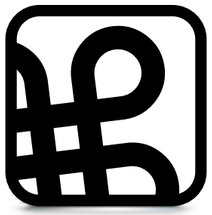
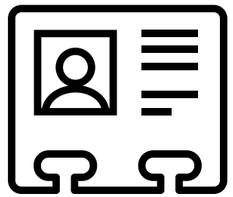


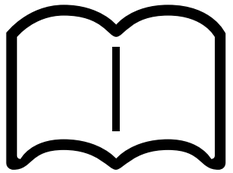
Macoun

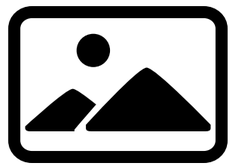
Einstellungen in der Wolke

Ortwin Gentz
@ortwingentz


19







Was ist iCloud?

- Synchronisationsdienst
- Austausch von Nutzerdaten zwischen Geräten
- Sandbox bleibt bestehen

Was ist iCloud nicht?

- Austausch zwischen Freunden
- Austausch zwischen beliebigen Apps
- Dropbox
- Messaging-Dienst

iCloud Backup

Reguläre Dokumente

Cache-Dateien in `NSCachesDirectory`

Dateien für Offline-Nutzung

```
[url setResourceValue:[NSNumber numberWithBool: YES]
                forKey:NSURLIsExcludedFromBackupKey
                 error:&error];
```

