# Basic Exercises Part 6.2 Parsing Data … continuation.

## … continuation

* **1.1 Load a sample response data**

In previous project we deserialize the Data instance using the JSONSerialization API. The API is easy to understand. We invoke jsonObject(with:options:), passing in a Data instance and a collection of options. Notice that we ignore error handling by using the try! keyword. Once we have a solution, we can focus on error handling in the weather application. If no error is thrown, jsonObject(with:options:) returns an object of type Any.

### **1.2 Create model objects**

Extract two types of data from the JSON response.

* The location the weather data is from.
* Hourly weather data for that location.

Take a moment to inspect the structure of the JSON response. The latitude and longitude of the location the weather data is from are located at the root of the JSON object. That is easy. The hourly weather data is nested within the JSON object. That is going to be a bit trickier.

The hourly weather data is an array of data points. This means that **we need to create an array of model objects to store** the hourly weather data for the location. To store the weather data, we are going to create two structures, WeatherData and WeatherHourData.

Create a new file (File →New →File… →Swift → name it WeatherData →Next →Create. And the WeatherHourData). Also create the folder structure to contain these new files, name it Models.

**import** Foundation

**public** **struct** WeatherData {

**public** **let** lat: Double

**public** **let** long: Double

**public** **let** hourData: [WeatherHourData]

**public** **init**(lat: Double, long: Double, hourData: [WeatherHourData]) {

**self**.lat = lat

**self**.long = long

**self**.hourData = hourData

}

}

Note that hourData is of type [WeatherHourData]. This brings us to the implementation of the WeatherHourData struct.

**public** **struct** WeatherHourData {

**public** **let** time: Date

**public** **let** windSpeed: Int

**public** **let** temperature: Double

**public** **let** precipitation: Double

**public** **init**(time: Date, windSpeed: Int, temperature: Double, precipitation: Double) {

**self**.time = time

**self**.windSpeed = windSpeed

**self**.temperature = temperature

**self**.precipitation = precipitation

}

}

Great. We have defined the structures we plan to pour the JSON data into. The next challenge is decoding the JSON data and creating instances of the structures using the weather data of the Dark Sky API.

### **1.3 Convert the data into a dictionary**

Create an instance of WeatherData using the JSON response

**if** **let** JSON = JSON **as**? [String: AnyObject] {

**if** **let** lat = JSON["latitude"] **as**? Double, **let** long = JSON["longitude"] **as**? Double {

**let** weatherData = WeatherData(lat: lat, long: long, hourData: [])

}

}

No parse the hourly weather data. This is starting to look like a bad idea.

**if** **let** JSON = JSON **as**? [String: AnyObject] {

**if** **let** lat = JSON["latitude"] **as**? Double,

**let** long = JSON["longitude"] **as**? Double,

**let** hourlyData = JSON["hourly"]?["data"] **as**? [[String: AnyObject]] {

// Create Buffer

**var** hourData = [WeatherHourData]()

**for** hourlyDataPoint **in** hourlyData {

**if** **let** time = hourlyDataPoint["time"] **as**? Double,

**let** windSpeed = hourlyDataPoint["windSpeed"] **as**? Int,

**let** temperature = hourlyDataPoint["temperature"] **as**? Double,

**let** precipitation = hourlyDataPoint["precipIntensity"] **as**? Double {

// Convert Time to Date

**let** timeAsDate = Date(timeIntervalSince1970: time)

// Create Weather Hour Data

**let** weatherHourData = WeatherHourData(time: timeAsDate, windSpeed: windSpeed, temperature: temperature, precipitation: precipitation)

// Append to Buffer

hourData.append(weatherHourData)

}

}

**let** weatherData = WeatherData(lat: lat, long: long, hourData: hourData)

}

### }

Ok! We achieved the goal we set out to achieve, but the approach we took needs to change. The current implementation is too brittle and overly complex. Try something different.

### **1.4 Protocols and Extensions**

Create a new file in the sources group (folder), name it JSONDecodable.swift (remember why the termination “codable”?).

**protocol** JSONDecodable {

**init**?(JSON: **Any**)

}

We declare a protocol: JSONDecodable, with one method, a failable initializer that accepts a parameter of type Any. If we make the WeatherData and WeatherHourData structures conform to his protocol, we can move most of the logic we wrote earlier to their corresponding structures.

Open WeatherData.swift and add an extension for the JSONDecodable protocol. In the initializer, add the logic for creating an instance of the structure with an object type Any.

**extension** WeatherData: JSONDecodable {

**public** **init**?(JSON: **Any**) {

**guard** **let** JSON = JSON **as**? [String: AnyObject] **else** {**return** **nil**}

**guard** **let** lat = JSON["latitude"] **as**? Double **else** {**return** **nil** }

**guard** **let** long = JSON["longitude"] **as**? Double **else** {**return** **nil** }

**guard** **let** hourlyData = JSON["hourly"]?["data"] **as**? [[String: AnyObject]] **else** { **return** **nil** }

**self**.lat = lat

**self**.long = long

**var** buffer = [WeatherHourData]()

**for** hourlyDataPoint **in** hourlyData {

**if** **let** weatherHourData = WeatherHourData(JSON: hourlyDataPoint) {

buffer.append(weatherHourData)

}

}

**self**.hourData = buffer

}

}

We also need to conform the WeatherHourData structure to the JSONDecodable protocol. The implementation looks very similar.

**extension** WeatherHourData: JSONDecodable {

**public** **init**?(JSON: **Any**) {

**guard** **let** JSON = JSON **as**? [String: AnyObject] **else** { **return** **nil** }

**guard** **let** time = JSON["time"] **as**? Double **else** { **return** **nil** }

**guard** **let** windSpeed = JSON["windSpeed"] **as**? Int **else** { **return** **nil** }

**guard** **let** temperature = JSON["temperature"] **as**? Double **else** { **return** **nil** }

**guard** **let** precipitation = JSON["precipIntensity"] **as**? Double **else** { **return** **nil** }

**self**.windSpeed = windSpeed

**self**.temperature = temperature

**self**.precipitation = precipitation

**self**.time = Date(timeIntervalSince1970: time)

}

}

To create an instance of the WeatherData structure, we invoke the init(JSON:) failable initializer. This returns an optional. In other words, if the JSON data isn't structured correctly, the initialization fails. We could also make the initializer throwing, which may be a better option.

**if** **let** weatherData = WeatherData(JSON: JSON) {

print(weatherData)

}

**1.5 Check the attached files**

Check the code in the files attached. Make it works in your Xcode project.