Macoun'11

CDFinder
The Search Is Over

Gurke iTunes - Files aufs Fon

Norbert M. Doerner

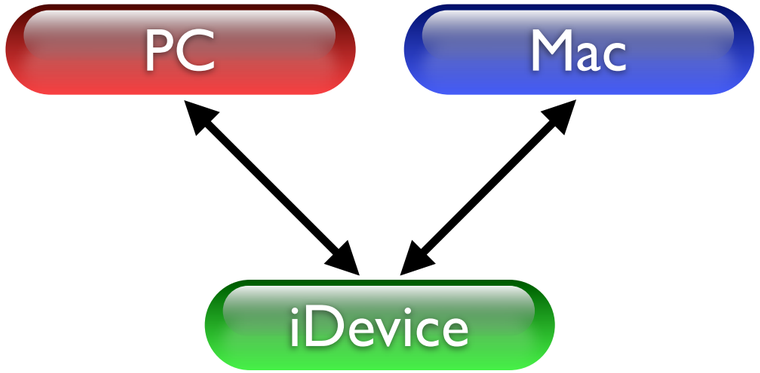
Gurke

"Die Gurke (Cucumis sativus) ist eine Art der Gattung Gurken (Cucumis) aus der Familie der Kürbisgewächse."

Quelle: Wikipedia

Files aufs Fon

Wozu?



Files aufs Fon

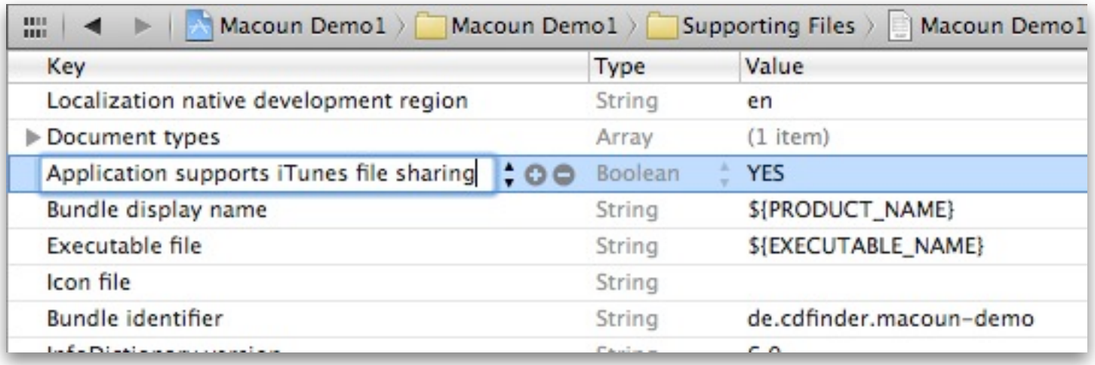
- iTunes File Sharing
- Bonjour & Sockets
- iCloud
- Dropbox

iTunes File Sharing

Info.plist

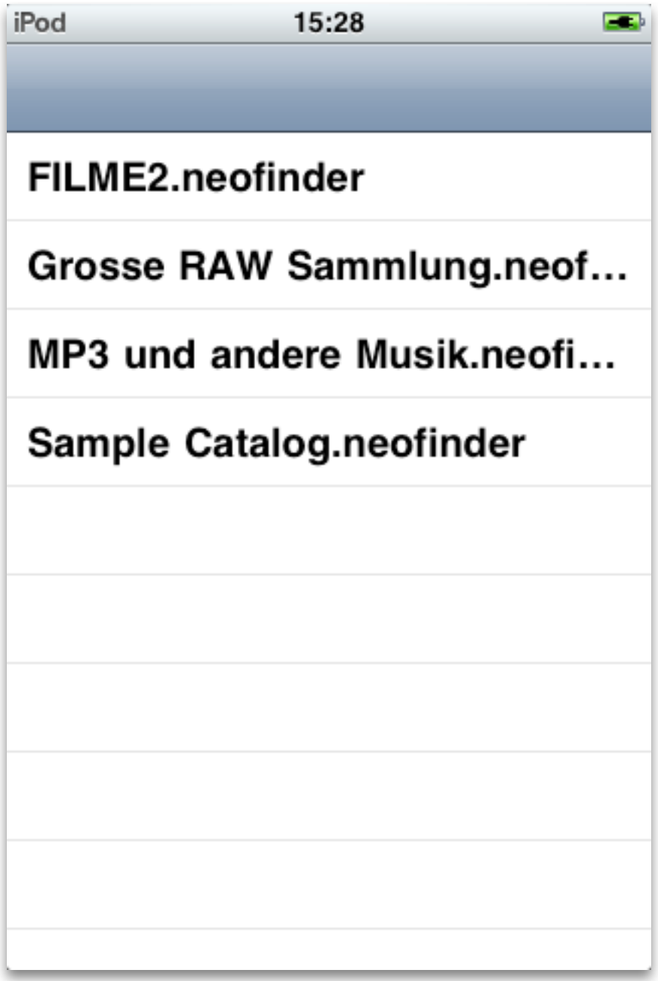
`UIFileSharingEnabled`

iTunes File Sharing



iTunes File Sharing

iTunes File Sharing



iTunes File Sharing

- ✓ Einfach
- ✓ Mac und Windows
- ✗ Grauenvolle GUI, wer findet denn so etwas?
- ✗ Keine Kontrolle über Dateitypen

Bonjour & Sockets

- Apple nutzt Bonjour für Time Machine, iTunes, Druckerauswahl...
- Bonjour selbst ist nur Service Discovery!

