
BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

# iOS App Development

Lecturer: Dr. Phan Ngoc Hoang

Tel.: 0776232919

Email: hoangpn [at] bvu [dot] edu [dot] vn


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

# SAVING DATA

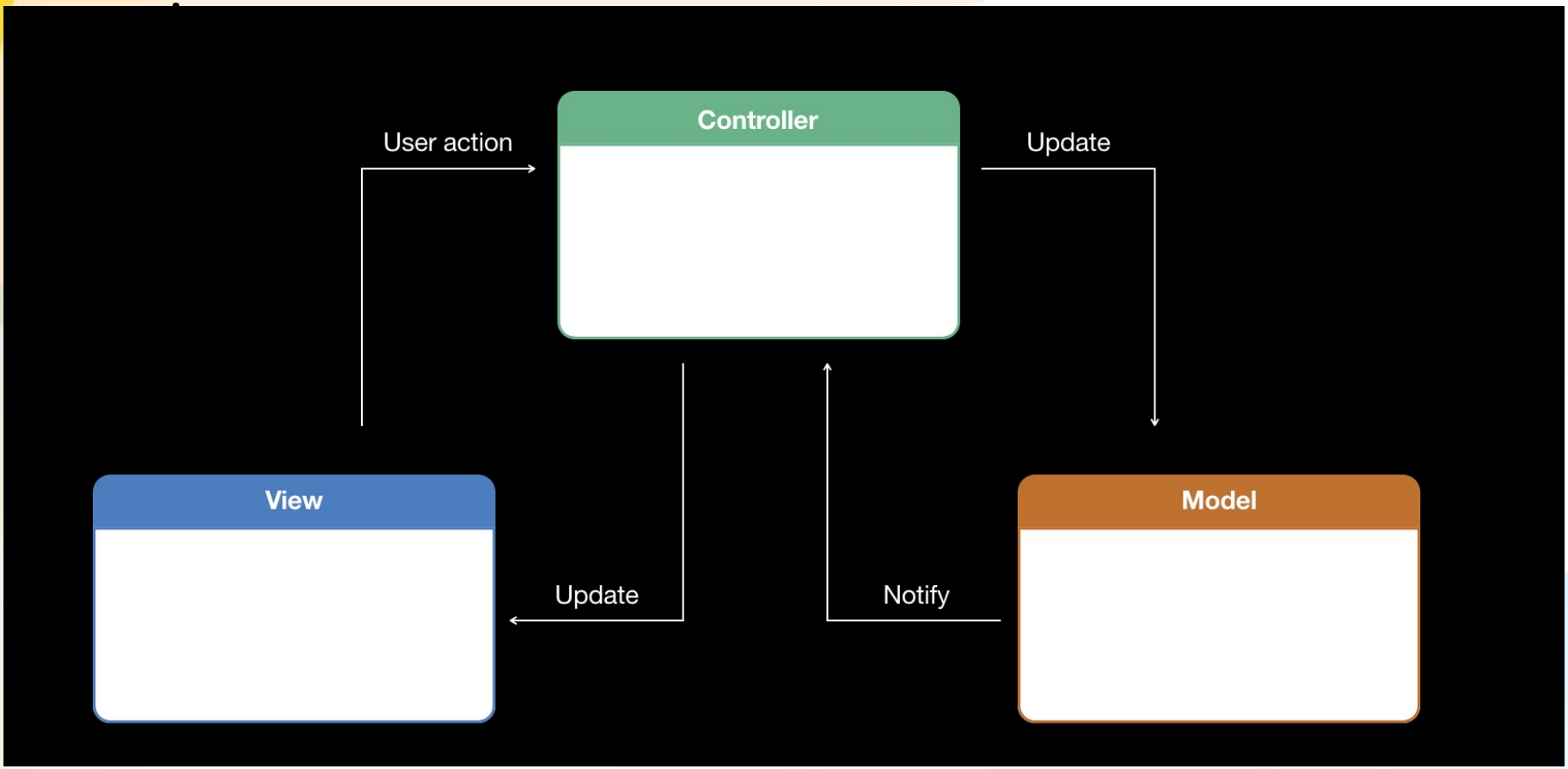
# Contents

- Encoding and Decoding with Codable
- Writing data to file
- Loading data from file
- Working with array

