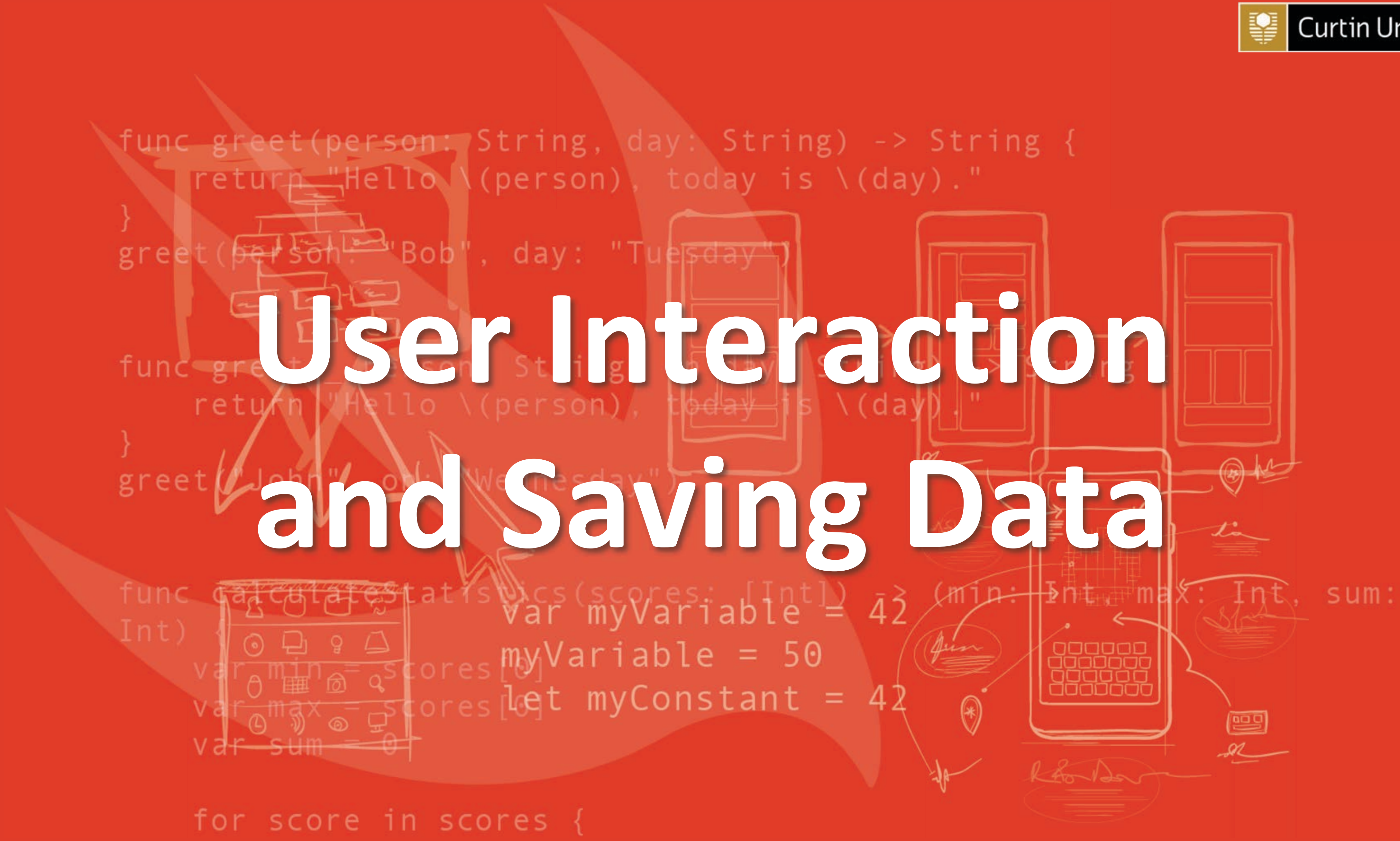


Curtin University

User Interaction and Saving Data

In this lesson, we'll learn...

- To create your own customised cells;
- To add and delete rows and use static tables;
- To add custom row actions;
- To encode and decode data to save and load;
- To read and write data from and to a file.

More Complex Table Views

Why Create Custom Views?

- Create a custom table view cell for different reasons:
  - to display more text, more buttons etc.;
  - customise object locations within cells;

Custom View Creation: Steps Involved

- Select the Cell;

- In the Attributes inspector set Style to Custom;

- Using Interface Builder tools, we can customise it;

Content Hugging

- The flag needs to fit snuggly into the content area;
- Change the horizontal field priority in Content Hugging Priority;
- From 251 to 252;
- Prioritises the placement by the Auto Layout Engine.