Bonjour & Sockets

Client

- Bonjour Browser
- Empfänger auswählen
- Daten senden

Server

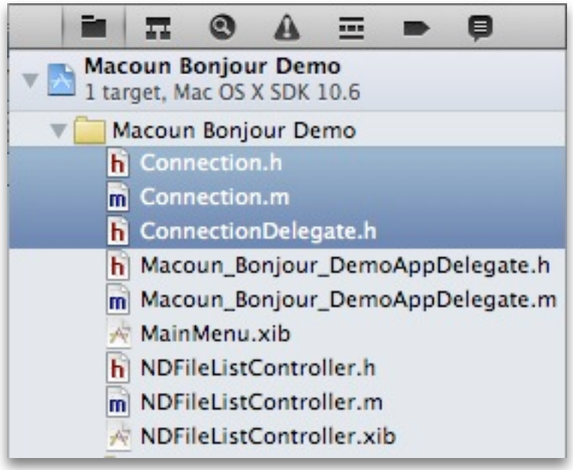
- Dienst über Bonjour freigeben
- Daten empfangen und verarbeiten

Bonjour & Sockets

- CFSocket, CFSocketNativeHandle, CFReadDataStreamRef
- CFWriteDataStreamRef, Object Serialization, CFRunLoops

Bonjour & Sockets

- Lösung: "Connection" aus dem Chatty Projekt
- `http://www.mobileorchard.com/wp-content/uploads/2009/05/chatty.zip`
- `http://mobileorchard.com/tutorial-networking-and-bonjour-on-iphone/`
- Peter Bakhyryev



Demo

Bonjour: Server

```
  // Socket anlegen und öffnen
  CFSocketContext socketCtxt = {0, self, NULL, NULL, NULL}; // link
auf "self" ist wichtig!

  listeningSocket = CFSocketCreate (kCFAllocatorDefault,
                  PF_INET,      // IPv4
                  SOCK_STREAM,     // der Socket soll Streaming sein
                  IPPROTO_TCP,     // wir wollen TCP
                  kCFSocketAcceptCallBack,
                  (CFSocketCallBack)&serverAcceptCallback,
                  &socketCtxt);
  if (listeningSocket == NULL)
     return;
```

Bonjour: Server

```
  struct sockaddr_in socketAddress;
  memset(&socketAddress, 0, sizeof(socketAddress));
  socketAddress.sin_len = sizeof(socketAddress);
  socketAddress.sin_family = AF_INET;    // IPv4
   socketAddress.sin_port = 0;    // wir nehmen jeden Port
  socketAddress.sin_addr.s_addr = htonl(INADDR_ANY);

NSData *socketAddressData = [NSData dataWithBytes:&socketAddress
length:sizeof(socketAddress)];

// das macht das "bind"
  if (CFSocketSetAddress(listeningSocket, (CFDataRef)
socketAddressData) != kCFSocketSuccess )
  { // Fehler }
```

Bonjour: Server

```
// welche PortNummer haben wir bekommen?

NSData *socketAddressActualData = [(NSData *)CFSocketCopyAddress
(listeningSocket) autorelease];

struct sockaddr_in *socketAddressActual = (struct sockaddr_in *)
[socketAddressActualData bytes];

port = ntohs(socketAddressActual->sin_port);
```

Bonjour: Server

```
  // den CFSocket in die RunLoop hängen

  CFRunLoopRef currentRunLoop = CFRunLoopGetCurrent();

  CFRunLoopSourceRef runLoopSource = CFSocketCreateRunLoopSource
(kCFAllocatorDefault, listeningSocket, 0);

CFRunLoopAddSource(currentRunLoop, runLoopSource,
kCFRunLoopCommonModes);

  CFRelease(runLoopSource);
```

Bonjour: Server

```
  #define kMacounDemoServicesTypeIdentifier     @"_gurken._tcp"

// Bonjour Bescheid sagen, daß wir einen Dienst anbieten

  syncNetService = [[NSNetService alloc] initWithDomain:@""
type:kMacounDemoServicesTypeIdentifier name:@"" port:port];

  [syncNetService scheduleInRunLoop:[NSRunLoop currentRunLoop]
forMode:NSRunLoopCommonModes];

  [syncNetService setDelegate:self];
  [syncNetService publish];
```

Bonjour: Server

```
static void serverAcceptCallback(CFSocketRef socket,
CFSocketCallBackType type, CFDataRef address, const void *data, void
*info)
{
  RootViewController      *myViewController = (RootViewController *)
info;
  if (type != kCFSocketAcceptCallBack) return;
  CFSocketNativeHandle nativeSocketHandle = *(CFSocketNativeHandle*)
data;
  Connection* connection = [[Connection alloc]
initWithNativeSocketHandle:nativeSocketHandle];

  [myViewController addConnection:connection];
}
```