iOS 5.1+, Apple QA1719

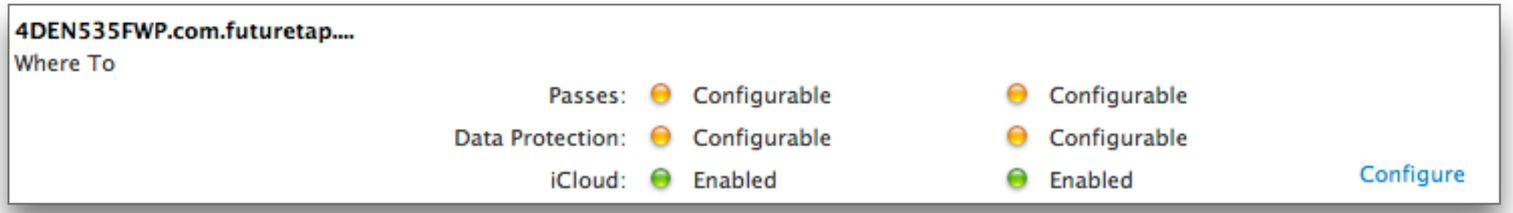
iCloud Storage

- Document Store
- Core Data Store
- Key-Value Store

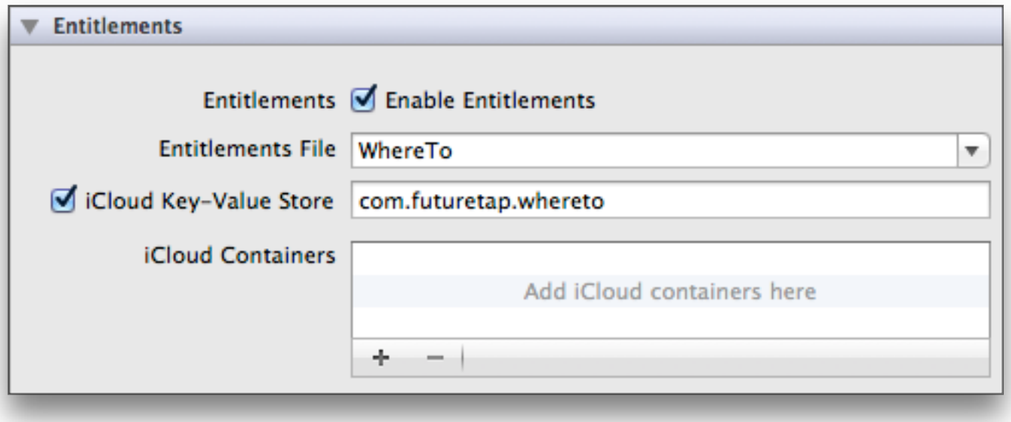
Provisioning

Provisioning

1. Provisioning Portal: „Enable for iCloud“



2. Xcode Entitlements



Grundsätzliches

- Mindestens zwei Geräte zum Test nötig, kein Simulator
- Mac-Apps nur über Mac App Store
- Document und Core Data Store: „Dokumente und Daten“ einschalten
- Fehler wie „NSUbiquitousKeyValueStore error: no valid com.apple.developer.ubiquity-kvstore-identifier entitlement“: App löschen und clean Build neu installieren
- Throttling bei zu vielen Updates