New Cell SubClass

- A custom table view class is needed;
- We can create outlets for configuring the cell;
- Ensure it is a Custom class of **`FlagTableViewCell`**

Editing Table Views

- In editing mode the table view calls the delegate method:
  - **`tableView(_: editingStyleForRowAt: )`**
- There are 3 options:
  - **`.none`**
  - **`.delete`**
  - **`.insert`**

Delegate Methods

- Delegate methods are called in order in edit mode:
  1. **`tableView(_: canEditRowAt: )`**
  2. **`tableView(_: editingStyleRowAt: )`**
  3. User does something here…
  4. **`tableView(_: commit: forRowAt: )`**

Adding to the Flags

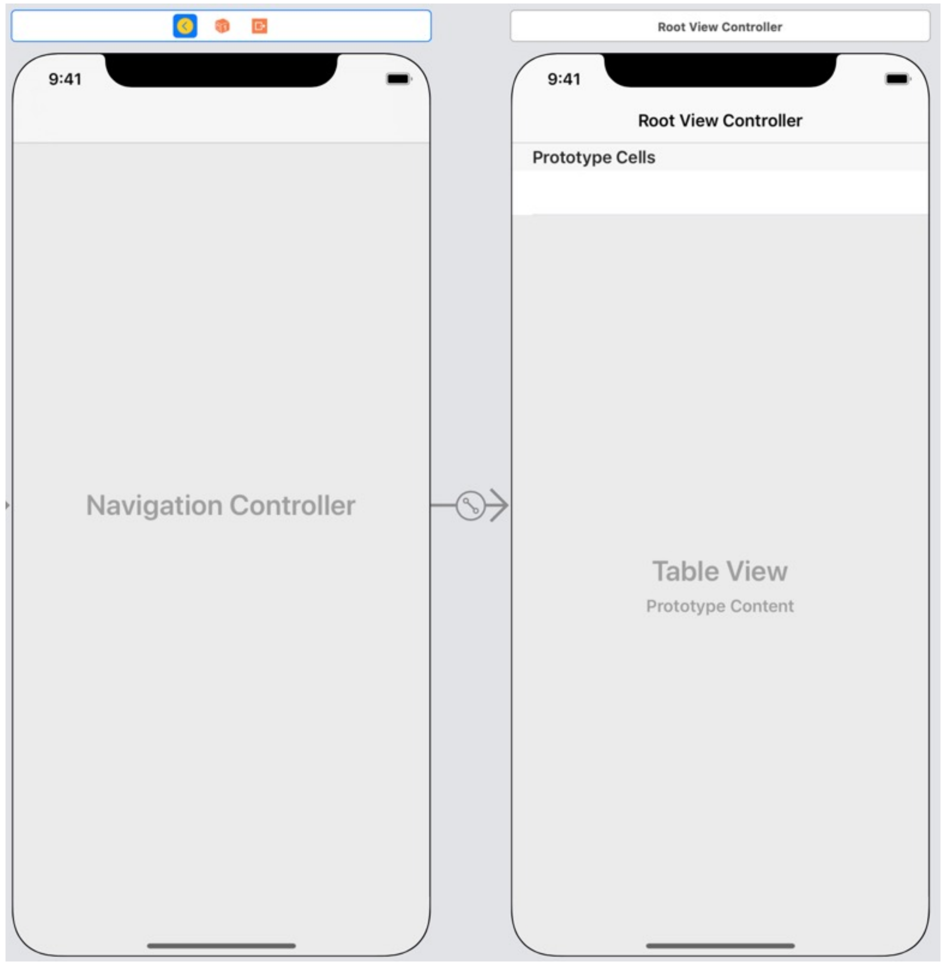
- Add a + button to the navigation bar;
- Use a new view controller to add details;
- The same view controller for edits and additions can be used;
- A Static Table View is used in this situation.

