Table of Contents

[? observable vs promise ? 1](#_Toc496003167)

[How to convert response type in observable / through map 2](#_Toc496003168)

[Step by step – how to create lazy loading of pages in ionic3? 2](#_Toc496003169)

[How to create ionic3 page manually/without CLI: 2](#_Toc496003170)

[Q. steps to install pouchdb: 4](#_Toc496003171)

[Q. Main steps to create synchronization in couchdb/ pouchdb: 4](#_Toc496003172)

[Q. How did you used replication in your project? 4](#_Toc496003173)

[Q. How to create page with CLI? 5](#_Toc496003174)

[CouchDB replication: 6](#_Toc496003175)

[Q How to use pouchdb in database? 7](#_Toc496003176)

[Q. One-way replication to pouchdb.. 7](#_Toc496003177)

[Q. Providers – how to steps 7](#_Toc496003178)

[CouchDB notes: 7](#_Toc496003179)

[Securing couchdb? 8](#_Toc496003180)

----------------------------------------------------------------------------------------------------

## ? observable vs promise ?

? where did you applied promise in angular4 project?

Ans. Get() and set() method of storge returns promise. While working with storage I used promise.. then() .

getUserName(): Promise<string> {

return this.storage.get(Preferences.APP\_USER).then(user => user.username);

}

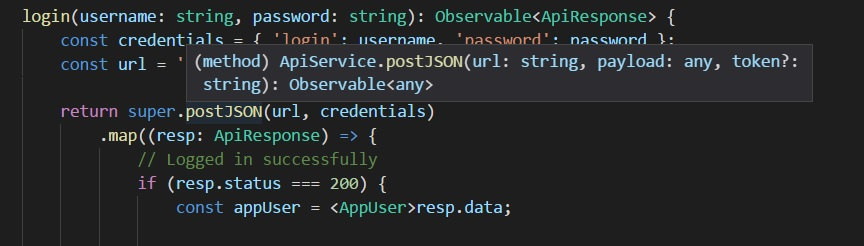
hasLoggedIn(): Promise<boolean> {

return this.storage.get('HAS\_LOGGED\_IN').then(value => value === true);

}

this.storage.set('HAS\_LOGGED\_OUT', false);

## How to convert response type in observable / through map



In this case observable is changed to ApiResponse…

## Step by step – how to create lazy loading of pages in ionic3?

If you put cli command ionic g page <PageName> then automatically it will take care of lazyloading of page.

After this you can directly use in app.component.ts rootPage: string = 'Page2Page'; you don’t have to include import <pagename> from \*\*\*\*

## How to create ionic3 page manually/without CLI:

Lets start by creating a page without using the CLI generator.

After generating your project, look for the pages folder then create a folder with the name of your page

Lets say I want to create Home page.

cd myProject/src/pages

mkdir home

Then create three files: home.ts, home.html and home.scss

home.ts :- will contain the TypeScript code of our page .

home.html :- will contain HTML code of our page view .

home.scss :- will contain Sass styles of our page view .

touch home.ts

touch home.html

touch home.scss

Open home.ts and add the following code to create a page

import { Component } from '@angular/core';

@Component({

selector: 'page-home',

templateUrl: 'home.html'

})

export class HomePage {

constructor() {

}

}

So as you can see a page in Ionic is just an Angular component which itself a TypeScript class decorated with *@Component* decorator imported from *@angular/core* .

The component/page gets the information about its view using the templateUrl which points to a path of an HTML page, in this case home.html which resides in the same folder .

You can instead provide an inline template using

template : '<p>Hello HomePage</p>'

The selector is the custom HTML tag which can be used to use the component inside another HTML view .

<page-home></page-home>

We can also use it to style the component, inside home.scss

page-home {

background : #000;

}

Next open home.html and add

<ion-header>

<ion-navbar>

<ion-title>

Ionic 3 Tutorial

</ion-title>

</ion-navbar>

</ion-header>

<ion-content padding>

<p>Hello HomePage</p>

</ion-content>

As you can see ,we use different Ionic specific custom tags or UI components to build the UI of our HomePage . The names of tags are self explanatory .

