一、Expression2Sql 介绍

Expression2Sql 是一个可以将 Expression 表达式树解析成 Transact-SQL 的开源项目。简单易用，几分钟即可上手使用，因为博主在设计 Expression2Sql 的时候就尽可能的按照 Transact-SQL 的语法语义风格来设计，只要调用者熟悉基本的 Transact-SQL 语法即可迅速开码，大大降低了学习 Expression2Sql 的成本，甚至零成本。对象化操作，链式编程，任意组装 sql，自动生成表别名，参数化赋值，防止 sql 注入，支持多数据库，生成极度美观的 sql 字符串(格式化)，优点 A，优点 B，优点 C，优点...还是等你来发现吧！O(∩_∩)O~

由于 insert 操作基本上是纯反射，很难和表达式树挂上钩，所以就不提供 insert 操作的方法了。Expression2Sql 目前推出的首个版本是 1.0，所以功能完善程度不高，只能做一些简单的表达式树解析成 sql 的操作。后期博主会持续更新维护，陆陆续续的增加智能缓存、日志埋点、sql 监控、sql 合法性检查等，让 Expression2Sql 逐渐的日益完善。

由于 Expression2Sql 的职责非常单一、干净清爽，纯粹就是输入表达式树，然后经过它的解析之后，便可返回 Transact-SQL 给调用方。所以它的使用场景主要是用于和第三方的 ORM 或者是基于 ado.net 的原生 DbHelper 帮助类做对接，使其能够支持对象化、表达式树的链式编程。

Expression2Sql 源码托管地址：
https://github.com/StrangeCity/Expression2Sql
诸多开源项目收录：
http://www.cnblogs.com/StrangeCity/p/OpenSourceProject.html

拉轰兮兮的 YY 了这么久，那么接下来博主将以图文并茂的方式来展示一下 Expression2Sql 的使用示例。

二、单表简单查询

```
//通过静态属性DatabaseType或者静态方法Init均可配置数据库类型
Expre2Sql.DatabaseType = DatabaseType.SQLServer;
Expre2Sql.Init(DatabaseType.SQLServer);

Printf(
    Expre2Sql.Select<UserInfo>(),
    "查询单表所有字段"
);

Printf(
    Expre2Sql.Select<UserInfo>(u => u.Id),
    "查询单表单个字段"
);

Printf(
    Expre2Sql.Select<UserInfo>(u => new { u.Id, u.Name }),
    "查询单表多个字段"
);
```

三、Where 条件

3.1、where like

```
Printf(
    Expre2Sql.Select<UserInfo>(u => u.Id).
              Where(u => u.Name.Like("b")),
    "查询单表，带where Like条件"
);


Printf(
    Expre2Sql.Select<UserInfo>(u => u.Id).
              Where(u => u.Name.LikeLeft("b")),
    "查询单表，带where LikeLeft条件"
);


Printf(
    Expre2Sql.Select<UserInfo>(u => u.Id).
              Where(u => u.Name.LikeRight("b")),
    "查询单表，带where LikeRight条件"
);
```

```
Expression2SqlTest
查询单表，带where Like条件
select a.Id
from UserInfo a
where a.Name like '%' + @param0 + '%'
[@param0, b]


查询单表，带where LikeLeft条件
select a.Id
from UserInfo a
where a.Name like '%' + @param0
[@param0, b]


查询单表，带where LikeRight条件
select a.Id
from UserInfo a
where a.Name like  @param0 + '%'
[@param0, b]
```

3.2、where in

```
Printf(
    Expre2Sql.Select<UserInfo>(u => u.Name).
              Where(u => u.Id.In(1, 2, 3)),
    "查询单表，带where in条件，写法一"
);

int[] aryId = { 1, 2, 3 };
Printf(
    Expre2Sql.Select<UserInfo>(u => u.Name).
              Where(u => u.Id.In(aryId)),
    "查询单表，带where in条件，写法二"
);

Printf(
    Expre2Sql.Select<UserInfo>(u => u.Name).
              Where(u => u.Name.In(new string[] { "a", "b" })),
    "查询单表，带where in条件，写法三"
);
```

```
Expression2SqlTest
查询单表，带where in条件，
select a.Name
from UserInfo a
where a.Id in(1,2,3)


查询单表，带where in条件，
select a.Name
from UserInfo a
where a.Id in(1,2,3)


查询单表，带where in条件，
select a.Name
from UserInfo a
where a.Name in('a','b')
```