Curtin University

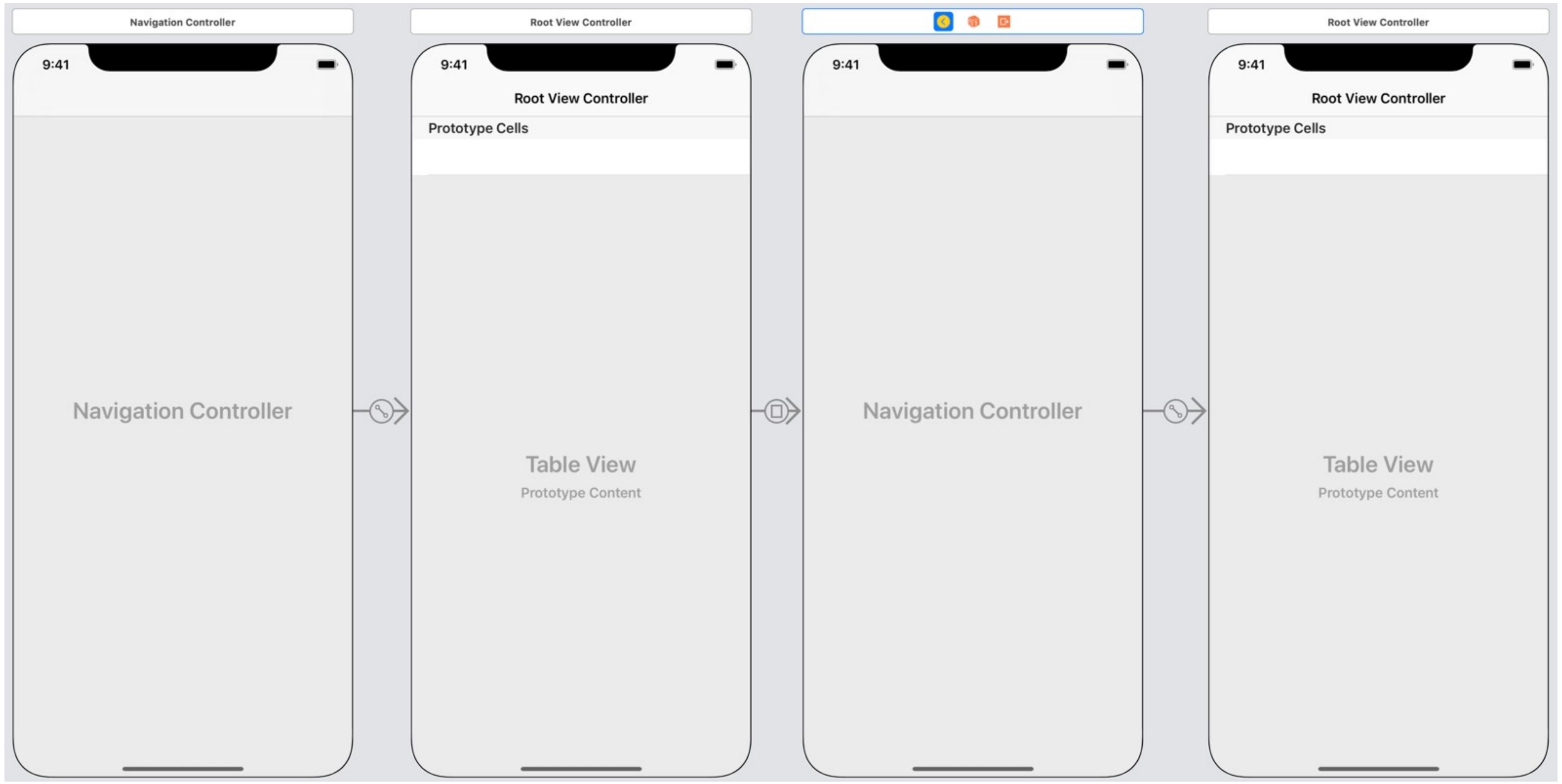
Static Table Views

- Use a table view controller;
- Do NOT implement the data source protocol;
- Populate the table view using **viewDidLoad()**

Add a Navigation Controller



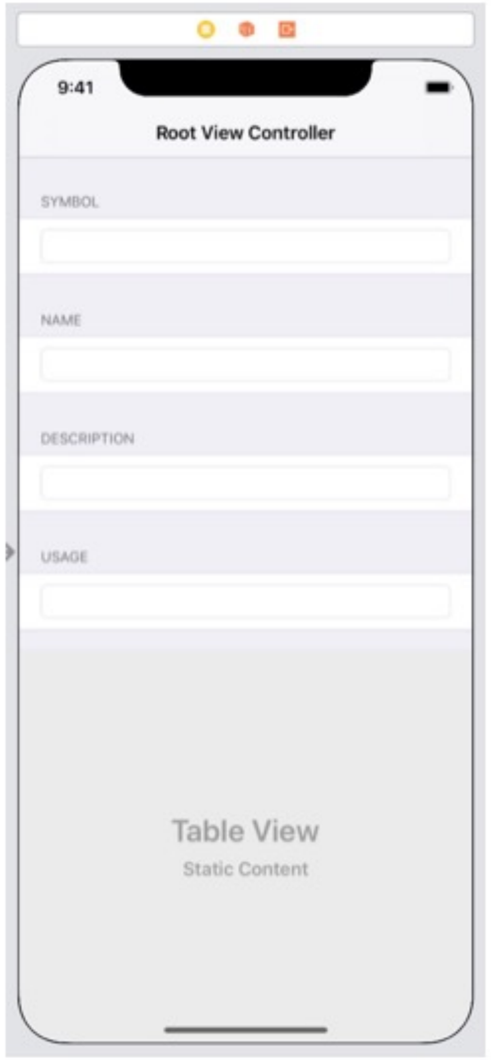
To the Navigation Controller



Curtin University

Develop the Table View

- Setting the content to Static Cells;
- Change the labels to reflect the content.

