Blocks

Martin Banas
iOS Developer
martin.banas@inmite.eu

return type name arguments

```
int myFunction (int a) {
    // code goes here
    return a;
}
```

return type **name** **arguments** **arguments**

```
int (^myFirstBlock) (int) = ^(int a) {
    // code goes here
    return a;
};
```

“ language - level feature added to C, Objective-C and C++, which allow you to create distinct segments of code that can be passed around to methods or functions as if they were value ”

block variable **block literal**

```
int (^myFirstBlock) (int) = ^(int a) {
    // code goes here
    return a;
};
```

```
^(int a) {
    // code goes here
    return a;
};
```

```objc
- (void)viewDidLoad {
    [super viewDidLoad];

    [self myFunctionWithBlock:^(int a) {
        NSLog(@"%i",a);
    }];
}

- (void)myFunctionWithBlock:(void (^)(int a))block {
    if (block) {
        block(3);
    }
}
```

Why use blocks?

- Blocks make your code easier to read and reuse
- Blocks allow you to perform advanced tasks easier
- Things like concurrency and callbacks become much easier
- …because you have to

Where use blocks?

- Callbacks of any kind are likely candidates for using blocks
- Anywhere you have a delegate, blocks are a good candidate for replacement
- Completion handlers or failure handlers
- Enumarations, NSOperations, etc..

Creating blocks

```
// As a local variable
int (^blockName)(int) = ^int (int a) {…};

// As a property
@property (nonatomic, copy) void (^blockName)(int a);

// As a method to parameter
[self myFunctionWithBlock:^(int a) {…}];

// As an argument to a method call
- (void)myFunctionWithBlock:(void (^)(int a))block {…}

// As a typedef
typedef void (^MyBlock)(int a);
```

Mutable variables

```
NSArray *array = @[];

__block BOOL anyResults = NO;
BOOL anyResultsFail = NO;

[array enumerateObjectsUsingBlock:^(id obj, NSUInteger i, BOOL
*stop) {
   if (obj == objectWeAreLookingFor) {
      anyResults = YES;
      anyResultsFail = YES;
      *stop = YES;
    }

}];
```

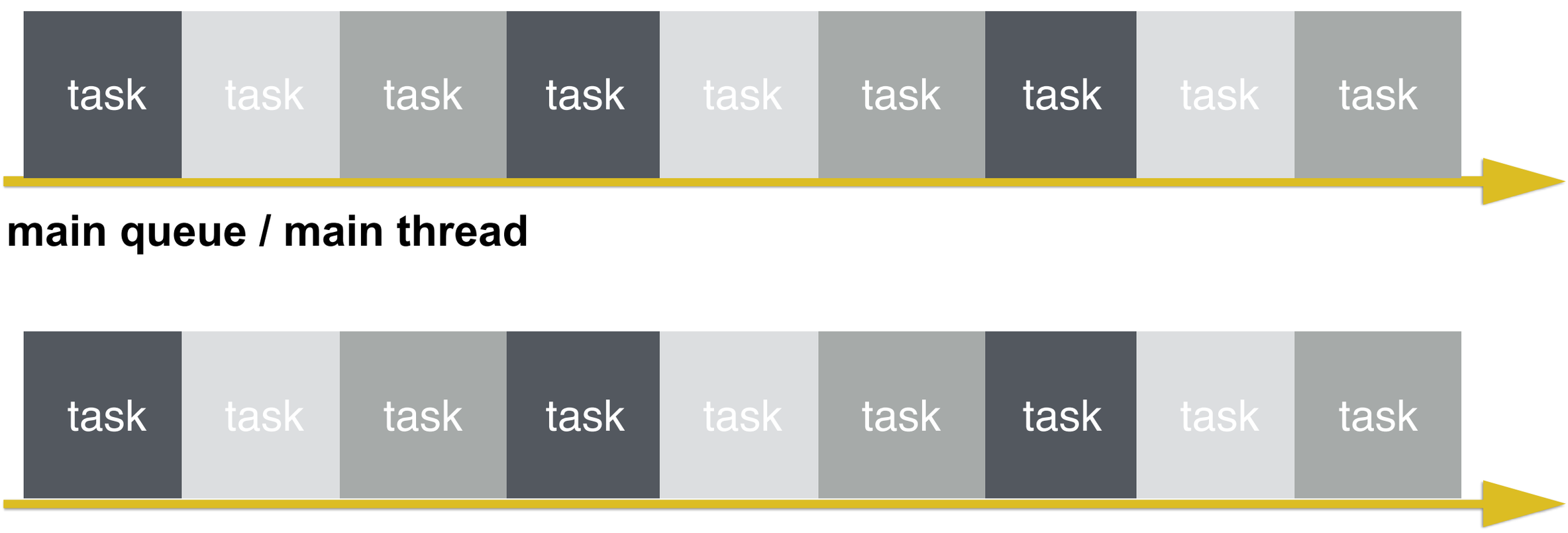
Memory management

- With ARC it just works
- Use copy instead of retain (strong)
- Adding block pointers to a collection, you need to copy them first
- Retain cycles are dangerous

