Beginning Realm on iOS

Hands-on Challenges

Beginning Realm on iOS Hands-On Challenges

Challenge A: Convenience initializers

Since often times it makes sense to set the values of a group of properties on your Realm objects, you will probably end up adding a number of custom initializers to your model classes.

In the context of your `Task` class the only required properties would be `title` and `priority`; the rest can use their default values.

Open **Task.swift** and add to `Task` a simple convenience `init`:

```
convenience init(title: String, priority: Int) {
    self.init()
    self.title = title
    self.priority = priority
}
```

First you need to call the default `init()` to make sure your object is initialized properly – `Object` is a sub-class of `NSObject` so you call up into the `init()` chain.

Then you assign the values to the object properties and you're done. As you see from this example your models are full fledged classes so you can add custom methods and/or initializers as you please.

Challenge B: Dynamic properties

Realm will not try to initialize any properties that do not have a setter – this includes any read-only and dynamic properties. This comes handy because you don't need to add an `ignoredProperties()` method because of any of those.

In this quick challenge you will add two properties to add some additional information to your models.

First you will add a method that returns the priority label text depending on the priority assigned to the current task. Add at the bottom of **Task.swift**:

```
extension Task {
  var priorityText: String {
    return priority > 0 ? "High" : "Default"
  }
}
```

For any priority higher than `0` your return *High* and all other values will be denoted with *Default*.

Since this property does not have a setter, Realm will not try to persist its value on disc or in memory (e.g. it will persist `priority` but not `priorityText`)

Add one more property inside the extension body to also provide a custom color for the task’s title based on the level of priority it has:

```
var priorityColor: UIColor {
    return priority > 0 ? UIColor.redColor() : UIColor.blueColor()
}
```

When you wire up the table view’s cells, in a later video, you will display the priority text in the color this property returns (red for high priority, otherwise – blue).