CE373 Mobile Application Development

(Special Session on iOS Application Development)

Neel Makhecha (Oct. 14, 2020)

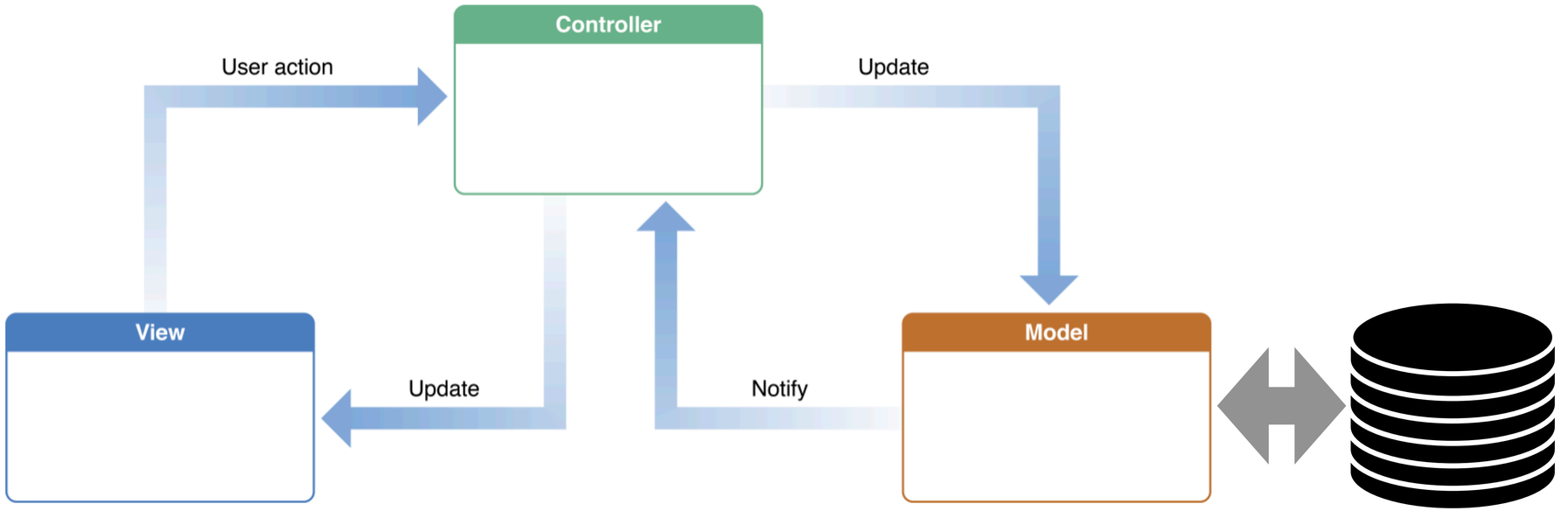
Session Agenda

- Data persistence on iOS (`UserDefaults`)
- Brief introduction to different databases (types) on iOS
- Using SQLite database on iOS

Data Persistence

What is data persistence?

Mechanisms for *data storage* on the disk such that data persist even if the process is terminated or the device is turned off.



Data Persistence

What is data persistence?

Mechanisms for *data storage* on the disk such that data persist even if the process is terminated or the device is turned off.

Data Persistence can be achieved in two different ways (broadly):

Data Persistence on *Remote* storage.

Data Persistence in *Local* storage.

Data Persistence Locally (on-device)

1. UserDefaults
2. Core Data
3. SQLite

UserDefaults

Formal Definition: An interface to the user's defaults database, where you store key-value pairs persistently across launches of your app.

Mainly used to save small data. Eg.: default settings of user (theme enum), whether user is logged in or not (boolean), etc.

Pretty much any datatype (including user defined) can be saved (requires conformance to some protocols)

UserDefaults - General Info

- Generates plist file to store data. (Different for different apps)
- Supports automatic caching of data for better performance.
- Synchronous data write within the process and automatic asynchronous data write for persistence.

Declaration:

```
class UserDefaults : NSObject
```

Storing Objects

UserDefaults can store following data types only:

NSData, NSString, NSNumber, NSDate, NSArray or NSDictionary.

For other data types, the object is first archived to create an instance of NSData.

