Асинхронное подключение к SQLite

Последнее обновление: 17.09.2016

В прошлых темах для подключения к базе данных SQLite мы использовали синхронные методы, которые на время вызова и выполнения операций с данными блокировали основной поток графического приложения. Однако библиотека SQLite.NET предоставляет также асинхронный API. Он предоставляет методы, которые во многом похожи на синхронные.

Асинхронное подключение в SQLite.NET представляет объект класса **SQLiteAsyncConnection**. А весь асинхронный API раскрывается через методы этого класса:

- **CreateTableAsync()**: создание таблицы
- **DropTableAsync()**: удаление таблицы
- **Table<T>()**: получение данных из таблицы
- **GetAsync<T>()**: получение объекта
- **DeleteAsync()**: удаление объекта
- **UpdateAsync()**: обновление существующего объекта
- **InsertAsync()**: добавление нового элемента
- **ExecuteAsync()**: выполнение sql-выражения
- **ExecuteScalarAsync()**: выполнение sql-выражения и получение результата
- **FindAsync()**: поиск объекта в БД
- **QueryAsync()**: выполнение запроса SQL
- **InsertAllAsync()**: добавление нескольких объектов
- **UpdateAllAsync()**: обновление нескольких объектов

Для применения асинхронного API добавим в проект новый класс FriendsAsyncRepository:

```
using System.Collections.Generic;
using System.Threading.Tasks;
using SQLite;
```

```
using Xamarin.Forms;

namespace SQLiteApp
{
    public class FriendAsyncRepository
    {
        SQLiteAsyncConnection database;

        public FriendAsyncRepository(string filename)
        {
            string databasePath = DependencyService.Get<ISQLite>().GetDatabasePath(filename);
            database = new SQLiteAsyncConnection(databasePath);
        }

        public async Task CreateTable()
        {
            await database.CreateTableAsync<Friend>();
        }
        public async Task<List<Friend>> GetItemsAsync()
        {
            return await database.Table<Friend>().ToListAsync();

        }
        public async Task<Friend> GetItemAsync(int id)
        {
            return await database.GetAsync<Friend>(id);
        }
        public async Task<int> DeleteItemAsync(Friend item)
        {
            return await database.DeleteAsync(item);
        }
        public async Task<int> SaveItemAsync(Friend item)
        {
            if (item.Id != 0)
            {
                await database.UpdateAsync(item);
                return item.Id;
            }
            else
            {
                return await database.InsertAsync(item);
            }
        }
    }
}
```

В отличие от синхронной версии здесь создание таблицы вынесено в отдельный метод.

И далее похожим образом мы сможем использовать этот репозиторий. Установим его в классе App:

```
using Xamarin.Forms;

namespace SQLiteApp
{
    public partial class App : Application
    {
        public const string DATABASE_NAME= "friends2.db";
        public static FriendAsyncRepository database;
        public static FriendAsyncRepository Database
        {
            get
            {
                if (database == null)
                {
                    database = new FriendAsyncRepository(DATABASE_NAME);
```

```
                }
                return database;
            }
        }
        public App()
        {
            MainPage = new NavigationPage(new MainPage());
        }

        protected override void OnStart()
        { }

        protected override void OnSleep()
        { }

        protected override void OnResume()
        { }
    }
}
```

И затем используем на главной странице приложения:

```
public partial class MainPage : ContentPage
{
    public MainPage()
    {
        InitializeComponent();
    }

    protected override async void OnAppearing()
    {
        // создание таблицы, если ее нет
        await App.Database.CreateTable();
        // привязка данных
        friendsList.ItemsSource = await App.Database.GetItemsAsync();

        base.OnAppearing();
    }
}
```


Назад **Содержание** **Вперед**

Ярослав Орлов • год назад

Доброй ночи, скажите, а как сделать потом этот объект Task в свою viewmodel при помощи select, или как потом реализовать поиск и фильтрацию если операции Linq не применяются к Task.

^ v • Ответить • Поделиться ›

Metanit Модератор ➚ Ярослав Орлов • год назад

для подобных задач наверное лучше выделить в репозитории отдельный метод типа

```
public Task<List<Friend>> FilterByName(string name)
{
return database.Table<Friend>().Where(x => x.Name==name).ToListAsync();

}
```

^ v • Ответить • Поделиться ›

Ярослав Орлов ➚ Metanit • год назад

Да это будет отличный вариант, но назрело еще два вопроса, если не составит труда помогите с ними:

1) в чем тогда отличие асинхронного обращения к БД и соответственно метода ToListAsync()? Как я вижу в вашем примере уже нет async await.

2) И у меня стала проблема, работая до этого не асинхронно я выводил свою NoteViewModel и NotesListViewModel на страницы и к ним биндился соответственно, при это использовал обычный Select для каста. Но как у вас показано, то сама модель сразу стает источником для ListView и я не нашел как я ее кастануть, помогите.

^ v • Ответить • Поделиться ›

Metanit Модератор ➚ Ярослав Орлов • год назад

1) да там async/await. Я просто забыл прописать

2) не понятно что во что преобразовывать?

^ v • Ответить • Поделиться ›

Ярослав Орлов ➚ Metanit • год назад

За счет того что база данных создается асинхронно, а именно SQLiteAsyncConnection вместо SQLiteConnection, то Linq операции типа Select нельзя применить, и приходиться выводить в ListView мою базовую модель Note, вместо того как я делал до этого NoteViewModel (..Select(x => new NoteViewModel(){....})). Пока не понимаю как это

обойти или по правильному сделать.

• Ответить • Поделиться ›

Metanit Модератор ➚ Ярослав Орлов • год назад

сначала загружать в память, потом выполнять преобразование

```
public async Task<list<string>> FilterByName(string name)
{
var friends= await database.Table<friend>().Where(x =>
x.Name==name).ToListAsync();

return friends.Select(X=>X.Email).ToList();
}
```

• Ответить • Поделиться ›

Ярослав Орлов ➚ Metanit • год назад

Спасибо огромное все очень круто и замечательно работает. Разобрался что к чему. Спасибо за все ваши ответы и пояснения!

• Ответить • Поделиться ›