LINQ

- „LINQ extends the language by the addition of query expressions, which are akin to SQL statements, and can be used to conveniently extract and process data from arrays, enumerable classes, XML documents, relational databases, and third-party data sources.”

LINQ

- Language Integrated Query,
- A set of common interfaces to operate on data,
- Built-in support for objects, SQL databases, and XML,
- Works on IEnumerable and IQueryable,
- Suported in .NET since 3.5,
- Partially integrated into C# itself.

Standard query operators

- Select – performs a projection on the collection to select interesting aspects of the elements,
- Where – defines a set of rules checked for every object in a collection, and only objects matching are returned,
- Sum/Min/Max/Average – take a function that retrieves a certain numeric value, and find Sum, Min…
- Take and Skip – take takes first N objects, while Skip skips first N objects,
- OrderBy

Standard query operators

- Reverse – reverses a collection,
- Single and SingleOrDefault – returns first element matching function. If none or more elements matching exception (or null in case of Default) is thrown,
- First and FirstOrDefault – same, but without exception if more elements are found,
- Contains – checks if collection contains an element,
- Count – counts the number of elements in collection (all or just matching function).

Conversions

- ToArray,
- ToList,
- AsEnumerable,
- AsQueryable.

Match functions, but how?

- Some operators take function as a parameter to match elements to,
- Using delegates,
- But you should use lambda expressions to make thinks easier,
- Functions for matching LINQ operators are called predicates,
- Var construct is useful.

LINQ example

```
List<MyColor> colors = new List<MyColor>();
colors.Add(new MyColor() { Color = Colors.Azure, Name = "Azure" });
colors.Add(new MyColor() { Color = Colors.Beige, Name = "Beige" });
colors.Add(new MyColor() { Color = Colors.Chocolate, Name = "Chocolate" });
colors.Add(new MyColor() { Color = Colors.DarkRed, Name = "Dark Red" });

var result = colors.Where(x => x.Name == "Azure").ToArray();
```

LINQ example (2)

- var r
- For e ed, with ist,

```
public MainPage()
{
    this.InitializeComponent();

    List<MyColor> colors = new List<MyColor>();
    colors.Add(new MyColor() { Color = Colors.Azure, Name = "Azure" });
    colors.Add(new MyColor() { Color = Colors.Beige, Name = "Beige" });
    colors.Add(new MyColor() { Color = Colors.Chocolate, Name = "Chocolate" });
    colors.Add(new MyColor() { Color = Colors.DarkRed, Name = "Dark Red" });

    var result = colors.Select(x => new { x.Name });

    list.ItemsS
}

}

public class MyColo
{
    public string N
    public Color Co
}

public class MyColorCo
{
    public object Conv                                    getType, object parameter, string language)
    {
```

result {System.Linq.Enumerable.WhereSelectListIterator<App1.MyColor, <>f__AnonymousType0<string>>}

Current	null
System.Collections.IEnumerator.Current	null
_enumerator	{System.Collections.Generic.List<App1.MyColor>.Enumerator}
_predicate	null
_selector	{Method = {<>f__AnonymousType0`1[System.String] <.ctor>b__0_0(App1.MyColor)}}
_source	Count = 4
_threadId	3
current	null
state	-1
Results View	Expanding the Results View will enumerate the IEnumerable

[0] { Name = "Azure" }
[1] { Name = "Beige" }
[2] { Name = "Chocolate" }
[3] { Name = "Dark Red" }

LINQ example (3)

- var r = colors.Count(x => x.Name.StartsWith("A") || x.Name.StartsWith("B"));

- Counts only that X elements from a collection which names start with A or B.

LINQ example (4)

- var result = colors.Sum(x => x.Name.Length);

- Sums only results of function taking each element of a collections and returning its name’s length.

