

iPhone Application Programming

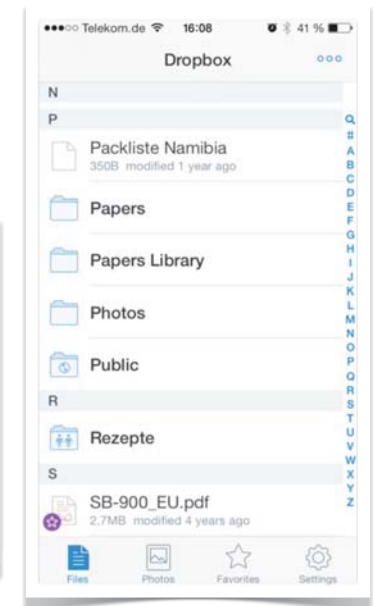
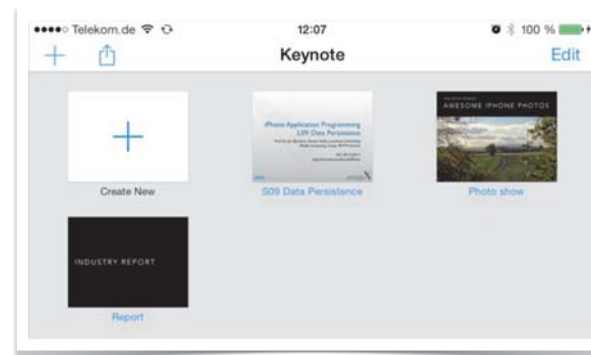
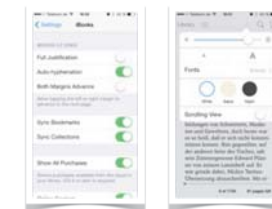
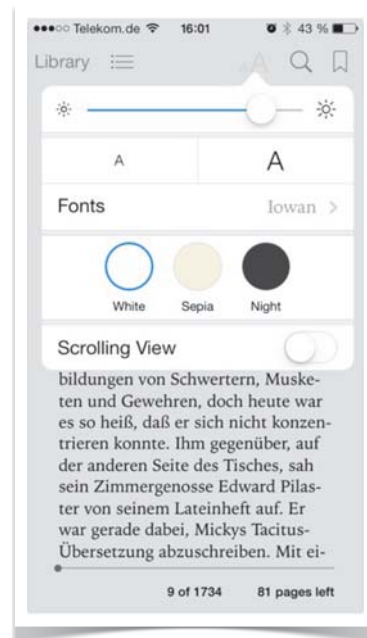
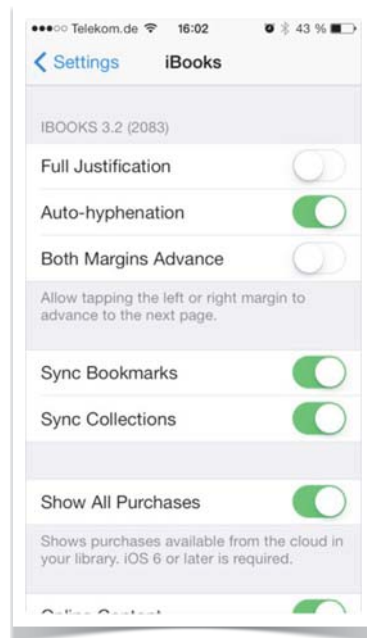
Lecture 09: Data Persistence

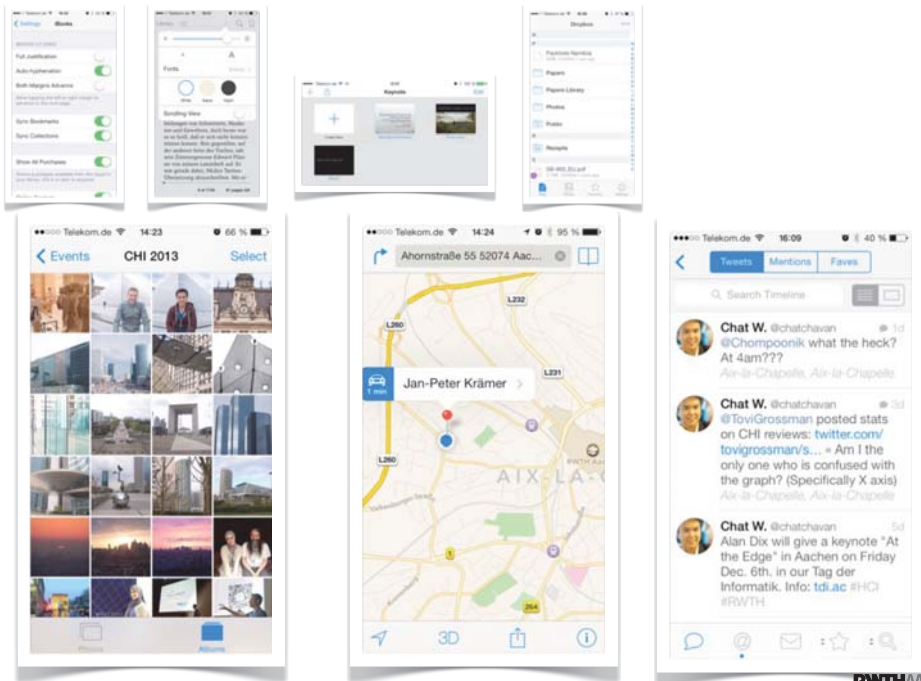
Jan-Peter Krämer
 Media Computing Group
 RWTH Aachen University
 Winter Semester 2013/2014
<http://hci.rwth-aachen.de/iphone>



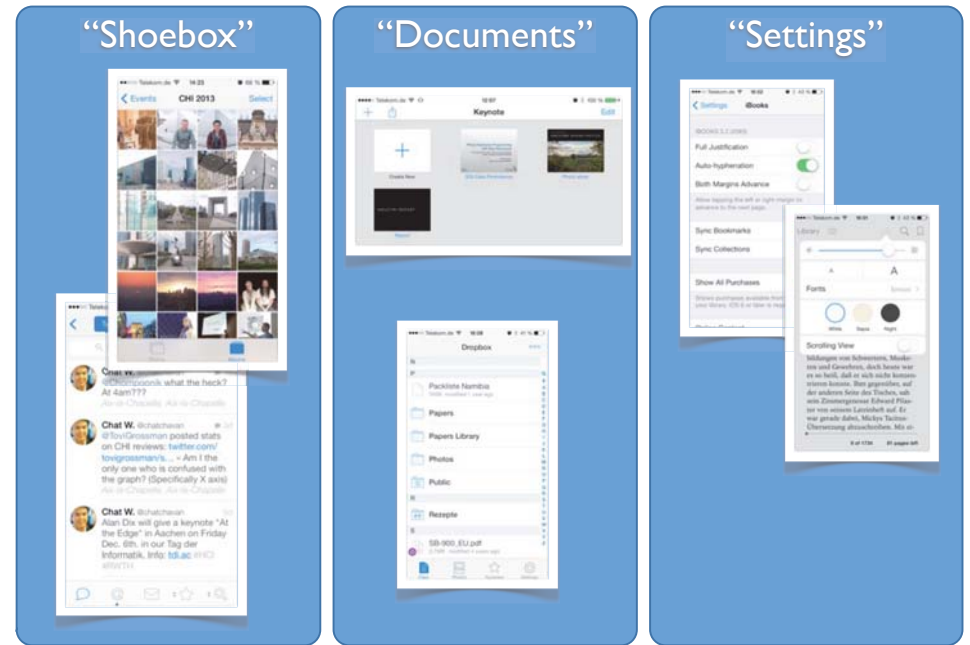
What data do we have on devices?

