

Разработване на приложения за iOS



Петко Пенчев, TapBits.com

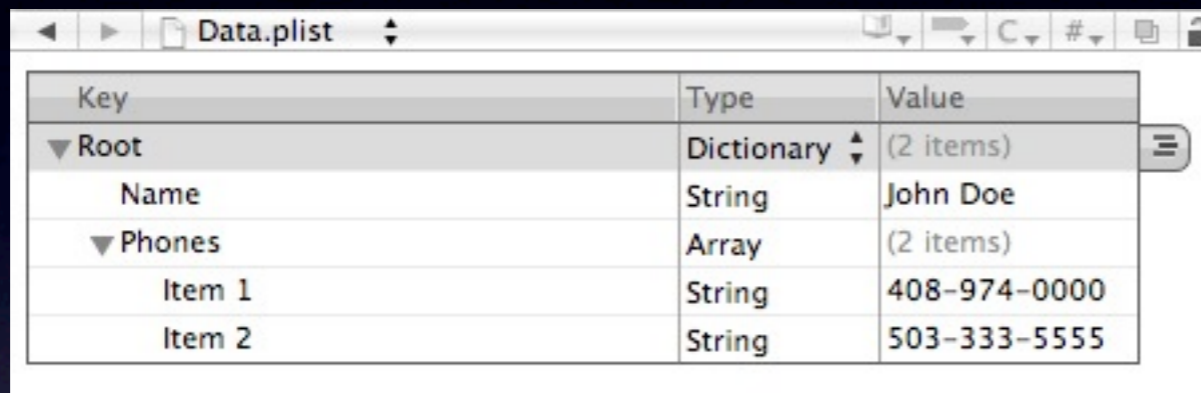
PropertyList / plist

- Списък от свойства
- Подходящ за съхранение на малък обем от данни
- NSArray, NSDictionary, NSString, NSData, NSDate, NSNumber

PropertyList / plist

- XML (по-гъвкави, могат да бъдат ръчно редактирани)
- Binary (по-компактни)
- Old-style ASCII (read-only, наследени от OpenStep)

PropertyList / plist



The screenshot shows a window titled "Data.plist" with a table representing the plist structure. The table has three columns: Key, Type, and Value.

Key	Type	Value
▼ Root	Dictionary (2 items)	
Name	String	John Doe
▼ Phones	Array (2 items)	
Item 1	String	408-974-0000
Item 2	String	503-333-5555

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Name</key>
  <string>John Doe</string>
  <key>Phones</key>
  <array>
    <string>408-974-0000</string>
    <string>503-333-5555</string>
  </array>
</dict>
</plist>
```

Чтение от plist

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    NSString *errorDesc = nil;
    NSDictionary *format;
    NSString *plistPath;

    NSString *rootPath = [NSSearchPathForDirectoriesInDomains(
        NSDocumentDirectory,
        NSUserDomainMask, YES) objectAtIndex:0];

    plistPath = [rootPath stringByAppendingPathComponent:@"Data.plist"];

    if (![NSFileManager defaultManager] fileExistsAtPath:plistPath) {
        plistPath = [[NSBundle mainBundle] pathForResource:@"Data" ofType:@"plist"];
    }
    ...
}
```

Чтение от plist

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

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

if (!temp) {
    NSLog(@"Error reading plist: %@, format: %d", errorDesc, format);
}

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

...
return YES;
}
```

Запазване на plist

```
- (NSApplicationTerminateReply)applicationShouldTerminate:(NSApplication *)sender {
    NSString *error;
    NSString *rootPath = [NSSearchPathForDirectoriesInDomains(
        NSDocumentDirectory,
        NSUserDomainMask, YES)
        objectAtIndex:0];
    NSString *plistPath = [rootPath stringByAppendingPathComponent:@"Data.plist"];

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

    NSData *plistData = [NSPropertyListSerialization dataFromPropertyList:plistDict
        format:NSPropertyListXMLFormat_v1_0
        errorDescription:&error];

    if(plistData) {
        [plistData writeToFile:plistPath atomically:YES];
    }
    else {
        NSLog(error);
        [error release];
    }

    return NSTerminateNow;
}
```

Архивиране & Сериализация

- Подходи за създаване на архитектурно независими `byte stream` от иерархично структури данни
 - запазване във файл
 - трансферирани към друг процес
 - прехвърляни по мрежа
- Object Graph

Архивиране

- NSCoder
- Механизъм за създаване на 'Graph persistent' обект от всякакъв друг тип
- Пример: Interface Builder
- Задължително е обекта да имплементира NSCoder protocol
 - (void)encodeWithCoder:(NSCoder *)coder;
 - initWithCoder:(NSCoder *)coder;

Архивиране

Sequential archives*

NSArchiver
NSUnarchiver

Keyed archives

NSKeyedArchiver
NSKeyedUnarchiver

Distributed archives

NSPortCoder

NSObject

NSObject

* Употребата не се препоръчва от Mac OS X 10.2

Encoding & Decoding