Sqlite

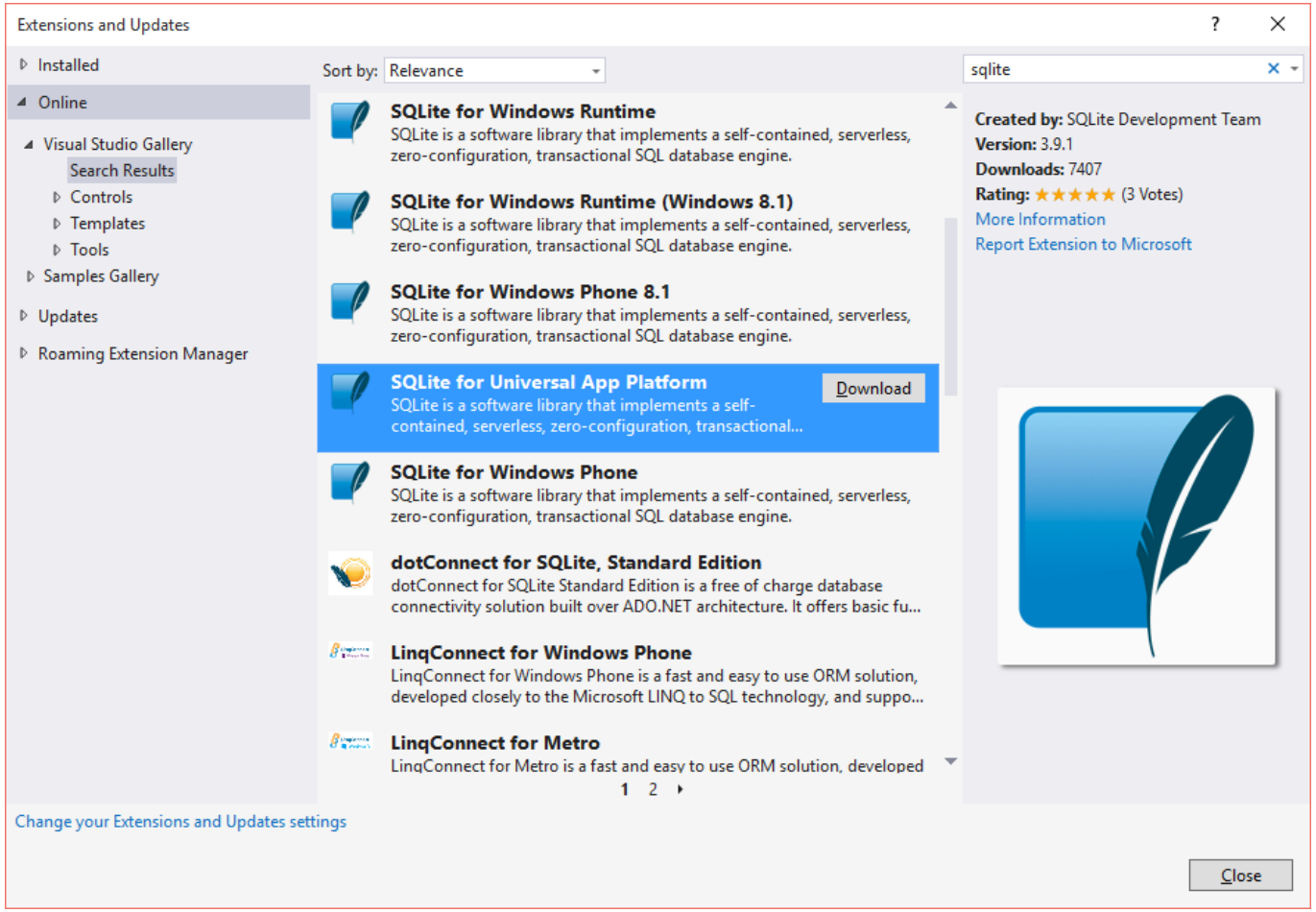
SQL CE?

- SQL CE was avilable in Windows Phone 7.5
- But was marked as obsolete in WinRT
- Open-source SQLite is the successor
- LINQ to SQL is not available out of the box