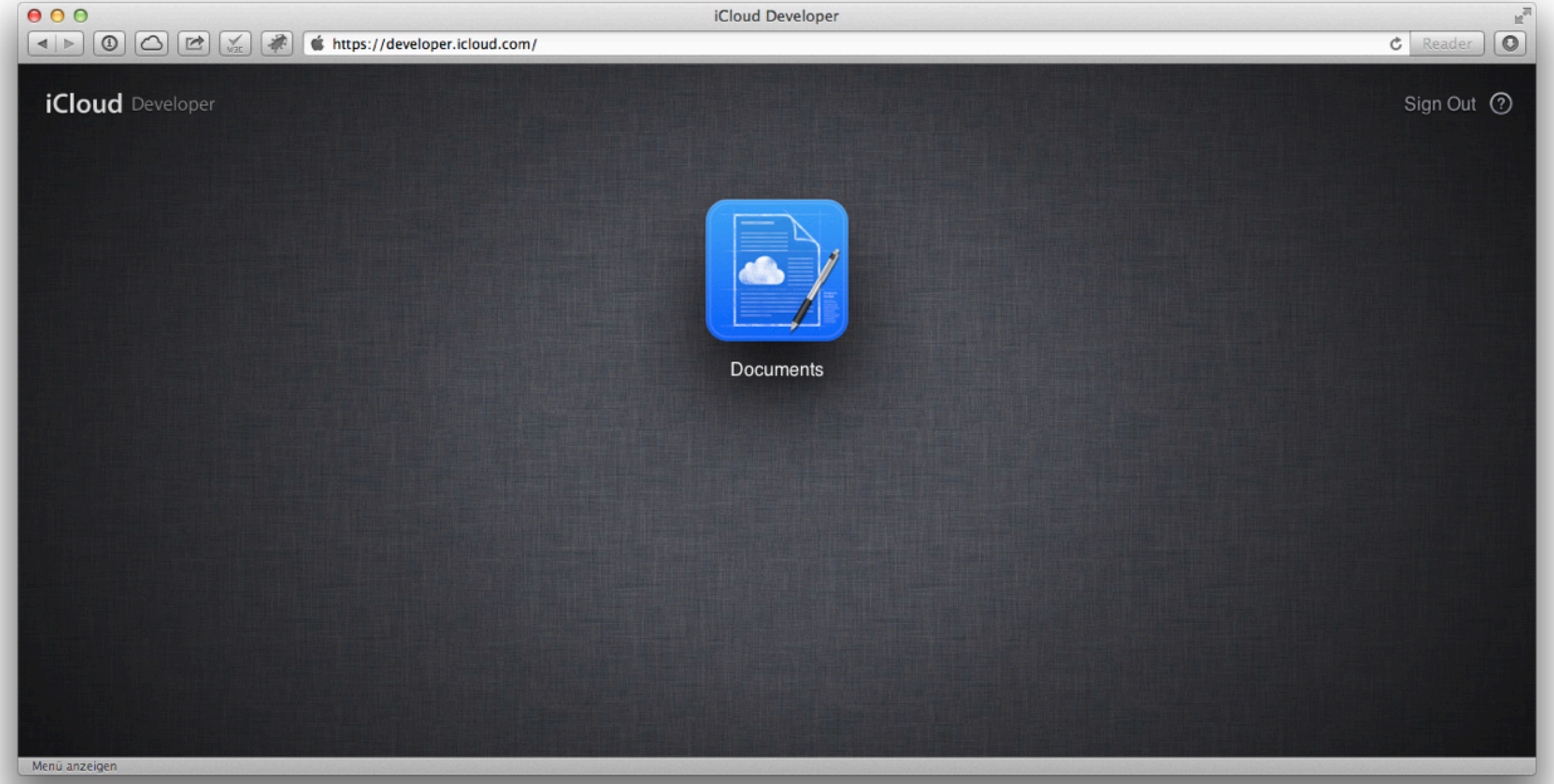
Document Store

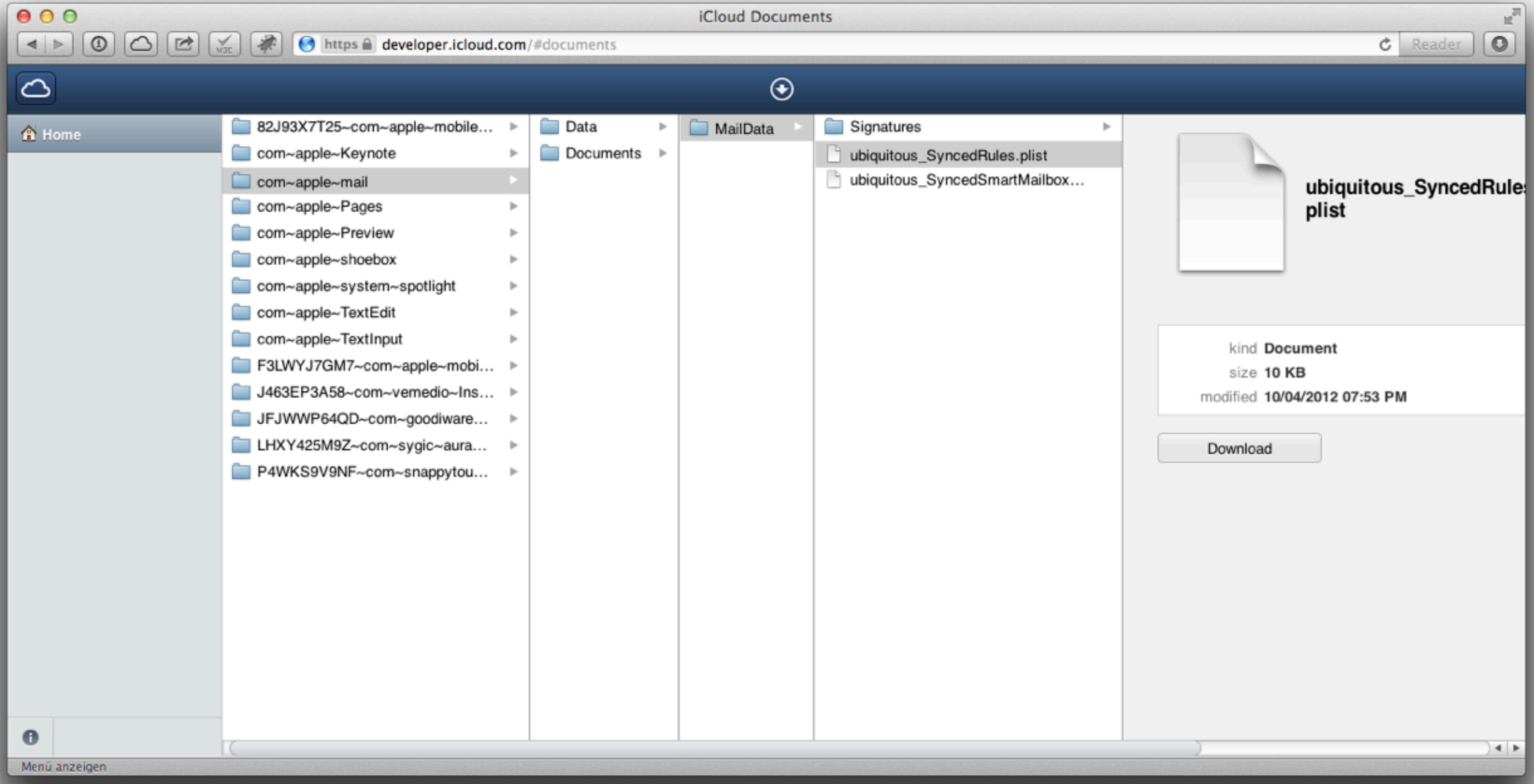
Document Store

- Zugriff per `NSFileManager` (`ubiquity/ubiquitous` Methoden)
- Schreib-/Lesekoordinierung über `NSFileCoordinator`
- Kommuniziert mit eigenem `<NSFilePresenter>` Objekt
- Alternative: `UIDocument` Subklasse, implementiert `<NSFilePresenter>`

