

7

Übersicht

- GIT in XCode
- Die Grundlagen (40min)
- Programmieraufgabe (0,5h – 1h)
- Erweiterte UIElemente (30min)
- Programmieraufgabe (1h - 2h)

Teil I

- GIT in XCode
- Obj-C Sprachfeatures
 - Properties
 - Collections
 - id-Pointer
 - Protocols
- Garbage Collection (ARC) vs Memory-Management
- Delegate-Pattern in iOS

Teil II

- UI-Elements:
 - UITableView
 - UINavigationController
 - TabBarController
- Daten senden und Empfangen per HTTP
 - synchron
 - asynchron
- JSON in Obj-C

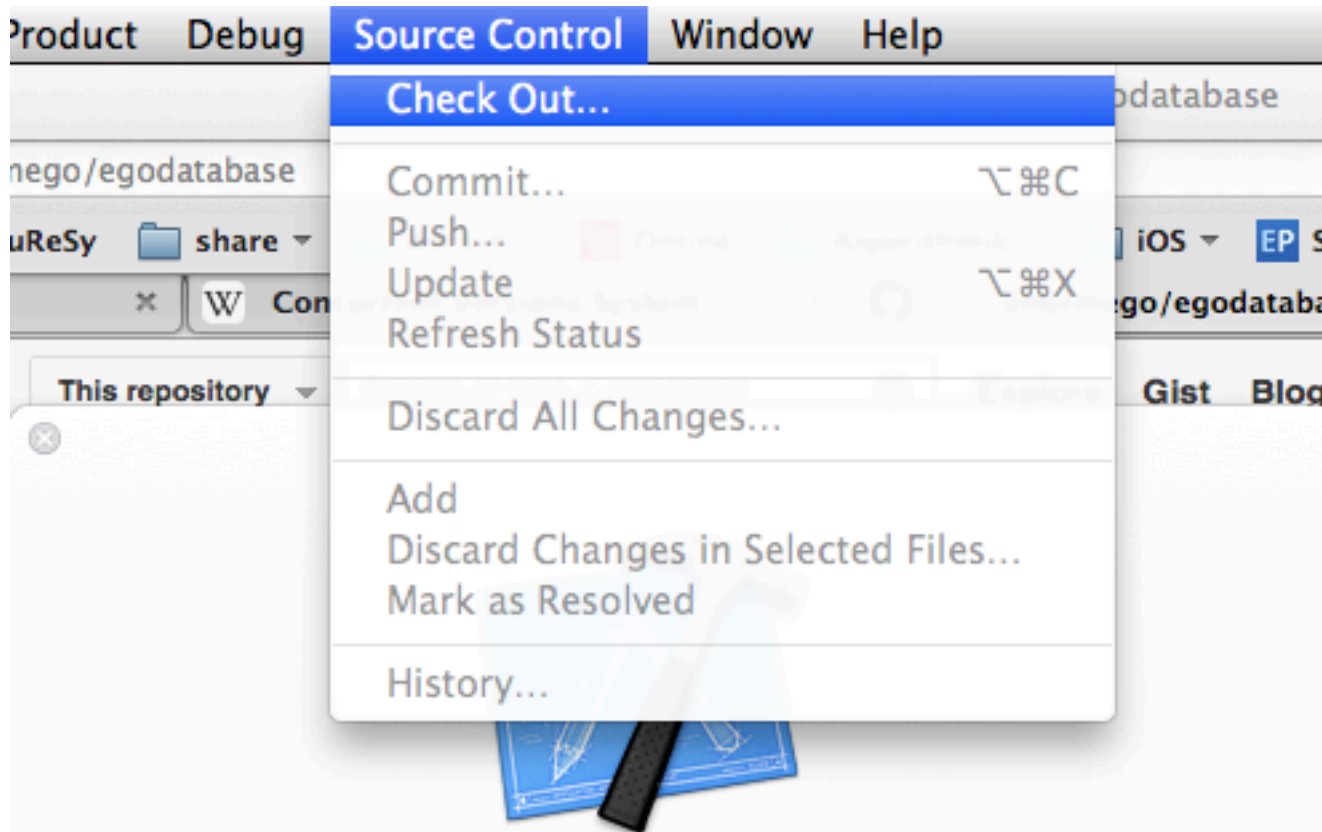
Versionsverwaltung

- Änderungen werden mit Zeitstempel und Diff gespeichert
- Erzeugt eine Änderungskette der Datei
- alte Versionen können wiederhergestellt werden
- Ermöglicht gleichzeitiges Entwickeln

