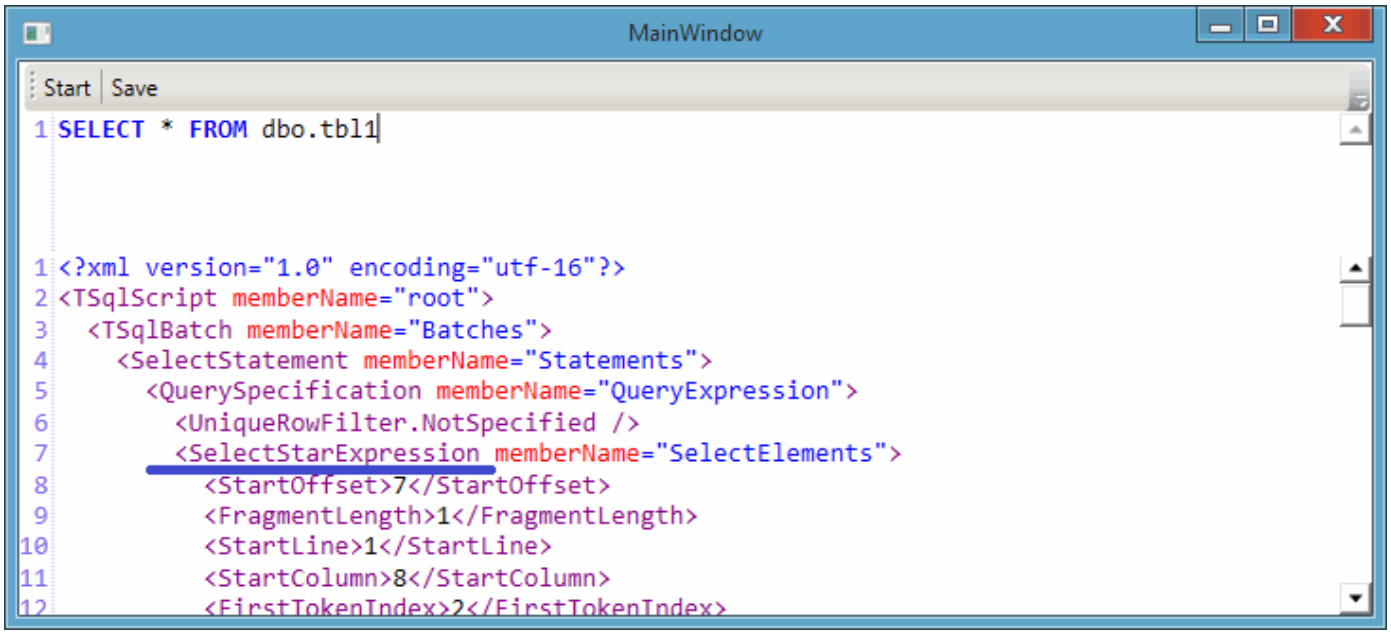
عنوان: **استفاده از SQLDom برای آنالیز عبارات T-SQL، قسمت دوم**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۳۰ ۱۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, T-SQL, ScriptDom

مدتی قبل مطلبی را در مورد کتابخانه‌ی ویژه SQL Server که یک T-SQL Parser تمام عیار است، در این سایت مطالعه کردید . در این قسمت، همان مطلب را به نحو بهتر و ساده‌تری بازنویسی خواهیم کرد.
مشکلی که در دراز مدت با SQLDom وجود خواهد داشت، مواردی مانند SelectStarExpression و CreateProcedureStatement و امثال آن هستند. این‌ها را از کجا باید تشخیص داد؟ همچنین مراحل بررسی این اجزاء، نسبتا طولانی هستند و نیاز به یک راه حل عمومی‌تر در این زمینه وجود دارد.

راه حلی برای این مشکل در مطلب « XML ‘Visualizer’ for the TransactSql.ScriptDom parse tree » ارائه شده‌است. در اینجا تمام اجزای TSqlFragment توسط Reflection مورد بررسی و استخراج قرار گرفته و نهایتا یک فایل XML از آن حاصل می‌شود.
اگر نکات ذکر شده در این مقاله را تبدیل به یک برنامه با استفاده مجدد کنیم، به چنین شکلی خواهیم رسید:



این برنامه را از اینجا می‌توانید دریافت کنید:
DomToXml.zip

همانطور که در تصویر مشاهده می‌کنید، اینبار به سادگی، SelectStarExpression قابل تشخیص است و تنها کافی است در T-SQL پردازش شده، به دنبال SelectStarExpressionها بود. برای اینکار جهت ساده شدن آنالیز می‌توان با ارث بری از کلاس پایه TSqlFragmentVisitor شروع کرد:

```
using System;
using System.Linq;
using Microsoft.SqlServer.TransactSql.ScriptDom;

namespace DbCop
{
    public class SelectStarExpressionVisitor : TSqlFragmentVisitor
    {
        public override void ExplicitVisit(SelectStarExpression node)
```

```
        {
            Console.WriteLine(
                  "`Select *` detected @StartOffset:{0}, Line:{1}, T-SQL: {2}",
                  node.StartOffset,
                  node.StartLine,
                  string.Join(string.Empty, node.ScriptTokenStream.Select(x => x.Text)).Trim());

            base.ExplicitVisit(node);
        }
    }
}
```