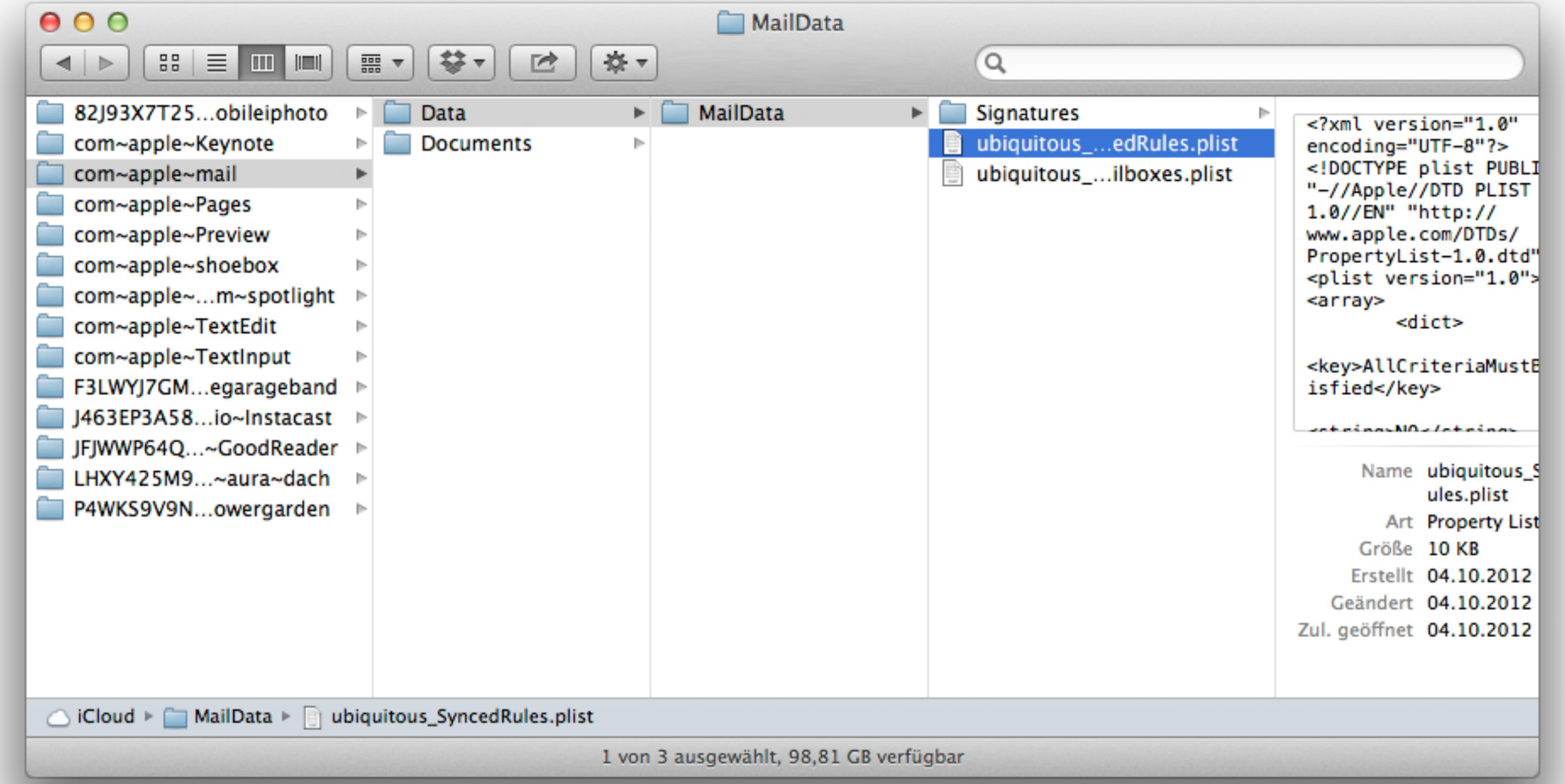
Document Store

- Suche: `NSMetadataQuery`
- Lazy Loading:
  `[NSFileManager startDownloadingUbiquitousItemAtURL:error:]`
- Test cases: Löschen, Aktualisieren, Bewegen, Ausloggen, Einloggen


iCloud Developer
https://developer.icloud.com/
Reader
iCloud Developer
Sign Out
Documents
Menü anzeigen


iCloud Documents
https developer.icloud.com/#documents
Reader
Home
82J93X7T25~com~apple~mobile…
com~apple~Keynote
com~apple~mail
com~apple~Pages
com~apple~Preview
com~apple~shoebox
com~apple~system~spotlight
com~apple~TextEdit
com~apple~TextInput
F3LWYJ7GM7~com~apple~mobi…
J463EP3A58~com~vemedio~Ins…
JFJWWP64QD~com~goodiware…
LHXY425M9Z~com~sygic~aura…
P4WKS9V9NF~com~snappytou…
Data
Documents
MailData
Signatures
ubiquitous_SyncedRules.plist
ubiquitous_SyncedSmartMailbox…
ubiquitous_SyncedRules
plist
kind Document
size 10 KB
modified 10/04/2012 07:53 PM
Download
Menü anzeigen

~/Library/Mobile Documents/



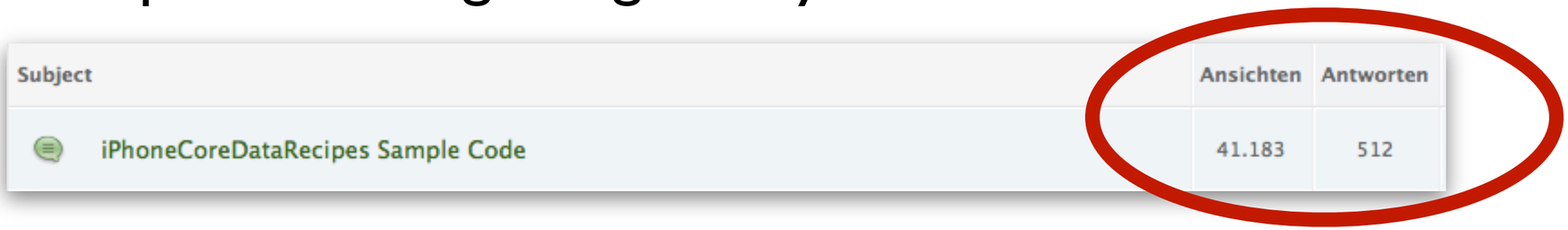
Core Data Store

Core Data Store

- Dokument-Apps: `UIManagedDocument`
- Shoebox-Apps: Option `NSPersistentStoreUbiquitousContentNameKey`
- Benachrichtigung über Cloud-Änderungen: `NSPersistentStoreDidImportUbiquitousContentChangesNotification`
- http://tinyurl.com/iCloudRecipes (Apple Sample Code)
- http://github.com/atomicbird/CloudNotes (WWDC 2011)

