Assignment 3: Game Shop

Ziele

Im dritten Assignment lernen Sie die Implementierung von Table Views und deren Delegates kennen. Dies soll am Beispiel eines kleinen Shopsystems zum Kauf von Spielen erfolgen.

Weiterhin sollen Sie den Umgang mit Datenmodellen lernen. Hierzu wird im zweiten Teil der Aufgabe die Datenstruktur für die Objekt-Orientierung vorbereitet und verwendet.

Lesen Sie zuerst die Aufgabenstellung komplett durch, verstehen Sie die darin geforderten Anforderungen, planen Sie danach Ihre Vorgehensweise und machen sich danach an die Implementierung!

Hinweis: Der erste Teil des Praktikums lässt sich durch das schreiben von weniger als 40 Zeilen Code lösen. Ggf. brauchen Sie etwas mehr, sofern Sie ein paar Zwischenschritte einfügen. Sollten Sie deutlich mehr als 60 Zeilen Code brauchen, überdenken Sie Ihren Ansatz noch einmal!

Im zweiten Teil fügen Sie wenige Zeilen Code in Form einer Klasse hinzu und ersetzten einen Teil des Codes aus dem ersten Teil der Aufgabe, so dass sich das komplette Praktikum mit insgesamt weniger als 40 Zeilen Code lösen lässt.

Materialien

- Zusammen mit der Praktikumsaufgabe erhalten Sie das Datenmodell in Form eines plist-Files und die notwendigen Resourcen in Form von Bildern. Es steht Ihnen frei selbst einen eigenen Warenkorb mit Produkten zu erstellen und diesen für das Praktikum zu verwenden. Voraussetzung ist allerdings dass Ihr Table View am Ende genug Items beinhaltet um scrollbar zu sein. Die Aufgabenbeschreibung bezieht sich auf das zur Verfügung gestellte Datenmodell und Resourcen.

- Sprechen Sie mit Ihren Kommilitonen, ggf. ist einer Ihrer Kommilitonen auf das gleiche oder ein ähnliches Problem gestoßen und kann Ihnen behilflich sein. Sie sollten keinen Code kopieren, stattdessen zusammen ein Problem verstehen, dafür eine Lösung finden und diese dann in Ihrer Praktikumsgruppe implementieren.

- In jedem Fall sollten Sie den Code den Sie schreiben, auch vollständig verstanden haben. Rechnen Sie damit, dass Sie bei einem Testat detailiert nach Ihrer Implementierung gefragt werden.

Aufgaben

1. Legen Sie sich ein neues iPhone Projekt an. Löschen Sie den vorhandenen Default View Controller im *Project Navigator* und ebenfalls im *Storyboard*. Ziehen Sie nun einen `TableViewController` aus der *Object Library* in das *Storyboard* und verwenden diesen als initialen View Controller.
2. Erstellen Sie die neue Klasse `GameShopTableViewController` als Subclass von `UITableViewController`. Setzen Sie Ihre neue Klasse für den Table View Controller im Storyboard.
3. Erstellen Sie eine weitere neue Klasse `GameTableViewCell` als Subclass von `UITableViewCell`. Verwenden Sie als *Table View Cell Style* Ihres Table Views im *Storyboard* den Typ `Custom` und vergeben Sie an die Prototype Cell einen eindeutigen *Identifier*.

   Die Klasse hat vier Outlets: `title`, `system`, `price` vom Type `UILabel` und `cover` vom Typ `UIImageView`. Diese werden Sie im nächsten Schritt in der Prototype Cell designen und dann als Outlet mit Ihrer Klasse verbinden.

   Vergessen Sie nicht die Klasse Ihrer Prototype Cell auf `GameTableViewCell` zu setzen!
4. Designen Sie nun eine Custom Cell im Storyboard. Halten Sie sich dabei grob an die Vorgabe des Screenshots weiter unten. Sie brauchen dafür:

   - `UIImageView` zum anzeigen des Covers
   - "großes" `UILabel` zum anzeigen des Titels
   - "kleines" `UILabel` zum anzeigen des Systems
   - "kleines" `UILabel` zum anzeigen des Preises

   Wahrscheinlich wollen Sie die Höhe der Cell modifizieren. 78 Points haben sich dafür als praktikabel erwiesen, wobei die tatsächliche Höhe auch vom verwendeten Device abhängt. Generell gilt, Ihre Cell sollte optisch ansprechend designed sein.

