

Silverlight(WP7)で ローカルDB使ってみよう

Sterling OODB の紹介



Silverlight/Phone Quest I
2011/08/27 おだ

自己紹介

Silverlight
Square

.com

- 織田 信亮(おだ しんすけ)
- 大阪で開発者しています
- MS 系の情報を追っかけています。
- SQL World 代表
- JGGUG でも活動しています。



JGGUG
JAPAN GRAILS/GROOVY USER GROUP

- <http://d.hatena.ne.jp/odashinsuke>
- Twitter: @shinsukeoda

お話しすること

Silverlight
Square

.com

- 色んなローカルDB の紹介
- Sterling OODB の紹介
- Sterling OODB を使ってみる
- まとめ

お話しすること

- 色んなローカルDB の紹介
- Sterling OODB の紹介
- Sterling OODB を使ってみる
- まとめ

➤ WP7

- Mango から使えるようになる
- WP7.0 ではダメ！

➤ Silverlight

- 通常は使えない
- OOB + COM で使える
 - ✓ Silverlight COM Toolkit を使った例

<http://erikej.blogspot.com/2010/02/access-local-sql-compact-database-from.html>

C#SQLite

Silverlight
Square

.com

➤ Silverlight/WP7 ともに対応している

<http://code.google.com/p/csharp-sqlite/>

<http://csharpsqlite.uservoice.com/forums/41270-general/suggestions/996353-support-silverlight-windows-phone-7?ref=title>

C#
SQLite

General Forum

[← return to General Forum](#)

17
votes


Support Silverlight & Windows Phone 7

Hi, I'd love to have C#-SQLite available for WP7.

by [Craig D](#) | [0 comments](#)

Status: **completed**

Silverlight is now fully supported; Thank you Alex West!

 [noah_hart](#)
admin

- .NET, Mono, Silverlight, MonoTouch, MonoAndroid, CompactFramework, WP7 で動く OODB

<http://siaqodb.com/>

- Sync Framework の Provider が用意されていて、SQL Server と同期可能！

但し、有償です！

(30日トライアル期間有り)

お話しすること

- 色んなローカルDB の紹介
- Sterling OODB の紹介
- Sterling OODB を使ってみる
- まとめ

Sterling OODB

Silverlight
Square

.com

- .NET 4, Silverlight 4/5, WP7 で動く
OODB

<http://sterling.codeplex.com/>

<http://www.sterlingdatabase.com/>

無償です！ (Ms-PL 緩いライセンス)

<http://www.microsoft.com/japan/resources/sharedsource/licensingbasics/sharedsourcelicenses.aspx#E4B>

<http://www.microsoft.com/japan/resources/sharedsource/licensingbasics/permisivelicense.aspx>

➤ 負担を掛けない

永続化するクラスを変更しなくてOK

➤ 軽量

アセンブリは 100KB未満。永続化したいだけなのに、プロジェクトが肥大化するなんてありえない！

➤ 柔軟

アセンブリは軽いけど、シリアライズして、LINQ to Object 経由で超簡単にアクセス出来る！

➤ 移植

.NET, Silverlight, Windows Phone で動く

主な特徴 (1)

- 高速で軽量なバイナリシリアル化
- ダーティサポート(ダーティリード?)
- 複数のドライバをサポート
 - メモリ
 - ローカルファイルシステム
 - 分離ストレージ
- 暗号化/複合のサポート
- バックアップ/リストア

主な特徴 (2)

- 循環参照の型も大丈夫
- IList, IDictionary, Arrays のサポート
- WriteableBitmaps のサポート
- Interface, abstract properties もシリアライズ対象
- KEY は色々な型をサポート (複合キーも)
- インデックスをサポート(カバリングインデックス含む)

主な特徴 (3)

Silverlight
Square

.com

- Lazy<T> と Tuple<T> を WP7 にも実装した
- Nullable<T> のサポート
- Enum の自動変換サポート
- 基本的な型である decimal, Uri, Guid, DateTime, 等をサポート(SL の BinaryWriter では未サポート)

主な特徴 (4)

- 全ての型にデフォルトコンストラクタがあれば、ネストした型も処理出来る
- トリガーのサポート(before save, after save, before delete)

お話しすること

- 色々なローカルDB の紹介
- Sterling OODB の紹介
- Sterling OODB を使ってみる
- まとめ

セットアップ

Silverlight
Square

.com

- Visual Studio or Web Developer Express の方は、NuGet で
- Express for Windows Phone の方は、ダウンロード後に、参照の追加から

<http://nuget.org/List/Packages/Sterling>

```
PM> Install-Package Sterling
```

<http://nuget.org/List/Packages/SterlingPhone>

```
PM> Install-Package SterlingPhone
```

永続化するクラスを定義 (1)



➤ POCO(plain old CLR object)

➤ 使える型は？

BinaryWriter で使える型			追加で拡張した型	
bool	double	short(Int16)	DateTime	Uri
byte	float(single)	string	DateTimeOffset	WriteableBitmap
byte[]	int(Int32)	uint	decimal	
char	long(Int64)	ulong	Guid	
char[]	sbyte	ushort	TimeSpan	

