Knihovna Foundation

Seminář iOS

Martin Hrubý, FIT VUT v Brně

Úvod

- Foundation je nosná páteř všech aplikací v iOS a OS X.
- Je společná pro oba systémy, tzn. jádro vaší aplikace je přenositelné.
- http://opensource.apple.com/source/CF/CF-855.17/

Zajímavé části

- Kolekce - NSArray, NSSet, NSDictionary, …
  - + NSPredicate, NSSortDescriptor
- NSCoding, NSKeyValueCoding protokol, NSData
- NSCalendar, NSDate
- NSFileManager (a dokumenty), NSUserDefaults
- NSOperation, NSTimer, NSRunLoop, NSNotif…
- NSURLSession a další

Kolekce

- Non-mutable (bez “Mutable” v názvu) — pouze pro čtení. Pokud není důvod mít mutable verzi, vždy provést konverzi.
- Mutable — lze zapisovat. Typické použití:
  - dočasné — vytvoření pole v nějakém cyklu.
  - NSArrayController — obdoba NSFetchedResultsController pro OS X.
  - jinde atribut NSMutableArray sotva najdete.

NSArray

- Vytvoření:
  - z pole: [NSArray arrayWithArray: pole],
  - konstanta v programu @[obj1, obj2, …],
  - z množiny [aSet allObjects], pořadí prvků.
  - operací nad polem.
- [anArray count], anArray.count
- [anArray objectAtIndex:]
  - dnes již anArray[idx]. Pozor na rozsah (Exception)!

Proxy nad kolekcí

```
@interface Zakaznik : NSObject
@property (readonly, nonatomic, strong) NSArray *ucty;
@end

@implementation Zakaznik
-(NSArray *) ucty {
    return [self mutableArrayValueForKey: @“pucty”];
}
-(NSUInteger) countOfPucty {
    return …;
}
-(id) objectInPuctyAtIndex:(NSUInteger)index {
    return …;
}
@end

Zakaznik *zak = [[Zakaznik alloc] init];
NSLog@(“Ucet %@“, zak.ucty[12312848481]);
```

NSSet

- Vytvoření:
  - z pole: [NSSet setWithArray:],
  - z mutable: [NSSet setWithSet:],
  - operací nad množinou.
- [aSet anyObject] — není random!
- [aSet containsObject:]
  - rychlejší prohledávání (naděje) než u NSArray (metodu má taky)

Set, random object

```
@interface NSSet (randomObj)
-(id) randomObject;
@end

@implementation NSSet(randomObj)
-(id) randomObject
{
    if (self.count == 0)
        return nil;

    return [[self allObjects] objectAtIndex:
arc4random_uniform(self.count)];
}
@end
```

NSPredicate

- Základní operací nad kolekcí je enumerace.
- Definice:
  - řetězcem — očekává se key-value.
  - blokem — nelze pro Core Data dotazy.
- instance NSPredicate se použije:
  - filtrování kolekce — filteredArray…
  - NSFetchRequest
- NSCompoundPredicate

NSPredicate demo

```
NSArray *zakaznici;
NSPredicate *zakMladi = [NSPredicate predicateWithFormat: @"vek < 50"];
NSPredicate *zakaznikPepa = [NSPredicate predicateWithFormat:
    @“jmeno=%@" argumentArray:@[@"Pepa"]];

NSArray *mladi = [zakaznici filteredArrayUsingPredicate: zakMladi];
NSArray *pepa = [zakaznici filteredArrayUsingPredicate: zakaznikPepa];


// Core Data
NSFetchRequest *req = [[NSFetchRequest alloc] initWithEntityName:
@"Zakaznik"];
req.predicate = zakMladi;
```

Enumerace kolekce

- Kolekce jsou heterogenní
- for (id i in kolekce) ;
- for (NSString *jmeno in kolekce) ;
- Metaúroveň objektů:
  - [object class], nelze vždy porovnávat [@“string” class]
  - [object isKindOfClass:]