Git

- Nicht-lineare Entwicklung
 - Branches
- Kein zentraler Server
 - jeder Entwickler hat lokale Kopie des ganzen Repositories

Git in XCode



Welcome to Xcode

Version 5.0.1 (5A2053)

<http://posdorfer.de/mlab/>

Check Out

Choose an item:

Recents

Favorites

Repositories



egodatabase.git

<https://github.com/enormego/egodatabase.git>



Receiver

<https://github.com/Estimote/beacons-demo.git>



iSturesy

<https://subversion.assembla.com/svn/isturesy/trunk/iSturesy>



UI7Kit

<https://github.com/youknowone/UI7Kit.git>



Underwriter

9posdorf@mobisvm1.informatik.uni-hamburg.de:/srv/mlab2013/repos/capgeminiX.git



VVDocumenter-Xcode

<https://github.com/posdorf/VVDocumenter-Xcode.git>

Or enter a repository location:

<https://github.com/enormego/egodatabase.git>

Cancel

Previous

Next

Check Out

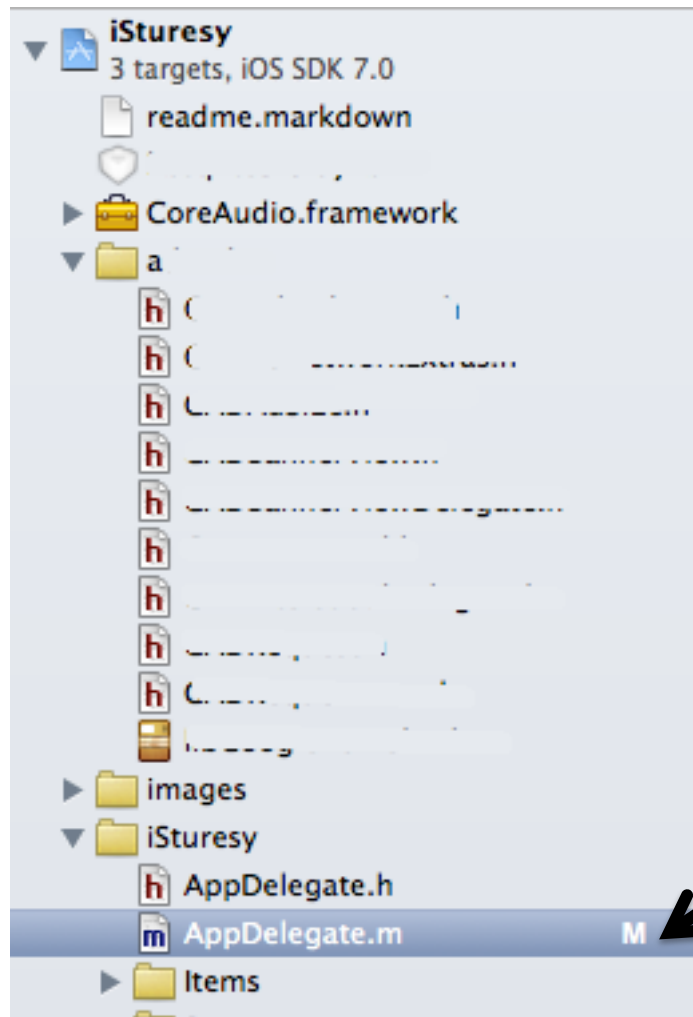


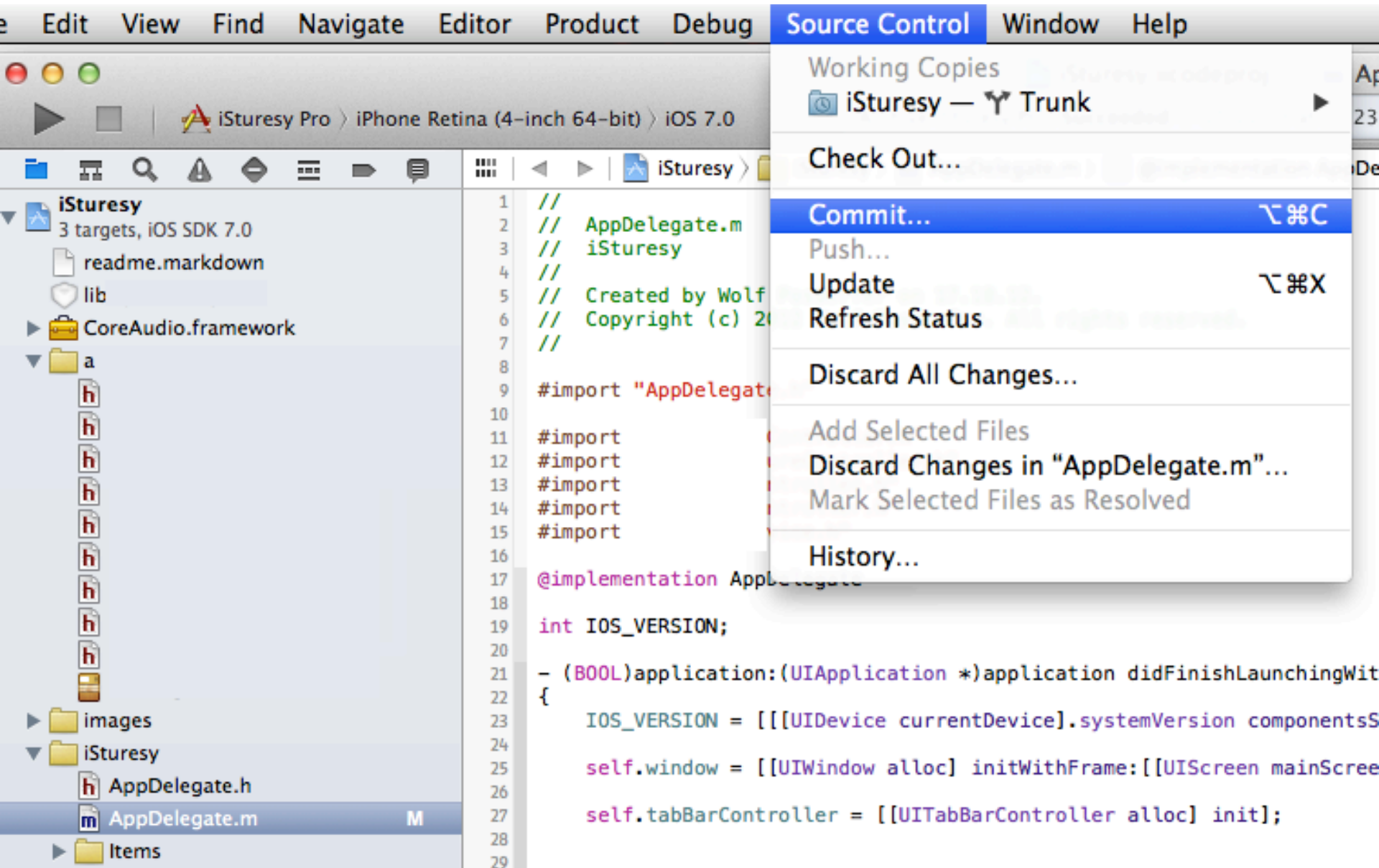
Check Out Successful.