➤ 他の型を使いたい場合は？

- 自前でカスタムシリアライザーを実装してね

永続化するクラスを定義 (2)

- public なフィールド、プロパティが対象
 - readonly(setter が private)、static、SterlingIgnore Attribute が付与された物は除く
 - readonly フィールドは、読み込む時に例外が出るので、SterlingIgnore 属性を付与すること。
- ネストしたクラスもOK

```
public class Foo
{
    public int Id { get; set; }
    public Bar Instance { get; set; }
}
```

永続化するクラスを定義 (3)

➤ Array や List も OK

- 保存したいクラス内に Array や List のフィールド/プロパティ が居ても OK
- 自前で実装したカスタム List でも OK
データベースに定義するクラスとして、List を利用

➤ Dictionary も OK

```
public class Foo
{
    public int Id { get; set; }
    public List<Bar> Items { get; set; }
}
```

データベースクラスを定義 (1)

- BaseDatabaseInstance を継承する
- RegisterTables メソッドを override
 - 保存したいクラスを定義する
(複数のクラスを定義可能)
 - キーとインデックスを決める
 - ✓キーは必須(一意になる必要がある)
 - ✓インデックスは任意(複数設定可能)

```
using System;
```

```
// 永続化したいクラス
```

```
public class ハード
```

```
{
```

```
    public string 名前 { get; set; }
```

```
    public int 価格 { get; set; }
```

```
}
```

```
public class ゲームソフト
```

```
{
```

```
    public int Id { get; set; }
```

```
    public string ハード { get; set; }
```

```
    public string タイトル { get; set; }
```

```
    public DateTime 発売日 { get; set; }
```

```
}
```

```

using System;
using Wintellect.Sterling.Database;
using System.Collections.Generic;

public class Database : BaseDatabaseInstance {
    public const string タイトルIndex = "IX_タイトル";
    public const string ハード発売日Index = "IX_ハード発売日";
    protected override List<ITableDefinition> RegisterTables() {
        return new List<ITableDefinition>() {
            CreateTableDefinition<ハード, string>(e => e.名前),
            CreateTableDefinition<ゲームソフト, int>(e => e.Id)
                .WithIndex<ゲームソフト, string, int>(
                    タイトルIndex, e => e.タイトル)
                .WithIndex<ゲームソフト, string, DateTime, int>(
                    ハード発売日Index, e =>
                        new Tuple<string, DateTime>(e.ハード, e.発売日)
                ),
        };
    }
}

```

データベースクラスを定義 (2)

- 基底のクラスで定義し、派生クラスを保存する事も出来るがお勧めしない。

```
public class Base { }  
public class Foo : Base { }  
public class Bar : Base { }
```

Base で登録すると、Foo、Bar のインスタンスを保存出来るが、Base で定義された項目だけ保存する。

これを回避しようとする、保存する際にややこしい overload メソッドを呼ぶ必要がある。

また、取り出す時も Base 型として取れるので、キャストが必要。

- テーブルには、キーが必要
- 永続化するクラスを一意に識別する
- キーはメモリ上に持つ
- 使える型が決まっている

プリミティブな型(string, int 等), GUID

他の型を使いたい場合は、カスタムシリアライザーを実装する

キーの型は、Equalable/Comparable であり、HashCode を適切に実装している必要がある

インデックス

- インデックスもメモリ上に持つ
 - LINQ で高速に取得/フィルタリング出来る
- 2つまでの値をサポート
- キーと同じ形でシリアライズされる
 - キーと同じ制約がある(はず)
- インデックスは名前付けだよ
 - よく使う項目に名前を付けて、高速アクセス可能にしている

データベースを生成する (1)

- SterlingEngine にデータベースを登録することで、データベースを生成する

```
using (var engine = new SterlingEngine())
{
    engine.Activate();
    var database = engine.SterlingDatabase
        .RegisterDatabase<Database>(new IsolatedStorageDriver());
    // 以降 database インスタンス を使って、CRUD 可能
}
```

- IsolatedStorageDriver を渡すこと！
 - 渡さない場合、永続化先が メモリ になる。

データベースを生成する (2)

➤ SterlingEngine で出来る事

- データベースの登録/インスタンスの管理
- カスタムシリアライザーの登録
- ロガーの登録

```
engine.SterlingDatabase.RegisterLogger((lv, msg, ex) => {  
    Debug.WriteLine(lv + msg);  
});
```

- バックアップ/リストア

データベースを生成する (おまけ)

- SterlingEngine の生存期間がアプリケーションの開始 - 終了 の場合、Application. ApplicationLifetimeObjects で管理するのが一般的？
 - IApplicationService
 - IApplicationLifetimeAwareを実装したクラスで、SterlingEngine を管理し、SL/WP7 の Application クラスの ApplicationLifetimeObjects に設定する

データベースを生成する (おまけ)

Silverlight
Square

.com

```
<Application
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SilverlightApplication3.App"
  xmlns:database="clr-namespace:SilverlightApplication3.Database">
  <Application.ApplicationLifetimeObjects>
    <database:SterlingService />
  </Application.ApplicationLifetimeObjects>
  <Application.Resources>

  </Application.Resources>
</Application>
```