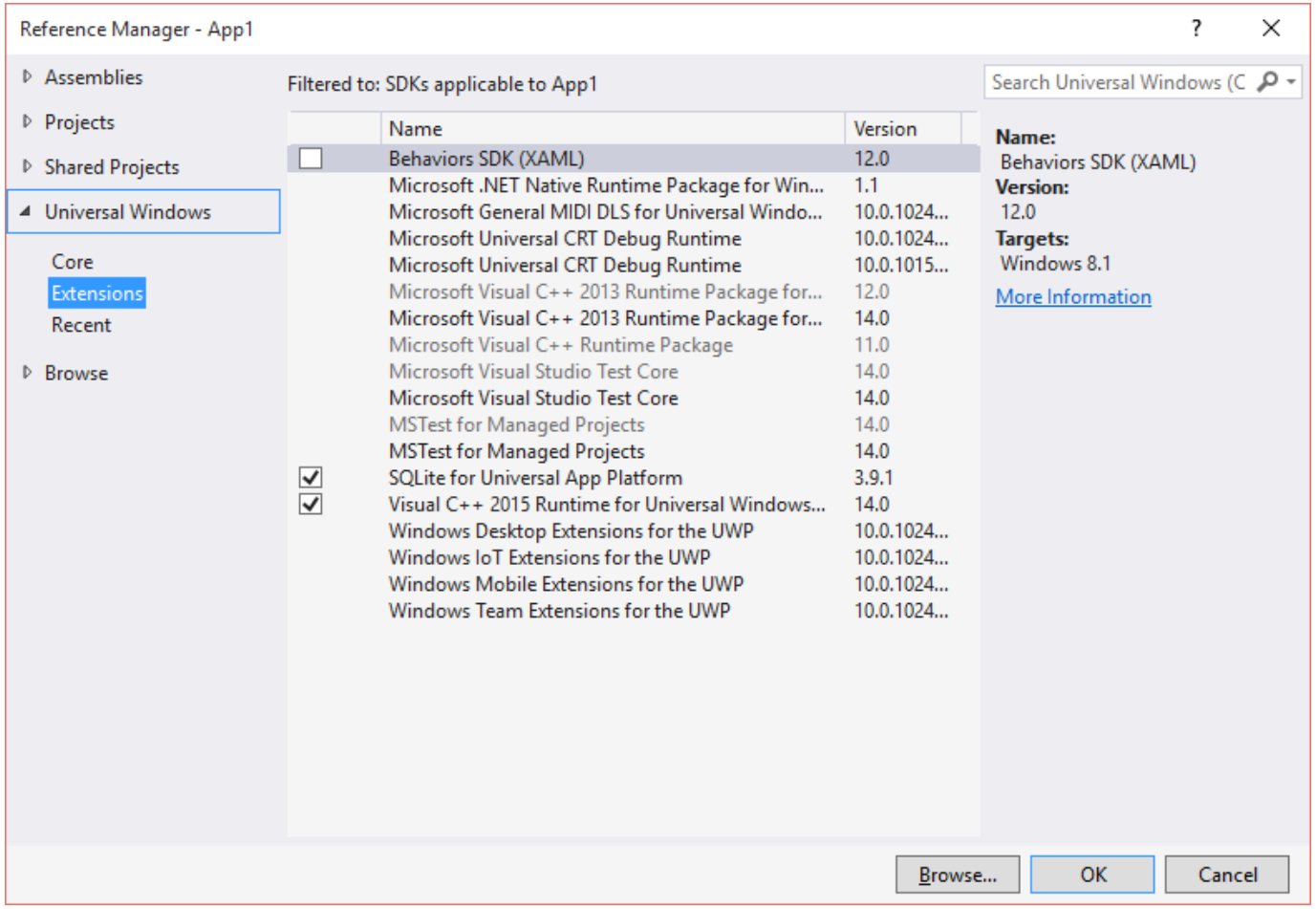
SQLite

- Embedded database system
- Data in one file
- Available from sqlite.org for many platforms
- Official support in Visual Studio for UWP apps
- But also for older platforms and desktop