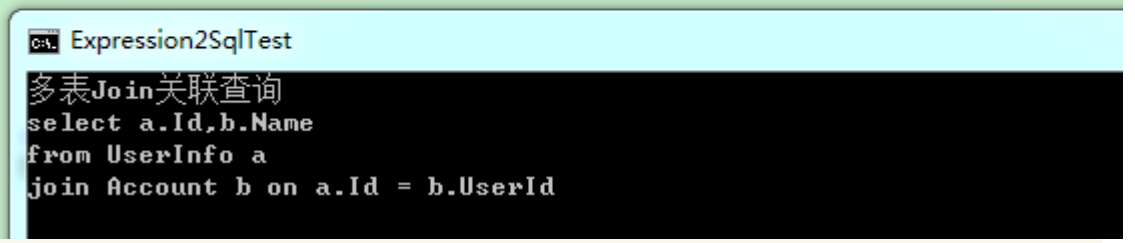
3.3、多个 where 条件组合

```
Printf(
    Expre2Sql.Select<UserInfo>(u => u.Id).
              Where(
                    u => u.Name == "b"
                      && u.Id > 2
                      && u.Name != null
                      && u.Id > int.MinValue
                      && u.Id < int.MaxValue
                      && u.Id.In(1, 2, 3)
                      && u.Name.Like("a")
                      && u.Name.LikeLeft("b")
                      && u.Name.LikeRight("c")
                      || u.Id == null
                    ),
    "查询单表，带多个where条件"
);
```

四、多表关联查询

4.1、join

```
Printf(
     Expre2Sql.Select<UserInfo, Account>((u, a) => new { u.Id, a.Name }).
               Join<Account>((u, a) => u.Id == a.UserId),
     "多表Join关联查询"
);
```



4.2、inner join

```
Printf(
     Expre2Sql.Select<UserInfo, Account>((u, a) => new { u.Id, a.Name }).
               InnerJoin<Account>((u, a) => u.Id == a.UserId),
     "多表InnerJoin关联查询"
);
```

```
Expression2SqlTest
多表InnerJoin关联查询
select a.Id,b.Name
from UserInfo a
inner join Account b on a.Id = b.UserId
```

4.3、left join

```
Printf(
     Expre2Sql.Select<UserInfo, Account>((u, a) => new { u.Id, a.Name }).
               LeftJoin<Account>((u, a) => u.Id == a.UserId),
     "多表LeftJoin关联查询"
);
```

```
Expression2SqlTest
多表LeftJoin关联查询
select a.Id,b.Name
from UserInfo a
left join Account b on a.Id = b.UserId
```

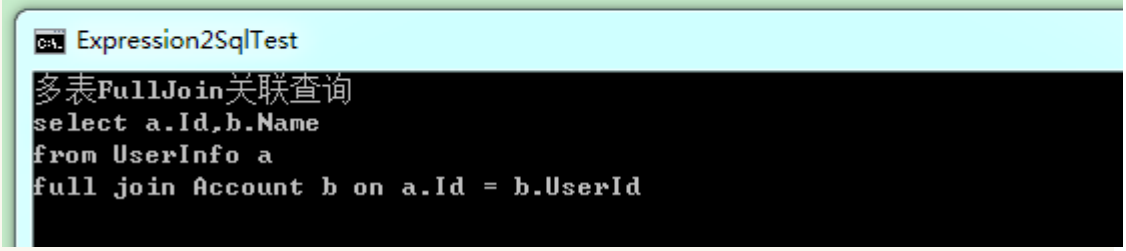
4.4、right join

```
Printf(
     Expre2Sql.Select<UserInfo, Account>((u, a) => new { u.Id, a.Name }).
               RightJoin<Account>((u, a) => u.Id == a.UserId),
     "多表RightJoin关联查询"
);
```

```
Expression2SqlTest
多表RightJoin关联查询
select a.Id,b.Name
from UserInfo a
right join Account b on a.Id = b.UserId
```

