Uložení dat

Martin Hrubý
seminář iOS, FIT VUT v Brně

Způsoby uložení dat

- Dokumenty (UIDocument)
  - typické zapojení NSCoding protokolu.
- Objektová databáze CoreData

NSCoding protokol

- Specifikuje dvě metody:
  - -(id) initWithCoder: (NSCoder *)
  - -(void) encodeWithCoder: (NSCoder *)
- Serializační třídy typu
  - Archiver
  - Unarchiver

Archivace

- Vstupem je objektová paměť libovolné struktury (provázanosti objektů)
- Výstupem je NSData *
- Smysl:
  - výsledný balík dat lze uložit jako soubor nebo jakkoliv někam poslat (síťová komunikace, mezi-aplikační komunikace)
  - Příklad: CKRecord v CloudKit, blob v CD
- NSArchiver, NSKeyedArchiver

```objc
@interface IOZakaznik : NSObject
@property (nonatomic, strong) NSString *jmeno;
@property (nonatomic, strong) NSNumber *cislo;
@end

@implementation IOZakaznik

-(id) initWithCoder:(NSCoder *)aDecoder
{
    self = [self init];

    if (self) {
        self.cislo = [aDecoder decodeObjectForKey: @"cislo"];
        self.jmeno = [aDecoder decodeObjectForKey: @"jmeno"];
    }

    return self;
}

-(void) encodeWithCoder:(NSCoder *)aCoder
{
    [aCoder encodeObject: self.cislo forKey: @"cislo"];
    [aCoder encodeObject: self.jmeno forKey: @"jmeno"];
}
@end
```

Archivace

```
IOZakaznik *zak = [[IOZakaznik alloc] init];

zak.jmeno = @"Pepa Novak";
zak.cislo = @(1234567890);

NSData *seria = [NSKeyedArchiver archivedDataWithRootObject: zak];

IOZakaznik *zakReconstructed = [NSKeyedUnarchiver
unarchiveObjectWithData: seria];

NSLog(@"Rekonstruovany objekt: %@", zakReconstructed);
```

IOAdvancedKeyValue

```
@protocol IOAdvancedKeyValue <NSObject, NSCoding>
-(NSArray *) codingAllKeys;
@end

@interface IOObject : NSObject<IOAdvancedKeyValue>
@end
```

IOObject — implementace

```
-(id) initWithCoder:(NSCoder *)aDecoder
{
    self = [self init];

    if (self) {
        for (NSString *key in [self codingAllKeys]) {
            [self setValue: [aDecoder decodeObjectForKey: key]
                    forKey: key];
        }
    }

    return self;
}


-(void) encodeWithCoder:(NSCoder *)aCoder
{
    for (NSString *key in [self codingAllKeys]) {
        [aCoder encodeObject: [self valueForKey: key]
                      forKey: key];
    }
}
```

Základy UIDocument

- Třída odvozená od UIDocument
- V jedné property drží referenci na datový obsah
- Implementuje dvě základní metody:
  - -(id) contentsForType:
  - -(BOOL) loadFromContents:
- NSFileWrapper, NSFilePresenter, … — později

```
@implementation CDDocument


-(id) contentsForType:(NSString *)typeName
                error:(NSError *__autoreleasing *)outError
{
    NSData *outData = [NSKeyedArchiver archivedDataWithRootObject:
self.text];

    return outData;
}



-(BOOL) loadFromContents:(id)contents
                  ofType:(NSString *)typeName
                   error:(NSError *__autoreleasing *)outError
{
    NSData *inData = (NSData *) contents;
    NSKeyedUnarchiver *unarch = [[NSKeyedUnarchiver alloc]
initForReadingWithData: inData];

    self.text = [unarch decodeObject];

    return YES;
}

@end
```

Uložení / načtení dokumentu

```
CDDocument *doc = [[CDDocument alloc] initWithFileURL: tosaveURL];
// vlozit obsah
doc.text = @“jjjjjj”;

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

[doc openWithCompletionHandler:^(BOOL success) {
        if (success) {
            // poslat zpravu o dokonceni doc-load
        }
    }];
```

Objektová databáze Core Data

- Objektová paměť. Její perzistence.
- Definujeme třídy jako entity ER modelu
  - atributy tříd
  - reference na další entity (objekty), 1:1, 1:N, M:N
- Popis ER modelu je držen v NSManagedModel
- K entitám se generují třídy v Objective-C (NSManagedObject)

Objektová databáze

- Nenajdeme table join.
- Pracujeme s objektovou pamětí.
- Načtením záznamu získáme objekt. Přístupem na atributy (referenční) se automaticky načítají další objekty.