Bonjour: Server

```
- (void)addConnection:(Connection *)newConnection
{
  [newConnection setDelegate:self];
}  //  addConnection
```

Bonjour: Server

```
- (void) receivedNetworkPacket:(NSDictionary*)packet viaConnection:
(Connection*)connection
{
  if ([packet objectForKey:@"liste"] != nil)
  {
      // Dateiliste zurückschicken
      [connection sendNetworkPacket:[NSDictionary
dictionaryWithObject:mLocalFilesArray forKey:@"liste"]];
      return;
  }
}
```

Bonjour: Client

```
- (void)applicationDidFinishLaunching:(NSNotification *)
aNotification
{
  services = [[NSMutableArray alloc] init];

  netServiceBrowser = [[NSNetServiceBrowser alloc] init];
  [netServiceBrowser setDelegate:self];
  [netServiceBrowser
searchForServicesOfType:kMacounDemoServicesTypeIdentifier
inDomain:@""];
}
```

Bonjour: Client

```
- (void)netServiceBrowser:(NSNetServiceBrowser *)netServiceBrowser
didFindService:(NSNetService *)netService moreComing:(BOOL)
moreServicesComing
{
  [services addObject:netService];
  if (moreServicesComing == NO)
     [self sortAndUpdateUI];
}
```

Bonjour: Client

```
- (void)netServiceBrowser:(NSNetServiceBrowser *)netServiceBrowser
didRemoveService:(NSNetService *)netService moreComing:(BOOL)
moreServicesComing
{
  [services removeObject:netService];
  if (moreServicesComing == NO)
     [self sortAndUpdateUI];
}
```

Bonjour: Client

```
- (IBAction)verbinden:(id)sender
{
  NSNetService   *selectedService = [services objectAtIndex:
[bonjourListView selectedRow]];

  dateiLister = [[NDFileListController alloc] init];
  [dateiLister window];
  [dateiLister verbindeMitService:selectedService];
  [netServiceBrowser stop];  // damit der nicht endlos weiter rödelt
}
```

Bonjour: Client

```
- (void)verbindeMitService:(NSNetService *)service
{
  [[self window] makeKeyAndOrderFront:self];
  theService = [service retain];
  [theService setDelegate:self];

  [theService resolveWithTimeout:5.0];
}  //  verbindeMitService
```

Bonjour: Client

```
- (void)netService:(NSNetService *)sender didNotResolve:
(NSDictionary *)errorDict
{
  NSLog (@"didNotResolve");
}

- (void)netServiceDidResolveAddress:(NSNetService *)sender
{
  [self dateiListeHolen];
}
```

Bonjour: Client

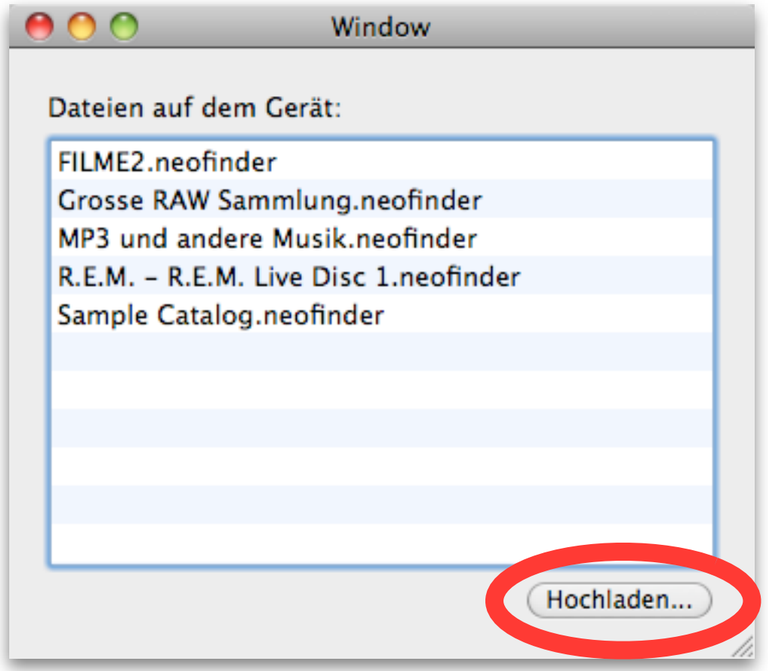
```
- (void)dateiListeHolen
{
  [dateiListe removeAllObjects];
  [dateiListe release];
  if (myConnection == nil)
  {
     myConnection=[[Connection alloc]initWithNetService:theService];
     [myConnection setDelegate:self];
     [myConnection connect];
  }
  [myConnection sendNetworkPacket:[NSDictionary
dictionaryWithObject:@" " forKey:@"liste"]];
}  //  dateiListeHolen
```

Bonjour: Client

```
- (void) receivedNetworkPacket:(NSDictionary*)packet viaConnection:
(Connection*)connection
{
  NSArray *neueDateiListe = [packet objectForKey:@"liste"];

  if (neueDateiListe != nil)   // Liste angekommen
  {
     dateiListe = [[neueDateiListe mutableCopy] retain];
     [dateiListView reloadData];
  }
}
```

