Lab – Foundations

OVERVIEW

Date Due: Thursday, Feb. 18th by 11:59 p.m. EST. ✧ **Value:** 10 points

Objective: Practice using Foundation collection classes (NSArray, NSDictionary) and property lists.

This week you will be adding some new functionality to the ToDoEvent model object from Lab1. We'd like to be able to display events sorted by various criteria. Your application will be able to display ordered events (multi-line cells with titles, and dates), title strings, and date strings. We also want the ability to save events and load them from disk, specifically using a plist format. You will be writing code to make all of this possible.

ASSIGNMENT

1. Download today's project from http://www.cs.umd.edu/class/spring2010/cmsc498i/files/lab2b_resources.zip. This contains some pre–written code you to get you started. It will look somewhat familiar to you. Open ToDoEvent.h, and check out the methods listed in the "Assignment" category. This is what you need to implement.

2. Build and Run the application. At this point, the application will display no data. This is expected...

3. Reading Property Lists. Implement code to read a property list, and return an array of ToDoEvent objects.

    - To get you started, a SampleEvents.plist file is included with the project. It shows the expected plist format, and provides a place to start. Take a look and make sure you understand the plist format.

    - Implement the +`eventsFromContentsOfPropertyListURL:` factory method. This method will take a URL which references a plist, in this case SampleEvents.plist. It should read the plist and convert what it finds into an array of ToDoEvent objects. You should be able to accomplish this using only NSArray, NSDictionary, and ToDoEvent APIs.

4. Build and Run the application. You should see some events displayed on screen.

5. Try clicking on any of the categories in the navigation bar area: Event, Dates, Titles. Unfortunately, clicking on these now will cause the program to crash, due to an exception being raised. Clicking on these will eventually cause data to be displayed sorted by the selected property. For example, clicking on "Titles" should cause titles to be displayed sorted. To make this happen, you need to write some support code. Read on…

6. Sorting Data. To add sorting support, implement the ToDoEvent class methods that start with "+sorted". They each should return an array of sorted objects. Each array will contain different item types. You will be returning an array of NSStrings (+`sortedTitlesFromEvents:`), ToDoEvents (+`sortedEventsFromEventsByDate:`), and NSDates (+`sortedDatesFromEvents:`).

7. Writing Your Own Data. You can now display sorted data read from a file. Instead of using SampleEvents.plist, we now want to use your own data.

    7.1. When a user clicks the "Generate" button, the starter code will call your +sampleToDoEvents method. You should implement this method to create 5 or more events, insert them into an array and return them to the caller.

    7.2. After acquiring the events, the starter code will ask for each ToDoEvent's -dictionaryRepresentation. These are inserted into an array and then written out to disk by the starter code. From this point on, whenever your application runs, it will load data from this newly created plist instead of SampleEvents.plist.

    7.3. If you have any problems, the path of the file written is logged to the gdb console so you can inspect the data. To complete this step, you must implement +sampleToDoEvents, and -dictionaryRepresentation.

8. Build and Run. Make sure all the sorted views work, and that your data is used even after you quit and relaunch.

9. You are done.

NOTES

Exceptions and Cocoa Collection Classes

Cocoa's collection classes (NSArray, NSDictionary, NSSet, ...) will raise exceptions if you do something that doesn't make any sense. In Cocoa, excptions are used only to indicate programming errors by convention. For example, say you write some code to load a file and there is an error loading the file (say due to a network error, or permissions problem). In Cocoa, this sort of thing would not raise an exception. However, say your NSArray contains 10 items, and you try to access the item at index 20. In Cocoa, this does raise an exception.

What sort of things raise an exception in the Cocoa collection classes?

- Attempts to use an invalid index, e.g. `[array objectAtIndex:20]` when the array only has 10 items.

- Inserting *nil* into an array. e.g. `[mutableArray addObject:nil]`

- Mutating an immutable object. e.g. `NSArray *array = [NSArray array]; [array addObject: ......]`

- Using *nil* as a dictionary key. e.g. `[dictionary setObject:foo forKey:nil]`

Want to catch an exception in the debugger and see the backtracks? If necessary, just add a breakpoint on the objective-c runtime method that is used to throw exceptions: objc_exception_throw( ). So, in the gdb console, you would do this: (gdb) b objc_exception_throw.