```
- (void)encodeWithCoder:(NSCoder *)coder {
    [super encodeWithCoder:coder];

    [coder encodeObject:mapName forKey:@"MVMapName"];
    [coder encodeFloat:magnification forKey:@"MVMagnification"];
    [coder encodeObject:legendView forKey:@"MVLegend"];
    [coder encodeConditionalObject:auxiliaryView forKey:@"MVAuxView"];
}

- (id)initWithCoder:(NSCoder *)coder {
    self = [super initWithCoder:coder];

    mapName = [[coder decodeObjectForKey:@"MVMapName"] retain];
    legendView = [[coder decodeObjectForKey:@"MVLegend"] retain];
    auxiliaryView = [coder decodeObjectForKey:@"MVAuxView"];
    magnification = [coder decodeFloatForKey:@"MVMagnification"];

    return self;
}
```

Създаване на обектен архив

```
MapView *myMapView;
```

```
NSString *archivePath = [NSTemporaryDirectory() stringByAppendingPathComponent:@"Map.archive"];
```

```
result = [NSKeyedArchiver archiveRootObject:myMapView toFile:archivePath];
```

Възстановяване от обектен архив

```
MapView *myMapView;
```

```
NSString *archivePath = [NSTemporaryDirectory() stringByAppendingPathComponent:@"Map.archive"];
```

```
myMapView = [NSKeyedUnarchiver unarchiveObjectWithFile:archivePath];
```

Сериализиране

- Съхранява само стойностите и тяхната позиция в иерархията
- Не съхранява типа на данните, връзката помежду им
- Позволява запазване на инстанции на NSArray, NSDictionary, NSString, NSDate, NSNumber и NSData

Serialization plist

```
NSData *dataRep;

NSString *errorStr = nil;
NSDictionary *propertyList;

propertyList = [NSDictionary dictionaryWithObjectsAndKeys:
                @"Javier", @"FirstNameKey",
                @"Alegria", @"LastNameKey", nil];

dataRep = [NSPropertyListSerialization dataFromPropertyList: propertyList
                                       format: NSPropertyListXMLFormat_v1_0
                                       errorDescription: &errorStr];

if (!dataRep) {
    // Handle error
}
```

Deserialization plist

```
NSData *dataRep; // Assume this exists
NSString *errorStr = nil;
NSDictionary *propertyList;
NSPropertyListFormat format;

propertyList = [NSPropertyListSerialization propertyListFromData: dataRep
                                             mutabilityOption: NSPropertyListImmutable
                                             format: &format
                                             errorDescription: &errorStr];

if (!propertyList) {
    // Handle error
}
```

User Defaults

- NSUserDefaults
- поддържа само обекти, които могат да бъдат сериализирани в plist
- обектите, които имплементират NSCoder могат да бъдат архивирани в NSData (plist съвместим тип)

Съхранение в NSUserDefaults

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {  
  
    ...  
    NSInteger pushBadgeCount=application.applicationIconBadgeNumber;  
    NSLog(@"application.applicationIconBadgeNumber: %d", pushBadgeCount);  
  
    [[NSUserDefaults standardUserDefaults] setObject:[NSNumber numberWithInt:pushBadgeCount]  
                                                    forKey:@"badgecount"];  
    [[NSUserDefaults standardUserDefaults] synchronize];  
  
    ...  
}  
  
...  
lastvalid_username = [[NSUserDefaults standardUserDefaults] objectForKey:@"lastvalid_username"];  
...  
}
```

Съхранение в NSUserDefaults

```
-(void)saveToUserDefaults:(NSString*)Value forKey:(NSString *)key
{
    NSUserDefaults *standardUserDefaults = [NSUserDefaults standardUserDefaults];

    if (standardUserDefaults) {
        [standardUserDefaults setObject:Value forKey:key];
        [standardUserDefaults synchronize];
    }
}

-(NSString*)retrieveFromUserDefaults:(NSString *)key
{
    NSUserDefaults *standardUserDefaults = [NSUserDefaults standardUserDefaults];
    NSString *val = nil;

    if (standardUserDefaults)
        val = [standardUserDefaults objectForKey:key];

    return val;
}
```

NSUserDefaults + NSColor

```
NSColor *aColor;
aColor = ..
// съхранение на aColor в user defaults по ключ aKey

NSString *aKey = @"my preferred color";
NSData *theData=[NSArchiver archivedDataWithRootObject:aColor];
[[NSUserDefaults standardUserDefaults] setObject:theData forKey:aKey];
...

// извличане на aColor от user default по ключ aKey
NSColor * aColor =nil;

NSString *aKey = @"my preferred color";
NSData *theData=[[NSUserDefaults standardUserDefaults] dataForKey:aKey];

if (theData != nil)
    aColor =(NSColor *)[NSUnarchiver unarchiveObjectWithData:theData];
...