# Learning outcomes

- How to write and access files in your app's Documents directory
- How to serialize model data to a format that can be saved
- How to serialize saved data to model data that can be used in the app

# 1. Encoding and Decoding with Codable



# 1. Encoding and Decoding with Codable


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

```
class Note: Codable {
    //Encoder
    //Decoder
}
```

- Use an Encoder object to encode
- Use a Decoder object to decode.

# 1. Encoding and Decoding with Codable

- Create a new playground called “PersistencePractice” (File > New > Playground).
- Declares a simple Note model object that adopts the Codable protocol.

```
class Note: Codable {
    let title: String
    let text: String
    let timestamp: Date
}
```

# 1. Encoding and Decoding with Codable


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- Create an instance of Note that can be encoded.

```
let newNote = Note(title: "Dry cleaning",
    text: "Pick up suit from dry cleaners",
    timestamp: Date())
```

# 1. Encoding and Decoding with Codable


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- Use an Encoder object to encode a value to a plist.

```
let propertyListEncoder = PropertyListEncoder()
if let encodedNote = try?
    propertyListEncoder.encode(newNote) {
    print(encodedNote)
}
```

- PropertyListEncoder's encode(_:) method is a throwing function, requiring you to use either the do-try-catch syntax or the keyword try?.
- By using try? in this example, encode will simply return optional Data instead of throwing any errors.

# 1. Encoding and Decoding with Codable


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

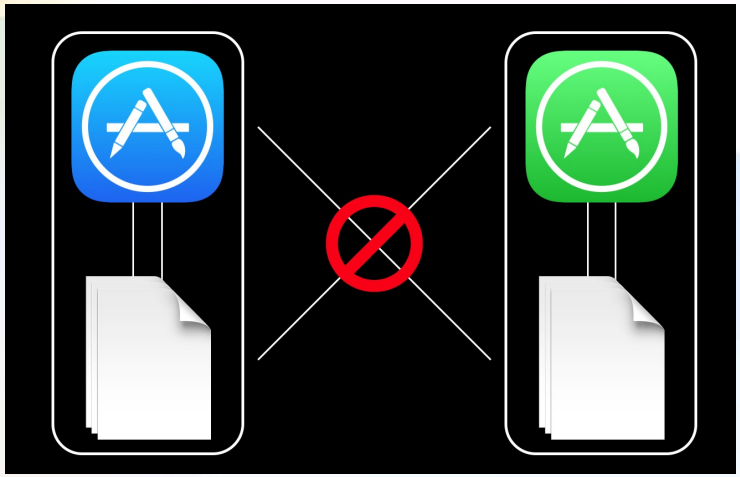
- Use an Decoder object to decode a value from plist (similar encode).

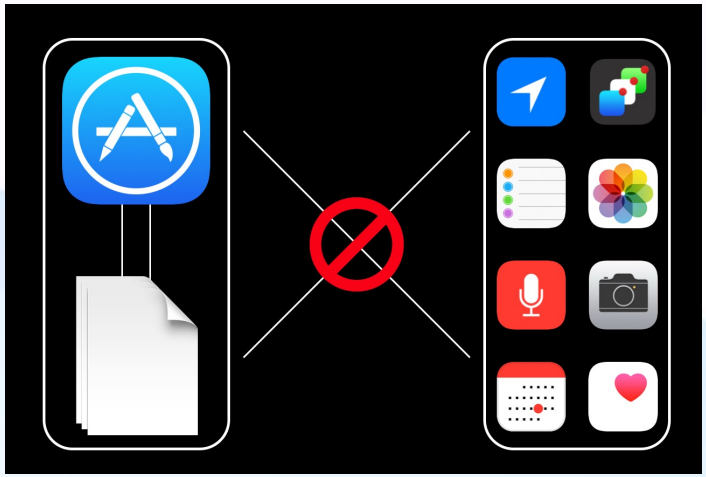
```
let propertyListEncoder = PropertyListEncoder()
if let encodedNote = try?
    propertyListEncoder.encode(newNote) {
    print(encodedNote)

    let propertyListDecoder = PropertyListDecoder()
    if let decodedNote = try?
        propertyListDecoder.decode(Note.self, from:
        encodedNote)  {
        print(decodedNote)
    }
```