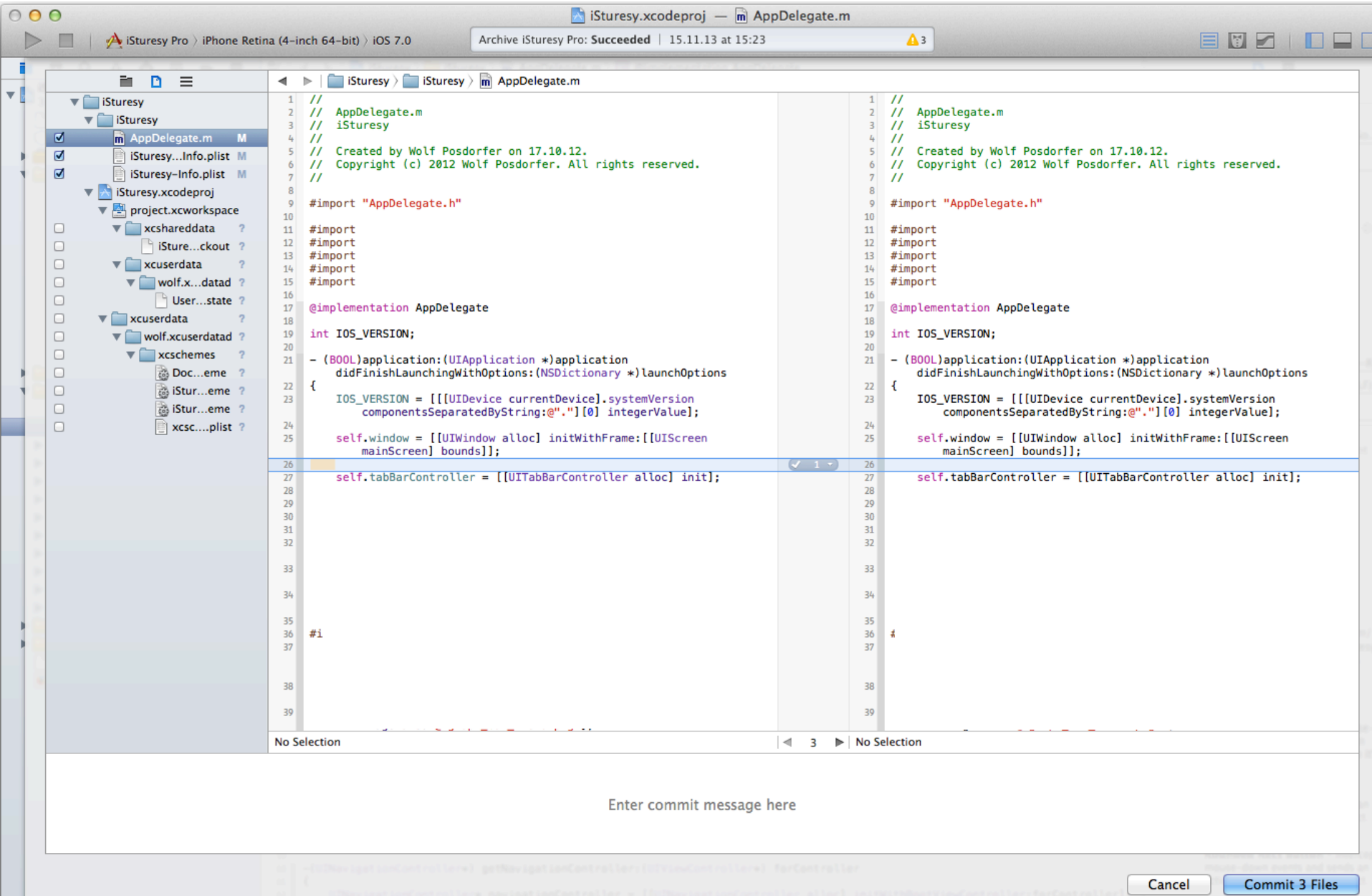
Cancel

Previous

Show



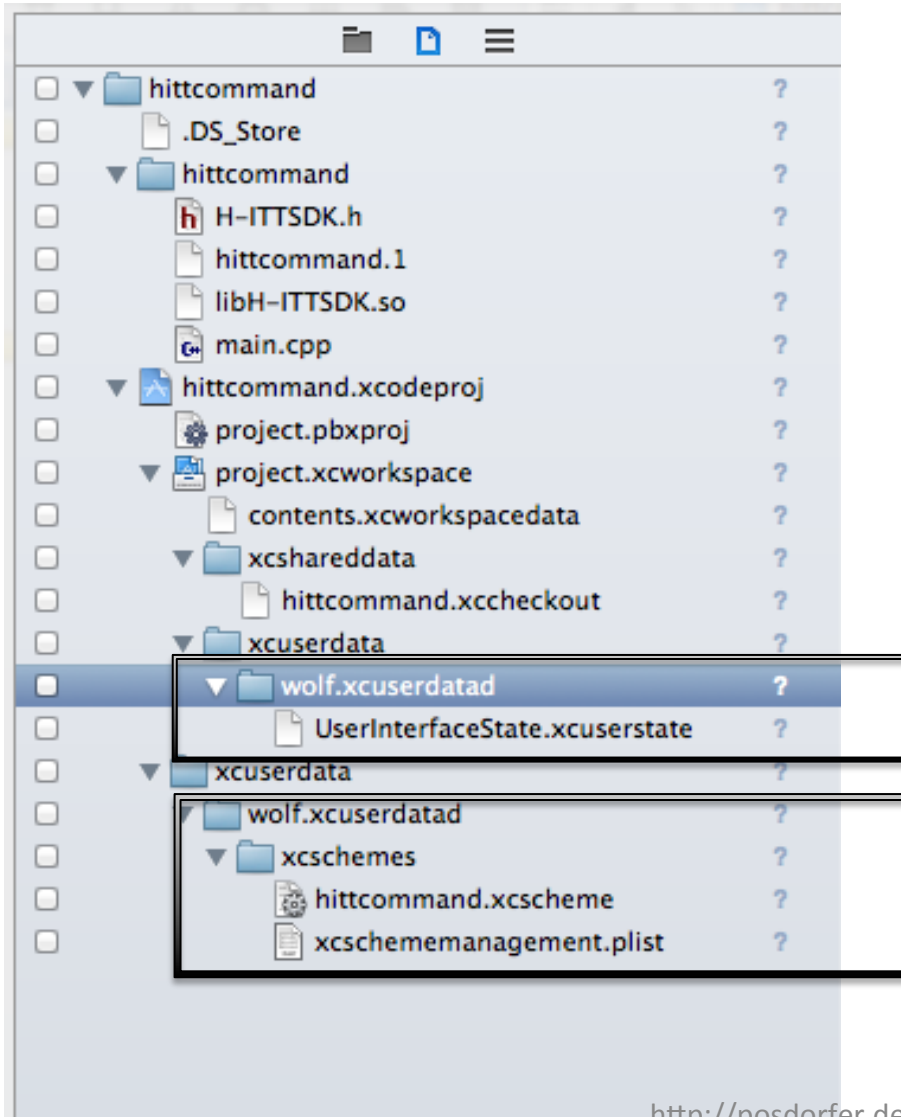




Projekt zu Git hinzufügen

- Leeres Git Repository clonen
- Neues XCode Projekt öffnen
- Speicherort des Projektes innerhalb des geclonten lokalen GIT-Repos legen
- Source-Control -> Commit
- Source-Control -> Push

Beachtenswertes



Userdaten sollten nicht comittet werden.
Ist aber nicht schlimm wenns dochmal passiert