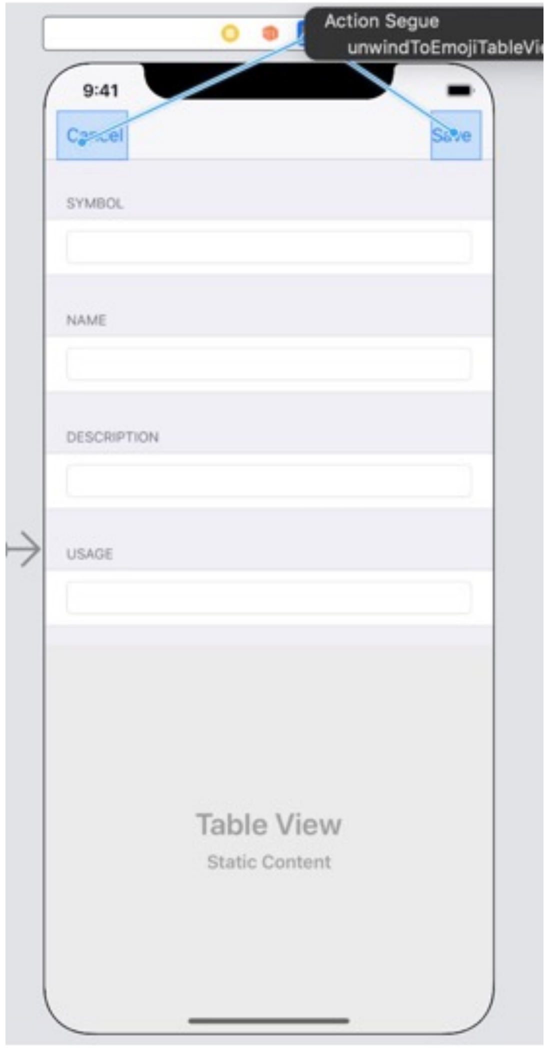


Saving Canceling

- Add Navigation Bar items;

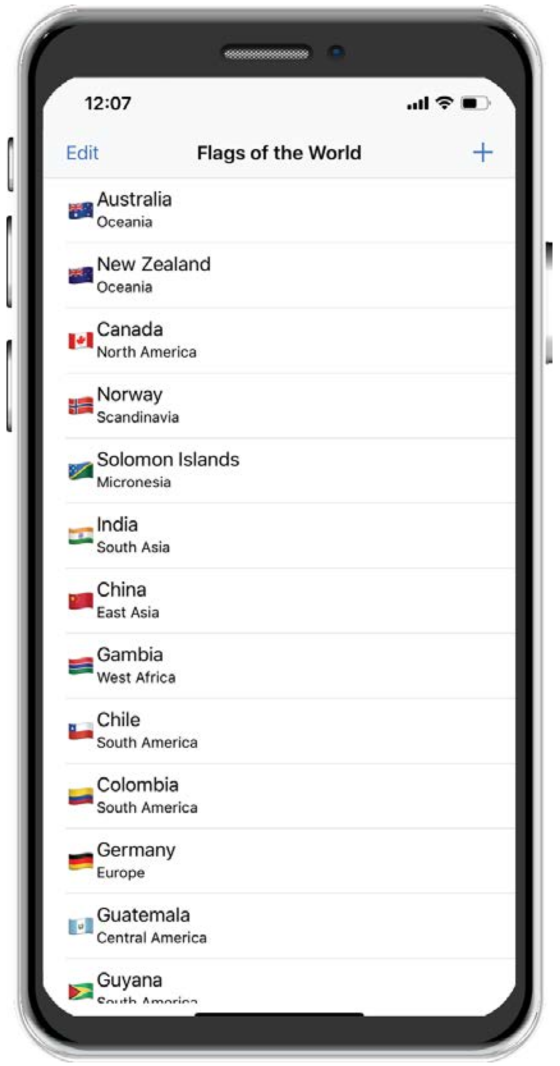
- Almost all Apps have these buttons:
  - Save; and
  - Cancel.

- Save only when something changed.



Curtin University

What it may look like



12:07

Edit Flags of the World +

Australia
Oceania

New Zealand
Oceania

Canada
North America

Norway
Scandinavia

Solomon Islands
Micronesia

India
South Asia

China
East Asia

Gambia
West Africa

Chile
South America

Colombia
South America

Germany
Europe

Guatemala
Central America

Guyana

What did we just do?

- We have been looking at:
  - Creating custom cells;
  - Adding, deleting and editing rows in a table view;
  - Using static table views.

Saving Data

Where did it go?

- An App that doesn’t save data is like a pub with no beer;
- “*Sorry boss, the App doesn’t save my work, I will try again tomorrow!*”
- Apps must save data.

MVC Architecture & Persisting

- Persisting data happens by creating storage layer;
- Controller: controls the View and Model to ensure correct data is displayed;
- Controller object allows us to access the storage layer.