# 2. Writting data to file
# App Sandbox

BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

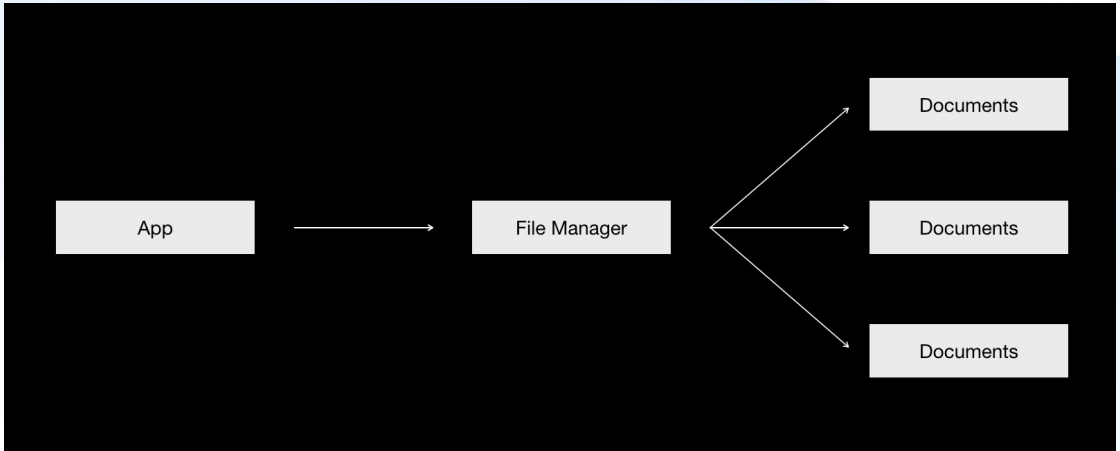
- Each app has its own environment where it can create, modify, or delete data—its own sandbox—but it doesn't have access to resources outside of the sandbox.





# 2. Writting data to file
# App Sandbox

- App has a few directories that it can use to save data.  One of those is called the Documents, where you can save and modify information related to app.
- Feature of the sandbox security model is that the file path to the Documents directory will change each time your app is loaded into memory.



# 2. Writting data to file Document Directory


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- Add code to access to the Documents directory (wherever it is) and enables it to read and write files in that directory.
- Add the full path where you will write your Note object's data to file (appData.plist).

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

# 2. Writting data to file
# Writing Data


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- Use methods on Data to write directly to and from files in that directory.
- you can use the file path at which to save encodedNote and the write(to:options:) method on Data to write to that path.

```
let propertyListEncoder1 = PropertyListEncoder()
let encodedNote = try?
    propertyListEncoder1.encode(newNote)
try? encodedNote?.write(to: archiveURL,
    options: .noFileProtection)
```

# 2. Writting data to file Retrieving Data

- To retrieve the data from the file, you can initialize a Data object using its throwing initializer init(contentsOf:) and pass it the URL at which the data is stored.

```
let propertyListDecoder1 = PropertyListDecoder()
if let retrievedNoteData = try? Data(contentsOf: archiveURL),
    let decodedNote = try?
    propertyListDecoder1.decode(Note.self, from:
    retrievedNoteData) {
    print(decodedNote)
}
```

- You have successfully encoded, saved, loaded, and decoded your note.