How do we present data to the user?





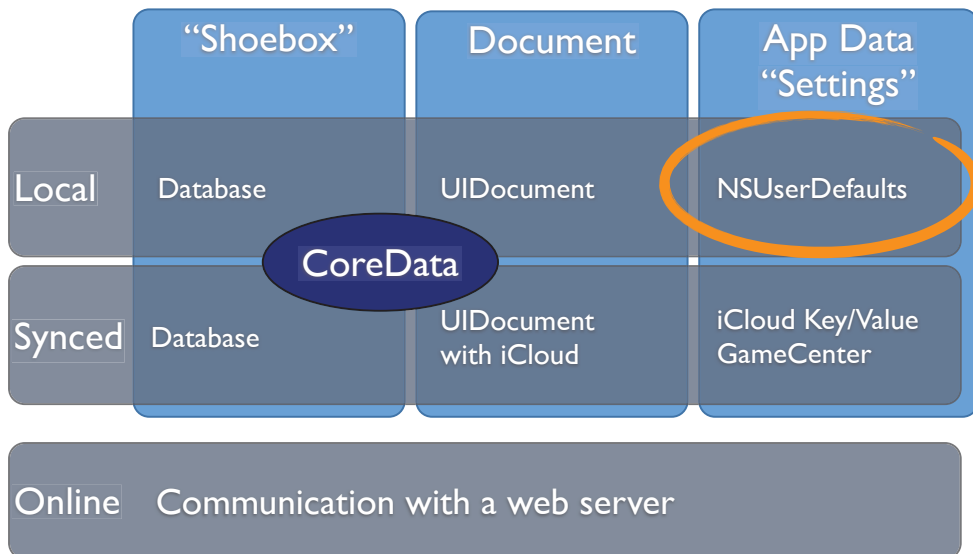
5 iPhone Application Programming • Prof. Jan Borchers



6 iPhone Application Programming • Prof. Jan Borchers



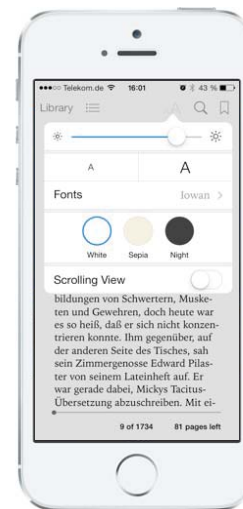
Data Handling Overview



7 iPhone Application Programming • Prof. Jan Borchers



Preferences and Settings

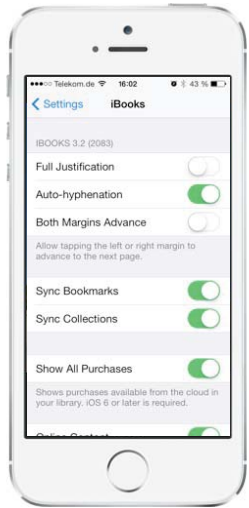


- NSUserDefaults
 - Singleton
 - Key/value pairs
 - Provides standard values (factory defaults)
- Settings Bundle
 - Describes preferences managed in settings
 - Same keys as in NSUserDefaults

8 iPhone Application Programming • Prof. Jan Borchers

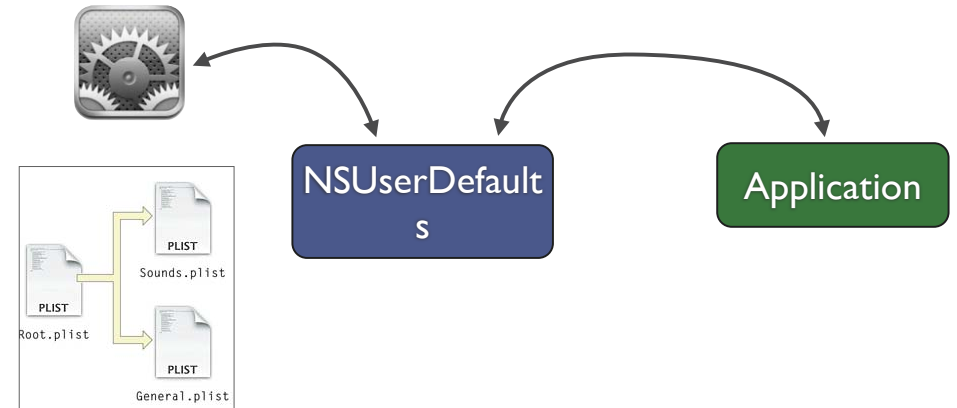


Preferences and Settings



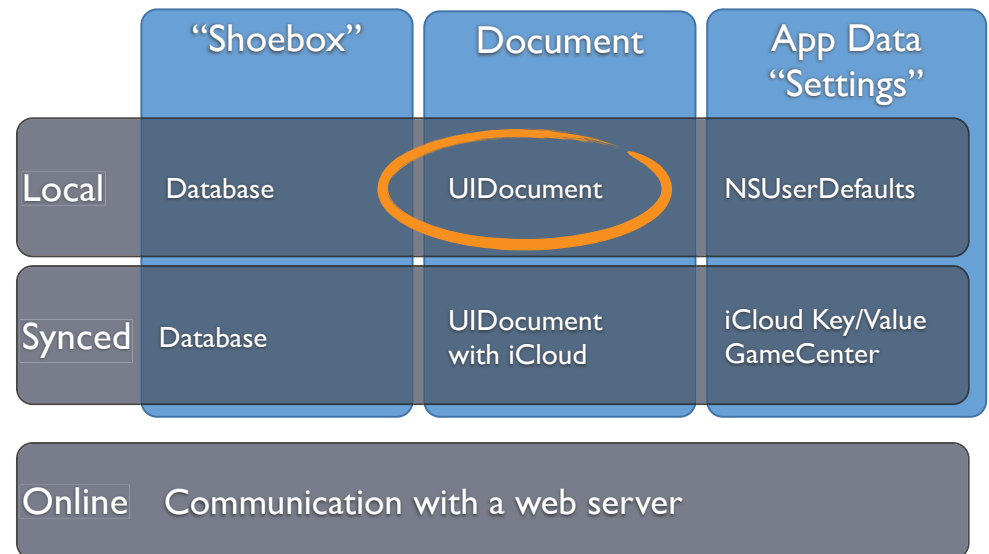
- NSUserDefaults
 - Singleton
 - Key/value pairs
 - Provides standard values (factory defaults)
- Settings Bundle
 - Describes preferences managed in settings
 - Same keys as in NSUserDefaults