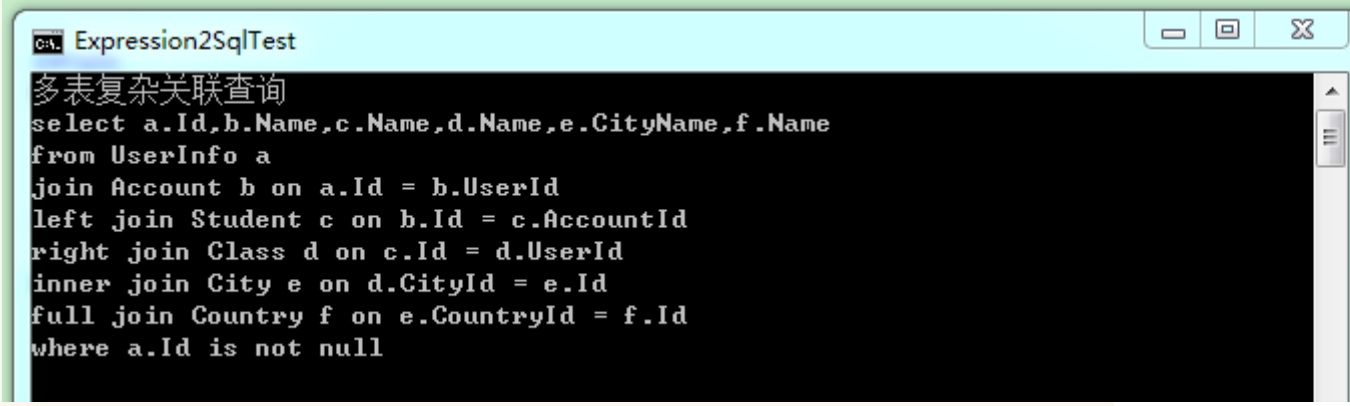
4.5、full join

```
Printf(
     Expre2Sql.Select<UserInfo, Account>((u, a) => new { u.Id, a.Name }).
               FullJoin<Account>((u, a) => u.Id == a.UserId),
     "多表FullJoin关联查询"
);
```



4.6、多表复杂关联查询

```
Printf(
     Expre2Sql.Select<UserInfo, Account, Student, Class, City, Country>((u, a, s, d, e, f) =>
               new { u.Id, a.Name, StudentName = s.Name, ClassName = d.Name, e.CityName, Count
               Join<Account>((u, a) => u.Id == a.UserId).
               LeftJoin<Account, Student>((a, s) => a.Id == s.AccountId).
               RightJoin<Student, Class>((s, c) => s.Id == c.UserId).
               InnerJoin<Class, City>((c, d) => c.CityId == d.Id).
               FullJoin<City, Country>((c, d) => c.CountryId == d.Id).
               Where(u => u.Id != null),
     "多表复杂关联查询"
);
```



五、group by

```
Printf(
      Expre2Sql.Count<UserInfo>(u => u.Id).
                GroupBy(u => u.Name),
      "GroupBy分组查询"
);
```

```
Expression2SqlTest
GroupBy分组查询
select count(a.Id) from UserInfo a
group by a.Name
```

六、order by

```
Printf(
      Expre2Sql.Select<UserInfo>().
                OrderBy(u => u.Id),
      "OrderBy排序"
);
```

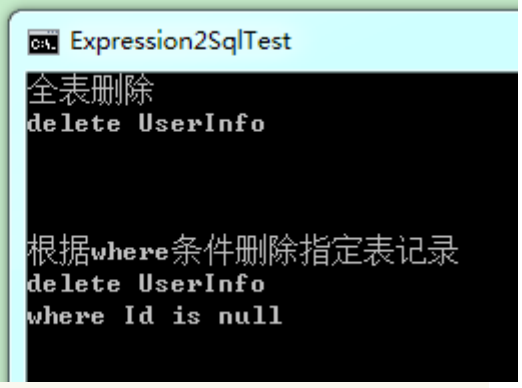
```
Expression2SqlTest
OrderBy排序
select *
from UserInfo a
order by a.Id
```

七、函数

```
Printf(
     Expre2Sql.Max<UserInfo>(u => u.Id),
     "返回一列中的最大值。NULL 值不包括在计算中。"
);

Printf(
     Expre2Sql.Min<UserInfo>(u => u.Id),
     "返回一列中的最小值。NULL 值不包括在计算中。"
);

Printf(
     Expre2Sql.Avg<UserInfo>(u => u.Id),
     "返回数值列的平均值。NULL 值不包括在计算中。"
);

Printf(
     Expre2Sql.Count<UserInfo>(),
     "返回表中的记录数"
);

Printf(
     Expre2Sql.Count<UserInfo>(u => u.Id),
     "返回指定列的值的数目（NULL 不计入）"
);

Printf(
     Expre2Sql.Sum<UserInfo>(u => u.Id),
     "返回数值列的总数（总额）。"
);
```

八、**delete** 删除

```
Printf(
     Expre2Sql.Delete<UserInfo>(),
     "全表删除"
);

Printf(
     Expre2Sql.Delete<UserInfo>().
               Where(u => u.Id == null),
     "根据where条件删除指定表记录"
);
```



九、**update** 更新

```
Printf(
     Expre2Sql.Update<UserInfo>(() => new { Name = "", Sex = 1, Email = "123456@qq.com" }),
     "全表更新"
);

Printf(
     Expre2Sql.Update<UserInfo>(() => new { Name = "", Sex = 1, Email = "123456@qq.com" }).
               Where(u => u.Id == 1),
     "根据where条件更新指定表记录"
);
```