Třídy Core Data

- NSManagedObject
- NSManagedModel
- NSManagedObjectContext
- NSPersistentStoreCoordinator
- Kód vygeneruje XCode. Je třeba určit pouze URL modelu (v bundle) a URL souboru pro uložení dat.

NSManagedObjectContex (MOC)

- Záznamy na disku (Sqlite3) a objekty v paměti.
- Správu objektů zařizuje MOC.
  - Vložení nového objektu, smazání objektu.
  - Dotazy (NSFetchRequest).
- Není thread-safe. Lze mít více MOC.

NSManagedObject

- Odvozené třídy jsou modely entit.
  - Abstraktní entita. Parent entita. Dědění entit.
- Třída a instance. Lze libovolně doprogramovat.
- DB-atributy jsou property (retain). NSNumber.
- DB-relationship je property NSSet, NSArray
  - nelze přistupovat přímo (zápis, mazání)

Entita Student

```
@class Kurz;
@class ISIC;

@interface Student : NSManagedObject

@property (nonatomic, retain) NSString * jmeno;
@property (nonatomic, retain) NSDate * narozen;
@property (nonatomic, retain) NSSet *kurzy;
@property (nonatomic, retain) ISIC *isic;
@end

@interface Student (CoreDataGeneratedAccessors)

- (void)addKurzyObject:(Kurz *)value;
- (void)removeKurzyObject:(Kurz *)value;
- (void)addKurzy:(NSSet *)values;
- (void)removeKurzy:(NSSet *)values;

@end
```

Entita ISIC

```
@class Student;

@interface ISIC : NSManagedObject

@property (nonatomic, retain) NSNumber * idCislo;
@property (nonatomic, retain) Student *student;

@end
```

Entita Kurz

```
@class Student;

@interface Kurz : NSManagedObject

@property (nonatomic, retain) NSString * nazev;
@property (nonatomic, retain) NSSet *zapsani;
@end

@interface Kurz (CoreDataGeneratedAccessors)

- (void)addZapsaniObject:(Student *)value;
- (void)removeZapsaniObject:(Student *)value;
- (void)addZapsani:(NSSet *)values;
- (void)removeZapsani:(NSSet *)values;

@end
```

Vytvoření objektu

```
ISIC *nIsic = [NSEntityDescription insertNewObjectForEntityForName:
@"ISIC" inManagedObjectContext: self.MOC];

Student *nStudent = [NSEntityDescription
insertNewObjectForEntityForName: @"Student" inManagedObjectContext:
self.MOC];

nStudent.jmeno = @"Pepa Novak";
nStudent.isic = nIsic;

NSError *error = nil;
// !!!
[self.MOC save: &error];
```

Vazby mezi objekty

- Apple doporučuje, aby každá vazba byla obousměrná.
  - vazbu pak lze vytvořit/zrušit z libovolného směru.
  - upevňuje to konzistenci databáze.
- 1:1, 1:N, M:N
- Na kolekce s vazbami NEpřistupovat primitivně.

Implementace NSManagedObject

- DB-atributy jsou @dynamic
- @dynamic — překladač NEgeneruje getter / setter
  - key-value přístup proto jde do valueForKey:, setValue:forKey:
  - tyto metody obsluhuje NSManagedObject (a dodá do proměnné hodnotu z DB)
  - odložený load objektu

Property znova

- @property NSString *jmeno;
- objekt.jmeno — překladač umožní psát.
  - Pokud existuje getter, volá se. @synthesize
  - Jinak se volá valueForKey: (může dojít až k výjimce). Lze volat i přímo.
- @dynamic — překladač negeneruje getter/sett.
  - Jde se rovnou na valueForKey:
- [objekt jmeno].

