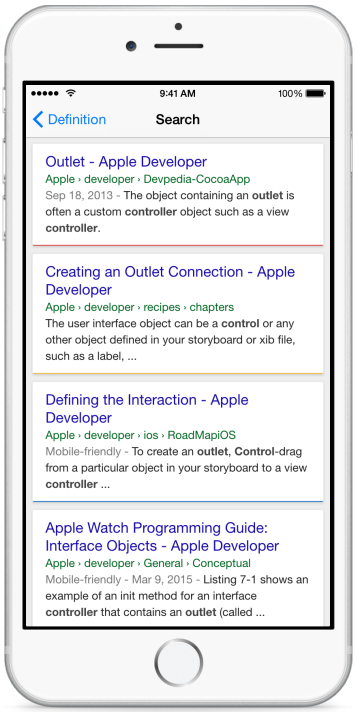
Flashcards
Lesson 8

Description

Refactor the `SearchController` `viewDidLoad` method to remove the nested `if let` statements.



Learning Outcomes

- Critique a code listing to evaluate its readability and brevity.
- Practice implementing properties and methods.
- Discover the Swift multiple optional binding syntax.
- Apply refactoring techniques to improve the quality of existing code.

Vocabulary

helper method	optional	optional binding

Materials

- **Flashcards Lesson 8** Xcode project

Opening

How can we reduce the multiple nesting of the `if let` statements in the `SearchController`?

Agenda

- Discuss `SearchController viewDidLoad` method, and how three optionals necessitate the use of three optional bindings with `if let`.
- Extract the String literal for the URL into a property.

```
let baseSearchURL = "https://google.com/search?q=apple developer"
```

- Add a private `searchURLString` helper method to encapsulate the escaping of the URL string.

```
private func searchURLString(base: String, term: String) -> String? {
   return "\(base) \(term)"
      .stringByAddingPercentEncodingWithAllowedCharacters(.URLQueryAllo
wedCharacterSet())
}
```

- Discuss how the `searchURLString:term:` method encapsulates the escaping of the URL string, which returns a `String?`.
- Explain how Swift allows the expression of multiple optional bindings in one `if let` statement.
- Refactor the `SearchController viewDidLoad` method to remove the nested `if let` statements.

```
override func viewDidLoad() {
   super.viewDidLoad()
   if let card = flashcard,
      escapedURLString =
         searchURLString(baseSearchURL, term: card.term),
      url = NSURL(string: escapedURLString) {
            webView.loadRequest(NSURLRequest(URL: url))
   }
}
```

- Explain how multiple optional bindings can be separated by commas, using one `if let` statement; and how the `flashcard` property, the return value of `searchURLString:term:`, and the `NSURL` instance are all optional types.
- Run the app (⌘R), and observe that the functionality remains unchanged.
- Discuss whether the use of a helper method and multiple optional bindings has increased readability and succinctness.

Closing

How is the user experience affected when we change the segue style to **Present Modally**, and set the **Transition** attribute to **Flip Horizontal**? How might we move back from a modal transition? Instead of using an explicit Definition button to view the back of the flashcard, how might we use gesture recognizers to invoke the segue? What would you need to do to add a tool bar button that enables the user to add new flashcards to the app?

Modifications and Extensions

- Add a toolbar and an **Add** button that presents a modal view with text fields for adding new flashcards to the deck.
- Investigate how to preserve app state, such that the same view is displayed when the app restarts.
- Investigate how to load and save the deck of flashcards using Core Data.

Resources

The Swift Programming Language: Properties https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Properties.html

The Swift Programming Language: Methods https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Methods.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