アプリケーション
の起動 - 終了で
インスタンスを
管理できる

保存 (Save)

➤ Save メソッドで保存

```
database.Save<ハード>(new ハード() {  
    名前 = "DS", 価格 = 18000  
});
```

➤ Save 後は Flush 推奨

- Flush を呼ばなくても、シャットダウン時に書き込まれるがアプリケーションが突然停止した場合等は、書き込まれないので適宜 Flush を呼ぼう！

➤ 件数が多い時用に、非同期もサポート

- BackgroundWorker SaveAsync(IList<Model>)

取得 (Load/Query)

- Load は、キー指定で 1件取得
 - 読み込みをキャッシュしない
- Query は、キーやインデックスでフィルタリングが出来る
 - 実際の値は、遅延ロード(Lazy<T>)される
 - キーやインデックスは Lazy ではない！
 - ✓ カバリングインデックス が効く
 - キャッシュします
 - ✓ **同一参照**なので注意が必要！
 - ✓ Save や Delete が呼ばれると再取得しに行く

```
var o = database.Load<FooModel>(1);
var n = database.Query<FooModel, int>()
    .Where(e => e.Key == 1)
    .First().LazyValue.Value; // Lazy から実際の値を取得
var o2 = database.Load<FooModel>(1);
var n2 = database.Query<FooModel, int>()
    .Where(e => e.Key == 1)
    .First().LazyValue.Value;
Debug.Assert(o != o2);
Debug.Assert(n == n2); // 同一参照！
database.Save(new FooModel () { Id = 1, Name = "変更し
た" });
o2 = database.Load<FooModel>(1);
n2 = database.Query<FooModel, int>()
    .Where(e => e.Key == 1)
    .First().LazyValue.Value;
Debug.Assert(o != o2);
Debug.Assert(n != n2);
```

削除 (Delete/Truncate/Purge)

- Delete は、インスタンス指定/キー指定で1件削除
- Truncate は、指定したクラスのデータを全部削除
- Purge は、データベースに定義したクラスのデータを全部削除

```
database.Delete<ハード>(instance);  
database.Delete (typeof(ハード), key);  
database.Truncate(typeof(ハード));  
database.Purge();
```

➤ Trigger

検証や自動採番、登録済フラグなんかの更新に
BeforeSave/AfterSave/BeforeDelete

- 対象のインスタンスが渡ってくる。
(値の書換え等が可能)
- Before~ の戻り値は bool。false を返せば
後の処理が行われない。

false を返すと Save/Delete の呼出元で、
例外が発生 (SterlingTriggerException)する。

```
public class ゲームソフト {
    public int Id { get; set; }
    ...
    [SterlingIgnore]
    public bool Changes { get; set; }
}

public class Trigger : BaseSterlingTrigger<ゲームソフト, int> {
    private int _nextId;
    public Trigger(int maxId) { _nextId = maxId; }
    public override bool BeforeSave(ゲームソフト instance) {
        if (instance.Id == 0) instance.Id = ++_nextId;
        return true;
    }
    public override void AfterSave(ゲームソフト instance) {
        instance.Changes = false;
    }
    public override bool BeforeDelete(int key) {
        return false; // ゲームソフトは削除出来ない
    }
}
```

```
using (var engine = new SterlingEngine())
{
    engine.Activate();
    var database = engine.SterlingDatabase
        .RegisterDatabase<Database>(new IsolatedStorageDriver());

    var maxId = database.Query<ゲームソフト, int>().Any() ?
        database.Query<ゲームソフト, int>().Max(e => e.Key) : 0;
    database.RegisterTrigger(new Trigger(maxId));

    // 以降 database インスタンス を使って、CRUD 可能
}
```

データベース登録後に、
トリガーも登録する

おまけ (Interceptor)

➤ Interceptor

暗号化/複合、圧縮 等に使おう

Save/Load

- byte[] を受け取り、byte[] を返す
- Save と Load は対称的に実装しよう

他にも…

- バックアップ/リストア
- シリアライズ
- ロギング
- テクニク
 - ネストしたクラスの解析
 - シングルトンインスタンスの管理
 - WP7 の Tombstone 対策
 - 基本型のテーブルに派生型を渡す
 - ロールバック
 - Nullable な キー/インデックス
 - etc…

お話しすること

- 色々なローカルDB の紹介
- Sterling OODB の紹介
- Sterling OODB を使ってみる
- まとめ

まとめ

- 無償で使える
- SL/WP7 で同じコードが使えるそう
- 基本的な機能は揃っている

トランザクション？ SL/WP7 のローカルDBで
同時実行とか考慮する？

日本語の情報が少ないので、
発信しよう！

CodePlex

<http://sterling.codeplex.com/>

Sterling User's Guide

<http://www.sterlingdatabase.com/home>

MSDN Magazine 2011 June

Windows Phone 7 の分離ストレージ用の
Sterling

<http://msdn.microsoft.com/ja-jp/magazine/hh205658.aspx>