CSCI 428/628 - Advanced Web Programming - Assignment: Reading JSON from a Server

---

Overview

In this assignment you will write an app to read a JSON file from a remote server and display the file contents in an easily readable and labeled manner.

The JSON file follows the format shown here:

```
{
"clients":
[
{"name": "Jim Lincoln",
"profession": "Musician",
"dob": "Oct. 2, 1945",
"children":[
"Jane",
"Peter"
]
},
{
"name": "Mary Jones",
"profession": "Programmer",
"dob": "April 2, 1980",
"children":[
"Joe",
"Sue"
]
},

// more entries....


{
"name": "Anne Sprague",
"profession": "Urban Planner",
"dob": "January 12, 1970",
"children":[
  ]
}
]
}
```

The JSON file can be found at "http://www.seasite.niu.edu/cs680Android/JSON/Client_list_json.txt" and will be served by that site's web server, so it can be read via an http Get command. (It can be displayed using a web browser if you are curious to see the exact whole file.) Since the Get command will result in a data stream that will be read by your program's code, and since these read calls will block if the next chunk of data from the server is delayed, your code to retrieve the JSON data should execute in a separate Thread of execution.

Note that this file consists of a named *array* of (1 - many) objects (*clients*). In this case each object consists of information about one person, including a list/array of that person's children (0 - many).

Incidentally, http://www.jsonlint.com has a nice online JSON syntax checker if you need to create your own JSON file and need to find out if there are any syntax errors that would trip up your parsing code.

Platform Implementation Notes

Android

Create a simple app screen with text instructing the user to press the *Go* button to download the file. This text should also inform the user that the app's output will be to the LogCat console and app screen. Obviously, the app should include a functioning *Go* button.

Do not spend time on an elegant screen display of the JSON information - just echo it to LogCat and also create a big multiline String and display it in a TextView. Each person in the file should be displayed as follows:

Name: Mary Jones
Profession: Programmer
DOB: April 2, 1980
Children: Joe, Sue

Extend AsynchTask to do the JSON file read in a separate Thread. Implement this as an inner class so it will have access to the TextView display widget. Write *doInBackground()* to set up the http connection, read the file (building a StringBuffer object, creating a String from it, and then returning a JSONObject created from that String. *onPostExecute()* (or a called custom method in this class) will then parse the JSONObject returned by *doInBackground()* and display the file contents on LogCat and screen.

Your code should write informative error messages to LogCat in the event that there is a parsing error on the JSON file or other error condition.

iOS

Parse JSON-formatted data and display them in two views -- i.e. using the master-detail iOS application template. The first view contains a UITableview. Each cell of the table should contain a name and profession. Clicking on a table cell should take the user to the detail view. The title of the detail view should reflect the selected name and the detail view must contain a profession, date of birth and children (if any).

An excellent code example of working with JSON in iOS 5 and up can be found here:

http://www.raywenderlich.com/5492/working-with-json-in-ios-5

Windows Phone

Use a *WebClient* to connect to the Uri identified above. Fetch the json file using *DownloadStringAsync*

and attach an event handler to the *DownloadStringCompleted* event. In the DownloadStringCompleted event handler, put the fetched json string into an array of byte and then create a *MemorySream* from that byte array. Now create an instance of *DataContractJsonSerializer* and use it to ReadObject() from the stream. The object that is read should contain a List<Client> where a Client object contains name, profession, dob, and a List<string> for the children. Assign the List<Client> to the page's DataContext and put a ListBox in the UI that uses data binding to display the pieces of information in each client object.