Chapter 14: Caching

Section 14.1: Caching using Akavache

About Akavache

Akavache is an incredibly useful library providing reach functionality of caching your data. Akavache provides a key-value storage interface and works on the top of SQLite3. You do not need to keep your schema synced as it's actually No-SQL solution which makes it perfect for most of the mobile applications especially if you need your app to be updated often without data loss.

Recommendations for Xamarin

Akavache is definetly the best caching library for Xamarin application if only you do not need to operate with strongly relative data, binary or really big amounts of data. Use Akavache in the following cases:

- You need your app to cache the data for a given period of time (you can configure expiration timeout for each entity being saved;
- You want your app to work offline;
- It's hard to determine and freeze the schema of your data. For example, you have lists containing different typed objects;
- It's enough for you to have simple key-value access to the data and you do not need to make complex queries.

Akavache is not a "silver bullet" for data storage so think twice about using it in the following cases:

- Your data entities have many relations between each other;
- You don't really need your app to work offline;
- You have huge amount of data to be saved locally;
- You need to migrate your data from version to version;
- You need to perform complex queries typical for SQL like grouping, projections etc.

> Actually you can manually migrate your data just by reading and writing it back with updated fields.

Simple example

Interacting with Akavache is primarily done through an object called `BlobCache`.

Most of the Akavache's methods returns reactive observables, but you also can just await them thanks to extension methods.

```
using System.Reactive.Linq;   // IMPORTANT - this makes await work!

// Make sure you set the application name before doing any inserts or gets
BlobCache.ApplicationName = "AkavacheExperiment";

var myToaster = new Toaster();
await BlobCache.UserAccount.InsertObject("toaster", myToaster);

//
// ...later, in another part of town...
//
```

```
// Using async/await
var toaster = await BlobCache.UserAccount.GetObject<Toaster>("toaster");

// or without async/await
Toaster toaster;

BlobCache.UserAccount.GetObject<Toaster>("toaster")
    .Subscribe(x => toaster = x, ex => Console.WriteLine("No Key!"));
```

Error handling

```
Toaster toaster;

try {
    toaster = await BlobCache.UserAccount.GetObjectAsync("toaster");
} catch (KeyNotFoundException ex) {
    toaster = new Toaster();
}

// Or without async/await:
toaster = await BlobCache.UserAccount.GetObjectAsync<Toaster>("toaster")
    .Catch(Observable.Return(new Toaster()));
```