Probleme

- Löschen und Neuinstallieren der App
- Ausloggen und neu Einloggen, evtl. mit anderem Account
- Migrieren von Legacy-Daten zu iCloud
- Komplett-Verweigerung des Sync



Alternativen

- Simperium (kommerziell, kostenlos bis 2500 User/Monat)
- TICoreDataSync (MIT)

Key/Value Store

Key/Value Store

- `NSUbiquitousKeyValueStore`
- 1 MB, 1024 Keys (verschachtelte Strukturen möglich)
- Keine Anrechnung auf Quota
- „Dokumente und Daten“ muss nicht eingeschaltet werden
- `NSUserDefaults/NSDictionary` like Zugriff
- Letzte Änderung gewinnt

Key/Value Store

- `NSUbiquitousKeyValueStoreDidChangeExternallyNotification`
  - object: `NSUbiquitousKeyValueStore`
  - userInfo:

```
@{NSUbiquitousKeyValueStoreChangedKeysKey: @[keys],
 NSUbiquitousKeyValueStoreChangeReasonKey: <NSNumber>}
```

  - Reasons: ServerChange, InitialSyncChange, QuotaViolationChange, AccountChange

UserDefaults in der Wolke

UserDefaults in der Wolke

- Gewohnter Zugriff über NSUserDefaults
- Alle Änderungen in Key/Value-Store spiegeln
- Updates von iCloud in NSUserDefaults spiegeln

Schreiben in die Wolke

- Elementare Schreib-Methoden von NSUserDefaults:
  - `-setObject:forKey:`
  - `-removeObjectForKey:`
- Alle anderen Setter rufen die Elementar-Methoden auf

Schreiben in die Wolke

```
@implementation NSUserDefaults(iCloud)

+ (void)initialize {
  Swizzle([NSUserDefaults class], @selector(setObject:forKey:),
   @selector(my_setObject:forKey:));
}

- (void)my_setObject:(id)object forKey:(NSString*)key {
   [self my_setObject:object forKey:key]; // call original implementation
   [[NSUbiquitousKeyValueStore defaultStore] setObject:object forKey:key];
}

@end
```

Lesen von der Wolke

```
+ (void)updateFromiCloud:(NSNotification*)notification {
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    NSDictionary *dict = [[NSUbiquitousKeyValueStore defaultStore]
dictionaryRepresentation];

    [dict enumerateKeysAndObjectsUsingBlock:
        ^(NSString* key, id obj, BOOL *stop) {
            [defaults my_setObject:obj forKey:key]; // call original impl.
    }];

    NSMutableArray *removedKeys = [NSMutableArray arrayWithArray:[defaults
        dictionaryRepresentation].allKeys];
    [removedKeys removeObjectsInArray:dict.allKeys];
    [removedKeys enumerateObjectsUsingBlock:
        ^(NSString* key, NSUInteger idx, BOOL *stop) {
            [defaults my_removeObjectForKey:key]; // original impl.
    }];
    [defaults synchronize];

    [[NSNotificationCenter defaultCenter] postNotificationName:
        FTiCloudSyncDidUpdateNotification object:nil];
}
```

Stolpersteine

Stolpersteine

- Standard Apple Keys

```
(lldb) po [[[NSUserDefaults standardUserDefaults]
dictionaryRepresentation] allKeys]
(id) $2 = 0x07684ca0 <__NSArrayI 0x7684ca0>(
NSLanguages,
AppleITunesStoreItemKinds,
AppleLocale,
AppleLanguages,
NSInterfaceStyle
)
```

Stolpersteine

- Standard Apple Keys

  ➔ Blacklist

```
@"(^!|^Apple|^ATOutputLevel|Hockey|DateOfVersionInstallation|^MF|^NS|
Quincy|^BIT|^TV|UsageTime|^Web|preferredLocaleIdentifier)"
```