Ukázka @dynamic property

```
@dynamic dynHodnota;

-(id) valueForKey:(NSString *)key
{
    if ([key isEqualToString: @"dynHodnota"])
        return _ulozenaDynHodnota;

    return [super valueForKey: key];
}

-(void) setValue:(id)value forKey:(NSString *)key
{
    if ([key isEqualToString: @"dynHodnota"]) {
        _ulozenaDynHodnota = [NSString stringWithFormat: @"je: %@",
value];
    } else
        [super setValue: value forKeyPath: key];
}
```

Primitivní přístup na atribut

```
-(NSString *) jmeno
{
    [self willAccessValueForKey: @"jmeno"];
    NSString *jm = [self primitiveValueForKey: @"jmeno"];
    [self didAccessValueForKey: @"jmeno"];

    return jm;
}

-(void) setJmeno:(NSString *)jmeno
{
    [self willChangeValueForKey: @"jmeno"];
    [self setPrimitiveValue: jmeno forKey: @"jmeno"];
    [self didChangeValueForKey: @"jmeno"];
}
```

Provedení dotazu

- NSFetchRequest
  - predicate
  - entity
  - sortDescriptor
- NSManagedObjectContext

Dotaz

```
NSFetchRequest *request = [NSFetchRequest
fetchRequestWithEntityName:@"Student"];

NSPredicate *nemaISIC = [NSPredicate predicateWithFormat: @"isic=nil"];
NSSortDescriptor *sd = [NSSortDescriptor sortDescriptorWithKey: @"jmeno
ascending:YES];

request.predicate = nemaISIC;
request.sortDescriptors = @[sd];

NSError *error = nil;
NSArray *results = [self.MOC executeFetchRequest: request
                                            error: &error];
```

Mazání objektu

- Zpráva deleteObject: (id) -> MOC
- V modelu popis metody kaskádového mazání napojených objektů (null, kaskáda, …)
- Je třeba ovšem brát VELMI vážně ARC (obecně delete objektu):
  - Pokud je objekt uložen v NE-CoreData kolekci (NSArray), pak přežije.

Verze modelu

- Pokud modifikujete CD-Model, pak CD odmítne otevřít původní databázi.
  - Během vývoje aplikace je praktičtější pracovní DB smazat.
  - Jinak je třeba verzovat.
- Konverze:
  - Existuje nějaká podpora pro automatické konverze DB.
  - Manuální (otevřít DB se starým modelem a kopírovat objekty do nového modelu).

Pokročilé CD

- Zvládnout verzování DB.
- UndoManagement.
- Vlastní PersistentStore
  - BP s Dropbox DB API
  - CloudKit — asynchronní přístup.

Spojení UITableView a CD

- DataSource pro UITableView. Data z:
  - NSArray — manuální.
  - NSFetchResultsController (FRC) — automatizované.
- FRC je observer svého obsahu a MOC.
  - vložení nového objektu vyhovující FR, smazání objektu.
  - modifikace objektu.

Použití FRC jako dataSource

```
request.sortDescriptors = @[sd];

FRC = [[NSFetchedResultsController alloc] initWithFetchRequest: request
managedObjectContext: self.MOC sectionNameKeyPath: nil cacheName:nil];

FRC.delegate = self;
self.tableView.dataSource = self;
[FRC performFetch: &error];
```

```
-(void) controllerDidChangeContent:(NSFetchedResultsController
*)controller
{
    [self.tableView reloadData];
}
-(NSInteger) numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}
-(NSInteger) tableView:(UITableView *)tableView numberOfRowsInSection:
(NSInteger)section
{
    return [FRC.fetchedObjects count];
}

-(UITableViewCell *) tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    Student *st = [FRC objectAtIndexPath: indexPath];
    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier: @"cell"];

    cell.textLabel.text = st.jmeno;

    return cell;
}
```

Koncepční dataSource

- Kód pro různé UITableViewController se stále opakuje.
- Vytvořte si wrapper nad FRC implementující UITableViewDataSource protokol.