Enumerace, objc.io

```
// First variant, using `indexesOfObjectsWithOptions:passingTest:`.
NSIndexSet *indexes = [randomArray indexesOfObjectsWithOptions:NSEnumerationConcurrent
                                    passingTest:^BOOL(id obj, NSUInteger idx, BOOL *stop) {
    return testObj(obj);
}];
NSArray *filteredArray = [randomArray objectsAtIndexes:indexes];

// Filtering using predicates (block-based or text)
NSArray *filteredArray2 = [randomArray filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(id obj, NSDictionary
*bindings) {
    return testObj(obj);
}]];

// Block-based enumeration
NSMutableArray *mutableArray = [NSMutableArray array];
[randomArray enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {
    if (testObj(obj)) {
        [mutableArray addObject:obj];
    }
}];
```

Enumerace, objc.io

```
// Classic enumeration
NSMutableArray *mutableArray = [NSMutableArray array];
for (id obj in randomArray) {
    if (testObj(obj)) {
        [mutableArray addObject:obj];
    }
}

// Using NSEnumerator, old school.
NSMutableArray *mutableArray = [NSMutableArray array];
NSEnumerator *enumerator = [randomArray objectEnumerator];
id obj = nil;
while ((obj = [enumerator nextObject]) != nil) {
    if (testObj(obj)) {
        [mutableArray addObject:obj];
    }
}

// Using objectAtIndex: (via subscripting)
NSMutableArray *mutableArray = [NSMutableArray array];
for (NSUInteger idx = 0; idx < randomArray.count; idx++) {
    id obj = randomArray[idx];
    if (testObj(obj)) {
        [mutableArray addObject:obj];
    }
}
```

Enumerace, objc.io

Enumeration Method / Time [ms]	10.000.000 elements	10.000 elements
`indexesOfObjects:,concurrent`	1844.73	2.25
`NSFastEnumeration (for in)`	3223.45	3.21
`indexesOfObjects:`	4221.23	3.36
`enumerateObjectsUsingBlock:`	5459.43	5.43
`objectAtIndex:`	5282.67	5.53
`NSEnumerator`	5566.92	5.75
`filteredArrayUsingPredicate:`	6466.95	6.31

Sorting

- NSSortDescriptor (ascending: (BOOL))
  - definice jménem klíče (key-value protocol)
  - blokem
  - názvem metody (SEL)
- Datový typ “SEL” - název metody
  - @selector(cosi:kdesi:jaksi:)
  - typické pro zadávání metody pro obsluhu události (např. tlačítko UIBarButtonItem)

Sorting Demo

```
NSSortDescriptor *vekSD = [NSSortDescriptor sortDescriptorWithKey:
@"vek" ascending:YES];

NSArray *podleVeku = [zakaznici sortedArrayUsingDescriptors: @[vekSD]];

NSArray *podleVekuBlock = [zakaznici sortedArrayUsingComparator:
                ^NSComparisonResult(Zakaznik *obj1, Zakaznik *obj2) {
                    return [obj1.vek compare: obj2.vek];
                }];

// lze na mutable verzi
NSMutableArray *zakMutable = [NSMutableArray arrayWithArray: zakaznici];
[zakMutable sortUsingDescriptors: @[vekSD]];


// Core Data
// pro NSFetchedResultsController povinne
// musi byt definovano sorting
NSFetchRequest *req;
req.sortDescriptors = @[vekSD];
```

NSCalendar

- Nikdy nezkoušejte vlastní kalendářové výpočty
- [NSDate date], NSTimePeriod
  - klasická absolutní informace o počtu sekund od počátku cal.
  - [NSDate dateWithTimeIntervalSince1970:]
- NSDateFormatter, NSDateComponents
- NSCalendar — pozor, není thread-safe

Calendar, demo

```
NSDate *now = [NSDate date];
NSDate *nowPlus = [now dateByAddingTimeInterval: 60*10];
NSTimeInterval diff = [nowPlus timeIntervalSinceDate: now];

NSCalendar *calendar = [NSCalendar currentCalendar];
NSDateComponents *comps = [calendar components: NSYearCalendarUnit |
NSMonthCalendarUnit fromDate: now];

NSDateComponents *addc = [[NSDateComponents alloc] init];
addc.month = 1;
NSDate *nextMonth = [calendar dateByAddingComponents: addc toDate: now
options:0];


NSDateFormatter *form = [[NSDateFormatter alloc] init];
[form setTimeStyle: NSDateFormatterMediumStyle];
[form setDateStyle: NSDateFormatterShortStyle];
NSString *outp = [form stringFromDate: now];
NSLog(@"Date: %@", outp);
// Date: 14.10.14 13:54:19
```