Available function overloads for setting values

```
func set(Any?, forKey: String)

func set(Float, forKey: String)

func set(Double, forKey: String)

func set(Int, forKey: String)

func set(Bool, forKey: String)

func set(URL?, forKey: String)
```

Code Snippets

#1 Get a UserDefaults object and storing a bool

```
func saveBoolean(_ userDefaults: UserDefaults = UserDefaults.standard, _ val: Bool) {
        userDefaults.set(val, forKey: "key_one")
}
```

#2 Retrieve the same boolean from [standard] UserDefaults container

```
func getBooleanStatus(_ userDefaults: UserDefaults = UserDefaults.standard) -> Bool {
    return userDefaults.bool(forKey: "key_one")
}
```

Code Snippets

#3 Storing objects of user-defined datatypes

```
class Student {
    //...
}


func storeStudentData(_ userDefaults: UserDefaults = UserDefaults.standard, _ object: Student) -> Bool {

    do {
        let encodedData: Data = try NSKeyedArchiver.archivedData(withRootObject: object, requiringSecureCoding: false)
        userDefaults.set(encodedData, forKey: "key_one")
        return true
    } catch {
        NSLog(error.localizedDescription)
    }

    return false
}
```

Code Snippets

#4 Retrieving objects of user-defined datatypes

```
func retrieveStudentObject(_ userDefaults: UserDefaults = UserDefaults.standard) -> Student? {
    guard let retrievedData = userDefaults.data(forKey: "key_one") else {
        NSLog("No Student Data")
        return nil
    }

    do {
        guard let decodedObject = try NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(retrievedData) as? Student else {
            NSLog("Decoded object is nil")
            return nil
        }

        return decodedObject
    } catch {
        fatalError("Thrown by NSKeyedUnarchiver: \(error.localizedDescription)")
    }
}
```

Code Snippets

#5 Observing changes in UserDefaults

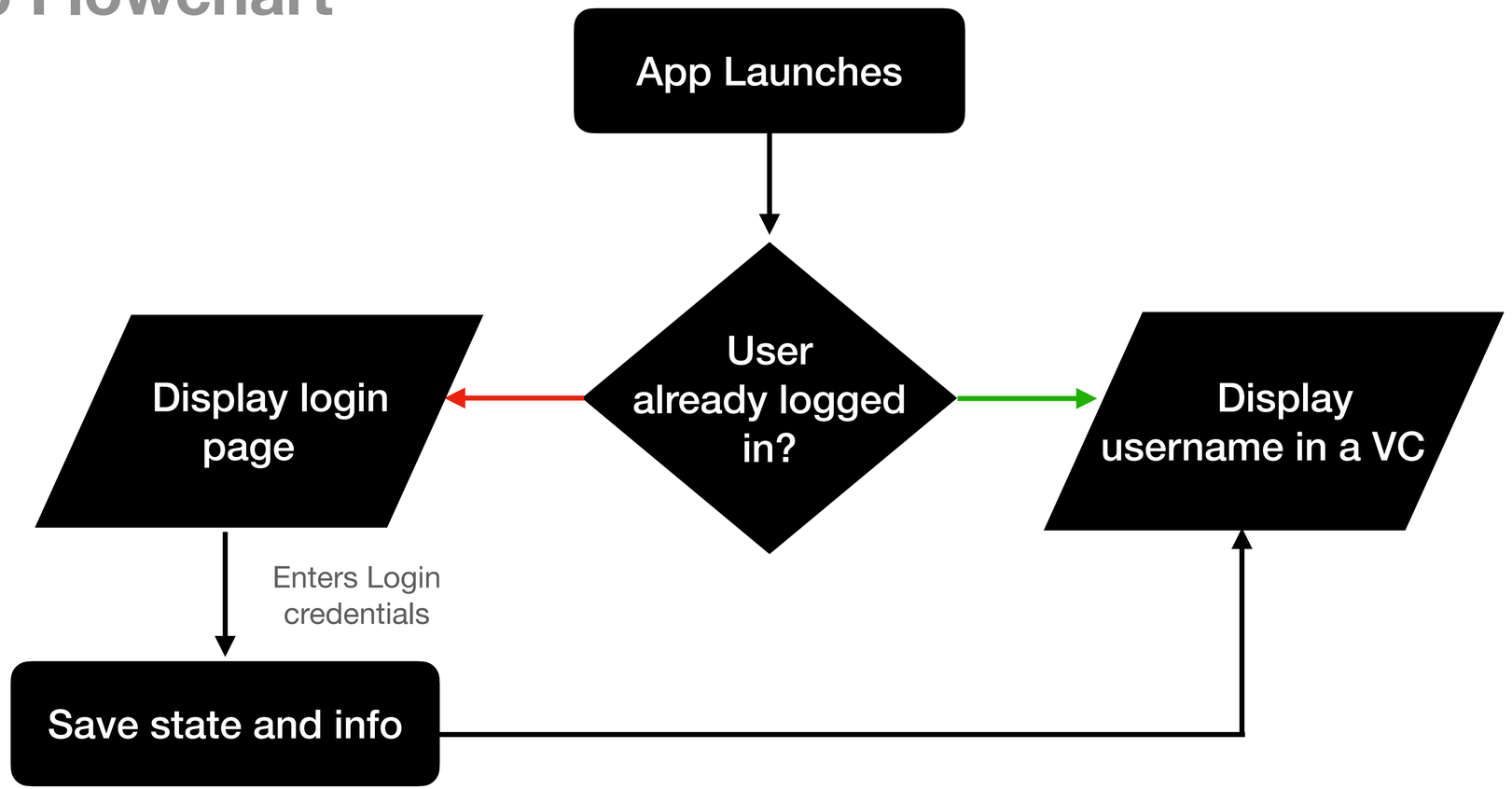
```
//Adding observer for a keyPath prior to any change.
//For example, adding observer for the standard UserDefaults right from viewDidLoad.
override func viewDidLoad() {
    super.viewDidLoad()
    UserDefaults.standard.addObserver(self, forKeyPath: "key_one", options: .new, context: nil)
}

//A sample function that gets executed upon change
func changeBGColor() {
    self.view.backgroundColor = UIColor.blue
}

//Inherited and overridden from NSKeyValueObserving which gets called upon each change with a keyPath as an arg.
override func observeValue(forKeyPath keyPath: String?, of object: Any?, change: [NSKeyValueChangeKey : Any]?,
context: UnsafeMutableRawPointer?) {
    if keyPath == "key_one" {
        self.changeBGColor()
    }
}
```