That is, you have created an Ionic page using Angular components and decorators .

## Q. steps to install pouchdb:

1. npm install pouchdb --save

2. npm install @types/pouchdb --save --save-exact

3. npm install -g add-cors-to-couchdb

4. add-cors-to-couchdb

## Q. Main steps to create synchronization in couchdb/ pouchdb:

Constructor in the service:-

import PouchDB from 'pouchdb';

data: any;

db: any;

remote: any;

constructor() {

this.db = new PouchDB('cloudo');

this.remote = 'http://localhost:5984/cloudo';

let options = {

live: true,

retry: true,

continuous: true

};

this.db.sync(this.remote, options);

}

## Q. How did you used replication in your project?

It was one-way replication mobile to server…. Database.replicate() is a promise function.

// Mobile -> server unidirectional replication

this.replication = this.database.replicate.to(this.remote, { live: true, retry: true });

this.replication

.on('complete', () => { console.log('Replication complete') })

.on('change', (change) => console.log('Replication change', change))

// .on('paused', (info) => console.log('Replication was paused'))

.on('error', (err) => console.error('Replication ERROR', err));

## Q. How to create page with CLI?

Use your terminal to type :

ionic g page HomePage

A new page will be created for you .

How to use pages in Ionic ?

You use pages to build apps ,an app is a collection of pages ,in two ways :

* By declaring and importing them when the app is starting .
* By lazy loading them .

So lets continue with our HomePage .To be able to use it ,you need to declare it in src/app/app.module.ts

* You start by importing it
* Then add it to declarations and entryComponents arrays of your app main NgModule

import { HomePage } from '../pages/home/home';

@NgModule({

declarations: [

MyApp,

HomePage

],

imports: [

BrowserModule,

IonicModule.forRoot(MyApp)

],

bootstrap: [IonicApp],

entryComponents: [

MyApp,

HomePage

],

providers: [

/\* .... \*/

]

})

export class AppModule {}

You can then make this page

* The root page of your app (from your main app component) .
* As part of a tab system .
* The view of a modal dialog .
* A normal page that you can navigate to it from other pages or menu .

You can also use a page by lazy loading it, which is good for performance, because the page is only loaded when it's requested by the app .

To lazy load HomePage, you need to do these steps :

* First create an NgModule for HomePage
* Decorate HomePage with IonicPage decorator
* Then reference HomePage by a string 'HomePage'

So go inside src/pages/home and create home.module.ts then add this code

import { NgModule } from '@angular/core';

import { IonicPageModule } from 'ionic-angular';

import { HomePage } from './home';

@NgModule({

declarations: [

HomePage,

],

imports: [

IonicPageModule.forChild(HomePage),

],

exports: [

HomePage

]

})

export class HomePageModule {}

next import IonicPage ionic-angular and decorate HomePage with it :

import { IonicPage } from 'ionic-angular';

@IonicPage()

@Component({

selector: 'page-home',

templateUrl: 'home.html'

})

export class HomePage {

/\* ... \*/

Now you can use the HomePage using the string 'HomePage' without the need to import it .

## CouchDB replication:

A CouchDB database can easily be replicated to another database (which we will be making use of in this tutorial), and this replication can be:

1. One way (PouchDB database is replicated to the CouchDB database)
2. Bi-directional (PouchDB database is replicate to the CouchDB database, and vice versa)
3. Ad hoc (replication is triggered manually)
4. Continuous (database is continually replicated as necessary, changes are replicated instantly)

Pouchdb is used to get awesome offline sync functionality. We have used bi-directional and continuous replication.