NSFileManager

- Operace nad soubory. App sandbox.
  - vytvoření, smazání, přesun, …
- Soubor vždy identifikován NSURL, url.path
- Základní přístup k souborům. Dokumenty.
- Dokumenty:
  - Sandbox Documents.
  - Ubiquitous iCloud
  - iCloud Drive

NSFileManager

```
// z app bundle. CoreData DB Model.
NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"TableViewCD"
withExtension:@“momd"];

-(NSURL *) applicationDocumentsDirectory {
    NSFileManager *fm = [NSFileManager defaultManager];
    NSArray *dirs = [fm URLsForDirectory: NSDocumentDirectory
                                inDomains: NSUserDomainMask];

    return [dirs lastObject];
}

// CoreData Sqlite3 store
NSURL *storeURL = [[self applicationDocumentsDirectory]
URLByAppendingPathComponent:@“TableViewCD.sqlite"];
```

Soubory low-level + UIDocument

```
NSFileManager *fm = [NSFileManager defaultManager];
// Obsah souboru
NSURL *soubor = [[self applicationDocumentsDirectory]
       URLByAppendingPathComponent: @"soubor"];

NSData *cont = [fm contentsAtPath: soubor.path];
NSString *contString = [NSString stringWithUTF8String: cont.bytes];
NSScanner *scanner = [NSScanner scannerWithString: contString];
int cislo;
[scanner scanInt: &cislo];

// format dokumentu v doc-based app
UIDocument *doc;
[doc saveToURL: soubor forSaveOperation: UIDocumentSaveForCreating
completionHandler:^(BOOL success) {
        NSLog(@"Soubor zapsan: %d", (int) success);
    }];
```

iCloud Ubiquitous Document

```
NSURL *iCloudURL = [fm URLForUbiquityContainerIdentifier:
@"iCloud.eu.cosi.CloudDoc"];

NSURL *iCloudDoc = [iCloudURL URLByAppendingPathComponent: @"Documents"
isDirectory:YES];

NSURL *tosaveURL = [iCloudDoc URLByAppendingPathComponent: @"nazev"];

CDDocument *doc = [[CDDocument alloc] initWithFileURL: tosaveURL];

[doc saveToURL: tosaveURL forSaveOperation:UIDocumentSaveForCreating
completionHandler:^(BOOL success) {
        if (success) NSLog(@"Document saved");
          else NSLog(@"Error saving the document");
    }];
```

iCloud Drive

- Picker pro soubor. Umožněno pouze import/export. Není home adresář.
- Ukazka pickeru.
- Vybraný soubor pro import se klonuje do tmp sandboxu aplikace.

iCloudDrive, PDF Reader

```
-(void) buttonOpen:(id)sender
{
    UIDocumentPickerViewController *vc =
[[UIDocumentPickerViewController alloc] initWithDocumentTypes:
@[@"public.composite-content", @"com.adobe.pdf"] inMode:
UIDocumentPickerModeImport];

    vc.delegate = self;
    vc.modalPresentationStyle = UIModalPresentationFormSheet;

    [self presentViewController: vc animated: YES completion:^{ }];
}

-(void) documentPicker:(UIDocumentPickerViewController *)controller
  didPickDocumentAtURL:(NSURL *)url
{
    NSLog(@"URL: %@", url);

    self.pdf = CGPDFDocumentCreateWithURL((__bridge CFURLRef) url);
}
```

NSUserDefaults

- Perzistentní dictionary pro aplikaci. Metody pro set/get typu int, bool, string, array.
- Argumenty příkazové řádky.
- Register defaults. Synchronize.
- NSUD mají i extensions (iOS 8) — typicky od hostitelske aplikace.
- App Settings Bundle.

NSUserDefaults

```
NSUserDefaults *UD = [NSUserDefaults standardUserDefaults];
NSURL *facDefs = [[NSBundle mainBundle]
    URLForResource:@"DefaultPreferences" withExtension:@“plist"];

NSDictionary *defaultPrefs = [NSDictionary
    dictionaryWithContentsOfURL: facDefs];

// reg. defaults neni perzistentni!
// volat vzdy pri app:didFinishLaunchingWithOptions:
[UD registerDefaults:defaultPrefs];

BOOL firstRun = [UD boolForKey: @"firstRun"];
[UD setBool: NO forKey: @"firstRun"];
[UD synchronize];

if (firstRun) {
    // Vitaci obrazovka
}

// Today App Extension (iOS 8)
NSUserDefaults *UDExt = [[NSUserDefaults alloc] initWithSuiteName:
@"com.cosi.App"];
```

NSTimer

- Alokace timeru a **registrace** v run loop.
- Deaktivovat timery při přechodu do background módu.