   Vergessen Sie nicht die Verbindung der UI-Komponenten als Outlets in Ihrer Klasse!
5. Als nächstes fügen Sie das File *Data.plist* und die Cover Bilder im jpg-Format zu Ihrem Xcode Projekt hinzu. Sie können die Files einfach über den *Finder* direkt in das Xcode-Projekt (*Projekt Navigator*) ziehen. Setzen Sie die Option *copy items if needed* um die Dateien tatsächlich zu kopieren und nicht nur zu referenzieren.
6. Schauen Sie sich in Xcode den Aufbau des plist-Files an. Darin enthalten sind alle Artikel, jedoch in einem nicht sehr Objekt-Orientierten Format. Für den ersten Teil der Aufgabe ist es ok diese Daten als vier Arrays eines geeigneten Typs im ViewController zu speichern (der Zugriff auf den Index i in jeder der vier Arrays ergibt dann die Information eines kompletten Artikels).

   Suchen Sie sich einen geeigneten Ort um die Daten aus dem plist-File in die Arrays zu speichern. Um das plist-File zu lesen, brauchen Sie zuerst einen Pfad zum File. Dies erhalten Sie mittels der Funktion `func pathForResource(name: String?, ofType ext: String?) -> String?` aus dem `mainBundle` (Klasse `NSBundle`).

   Sobald Sie den Pfad erhalten haben, können Sie mittels der Klasse `NSDictionary` direkt den Inhalt von `Data.plist` in ein Dictionary einlesen. Nutzen Sie dafür den Convenience Initializer der Klasse: `NSDictionary(contentsOfFile path: String)`. In Swift wollen wir aber mit Swift-Dictionaries arbeiten und nicht mir der Klasse `NSDictionary`. Casten Sie also das `NSDictionary` in ein Swift Dictionary mittels des Casting Operators `as`. Achten Sie auf den richtigen Typ des Dictionaries! Die Datentypen sind aus dem plist-File ersichtlich.

Aufgaben fortgesetzt auf nächster Seite...

7. Nachdem Sie die Arrays mit Daten befüllt haben, können Sie nun den Table View füllen. Da Ihre Subclass die Protokoll-konformität zu Table Views mitbringt, implementieren Sie die benötigten Funktionen zum befüllen des Table Views.

   - `numberOfSectionsInTableView(tv) -> Int`
   - `tableView(tv, numberOfRowsInSection: Int) -> Int`
   - `tableView(tv, heightForRowAtIndexPath: NSIndexPath) -> CGFloat`
   - `tableView(tv, cellForRowAtIndexPath: NSIndexPath) -> UITableViewCell`

   Beachten Sie, dass Sie `title`, `system` und `price` einer Cell mit einem String, das `cover` dagegen mit einem `UIImage` befüllen. Das Bild müssen Sie mittels des Bildnamens (String des `cover`) laden.

   Hinweis: Zum jetzigen Zeitpunkt sollten Ihnen alle Artikel im Table View angezeigt werden. Ist das nicht der Fall, machen Sie nicht mit der Aufgabe weiter, sondern finden Sie die Fehler in Ihrer Implementierung, korrigieren Sie diese und machen erst weiter, wenn alle Artikel im Table View angezeigt werden.

8. Sie müssen nun noch dafür sorgen, dass ein Artikel zum Warenkorb hinzugefügt werden kann. Dies realisieren Sie durch "antippen" der Cell mit dem Finger. Ein Artikel ist dann im Warenkorb, wenn rechts in der Cell ein kleiner Haken zu sehen ist. Dies erreichen Sie durch setzen des `accessoryType` einer Cell auf `UITableViewCellAccessoryType.Checkmark`. Wird eine Cell erneut "angetippt", wird der Artikel wieder entfernt (`UITableViewCellAccessoryType.None`). Als letztes soll eine Cell niemals permanent `selected` (grau hinterlegt) sein. Die Klasse `UITableView` bietet dafür die Funktion `deselectRowAtIndexPath` an.

   Die komplette Funktionalität für diesen Aufgabenpunkt können Sie in der Funktion `tableView(tv, didSelectRowAtIndexPath: NSIndexPath)` implementieren.

9. Wenn Sie Ihre App starten, dann fällt Ihnen sicherlich auf, dass die erste Zelle in den System Bar gezeichnet wird. Dies wollen Sie für den Start wahrscheinlich nicht haben. Stattdessen soll die Cell 20 Points weiter unten gezeichnet werden (der System Bar ist 20 Points hoch). Dies können Sie, nachdem der View geladen wurde, mittels `self.tableView.contentInset = UIEdgeInsetsMake(20, 0, 0, 0)` erreichen.

