RSSReader
Lesson 8

Description

Add an additional asynchronous method invocation to retrieve and display the media image.

Learning Outcomes

- Repeat using an asynchronous method invocation with `NSURLRequest`.
- Practice using `NSURLRequest`, `NSURLConnection`, `NSJSONSerialization`, and `NSDictionary` classes to inspect RSS feed data.
- Describe the behavior of asynchronous method calls, and relate asynchronous method calls to retrieving data over a network.
- Describe how closures may be passed to methods for invocation.

Vocabulary

RSS feed	HTTP	URL
`NSURL`	request	`NSURLRequest`
`NSURLConnection`	asynchronous method	closure
`NSDictionary`	`UIImageView`	`UIImage`

Materials

- **RSSReader Lesson 8** Xcode project
- Internet connectivity to the **ax.itunes.apple.com** domain

- RSS sample data example text file (**sampledata.rss.txt**)
- **Asynchronous Methods** presentation
- **Closures** presentation

Opening

How can we display a thumbnail image of the media type in our views?

Agenda

- Examine the RSS feed data, drawing attention to the `im:image` key that contains an array of URLs for image files.
- Open an image URL in a web browser, and observe the image that appears.
- Discuss how an image URL might be used to create another `NSURLConnection` that retrieves the actual image data.
- Update the controller `viewDidLoad` method to retrieve the image URL from the RSS data, create an `NSURL` object, and pass the `NSURL` to a controller method that will asynchronously retrieve the image.

```
...
artist = feed.valueForKeyPath("feed.entry.im:artist.label") as? String,
imageURLs = feed.valueForKeyPath("feed.entry.im:image") as? [NSDictionary] {
   if let imageURL = imageURLs.last,
      imageURLString = imageURL.valueForKeyPath("label") as? String {
         self.loadImageFromURL(NSURL(string:imageURLString)!)
   }
   self.titleLabel.text = title
...
```

- Implement the `loadImageFromURL:` method.

```
func loadImageFromURL(URL: NSURL) {
   let request = NSURLRequest(URL: URL)
   NSURLConnection.sendAsynchronousRequest(request,
      queue: NSOperationQueue.mainQueue()) { response, data, error in
      self.imageView.image = UIImage(data: data)
   }
}
```

- Discuss the similarities between `loadImageFromURL:` and the RSS data retrieval in `viewDidLoad`.
- Explain how a request is sent to a server with the `NSURLConnection sendAsynchronousRequest:queue:completionHandler:` method.
- Present the concept of asynchronous methods.

- Discuss how the `queue:` parameter specifies the run loop that the closure should execute within, and the best practice of using the `mainQueue` to execute closures that update the interface.
- Discuss how the view appears while the request for RSS data is sent asynchronously, and how the `completionHandler:` argument specifies a closure that is invoked once the data is obtained from the server.
- Present the concept of closures.
- Discuss how, once the image data is retrieved, the `completionHandler:` closure instantiates a `UIImage` object, and assigns it to the controller `imageView.image` property.
- Run the app (⌘**R**), interact with each tab, and observe the title, artist and image appear in each view.

Closing

Why does each view take a moment to appear the first time the tab is selected, but not for subsequent appearances?
How might you extract a link from the RSS data and enable the user to view the song, album, app or movie on the App Store?
What models are missing from our app, and what should they encapsulate?
If the data takes a long time to retrieve, how might you add a progress indicator to display in each view while the data is being retrieved?
What happens if the top media in the RSS feed changes and the app has not been restarted? How might you implement a feature that ensures that the data is the most current?
What other RSS feeds are there, and how might you incorporate their data into an app?
How should the app behave when the data is not available, or if no network connection is present?

Modifications and Extensions

- Use appropriate error handling in `loadImageFromURL:` to ensure the app is as robust as possible.
- Investigate how a user might be able to customize the tabs of the app.
- Enable the user to tap on each image, which displays the media in the iTunes store or App Store.
- Add a `MediaItem` model to the app that encapsulates the data retrieval.
- Enable the user to refresh the data in the app.
- Investigate how to preserve app view state when the app is quit and is restarted, such that the same tab is visible when the app starts.

Resources

Apple RSS Feeds http://www.apple.com/rss/

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

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

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

NSOperationQueue Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/NSOperationQueue_class/index.html

NSJSONSerialization Class Reference https://developer.apple.com/library/ios/documentation/Foundation/Reference/NSJSONSerialization_Class/

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

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

UIImageView Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIImageView_Class/index.html

UIImage Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIImage_Class/