Bonjour: Client



Bonjour: Client

```
- (IBAction)selectFile:(id)sender
{
  NSOpenPanel       *myPanel = [NSOpenPanel openPanel];
  [myPanel setCanChooseDirectories:NO];
  [myPanel setCanChooseFiles:YES];
  [myPanel setAllowsMultipleSelection:NO];
  [myPanel setAllowedFileTypes:[NSArray
arrayWithObjects:@"neofinder", nil]];
  if ([myPanel runModal] == NSFileHandlingPanelOKButton)
  {
     NSArray     *selectedFiles = [myPanel filenames];
     for (NSString *onePath in selectedFiles)
        [self dateiHochladen:onePath];
  }
}  //  selectFile
```

Bonjour: Client

```
- (void)dateiHochladen:(NSString *)path
{
  NSData        *fileData = [NSData dataWithContentsOfFile:path];
  NSString      *fileName = [path lastPathComponent];
  if (myConnection == nil)
  {...}
  NSMutableDictionary   *dict = [[NSMutableDictionary dictionary]
autorelease];
  [dict setObject:fileName forKey:@"name"];
  [dict setObject:fileData forKey:@"datei"];
  [myConnection sendNetworkPacket:dict];
}  //  dateiHochladen
```

Bonjour: Server, Nachtrag

```
- (void) receivedNetworkPacket:(NSDictionary*)packet viaConnection:
(Connection*)connection
{
  if ([packet objectForKey:@"datei"] != nil)
  {
     NSString    *dateiName = [packet objectForKey:@"name"];
     NSData      *dateiInhalt = [packet objectForKey:@"datei"];
     [dateiInhalt writeToFile:[documentsPath
stringByAppendingPathComponent:dateiName] atomically:YES];
     [self updateFiles];
     [self.tableView reloadData];
     [connection sendNetworkPacket:[NSDictionary
dictionaryWithObject:mLocalFilesArray forKey:@"liste"]];
     return;
  }
}
```

Bonjour & Sockets

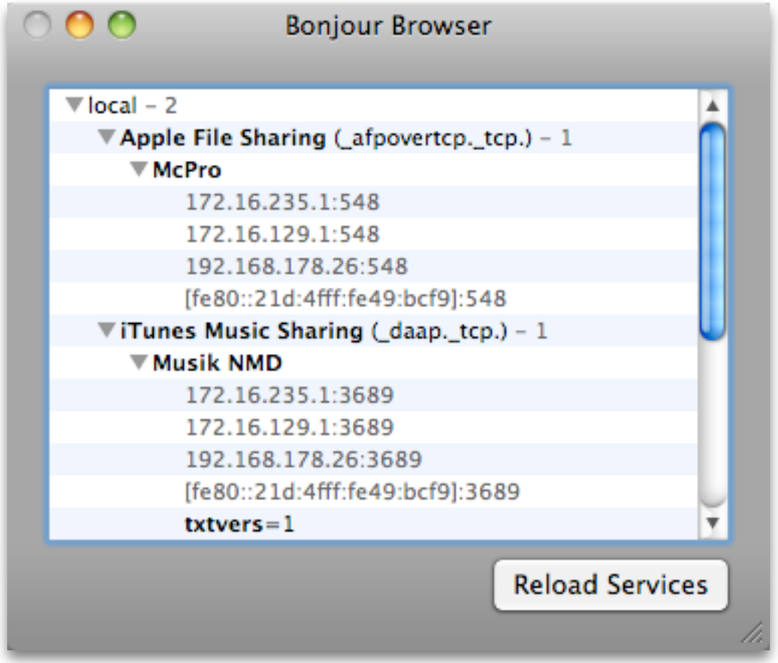
- vor dem echte Shippen den DNS Service Type registrieren!
- `http://www.dns-sd.org/ServiceTypes.html`

Bonjour & Sockets

- Session 211 von der WWDC 2011
- "Bonjour Network Discovery and Connectivity"

Bonjour & Sockets

- Nützlich: Bonjour Browser
- `http://www.tildesoft.com/`



Bonjour & Sockets

- ✓ Volle Kontrolle
- ✓ Desktop Sync-Funktion ist “sichtbar”
- ✓ Für Mac und Windows
- ✗ Recht viel Code nötig
- ✗ Nur im gleichen Subnetz (mit Ausnahmen)

Aussichten: Wolkig

iCloud

- "I really shouldn't have to do that myself"

iCloud

- iCloud Storage: Viel mehr als nur eine Datei-Synchronisation
- Tief im System integriert
- sofortige Synchronisation, vollautomatisch

iCloud Storage API

- NSFileManager, NSFileCoordinator
- NSMetaDataQuery
- UIDocument / NSDocument
- Key-Value Store (iOS und Lion, NSUbiquitousKeyValueStore)

iCloud Storage API

NSFileManager

```
- (NSURL *)URLForUbiquityContainerIdentifier:(NSString *)
containerID;

- (BOOL)isUbiquitousItemAtURL:(NSURL *)url;

- (BOOL)setUbiquitous:(BOOL)flag itemAtURL:(NSURL *)url
destinationURL:(NSURL *)destinationURL error:(NSError **)errorOut;

- (BOOL)startDownloadingUbiquitousItemAtURL:(NSURL *)url error:
(NSError **)errorOut;
```