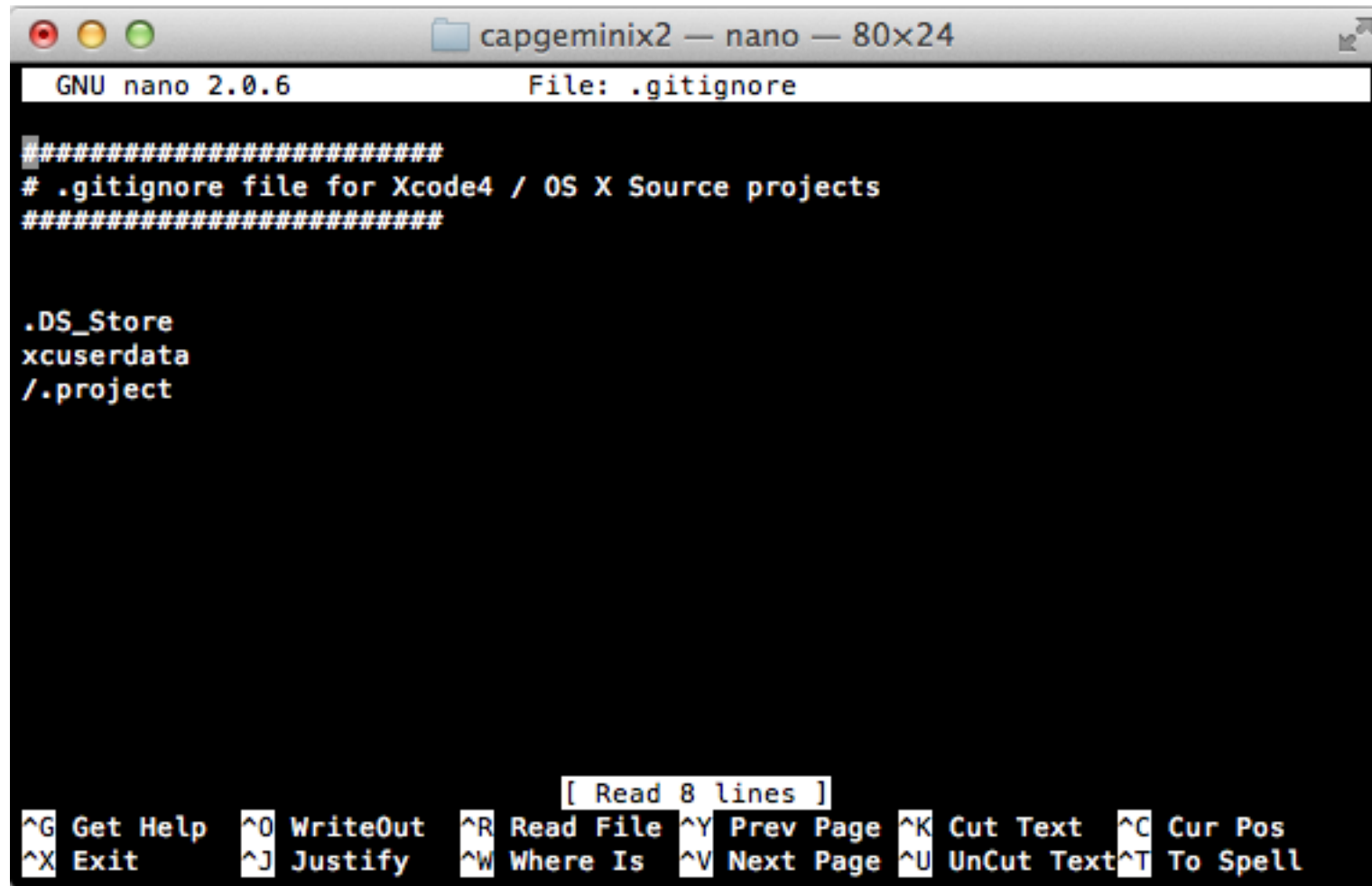
.gitignore

- In der Datei „.gitignore“ können Dateien auf eine Blacklist gesetzt werden (zB Userdata)
- „Leider“ nur übers Terminal erreichbar:

```
cd somePath/mygitfolder/project/  
nano .gitignore
```

.gitignore

1. Dateinamen eintragen
2. CTRL + X
3. Y
4. Enter



The screenshot shows a terminal window titled "capgeminix2 — nano — 80x24". The editor is GNU nano 2.0.6, editing a file named ".gitignore". The file content is as follows:

```
#####  
# .gitignore file for Xcode4 / OS X Source projects  
#####  
  
.DS_Store  
xcuserdata  
/.project
```

At the bottom of the window, a status bar displays "[Read 8 lines]" and a list of keyboard shortcuts:

^G Get Help	^O WriteOut	^R Read File	^Y Prev Page	^K Cut Text	^C Cur Pos
^X Exit	^J Justify	^W Where Is	^V Next Page	^U UnCut Text	^T To Spell

Übersicht

- GIT in Xcode ✓
- Die Grundlagen (40min)
- Programmieraufgabe (0,5h – 1h)
- Erweiterte UIElemente (30min)
- Programmieraufgabe (1h - 2h)

Obj-C

- Objekt orientiert
- C-Funktionen verfügbar
- Methoden im Smalltalk Style
 - [*someObject* *doSomething:VALUE with:VALUE*];
- Getters/Setters über Punktnotation
 - *someObject.someField* = *VALUE*
[*someObject setSomeField:VALUE*];
 - *VALUE* = *someObject.someField*;
VALUE = [*someObject someField*];
 - Punktnotation kann für Nebeneffekte misbraucht werden...

Objects

- Strikte Trennung von Schnittstelle und Implementation