Preferences and Settings



Demo

Data Handling Overview



Where to put files?

```
- (NSURL *) localDocumentsDirectoryURL
{
    static NSURL *localDocumentsDirectoryURL = nil;
    if (localDocumentsDirectoryURL == nil)
    {
        // use system function
        NSString *documentsDirectoryPath =
            [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                                NSUserDomainMask,
                                                YES) objectAtIndex:0];

        localDocumentsDirectoryURL
            = [NSURL fileURLWithPath:documentsDirectoryPath];
    }
    return localDocumentsDirectoryURL;
}
```

UIDocument

- High level API for file access
- Always represents a user document
- Can handle file packages
- Autosaving
- Undo
- Handles remote changes from iCloud
- UIManagedDocument for CoreData

Creating a new document

```
// creating a new document
self.document = [[UIDocument alloc] initWithFileURL:fileURL];
// document created in memory
[self.document saveToFileURL:self.document.fileURL
    forSaveOperation:UIDocumentSaveForCreating
    completionHandler:^(BOOL success)
    {
        // file created on disk
        if (success)
            [self displayFileContents];
    }];
```

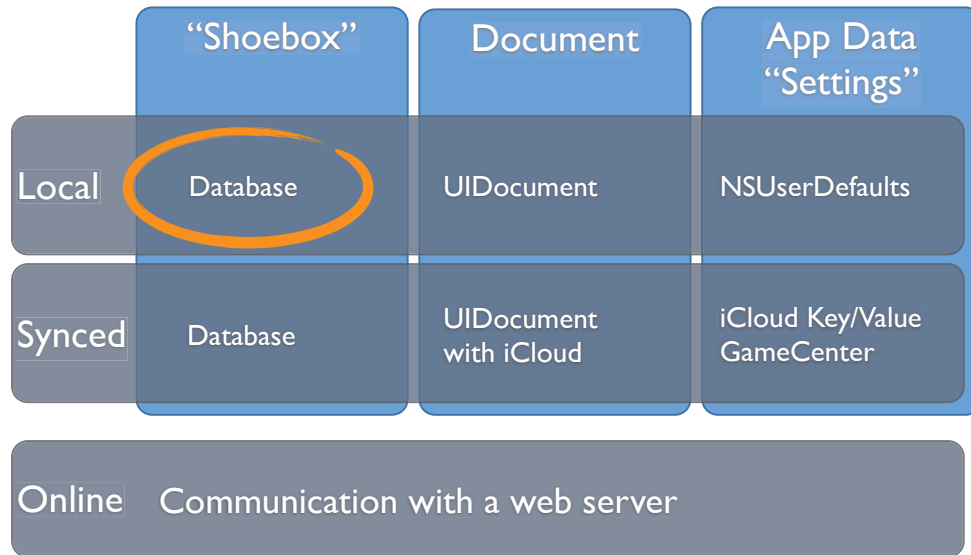
```
// UIDocument
- (id) initWithFileURL:(NSURL *)url {
    self = [super initWithFileURL:url];
    if (self)
        self.textcontents = @"Hello world!";
    return self;
}
```

Saving a UIDocument

- gets saved when file is closed or
- autosaved when changes are pending
- NSUndoManager
- or call updateChangeCount:

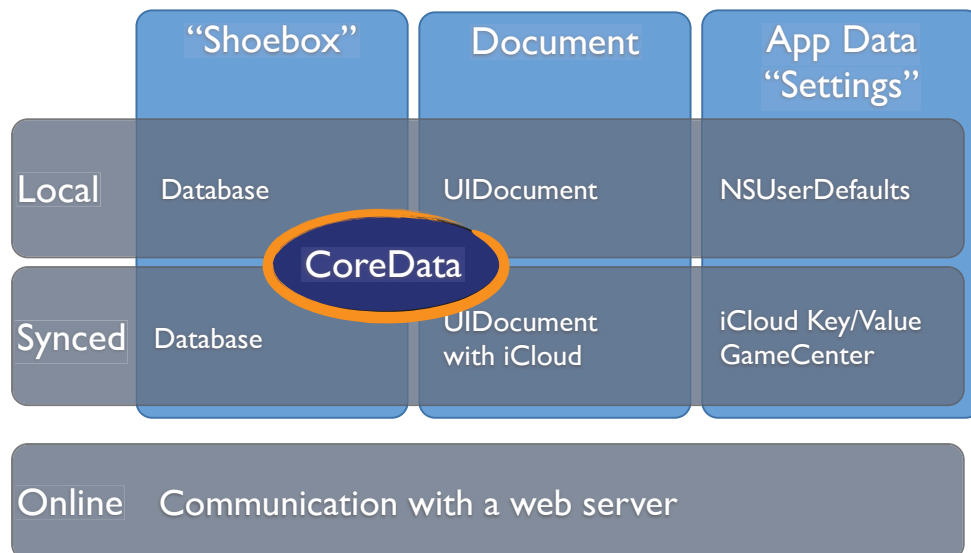
```
// Return NSData representation for saving
- (id) contentsForType:(NSString *)typeName
    error:(NSError **)outError
{
    NSData *data = [self.contents dataUsingEncoding:
        NSUTF8StringEncoding];
    return data;
}
```