   Die Funktionalität ist nun vollständig implementiert, ein guter Zeitpunkt also um sich den ersten Teil der Aufgabe testieren zu lassen. Die folgenden Punkte beschränken sich nur noch auf die Modifikation des Datenmodells, die eigentliche Funktion wird nicht mehr modifiziert.

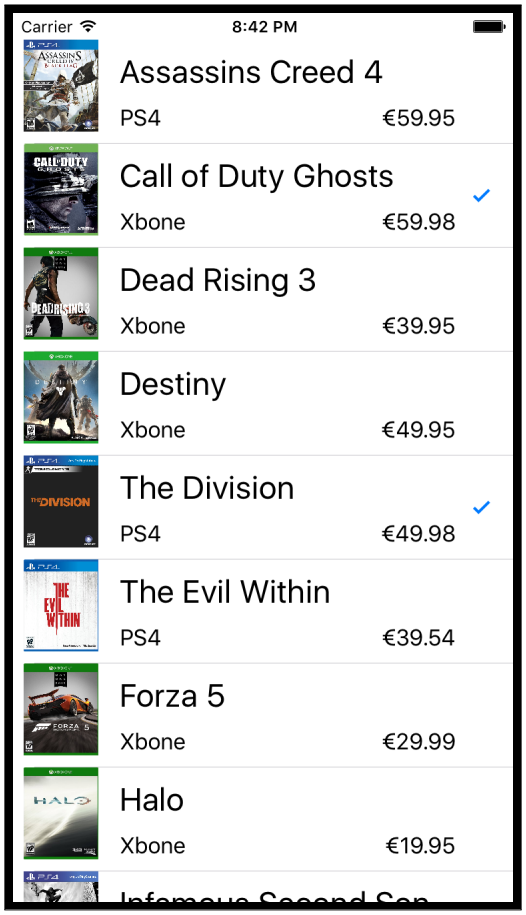
10. Erstellen Sie eine neue Klasse `Game` mit den Properties `title`, `price`, `system` und `cover`, alle vom Typ `String` und einen dazu passenden Initializer. Das Property `cover` beinhaltet lediglich den Namen des Bildes welches später geladen wird, nicht das Bild selbst. Die Properties haben einen public getter, aber einen private setter.

11. Statt die Informationen aus dem Dictionary nun in vier verschiedene Arrays zu speichern, speichern Sie die Informationen in einem einzigen Array, welches Objekte vom Typ `Game` beinhaltet. Dazu müssen Sie in geeigneter Weise über das Array iterieren, zu jedem Artikel ein `Game`-Objekt erstellen und dieses dem Array hinzufügen. Dies kann in weniger als 5 Zeilen Code erledigt werden.

12. Passen Sie das Befüllen der Cells im Table View an Ihr neues Datenmodell an. Beziehen Sie die Informationen also nicht aus vier verschiedenen Arrays, sondern aus einem `Game`-Objekt welches aus dem Array kommt.

Screenshots

Orientieren Sie sich beim Design Ihrer App an folgendem Screenshot.



Hinweise

Die größten Probleme werden Sie wahrscheinlich mit dem Identifizieren und bearbeiten der Datenstruktur aus dem plist-File haben. Es ist essentiell diese Datenstruktur zu verstehen und den Zugriff darauf korrekt anzuwenden. Es werden dafür nur Strings, Arrays und ein Dictionary (Key-/Value Pair) verwendet. Dies ist bereits aus PAD bekannt, kann aber in der Kombination miteinander vielleicht Anfangs etwas verwirrend sein. Nehmen Sie sich ruhig ein Blatt Papier und zeichnen sich die Datenstruktur auf.

Alternativ zur in der Aufgabenstellung gegebenen Vorgehensweise, können Sie Ihren Table View auch zuerst mit Dummy-Daten befüllen (jede Cell zeigt die gleichen Daten an). Auf diese Weise können Sie sicherstellen, dass alle Protokollfunktionen richtig implementiert sind.

Nutzen Sie den Debugger und `print`, um den Inhalt Ihrer Variablen/Konstanten auf der Konsole auszugeben. So können Sie sicherstellen, dass Sie tatsächlich die richtigen Daten erhalten. `print` funktioniert auch für Arrays und Dictionaries!

Links

- The Swift Programming Language
- UITableViewController Class Reference
- UITableView Class Reference
- UITableViewCell Class Reference
- UIImage Class Reference
- UIImageView Class Reference
- NSDictionary Class Reference