

Softwareentwicklung für iOS mit Objective-C und Xcode

App Katalog

Nils Fischer

Universität Heidelberg – Sommersemester 2014

Inhaltsverzeichnis

1	Über dieses Dokument	3
2	Hello World	4
2.1	Das erste Xcode Projekt	4
2.2	@ "Hello World!"	5
2.3	@ "Hello World!" on Device	7
2.4	Grundlagen der Programmierung	7

1 Über dieses Dokument

Dieser App Katalog enthält Schritt-für-Schritt Anleitungen für die im Rahmen unseres Kurses erstellten Apps sowie die wöchentlich zu bearbeitenden Übungsaufgaben und wird im Verlauf des Semesters kapitelweise auf der Vorlesungsseite ^[1] zur Verfügung gestellt.

Er dient jedoch nur als Ergänzung zum parallel verfügbaren **Skript**, auf das hier häufig verwiesen wird. Dort sind die Erläuterungen zu den verwendeten Technologien, Methoden und Begriffen zu finden.

Beispiellösungen zu den Übungsaufgaben sind ebenfalls auf der Vorlesungsseite zu finden.

¹<http://ios-dev-kurs.github.io/>

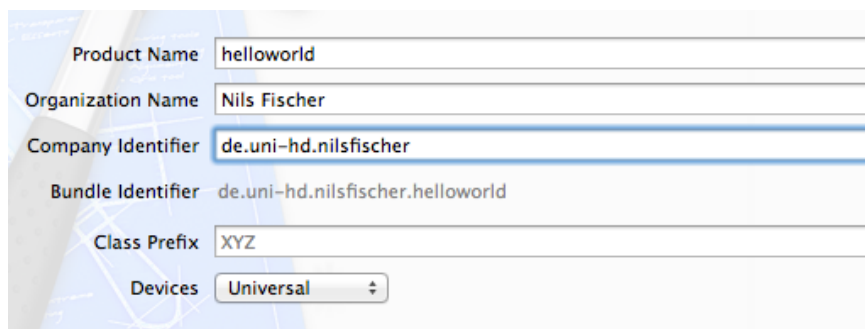
2 Hello World

Was ist schon ein Programmierkurs, der nicht mit einem klassischen *Hello World* Programm beginnt? Wir werden jedoch noch einen Schritt weitergehen und diesen Gruß vom iOS Simulator oder, soweit vorhanden, direkt von unseren eigenen iOS Geräten ausgeben lassen. Außerdem wird in die objektorientierte Programmierung eingeführt.

Relevante Kapitel im Skript: Xcode, Objective-C

2.1 Das erste Xcode Projekt

1. Mit `⌘ + ⌥ + N` rufen wir zunächst den Dialog zur Erstellung eines neuen Projekts auf und wählen das Template `iOS > Application > Single View Application`.
2. Tragt im erscheinenden Konfigurationsdialog entsprechend der Konventionen den Product Name **helloworld**, euren Vor- und Nachnamen als Organization Name und **de.uni-hd.deinname** als Company Identifier ein (s. S. 4, Abb. 2.1). Das führt zu der Bundle ID **de.uni-hd.deinname.helloworld**. Einen Class Prefix benötigen wir erstmal nicht. Speichert das Projekt in einem Verzeichnis eurer Wahl.



The image shows the 'New Project' dialog in Xcode. The 'Product Name' field is set to 'helloworld'. The 'Organization Name' field is set to 'Nils Fischer'. The 'Company Identifier' field is set to 'de.uni-hd.nilsfischer' and is highlighted with a blue border. The 'Bundle Identifier' field shows the generated value 'de.uni-hd.nilsfischer.helloworld'. The 'Class Prefix' field is set to 'XYZ'. The 'Devices' dropdown menu is set to 'Universal'.

Abbildung 2.1: Damit es keine Konflikte zwischen verschiedenen Apps gibt, gibt es Konventionen bei der Konfiguration

3. Wir sehen nun Xcodes Benutzeroberfläche und können sie mit den Schaltflächen rechts in der Toolbar anpassen. Verwendet zunächst die Konfiguration mit eingblendetem Navigator, verstecktem Debug-Bereich und Inspektor und Standard-Editor. Wählt im Project Navigator das Projekt selbst aus (s. S. 5, Abb. 2.2).

