Aufgabe 1: Kaltes Wasser

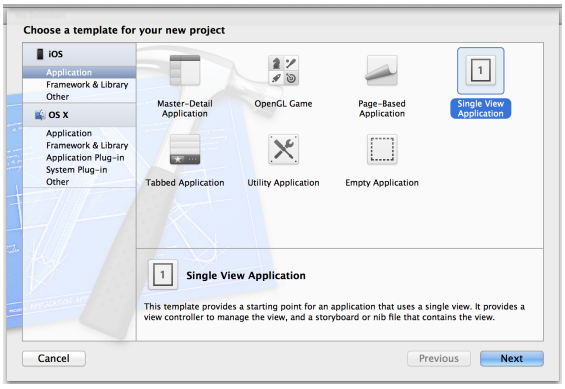
Ziel

Das Ziel dieser Aufgabe ist, Xcode und die Programmiersprache Objective-C praktisch kennen zu lernen. Kein fundiertes Wissen wird das Ergebnis sein sondern ein erstes “Hallo”, damit man bei den nachfolgenden Themen weiss, um was es überhaupt geht.

Aufgabe

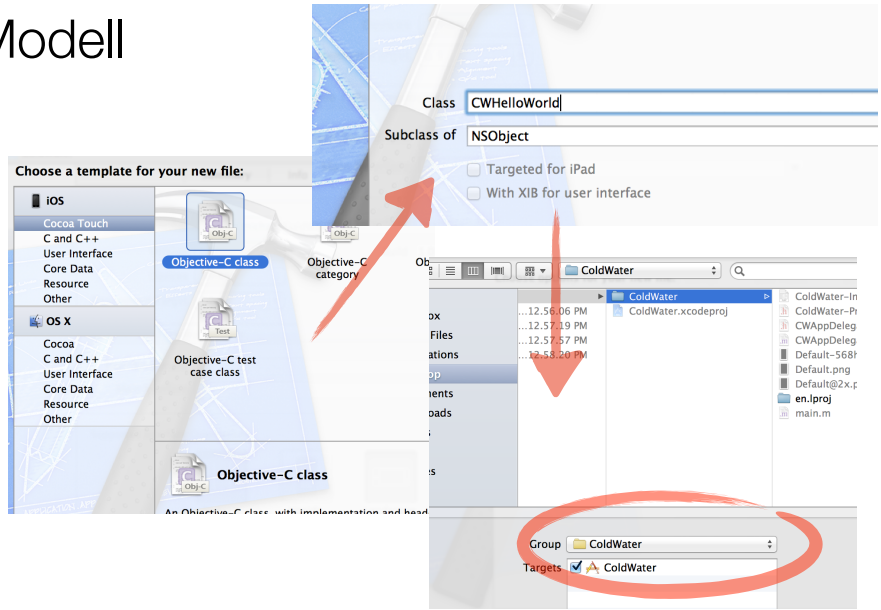
Bitte folgen Sie den nachfolgenden Aufgaben so gut es geht und probieren Sie, mit Hilfe der Dokumentation und Ihrer Nachbarn sie zu lösen. Es kann sein, dass Sie nicht alles direkt verstehen ... keine Angst: Es wird alles noch ausführlich erklärt.

Xcode starten und Projekt erstellen



- Xcode starten
- *File* > *New* > *Project...*
- Neues Projekt *Single View Application* erstellen
- Product Name: ColdWater
- Organisation Name: FHNW
- Company Identifier: ch.fhnw.<Ihr-Kürzel>
- Class Prefix: CW
- Devices: iPhone
- Check: Use Storyboards
- Check: Use Automatic Reference Counting
- Projekt an einem beliebigen Ort erstellen/speichern.

Erstellen einer ersten Modell Klasse



- *File* > *New* > *File...*
- Objective-C class auswählen und die Klasse “CWHelloWorld” nennen und von `NSObject` ableiten.
- Klasse soll im Ordner *ColdWater* und in der Group *ColdWater* abgelegt werden

- Im Interface (***CWHelloWorld.h***) soll die Methode `-sayHello` angelegt werden, welche eine Referenz auf `NSString` zurück gibt.

```
@interface CWHelloWorld : NSObject
- (NSString*)sayHello;
@end
```

- In der Implementation (***CWHelloWorld.m***) implementieren wir nun die Methode, so dass sie den Text “Hello World” zurückgibt und zusätzlich ins Log schreibt, dass sie aufgerufen wurde:

```
@implementation CWHelloWorld

- (NSString*)sayHello {
    return @"Hello World";
}

@end
```

