1. **How to save Model class in NSUserDefault ?**

Answer: When you want to save your “Model class” into NSUserDefault. You can not save directly to NSUserDefault.

For saving model , you will need to archive the model class into NSData then only you can save it to NSUserDefault.

For retrive it from NSUserDefault, you should unarchive it again.

**Note** : If you want to copy model class then you should implement NSCoding protocol.

From Swift 4, has ”Codable” protocol that will replace NSCoding protocol, it allows you to copy ENUM and Struct also.

For Example:

// Create Model class

class Student: NSObject, NSCoding {

//MARK: Declare Variables

let firstName: String

let lastName: String

init(firstName: String, lastName: String) {

self.firstName = firstName

self.lastName = lastName

}

//MARK: NSCoding Delegates methods

required convenience init?(coder aDecoder: NSCoder) {

let firstName = aDecoder.decodeObject(forKey: "firstName") as! String

let lastName = aDecoder.decodeObject(forKey: "lastName") as! String

self.init(firstName: firstName, lastName: lastName)

}

func encode(with aCoder: NSCoder) {

aCoder.encode(firstName, forKey:"firstName")

aCoder.encode(lastName, forKey:"lastName")

}

}

//Save Model class student into NSUserDefault

let st\_mark = Student(firstName: "Mark", lastName: "Adam")

let userDefault = UserDefaults.standard

let archiveData = NSKeyedArchiver.archivedData(withRootObject: st\_mark)

userDefault.set(archiveData, forKey: "student\_Mark")

userDefault.synchronize()

print(userDefault.value(forKey: "student\_Mark") ?? "Failt to save student model")

// Retrive Model class from NSUserDefaults

let unArchiveData = userDefault.object(forKey: "student\_Mark")

let unArchiveSt\_Mark = NSKeyedUnarchiver.unarchiveObject(with: unArchiveData as! Data) as! Student

print(unArchiveSt\_Mark)

1. **Describe view controller life cycle order when another view controller starts from current view controller ?**

Answer:

UIViewController Life cycle steps:

**Init(coder:) :**

If your view controller is created using the storyboard. Then first **init(coder: )**  ia the intializer that gets called and you have to override it.

Until this time the view of the view controller has stil not initiantiated.so if you try to access view property of UIViewController, the loadView() method will called.

**init(nibName: bundle:) :**

if you will intialize your view controller with nib in place of storyBoard. This method will call.

**loadView() :**

This is method that create view for the view controller. If you want to builds whole interface of the view controller using code then you should override this method.

**viewDidLoad():**

When this method gets called, the view of view controller has been created and all outlets are in place.

If user can load those data that you need to only once when the screen is loaded. This method will call once in life cycle of the view controller.

**viewWillAppear(\_:) :**

If you want to repeat some task every time when you comes on screen then you can do those task in this method.

When you usin this method, keep in mind this will call mutiple times for same instance of the view controller. Usually we use this method to update user inteface with data that might have changed while the viewcontroller was not on the screen.

**viewDidAppear(\_:) :**

This methods called after the view controller appears on screen. You can user it to start animation, video and sound etc.

**didReceiveMemoryWraning() :**

ios device has limited amount of memory and power. When the memory starts to fill up, iOS does not user the limited ghard disk space to move data out of the memory like computer does. Due this reason you are responsible to keep memory footpriint of your app low. If your app consuming more memory iOS will notify you.

Since view controllers perform resoure management , these method deliver through this method to the view controller. In this method you should take action and free some memory. If you ignore it after some thresold valye you app will start crashing.

**viewWillDisappear(\_:) :**

Before transition to the next view controller happens and the origin view contoller gets removed from screen, this method gets called, If you same task that you want to do berfore the orign view controller about to remove from screen,

**viewDidDisappear(\_:) :**

After the view controller removed from the screen this method gets called. You usually use this method to stop task tha are should not run while the view controller is not on screen.

For Example:

Stop listening Notification, observing other objects properties, other sensor and network that is not needed any more.

**Deinit():**

Like every other object, before a view controller is removed form memory, it gets deinitialized. You usually use this method to clean reasoures of view controller has allocated that are not freed by ARC.

1. **How to support multi – resolution and multi-language in iOS app?**

Answer:

**Support multi resolution:**

In iOS there are many resolutions devices available.

Like 3.5, 4, 4.7 ,5.5, 7.9, 9.7, 10.5 12. And 1.32 inch etc.

So, previously we are using autoresizing technique to supoort multi resolution app. But after release of iOS 6 introduce a new interface tool for iOS developer called “**Autolayout**”.