iCloud Storage API

NSMetadataQuery

`NSMetadataQueryLocalDocumentsScope`

`NSMetadataQueryUbiquitousDocumentsScope`

iCloud Storage API

```
  // alle Dateien auflisten, die in "Documents" liegen:
NSMetadataQuery          *metadataQuery;
metadataQuery = [[[NSMetadataQuery alloc] init] autorelease];
[metadataQuery setSearchScopes:[NSArray
arrayWithObject:NSMetadataQueryUbiquitousDocumentsScope]];
[metadataQuery setPredicate: [NSPredicate predicateWithFormat: @"%K
like '*'", NSMetadataItemFSNameKey]];

[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(metadataQueryDidFinishGathering:)
name:NSMetadataQueryDidFinishGatheringNotification object:nil];

[metadataQuery setDelegate:self];
[metadataQuery startQuery];
```

iCloud Storage API

```
- (void)metadataQueryDidFinishGathering:(NSNotification *)
notification
{
    NSArray                    *results;
    NSMetadataQuery            *metadataQuery = (NSMetadataQuery*)
[notification object];
    results = [metadataQuery results];
    [metadataQuery stopQuery];
    // Ergebnisse anschauen und gewünschte Dateien herunterladen
    for (NSMetadataItem *oneItem in results)
      NSURL *url = [oneItem valueForAttribute:NSMetadataItemURLKey];
```

iCloud Storage API

NSMetadataItem

```
NSString * const NSMetadataItemIsUbiquitousKey;
NSString * const NSMetadataUbiquitousItemHasUnresolvedConflictsKey;
NSString * const NSMetadataUbiquitousItemIsDownloadedKey;
NSString * const NSMetadataUbiquitousItemIsDownloadingKey;
NSString * const NSMetadataUbiquitousItemIsUploadedKey;
NSString * const NSMetadataUbiquitousItemIsUploadingKey;
NSString * const NSMetadataUbiquitousItemPercentDownloadedKey;
NSString * const NSMetadataUbiquitousItemPercentUploadedKey;
```

iCloud Storage API

- UIDocument
  - Asynchrones Lesen und Schreiben
  - ähnlich zu NSDocument auf Mac OS X

iCloud Storage API

- NSFileCoordinator
  - Zugriff mit anderen Prozessen koordinieren
- NSFilePresenter
- NSFileVersion

iCloud

com.apple.developer.ubiquity-container-identifiers

For more information on developing external accessories to communicate with your iOS application, view MFi Program.

For more information on utilizing iCloud in your applications, view What's New in iOS 5.

Description	Apple Push Notification service	In App Purchase	Game Center	iCloud	Action
[redacted]34A.de.cdfinder.ma... Macoun Demo Apps	Configurable for Development Configurable for Production	Enabled	Enabled	Enabled	Configure
[redacted]4A.de.cdfinder.ne... NeoFinder2go	Configurable for Development Configurable for Production	Enabled	Enabled	Enabled	Configure

iCloud

http://iphonesdkdev.blogspot.com/2011/06/icloud-app-sample-and-entitlement-code.html

iCloud

Tuesday.