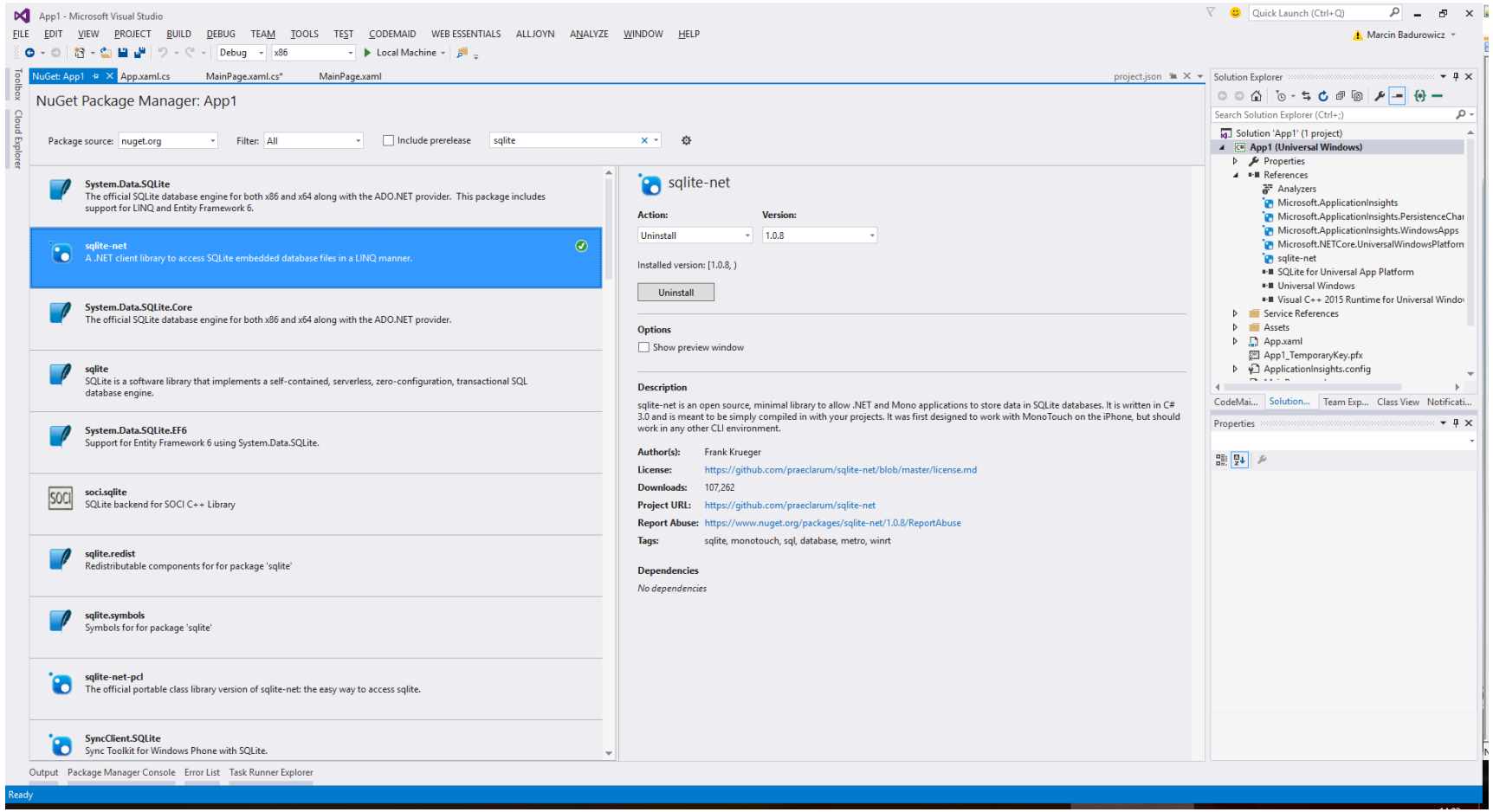
Installing SQLite extension for VS



Adding references



.NET wrapper installation



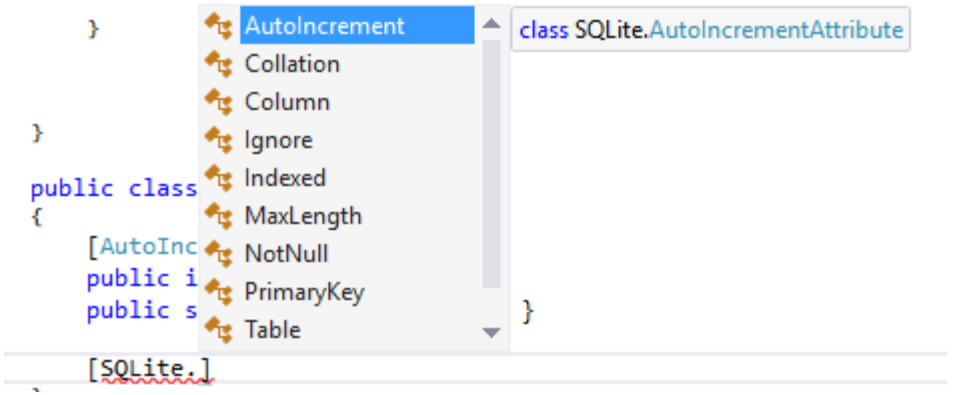
Working with models

```
public class Fruit
{
      [AutoIncrement, PrimaryKey]
      public int Id { get; set; }
      public string Name { get; set; }
}

var path = Path.Combine(ApplicationData.Current.LocalFolder.Path,
 "database.db");
using (var dbconn = new SQLiteConnection(path))
{
      dbconn.CreateTable<Fruit>();
}
```

Working with models

```
dbconn.Insert(new Fruit() { Name = "Apple" });

var apple = dbconn.Find<Fruit>(x => x.Name == "Apple");

var apple = dbconn.Find<Fruit>(x => x.Name == "Apple");
apple.Name = "Orange";
dbconn.Update(apple);


var allFruits = dbconn.Table<Fruit>();


var f = dbconn.Find<Fruit>(x => x.Name == "Pear");
dbconn.Delete<Fruit>(f);
```

Model attributes



Working with pure SQL

```
dbconn.Execute("DELETE FROM Fruit WHERE Id = ?", 1);

var result = dbconn.Query<Fruit>("SELECT * FROM Fruit");
```