-

Erstellen des Controllers für unsere erste View

- Im Interface (***CWViewController.h***) der bestehenden Klasse *CWViewController* zwei *Properties* erstellen. Eines für die Referenz für das Label in der View, das zweite für unsere Modell Klasse *CWHelloWorld*. (Tipp: Interface von *CWHelloWorld* muss zuerst importiert werden, damit diese Klasse bekannt ist.

```
#import <UIKit/UIKit.h>
#import "CWHelloWorld.h"

@interface CWHelloWorldViewController : UIViewController

@property IBOutlet UILabel* label;
@property CWHelloWorld* model;

@end
```

- In der Initialisierungsmethode des Controllers `-viewDidLoad`, erstellen wir eine Instanz unserer Modellklasse CWHelloWorld (CWHelloWorldViewController.m):

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    _model = [[CWHelloWorld alloc] init];
}
```

-

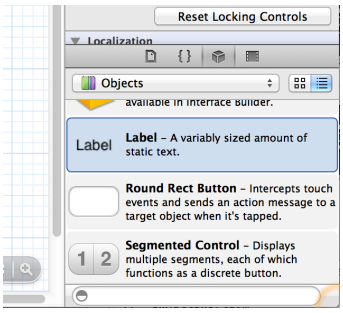
- Direkt nach der Methode `–viewDidLoad` fügen wir eine neue Methode `–sayHello:` hinzu, welche als *Callback* aus der View dient, um nun effektiv etwas anzuzeigen:

```
    // Do any additional setup after loading the view.
    _model = [[CWHelloWorld alloc] init];
}

- (IBAction)sayHello:(id)sender {

    if ([self.label.text isEqualToString:@""]) {
        self.label.text = [self.model sayHello];
    } else {
        self.label.text = @"";
    }
}
```

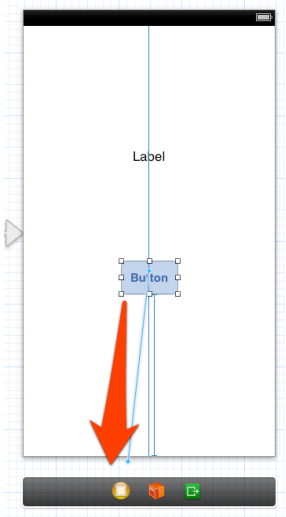
-

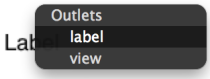
Erste View erstellen

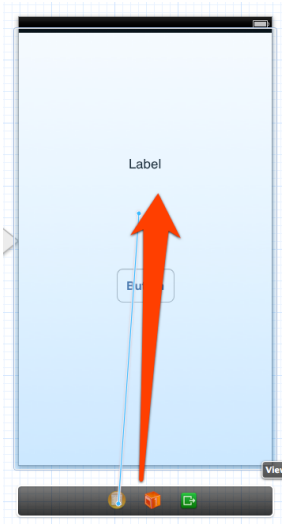
- Alles was wir jetzt noch brauchen, ist im Interface Builder ein Label und ein Button auf der View zu setzen.
- Öffnen Sie dazu das Storyboard: Datei *MainStoryboard.storyboard.*
- Fügen Sie aus der Objektliste rechts unten per Drag&Drop ein *Label* und ein *Round Rect Button* auf die bestehende View hinzu.

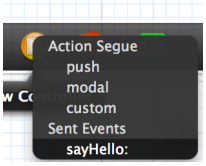


- Danach verbinden wir die View Elemente mit unseren entsprechenden Implementationen im Controller:
- *Ctrl* drücken und eine blaue Verbindungslinie ziehen mit einen Drag&Drop vom ViewController Symbol ( ) unter der View auf das Label. In der erscheinenden Liste `label` auswählen.









•Danach das gleiche in die umgekehrte Richtung vom Button auf das ViewController Symbol ( ). In der erscheinenden Liste unsere `–sayHello` Methode auswählen.

Applikation starten

- Bitte nun mit den Simulator mit *Run* starten und unseren Button drücken.

Fazit

Was haben wir nun gesehen:

- Wie ein neues Projekt in Xcode erstellt wird
- Wie Klassen erstellt, implementiert und instanziert werden
- Wie Properties erstellt werden und wie man auf sie zugreift
- Wie Methoden aufgerufen werden (Messages versandt werden)
- Wie GUIs mit dem Interface Builder erstellt werden und wie man sie mit der Implementation einbindet
- Wie das MVC Pattern mit CocoaTouch interpretiert wird