Assignment – "Weather"

OVERVIEW

Date Due: Thursday, April 8th by 11:59 p.m. EST. ✧ **Value:** 40 points

Features

This project focuses on using WebServices, and parsing XML responses with NSXMLParser. Additionally, you will be asked to archive application state using an NSCoding / NSKeyedArchiver.

We are going to be using data freely available from weather.gov to display a simple forcast. This application is fairly straightforward from a user's perspective. The user enters a zip in a textfield and clicks a "Go" button. In response, the application goes to weather.gov to gather data needed to display the high temperature for the next 5 days. Finally, the application remembers the last chosen zipcode and high temperatures so that on relaunch it can redisplay the previous state without having to redo any parsing.

You are being provided some starter code that will help you with dates (conversion between dates and strings). The starter code is here: http://www.cs.umd.edu/class/spring2010/cmsc498i/files/labs/Lab8_WeatherXML_resources.zip.

XML Parsing

– You must use NSXMLParser, but can submit an XPath / TouchXML version for extra credit.

WebService Request

– Use http://www.weather.gov/forecasts/xml/SOAP_server/ndfdXML.htm

– Request the max temperatures for the next 5 days for the input zip code

Saving App State

– The application should archive the zip code and enough information to be able to redispaly the 5 day forcast upon launch (without having to reissue a query)

– You may not use NSUserDefaults to save state – instead use NSCoding / NSKeyedArchiver

Extra Credit Opportunities

** You may attempt one of the following for extra credit **

1. Display the city. Along with the forcast data, weather.gov returns (when the issued request contains a zip) longitude and latitude information. Using this data, determine the city and display it in the UI somewhere. You are free to use any reverse geocoding webservice API or CoreLocation APIs **Value**: 10 points

2. Submit a sceond version of the project that uses TouchXML and XPath queries. Either submit a second project (2nd xcodeproj file...), or allow a compile time constant to switch between NSXMLParser, and TouchXML. **Value**: 10 points

SCREENSHOTS

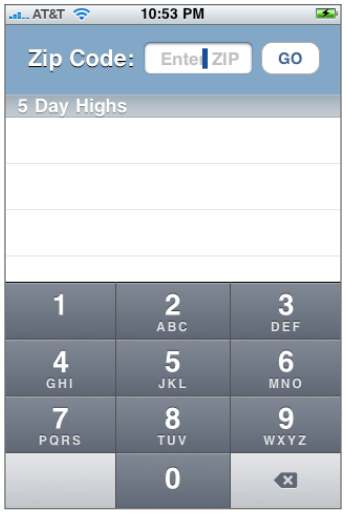

Figure 1

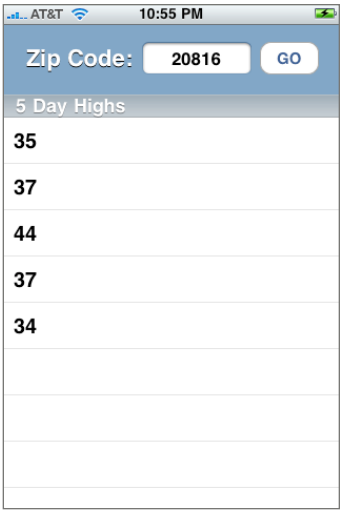

Figure 2

Figure 1 shows the application after first launch. After entering a zip code, the high temperatures for the next five days are retrieved and displayed. After quiting and re-launching, the app should display the same screen shown in Figure 2.

DETAILS

Weather Request

In addition to learning how to parse XML, part of the project is about figuring out how to create request URLs.

To help construct the weather request URL, use: http://www.weather.gov/forecasts/xml/SOAP_server/ndfdXML.htm. It lets you to graphically configure the infomation you want. Choose NDFD Data, configure all the various options and hit "Submit". A request URL will be created and issued. You can see the request in the browser URL field, and the response XML in the browser (In Safari you can see the raw XML with 'View->Show Source'). You will need to parameterize the request URL with the zip entered by the user, and dates based on the current day. To get started however, just copy and paste the request URL into your application code. Once you have the parsing working, then you can worry about parameterizing the request URL.

More information about the returned XML schema can be found here:

http://www.nws.noaa.gov/xml/docs/elementInputNames.php
http://www.nws.noaa.gov/xml/

Error Handling – Don't worry about validating input entered by the user. If the user enters a zip that turns out to be invalid, just issue the request anyways. The important thing is that the app not crash in a situation like this.

NOTES

When to Request Data

Avoid a Slow Launch – You should not contact Weather.gov during launch. Downloading data over the net can certainly be slow, and should be avoided at all cost during launch time in particular. If iPhone OS detect that your application is blocked and slow to launch it may end up assuming it is hung and kill the application on behalf of the user. Besides this unpleasantness, a slow launching application is a subpar user experience. Consider using "perform after delay" since we haven't discussed threading yet.

Date Parsing

The starter code should hande all of your date parsing needs. However, it's worth discussing how this code works just in case you need custom date parsing for your particular request URL and response. That said, you probably won't need to do that.

Dates in the request URL differ from those in the response only in that the request does not specify the timezone offset. This makes sense since the timezone should be derived from the input zip code. Dates look like "2010-03-17T08:00:00" in the request URL, and "2010-03-17T08:00:00-05:00" in the resulting response. NSDateFormatter is used to convert between NSDates and NSStrings by specifying a custom date format string. Dates in the request URL use the following format: "yyyy-MM-dd'T'HH:mm:ss". Dates in the response are formatted as "yyyy-MM-dd'T'HH:mm:ssZ". A wealth of information about date formatting, in particular standard date-time formats, can be found here:

http://www.w3.org/TR/xmlschema-2/#dateTime
http://unicode.org/reports/tr35/tr35-4.html#Date_Format_Patterns
http://unicode.org/reports/tr35/tr35-6.html#Date_Format_Patterns
http://developer.apple.com/mac/library/documentation/cocoa/conceptual/DataFormatting/Articles/dfDateFormatting10_4.html
http://cocoawithlove.com/2009/05/simple-methods-for-date-formatting-and.html

Using TouchXML and XPath

If you choose to work on the XPath based parsing extra credit, you will need to include TouchXML in your project. TouchXML can be found here: http://code.google.com/p/touchcode/wiki/TouchXML. Download the project, then copy all of the source and headers into your projects. TouchXML is based on libxml2, so you will need to configure your project to have libxml2 headers in your search path, and to link against the libxml2 library. To do so, select your main application target, and then choose File->GetInfo. Select the "Build" tab, and configure values for the "All Configurations" configuration. Set the "Header Search Path" to include "/usr/include/libxml2, and set the "Other Linker Flags" to include "-lxml2". Now you are good to go!

A quick Xpath syntax reference can be found here: http://www.w3schools.com/Xpath/xpath_syntax.asp