Data Handling Overview



SQLite

Data Handling Overview

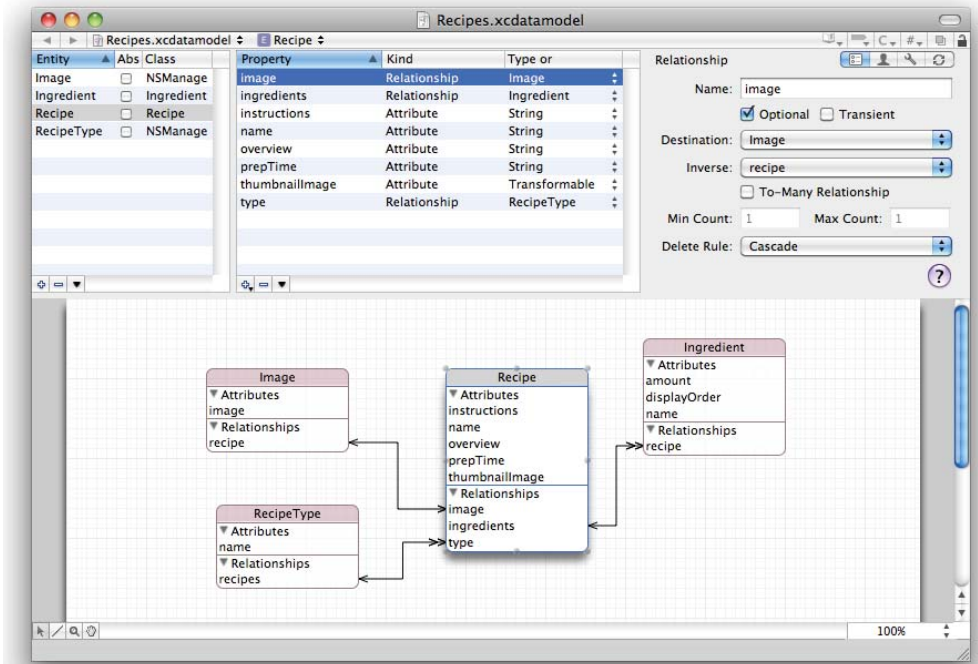
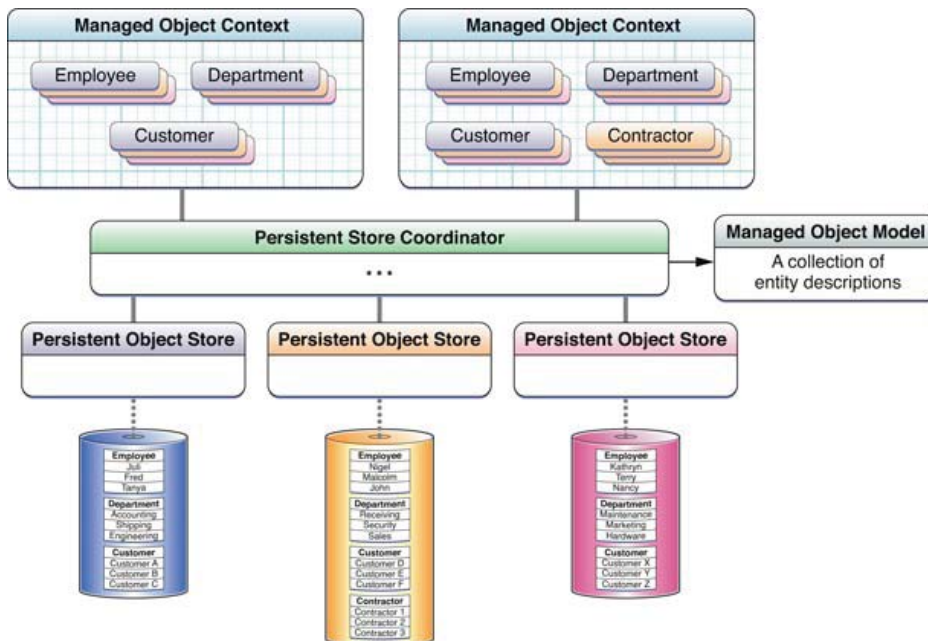
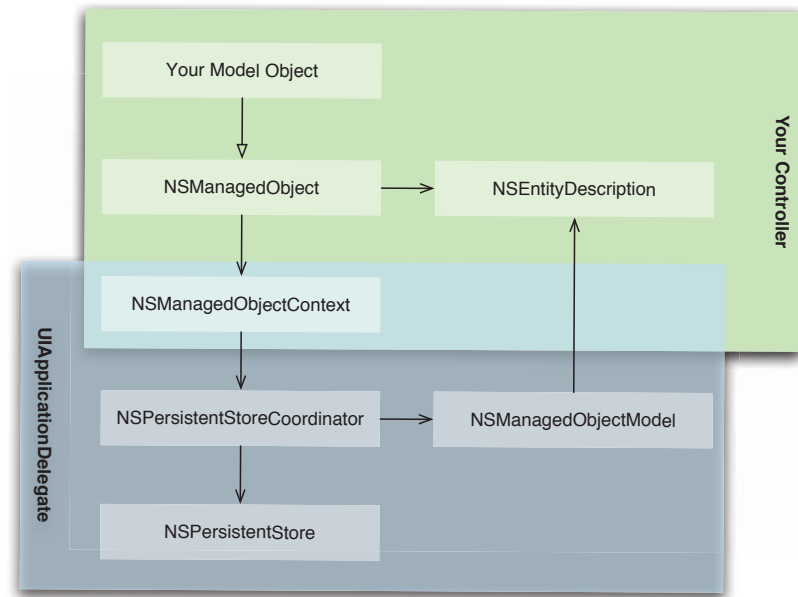


Core Data

Core Data

- High level relational data storage
- Automatic persistence
- High performance access
- Get undo/redo for free
- Available data stores
 - Memory
 - Binary file
 - SQLite

Core Data Stack



Recipes.xcdatamodel

Recipe

Property	Kind	Type or
image	Relationship	Image
ingredients	Relationship	Ingredient
instructions	Attribute	String
name	Attribute	String
overview	Attribute	String
prepTime	Attribute	String
thumbnailImage	Attribute	Transformable
type	Relationship	RecipeType

Relationship

Name: image

Optional Transient

Destination: Image

Inverse: recipe

To-Many Relationship

Min Count: 1 Max Count: 1

Delete Rule: Cascade

```
graph TD; Image --> Recipe; RecipeType --> Recipe; Ingredient --> Recipe;
```

Recipes.xcdatamodel

Recipe

Property	Kind	Type or
image	Relationship	Image
ingredients	Relationship	Ingredient
instructions	Attribute	String
name	Attribute	String
overview	Attribute	String
prepTime	Attribute	String
thumbnailImage	Attribute	Transformable
type	Relationship	RecipeType

Relationship

Name: image

Optional Transient

Destination: Image

Inverse: recipe

To-Many Relationship

Min Count: 1 Max Count: 1

Delete Rule: Cascade

```
graph TD; Image --> Recipe; RecipeType --> Recipe; Ingredient --> Recipe;
```

Recipes.xcdatamodel

Recipe

Property	Kind	Type or
image	Relationship	Image
ingredients	Relationship	Ingredient
instructions	Attribute	String
name	Attribute	String
overview	Attribute	String
prepTime	Attribute	String
thumbnailImage	Attribute	Transformable
type	Relationship	RecipeType

Relationship

Name: image

Optional Transient

Destination: Image

Inverse: recipe

To-Many Relationship

Min Count: 1 Max Count: 1

Delete Rule: Cascade

```
graph TD; Image --> Recipe; RecipeType --> Recipe; Ingredient --> Recipe;
```

