Aufgabe 6: TwitterFeed UI

Ziel

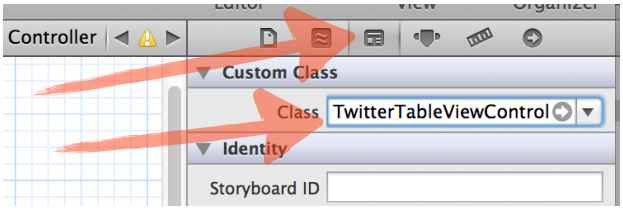
In dieser Aufgabe lernen wir, die gelesenen Twitter Nachrichten über eine Tabelle darzustellen und Details in einer separaten Anzeige darzustellen.

Aufgabe

1. *TwitterFeedUI* Projekt kopieren und öffnen

2. *Storyboard* öffnen, sämtliche Views entfernen, falls vorhanden, und neuen *Table View Controller* von der Objektliste rechts unten auf die Designerfläche ziehen.

3. Im *Identity Inspector* des *TableViewControllers* die bestehende Klasse `TwitterTableViewController` setzen.



4. In der bestehenden Klasse `TwitterTableViewController` fügen wir ein neues Property vom Typ `NSArray` hinzu, welches die gelesenen Tweets beinhaltet.

5. Schauen Sie sich kurz die Klasse TweetService an (inklusive Header Datei) sowie in der Klasse `TwitterTableViewController` die Methode `-viewDidLoad`. Erkennen Sie das Delegate Pattern mit dem dazugehörigen Protokoll?

6. Im Interface des TwitterTableViewControllers müssen wir nun das Protokoll vom TweetService deklarieren.

7. In der Klasse `TwitterTableViewController` müssen wir nun die Delegate Methode – `(void)retrievedTweets:(NSArray *)tweets` implementieren. Diese Methode weist das empfangene Array der Instanzvariable für die Tweets zu und informiert anschliessend der Tabelle mit, dass sie neue Daten hat und sich aktualisieren soll.

```
[self.tableView reloadData]; //Nur im Main Thread erlaubt

[self.tableView
     performSelectorOnMainThread:@selector(reloadData)
                      withObject:nil
                   waitUntilDone:NO];
```

   Um die Tabelle zu aktualisieren.

8. Nun müssen wir die Methode –`numberOfRowsInSection` implementieren. Diese soll die Anzahl Tweets zurückgeben.

9. Um nun in der Zelle wirklich etwas darzustellen, müssen wir die Methode -cellForRowAtIndexPath implementieren. Wichtig ist hier zu beachten, dass der *Identifier* mit jenem im *Interface Builder*

zusammenpasst.
Tipp: Der Zeilenindex ist im Object `indexPath` als Property `row` vorhanden.

10. Jetzt können Sie die Applikation im Simulator starten und sollten, nach einer kurzen Wartezeit, eine Liste mit Tweets sehen.

Navigation Controller und Segue zu Detailansicht

11. Im Storyboard ziehen wir nun einen *Navigation Controller* hinein. Den automatisch erstellten Table View Controller können wir löschen, da wir ja bereits einen eigenen haben. Verbinden Sie diese beiden mit Ctrl und Drag&Drop. Im Dialog den Segue Typ *Relationship 'root view controller'* auswählen.

12. Ziehen Sie nun für die Detailansicht einen leeren *View Controller* hinein. Setzen Sie im *Identity Inspector* dieses View Controllers die bestehende Klasse `TweetViewController`.

13. Erstellen Sie einen "push" *Segue* zwischen der Zelle und unserer neuen Detail View. Setzen Sie als *Identifier* auf diesem *Segue* "showDetail"

14. Auf der Detailview brauchen wir nun zwei Elemente für den Autor und den Text. Zum Beispiel ein *Label* und eine *Text View*. Designen Sie die View entsprechend.

15. Erstellen Sie im `TweetViewController` zwei *Outlets* für die Elemente und verbinden Sie diese im *Interface Builder* entsprechend.

16. Weisen Sie in der Methode `–viewDidLoad` in der Klasse `TweetViewController` den View Elementen die entsprechenden Werte zu (Tipp: Es existiert bereits ein *Property* für ein Tweet-Objekt).

17. Was nun noch fehlt, ist, dass beim Auswählen einer Zelle das entsprechende Objekt an die Detailansicht weitergereicht wird. Dies geschieht in der Klasse `TwitterTableViewController` in der Methode `–prepareForSegue:sender:`. Der Code existiert bereits und muss nur noch einkommentiert werden.

18. Wenn Sie nun wieder die Applikation im Simulator starten, können Sie nun eine Zeile auswählen und in der Detailansicht die komplette Nachricht sehen.

Tipps:

- Um ein Element aus einem Array zu erhalten verwenden Sie die Methode `–objectAtIndex:`

```
[self.arrayReferenz objectAtIndex:indexPath.row];
```

- Tabellen-Zellen, mit einem Standard Style, können über die Properties `detailTextLabel` und `textLabel` gefüllt werden.
- Bei Zellen mit einem Custom Style müssen die Elemente über die Methode `–viewWithTag:` gefunden werden. Beachten Sie, dass die Tags mit jenen im Interface Builder dieser Elemente übereinstimmen müssen.

Optionale Erweiterungen

Falls Sie Lust haben, können Sie folgende Erweiterungen probieren:

- Momentan werden die Daten nur einmal beim Start geladen. Gut wäre ein Button um die Daten zu laden oder per "Scroll-Down" die Daten zu aktualisieren. (Siehe dazu *Bar Button Items* oder `UIRefreshControl`)
- Man sieht nicht, ob im Hintergrund etwas am Laufen ist oder nicht. Ein Progress Indicator wäre sehr hilfreich.
- Könnte die Applikation nicht noch etwas schöner aussehen?

- Momentan werden nur Autor und Text der Nachricht ausgelesen und dargestellt. Wie wäre es mit Erstellungsdatum, weiteren Tags, Profile-Bild?

Fazit

Folgende Punkte haben wir gelernt:

- Wir haben gesehen, wie mit einem Delegate Pattern die Daten aus dem Modell gelesen werden.
- Wir haben unser erstes komplettes Storyboard erzeugt mit zwei *Scenes*, die in einer Navigation eingebettet sind.
- Wir haben *Segues* erstellt und gesehen, wie sie funktionieren.
- Wir haben gesehen, wie das MVC Pattern in iOS konkret umgesetzt wird.