演習の手引き

この資料について

この資料は、自己解決能力を養うために必要最小限の情報のみ記載しています...

開発をする上で重要になる項目

- IBOutletとIBAction

  IBOutletは、Storyboard上のUI部品とクラスのプロパティを紐付けるために使用します。

  `IBOutlet 接続`などのキーワードで検索すると、欲しい情報が手に入ると思います。

- UIViewControllerのライフサイクルイベント

  ライフサイクルイベントとは、画面読み込みなどのタイミングで呼び出されるイベントのことです。

  例えば、`ViewWillAppear`というメソッドは画面が読み込まれる直前に必ず呼ばれるメソッドになります。

  詳しい情報は`UIViewController ライフサイクル`などで調べると良いと思います。

- スレッドについて

  なにか重たい作業を行いたい場合(今回はないですが)、重たい作業をメインスレッドで実行させてしまうと、その間フリーズしてしまいます。

  重たい作業を行いつつ、なめらかな処理を実現したい場合は別スレッドで非同期処理を行うなどする必要があります。

  ほかに重要な点としては、UI操作はメインスレッドで行う必要があるということです。

  `DispatchQueue メインスレッド`などのキーワードで検索すると情報を得ることができます。

- 画面遷移

  今回は`Manual Segue`を使用する想定です。

  画面遷移については`Manual Segue 呼び出し`などのキーワードで検索しましょう。

- アラートを表示したい場合

  `UIAlertController`に`UIAlertAction`を追加する形で使用します。

- ユーザデータの保存をしたい場合

  `UserDefaults`を使用します。

- ボタンやラベルの内容を変更したい場合

  `UIButton`や`UILabel`を調べましょう。

- ボタンが押されたときの処理を追加したい場合

  `IBAction`について調べましょう。

- 設定画面に使われているテーブルを実装したい

  `UITableViewDataSource`と`UITableViewDelegate`、この2つのプロトコルを実装する必要があります。

  実装したあとに、UITableViewに対して`delegate`と`datasource`を設定すれば完成です(よく忘れがちです)

  Extensionを使って見やすいコードを書いてもいいかもしれません。

- n秒後になにか処理をしたい

  `DispatchQueue.main.asyncAfter`などを使用する場合が多いです。

  `DispatchQueue`系の情報は知っておくと業務で役に立ちます。

- 乱数がほしい

  `arc4random_uniform`を使いましょう。

- シミュレータがない

  `Deployment Target`を変更しましょう。シミュレータが存在しないのは、Deployment Targetのバージョンが高すぎて使用中のXcodeにシミュレータが存在しないためです。

- IBOutletでなぜ`weak`を使うのか

  メモリを無駄遣いしないためです。余裕がある人で詳しく知りたいなら弱参照や`ARC`について調べましょう。メモリ管理周りの話なので、余裕のない人は後回しでいいです。

  これとか個人的にわかりやすい気がします。

開発の大まかな流れ

Obj-C版の資料や完成版コードを見るか、キーワードで検索しましょう。

私に聞いたほうが速いかもしれません

1. IBOutletを用意する
2. ~~Manual SegueにIDをつける~~
3. 設定画面をつくる
4. ユーザ名入力画面をつくる
5. おみくじ結果表示のロジックをつくる(乱数で結果選択)
6. おみくじ結果の画面遷移部分を書く
7. UserDefaultsでユーザ名を記憶させる
8. 日時取得＋反映部分を書く(難しいのでオプション扱い)

結果表示画面 -> 設定画面 -> ユーザ名設定画面