Recipes.xcdatamodel

Recipe

Property	Kind	Type or
image	Relationship	Image
ingredients	Relationship	Ingredient
instructions	Attribute	String
name	Attribute	String
overview	Attribute	String
prepTime	Attribute	String
thumbnailImage	Attribute	Transformable
type	Relationship	RecipeType

Relationship

Name: image

Optional Transient

Destination: Image

Inverse: recipe

To-Many Relationship

Min Count: 1 Max Count: 1

Delete Rule: Cascade

```
graph TD; Image --> Recipe; RecipeType --> Recipe; Ingredient --> Recipe;
```

Custom Model Class

```
// Recipe.h
@interface Recipe : NSManagedObject {}
@property(retain) NSString *title;

// Recipe.m
#import "Recipe.h"

@implementation Recipe
@dynamic title;
@end
```

Creating / Deleting Managed

- Create (Insert)
 - NSManagedObjectContext
 - Entity name

```
[NSEntityDescription insertNewObjectForEntityForName:
[entity name] inManagedObjectContext:context]
```
- Delete
 - NSManagedObjectContext
 - NSManagedObject

```
[context deleteObject:managedObject];
```
- Save context to make change persistent

Retrieving Managed Objects

- NSFetchRequest
 - Entity: NSEntityDescription
 - Sorting criteria: NSSortDescriptors
 - Search criteria: NSPredicate
- NSFetchedResultsController
 - Tailored to provide data for UITableViews
 - Requires: NSFetchRequest, NSManagedObjectContext
 - Change tracking via delegate

Fetch Data

```
//Set up the fetch request
NSFetchRequest *request = [[NSFetchRequest fetchRequestWithEntityName:@"Recipe"];

//Configure the fetch request
NSSortDescriptor *sortDescriptor = [NSSortDescriptor
sortDescriptorWithKey:@"creationDate" ascending:NO];
[request setSortDescriptors: @[ sortDescriptor ]];

// Execute the request
NSError *error = nil;
NSArray *fetchResults;
fetchResults = [managedObjectContext executeFetchRequest:request error:&error];
if (error) {
    // Handle the error.
}
```


Undo Manager

```
// attach undo manager
undoManager = [[NSUndoManager alloc] init];
[self.managedObjectContext setUndoManager:undoManager];

//perform undo
[self.managedObjectContext undo];
```

Demo

Performance

- Use batch fetches
- Just fetch needed values
- Use predicates to filter results
- Use SQLite for calculations
- Use SQL Debugging

```
[request setPropertiesToFetch:[ @"name" ]];
[request setPropertiesToGroupBy:[ @"major" ]];
```

Performance

- Use batch fetches
- Just fetch needed values
- Use predicates to filter results
- Use SQLite for calculations
- Use SQL Debugging

```
request.predicate = [NSPredicate predicateWithFormat:
    @"age > %@ AND name == %i", name, age];
```

Numeric comparison
is cheap

Text comparison
is expensive

Performance

- Use batch fetches
- Just fetch needed values
- Use predicates to filter results
- Use SQLite for calculations
- Use SQL Debugging

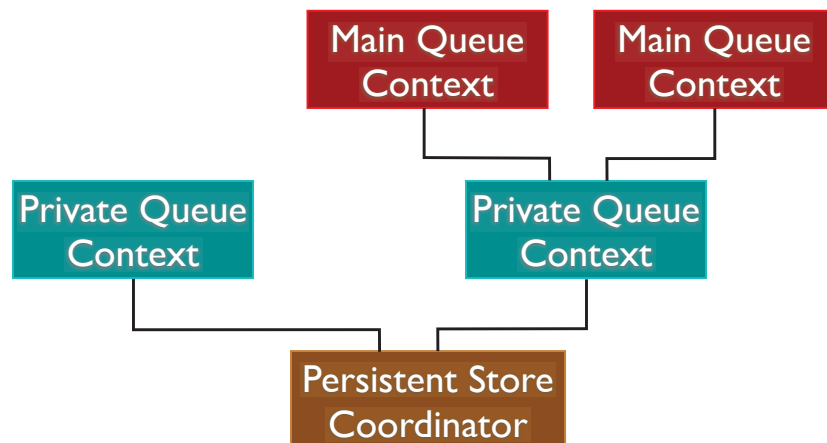
```
NSEXPRESSIONDESCRIPTION *ed =  
[[NSEXPRESSIONDESCRIPTION alloc]  
init];  
  
ed.name = @"minimum";  
ed.expression = [NSEXPRESSION  
expressionForFunction:@"min:"  
arguments:@[ [NSEXPRESSION  
expressionForKeyPath:@"grade"] ]];  
  
[request  
setPropertyToFetch:@[ ed ]];
```

Performance

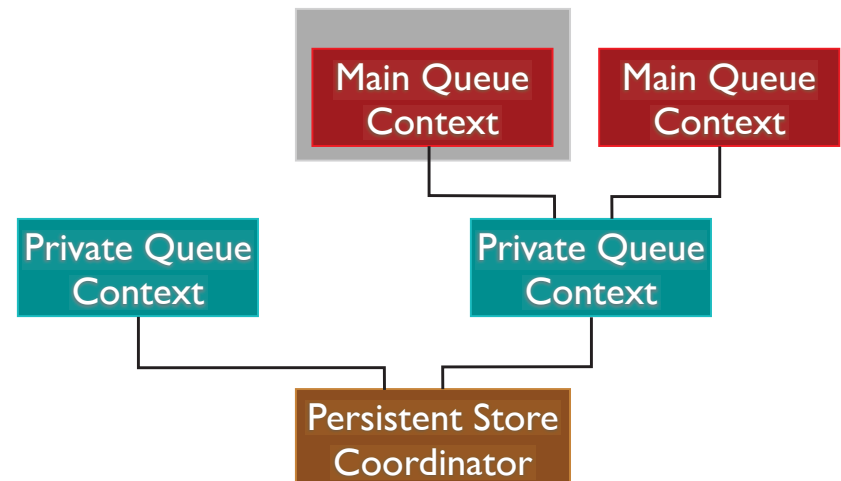
- Use batch fetches
- Just fetch needed values
- Use predicates to filter results
- Use SQLite for calculations
- Use SQL Debugging

```
-com.apple.CoreData.SQLDebug 1
```

