2.0 Lesson - Weather App Data

Paul Solt - Paul@SuperEasyApps.com
SuperEasyApps.com

2.1 Lecture - Weather Web Services and App Data

In this lesson you will learn how to request weather information for a particular location and update your app's UI. You will learn the building blocks for making web requests to servers and how to handle situations when errors occur.

2.2 Tutorial - Connect UI Outlets

In order to programmatically change the weather label, icon image, and weather summary you need to create outlets for the temperature, weather image, summary, and date labels. Drag and connect the UI from storyboard to code with the Assistant Editor.

```
@IBOutlet weak var temperatureLabel: UILabel!
@IBOutlet weak var weatherImageView: UIImageView!
@IBOutlet weak var summaryLabel: UILabel!
@IBOutlet weak var dateLabel: UILabel!
```

2.3 Tutorial - DarkSky Weather API

1. Create an account on DarkSky to request weather information
    1. https://developer.forecast.io
2. Create instance variables at the top of the ViewController.swift class to store the API, web server address (URL), your latitude, and your longitude.

```
// TODO: Set the API Key
var apiKey = "YOUR_API_KEY" //  "dcfbc6...8385"
let weatherAPI = "https://api.forecast.io/forecast"

// TODO: Set your lat/long ... this is Rochester, NY
// Get latitude and longitude here: http://www.latlong.net
var latitude: Double = 43.161030
var longitude: Double = -77.610922
```

3. Copy/paste your API key for the placeholder
4. Copy/paste your latitude and longitude from:
    1. http://latlong.net

2.4 Tutorial - Create a Server URL Request Address

Now you will create a method that will make a web request to the server. The weather API you are using requires that the web address be constructed in a specific way so that you can get your weather information.

1. Create the requestWeather() method

```
func requestWeather() {

}
```

2. Construct the URL for the web request

```
func requestWeather() {
    // construct URL
    let weatherURL = NSURL(string: "\(weatherAPI)/\(apiKey)/\(latitude),\(longitude)")
    print("URL:", weatherURL)
}
```

3. Optional: If you don't want the default units to be US units (ºF, miles per hour, etc.), you need to change your URL to include SI as an optional parameter: `?units=SI`.

    Read the documentation for a full list of customizations for your API request format.
    https://developer.forecast.io/docs/v2

```
// SI Units
func requestWeather() {
```

```
    // construct URL
    let weatherURL = NSURL(string: "\(weatherAPI)/\(apiKey)/\(latitude),\(longitude)?
units=si")
    print("URL:", weatherURL)
}
```

4. Test your constructed URL in your web browser by visiting the URL printed to your console.
    1. It should look like:
    `https://api.forecast.io/forecast/YOUR_API_KEY/43.16103,-77.610922`
    2. If it is formed correctly, you should see the JSON response data.
    3. Otherwise you will see a `400 Bad Request`, which means your URL or API key are wrong.

2.5 Tutorial - Web APIs and NSURLSession

After you have a working URL with your API key you are ready for the next step to make the weather request from your iPhone app. You will use the NSURLSession to make the HTTP web request.

Update your requestWeather() method with the following code to make an API request using NSURLSession.

```
func requestWeather() {
    // construct URL
    let weatherURL = NSURL(string: "\(weatherAPI)/\(apiKey)/\(latitude),\(longitude)")
    print("URL:", weatherURL)
    if let weatherURL = weatherURL {
        let weatherTask = NSURLSession.sharedSession().dataTaskWithURL(weatherURL) { (data:
NSData?, response: NSURLResponse?, error: NSError?) -> Void in
            // Handle response
        }
        weatherTask.resume()
    }
}
```

Links

- NSURLSession Reference - apple.com
- What's New in Foundation Networking WWDC 2014 Video - apple.com
- Networking with NSURLSession WWDC 2015 Video - apple.com