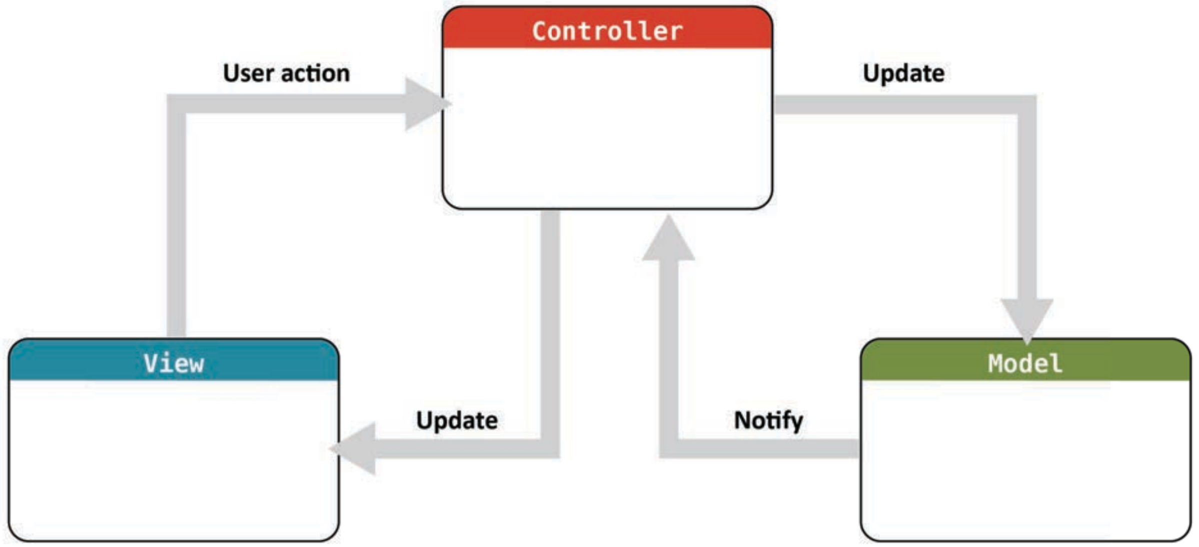


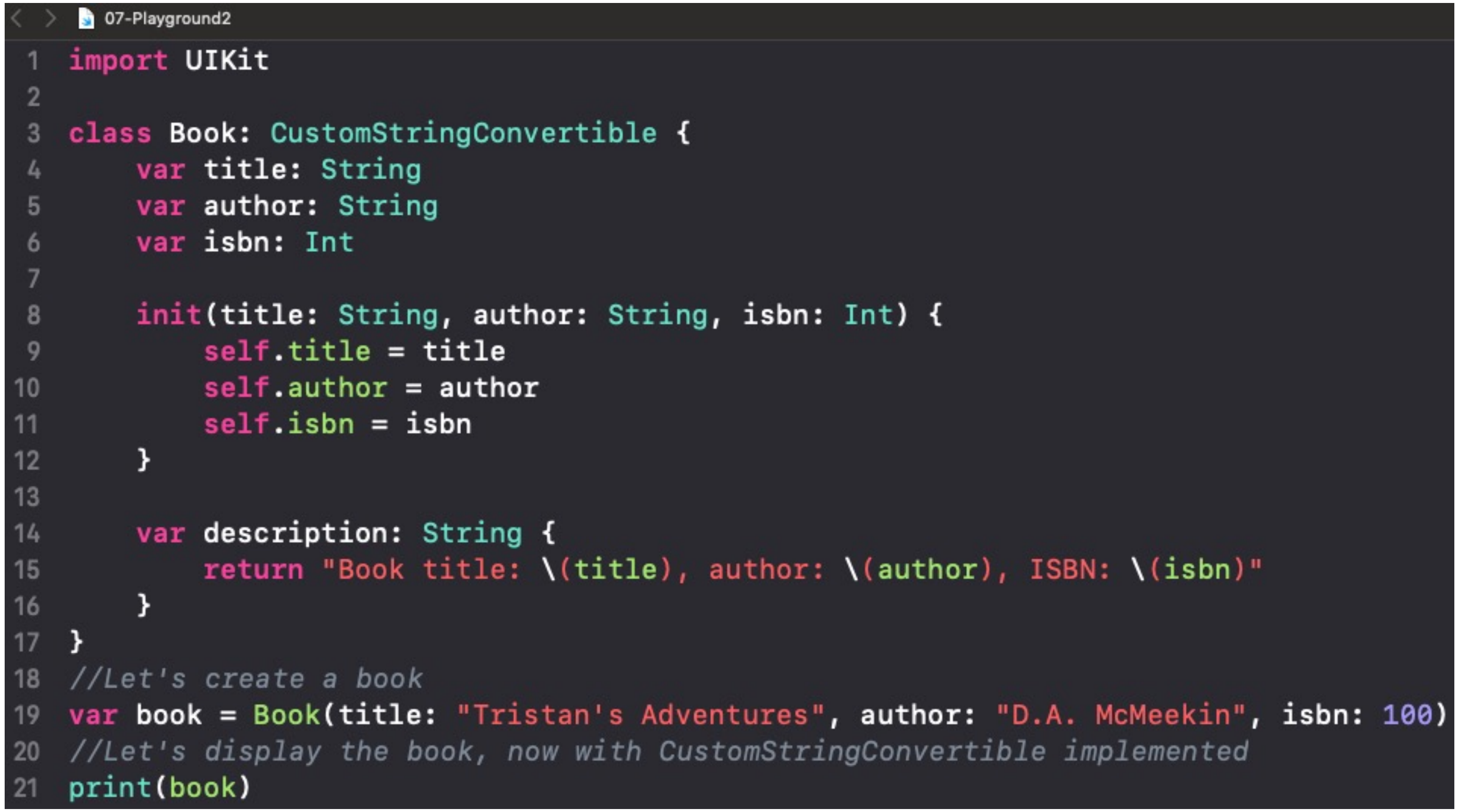
Image adapted from: https://developer.apple.com

Curtin University

Protocols

- Remember: CustomStringConvertible & Equatable;
- To save data the Codable protocol is implemented;
- An object that conforms to Codable can save & load data;
- To conform to Codable protocol requires two methods to be implemented;
- Most built in Swift types already conform.