Concurrent programming

Martin Banas
iOS Developer
martin.banas@inmite.eu

Working with queues



Options in iOS

- Classes with built-in concurrency

  many classes have blocks

- Grand Central Dispatch (GCD)

  simple, C function calls

- NSOperation & NSOperationQueue

  built on top of GCD, better control

- Manual multithreading (threads)

Threading

- **Thread** is used to refer to a separate path of execution for code
- **Process** is used to refer to a running executable, which can encompass multiple threads
- **Task** is used to refer to the abstract concept of work that needs to be performed

Options in iOS

- Classes with built-in concurrency

  many classes have blocks

- Grand Central Dispatch (GCD)

  simple, C function calls

- NSOperation & NSOperationQueue

  built on GCD, better control

- Manual multithreading (threads)

Grand Central Dispatch

(GCD)

Benefits of GCD

- Straightforward and simple programming interface
- Automatic thread pool management
- More memory efficient
- Tasks cannot deadlock the queue
- More efficient alternative to locks and synchronization primitives

Using GCD

1. Create a new queue

    give it a name (reverse DNS)

Using GCD

1. Create a new queue

   give it a name (reverse DNS)

2. Add tasks (blocks) to the queue

Using GCD

1. Create a new queue

   give it a name (reverse DNS)

2. Add tasks (blocks) to the queue

3. There is no step 3

dispatch_queue_create

```
dispatch_queue_t backgroundQueue = dispatch_queue_create("backgroundQ", NULL);
```

dispatch_queue_create

```
dispatch_queue_t backgroundQueue = dispatch_queue_create("backgroundQ", NULL);
```

dispatch_async

```
dispatch_async(backgroundQueue, ^{ // code goes here });
```

Queues

1. Main queue

   main thread, FIFO order, don't block UI!

2. Serial queues

   may switch to a different thread between tasks

   always wait for a task to finish before going to the next one - FIFO

3. Concurrent queues

   submit tasks to any available thread or even make new threads

   FIFO order, but order of completion is not guaranteed.

Creating or getting queues

```
// Create a serial or concurrent queue
dispatch_queue_create

// Get the one and only main queue
dispatch_get_main_queue

// Get one of the global concurrent queues
dispatch_get_global_queue
```

Adding tasks to the Queues

```
// Asynchronous
dispatch_async
dispatch_after
dispatch_apply
```

```
// Synchronous
dispatch_once
dispatch_sync
```

Operation queues

NSOperation & NSOperationQueue

NSOperation

- Built on top of GCD
- Much better control
- Object-oriented approach
- Thread-safe, state, priority, dependencies, cancellation

State

- **isReady** returns YES to if the initialization steps are finished
- **isExecuting** returns YES if the operation is currently working on its task
- **isFinished** returns YES if the operation's task finished or if the operation was cancelled

Priority

- NSOperationQueuePriorityVeryHigh
- NSOperationQueuePriorityHigh
- NSOperationQueuePriorityNormal
- NSOperationQueuePriorityLow
- NSOperationQueuePriorityVeryLow

GCD vs Operations

Core Data

Martin Banas
iOS Developer
martin.banas@inmite.eu

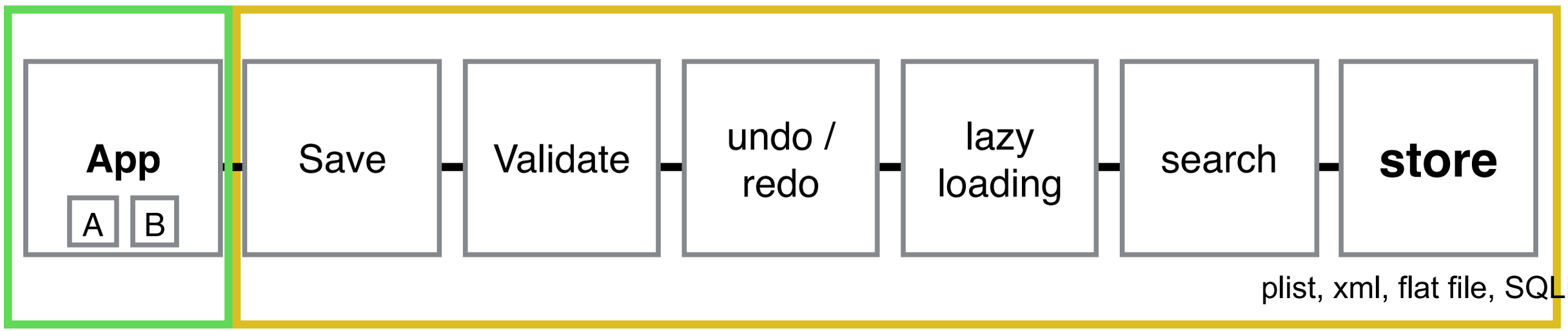
“ … Core Data is a schema-driven object graph management and persistence framework. ”