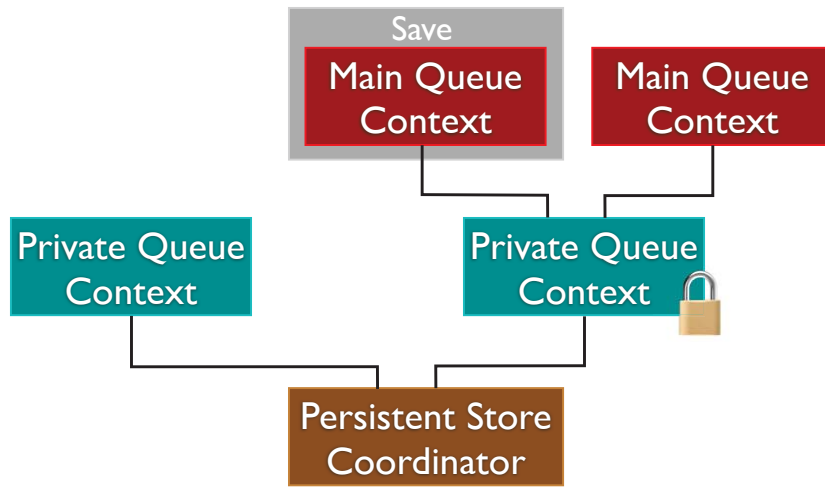
Concurrency



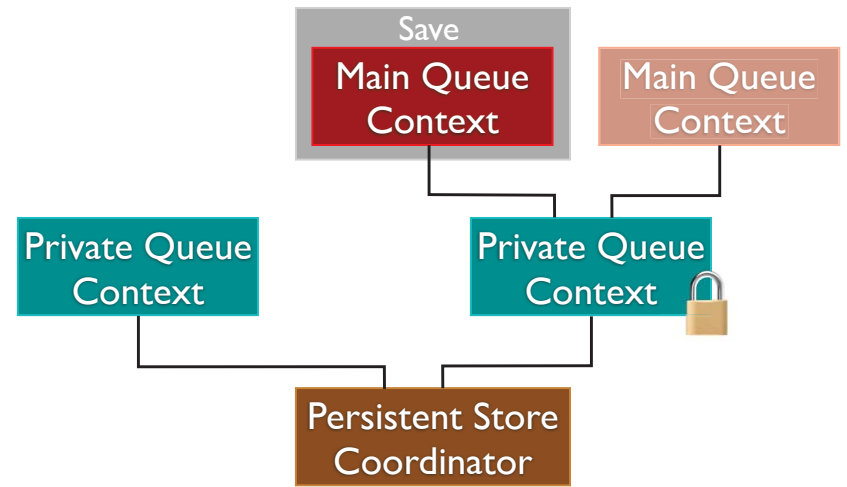
Concurrency



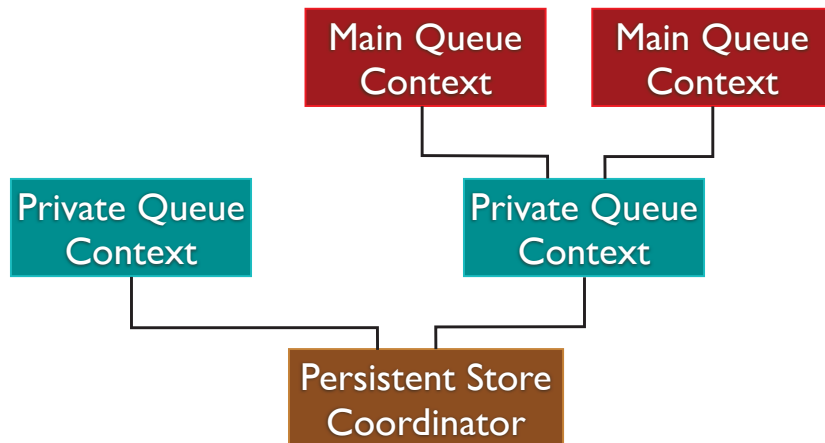
Concurrency



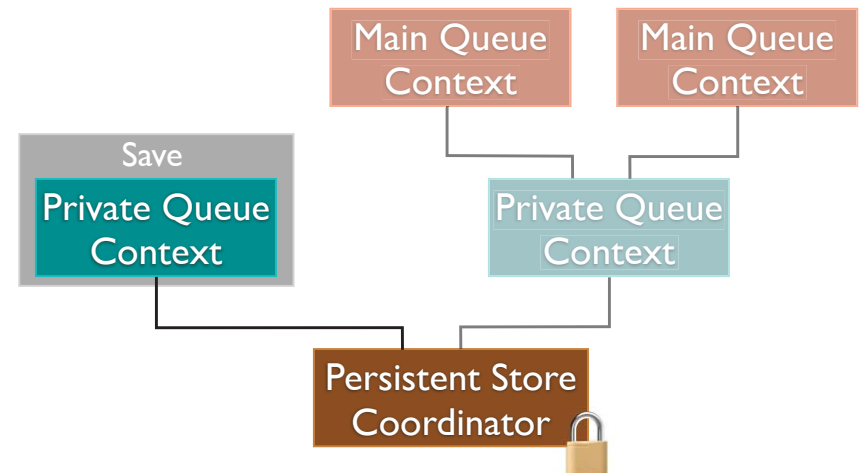
Concurrency



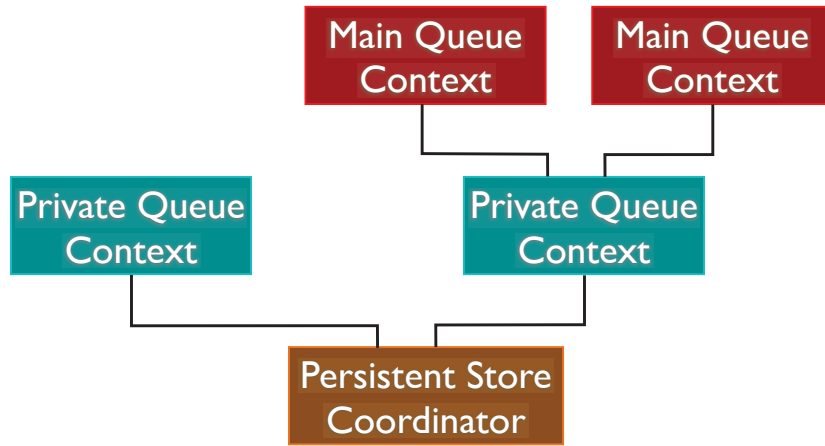
Concurrency



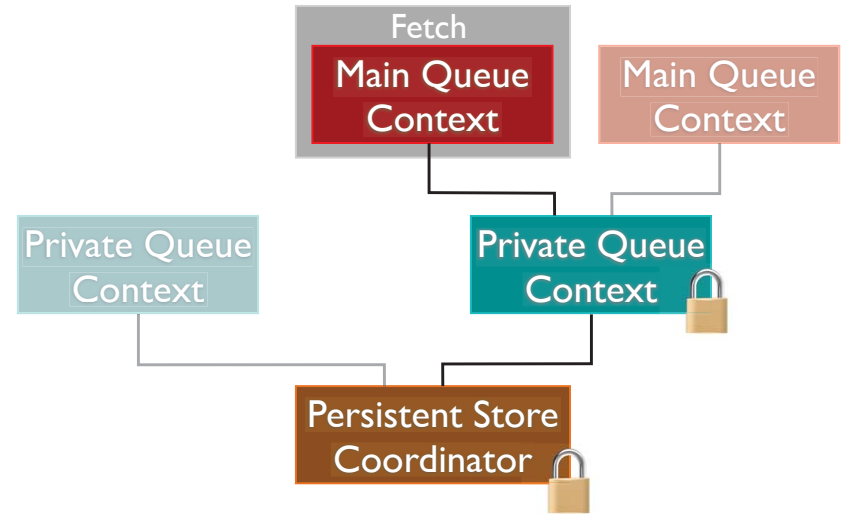
Concurrency



Concurrency



Concurrency



Data Handling Overview

	“Shoebox”	Document	App Data “Settings”
Local	Database	UIDocument	NSUserDefaults
Synced	Database	UIDocument with iCloud	iCloud Key/Value GameCenter
Online	Communication with a web server		