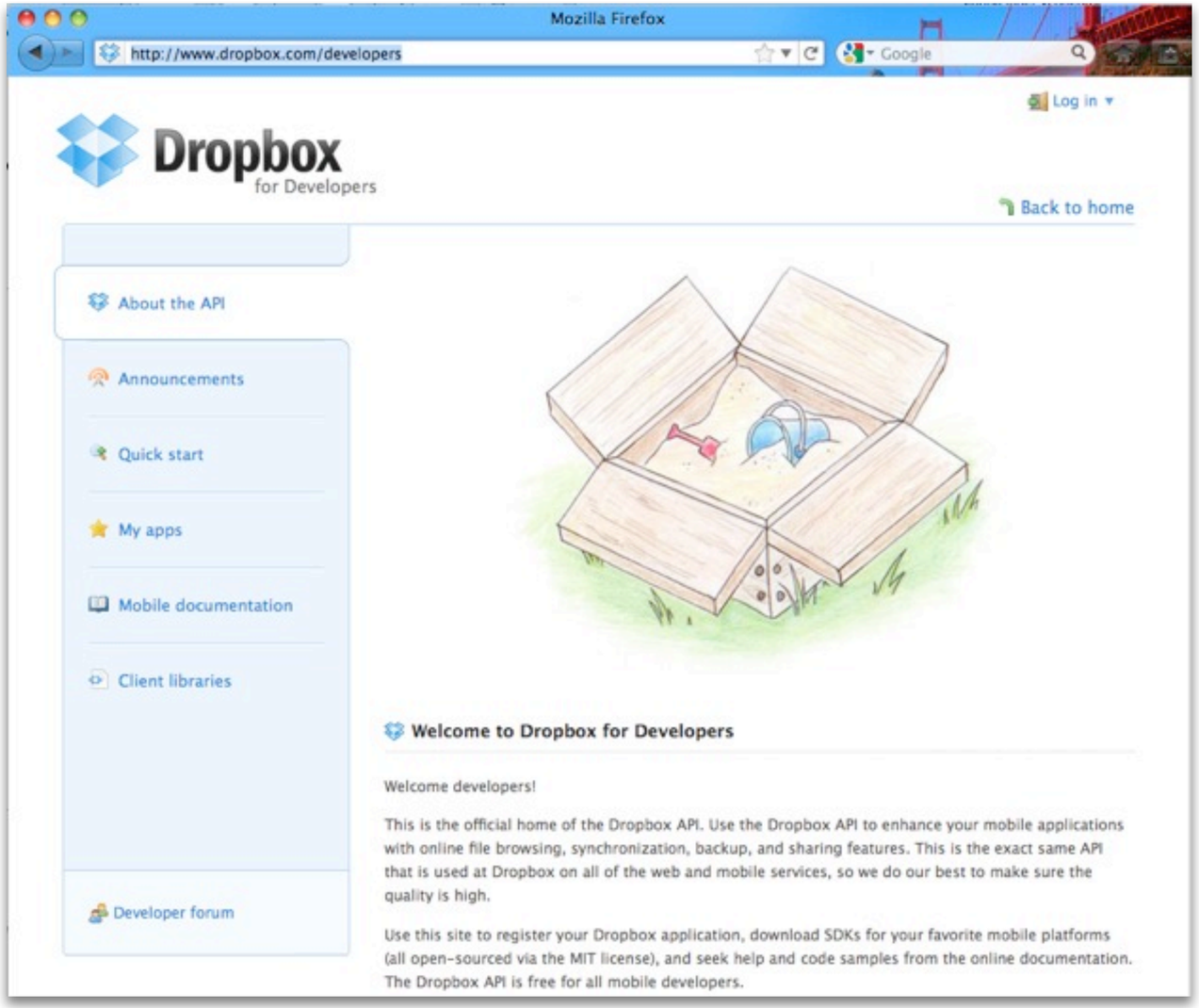
iCloud

- ✓ Einigermaßen einfach zu implementieren
- ✗ Mac OS X nur Programme in der Sandbox
- ✗ Überwiegende Käfighaltung, Windows??
- ✗ API brandneu, muß sich erst noch bewähren
- ✗ nur für Mac OS X 10.7.2 und iOS 5 und neuer

Dropbox

- http://www.dropbox.com/developers

Dropbox


Mozilla Firefox
http://www.dropbox.com/developers
Google
Log in
Dropbox
for Developers
Back to home
About the API
Announcements
Quick start
My apps
Mobile documentation
Client libraries
Developer forum
Welcome to Dropbox for Developers
Welcome developers!
This is the official home of the Dropbox API. Use the Dropbox API to enhance your mobile applications with online file browsing, synchronization, backup, and sharing features. This is the exact same API that is used at Dropbox on all of the web and mobile services, so we do our best to make sure the quality is high.
Use this site to register your Dropbox application, download SDKs for your favorite mobile platforms (all open-sourced via the MIT license), and seek help and code samples from the online documentation. The Dropbox API is free for all mobile developers.

Dropbox

- SDK für iOS herunterladen:

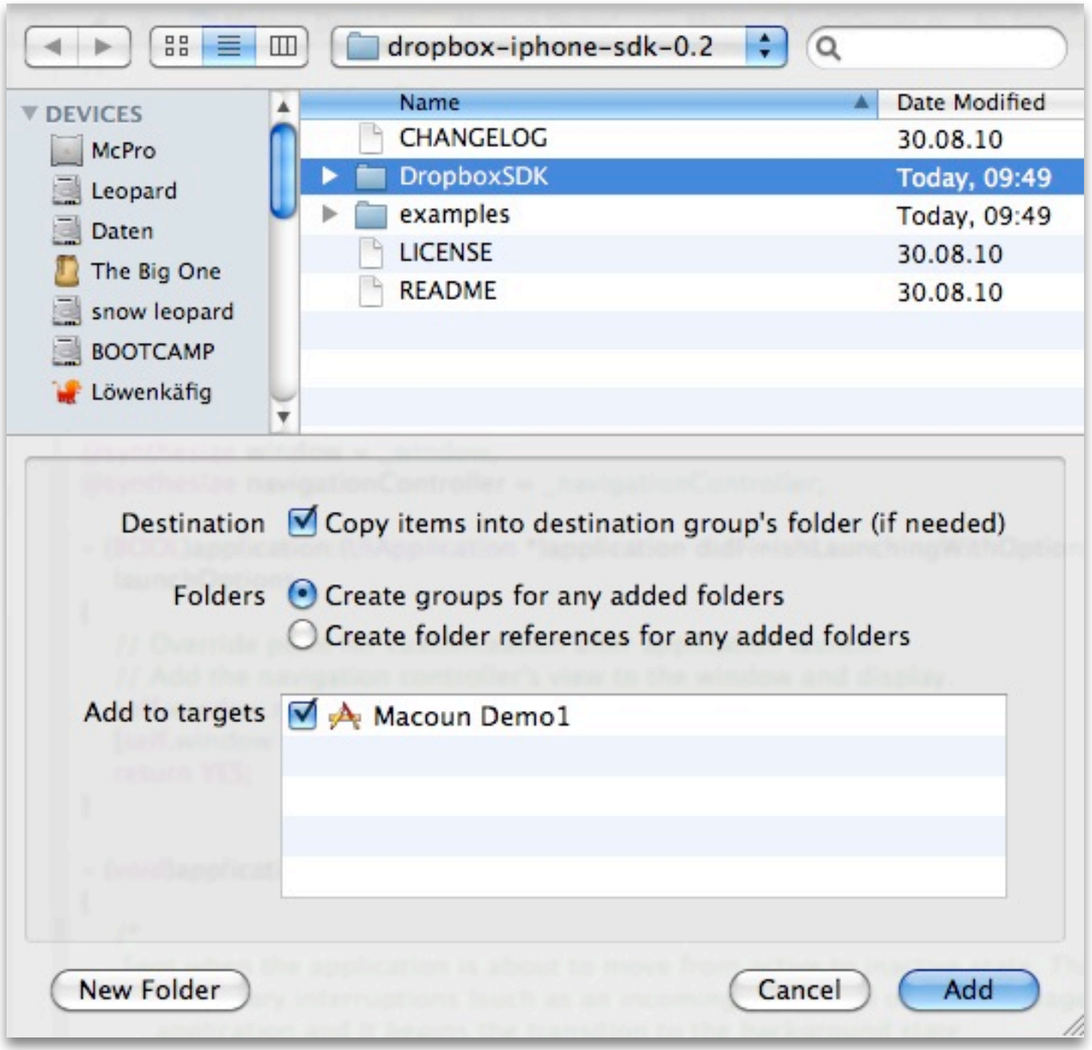
  `http://www.dropbox.com/developers/releases`