# 2. Writting data to file
# Writting array


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- Create array of notes

```
let note1 = Note(title: "Note One", text: "This is a sample
    note.", timestamp: Date())
let note2 = Note(title: "Note Two", text: "This is another
    sample note.", timestamp: Date())
let note3 = Note(title: "Note Three", text: "This is yet another
    sample note.", timestamp: Date())
let notes = [note1, note2, note3]
print(notes)
```

# 2. Writting data to file
# Writting array


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- Get URL to save note array (notes_test.plist)
- Encode and save note array to URL.

```
let arrayArchiveURL =
    documentsDirectory.appendingPathComponent("notes_test")
    .appendingPathExtension("plist")

let propertyListEncoder2 = PropertyListEncoder()
let encodedNotes = try? propertyListEncoder2.encode(notes)
try? encodedNotes?.write(to: arrayArchiveURL,
    options: .noFileProtection)
```

# 2. Writting data to file
# Writting array

- Get note array from file and decode them.

```
let propertyListDecoder2 = PropertyListDecoder()
if let retrievedNotes = try? Data(contentsOf: arrayArchiveURL),
    let decodedNotes = try?
    propertyListDecoder2.decode(Array<Note>.self, from:
    retrievedNotes) {
    print(decodedNotes)
}
```

# 3. Guide: EmojiDictionary


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- Use the Codable protocol, the FileManager, and methods on Data to persist information between app "EmojiDictionary" launches.

# 3. Guide: EmojiDictionary Implement Saving


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- To persist your app's information across app launches, you need to be able to write its information to a file on disk. Start by making your Emoji struct conform to the Codable protocol in its declaration.

# 3. Guide: EmojiDictionary Implement Saving


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- Remember that you can implement saving and loading as static methods on the model. Add the static method signature for a saveToFile(emojis:) function in the Emoji class. It should take an array of Emoji objects as a parameter. Now add a static loadFromFile() method. It shouldn't take any parameters, but should return an array of Emoji objects.

# 3. Guide: EmojiDictionary Implement Saving


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- In each of these methods, you'll use either a PropertyListEncoder to encode your Emoji object or a PropertyListDecoder to decode it. But to write data to a file, you need to have a file path. Add a static property ArchiveURL to your Emoji class that returns the file path for Documents/emojis.plist. If you need a refresher on how to do this, go back and read this lesson's section on sandboxing.

# 3. Guide: EmojiDictionary Implement Saving


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- Now that you have a path for saving your Emoji object, fill in the method bodies of saveToFile(emojis:) and loadFromFile(), using Emoji.ArchiveURL as the file path. Your saveToFile(emojis:) method should save the supplied emojis array by encoding it and the writing to Emoji.ArchiveURL; and your loadFromFile() method should read the data from the file and decode it as an [Emoji] array, and return the array.

# 3. Guide: EmojiDictionary Implement Saving


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- Create a static loadSampleEmojis() method that will create and return a predefined [Emoji] collection. You can use the list assigned to emojis in EmojiTableViewController as your list of items.

# 3. Guide: EmojiDictionary Save and Load


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- Update emojis to be initialized to an empty collection rather than a large sample collection. When the viewDidLoad() method is called, you should check the Documents/emojis directory for any previously-saved Emoji objects using loadFromFile(). If they're found, assign them to emojis. If not, assign emojis to the result of loadSampleEmojis().

# 3. Guide: EmojiDictionary Save and Load


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- Take a moment to think about when it might be appropriate to save your Emoji objects.
- In this case, the central spot for your data is the emojis array on the EmojiTableViewController—which means it would be appropriate to call saveToFile(emojis:) whenever the emojis property is changed.

# 3. Guide: EmojiDictionary Save and Load


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- Next, think about when it might be appropriate to load your archived Emoji objects. Again, in this simple case, there's really only one point where the archived data will need to be unarchived: when the first view loads. You should already be calling this method in the first view controller's viewDidLoad().

# 3. Guide: EmojiDictionary Save and Load


BARIA VUNGTAU UNIVERSITY
CAP SAINT JACQUES

- By now, your emoji data should be properly saving and loading. Run your app. Try adding a new emoji and tapping the Save button. Then close the app, reopen it, and observe whether or not the emoji persists in the table view. Repeat this process for editing an already-existing emoji.

# References


BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

- Apple – App Development in Swift