در کلاس پایه TSqlFragmentVisitor به ازای تمام اشیاء شناخته شده‌ی ScriptDom، یک متد ExplicitVisit قابل بازنویسی درنظر گرفته شده‌است. در اینجا برای مثال نمونه‌ی SelectStarExpression آن را بازنویسی کرده‌ایم.
مرحله‌ی بعد، اجرای این کلاس Visitor است:

```
public static class GenericVisitor
    {
        public static void Start(string tSql, TSqlFragmentVisitor visitor)
        {
            IList<ParseError> errors;
            TSqlScript sqlFragment;
            using (var reader = new StringReader(tSql))
            {
                var parser = new TSql120Parser(initialQuotedIdentifiers: true);
                sqlFragment = (TSqlScript)parser.Parse(reader, out errors);
            }

            if (errors != null && errors.Any())
            {
                var sb = new StringBuilder();
                foreach (var error in errors)
                    sb.AppendLine(error.Message);

                throw new InvalidOperationException(sb.ToString());
            }
            sqlFragment.Accept(visitor);
        }
    }
```

در اینجا متد Accept کلاس TSql120Parser، امکان پذیرش یک Visitor را دارد. به این معنا که Parser در حال کار، هر زمانیکه در حال آنالیز قسمتی از T-SQL دریافتی بود، نتیجه را به اطلاع یکی از متدهای کلاس پایه TSqlFragmentVisitor نیز خواهد رساند. بنابراین دیگر نیازی به نوشتن حلقه و بررسی تک تک اجزای خروجی TSql120Parser نیست. اگر نیاز به بررسی SelectStarExpression داریم، فقط کافی است Visitor آن‌را طراحی کنیم.

مثالی از نحوه‌ی استفاده از کلاس GenericVisitor فوق را در اینجا ملاحظه می‌کنید:

```
var tsql = @"WITH ctex AS (
SELECT * FROM sys.objects
)
SELECT * FROM ctex";
GenericVisitor.Start(tsql, new SelectStarExpressionVisitor());
```

عنوان: **چگونه یک عبارت sql را فرمت شده نمایش دهیم؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۰۹ ۱۲:۵۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, T-SQL, ScriptDom, DNTProfiler

در مورد کتابخانه‌ی SQLDom مطالبی را پیشتر در این سایت مطالعه کرده‌اید (^ و ^). یکی دیگر از کاربردهای آن، فرمت عبارات SQL است. برای مثال تبدیل عبارتی مانند

```
SELECT * FROM tb1 WHERE x1 = '12';
```

به نمونه‌ی فرمت شده‌ی آن:

```
SELECT *
FROM tb1
WHERE x1 = '12';
```

برای اینکار می‌توان از کلاس ذیل کمک گرفت:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using Microsoft.SqlServer.TransactSql.ScriptDom;

namespace SqlDomAnalyzer.Core
{
    public static class PrettyPrintTSql
    {
        public static string FormatTSql(string tSql)
        {
            IList<ParseError> errors;
            TSqlScript sqlFragment;
            using (var reader = new StringReader(tSql))
            {
                var parser = new TSql120Parser(initialQuotedIdentifiers: true);
                sqlFragment = (TSqlScript)parser.Parse(reader, out errors);
            }

            if (errors != null && errors.Any())
            {
                var sb = new StringBuilder();
                foreach (var error in errors)
                    sb.AppendLine(error.Message);

                throw new InvalidOperationException(sb.ToString());
            }

            var sql110ScriptGenerator = new Sql120ScriptGenerator(new SqlScriptGeneratorOptions
            {
                SqlVersion = SqlVersion.Sql120
            });
            string finalScript;
            sql110ScriptGenerator.GenerateScript(sqlFragment, out  finalScript);
            return finalScript;
        }
    }
}
```

در اینجا ابتدا عبارت SQL ورودی Parse شده و سپس به کتابخانه‌ی تولید اسکریپت ScriptDom ارسال می‌شود. خروجی آن، یک خروجی فرمت شده‌است.

نکته‌ی جالب دیگری که در اینجا وجود دارد، تهیه‌ی یک خروجی همواره یک شکل است. برای نمونه سه عبارت SQL زیر را در نظر بگیرید:

```
SELECT * from tb1 WHERE x1 = '12';
SELECT * from tb1 where x1 = '12';
select * from tb1 WHERE x1 = '12';
```

در اینجا در عبارت اول، from با حروف کوچک نمایش داده شده‌است. در عبارت دوم، where نیز با حروف کوچک نمایش داده شده‌است و در عبارت سوم اینکار در مورد select نیز تکرار شده‌است.
در هر سه حالت یا هر حالت قابل تصور دیگری، خروجی SQL فرمت شده‌ی حاصل یک چنین شکلی را دارد:

```
SELECT *
FROM tb1
WHERE x1 = '12';
```

موارد کاربرد آن؟

علاوه بر نمایش زیبای SQL فرمت نشده، احتمالا برنامه‌های Profiler ایی را دیده‌اید که عنوان می‌کنند قادرند عبارات SQL همانند را تشخیص دهند (جهت یافتن Lazy loading اشتباه). یک چنین خروجی یکسانی، قابلیت تهیه Hash عبارات SQL دریافتی را میسر می‌کند؛ چون دیگر اینبار مهم نیست که اجزای تشکیل دهنده‌ی یک عبارت SQL با حروف بزرگ هستند یا کوچک و فاصله‌ی بین آن‌ها چقدر است و آیا در این بین خطوط جدیدی نیز وجود دارند و امثال آن. خروجی نهایی نرمال شده‌ی توسط Sql120ScriptGenerator همواره یک شکل است. از این دو قابلیت در برنامه‌ی DNTProfiler استفاده شده‌است.