Assignment 4

UpcomingMovies: Tables, View Navigation, NSData + JSON, Web View, GCD

Due: November 23rd, 2015. 9:00 AM **Group size:** 2

Description
In this assignment, you will learn how to design a multi-view application, get data from a website and parse it, and finally manage your threads.

Task
Part 1: Create UpcomingMovies app

Many movie management websites provide users access to their databases. In this app you will get data about upcoming movies from **themoviedb.org**. You will show that movies in a table view and provide users with additional information in a detail view.

1. Create a project from the Master-Detail application template (`Universal`, keep `Use Size Classes` enabled)
2. We want to have two tabs: “upcoming" and “favourite”. Embed the navigation controller that is connected to the table in a tab bar controller (Hint: select the navigation controller then from Editor > Embed >Tab Bar Controller)
3. Select the same navigation controller (the one connected with the tab bar controller and table) and in the Identity Inspector give it a Storyboard ID: NavigationController
4. Clean your MasterViewController. Remove methods: commitEditingStyle; canEditRowAtIndexPath; insertNewObject and didReceiveMemoryWarning. Remove all occurrences of “as! NSDate”. Remove everything in viewDidLoad but super.viewDidLoad( )
5. Convert the variable objects in MasterViewController to an array of dictionaries that take string values and keys.

Part 2: Getting the data

1. Add the JSON helper file “SwiftyJSON.swift" to your project. This file helps parse the JSON
2. Log in to **themoviedb.org** and sign up to get an API key. You will need this key to access the database. Look at the data structure of the json (simple paste the url on your browser).
3. Use the following url: http://api.themoviedb.org/3/movie/upcoming?api_key= + APIKey to get the JSON data. Use the helping code:
   3.1. NSURL(string: urlPath)

3.2. data = NSData(contentsOfURL: url)

3.3. json = JSON(data: data)

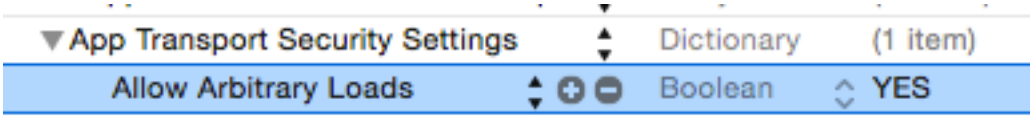
4. Parse the json object . Use the helping code:

   4.1. for result in json[“results"].arrayValue

   4.2. let title = result[“title"].stringValue

5. We want to obtain the following information from the json: title, overview, release_date, poster_path, vote_average

6. If you run your app Xcode will issue an excepting because it doesn't trust your domain. A brut force solution will be to add the following key to your project’s plist:



7. Make sure you catch all errors possible when getting the data online. Use Alerts.

8. Challenge yourself: how can you detect internet connection failure? NSData(contentsOfURL: NSURL, options:NSDataReadingOptions) throws

Part 3: Present details

1. In the DetailView replace the view with a WKWebView

2. Use the following code to display the “overview” of each selected movie

```
var html = "<html>"
html += "<head>"
html += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">"
html += "<style> body { font-size: 150%; } </style>"
html += "</head>"
html += "<body>"
html += overview
html += "</body>"
html += "</html>"
```

3. Your detail view should allow the user to go back to the master

Part 4: UI

1. Design the table cell view to show the the title, votes, release data, and votes. You need to use this url to get the image of each movie: http://image.tmdb.org/t/p/w300 + poster_path

2. Challenge yourself: improve the detail view with better formatting

3. Add a button to the table view that allows uses to filter the movies based on voting (the granularity is up to you). Make use of modal views. The user should be able to disable the filter.

Part 5: Threading

By downloading data from the internet in viewDidLoad() your app will freeze and will not be responsive until everything is downloaded (worse when the internet connect in slow). Heavy waited work should be done on the background thread and the main thread should only be for user interaction. Grand Central Dispatch framework takes care of creating and destroying threads for you upon request. It takes 2 parameters: the first is type of thread you need, e.g., background or main, and the second is a collies contain the code that should be executed on that thread.

1. Call dispatch_async() to make all the loading code in viewDidLoad run in the background queue with User Initiated quality of service.

2. The parts of your code that are related to the interface e.g., the reload table view or show alters must be wrapped by another dispatch_async() that calls the main queue

Submission
Create a zip archive including the following items

- ❏ `UpcomingMovies Xcode project`
- ❏ `Members.txt` — (Only for new teams)
- ❏ (optional) `addendum.pdf` 1-page of anything further than the required submission

Email your submission to hamdan@cs.rwth-aachen.de with subject [iPhone 2015] A034submission

Grading
We will grade this assignments using the following questions.

• The app is working as expected in all simulations (without warning or errors)?

• Is the app responsive?

• The UI design follows the iOS Human Interface Guidelines?

• The structure of the app follows MVC correctly?

• All projects provide modular implementation (use function for concrete tasks instead of a code jam)?

★ 1.0 — Accomplish all “challenge yourself” tasks and clearly went above and beyond what was given in the assignment sheet by improving usability, features, or performance of the implementation.

Incomplete submission will receive at maximum 2.3.
Late submissions *will not be graded*.

Looking forward
For advanced students, the following pointers will shape your mindset for the topic we will discuss in the next lab and beyond this class.

• Check out the role of NSOperation in iOS

• Look more deeply into threads

• Look more onto GCD