2.6 Tutorial - Handle the NSURLResponse JSON Data

1. In the completion block for the dataTaskWithURL() method call add the following to handle the response:

```
let weatherTask = NSURLSession.sharedSession().dataTaskWithURL(weatherURL,
completionHandler: { (data: NSData?, response: NSURLResponse?, error: NSError?) -> Void in

    // handle response
    print("received a response")

    // 1. test the error
    if error == nil {
        // 2. Test web response (200 code)
        if let response = response as? NSHTTPURLResponse {
            // 200 is ok
            if response.statusCode == 200 {

                // 3. Unwrap the data
                if let data = data {

                    // 4. Convert data to JSON (swift error handling)
                    do {
                        let json = try NSJSONSerialization.JSONObjectWithData(data,
options: [])
                        print("json:", json)

                        // 5. Process the data

                    } catch let jsonError as NSError {
                        print("Error converting JSON data", jsonError.localizedDescription)
                    }
                }
            } else {
                print("Error HTTP Response code:", response.statusCode)
            }
        }

    } else { // error
        print("Error: NSURLSession:", error!.localizedDescription)
    }

})
```

Links

- NSJSONSerialization - apple.com

- NSHTTPURLResponse - apple.com

2.7 Tutorial - Parse the JSON Data as Swift Dictionary Types

1. Create a new method called parseWeather() that will take a Dictionary parameter.

```
func parseWeather(weatherJSON: Dictionary<String, AnyObject>) {
}
```

2. Below the `let json = try ...` line, convert json `AnyObject` data into a Swift dictionary of type [String: AnyObject] and send it to the `parseWeather()` method.

```
// 4. Convert data to JSON (swift error handling)
do {
    let json = try NSJSONSerialization.JSONObjectWithData(data, options: [])
    print("json:", json)

    // 5. Process the data
    if let weatherJSON = json as? Dictionary<String, AnyObject> {
        self.parseWeather(weatherJSON)
    }

} catch let jsonError as NSError {
    print("Error converting JSON data", jsonError.localizedDescription)
}
```

3. Using the Dictionary you need to access the `currently` weather information and then the `temperature`, `summary`, and `icon` data.

```
func parseWeather(weatherJSON: Dictionary<String, AnyObject>) {

    if let currentWeather = weatherJSON["currently"] as? Dictionary<String, AnyObject> {
        print("currently:", currentWeather)
        if let temperature = currentWeather["temperature"] as? Float {
            print("temperature:", temperature)
        }
        if let summary = currentWeather["summary"] as? String {
            print("summary:", summary)
        }
        if let icon = currentWeather["icon"] as? String {
            print("icon:", icon)
```

```
        }
        // Update the User Interface on the Main Thread
    }
}
```

2.8 Tutorial - Swift Data Structure and Updating the Weather UI

Create a Swift structure to store the data from JSON. This structure will be how you can share information that was downloaded to your user interface.

1. Create a new Swift code file called `WeatherDataPoint`

```
import Foundation
struct WeatherDataPoint {
    var temperature: Float?
    var summary: String?
    var icon: String?
}
```

2. Create a WeatherDataPoint variable and set the appropriate values from the parsing code.

```
func parseWeather(weatherJSON: Dictionary<String, AnyObject>) {
    if let currentWeather = weatherJSON["currently"] as? Dictionary<String, AnyObject> {
        print("currently:", currentWeather)
        var weatherData = WeatherDataPoint()

        if let temperature = currentWeather["temperature"] as? Float {
            print("temperature:", temperature)
            weatherData.temperature = temperature
        }
        if let summary = currentWeather["summary"] as? String {
            print("summary:", summary)
            weatherData.summary = summary
        }
        if let icon = currentWeather["icon"] as? String {
            print("icon:", icon)
            weatherData.icon = icon
        }
        // Update the User Interface on the Main Thread
    }
}
```

3. Add a method to update the user interface in ViewController.swift

```
func updateWeather(weatherData: WeatherDataPoint) {
    print("update weather UI")
}
```

4. Most data processing happens in the background so that your UI remains responsive.
5. At the bottom of parseWeather(_:) you need to add a call to update the user interface with the data you parsed.

```
// Update the User Interface on the Main Thread
// Pass the data object
dispatch_async(dispatch_get_main_queue(), { () ->       Void in
    self.updateWeather(weatherData)
})
```

