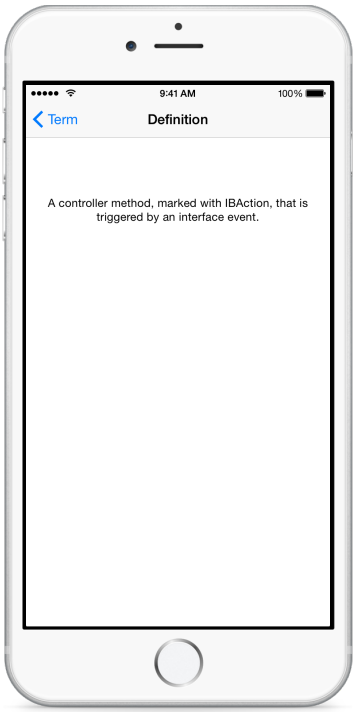
Flashcards
Lesson 5

Description



Create a `DefinitionController`, and add a `Flashcard` property to both view controllers. Pass the `Flashcard` object to the `DefinitionController` via the segue.

Learning Outcomes

- Practice adding a new class to an Xcode project.
- Implement a subclass, making use of inheritance and method overriding.
- Practice using Interface Builder to bind a view controller to a specific class.
- Practice using optional properties and conditional binding.
- Discover and describe how segues can be used to pass data between view controllers.
- Observe a usage of the Swift conditional type cast operator.

Vocabulary

inheritance	extend	subclass
Identity Inspector	optional type	optional binding
method overriding	segue	`UIStoryboardSegue`
type casting		

Materials

- **Flashcards Lesson 5** Xcode project

Opening

How can we display the appropriate flashcard definition in the second view controller?

Agenda

- Add a new (⌘N) Swift class to the project called `DefinitionController`.

```
import UIKit

class DefinitionController: UIViewController {

}
```

- Discuss how `DefinitionController` extends the `UIViewController` base class, and that `DefinitionController` "is a" `UIViewController`.
- Using Interface Builder and the Document Outline (◧), select the Definition Controller,and use the Identity Inspector (⌥⌘3) to set the **Class** to `DefinitionController`.
- Run the app (⌘R), tap the Definition button, and observe how the default text view text still appears.
- Discuss how the `TermController` obtains a `Flashcard` object, and the need to provide the same `Flashcard` object to the `DefinitionController`, so that it can display the `definition` of the particular `Flashcard`.
- Add a `Flashcard?` property to the `DefinitionController` class.

```
var flashcard: Flashcard?
```

- Discuss that the property is optional, because the `DefinitionController` initializer will not initialize the property; and that the property is a variable, because the controller will present definitions of different `Flashcard` objects.
- Using Interface Builder and the Assistant Editor (⌥⌘↩), select the Definition Controller and create a connection from the text view to an outlet in the `DefinitionController` class.

```
@IBOutlet weak var definition: UITextView!
```

- Implement a `viewDidLoad` method in the `DefinitionController`, to set the definition text using the `Flashcard` property.

```
override func viewDidLoad() {
   super.viewDidLoad()
   if let card = flashcard {
      definition.text = card.definition
   }
}
```

- Discuss the optional binding of the `flashcard` property with `if let`.
- Using the Xcode Documentation and API Reference (⇧⌘0), examine the `UIViewController` method `prepareForSegue:sender:`.
- Discuss how, before a segue is performed, the `prepareForSegue:sender:` method is called, and receives a reference to both a `UIStoryboardSegue` object and a reference to the interface control that triggered the segue.
- Using the Xcode Documentation and API Reference (⇧⌘0), examine the `UIStoryboardSegue` class reference, and draw attention to the `sourceViewController` and `destinationViewController` properties.
- Add a `Flashcard?` property to the `TermController` class.

```
var card: Flashcard?
```

- Update the `TermController` `viewDidLoad` implementation, to assign a value to the `Flashcard` property.

```
override func viewDidLoad() {
   super.viewDidLoad()
   if let flashcard = deck.randomCard {
      self.flashcard = flashcard
      termLabel.text = flashcard.term
   }
}
```

- Discuss the differences between the `deck` and `flashcard` properties in the `TermController` class.
- Implement a `prepareForSegue:sender:` method in the `FlashcardController` class.

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

- Explain how an object is retrieved from the `segue`, is casted to a `DefinitionController` using the `as?` type cast operator, and how the `TermController` uses its `flashcard` property to assign a `Flashcard` object to the `DefinitionController` `flashcard` property.
- Run the app (⌘R), tap the Definition button, and observe the correct definition appear.

Closing

Imagine if we had two buttons and two segues that transitioned to two different view controllers. How might you use the `sender` parameter of `prepareForSegue:sender:` to prepare each destination view controller depending on which button is tapped?

Modifications and Extensions

- Create a custom `UIStoryboardSegue` that encapsulates the assignment of the `Flashcard` object to the destination view controller.
- Create a custom `UIStoryboardSegue` class, bind it to the segue connection between the flashcard and definition view controllers, and override its `perform` method with a custom Core Graphics transition animation.

Resources

The Swift Programming Language: Classes and Structures https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/ClassesAndStructures.html

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

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

UIStoryboardSegue Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIStoryboardSegue_Class/index.html

Coordinating Efforts Between View Controllers https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/ManagingDataFlowBetweenViewControllers/ManagingDataFlowBetweenViewControllers.html

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

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