2 Hello World

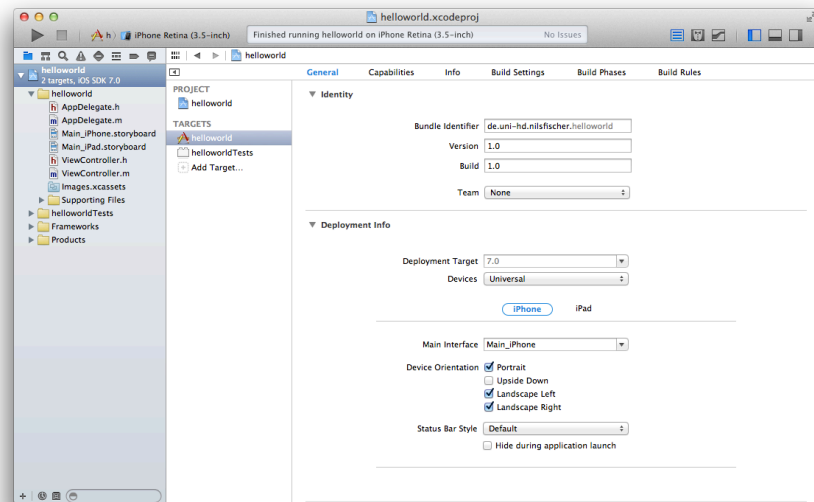






Abbildung 2.2: Wird das Projekt ausgewählt, sehen wir im Editor die Projekt- und Targetkonfiguration.

4. Im Editor wird die Projekt- und Targetkonfiguration angezeigt. Hier können wir bspw. die Bundle ID unserer App anpassen, die wir zuvor bei der Erstellung des Projekts aus Product Name und Company Identifier zusammengesetzt haben.
5. Links in der Toolbar sind die Steuerelemente des Compilers zu finden. Wählt das gerade erstellte Target und ein Zielsystem aus, bspw. den *iPhone Retina (3.5-inch)* Simulator, und klickt die *Build & Run* Schaltfläche. Das Target wird nun kompiliert und generiert ein *Product*, also unserer App, die im Simulator ausgeführt wird. Das kann bei der ersten Ausführung durchaus etwas dauern oder einen Fehler generieren. In Xcode kann mit  +  die Ausführung gestoppt und mit  +  (Tastenkürzel für *Build & Run*) dann neu gestartet werden.

2.2 @"Hello World!"

1. Besonders spannend ist diese App natürlich noch nicht. Das ändern wir jetzt spektakulär, indem wir eine Ausgabe hinzufügen. Wählt die Datei *AppDelegate.m* im Project Navigator aus.
2. Die Methode `application:didFinishLaunchingWithOptions:` wird zu Beginn der Ausführung der App aufgerufen. Zwischen den geschweiften Klammern ist bisher noch nicht viel zu finden:

2 Hello World

```
1 - (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
2     // Override point for customization after application launch.
3     return YES;
4 }
```

3. Ersetzt den Kommentar mit einem Befehl zur Ausgabe von Text in der Konsole:

```
1 - (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
2     NSLog(@"Hello World!"); // Dieser Befehl gibt den Text Hello World! in
        der Konsole aus
3     return YES;
4 }
```

4. Wenn wir unsere App nun erneut mit *Build & Run* kompilieren und ausführen, sehen wir den Text *Hello World!* in der Konsole. Dazu wird der zweigeteilte Debug-Bereich unten automatisch eingeblendet (s. S. 6, Abb. 2.3). Ist der Konsolenbereich zunächst versteckt, kann er mit der Schaltfläche in der rechten unteren Ecke angezeigt werden. Außerdem wird links automatisch zum Debug Navigator gewechselt, wenn eine App ausgeführt wird, in dem CPU- und Speicherauslastung überwacht werden können und Fehler und Warnungen angezeigt werden, wenn welche auftreten.

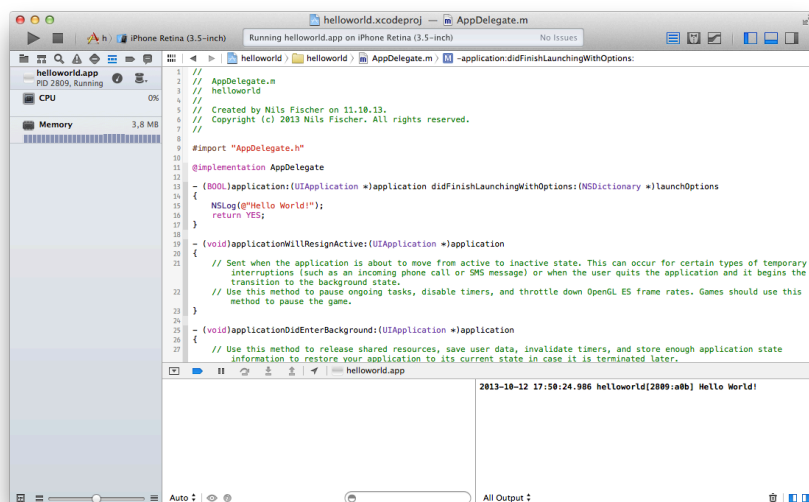


Abbildung 2.3: In der Konsole des Debug-Bereichs werden Ausgaben der laufenden App angezeigt