Book Class Implementing Codable



07-Playground2

```swift
import UIKit

class Book: CustomStringConvertible {
    var title: String
    var author: String
    var isbn: Int

    init(title: String, author: String, isbn: Int) {
        self.title = title
        self.author = author
        self.isbn = isbn
    }

    var description: String {
        return "Book title: \(title), author: \(author), ISBN: \(isbn)"
    }
}
//Let's create a book
var book = Book(title: "Tristan's Adventures", author: "D.A. McMeekin", isbn: 100)
//Let's display the book, now with CustomStringConvertible implemented
print(book)
```

Curtin University

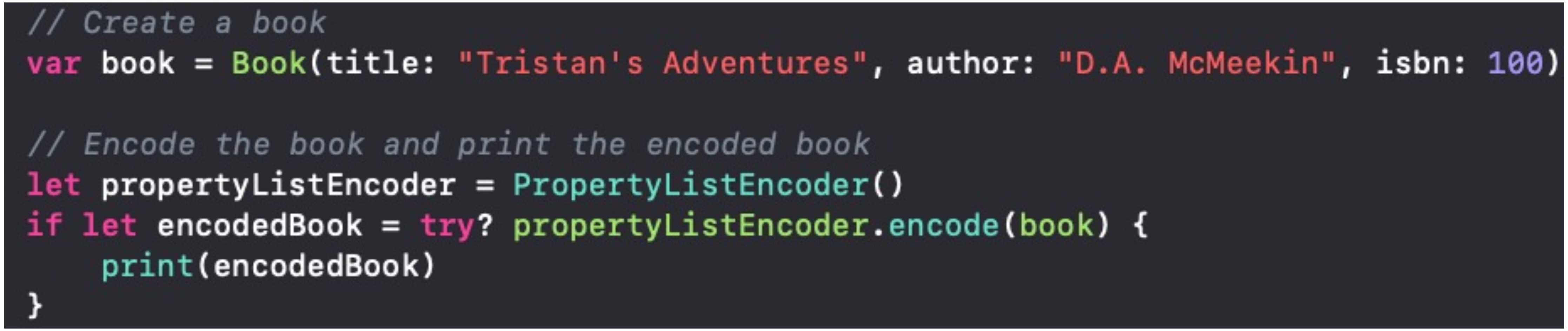
Book Class Implementing Codable

- In this example the data will be saved to a plist format;

- An Encoder object is used to encode the data for saving;

- A Decoder object is used to decode the data from its saved state.

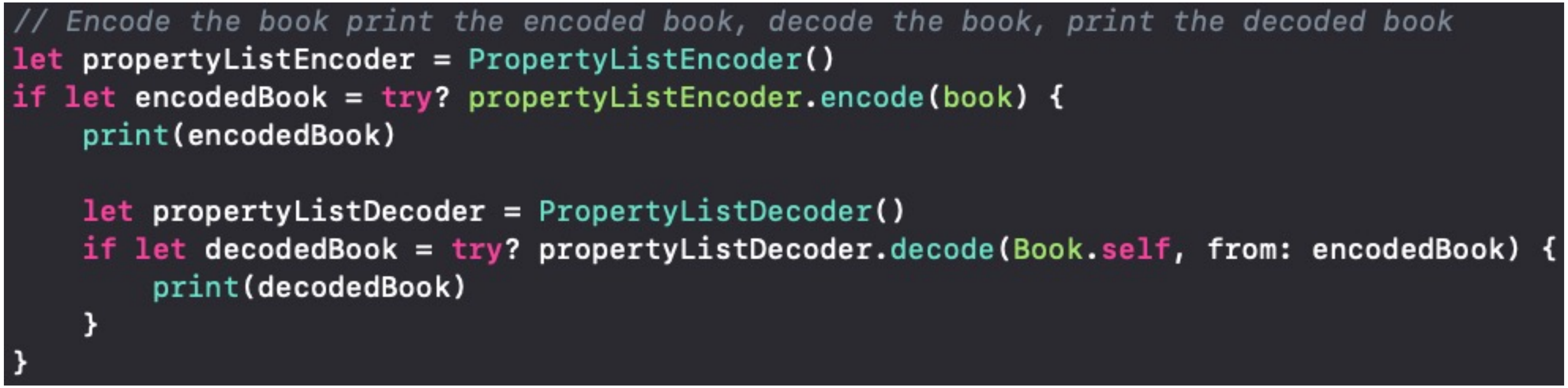
encode the data with Codable



```swift
// Create a book
var book = Book(title: "Tristan's Adventures", author: "D.A. McMeekin", isbn: 100)

// Encode the book and print the encoded book
let propertyListEncoder = PropertyListEncoder()
if let encodedBook = try? propertyListEncoder.encode(book) {
    print(encodedBook)
}
```

- The encode method is a throwing method, hence we use try? with it;
- It now will return optional Data instead of errors;
- Printing gives us the number of bytes in it.