Quick Demo

Demo App Flowchart



Databases for iOS

(non-exhaustive list)

- Core Data
- SQLite
- Firebase
- Realm
- MySQL
- And much much more…

SQLite on iOS

Neel Makhecha

Code Snippets

#1 Initializing with the database URL

```
var url: URL? = nil

do {
    let baseURL = try FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: nil,
create: false)
    url = baseURL.appendingPathComponent("database.sqlite")
} catch {
    NSLog(error.localizedDescription)
}
```

#2 Opening the SQLite Database (establishing connection)

```
var sqliteDB: OpaquePointer? = nil

if let dbURL = url {
    if sqlite3_open_v2(dbURL.absoluteString.cString(using: String.Encoding.utf8), &sqliteDB, SQLITE_OPEN_CREATE |
SQLITE_OPEN_READWRITE, nil) == SQLITE_OK {

    }
}
```

Code Snippets

#3 Inserting in a table

```
let username = "neel"
var preparedStatement: OpaquePointer? = nil
let sqliteStatement = "INSERT INTO usernames (username) values (\(username))"

if sqlite3_prepare(self.sqliteDB, sqliteStatement, -1, &preparedStatement, nil) == SQLITE_OK {

    if sqlite3_step(preparedStatement) == SQLITE_DONE {
        print("Inserted: \(username)")
    }
}

sqlite_finalize(preparedStatement)
```

Code Snippets

#4 Selecting data from a table

```
var preparedStatement: OpaquePointer? = nil
        let sqliteStatement = "SELECT * FROM usernames"

       if sqlite3_prepare(self.sqliteDB, sqliteStatement, -1, &preparedStatement, nil) == SQLITE_OK {

           while sqlite3_step(preparedStatement) == SQLITE_ROW {
               let id = sqlite3_column_int(preparedStatement, 0)
               let usernameText: UnsafePointer<UInt8> = sqlite3_column_text(preparedStatement, 1)

               let usernameString = String(cString: usernameText)
         }

        }
```

Where to go from here?

- Access demo projects: UserDefaults, SQLite

Q&A

Thank you