

No description, website, or topics provided.

Edit

Manage topics

🕒 2 commits

🌿 1 branch

📦 0 releases

👤 1 contributor

Branch: master ▾

New pull request


Create new file


Upload files


Find file

Clone or download ▾


This branch is even with LambdaSchool:master.

 Pull request

 Compare



 **SpencerCurtis** Update README.md with project instructions

Latest commit 44aa636 on Oct 3, 2018

 README.md

Update README.md with project instructions

5 months ago

 README.md 

Documents Obj-C

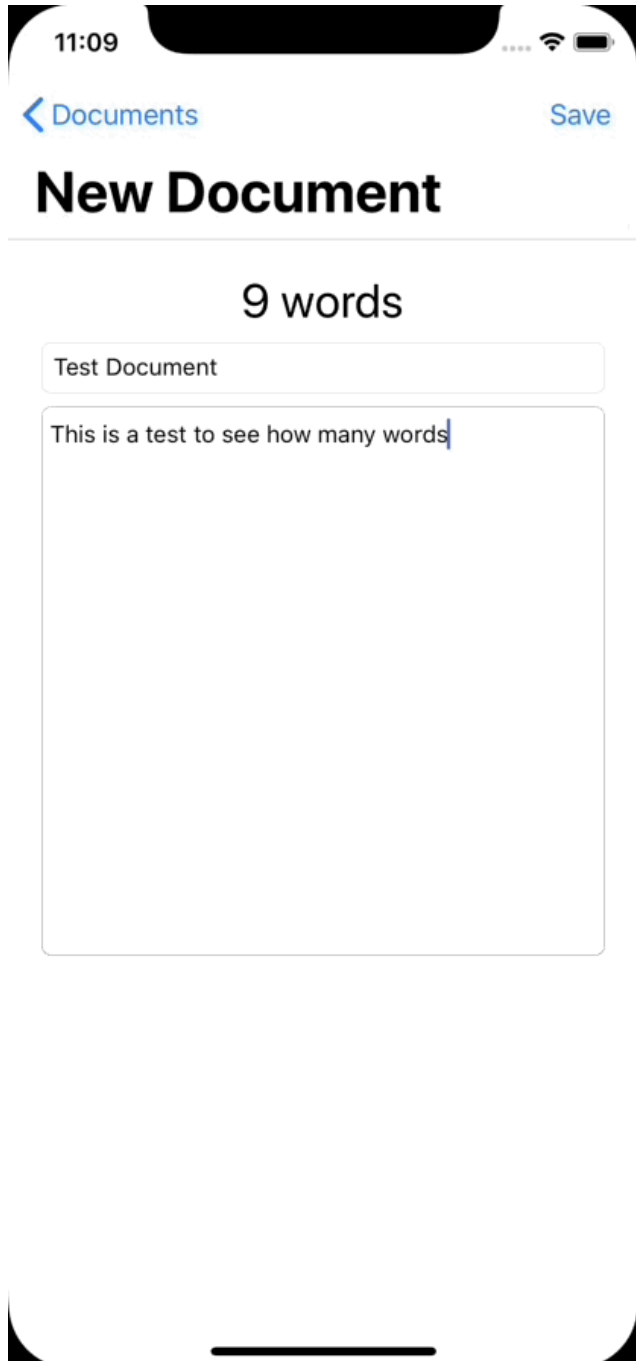
Introduction

The goal of this project is to create a project that allows a user to create documents, and while they write the documents, they will be able to see the word count of the document in real time. This will help you practice concepts learned today including custom property accessors and attributes, categories, and class extensions.

Instructions

Please fork and clone this repository, then create an Xcode project in it. Commit regularly as you complete the requirements in this project.

Please look at the screen recording below to know what the finished project should look like:



Part 1 - Storyboard

You will create a simple master-detail layout in the "Main.storyboard".

1. Add a `UITableViewController` scene and embed it in a navigation controller. Set the navigation controller as the initial view controller.
2. Create a Cocoa Touch subclass of the table view controller scene and set the scene's class to it.
3. Set the cell's identifier, and its style to subtitle.
4. Add an "Add" bar button item to the table view controller.
5. Add a `UIViewController` scene. Add the following to it:
 - A label that will display the amount of words in the document

- A text field for the user to enter the title of the document
 - A text view for the user to enter the actual text of the document.
 - A bar button item (after you add a navigation item) that uses the "Save" System Item.
6. Create a Cocoa Touch Subclass of `UIViewController` for this scene and create any outlets and/or actions you may need.
 7. Add segues from the cell and the "Add" bar button item to the `UIViewController` scene. Give the segues appropriate identifiers.

Part 2 - Category, Model, and Model Controller

NSString Category

1. Create a new *Objective-C* file in "File > New File...". Set its name to "WordCount". For the "File Type" select "Category". For the "Class", choose `NSString`
2. Add an instance method called `wordCount` that returns an `int` to the category. In the implementation, this should simply return the amount of words in the string.

Model

1. Create a Cocoa Touch Subclass of `NSObject` called `Document` (with your prefix).
2. Add properties with the appropriate types and property attributes for a document's title, text, and word count.
3. Create an initializer that will set the model's title and text, but not the word count.
4. Create a custom property accessor for the word count property. In the accessor, use the `wordCount` method you just created in the `NSString` category. (Remember to import the "NSString+WordCount.h" file or you won't be able to access the method)

Model Controller

1. Create a model controller for your `Document` model.
2. Add a property (with the appropriate property attributes) that will hold an array of documents.
3. In the initializer of this class, initialize the array with an empty array.
4. Add and implement three methods that will create, update, and delete a document, respectively.

Part 3 - View Controllers

1. Add a property with property attributes for an instance of your model controller in the table view controller subclass.
2. In the two initializers of the table view controller, instantiate the model controller property.
3. Implement the required `UITableViewDataSourceMethods`.
4. In your detail view controller, add two properties with property attributes; one for your model and one for your model controller.
5. In the `prepareForSegue` and depending on whether the user tapped the add button or a cell, pass the necessary properties to the detail view controller.
6. In the detail view controller, adopt the `UITextViewDelegate` protocol. Set the view controller as the delegate of the text view.
7. Implement the `textViewDidChange` method. This will get called every time the user enters a character on the keyboard while the text view is the first responder. In the method, update the label's text to show how many words are in the text view's text. Use the `wordCount` method from the `NSString+WordCount` category to do this.
8. Create an `updateViews` method that should place the document's properties in the appropriate UI elements for the

user to see.

9. When the user wants to save the document, either create a new document or update an existing one.

Go Further

1. Implement Core Data in this application. If you wish to do this, be aware of a few things:
 - In your data model file, you must change the "Code Generation" to Objective-C in the File Inspector.
 - Create a category for your `Document+Convenience` . You may need to import the header file for the generated Core Data class. It would be something like `"LSIDocument+CoreDataClass.h"`