**“**Autolayout is a constaint-based layout system. It allows to developers to create an adaptive interface that responds appropriately to changes in screen size and device orientation.**”**

So using Autolayout you can create multi resolution app easily instead using simple or autoresizing.

After iOS 8 release new technolgy comes in picture called “**Size Classes**”. Basically it is allows to user tocustomize your app for given device class. Based on its orientation and screen size.

There is two size classes: Vertical and Horizontal size classes.

|  |  |  |
| --- | --- | --- |
|  | **Vertical Size Class** | **Horizontal Size Class** |
| iPad Portrait | Regular | Regular |
| iPad Landscape | Regular | Regular |
| iPhone Portrait | Regular | Compact |
| iPhone Landscape | Compact | Compact |

With help of autolayout and size classes you can create multi resolution apps for iPhone, iPad and iWatch.

**Support multi-language :**

For support multi language in iOS app. We need to consider some thing while developing app. While you are using any images should not have any text on images. So you can localized those text easily, you don’t need to repeat same images same for different language.

For support different language we are creat different localization file for different languages.

There is some steps to convert ios app for Internationalisation and Localisation.

**Step 1:**

Click onproject - > info -> scroll down -> localization -> click on “+” and add your required language -> check as required -> finish

**Step 2:**

Click on “Supporting Files” -> right click -> New File -> select Resource on Left-side list -> select “String Files” -> next -> name it “Localizable” -> create

**Step 3:**

Click on newly created file -> go to utlities -> file inspector -> click on localize.. -> check all language

**Step 4:**

open Localizable.strings(English) write “Ok” = “OK”;

here “Ok” is a key “OK” is a value. You may change them as your wish.

Open other language.stiring write value for “OK” as per language translation.

**Step 5:**

NSSting \*message = NSLocalizedString(@“OK”,nil);

NSLog(@”%@”,str);

Now you can go to settings of device -> general -> International -> language -> select said language ot the language you have choosen.Run you app.

1. **Explain the error handling and exception handling ?**

**Answer :**

In objective c, for error handling we are using NSError class. We pass an error object reference to the method if you are getting object of NSError means there are some error occur during method execution.

In Swift errors are represented by values of types that conforms the Error protocol. This empty protocol indicates that a type can be used error handling.

Normally Swift are very useful to modling your error handling conditions that having associated values allowing for additional information about the nature of an error to be communicated.

For Example:

enum Database\_Error: Error {

// An error connecting to the database

case Connection\_Error(String)

// The database generated an error for query execution.

case Query\_Execution\_Error(String)

// An error in Transcation

case Transcation\_Error(String)

// No result found from query execution.

case NoResult\_Error(String)

// A syntex error while constucting the query.

case Syntex\_Error(String)

// The query or its execution is not supported.

case NotSupported\_Error(String)

}

extension Database\_Error: CustomStringConvertible{

public var description: String {

switch self {

case .Connection\_Error(let error):

return error

case .Query\_Execution\_Error(let error):

return error

case .Transcation\_Error(let error):

return error

case .NoResult\_Error(let error):

return error

case .Syntex\_Error(let error):

return error

case .NotSupported\_Error(let error):

return error

}

}

}

Throwing an error lets you indicate that something unexpected happened and the normal flow of execution can’t continue.

For Example:

**throw Database\_Error(“Error in database connection”)**

**Handling Errors :**

When an error is thrown, some surrounding piece of code must be responsible for handling the error. For example correcting the problem, trying an alternative approach, or informing the user of the failure.

There are four ways to handle some errors in swift.

* You can propagate the error from a function to the code that calls the function.
* Handle the error using **“do-catch”** statement,
* Handle the error as an optional value
* Asset that the error will not occur.

Each section is described below;

func dbOperation(sqlQuery: String, database: FMDatabase?, isReturnCollection:Bool, primaryKeyValue: AnyObject,isReturnModels:Bool)throws -> [Any]?

func dbOperation(sqlQuery: String, database: FMDatabase?, isReturnCollection:Bool, primaryKeyValue: AnyObject,isReturnModels:Bool) -> [Any]?

func dbOperation(sqlQuery: String, database: FMDatabase?, isReturnCollection:Bool, primaryKeyValue: AnyObject,isReturnModels:Bool)throws -> [Any]? {

do {

return try dbOperation(sqlQuery: sqlQuery, database: database, isReturnCollection: isReturnCollection, primaryKeyValue: primaryKeyValue, parameters: nil,isReturnModels:isReturnModels)

}catch let error as Database\_Error {

throw error

}

}