NSTimer demo

```
// objekt je referencovan v NSRunLoop
NSTimer *ts = [NSTimer scheduledTimerWithTimeInterval: 1 target: self
selector: @selector(mtimer:) userInfo: nil repeats: YES];

// zruseni. Pak znovu alokace.
[ts invalidate];

// odlozeny call
[self performSelector: @selector(stalose:) withObject: nil afterDelay:
10];

NSDate *dateFire;
NSTimer *timerAt = [[NSTimer alloc] initWithFireDate: dateFire interval:
0 target: self selector:@selector(mtimer:) userInfo: nil repeats:NO];

// registrace
NSRunLoop *runner = [NSRunLoop currentRunLoop];
[runner addTimer:timerAt forMode: NSDefaultRunLoopMode];
```

NSOperation a queue

- Předchůdce GCD. Dnes je implementováno pomocí GCD.
- Je to kód, ktery se zaradi do fronty. Má vlastní data. Lze znovu-použít.
- Nelze killnout vlákno nebo operaci.
  - Lze jim poslat zprávu.
- CloudKit — operace s CK jsou všechny NSOp.

NSOperation demo

```
@interface MojeOperace : NSOperation
-(void) main;
@end

@implementation MojeOperace

-(void) main
{
    while ([self workToDo]) {
        [self doSomeWork];

        if ([self.isCancelled])
            break;
    }
}

@end

NSOperationQueue *mq = [NSOperationQueue mainQueue];
MojeOperace *operace = [[MojeOperace alloc] init];
// nastav data operaci
[mq addOperation: operace];
```

Komunikace v aplikaci

- Objekty si zasílají zprávy (pokud se znají).
- Delegate. Protocol. Objekt má jednoho delegáta. Implementace a meta protokol. Nilu lze poslat lib. zprávu, ale objektu nelze poslat lib zprávu.
- NSNotificationCenter
- Pozor na typ vlákna, které kód provádí!

Delegate demo

```
// komunikacni protokol
@protocol Komunikace <NSObject>
-(void) objekt: (id) jaObject dokoncilPraci: (NSArray *) vysledek;

@optional
-(void) objektStaleZije:(id)jaObject;
@end

// jedna strana
@interface Pracujici : NSObject
@property (nonatomic, strong) id<Komunikace> delegate;
@end

@implementation Pracujici
-(void) hokna {
    //
    NSArray *vysledky;

    [self.delegate objekt: self dokoncilPraci: vysledky];

    if ([self.delegate respondsToSelector: @selector(objektStaleZije:)])
        [self.delegate objektStaleZije: self];
}
@end
```

Delegate demo

```
@interface Konzumujici : NSObject<Komunikace>
@end

@implementation Konzumujici

-(void) objekt:(id)jaObject dokoncilPraci:(NSArray *)vysledek
{

}

@end
```

NSNotificationCenter

- Singleton. Velmi rozsáhlé sys zprávy.
- Registruje observery událostí. Srovnání s delegates.
- Výhodné pro události, kde je dynamický počet observerů.
- Při poslání zprávy do centra se okamžitě rozesílá do observerů, **tj ve stejném vlákně**.
- Pozor při operacích s UI.

Notifikace demo

```
#define MOJEZPRAVA @"tohle-je-nazev"
-(void) obsluhaNotif: (NSNotification *) notif
{
}
-(void) dealloc
{
    NSNotificationCenter *nc = [NSNotificationCenter defaultCenter];

    [nc removeObserver: self];
}
-(void) viewDidLoad {
    [super viewDidLoad];
    NSNotificationCenter *nc = [NSNotificationCenter defaultCenter];
    // registrace observeru
    [nc addObserver: self selector: @selector(obsluhaNotif:) name:
MOJEZPRAVA object: nil];
    // poslani zpravy
    NSNotification *notif = [NSNotification notificationWithName:
MOJEZPRAVA object: nil];
    [nc postNotification: notif];
    [nc postNotificationName: MOJEZPRAVA object: nil];
}
```