- interface: **SomeClass.h**

```
@interface SomeClass
    -(void) someMethod;
    +(void) staticMethod;
@end
```

- implementation: **SomeClass.m**

```
@interface SomeClass () } Optionale Redeklarationen
@end
@implementation SomeClass
    -(void) someMethod { .... }
    +(void) staticMethod { .... }
@end
```

Objects

SomeObject* soob = [[SomeObject alloc] init];

← Objekte mit *
← Speicher allozieren
← Objekt initialisieren

```
-(id) initWithX:(id) X andY:(id) Y andZ:(id) Z
{
    self = [super init];
    if(self) // (self != nil)
    {
        self.x = X; self.y = Y; self.z=Z;
    }
    return self;
}
```

Properties

Properties benutzen um Getter/Setter automatisch zu generieren

```
@interface SomeClass  
  
@property (...) int name;  
@property (...) NSString * name2;  
  
@end
```

Attributes:

Standard: (atomic, readwrite, assign)

Properties

- Synchronizing-Attributes:
 - **atomic**, synchronized → langsamer Zugriff
 - **nonatomic**, not synchronized → schneller Zugriff
- Access-Attributes:
 - **readonly**, Nur Getter
 - **readwrite**, Getter + Setter
- Reference-Attributes:
 - **assign**, Kein Referenz-Zählen, nur für nicht-Pointer Felder
 - **copy**, erzeugt eine Kopie von veränderlichen Objekten
 - **weak**, Kein Referenz-Zählen, für Pointer oder IBOutlets, verhindert Retain-Zyklen (siehe Delegate-Pattern)
 - **retain**, Erhöht Referenzzähler, für Pointer-Felder