- Bei Dropbox einmal als Entwickler registrieren

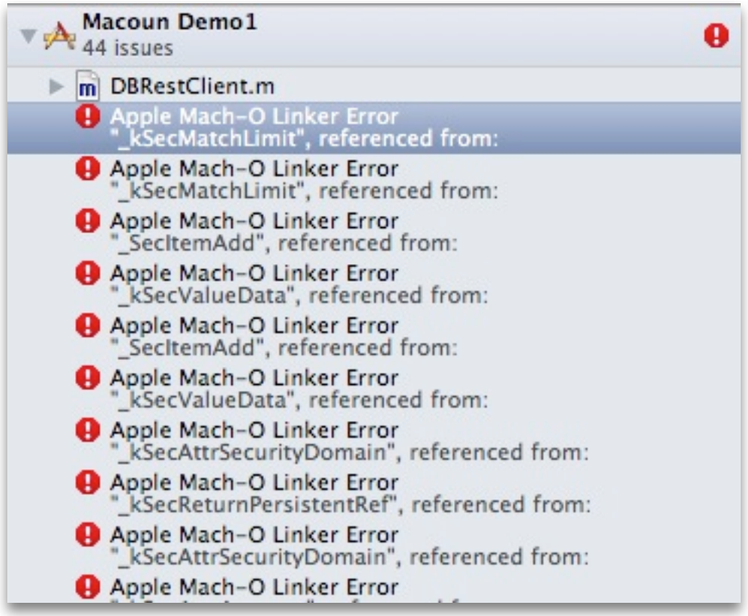
Dropbox

- Bei Dropbox "New App" anlegen
- Das erzeugt AppKey und AppSecret für Entwicklung
- Mit Entwicklerkey nur Zugriff auf eigene Dropbox!

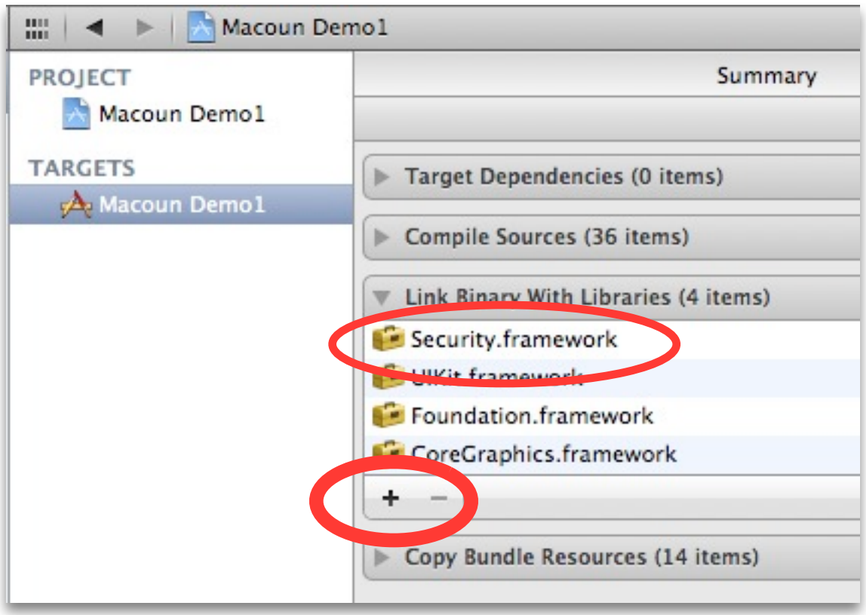
Dropbox


dropbox-iphone-sdk-0.2
DEVICES
McPro
Leopard
Daten
The Big One
snow leopard
BOOTCAMP
Löwenkäfig
Name
Date Modified
CHANGELOG
30.08.10
DropboxSDK
Today, 09:49
examples
Today, 09:49
LICENSE
30.08.10
README
30.08.10
Destination
Copy items into destination group's folder (if needed)
Folders
Create groups for any added folders
Create folder references for any added folders
Add to targets
Macoun Demo1
New Folder
Cancel
Add

