

# How to use SQLite Database in Swift

[How to use SQLite Database in Swift](#)

[iOS](#)

---

29 September 2014, 04:38 AM - By | Krupa Kadecha

# HOW TO USE **SQLITE** **DATABASE** IN **SWIFT**



**Free Download Full Source Code!!!**

**Objective:** In this tutorial we cover how to use SQLite database in Swift. This is an example showing how to create a application that performs Insert, Update, Delete operation on a table called StudentInfo.

To perform these operations we are using FMDB. FMDB aims to be fully featured wrapper for SQLite. You can clone it from <https://github.com/ccgus/fmdb>.

Following are the steps, which explains how perform operation like insert, update, delete on table in the SQLite Database.

## **Step - 1: Create Single View Application**

Open Xcode 6 and create one new project with single view application template as shown in below figure.

create single view application



In the next step enter DataBaseDemo as a product name and also select swift as a language as shown in below figure.

select swift as a language



**Step - 2: Design User Interface** Design user interface as below screenshot image. Define all buttons action method so that we can perform action on the tap of particular button.

design user interface



### Step - 3: Creating SQLiteDatabase

To create database you can use SQLite Manager Add-ons of the Firefox browser.

Open Firefox browser -> Tools -> SQLite Manager.

Create Database and insert table as per your requirement. In our case we are creating Database named DataBaseDemo.sqlite and insert table named StudentInfo with two fields student\_rollno and student\_name.

Copy your database file in application bundle.

**Step - 4: Creating model Class** Add new Class file which is subclass of NSObject name it StudentInfo. Declare properties as per number of fields having in your database table as below.

```
class StudentInfo: NSObject {  
    var studentRollNo: String = String()  
    var studentName: String = String()  
}
```

### Step - 5: Add Bridging Header

FMDatabase is in Objective C so if you want to use it in your swift project you need a bridging header file in your project. Import FMDatabase.h file in that file. You can refer our blog “How to use Objective C code in Swift” for this.

**Step - 6: Creating Database** Copy database file to application’s document directory by calling below method in applicationDidFinishLaunching method of AppDelegate, pass database name as argument in copyFile method.

```

class func copyFile(fileName: NSString) {
    var dbPath: NSString = getPath(fileName)
    var fileManager = NSFileManager.defaultManager()
    if !fileManager.fileExistsAtPath(dbPath) {
        var fromPath: NSString = NSBundle.mainBundle().resourcePath.stringByAppendingPathComponent(fileName)
        fileManager.copyItemAtPath(fromPath, toPath: dbPath, error: nil)
    }
}

```

### Step - 7: Initialize Database object

Add a swift file into your project name it ModelManager.

Declare a sharedInstance of type ModelManager outside the ModelManager class block.

```
let sharedInstance = ModelManager()
```

Declare FMDatabase object and initialize database object as below inside class variable of ModelManager.

```

var database: FMDatabase? = nil

class var instance: ModelManager {
    sharedInstance.database = FMDatabase(path: Util.getPath("DataBaseDemo.sqlite"))
    var path = Util.getPath("DataBaseDemo.sqlite")
    println("path : \(path)")
    return sharedInstance
}

```

### Step - 8: Querying the Database

#### INSERT:

Add addStudentData method to ModelManager class.

In that method open the database by calling open method of the FMDatabase class. Call executeUpdate method of the FMDatabase class and pass SQL Insert query and its parameters as argument. Close the database by calling close method of FMDatabase.

```

func addStudentData(studentInfo: StudentInfo) -> Bool {
    sharedInstance.database!.open()
    let isInserted = sharedInstance.database!.executeUpdate("INSERT INTO StudentInfo (student_rollno, student_name) VALUES (?, ?)", withArgumentsInArray: [studentInfo.studentRollNo, studentInfo.studentName])
    sharedInstance.database!.close()
    return isInserted
}