Properties redefinieren

interface: **SomeClass.h**

```
@interface SomeClass
@property (nonatomic, retain, readonly) NSString* description;
@end
```

implementation: **SomeClass.m**

```
@interface SomeClass ()
@property (nonatomic, retain, readwrite) NSString* description;
@end

@implementation SomeClass
@end
```

Hinter den Kulissen

```
@property (retain, readonly) NSObject* obj;
```

```
-(NSObject*) obj { //getter  
    @synchronized{  
        return _obj;  
    }  
}
```

```
-(void) setObj:(NSObject*) someValue { //setter  
    @synchronized {  
        [someValue retain]; // increment reference count  
        [_obj release];    // decrement reference count  
        _obj = someValue;  
    }  
}
```


Collections

- **NSArray**
 - „generische“ nichtveränderliche **Liste**
- **NSMutableArray** (Subclass von NSArray)
 - „generische “ veränderliche **Liste**
- **NSDictionary**
 - „generische“ nichtveränderliche **Map**
- **NSMutableDictionary** (Subclass von NSDictionary)
 - „generische “ veränderliche **Map**

NSArray

```
NSArray* array = [NSArray alloc] initWithObjects: id,....., nil];  
NSArray* array2 = @[value1,value2,value3]; //Syntaktischer Zucker
```

```
id returnVal = [array objectAtIndex:1];  
NSString* uncheckedCast = returnVal;
```

```
NSMutableArray* mutArra = [[NSMutableArray alloc] init];  
[mutArra addObject:someObject];
```

NSDictionaries

```
NSMutableDictionary* dict = [[NSMutableDictionary alloc] init];
```

```
[dict setObject: @"value" forKey: @"key"]; //HashMap<String,String>
```

```
id value = [dict objectForKey: @"key" ];
```

Protocol

- wie Java Interfaces, nur besser ;-)
- Unterstützt multiples Subtyping
- definieren zu implementierende Methoden
- definieren *optionale* Methoden
- definieren Properties
- können eigene Header-Datei bekommen, meist aber Teil einer anderen

Protocol Beispiel

```
@protocol EinProtocol
- (NSUInteger)eineZahl;

- (CGFloat)eineZahlMit:
  (NSUInteger)Index;

@property NSUInteger eineProperty;

@end
```

Protocol Beispiel 2

interface: **SomeClass.h:**

```
@protocol SomeDelegate; //protocol deklarieren
```

```
@interface SomeClass
```

```
@property(nonatomic,weak,readonly) id<SomeDelegate> delegate;  
@end
```

```
@protocol SomeDelegate //protocol definieren
```

```
-(void) someDelegate:(id<SomeDelegate>) delegate someAction:  
(int) integer;
```

```
@optional
```

```
-(void) someOtherStuff; //respondToSelector:
```

```
@end
```

UI Elemente

- UIElemente werden durch **IBOutlet**s im Code verankert
- Actions werden durch **IBAction**s im Code verankert
- Kann im Interface sein, für öffentliche Sichtbarkeit (Cont.h)
- oder in der Implementation für Private Sichtbarkeit (Cont.m)

Delegate Pattern I

- Funktionalitäten durch andere Klassen erledigen lassen
- Einzelner Beobachter (Observer-Pattern)
- Meistgenutztes Pattern im Cocoa/Cocoa-Touch Framework

```
@protocol SomeClassDelegate;
```

```
@interface SomeClass  
-(id) initWithDelegate:(id<SomeClassDelegate>) del;  
@end
```

```
@protocol SomeClassDelegate  
-(void) someClassDelegate:(id)del action:(int) integer;  
@end
```


Delegate Pattern II

```
@interface MyObject()<SomeClassDelegate>
// MyObject implementiert SomeClassDelegate
@end
```

```
@implementation MyObject
```

```
-(void) someMethod {
    SomeClass * clas = [[SomeClass alloc]
initWithDelegate:self];
}
```

```
-(void) someClassDelegate:(id)del action:(int) integer{
    ....
}
```