2.3 @"Hello World!" on Device

1. Nun möchten wir unsere neue App natürlich auch auf einem realen iOS Gerät anstatt des Simulators testen. Im Skript findet ihr eine Anleitung, wie ihr mit euren iOS Geräten unserem Developer Team der Uni Heidelberg beitreten könnt.
2. Habt ihr die Schritte befolgt und euren freigeschalteten Apple Developer Account in den Xcode-Accounteinstellungen hinzugefügt, öffnet ihr wieder die Project- und Targetkonfiguration im Project Navigator und wählt dort unser Developer Team (s. S. 7, Abb. 2.4) aus. Nun wird automatisch das richtige Provisioning Profile für die Bundle ID des Targets verwendet.

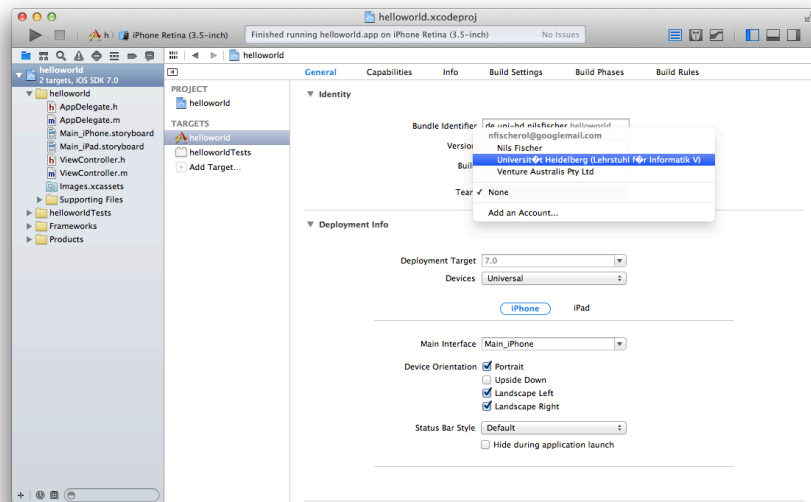


Abbildung 2.4: Mit der Wahl des zugehörigen Developer Teams in der Project- und Targetkonfiguration verwendet Xcode automatisch das passende Provisioning Profile

3. Verbindet euer iOS Gerät mit eurem Mac und wählt es in der Toolbar als Zielsystem aus. Mit einem *Build & Run* wird die App nun kompiliert, auf dem Gerät installiert und ausgeführt. In der Konsole erscheint wieder die Ausgabe *Hello World!*, diesmal direkt vom Gerät ausgegeben.

2.4 Grundlagen der Programmierung

1. Wir können nun beginnen, Objective-C Code zu schreiben. Öffnet dafür wieder die Datei *AppDelegate.m*.

2. In der Methode `application:didFinishLaunchingWithOptions:`, die wir schon zuvor verwendet haben, können wir nun zunächst die Grundlagen der Programmierung wie im Skript beschrieben ausprobieren.

Übungsaufgaben

1. Fibonacci

- a) Schreibt einen Algorithmus, der alle Folgenglieder $F_n < 1000$ der Fibonaccifolge

$$F_n = F_{n-1} + F_{n-2} \quad (2.1)$$

$$F_1 = 1, F_2 = 2 \quad (2.2)$$

in der Konsole ausgibt.

- b) **Extra:** Bei jeder geraden Fibonaccizahl F_j ist der Abstand $\Delta n = j - i$ zum vorherigen geraden Folgenglied F_i auszugeben.

2. Primzahlen

Schreibt einen Algorithmus, der alle Primzahlen $p_n < 1000$ in der Konsole ausgibt.

Hinweis: Mit dem Modulo-Operator `%` kann der Rest der Division zweier Integer gefunden werden:

```
1  int a = 20%3 // a ist jetzt 2
```