decode the data with Codable



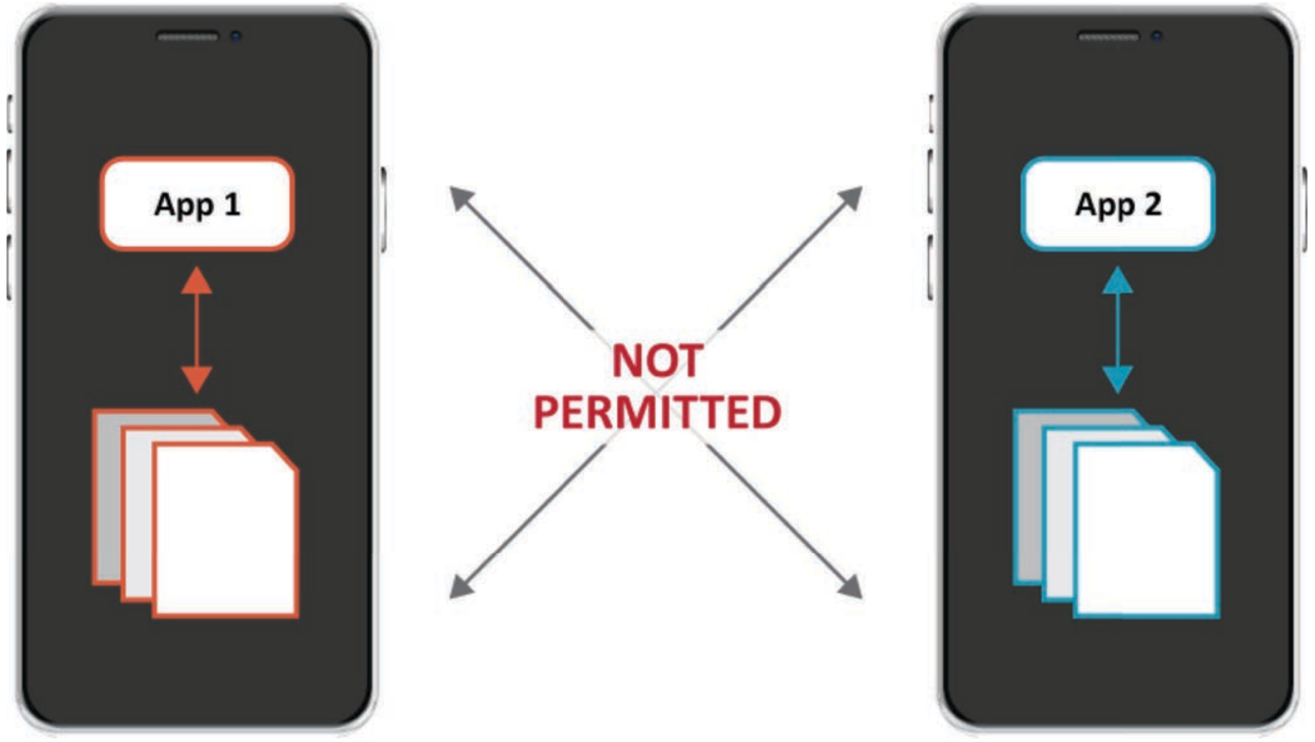
```swift
// Encode the book print the encoded book, decode the book, print the decoded book
let propertyListEncoder = PropertyListEncoder()
if let encodedBook = try? propertyListEncoder.encode(book) {
    print(encodedBook)

    let propertyListDecoder = PropertyListDecoder()
    if let decodedBook = try? propertyListDecoder.decode(Book.self, from: encodedBook) {
        print(decodedBook)
    }
}
```

- The decode method is also a throwing method, hence we use try? with it;
- It returns optional Data instead of errors;
- Printing gives us the book instance.

Writing Data to File

- Our App still does not have the data after exiting;
- Time to learn about the iOS file system;
- iOS uses sandboxing to protect it from rogue apps.

Sandboxing



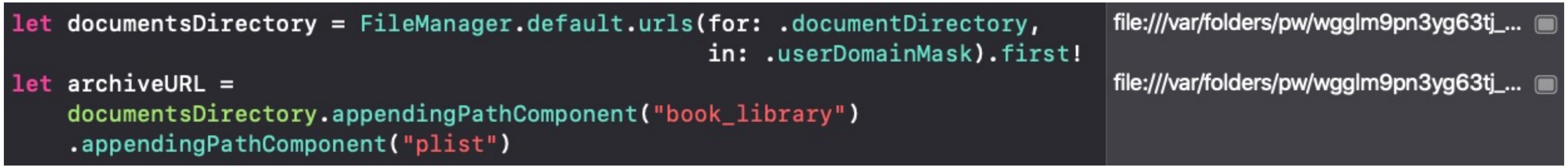
- App1 can not access any of App2's resources;
- Certain cases with user permission access is permitted.

Documents Folder

- An app has certain folders to save data to;
- Documents folder, the location your app saves & modifies its information;
- The path to the Documents folder changes;
- The path is like a URL to the folder;
- A FileManager class function gives access to the Documents folder.