```

Call this method from the action method of the insert button. And pass StudentInfo model class object which contains studentRollno and studentName.

```

@IBAction func btnInsertClicked(sender: AnyObject) {
    var studentInfo: StudentInfo = StudentInfo()
    studentInfo.studentRollNo = txtRollNo.text
    studentInfo.studentName = txtName.text

    var isInserted = ModelManager.instance.addStudentData(studentInfo)
    if isInserted {
        Util.invokeAlertMethod("", strBody: "Record Inserted successfully.", delegate: nil)
    } else {
        Util.invokeAlertMethod("", strBody: "Error in inserting record.", delegate: nil)
    }
    txtRollNo.text = ""
    txtName.text = ""
    txtRollNo.becomeFirstResponder()
}

```

**UPDATE:** Add updateStudentData method to ModelManager class. In that method open the database by calling open method of the FMDatabase class. Call executeUpdate method of the FMDatabase class and pass SQL Update query and its parameters as argument. Close the database by calling close method of FMDatabase.

```
func updateStudentData(studentInfo: StudentInfo) -> Bool {
    sharedInstance.database!.open()
    let isUpdated = sharedInstance.database!.executeUpdate("UPDATE StudentInfo SET student_name=? WHERE student_rollno=?", withArgumentsInArray: [studentInfo.studentName, studentInfo.studentRollNo])
    sharedInstance.database!.close()
    return isUpdated
}
```

Same as insert call this method from the action method of update button.

```
@IBAction func btnUpdateClicked(sender: AnyObject) {
    var studentInfo: StudentInfo = StudentInfo()
    studentInfo.studentRollNo = txtRollNo.text
    studentInfo.studentName = txtName.text

    var isUpdated = ModelManager.instance.updateStudentData(studentInfo)
    if isUpdated {
        Util.invokeAlertMethod("", strBody: "Record updated successfully.", delegate: nil)
    } else {
        Util.invokeAlertMethod("", strBody: "Error in updating record.", delegate: nil)
    }
    txtRollNo.text = ""
    txtName.text = ""
    txtRollNo.becomeFirstResponder()
}
```

**DELETE:** Add deleteStudentData method to ModelManager class. In that method open the database by calling open method of the FMDatabase class. Call executeUpdate method of the FMDatabase class and pass SQL Delete query and its parameters as argument. Close the database by calling close method of FMDatabase.

[SAY HELLO!](#)



The AppGuruz

```
func deleteStudentData(studentInfo: StudentInfo) -> Bool {
    sharedInstance.database!.open()
    let isDeleted = sharedInstance.database!.executeUpdate("DELETE FROM StudentInfo WHERE student_rollno=?", withArgumentsInArray: [studentInfo.studentRollNo])
    sharedInstance.database!.close()
    return isDeleted
}
```

Same as insert and update call this method from action method of delete button.

```
@IBAction func btnDeleteClicked(sender: AnyObject) {
    var studentInfo: StudentInfo = StudentInfo()
    studentInfo.studentRollNo = txtRollNo.text
    studentInfo.studentName = txtName.text

    var isDeleted = ModelManager.instance.deleteStudentData(studentInfo)
    if isDeleted {
        Util.invokeAlertMethod("", strBody: "Record deleted successfully.", delegate: nil)
    } else {
        Util.invokeAlertMethod("", strBody: "Error in deleting record.", delegate: nil)
    }
    txtRollNo.text = ""
    txtName.text = ""
    txtRollNo.becomeFirstResponder()
}
```

**DISPLAY:** Add getAllStudentData method to ModelManager class. In that method open the database by calling open method of the FMDatabase class. Call executeQuery method of the FMDatabase class and pass SQL Select query and its parameters as argument. Close the database by calling close method of FMDatabase.



```
func getAllStudentData() {
    sharedInstance.database!.open()
}
```

```
var resultSet: FMResultSet! = sharedInstance.database!.executeQuery("SELECT * FROM StudentInfo", withArgumentsInArray: nil)
var rollNoColumn: String = "student_rollno"
var nameColumn: String = "student_name"
if resultSet {
    while resultSet.next() {
        println("roll no : \(resultSet.stringForColumn(rollNoColumn))")
        println("name : \(resultSet.stringForColumn(nameColumn))")
    }
}
sharedInstance.database!.close()
}
```

