***Giáo trình***

**Lập trình Ứng dụng Di động**

**GS. Phan Viết Hoàng**

**Bài học 7 (Buổi 7)**

**Xử lý dữ liệu local**

# References

# ‘Building Mobile App with Ionic 2’*, Josh Morony, 2016*

**Cất dữ liệu**

**(Lesson 10: Saving Data)**

Let’s say you’ve create an Ionic 2 application where users can create shopping lists. A user downloads your application, spends 5 minutes adding their list and then closes the application… and all their data is gone.

Quite often when creating mobile applications you will want to store data that the user can retrieve later. A lot of the time this is done by storing the data somewhere that can be accessed through a remote server (think Facebook, Twitter etc.) which would require an Internet connection and fetching the data remotely (which we will discuss in the next lesson). In many cases though, we may also want to store some data locally on the device itself.

There’s a few reasons we might want to store some data locally:

* The application is completely self contained and there is no interaction with other users, so all data can be stored locally
* We need to store data locally for some specific functionality like remembering logged in users
* We could skip unnecessary calls to a server by storing preference settings locally
* We could skip unnecessary calls to a server by caching other data locally
* We want to sync online and oﬄine data so that the user can continue using the application even when they are oﬄine (Evernote is a good example of this)

HTML5 applications run in the browser, so we don’t have access to the storage options that native applications do. We do still have access to the usual browser storage options that websites have access to though like **Local Storage**, **Web SQL** (deprecated) and **IndexedDB**. These options might not always be ideal (for reasons I will cover shortly), but we can also access native data storage by using Cordova which can overcome the shortfalls of browser based data storage.

There are plenty of diﬀerent options out there for storing data locally, but we are going to focus on the main ones when it comes to Ionic 2 applications, and they are **Local Storage** and **SQLite**.

**Local Storage**

This is the most basic storage option available, which allows you to store up to **5MB** worth of data in the users browser. Remember, Ionic 2 applications technically run inside of an embedded browser.

Local storage gets a bit of a bad rap, and is generally considered to be unreliable. I think the browsers local storage can be a viable option and it is reasonably stable and reliable, but, it is possible for the data to be wiped, which means for a lot of applications it’s not going to be a great option. Even if it worked 99% of the time, that’s still not good enough for storing most types of data.

In general, you should only use it where data loss would not be an issue, and it shouldn’t ever be used to store sensitive data since it can be easily accessed. One example of where local storage might be a suitable option is if you wanted to store something like a temporary session token. This would allow you to tell if a user was already logged in or not, but if the data is lost it’s not really a big deal because the user will just need to enter in their username and password again to reauthenticate.

If you are just using local storage to cache data from a server it would also not be a big issue if the data is lost since it can just be fetched from the server again.

Local Storage is a simple key-value system, and can be accessed through the globally available localStorage object:

localStorage.setItem('someSetting', 'off');

**let** someSetting = localStorage.getItem('someSetting');

This is the native (as in native to web browsers, not iOS or Android native) way to set and retrieve local storage data.

**SQLite**

SQLite is basically an embedded SQL database that can run on a mobile device. Unlike a normal SQL database it does not need to run on a server, and does not require any configuration. SQLite can be utilized by both iOS and Android applications (as well as others), but the SQLite database can only be accessed natively, so it is not accessible by default to HTML5 mobile apps.

We can however use a Cordova to easily gain access to this functionality. Simply run the following command in your project to install it:

ionic plugin add https:*//github.com/litehelpers/Cordova-sqlite-storage*

The main benefits of using SQLite are that it:

* Provides persistent data storage
* There is no size limitation on how much can be stored
* It provides the SQL syntax, so it is a very powerful tool for managing data

Although there are some diﬀerences in supported commands between SQL and SQLite, it is almost exactly the same. Here’s an example of how you might execute some simple queries in SQLite:

**var** db = window.sqlitePlugin.openDatabase({name:"my.db"});

db.transaction(**function**(tx) {

tx.executeSql('DROP TABLE IF EXISTS test\_table');

tx.executeSql('CREATE TABLE IF NOT EXISTS test\_table (id integer primary key, data text, data\_num integer)');

tx.executeSql("INSERT INTO test\_table (data, data\_num) VALUES (?,?)", ["test", 100], **function**(tx, res) {

**console**.**log**("insertId: "+ res.insertId +" -- probably 1");

**console**.**log**("rowsAffected: "+ res.rowsAffected +" -- should be 1");

}, **function**(e) {

**console**.**log**("ERROR: "+ e.message);

});

});

The example above looks pretty freaky, but if you’re familiar with SQL then at least some of it should look familiar. This is the standard way to use SQLite with Cordova, but Ionic also provides its own service for using SQLite.

**Ionic Storage**

Fortunately, for most storage requirements we don’t really need to worry about the implementation de-tails. Ionic provides it’s own storage service that will automatically make use of the best available storage mechanism (whether that is plain old local storage, Web SQL, IndexedDB or SQLite). It provides us with a consistent API to use no matter which storage mechanism is being used underneath.

To use it, all you have to do it add it to your **app.module.ts** file by importing it:

**import** { Storage } from'@ionic/storage';

and adding it to the provider array:

providers: [Storage]

Then you can just inject it into whatever class you need. Take this provider as an example:

**import** { Storage } from'@ionic/storage'; **import** { Injectable } from'@angular/core';

**@Injectable**()

**export class** Data {

**constructor**(**public** storage: Storage){

}

getData(): any {

**return this**.storage.get('checklists');

}

save(data): **void** {

**let** newData = JSON.stringify(data);

**this**.storage.set('checklists', newData);

}

}

We simply call **this**.storage.set to store data on a particular key (in this case checklists) and then we can retrieve that same data later by accessing that key through **this**.storage.get.

If your data storage requirements are more complex this may not be the perfect option, but for many scenarios this provides a nice simple way to store the data we need.