Dropbox


Macoun Demo1
44 issues
DBRestClient.m
Apple Mach-O Linker Error
"_kSecMatchLimit", referenced from:
Apple Mach-O Linker Error
"_kSecMatchLimit", referenced from:
Apple Mach-O Linker Error
"_SecItemAdd", referenced from:
Apple Mach-O Linker Error
"_kSecValueData", referenced from:
Apple Mach-O Linker Error
"_SecItemAdd", referenced from:
Apple Mach-O Linker Error
"_kSecValueData", referenced from:
Apple Mach-O Linker Error
"_kSecAttrSecurityDomain", referenced from:
Apple Mach-O Linker Error
"_kSecReturnPersistentRef", referenced from:
Apple Mach-O Linker Error
"_kSecAttrSecurityDomain", referenced from:
Apple Mach-O Linker Error

Dropbox


Macoun Demo1
PROJECT
Macoun Demo1
TARGETS
Macoun Demo1
Summary
Target Dependencies (0 items)
Compile Sources (36 items)
Link Binary With Libraries (4 items)
Security.framework
UIKit.framework
Foundation.framework
CoreGraphics.framework
+
−
Copy Bundle Resources (14 items)

Demo

Dropbox

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
  self.window.rootViewController = self.navigationController;
  [self.window makeKeyAndVisible];

  DBSession* dbSession =
  [[[DBSession alloc]
    initWithConsumerKey:@"12345" consumerSecret:@"abcd"]autorelease];
  [DBSession setSharedSession:dbSession];

  return YES;
}
```

Dropbox

```
- (void)didPressLink
{
  if ([[DBSession sharedSession] isLinked] == YES)
  {
      [self loadFileListFromDropbox];
     return;
  }
  DBLoginController* controller = [[DBLoginController new]
autorelease];
  controller.delegate = self;
  [controller presentFromController:self];
}
```

Dropbox

```
- (void)loginControllerDidCancel:(DBLoginController*)controller
{
  NSLog (@"loginControllerDidCancel");
}


// Verbindung zu DropBox war erfolgreich, Verbindung steht
- (void)loginControllerDidLogin:(DBLoginController*)controller
{
  [self loadFileListFromDropbox];
}
```

Dropbox

```
- (DBRestClient*)restClient
{
  if (restClient == nil)
  {
     restClient = [[DBRestClient alloc] initWithSession:[DBSession
sharedSession]];
     restClient.delegate =  self;
  }
  return restClient;
}
```

Dropbox

```
- (void)loadFileListFromDropbox
{
  [self.restClient loadMetadata:@"/NeoFinder" withHash:nil];
}
```

Dropbox

```
- (void)restClient:(DBRestClient*)client loadedMetadata:
(DBMetadata*)metadata
{
  NSArray* validExtensions = [NSArray arrayWithObjects:@"neofinder",
nil];
  NSMutableArray* newPaths = [NSMutableArray new];
  for (DBMetadata* child in metadata.contents) {
     if (child.isDirectory == NO){
        NSString* extension = [[child.path pathExtension]
lowercaseString];
        if ([validExtensions indexOfObject:extension] != NSNotFound)
           [newPaths addObject:child.path];
     }
  }
  [self syncDropboxPaths:newPaths];
}
```

Dropbox

```
- (void)syncDropboxPaths:(NSArray *)dbPaths
{
  for (NSString *onePath in dbPaths)
  {
     NSString    *oneFileName = [onePath lastPathComponent];
     if ([mLocalFilesArray indexOfObject:oneFileName] == NSNotFound)
     {
        [self.restClient loadFile:onePath intoPath:[documentsPath
stringByAppendingPathComponent:oneFileName]];
     }
  }
}  //  syncDropboxPaths
```

Dropbox

```
- (void)restClient:(DBRestClient*)client loadedFile:(NSString*)
localPath
{
//   NSLog(@"File loaded into path: %@", localPath);
  [self updateFiles];
  [self.tableView reloadData];
}

- (void)restClient:(DBRestClient*)client loadFileFailedWithError:
(NSError*)error
{
  NSLog(@"There was an error loading the file - %@", error);
}
```

Dropbox

- Zur Distribution "in der Wildnis" neuen AppKey und AppSecret beantragen
- App URL, Icon, etc. bei Dropbox hinzufügen
- Fertig

Dropbox

- ✓ Recht einfach zu implementieren
- ✓ Bewährt
- ✓ Nutzung der API kostenlos
- ✓ Für Mac und Windows, auch Desktop Clients vorhanden
- ✗ API Dokumentation z. Zt. recht mager

Files aufs Fon

- "Echter" Code sollte dann natürlich auch syncen können!
- Fehlerbehandlung
- Fortschritts-Anzeigen

Aussichten: Sturmwarnung

Fragen?

Vielen Dank!

Norbert M. Doerner

ndoerner@cdfinder.de

www.cdfinder.de

Macoun'11