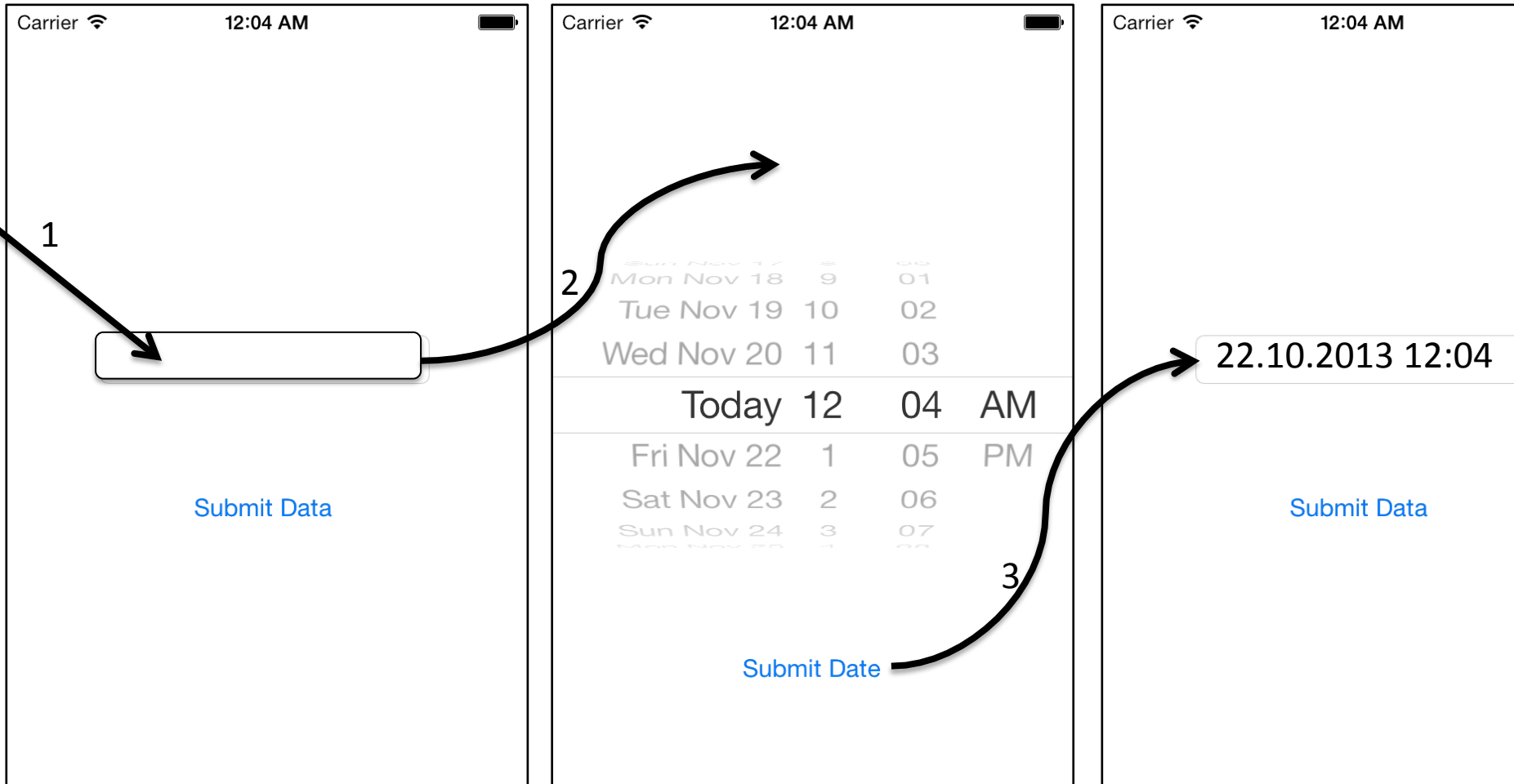
```
@end
```

Erste Aufgabe

EmptyApplication Projekt:

- Erster View mit einfachen Eingaben
 - Textfeld öffnet zweiten View
 - Senden Button
- Zweiter View enthält UIDatePicker
 - benachrichtigt ersten View über Datum
- erster View zeigt Datum in Textfeld
 - Senden Button gibt Datum auf Konsole aus

Erste Aufgabe



Nützliche Infos

```
[self presentViewController:CONT animated:YES completion:NULL]; //zeigt Controller  
[self dismissViewControllerAnimated:YES completion:NULL]; //versteckt Controller
```

```
@property UIPickerView* datepicker;  
NSDate* date = datepicker.date; // returns currently selected date, or use [datepicker date];
```

```
NSLog(NSStringFormattedString, Params...) // C-Funktion mit Wildcards  
NSLog( @"%@ %d", @"string", 5 ); // displays string 5
```

%@ für Strings

%d für integer

....