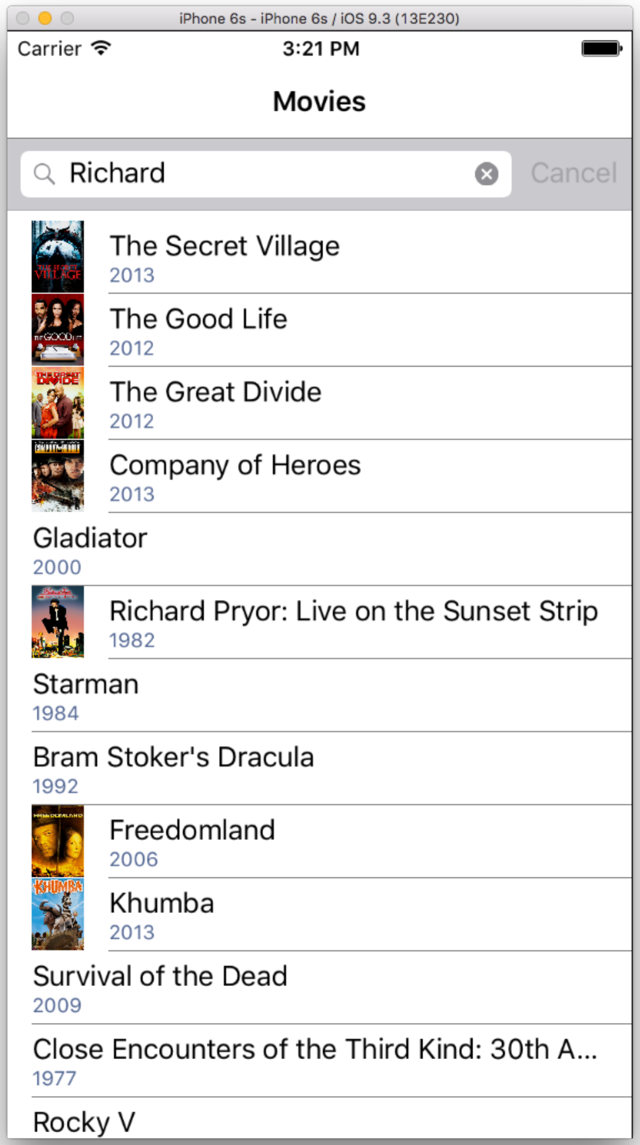
Data Access

By: Mosh Hamedani

Exercise 2

This exercise is going to be more challenging and may take 30 - 60 minutes of your time. Are you ready for a challenge?

Build an app to show the list of movies on Netflix. The user can look up movies by actor.



The Secret Village
2013

The Good Life
2012

The Great Divide
2012

Company of Heroes
2013

Gladiator
2000

Richard Pryor: Live on the Sunset Strip
1982

Starman
1984

Bram Stoker's Dracula
1992

Freedomland
2006

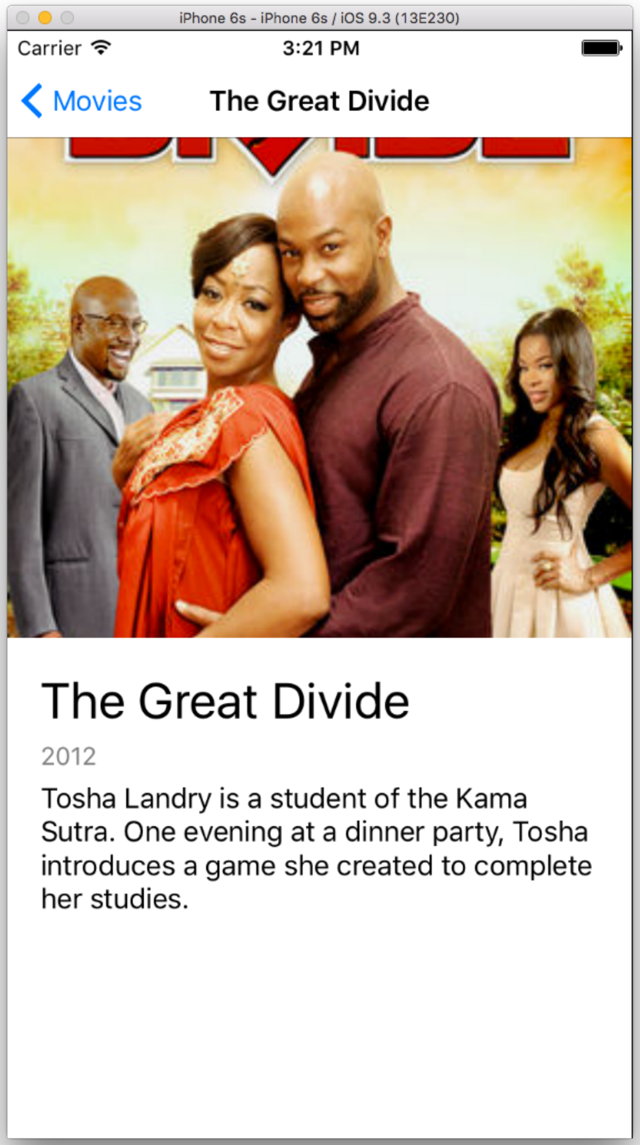
Khumba
2013

Survival of the Dead
2009

Close Encounters of the Third Kind: 30th A...
1977

Rocky V

They can select a movie to view the details about that movie.