2.9 Tutorial - Update the UI With the Temperature and Weather Summary

App UI needs to be updated on the main thread to prevent bugs and glitches. Add the logic to set your temperature and weather summary labels.

```
func updateWeather(weatherData: WeatherDataPoint) {
    print("update weather UI")
    if let temperature = weatherData.temperature {
        let numberFormatter = NSNumberFormatter()
        numberFormatter.maximumFractionDigits = 0
        temperatureLabel.text = numberFormatter.stringFromNumber(temperature)
    }
    if let weatherSummary = weatherData.summary {
        summaryLabel.text = weatherSummary
    }
    // update weather icon
}
```

2.10 Tutorial - Update the Weather Status Icon

1. The API documentation provides information on the icon's available values, which match your

weather icon images. https://developer.forecast.io/docs/v2

> *icon: A machine-readable text summary of this data point, suitable for selecting an icon for display. If defined, this property will have one of the following values: clear-day, clear-night, rain, snow, sleet, wind, fog, cloudy, partly-cloudy-day, or partly-cloudy-night. (Developers should ensure that a sensible default is defined, as additional values, such as hail, thunderstorm, or tornado, may be defined in the future.)*

1. You will use switch statement (like a compound if/else if statement) on the icon provided by the weather data to chose the appropriate image.

```
// update weather icon
if let icon = weatherData.icon {
    switch(icon) {
    case "clear-day":
        weatherImageView.image = UIImage(named: "Clear-Day")
    case "clear-night":
        weatherImageView.image = UIImage(named: "Clear-Night")
    case "rain":
        weatherImageView.image = UIImage(named: "Rain")
    case "snow":
        weatherImageView.image = UIImage(named: "Snow")
    case "sleet":
        weatherImageView.image = UIImage(named: "Sleet")
    case "wind":
        weatherImageView.image = UIImage(named: "Wind")
    case "fog":
        weatherImageView.image = UIImage(named: "Fog")
    case "cloudy":
        weatherImageView.image = UIImage(named: "Cloudy")
    case "partly-cloudy-day":
        weatherImageView.image = UIImage(named: "Partly-Cloudy-Day")
    case "partly-cloudy-night":
        weatherImageView.image = UIImage(named: "Partly-Cloudy-Night")
    case "hail":
        weatherImageView.image = UIImage(named: "Sleet")
    case "thunderstorm":
        weatherImageView.image = UIImage(named: "Lightning")
    case "tornado":
        weatherImageView.image = UIImage(named: "Cloudy")
    default:
        print("Error: unexpected icon name: \(icon)")
        weatherImageView.image = nil
    }
}
```

Links

- Control Flow - Switch - apple.com

2.11 Tutorial - Update the Date With NSDateFormatter

1. Add the code for updateDate() using the NSDateFormatter class

```
func updateDate() {
    let date = NSDate()
    let dateFormatter = NSDateFormatter()
    dateFormatter.dateFormat = "MMMM d"
    dateLabel.text = dateFormatter.stringFromDate(date)
}
```

2. At the bottom of your `viewDidLoad()` method make a call to `updateDate()`

```
override func viewDidLoad() {
    super.viewDidLoad()
    // make a request for weather
    requestWeather()
    updateDate()
}
```

3. Make a method called `clearWeather()` to clear the temporary weather information before loading the weather from the server.

```
func clearWeather() {
    weatherImageView.image = nil
    temperatureLabel.text = "-"
    summaryLabel.text = ""
}
```

4. Call your `clearWeather()` method at the end of `viewDidLoad()`

```
override func viewDidLoad() {
    super.viewDidLoad()
    if apiKey == "YOUR_API_KEY" {
        print("ERROR: Request an API key from https://developer.forecast.io")
    }
    // make a request for weather
```

```
    requestWeather()
    updateDate()
    // clear weather
    clearWeather()
}
```

5. Now install your app on your iPhone!

Links

- NSDateFormatter Reference - apple.com
- Date formatting symbols