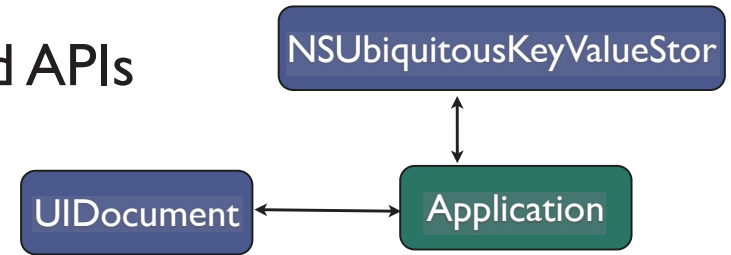
iCloud

iCloud Basics



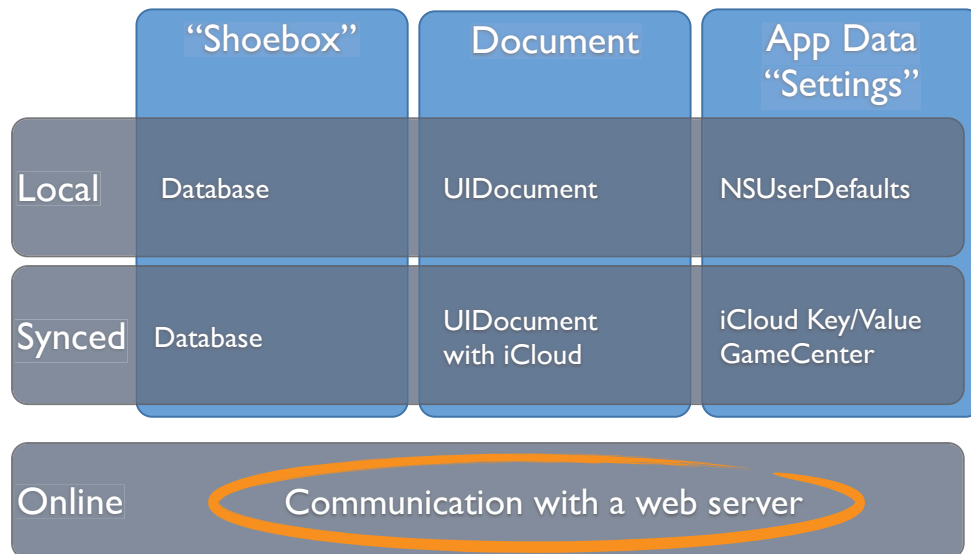
- Changes on one device are automatically pushed to other devices
- User can pick up any device and will always have access to her data
- Sync is per app and per user
- Free subscription is 5GB shared among all apps

iCloud APIs



- Settings: use NSUbiquitousKeyValueStore (similarly to UserDefaults)
 - 1 MB limitation
 - Syncs within minutes
 - Last value always wins
- Documents: UIDocument on a URL in a special folder
 - No limit
 - Syncs within seconds
 - Can merge conflicting files

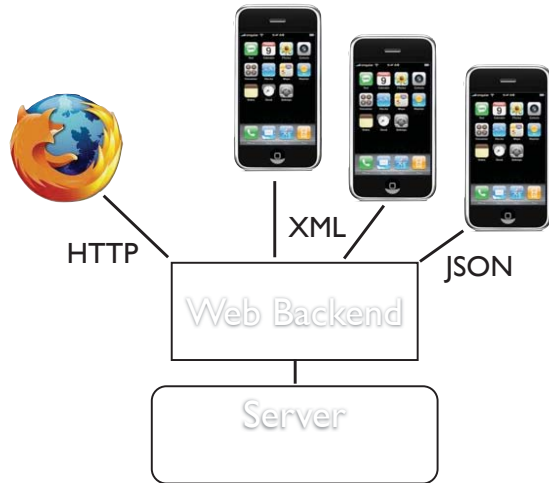
Data Handling Overview



Remote (Web) Objects



Remote (Web) Objects



Representational State Transfer (REST)

- Resource manipulation via HTTP operations
- URL describes resource
 - `http://my.server.com/person/127.xml`
 - Server Address
 - Resource Name
 - Resource Id
 - Format
- Data encoded in XML or JSON
- GET, PUT, POST, DELETE operations are used to manipulate resources

iPhone HTTP Requests

```

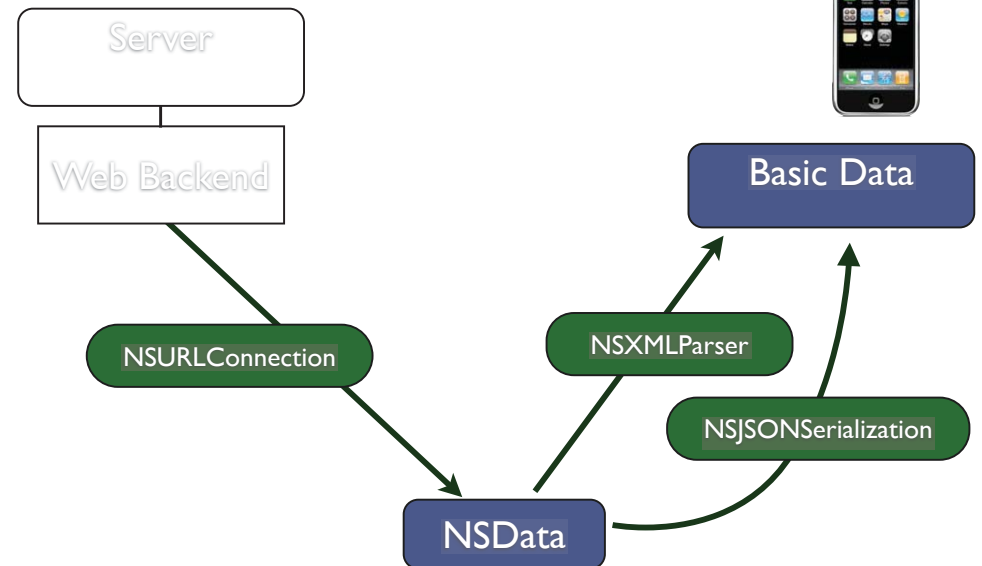
// create request
NSURL *url = [NSURL URLWithString:@"http://server/people.xml"];
request = [[NSMutableURLRequest alloc] initWithURL:url];

// set the HTTP operation
[request setHTTPMethod:@"POST"];

// set the post data
NSData *postData = [@"id=1&person[name]=Paul"
                    dataUsingEncoding:NSUTF8StringEncoding];
[request setHTTPBody:postData];

// fire the request
connection = [NSURLConnection connectionWithRequest:request
                                           delegate:self];
[connection start];
    
```

