

RSSReader

Lesson 5

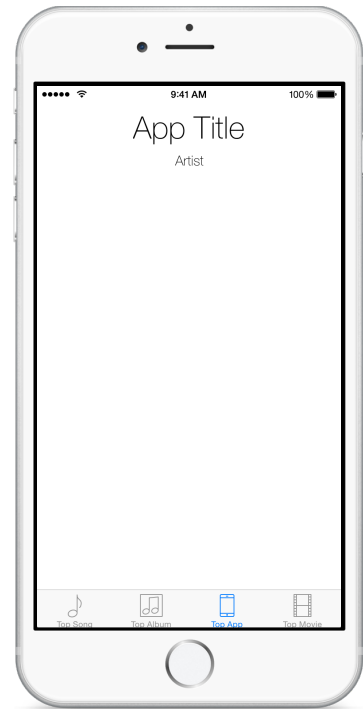


Description

Use Interface Builder to add a user defined runtime attribute for each view controller.

Learning Outcomes

- Practice declaring class properties.
- Distinguish optional type declarations from implicitly unwrapped optionals.
- Relate forced unwrapping to implicitly unwrapped optional types.
- Discover how to create user defined runtime attributes using Interface Builder.
- Describe how storyboards assign user defined runtime attributes to controller properties.
- Discover how `IBInspectable` property attributes expose properties to Interface Builder.
- Relate user defined runtime attributes to `IBInspectable` properties.



Vocabulary

decoupling	property	optional
forced unwrapping	user defined runtime attribute	Identity Inspector
implicitly unwrapped optional	declaration attribute	@IBInspectable

Materials

- **RSSReader Lesson 5** Xcode project



Opening

How can we use controller code to change the title label to Song Title, Album Title, and so on?

Agenda

- Discuss that, despite being able to specify the different label titles in Interface Builder, the `TopMediaController` will eventually need to handle the display of different data for each of the four scenes.
- Discuss how the `TopMediaController` implementation uses a hard-coded string to set the `titleLabel` text property with code.
- Discuss the difference between each of the view controller scenes, specifying the need to decouple the data (the title text) from the `TopMediaController` class implementation.
- Add a new property to the `TopMediaController` class.

```
var titleText: String?
```

- Discuss the property declaration using `var` and an optional type, because the controller class will not assign the property an initial value during initialization.
- Using Interface Builder and the Document Outline () , select the Top Song view controller and use the Identity Inspector () to add a new user defined runtime attribute with the Key Path `titleText`, the Type `String`, and the Value `Song Title`.

User Defined Runtime Attributes		
Key Path	Type	Value
titleText	String	⬇ Song Title
+ —		

- Repeat adding a new user defined runtime attribute for each respective controller, using an appropriate value for each (e.g., Album Title, App Title, Movie Title).
- Update the implementation of the controller `viewDidLoad` method to use the `titleText` property.

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    titleLabel.text = titleText!  
}
```

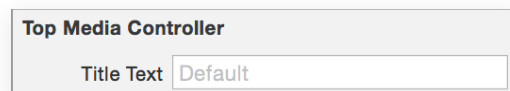
- Explain the forced unwrapping of the optional `titleText` property.
- Run the app (⌘R), interact with each tab, and observe how each title label is different.
- Explain how the user defined runtime attributes within Interface Builder are automatically assigned to controller properties when the controllers are instantiated.
- Discuss how declaring the `titleText` property as an implicitly unwrapped optional can remove the need to force-unwrap the property before use.
- Modify the `titleText` property declaration and the updating of the `titleLabel` in `viewDidLoad`.

```
var titleText: String!  
...  
titleLabel.text = titleText  
}
```

- Run the app (⌘R), interact with each tab, and observe that the functionality remains the same.
- Explain that, in addition to user defined runtime attributes, Interface Builder provides a means to set controller properties through a customizable GUI in the Attributes Inspector.
- Add the `IBInspectable` attribute to the `titleText` property declaration.

```
@IBInspectable var titleText: String!
```

- Using Interface Builder and the Document Outline (⌘⇧D) select each view controller and use the Identity Inspector (⌘⇧3) to delete each of the `titleText` user defined runtime attributes.
- Using Interface Builder and the Document Outline (⌘⇧D) select each view controller, and use the Attributes Inspector (⌘⇧4) to set each **Title Text** attribute to the appropriate value (e.g., Song Title, Album Title, App Title, Movie Title).



- Discuss how the `IBInspectable` attribute informs Interface Builder to present a user interface for setting a value for the property, and how the Attributes Inspector uses the name and data type of the property to display the interface (e.g., `titleText` becomes Title Text).

- Run the app (⌘R), interact with each tab, and observe how each title label is different.

Closing

If we didn't use the user defined runtime attributes or `IBInspectable`, what kind of code strategy would we have to take in order for each controller instance to display different data? Is there a way for a controller to know which tab it belongs to within a tab bar controller, and then make a decision about what data to display?

Modifications and Extensions

- Instead of using an `IBInspectable` attribute with Interface Builder, investigate how to set the `titleText` property of each controller instance at runtime, with code. There are many ways of doing this; consider reaching a solution that requires the least code. Investigate using the app delegate or a custom `UITabBarController` delegate.
- Consider how the values for the text labels are English words. Investigate the concepts of internationalization and localization, and improve the text values for all of the labels in the interface by using localized `String` values.

Resources

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

Interface Builder Help: Adding User Defined Runtime Attributes https://developer.apple.com/library/mac/recipes/xcode_help-interface_builder/Chapters/AddUserDefinedRuntimeAttributes.html

Creating a Custom View that Renders in Interface Builder https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/CreatingaLiveViewofaCustomObject.html

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