Работа с файлове и XML. Потребителски настройки.

Георги Пенчев

За какво ще си говорим

- Файловата система
- Работа с файлове
- Сериализация/десериализация
- Потребителски настройки
- Настройки на приложението
- XML: SAX парсер
- XML: DOM алтернативи

iOS Application Sandbox

- Application Bundle *vs* Application Sandbox
- Съдържание на нашия sandbox
  - Documents
  - Library
  - tmp
- Достъп до директориите
  - NSSearchPathForDirectoriesInDomains (NSDocumentDirectory, NSUserDomainMask,YES)
  - NSTemporaryDirectory()

NSFileManager

- Всякакви операции с файлове и директории
- Инициализация:

  [NSFileManager defaultManager]
- URLforDirectory - предпочитано

  NSURL docsurl = [fm **URLForDirectory**:*
  NSDocumentDirectory
  inDomain:NSUserDomainMask
appropriateForURL:nil create:YES error:&err];

Работа с директории

- Създаване на директории

  createDirectoryAtPath:*myfolder withIntermediateDirectories:NO attributes:nil error:&err*
- Пътища - ползваме

  stringByAddingPathComponent
- Съдържание на директория
  - *contentsOfDirectoryAtPath:* (плоско)
  - *subpathsOfDirectoryAtPath:* (дълбоко)
  - *NSDirectoryEnumerator* & enumeratorAtPath*
- Текуща директория

  changeCurrentDirectoryPath

Операции с файлове

- Съществуване

  fileExistsAtPath:error:
- Копиране

  copyFileAtPath:toPath:error:
- Местене

  moveFileAtPath:toPath:error:
- Триене

  removeFileAtPath:error:
- Логично, нали?

Четене и писане по файлове

- Използваме *NSHandle*
- Съвсем като в C++

```
NSData *databuffer;
NSFileHandle *file = [NSFileHandle
fileHandleForReadingAtPath: @"myfile.txt"];
if (file == nil) NSLog(@"Failed to open file");
[file seekToFileOffset: 10];
databuffer = [file readDataOfLength: 5];
[file closeFile];
[file release];
```

- Много класове поддържат автоматично писане във файл

  [myNSData writeToFile:...]

Сериализация в iOS

- Две основни стратегии
- Property Lists
  - удобно
  - но само за определен тип данни
  - рядко можем да минем само с него
- Archiving
  - трябва да си напишем сериализацията
  - по-гъвкаво

Property Lists

- Единични класове

  NSData, NSString, NSNumber, NSDate

  (и mutable версиите им)
- Колекции

  NSArray, NSDictionary (и mutable версиите им)
- Сериализация

  [myArray writeToFile:@"test.plist" atomically:YES]
- Десериализация

  [myArray initWithContentsOfFile:@"test.plist"]

NSData и NSCoding

- *NSData* - универсалният клас за двоични данни в Objective-C
- *NSCoding* - протокол, който дефинира, че класът се сериализира
  - - (void)encodeWithCoder:(NSCoder *)encoder
  - - (id)initWithCoder:(NSCoder *)decoder

Сериалзиация

- Имплементираме енкодинга

```
- (void)encodeWithCoder:(NSCoder *)encoder {
    [super encodeWithCoder:encoder];
    [encoder encodeObject:foo forKey:kFooKey];
    [encoder encodeInt:someInt forKey:kSomeIntKey];
}
```

- Сериализираме

```
NSMutableData *data = [[NSMutableData alloc] init];
NSKeyedArchiver *archiver = [[NSKeyedArchiver alloc]
initForWritingWithMutableData:data];
[archiver encodeObject:myObject forKey:@"myKey"];
[archiver finishEncoding];
BOOL success = [data writeToFile:@"myArchive.bin" atomically:YES];
```

Десериализация

- Имплементираме декодинга

```
- (id)initWithCoder:(NSCoder *)decoder {
if (self = [super initWithCoder:decoder]) {
        foo = [[decoder decodeObjectForKey:@"foo"] retain];
        someInt = [decoder decodeIntForKey:@"myInt"];
}
return self;
}
```

- Десериализираме

```
NSData *data = [[NSData alloc] initWithContentsOfFile:path];
NSKeyedUnarchiver *unarchiver = [[NSKeyedUnarchiver alloc]
initForReadingWithData:data];
self.object = [unarchiver decodeObjectForKey:@"myKey"];
[unarchiver finishDecoding];
```

Потребителски настройки

- iOS поддържа лек компонент за съхранение на прости данни - *NSUserDefaults*
- Работи като *NSDictionary*
- Варианти за задаване на стойности
  - Някакъв UI
  - Settings App
  - Прозрачно от потребителя