The Great Divide

2012

Tosha Landry is a student of the Kama Sutra. One evening at a dinner party, Tosha introduces a game she created to complete her studies.

Data Access

By: Mosh Hamedani

MoviesPage

- **Title:** Title of the page should be Movies.

- **Min 5 characters:** Initially ListView is not visible. Once the user types at least 5 characters, we call Netflix Roulette API to get the list of movies.

- **Endpoint:** http://netflixroulette.net/api/api.php?actor={0}

  Before writing any code, ensure that the endpoint works in your browser. Replace {0} with a name like Richard. Sometimes developers of Netflix Roulette API break things.

- **Deserializing JSON result:** The name of properties in JSON objects returned from the API do not follow standard C# naming conventions. To map your C# properties to these properties, use JsonProperty attribute in your Movie class. If you don't use this attribute, JsonConvert cannot deserialize objects properly.

```
public class Movie
{
     [JsonProperty(“release_year”)]
     public int ReleaseYear { get; set; }
}
```

- **Handle errors:** The application should catch and handle errors instead of crashing. Netflix Roulette API may be down or return an unexpected error. In these situations, we should display an alert with the message: “Could not retrieve the list of movies.” If the endpoint is functioning properly, simulate an error by manually throwing an exception in the code. Ensure that your application can respond to errors and doesn't crash.

- **No movies found:** If there are no movies matching the search criteria, this endpoint returns an HTTP NotFound error and this will cause httpClient.GetStringAsync to throw an exception. In this case you don't want to display an alert because this is not an unexpected exception. It's better to treat this as a scenario where the result is an empty list. To prevent an exception from being thrown, replace GetStringAsync with GetAsync:

```
var response = await _client.GetAsync(url);

if (response.StatusCode == HttpStatusCode.NotFound)
       movies = new List<Movie>();
```

When there are no movies in the result, hide the ListView and display a label below the search bar with the message "No movies found matching your search."

- **Frame:** To add a bit of padding around the label so it doesn't stick to the search bar, you can wrap it in a Frame element. A Frame is a simple container that we use to put shadows, border or padding around an element.

```
<Frame Padding="20" HasShadows="false">
   <Label … />
</Frame>
```

- **Movies found:** If there are movies in the search result, display their poster, title and the release year using an ImageCell. If the ListView is visible, the label should be invisible, and vice versa.

- **Cancelling:** If the user types something in the search bar and presses cancel, nothing should happen. (You need to ensure e.NewTextValue is not null in ItemSelected event handler, otherwise the application crashes.)

- **Selection:** When the user selects a movie, they should be navigated to MovieDetailsPage. When they come back, the selected movie should be de-selected.

- **ActivityIndicator**: Display an AnctivityIndicator when we call the API till we get the result. Bind IsRunning property of ActivityIndicator to a public property in

MoviesPage class. You should implement this property as a “bindable property”; otherwise, ActivityIndicator will not see the changes in this property.

MovieDetailsPage

- **Title:** Title of the movie should be displayed on top of the page (in the navigation bar). See the screen shot.

- **Parameter:** Constructor of this page should expect a Movie object. Ensure Movie is not null.

- **Call another Endpoint:** Even though we get the Movie in the constructor of this class, in a lot of real-world apps, we send another request to the server to get the full details about the given resource. Netflix Roulette API does not have an endpoint to get a movie by ID. The closest we can use is the following:

  http://netflixroulette.net/api/api.php?title={0}

- **Extract a Service:** At this point, you should see that you’ve hardcoded the base URL for the API in two different classes (MoviesPage and MovieDetailsPage).

  If tomorrow we decide to use a different API with a different URL and different message format, we have to change multiple classes. It’s better to extract the logic for finding movies by an actor or getting a movie into a class called MovieService.

  This class will be the only container in the application that knows how to work with our backend API. If tomorrow we decide to move to a different API or developers of Netflix Roulette API make changes to the API contract, there is only class in the application that should be modified: MovieService. The rest of the application will be unaffected.

  MovieService should have two methods:

```
IEnumerable<Movie> FindMoviesByActor(string actor)
Movie GetMovie(string title)
```

Both these methods should be asynchronous because they involve going over the network. We don't want to block the UI.

Once you do this refactoring, use MovieService in both pages.

- **Layout:** Display the poster, title, summary and release year of the movie. Use RelativeLayout and ensure the poster takes half of the page. (I tried with AbsoluteLayout but it didn't work quite well because of the NavigationBar on top of the page.)

- **Attributes:**

  - Title: FontSize = 30

  - ReleaseYear: FontSize = 15, TextColor = Gray