Remote (Web) Objects

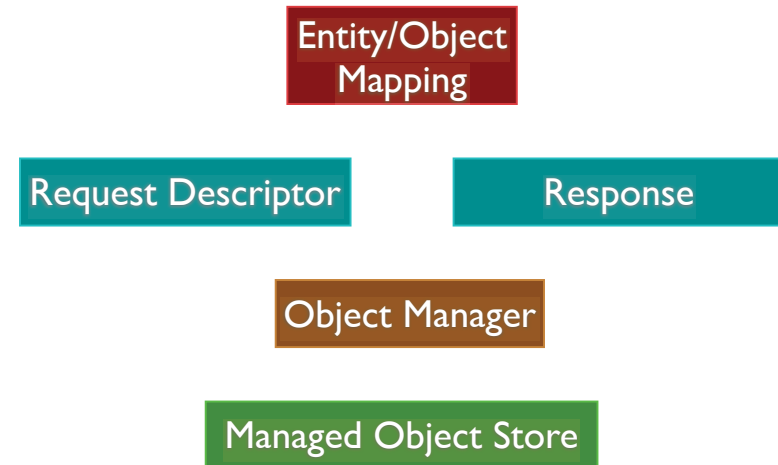


RestKit Basics



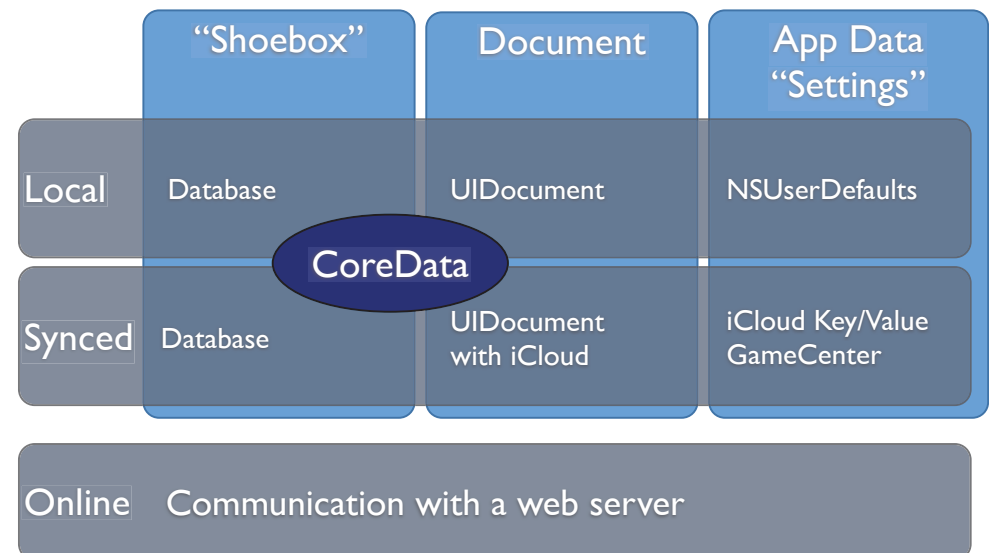
- RestKit knows RESTful Webservices
- Can interact with CoreData
- Key paradigm: Mapping response from the web to Objective-C objects

RestKit Structure



Demo

Data Handling Overview



More Information

- SQLite Online Documentation
- Apple Guides
 - Archives and Serialization Programming Guide
 - Core Data Programming Guide
 - Document-Based Application Programming Guide
 - URL Loading System Programming Guide
- restkit.org
- github.com/lichtschlag/iCloudPlayground
- github.com/lichtschlag/Chronos

Epilogue: Sharing data locally

- How to get data from app to app?
- “Open with...” ?
- UIDocumentInteractionController
 - Apps publish what files they can open
 - Sender app pushes document, user selects target app
 - Data is copied between app sandboxes
 - No way to track files

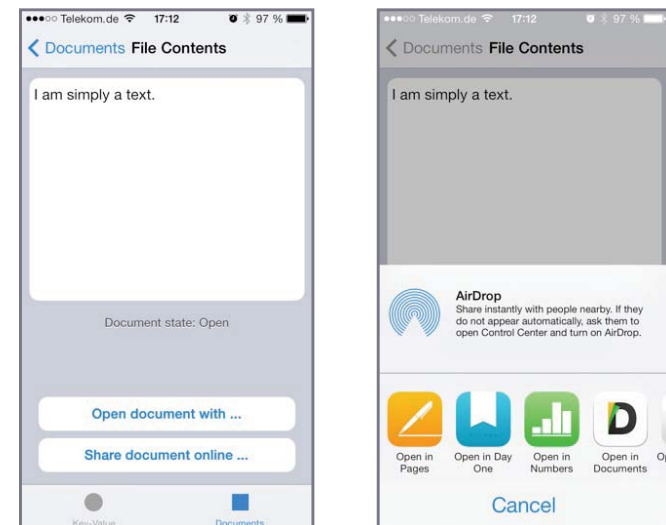
UIDocumentInteractionController

```
- (IBAction) openExternally:(id)sender
{
    // save first
    [self.document saveToURL:self.document.fileURL
                  forSaveOperation:UIDocumentSaveForOverwriting
                  completionHandler:^(BOOL success)
    {
        // bring up dialog from doc interaction controller
        self.docController = [UIDocumentInteractionController
                              interactionControllerWithURL:self.document.fileURL];

        BOOL didOpen = [docController presentOpenInMenuFromRect:CGRectZero
                              inView:self.openButton
                              animated:YES];

        if (!didOpen)
        {
            [[[UIAlertView alloc] initWithTitle:@"Cannot open file in other apps"
            message:@"Unfortunately, there is no app installed
            that can handle this kind of file."
            delegate:nil
            cancelButtonTitle:@"Ok"
            otherButtonTitles:nil] show];
        }
    }
};
```

UIDocumentController in Action



Check the iCloudPlayground demo code for more info