Curtin University

FileManager default URL

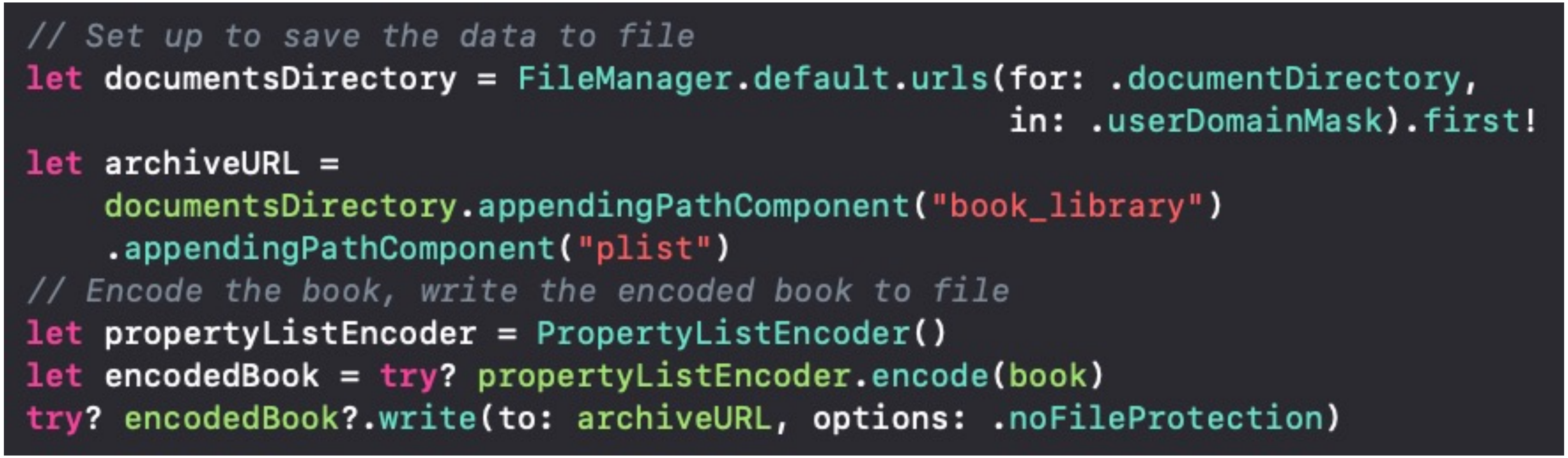


```
let documentsDirectory = FileManager.default.urls(for: .documentDirectory,
                                                   in: .userDomainMask).first!
let archiveURL =
    documentsDirectory.appendingPathComponent("book_library")
    .appendingPathComponent("plist")
```

```
file:///var/folders/pw/wgglm9pn3yg63tj_...
file:///var/folders/pw/wgglm9pn3yg63tj_...
```

- Create a FileManager object;
- Our concern is to use the .documentsDirectory;
- documentsDirectory holds the URL to that folder;
- The actual path is in the sidebar.

Write to File

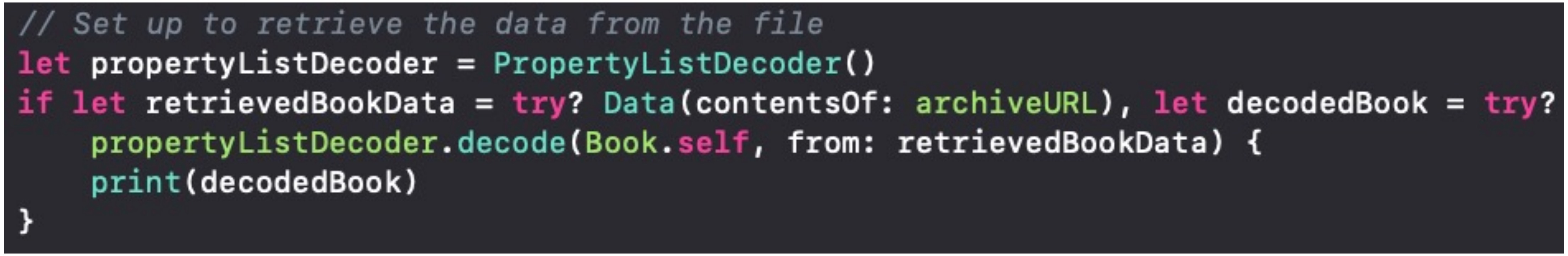


```swift
// Set up to save the data to file
let documentsDirectory = FileManager.default.urls(for: .documentDirectory,
                                                  in: .userDomainMask).first!
let archiveURL =
    documentsDirectory.appendingPathComponent("book_library")
    .appendingPathComponent("plist")
// Encode the book, write the encoded book to file
let propertyListEncoder = PropertyListEncoder()
let encodedBook = try? propertyListEncoder.encode(book)
try? encodedBook?.write(to: archiveURL, options: .noFileProtection)
```

- Now using try? Take the encodedObject (encodedBook) and write it to the archiveURL.

Curtin University

Retrieving from File



```swift
// Set up to retrieve the data from the file
let propertyListDecoder = PropertyListDecoder()
if let retrievedBookData = try? Data(contentsOf: archiveURL), let decodedBook = try?
    propertyListDecoder.decode(Book.self, from: retrievedBookData) {
    print(decodedBook)
}
```

- Create a PropertyListDecoder() object;
- Create and initialize a Data() object using the archiveURL contents;
- Unpack the data and print() the contents.

What did you do?

- Created a way to save data (a useful App);
- Implemented the Codable protocol;
- Encoded and decoded data;
- Wrote the data to a file;
- Retrieved the data from the file.

Curtin University

Wrapping Up

What we learned in this lesson:

- Created our own customised cells;
- Added and deleted rows, used static tables;
- Added custom row actions;
- Encoded and decoded data, saving and loading it;
- Read and wrote data from and to a file.