```

NSUserDefaults + Category

```
// NSUserDefaults+myColorSupport.h
#import <Foundation/Foundation.h>

@interface NSUserDefaults(myColorSupport)
- (void)setColor:(NSColor *)aColor forKey:(NSString *)aKey;
- (NSColor *)colorForKey:(NSString *)aKey;
@end
```

```
// NSUserDefaults+myColorSupport.m
#import "NSUserDefaults+myColorSupport.h"

@implementation NSUserDefaults(myColorSupport)

- (void)setColor:(NSColor *)aColor forKey:(NSString *)aKey
{
    NSData *theData=[NSArchiver archivedDataWithRootObject:aColor];
    [self setObject:theData forKey:aKey];
}

- (NSColor *)colorForKey:(NSString *)aKey
{
    NSColor *theColor=nil;
    NSData *theData=[self dataForKey:aKey];
    if (theData != nil)
        theColor=(NSColor *)[NSUnarchiver
            unarchiveObjectWithData:theData];
    return theColor;
}
@end
```

Файлова система

- Създаване на пътека до файл/директория
- Четене и запис на файл

- (NSString *)stringByAppendingPathComponent:(NSString *)component;

- (NSString *)stringByDeletingLastPathComponent;

- (BOOL)writeToFile:(NSString *)path atomically:(BOOL)flag
encoding:(NSStringEncoding)encoding //e.g.ASCII,ISOLatinI,etc. error:(NSError **)error;

- (NSString *)stringWithContentsOfFile:(NSString *)path usedEncoding:(NSStringEncoding *)encoding
error:(NSError **)error;

Файлова система

```
NSFileManager *filemgr;  
filemgr = [NSFileManager defaultManager];  
  
..  
currentPath = [filemgr currentDirectoryPath];
```

// Достъп до Documents

```
NSArray *dirPaths;  
NSString *docsDir;
```

```
dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,  
        NSUserDomainMask, YES);
```

```
docsDir = [dirPaths objectAtIndex:0];
```

Под IOS резултата ще бъде

```
/var/mobile/Applications/<app id>/Documents
```

Файлова система

```
// Достъп до Tmp и Home директорийте
NSTemporaryDirectory C function as follows:
NSString *tmpDir = NSTemporaryDirectory();

NSString *homeDir = NSHomeDirectory();
...

// Създаване на директория
NSFileManager *filemgr;
NSArray *dirPaths;
NSString *docsDir;
NSString *newDir;

filemgr = [NSFileManager defaultManager];
dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES);

docsDir = [dirPaths objectAtIndex:0];
newDir = [docsDir stringByAppendingPathComponent:@"data"];

if ([createDirectoryAtPath:newDir withIntermediateDirectories:YES attributes:nil error:NULL] == NO)
{
    // Failed to create directory
}
[filemgr release];
```

Файлова система

```
// Промяна на работната директория
```

```
NSFileManager *filemgr;
```

```
NSArray *dirPaths;
```

```
NSString *docsDir;
```

```
filemgr =[NSFileManager defaultManager];
```

```
dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,  
NSUserDomainMask,YES);
```

```
docsDir = [dirPaths objectAtIndex:0];
```

```
if ([filemgr changeCurrentDirectoryPath: docsDir] == NO)
```

```
{
```

```
    // Directory does not exist – take appropriate action
```

```
}
```

```
[filemgr release];
```

Файлова система

```
// Изтриване на директория
```

```
NSFileManager *filemgr;  
NSArray *dirPaths;  
NSString *docsDir;  
NSString *newDir;
```

```
filemgr =[NSFileManager defaultManager];
```

```
dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,  
NSUserDomainMask, YES);
```

```
docsDir = [dirPaths objectAtIndex:0];
```

```
newDir = [docsDir stringByAppendingPathComponent:@"data"];
```

```
if ([filemgr removeItemAtPath:newDir error:nil] == NO)  
{  
    // Directory removal failed.  
}  
[filemgr release];
```

Файлова система

// Показване на списък със съдържанието на директория

```
NSFileManager *filemgr;
```

```
NSArray *filelist;
```

```
int count;
```

```
int i;
```

```
filemgr =[NSFileManager defaultManager];
```

```
filelist = [filemgr contentsOfDirectoryAtPath:@"/" error:NULL];
```

```
count = [filelist count];
```

```
for (i = 0; i < count; i++)
```

```
    NSLog(@"%@@", [filelist objectAtIndex:i]);
```

```
[filemgr release];
```

NSBundle

- Път до ресурсите в приложението

```
NSString *myFilePath = [[NSBundle mainBundle] pathForResource:@"MyFile" ofType:@"txt"];
```

- Зареждане на съдържание в NSString

```
NSString *myFileContents = [NSString stringWithContentsOfFile:myFilePath  
                           encoding:NSUTF8StringEncoding  
                           error:nil];
```

- Запазване на съдържанието на NSString

```
NSString docDir = [arrayPaths objectAtIndex:0];  
NSString *newFilePath = [docDir stringByAppendingString:@"/NewFile.txt"];  
NSString *s = @"Примерно съдържание на стринг";
```

```
[s writeToFile:newFilePath  
   atomically:YES  
   encoding:NSUTF8StringEncoding  
   error:nil];
```