## Q How to use pouchdb in database?

import PouchDB from 'pouchdb';

@Injectable()

export class Todos {

  data: any;

  db: any;

  remote: any;

  constructor() {

    this.db = new PouchDB('cloudo');

    this.remote = '<http://localhost:5984/cloudo>';

    let options = {

      live: true,

      retry: true,

      continuous: true

    };

    this.db.sync(this.remote, options);

  }

## Q. One-way replication to pouchdb..

one way replication you could instead use this.db.replicate.to('http://localhost:5984/cloudo').

## Q. Providers – how to steps

* You can create a new provider using the command: ionic g provider MyProvider
* You must **import** the provider into **app.module.ts** and **add it** to the providers array
* You must **import** the provider into any component that uses it
* You must **inject** the provider in the constructor of any component that is using it

## CouchDB notes:

\_rev field is used to manage concurrency problem.

While defining attributes we don’t need to mention any datatype for the attribute which is not like rdbms.

## Securing couchdb?

By default couchdb is not secure, any one can access.

Steps to secure:

* Create admin user with username/password
* In configuration section put require valid user true (by default it will be false… change to true)

## Qn. Routing / Navigation in Ionic3?

Ans. I have used push, pop, setroot to use navigation. Used only component navigation stack did not use Deeplink.

Basically there are two ways of navigation we can make use in ionic

1) [DeepLink](http://blog.ionic.io/deeplinking-in-ionic-apps/)

2) component navigation stak

**DeepLink**

deeplink navigation acts like webpage navigation like below exaple you have to use @ionicpage anotation inorder to use deeplink navigation

<https://locallhost:8000/#/HomePage/SecondPage>

**Component Navigation**

You have to import your component in the respected ts file inorder to navigate

Totaly there are three key words push, pop, setRoot.

1. **setRoot**

**Example**:

this.navCtrl.setRoot(HomePage);

(or)

this.navCtrl.setRoot("HomePage"); //DeepLink navigation

Used to make the component as Root page in other words it creates an empty navigation stack where homepage is the root.

1. **push**

**Example:**

this.navCtrl.push(SecondPage);

(or)

this.navCtrl.push("SecondPage"); //DeepLink navigation

the above example has push keyword where the navigation stack has one component inside its stack followed by Homepage.i mean after homepage component you will be having secondpage component in the navigation stack.

3.**pop**

**Example:**

this.navCtrl.pop();

(or)

this.navCtrl.pop(); //DeepLink navigation

Consider you are in secondpage now and wanted to go back to the previous page which is home page then just use the above example it will pop one component from the navigation stack and gives you only the homepage component in the navigation stack

## Q. Less important: Difference between ionic page vs component

**Based on the conversation from the comments:**

It may be the same from the Angular point of view, but Pages and Components have a different meaning in Ionic. **In terms of Angular, both are just components**, but **in the context of Ionic, a Page is a component that will act as *an entire view*** (it may have nested components); we see Ionic pages as a **standalone concept**. A component will be just part of a bigger component most of the time in Angular apps, so I guess that's the biggest difference with Pages.

About when using Angular's lifecycle hooks, I like to use them when working in nested components, but I prefer Ionic lifecycle hooks when working on pages. Mostly because things like ionViewWillEnter doesn't make too much sense in the context of a simple component, where ngOnInit does. That being said, I also used some Angular lifecycle hooks on Pages, like the ngOnDestroy (I used it to remove all the subscriptions from a page when that page is going to be destroyed), but just like you said, ionViewWillUnload seems to be the right way to do it if we want to use Ionic's lifecycle hooks.

**I guess that *most* of the Ionic lifecycle hooks are more related to the way the user interacts with the page as a whole** (will enter to a page, will leave from a page, can enter to a page, can leave from a page...) **and Angular lifecycle hooks are more related to the different stages of the life of a single component** (the inputs has been initialized, the change detector has checked if there where changes in this component, ...), which as you can see, may not be directly related to the user interaction at all, and usually are things that the user is not aware of.

## Rxjs observable

Call returns an RxJS Observable response, which contains data about the response headers, the payload, status codes and the like. RxJS is a functional programming API, which means we can chain methods onto the observed payload and pull out only what we want to return.