Използване на NSUserDefaults

- Писане и четене

```
[[NSUserDefaults standardUserDefaults] setFloat:4.0 forKey:@"
myPreciousFloat"];
NSNumber* n = [[NSUserDefaults standardUserDefaults] objectForKey:@"
myObjectNumber"];
```

- Defaults

```
[[NSUserDefaults standardUserDefaults] registerDefaults:
    [NSDictionary dictionaryWithObjectsAndKeys:
    [NSNumber numberWithInt: 4],
    @"cardMatrixRows",
    [NSNumber numberWithInt: 3],
    @"cardMatrixColumns",
    nil]];
```

Използване на NSUserDefaults

- Кога да записваме данните
  - *applicationWillEnterBackground* в делегата
  - UIApplicationWillEnterBackground съобщение
- Кога да ги четем
  - при стартиране
  - при UIApplicationWillEnteForeground

Настройки на приложението

- Можем да добавяме опции в Settings App
  - не е сигурно, че потребителят ще ги нагласи
  - изисква излизане от приложението
- Добавяне на Settings.bundle
- Редакция на Root.plist
  - Identifier = key в NSUserDefaults
- Типове настройки
  - PSTextFieldSpecifier

XML 101

- Универсален формат за запазване на по-сложни данни
- Строга валидация
- Добра алтернатива ако моделът ни е лесно преносим към и от XML
- Обработка на XML
  - SAX
  - DOM
  - XPath

NSXMLParser

- iOS поддържа само SAX парсер
- Инициализация
- Обработка - *NSXMLParserDelegate*
  - *parser:didStartElement:namespaceURI:qualifiedName:attributes:*
  - *parser:didEndElement:namespaceURI:qualifiedName:*
  - *parser:foundCharacters:*

Проблеми на SAX

- Сложни йерархии
- *foundCharacters* се вика много пъти
- Запазване на състояние
- Едно решение - промяна на делегата по нивата

DOM алтернативи

- iOS не поддържа DOM
- libxml2 е C библиотека, която можете да ползвате във вашите програми
- 3rd party Objective-C варианти
  - TBXML - бързо, но ограничено, read-only
  - TouchXML - същото + XPath
  - KissXML - същото + писане и валидация
  - TinyXML - бързо, read-write, TinyXPath
  - GDataXML
  - http://www.raywenderlich.com/553/how-to-chose-the-best-xml-parser-for-your-iphone-project

libxml2

```
NSString* path = ……...
const char* filename = [path UTF8String];
xmlDocPtr doc  = xmlReadFile(filename, NULL,0);
xmlNode* root = xmlDocGetRootElement(doc);
traverse(root);
xmlFreeDoc(doc);
xmlCleanupParser();
void traverse(xmlNode* node) {
	xmlNode* curNode = NULL;
	for (curNode = node; curNode != NULL; curNode = curNode->next) {
		if (curNode->typo == XML_ELEMENT_NODE) {
			if (strcmp(curNode->name, "person") == 0) NSLog(@"person");
			else if (strcmp(curNode->name, "firstName") == 0) NSLog(@"
firstName: %s", curNode->children->content);
		traverse(curNode->children);
	}
}
```

TBXML

```
TBXML * tbxml = [[TBXML tbxmlWithXMLFile:@"books.xml"] retain];
TBXMLElement * root = tbxml.rootXMLElement;


TBXMLElement* child = [TBXML childElementNamed:@"author"
parentElement:root];
NSString* name = [TBXML valueOfAttributeNamed:@"name" forElement:
author];
NSString* description = [TBXML textForElement:descriptionElement];
```

TBXML

```
- (void) traverseElement:(TBXMLElement *)element {
    do {
        NSLog(@"%@",[TBXML elementName:element]);
        TBXMLAttribute * attribute = element->firstAttribute;
        while (attribute) {
        NSLog(@"%@->%@ = %@",
                        [TBXML elementName:element],
                        [TBXML attributeName:attribute],
                        [TBXML attributeValue:attribute]);
        attribute = attribute->next;
        }
        if (element->firstChild)
            [self traverseElement:element->firstChild];
    }
    while ((element = element->nextSibling));
}
```

KissXML

```
NSError *error = nil;
    DDXMLDocument *theDocument = [[DDXMLDocument alloc] initWithXMLString:source options:0 error:&error];
    NSArray *results = [theDocument nodesForXPath:@"/bookstore/book[price>35]" error:&error];
    for (DDXMLElement *book in results) {
        NSString *category = [[book attributeForName:@"category"] stringValue];
        NSLog(@"category:%@",category);
        for (int i = 0; i < [book childCount]; i++) {
            DDXMLNode *node = [book childAtIndex:i];
            NSString *name = [node name];
            NSString *value = [node stringValue];
            NSLog(@"%@:%@",name,value);
        }
    }
```