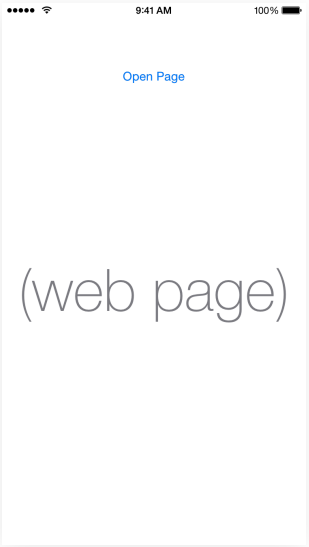
EasyBrowser
Lesson 2

Description

Add a web view component to the interface and enable the button to open a page in the web view.



Learning Outcomes

- Practice different means of establishing and inspecting action connections between the view and controller.
- Practice connecting outlet connections between the view and controller.
- Discover how `UIWebView` components open web pages using `NSURLRequest` objects.

Vocabulary

web view	`UIWebView`	connection well
Connections Inspector	URL	`NSURL`
request	`NSURLRequest`	

Materials

- **EasyBrowser Lesson 2** Xcode project

Opening

How can we view Web content within our app itself?

Agenda

- Using Interface Builder, change the button label to **Open Page**.
- Using the Object Library (⌥⌘L), drag a Web View object onto the center of the interface.
- Set constraints to horizontally center the web view; to pin its bottom space to the bottom layout guide; to pin its top space to the button; and to pin its left and right edges to the container margin.
- Resize the web view to occupy most of the interface, and update the constraints using the menu item *Editor* > *Resolve Auto Layout Issues* > *Update Constraints* (⇧⌘=).
- Using the Assistant Editor (⌥⌘↩), rename the controller method `openPageInSafari:` to `openPage:`.
- Discuss why the connection well next to the method turns hollow.
- Using Interface Builder, Control-click the button, and delete the existing connection from the button's *Touch Up Inside* event to the nonexistent `openPageInSafari:` action.
- With the Assistant Editor (⌥⌘↩) open, drag a connection from the hollow connection well next to the `openPage:` method to the button.
- With the button selected, open the Connections Inspector (⌥⌘6), and verify the new connection.
- Using Interface Builder and the Assistant Editor (⌥⌘↩), create an outlet property for the web view in the `ViewController` class.

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

- Discuss the different ways of establishing connections between views and controllers.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `UIWebView` class reference and the `loadRequest:` method.
- Update the implementation of `openPage:`.

```
@IBAction func openPage(sender: UIButton) {
    if let url = NSURL(string: "http://developer.apple.com") {
        let request = NSURLRequest(URL: url)
        webView.loadRequest(request)
    }
}
```

- Explain how an `NSURLRequest` represents the network request that a `UIWebView` makes to a web server to retrieve a web page.
- Run the app (⌘R), and tap the button to demonstrate the loading of the page.
- Discuss how the web view content does not fit entirely within the size of the web view.

- Using Interface Builder, select the web view component, open the Attributes Inspector (⌥⌘4), and check the **Scales Page to Fit** attribute.
- Run the app (⌘R), tap the button, and observe how the web content fits within the web view.
- Demonstrate ⌥-clicking the iOS Simulator to simulate two touches to zoom, pan or scroll the web content within the web view.

Closing

What does the iOS Simulator's *Hardware* > *Device* menu enable us to do?

Modifications And Extensions

- Extract the string literal for the web page URL into a configuration item within an app preferences **.plist** file, and use the `NSUserDefaults` API to obtain the web page URL within `openPage:`.

Resources

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

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

Creating an Action Connection https://developer.apple.com/library/ios/recipes/xcode_help-IB_connections/chapters/CreatingAction.html

Creating an Outlet Connection https://developer.apple.com/library/ios/recipes/xcode_help-IB_connections/chapters/CreatingOutlet.html

UIWebView Class Reference http://developer.apple.com/library/ios/documentation/uikit/reference/UIWebView_Class/Reference/Reference.html

NSURL Class Reference http://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/Reference/Reference.html

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

Apple URL Scheme Reference http://developer.apple.com/library/ios/featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html

The Swift Programming Language: Optional Binding https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID333