

Getting Started with Microsoft Azure Mobile Services

Microsoft built Microsoft Azure Mobile Services to be a great place to build and host the backend for any mobile app. Mobile Services provides a scalable and secure backend that can be used to power apps on any platform—iOS, Android, Windows, Mac, or Linux. With Mobile Services, it's easy to store app data in the cloud, authenticate users, and send push notifications, as well as add your custom backend logic in C# or Node.js.

In this lab, you'll build your first Mobile Services solution. You'll start with an iOS project provided by Mobile Services. The starting solution provided by Mobile Services is for a simple task list or "TO DO" application. This app runs as an iOS application that talks to a Node.js web serviced hosted up in Microsoft Azure

There are three exercises in this lab:

1. Create your first Azure Mobile Service
2. Create an Xcode Project
3. Explore Your Xcode Project

Prerequisites

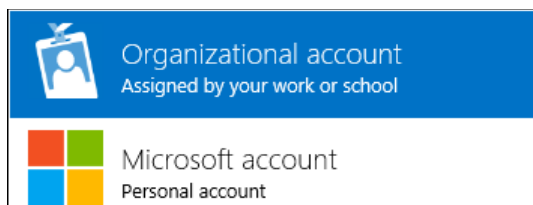
- Xcode (4.4 or later is required. This walkthrough uses Xcode 6.2 on OS X Yosemite)
- A Microsoft Azure account
- An Internet connection

Expected duration: 30 minutes

Exercise 1: Create your first Azure Mobile Service

In this exercise, you'll create your first Azure Mobile Service (AMS), using the Microsoft Azure portal.

1. Start your web browser (this walkthrough uses Safari on OS X Yosemite).
2. Navigate to <http://manage.windowsazure.com/> and log in. You'll need to choose an account that has the permissions needed to complete these labs.

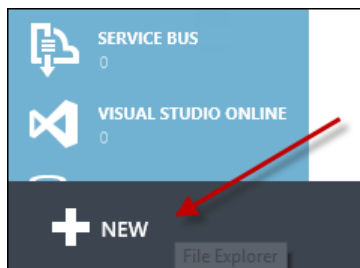


3. Once logged into the management portal, select **MOBILE SERVICES**.

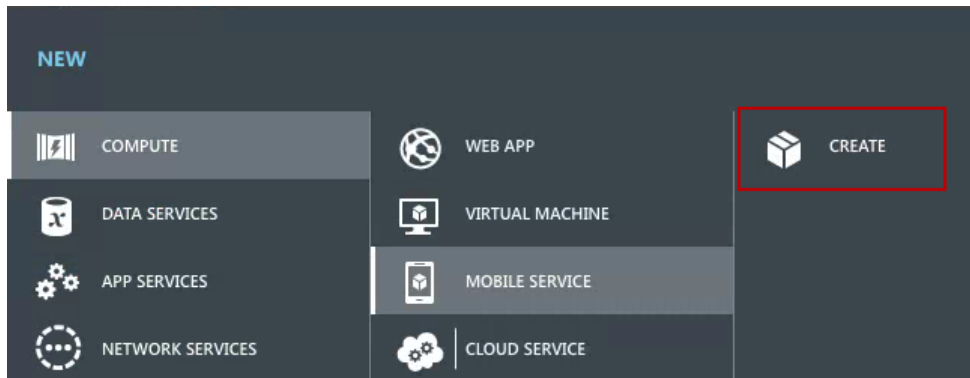
Microsoft Azure provides an almost dizzying array of products and services. You'll find MOBILE SERVICES on the left menu.



4. Click the **+ NEW** button at the bottom of the portal.



5. In the fly out menu, click **CREATE**.



6. In the **Create a Mobile Service** wizard (page 1), provide a valid URL prefix.

A screenshot of the "Create a Mobile Service" wizard page. The page has a header "NEW MOBILE SERVICE" and a main title "Create a Mobile Service". Below the title, there is a section labeled "URL" with a text input field. The input field is empty, and the text ".azure-mobile.net" is visible below it.

This prefix will form part of your mobile service's URL, so it must be unique. The wizard will prevent you from picking a name that someone else is already using

The prefix must be between 2 and 48 characters in length. It can only contain letters, numbers, and hyphens. It cannot be a reserved word. The first character must be a letter and the last character must be either a letter or number.

TIP Microsoft Azure will use the URL as a prefix for objects you will use in Visual Studio as part of these labs. Choose something reasonably short and easy to type. Warning it may modify it a little—for example turn hyphens into underscores.

Azure Mobile Services will use this prefix in quite a few places. Whenever this lab references your project, it will use the syntax <<Project Name>>.

For example, assume you named your project **AMSdemo**. If you saw the step *Open the folder named << PN>>*, you should interpret that as *Open the folder named AMSdemo*.

7. Choose **Create a free 20 MB SQL database**.

NOTE You only get one free 20 MB SQL database per Azure Subscription.

8. Choose a **Region** that makes sense for you based on where you're doing the lab.
9. For the **Backend**, choose **JavaScript**.

The following image shows a filled in wizard page.

NOTE You will most likely *not* be able to choose **amsl** for your URL.

NEW MOBILE SERVICE

Create a Mobile Service

URL

amsl ✓

.azure-mobile.net

DATABASE

Create a free 20 MB SQL database

REGION

West US

BACKEND

JavaScript

☐ CONFIGURE ADVANCED PUSH SETTINGS ?

→ 2

10. Ensure that the **Configure Advanced Push Settings** button is **not** checked.
11. Click the arrow button to continue.



12. Accept the default database **Name**.
13. Select a **New SQL database server** from the **Server** option.

As part of this lab, you will create a new SQL Database instance and server. You can reuse this new instance later. You can administer it as you would any other SQL Database instance in Microsoft Azure. If you already have a database in the same region as the new mobile service, you can instead choose Use existing Database and then select that database. However, you need to know the administrator information to complete the lab. It is not recommended that you use a database in a different region because of additional bandwidth costs and higher latencies.

14. Enter a name you'll remember in the **Server Login Name** field.
15. Enter a strong password you'll remember in the **Server Login Password** and **Confirm Password** fields.

IMPORTANT Make sure you remember or write down this information.

16. The **Region** should be the same as your Service's Region that you choose on the previous page.

17. Make sure *Configure Advanced Database Settings* is not checked.

NEW MOBILE SERVICE

Specify database settings

NAME
amsl_db ✓

SERVER
New SQL database server

SERVER LOGIN NAME
sqlAdmin ?

SERVER LOGIN PASSWORD
.....

CONFIRM PASSWORD
.....

REGION
West US

☐ CONFIGURE ADVANCED DATABASE SETTINGS

1

← ✓



18. Click the check mark button to continue. Azure will create your service and database server.

19. Wait for the **Status** column to show **Ready**.

Microsoft Azure

mobile services

NAME	STATUS
amsl	→ ✓ Ready

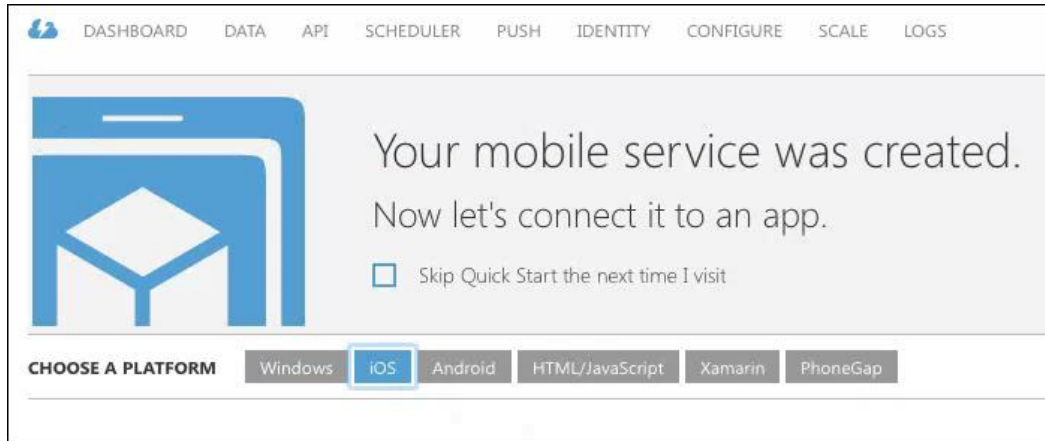
Your service is now ready to go.

Now it's time to get working with XCode.

Exercise 2: Create an Xcode Project

In this exercise, you will use Microsoft Azure to create a table in your SQL Database as well as iOS Xcode Project.

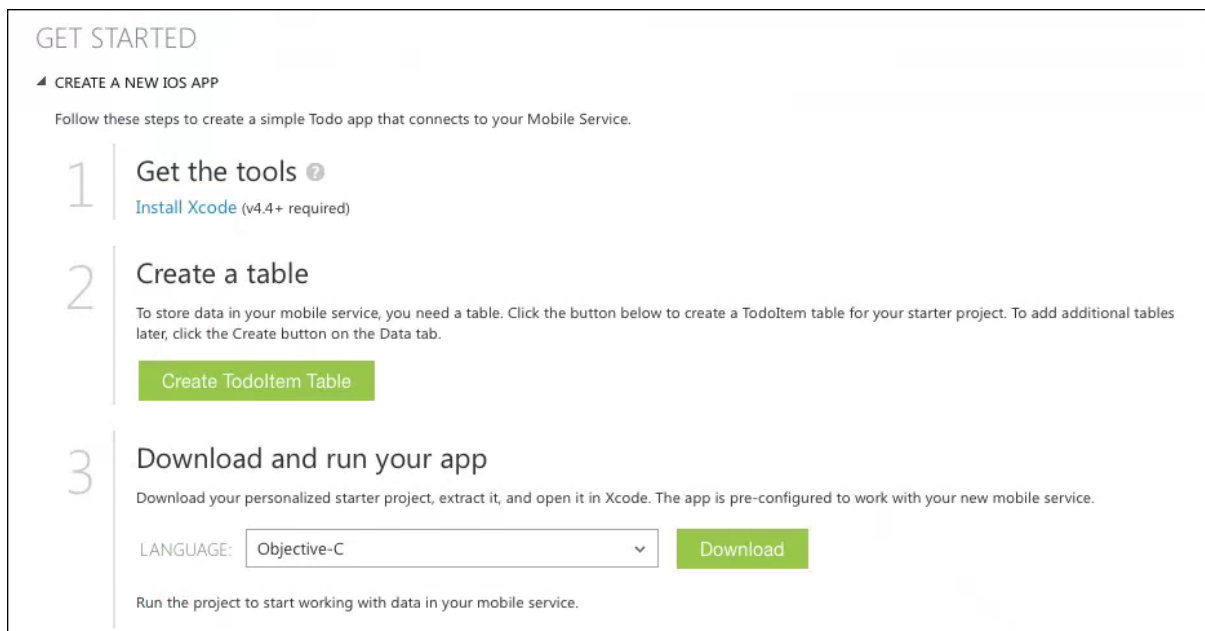
1. From the **Mobile Services** page, click the name of your Mobile Service. Your browser navigates to the **Quick Start** page of your service.
2. You'll want to ensure **iOS** is selected for **CHOOSE A PLATFORM**.



3. Expand **Create a new iOS app** under the GET STARTED section.

You find three steps that you need to perform.

It's assumed you've already installed Xcode. You'll need to do steps two and three, described below.



4. Click the **Create TodoItem Table** button.

Microsoft Azure will create this new table in your SQL Database you created earlier. This will only take about 30 to 60 seconds.

2

Create a table

To store data in your mobile service, you need a table. Click the button below to create a TodoItem table for your starter project. To add additional tables later, click the Create button on the Data tab.

[Create TodoItem Table](#)

We have created the TodoItem table for you.

5. Next ensure **Objective-C** is set for the Language. (Step 3 in the list of steps on the screen).
6. Click the **Download** button.
Azure generates a project. Safari downloads the project as a folder.
7. Use **View | Downloads** and then select the downloaded folder and choose **Show in Finder**. It will be named the same as your Azure Mobile Service.
8. Copy the folder to a location you'd like to work on it at, such as your Profile or Documents folder.

Exercise 3: Explore Your Xcode Project

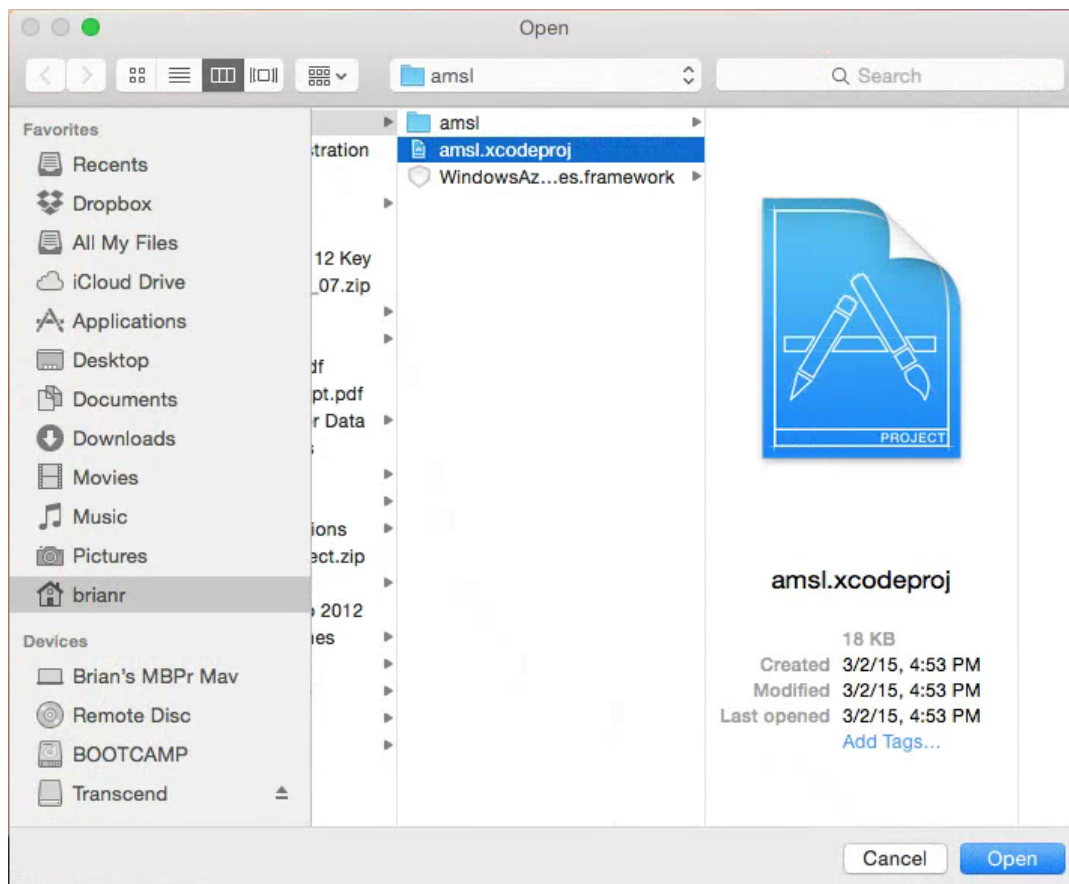
In this exercise, you will examine the project created by Azure Mobile Services and run the sample in the iOS Simulator.

1. If necessary, start Xcode. As mentioned at the beginning, this walkthrough was done on a Mac running OS X Yosemite with Xcode 6.3. Your application screens could look a bit different if you're not running the same version.



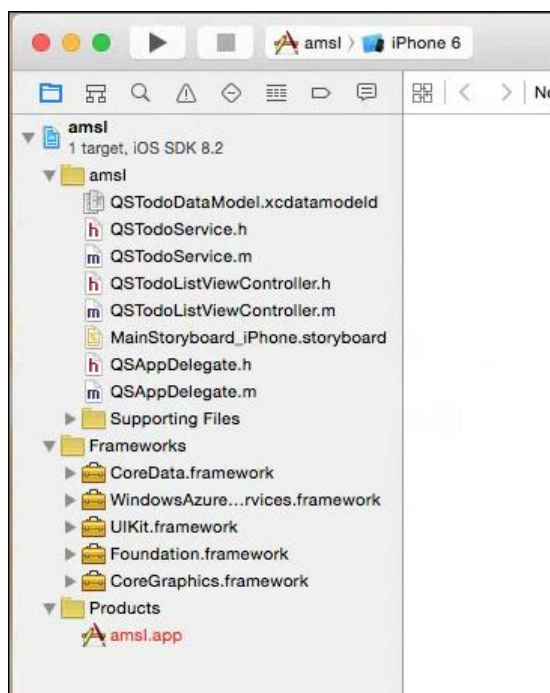
2. Select **File | Open**.
3. In the **Open** dialog, navigate to the location where you put the downloaded folder.

4. Select the Xcode project file—it will be the name of your Azure Mobile Service—and select **Open**.



Xcode opens your project in the workspace window.

5. Expand the folders in the Navigator area to see the various parts of your project.

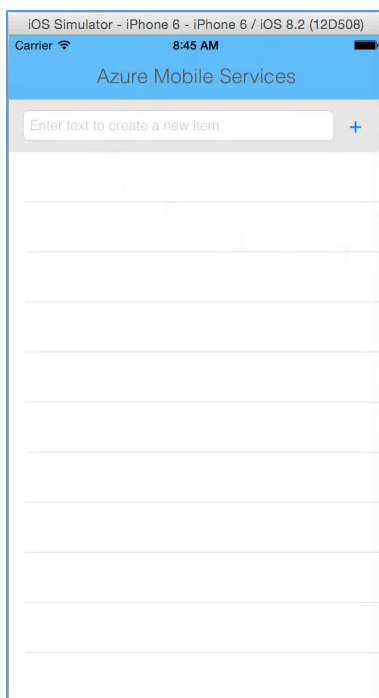


NOTE Because these notes are written using a test project, your Azure Mobile Services name won't match the test project. So, in the instructions that follow in this lab, if you see <<PN>>, you need to replace that with the name of your solution.

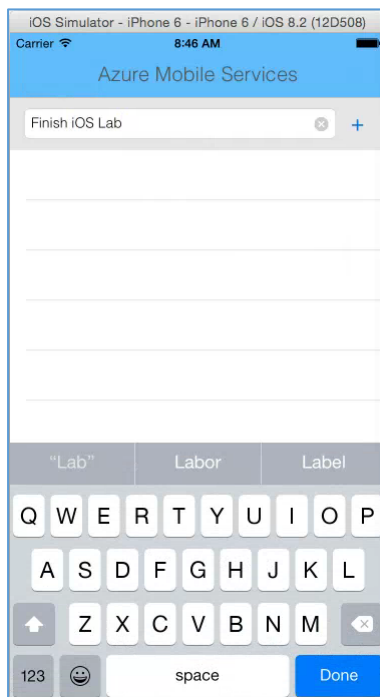
For example, if the service name is *myams*, and the notes says "Expand the <<PN>> folder", you need to look for the *myams* folder. On the other hand you might see an object like <<PN>>**Service** without a space. This means you need to find an object named *myamsService* in the code or user interface.

6. Select **Product** | **Run**.

Xcode will compile your project and start it in the iOS Simulator



7. Once it's started, enter a "to do" item, like **Finish iOS Lab** and then click the + symbol.

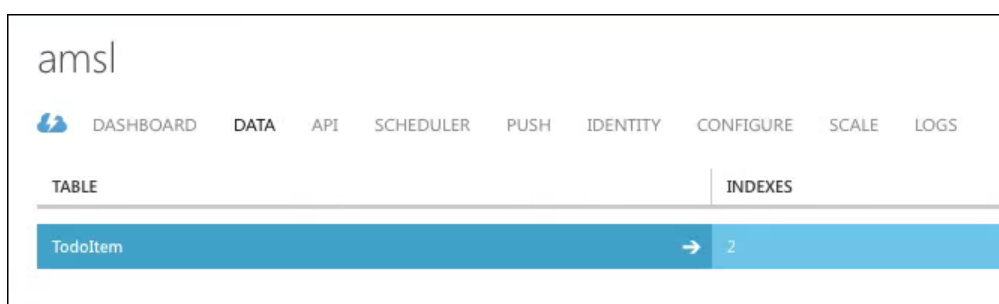


The data you entered is saved on the device and then sent up to Microsoft Azure and stored in the SQL Database.

8. Switch to Safari and your Azure Mobile Services home page.
9. Click the Data link.



10. Click the **Todoitem** table to browse the data.

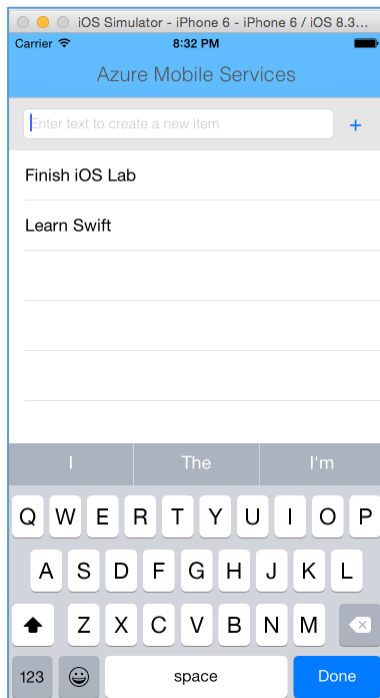


11. You should see your data in the list.



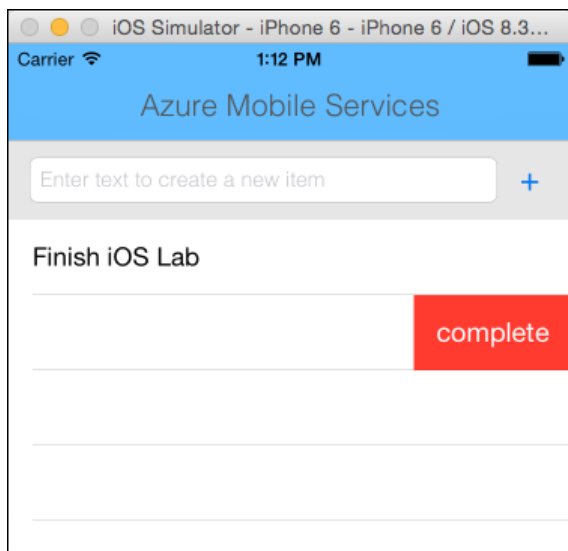
12. Switch back to the **iOS Simulator**.

13. Add a second item like **Learn Swift**.



14. If you want, you can jump back to Safari to verify the new item is there.

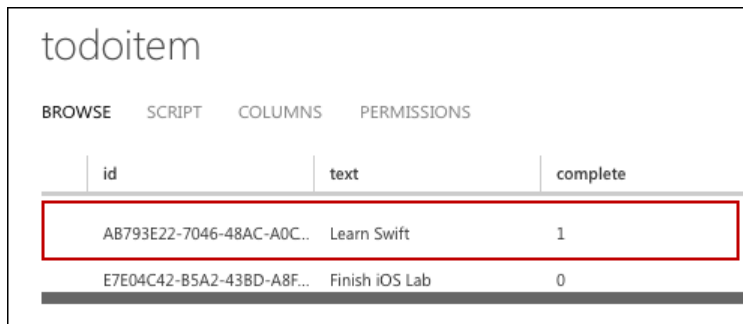
15. Now, “swipe” left on the *Learn Swift* to do item.



16. Click the **complete** button to mark this item done.

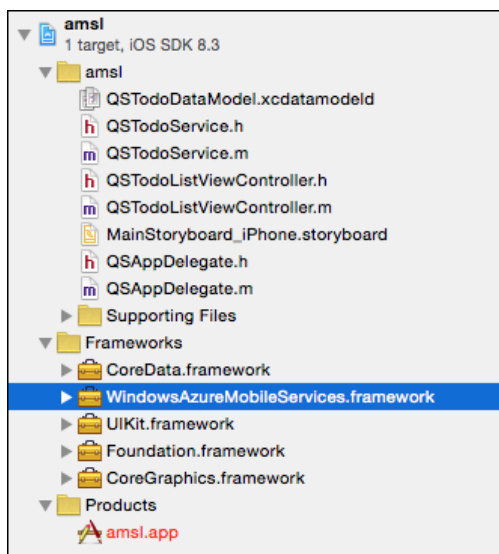
17. Switch to Safari and your Azure Mobile Services page.

18. Refresh the data page. Notice now there is a value in the complete column for the item you just completed.



id	text	complete
A8793E22-7046-48AC-A0C...	Learn Swift	1
E7E04C42-B5A2-43BD-A8F...	Finish iOS Lab	0

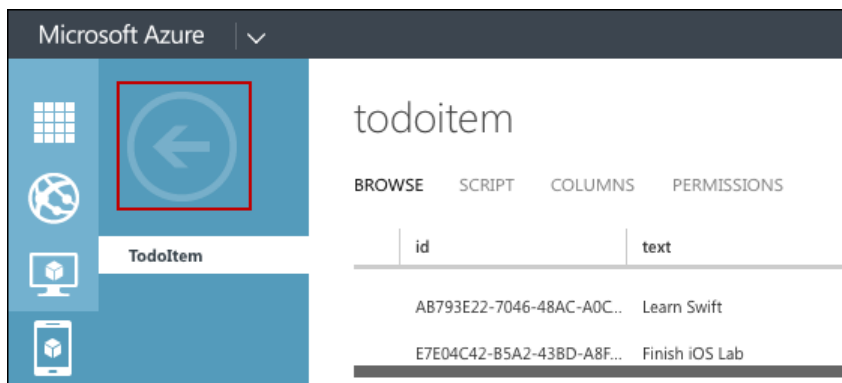
19. Switch back to Xcode and stop debugging.
20. In the Navigator window, expand the **Frameworks** folder.



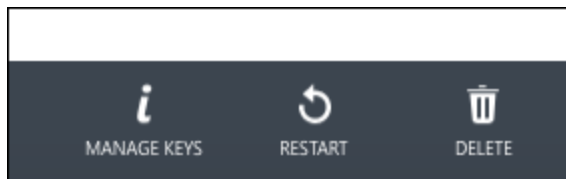
The project you download from Microsoft Azure automatically includes a reference to the Mobile Services SDK. This SDK takes care of the heavy lifting making REST calls to Azure and parsing the JSON data. You can of course handle this yourself if you have specialized needs.

21. If you open up the **QSTodoService.h** file, you'll see it defines an interface with a couple of members including **addItem**, **completeItem**, and **syncData**.
22. Open **QSTodoService.m** and you'll find the implementation details.
23. If you look at the **Init** member you'll find where your app stores the URL for your mobile service as well as your application's key.
24. Switch to Safari and your Azure Mobile Service.

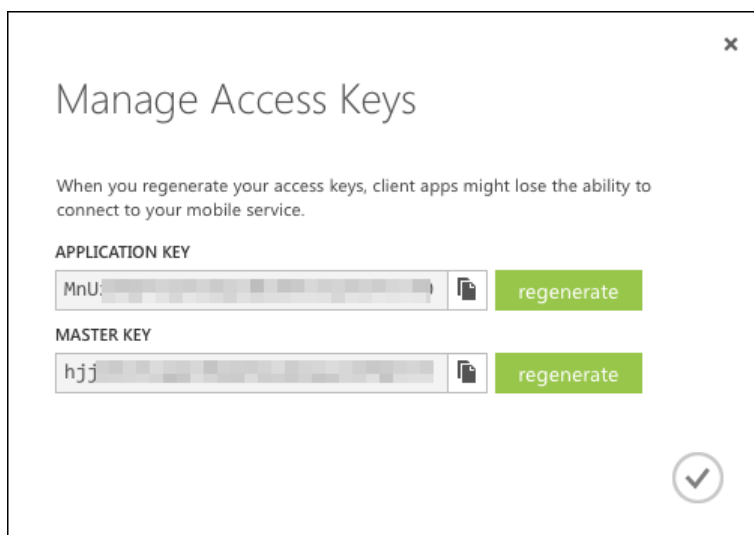
25. Click the arrow to return to the home page.



26. Down at the bottom of the page, there is a Manager Keys button. Click it.



A dialog opens showing you your application key as well as a master key.



The **application key** is generated by Mobile Services and is distributed with your app. Because Mobile Services is a publicly available Azure service, table resources are accessed by HTTP requests. When a client requests resources and provides an application key, Mobile Services validates the key. If the client has supplied the correct application key, access is allowed to any tables that require the application key as authorization.

The **master key** is used for administrator access to the service and to disable scripts. This key is an important secret that should not be distributed. This key can also be used to limit table access only to registered scripts executed by the service or to administrators.

27. At this point you can continue to investigate the generated code as well as additional features of Azure Mobile Services.

You created your first Azure Mobile Service and were able to use the iOS Simulator to run a simple application powered by the Azure Mobile Services SDK to exchange data via REST with the service in the cloud.