Übungsblatt 1

Abgabe

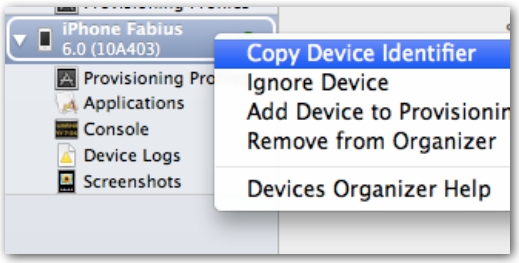
Geben Sie Ihre Lösung zu Aufgabe 4 bis zum 24.10.2012 12:00 Uhr in Uniworx ab. Alle Aufgaben sind einzeln zu bearbeiten.

Aufgabe 1

Installieren Sie die neueste Version von XCode über den Mac App Store und laden Sie ggf. Updates herunter, um die neuesten Versionen von APIs, Dokumentation und Simulator verwenden zu können.

Registrieren Sie sich im iOS Developer Center und schalten Sie Ihr iOS-Gerät zum Testen frei. Schicken Sie dazu bitte eine Email mit dem Betreff "PEM iOS Developer Center" an steinberger @ cip.ifi.lmu.de mit:

- Email-Adresse, mit der Sie als Developer registriert sein möchten (z.B. Ihre Apple-ID).
- Device-Identifier Ihres iOS-Geräts. Diese lässt sich über den XCode Organizer (siehe Abbildung) oder iTunes herausfinden, wenn das Gerät am Computer angeschlossen ist.



Wir laden Sie dann zum iOS Developer Center ein und schicken Ihnen ein Provisioning Profile, das es Ihnen ermöglicht, eigene Apps auf Ihrem iOS-Gerät zu testen.

> **Hinweis:** Die folgenden Aufgaben können auch ohne Developer Account gelöst werden. Für dieses Übungsblatt können Sie Ihre App im Simulator testen.

Aufgabe 2

Stellen Sie nächste Woche im Kurs kurz (ca. 1 Minute) eine Ihrer Lieblings-Apps vor. Gehen Sie dabei z.B. auf Zielgruppe, Funktionalität, Use Cases und Design ein.

Aufgabe 3

Unter http://goo.gl/P0A8r steht ein gemeinsames Google Doc bereit. Erstellen Sie dort gemeinsam ein Cheat Sheet für Objective-C. Tragen Sie sich außerdem in die Tips & Tricks Tabelle ein, um Ihren Kommilitonen einen Tip zur iOS Programmierung zu geben (z.B. zu NSLog, XCode Shortcuts...). Zunächst brauchen dies nur zwei Kursteilnehmer für nächste Woche machen (FCFS).

Aufgabe 4

Erstellen Sie eine erste iPhone App anhand der folgenden Schritte. Hierbei geht es in erster Linie darum, die Handhabung von XCode kennenzulernen und grundlegende Möglichkeiten der GUI-Programmierung unter iOS kennenzulernen.

> **Hinweis:** Seit XCode 4.4 (Juli 2012) gibt es einige Neuerungen in der Objective-C Programmierung. Achten Sie bitte auf folgende Dinge, falls Sie mit Tutorials arbeiten, die älter sind:
>
> - Properties: Neben @property in .h war früher zusätzlich ein @synthesize in .m notwendig, damit automatisch Getter/Setter erstellt werden. Dies geschieht nun automatisch durch den Compiler (Auto-Synthesis).
> - Syntax: Genauso wie man NSStrings durch @"Hello World" initialisieren kann, können nun auch NSNumbers, NSDictionaries und NSArrays erstellt werden (z.B. @["Yellow", "Green", "Red"]). Siehe http://mobile.tutsplus.com/tutorials/iphone/objective-c-literals/.
> - Methoden: Die Reihenfolge, in der Methoden in .m implementiert werden, ist nun egal.

- Erstellen Sie ein neues XCode-Projekt namens HelloWorld vom Typ Single View Application. Verwenden Sie Storyboards und ARC.
- Der Navigator auf der linken Seite zeigt die verschiedenen Dateien, die zum Projekt gehören. Gehen Sie diese Dateien durch und versuchen Sie deren Funktion zu verstehen.
- Öffnen Sie das Storyboard, dort finden Sie eine Scene (ein Displayinhalt). Markieren Sie die Scene und klicken Sie links auf den Identity Inspector. Dort sehen Sie, dass es sich hier um die Klasse ViewController handelt, die im Navigator sichtbar ist.
- Fügen Sie der Scene einen Round Rect Button hinzu, indem Sie ihn aus der Object Library unten links ziehen. Fügen Sie außerdem ein Label hinzu und setzen Sie den Text auf "ViewController".
- Ziehen Sie einen weiteren View Controller auf das Storyboard. Erstellen Sie dafür eine neue Klasse (File > New > New File) namens SecondViewController abgeleitet von UIViewController. Wechseln Sie wieder zum Storyboard, markieren Sie die neue Scene und ändern Sie im Identity Inspector die Klasse zu SecondViewController.
- Ziehen Sie ein Label ("SecondViewController") auf die zweite Szene.
- Verknüpfen Sie per Ctrl-Drag den zuvor erstellten Button mit der zweiten Scene (Modal). Die Verknüpfung zwischen zwei Scenes wird als Segue bezeichnet.
- Lassen Sie die Anwendung nun im Simulator laufen. Durch den Button müsste man nun zur zweiten Scene gelangen.
- Ziehen Sie einen Round Rect Button auf die zweite Scene und erstellen Sie eine Modal-Segue, so dass man über den Button zur ersten Scene gelangt.
- Ziehen Sie auf die zweite Scene ein weiteres Label sowie einen Stepper. Das Label soll zu Beginn 0 anzeigen, der Wert soll über den Stepper verändert werden. Gehen Sie dazu wie folgt vor.

- Blenden Sie den Assistant Editor ein, dieser zeigt den Code von SecondViewController.h an. Führen Sie ein Ctrl-Drag vom neuen Label zum Code durch und erstellen Sie dadurch ein Outlet für das Label. Erstellen Sie auf die gleiche Weise ein Outlet sowie eine Action (Value Changed) für den Stepper.

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

@interface SecondViewController : UIViewController

// UI elements
@property (weak, nonatomic) IBOutlet UILabel *stepperLabel;
@property (weak, nonatomic) IBOutlet UIStepper *stepper;

// UI actions
- (IBAction)stepperValueChanged:(id)sender;

@end
```

- Wechseln Sie zu SecondViewController.m und erzeugen Sie in der IBAction-Methode eine Logausgabe mit NSLog. Setzen Sie in der gleichen Methode den Labeltext auf den Stepper-Value. Benutzen Sie dafür die NSString-Methode stringWithFormat. Lassen Sie jetzt Ihre Anwendung laufen.
- **Für Master-Studierende:** Fügen Sie Ihrer Abgabe eine Master.txt hinzu. Erstellen Sie eine dritte Scene und verwenden Sie ein weiteres Label sowie ein weiteres UI Element Ihrer Wahl (z.b. Slider). Erstellen Sie diesmal kein Outlet für das Label, sondern vergeben Sie ein Tag für das Label im Attributes Inspector. Der Zugriff auf das Label erfolgt dann über UILabel *label = (UILabel*) [self.view viewWithTag:100];