# Saving Data in iOS

Hands-On Challenges

# Saving Data in iOS Hands-On Challenges

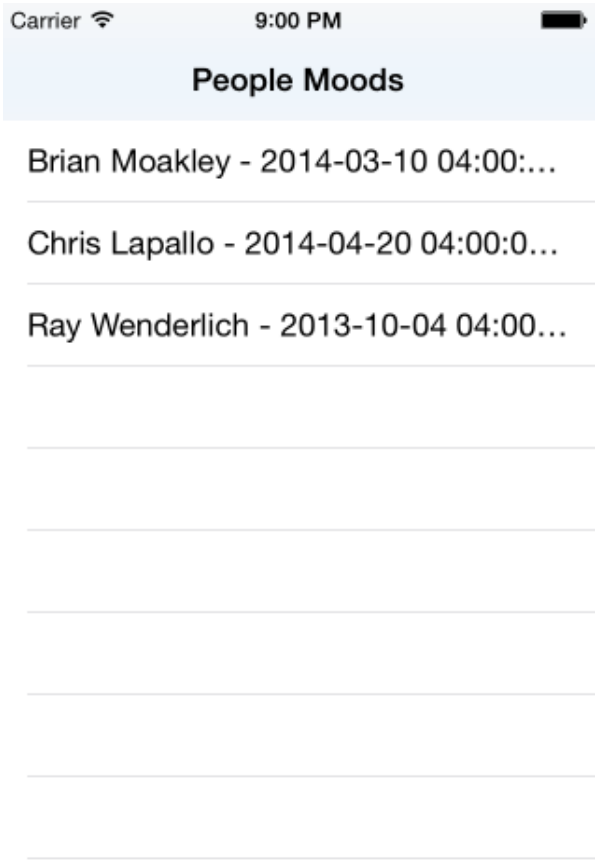


# Saving with XML

Throughout your iOS development career, chances are, you'll be reading a lot of XML (and just maybe, writing a bit of it as well). Learning how to work with XML is pretty crucial to being successful on the platform as you may want to hook into third party APIs or read data provide by other developers.

## Challenge A: Getting Moody with NSXMLParser

Open the first sample project and you'll notice an empty tableview. This table view is supposed to be populated with names, and when one of the names are tapped, the app will display the "mood" of each person and the reason for the actual mood.




Open the sample app and open **sample_data.xml**. This xml file contains all the data for each record. This data is the backing data for each RWTMood object. The

RWTMood object has properties that correspond to each field on the RWTMood object.

Your challenge is to create a new parser object that will parse the sample_data.xml file, create an array of RWTMood objects, then inside of **MoodListingViewController.m**, assign the mood objects to the moodList property.

Build and run, and you will see all of the moods.


# Challenge B: Getting Moody with Rapture XML

NSXMLParser can be a little clunky, especially when working with rather complex XML files. For smaller XML files, DOM parsers are more intuitive to use. RaptureXML is a small lightweight library that will get the job done for you.

First, navigate to github and download the library over here:

https://github.com/ZaBlanc/RaptureXML

Once downloaded, configure the sample project according to RaptureXML installation instructions.

Next, you will rewrite your parsing class. Remove all the NSXMLParser code and replace it with RaptureXML.

Build and run and your app should work as before.

## Uber Haxx0r Challenge: Saving XML

While the mood reading app is nice, it currently is just a read-only app. It's time to add some writing capabilities to it.

Before you start, download the XSWI library over here:

https://code.google.com/p/xswi/

Once you have imported the library and configured the sample app, it's time to integrate it into the app.

The first time the app runs, it should read from the sample xml and create all the RWTMood objects like the previous challenges. When the user adds a new entry and presses the save button, everything should be written to xml in the document directory. The next time the app runs, the app should read that new xml file and load in the new data. Good luck!