- Main Thread Blockade

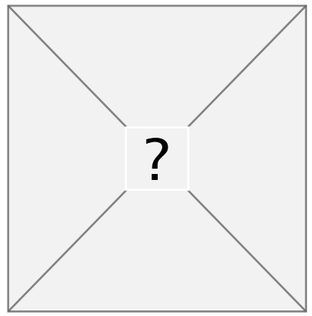
  ➔ dispatch_async()

- Fehlerbehandlung (Quota, iCloud-Verfügbarkeit)

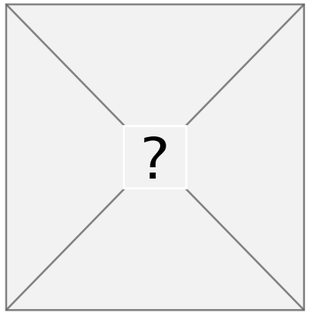
Vollständige Implementierung:

http://github.com/futuretap/FTiCloudSync

Einstellungen bearbeiten



oder in-App?



oder in-App?

- Nutzer verstehen Kontext-Switch nicht
- Einstellungen sind „weit weg“
- Einstellungen-App erlaubt keine dynamischen Elemente

InAppSettingsKit

InAppSettingsKit

- Akzeptiert das selbe Settings.bundle
- Arbeitet auf den selben NSUserDefaults
- Identische UI
- Diverse Add-Ons
- Kompatibel mit iCloud-Sync

InAppSettingsKit Add-Ons

- Eigene inApp Plists
- OpenURL
- MailCompose
- Button
- FooterText in MultiValue Listen
- Custom Group Header
- Custom ViewController
- TextAlignmentCenter/Right
- Icons
- Eigene Stores statt NSUserDefaults
- Notifications und Callbacks
- Dynamisches Cell-Hiding


Netzbetreiber
00:28
Wohin?
Favoriten
Kategorien
Verlauf

InAppSettingsKit anzeigen

```
@property (nonatomic, strong) IASKAppSettingsViewController *settingsVC;

- (void)viewDidLoad {
   [[NSNotificationCenter defaultCenter] addObserver:self
    selector:@selector(settingDidChange:) name:kIASKAppSettingChanged
    object:nil];
}

- (void)showSettings {
  self.settingsVC = [[IASKAppSettingsViewController alloc] init];
  [self.navigationController pushViewController:self.settingsVC
                                       animated:YES];
}
```

Dynamic Cell Hiding

```
- (void)settingDidChange:(NSNotification*)notification {
    if ([notification.object isEqual:@"Twitter"]) {
        BOOL enabled = (BOOL)[[notification.userInfo
         objectForKey:@"Twitter"] intValue];
        [self.settingsVC setHiddenKeys:(enabled ? nil :
         [NSSet setWithObjects:@"twitterId", @"twitterText", nil])
                              animated:YES];
    }
}
```

Erweiterbarkeit

- Custom Views und Section Header

```
- tableView:heightForSpecifier:
- tableView:cellForSpecifier:
- settingsViewController:tableView:didSelectCustomViewSpecifier:

- settingsViewController:tableView:heightForHeaderForSection:
- settingsViewController:tableView:viewForHeaderForSection:
```

- Subclassing-Unterstützung

Download:

http://github.com/futuretap/InAppSettingsKit

Fazit

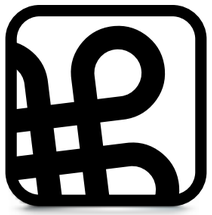
- Key/Value-Store robust und einfach in der Handhabung
- Einstellungen können einfach synchron gehalten werden
- Synchronisation fast ohne Aufwand
- InAppSettingsKit spart Code für Einstellungs-UIs

Fragen?

Ortwin Gentz
gentz@futuretap.com

Twitter @ortwingentz
App.net @ortwin

Vielen Dank



Macoun