Call this method from the action method of Display button.

```
@IBAction func btnDisplayRecordClicked(sender: AnyObject) {
    ModelManager.instance.getAllStudentData()
}
```

Output

insert-a-record1	insert-a-record-in-db	update-
		

I hope you found this blog helpful while Using **SQLite Database in Swift**. Let me know if you have any questions or concerns regarding **iOS**, please put a comment here and we will get back to you ASAP.

Got an Idea of **iPhone App Development**? What are you still waiting for? [Contact us](#) now and see the Idea live soon. Our company has been named as one of the best [iPhone App Development Company in India](#).

# Krupa Kadecha



I am [iOS developer](#) with an aspiration of learning new technology and creating a bright future in Information Technology.

PREVIOUS POST  
[HTML5 Tutorial: SVG Shape Morphing](#)

NEXT POST  
[Rigging in Maya: How To Set Reverse Foot For Human Walk Cycle With Easy Techniques](#)

0 Comments

www.theappguruz.in

Login ▼

♥ Recommend 🔗 Share

Sort by Best ▼

Start the discussion...

Be the first to comment.

✉ Subscribe

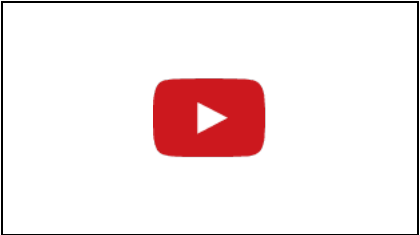
D Add Disqus to your site

🔒 Privacy

Search



NEW TABLE TENNIS GAME PLAY TRAILER



CATEGORIES

- 3DS Max
- Android
- Blog
- Graphics
- iOS
- Maya
- Unity
- Web

RECENT BLOG POSTS

- [Rigging in Maya: How To Set Reverse Foot For Human Walk Cycle With Easy Techniques](#)
- [Tapjoy Implementation In Unity](#)
- [Tips for 3D Character Modelers](#)
- [Using Render to Texture For Game Environment](#)
- [Basic Information to Implement Nextpeer for Multiplayer Game](#)
- [Game Main Menu Design](#)
- [Layout Element in Unity UI 4.6](#)
- [Dynamic Grid Generator](#)
- [Digital Portrait Painting Tutorial](#)
- [Smooth Camera Orbit Movement in Unity](#)
- [Indirect Lights in Computer Graphics](#)
- [Extrapolate Position in Unity](#)

TAGS

# Get in touch

## Receive our email newsletter

Subscribe to our ever growing technical blog if you want to learn more about mobile gaming & apps.

My email is

[Sign up](#)

## Get in touch

### India

T +91 9033 915992

[ravi@theappguruz.com](mailto:ravi@theappguruz.com)

3rd Floor, Shivalik 5,  
Gondal Road,  
Rajkot – 360 002.

### USA

T +305-3087507

9172 Collins Av Surfside,  
Fl 33154

## Start a Project

Like what you see? We'd love to hear from you!

[GET IN TOUCH](#)

- **Game Development**

- [3D Game Development](#)
- [IOS Game Development](#)
- [Android Game Development](#)
- [Game Development](#)

- **App Development**

- [Android App Development](#)
- [IOS App Development](#)
- [Application Development](#)
- [App Design](#)

- **Tools**

- [Android - JSON](#)
- [CSS Animation](#)

- **Company**

- [About us](#)
- [Vision & Values](#)
- [Career](#)
- [Client Testimonials](#)
- [Our Team](#)
- [Contact Us](#)

© 2012-2015 The App Guruz. All rights reserved