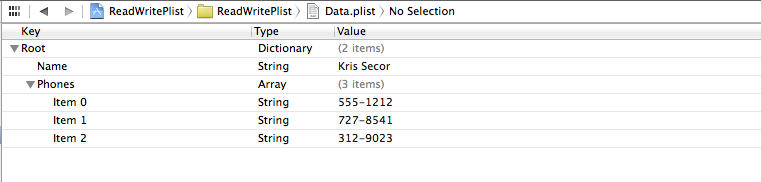
Reading and Writing from a Plist -

1. Create a new View Application Project
2. Create a new plist. Data.plist seems like a good idea
3. Create a field for Name and Phones



1. Create 4 text fields. 3 for phone numbers and 1 for a name
2. Add a button and some labels
3. Create Outlets for the text fields and an action for the button

If it helps, my h file looks like this:

#import "UIKit/UIKit.h"

@interface PlistTutorialViewController : UIViewController

{

IBOutlet UITextField \*nameEntered;

IBOutlet UITextField \*homePhone;

IBOutlet UITextField \*workPhone;

IBOutlet UITextField \*cellPhone;

NSString \*personName;

NSMutableArray \*phoneNumbers;

}

@property (nonatomic, retain) UITextField \*nameEntered;

@property (nonatomic, retain) UITextField \*homePhone;

@property (nonatomic, retain) UITextField \*workPhone;

@property (nonatomic, retain) UITextField \*cellPhone;

@property (nonatomic, retain) NSString \*personName;

@property (nonatomic, retain) NSMutableArray \*phoneNumbers;

- (IBAction) saveData;

- (IBAction) textFieldReturn:(id)textField;

@end

1. This is the ViewDidLoad Method:

// Data.plist code

// get paths from root direcory

NSArray \*paths = NSSearchPathForDirectoriesInDomains (NSDocumentDirectory, NSUserDomainMask, YES);

// get documents path

NSString \*documentsPath = [paths objectAtIndex:0];

// get the path to our Data/plist file

NSString \*plistPath = [documentsPath stringByAppendingPathComponent:@"Data.plist"];

// check to see if Data.plist exists in documents

if (![[NSFileManager defaultManager] fileExistsAtPath:plistPath])

{

// if not in documents, get property list from main bundle

plistPath = [[NSBundle mainBundle] pathForResource:@"Data" ofType:@"plist"];

}

// read property list into memory as an NSData object

NSData \*plistXML = [[NSFileManager defaultManager] contentsAtPath:plistPath];

NSString \*errorDesc = nil;

NSPropertyListFormat format;

// convert static property liost into dictionary object

NSDictionary \*temp = (NSDictionary \*)[NSPropertyListSerialization propertyListFromData:plistXML mutabilityOption:NSPropertyListMutableContainersAndLeaves format:&format errorDescription:&errorDesc];

if (!temp)

{

NSLog(@"Error reading plist: %@, format: %d", errorDesc, format);

}

// assign values

self.personName = [temp objectForKey:@"Name"];

self.phoneNumbers = [NSMutableArray arrayWithArray:[temp objectForKey:@"Phones"]];

// display values

nameEntered.text = \_personName;

homePhone.text = [phoneNumbers objectAtIndex:0];

workPhone.text = [phoneNumbers objectAtIndex:1];

cellPhone.text = [phoneNumbers objectAtIndex:2];

1. This is the saveData method called by the button:

- (IBAction) saveData

{

// get paths from root direcory

NSArray \*paths = NSSearchPathForDirectoriesInDomains (NSDocumentDirectory, NSUserDomainMask, YES);

// get documents path

NSString \*documentsPath = [paths objectAtIndex:0];

// get the path to our Data/plist file

NSString \*plistPath = [documentsPath stringByAppendingPathComponent:@"Data.plist"];

// set the variables to the values in the text fields

self.personName = nameEntered.text;

self.phoneNumbers = [[NSMutableArray alloc] initWithCapacity:3];

[phoneNumbers addObject:homePhone.text];

[phoneNumbers addObject:workPhone.text];

[phoneNumbers addObject:cellPhone.text];

// create dictionary with values in UITextFields

NSDictionary \*plistDict = [NSDictionary dictionaryWithObjects: [NSArray arrayWithObjects: personName, phoneNumbers, nil] forKeys:[NSArray arrayWithObjects: @"Name", @"Phones", nil]];

NSString \*error = nil;

// create NSData from dictionary

NSData \*plistData = [NSPropertyListSerialization dataFromPropertyList:plistDict format:NSPropertyListXMLFormat\_v1\_0 errorDescription:&error];

// check is plistData exists

if(plistData)

{

// write plistData to our Data.plist file

[plistData writeToFile:plistPath atomically:YES];

}

else

{

NSLog(@"Error in saveData: %@", error);

[error release];

}

}

1. Add the resign Responder:

- (IBAction) textFieldReturn:(id)textField

{

[textField resignFirstResponder];

}

Conclusion. Next we need to append to a plist and read then back in a table cell