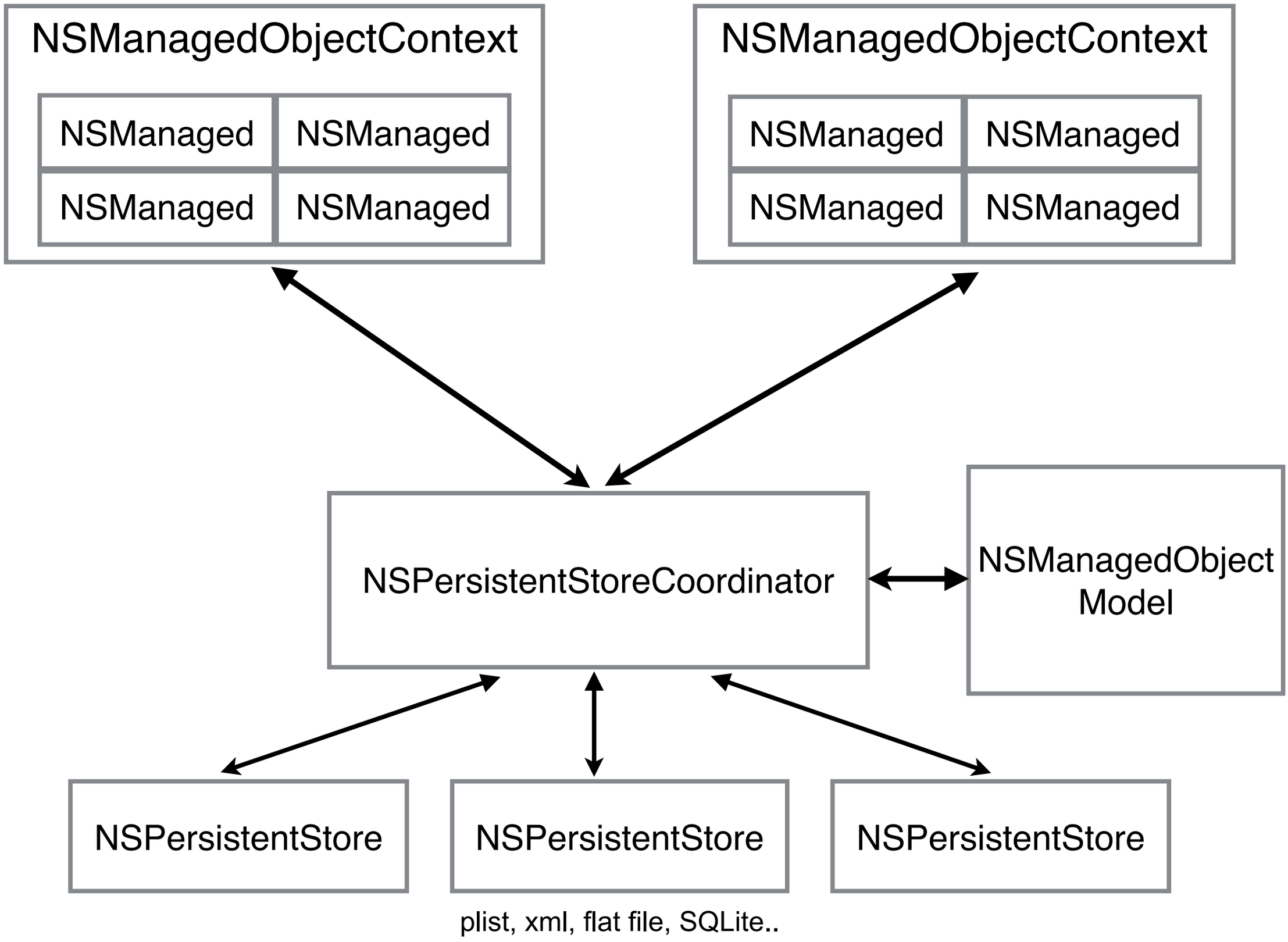
is Core Data..

… a database? no, but..

… ORM? no, but..

… like [insert what you want] ? no, but..


App
A
B
Save
Validate
undo /
redo
lazy
loading
search
store
plist, xml, flat file, SQLite..
Your App
Core Data


NSManagedObjectContext
NSManaged
NSManaged
NSManaged
NSManaged
NSManagedObjectContext
NSManaged
NSManaged
NSManaged
NSManaged
NSPersistentStoreCoordinator
NSManagedObject Model
NSPersistentStore
NSPersistentStore
NSPersistentStore
plist, xml, flat file, SQLite..

Modeling in Core Data

- Introduction
- Creating entities
- Creating and configuring attributes
- Modeling relationships

Saving in Core Data

- Creating managed objects
- Understanding the managed object context
- Saving the managed object context

Fetching in Core Data

- Creating and using a fetch request
- Ordering with sort descriptors
- Using predicates

Next steps

- iOS App Programming Guide
- iOS Human Interface Guidelines
- iOS Developer Library
- Development Videos (WWDC)
- Stanford's iTunes U App Development Course
- NSHipster.com
- Objc.io