通常我们都是像下面这样使用一个DataSet或者DataTable:

```
DataSet dataSet = PrepareDataSet();
var userCount = (int)dataSet.Tables["Users"].Rows.Count;
var userName = dataSet.Tables["Users"].Rows[0]["Name"] as string;
var userAge = (int)dataSet.Tables["Users"].Rows[0]["Age"];
```

那么我们有没有办法像下面这样使用DataSet或者DataTable呢?

```
var userCount = dynamicDataSet.Users.Count;
var userName = dynamicDataSet.Users.Name[0];
var userAge = dynamicDataSet.Users.Age[0];
```

利用.NET4.0中新的DLR可以很容易的就实现这个功能，代码如下:

```
    class Program
    {
        static void Main(string[] args)
        {
            dynamic dynamicDataSet = new DynamicDataSet(PrepareDataSet());
            var userCount = dynamicDataSet.Users.Count;
            var userName = dynamicDataSet.Users.Name[0];
            var userAge = dynamicDataSet.Users.Age[0];
        }

        public static DataSet PrepareDataSet()
        {
            DataTable dataTable = new DataTable("Users");
            dataTable.Columns.Add("Name", typeof(string));
            dataTable.Columns.Add("Age", typeof(int));
            DataRow dataRow = dataTable.NewRow();
            dataRow[0] = "Davis";
            dataRow[1] = "31";
            dataTable.Rows.Add(dataRow);
            DataSet dataSet = new DataSet();
            dataSet.Tables.Add(dataTable);
            return dataSet;
        }
    }
```

```
using System;
using System.Data;
using System.Dynamic;

namespace Zxf.ExpressionBuilder
```

```
{
    public class DynamicDataSet : DynamicObject
    {
        private readonly Object m_Data;
        private readonly string m_Field;

        public DynamicDataSet(DataSet obj)
        {
            m_Data = obj;
        }

        public DynamicDataSet(DataTable obj)
        {
            m_Data = obj;
        }

        public DynamicDataSet(DataRowCollection obj, string field)
        {
            m_Data = obj;
            m_Field = field;
        }

        public override bool TryGetIndex(GetIndexBinder binder, object[] indexes, out obje
        {
            if (m_Data is DataRowCollection)
            {
                var dtRows = m_Data as DataRowCollection;
                if (binder.CallInfo.ArgumentCount == 1 && indexes[0] is int)
                {
                    result = dtRows[Convert.ToInt32(indexes[0].ToString())][m_Field];
                    if (result is string)
                    {
                        result = result.ToString().Trim();
                    }
                    return true;
                }
            }

            result = null;
            return false;
        }

        public override bool TryGetMember(GetMemberBinder binder, out object result)
        {
            if (m_Data is DataSet)
            {
                var ds = m_Data as DataSet;
```

```
        if (ds.Tables.Contains(binder.Name))
        {
            result = new DynamicDataSet(ds.Tables[binder.Name]);
            return true;
        }
        result = null;
        return false;
    }

    if (m_Data is DataTable)
    {
        var dt = m_Data as DataTable;
        if (dt.Columns.Contains(binder.Name))
        {
            result = new DynamicDataSet(dt.Rows, binder.Name);
            return true;
        }
        if (binder.Name.ToUpper() == "COUNT")
        {
            result = dt.Rows.Count;
            return true;
        }
        result = null;
        return false;
    }

    if (m_Data is DataRowCollection && binder.Name.ToUpper() == "SUM")
    {
        var dataRows = m_Data as DataRowCollection;
        dynamic sumResult = 0;
        foreach (DataRow row in dataRows)
        {
            if (row[m_Field] is decimal)
            {
                sumResult = (decimal)sumResult + (dynamic)row[m_Field];
            }
            else
            {
                sumResult += (dynamic)row[m_Field];
            }

        }
        result = sumResult;
        return true;
    }

    result = null;
```

```
            return false;
        }
    }
}
```

也可以利用Expression.Dynamic来实现动态执行之前的dynamicDataSet.Users.Count语句:

```
    using System;
    using System.Data;
    using System.Linq.Expressions;
    using System.Runtime.CompilerServices;
    using Microsoft.CSharp.RuntimeBinder;
    using Zxf.ExpressionBuilder;

    class Program
    {
        static void Main(string[] args)
        {
            ParameterExpression expressionParameter = DynamicExpression.Parameter(typeof(D

            CallSiteBinder binderGetUsers = Microsoft.CSharp.RuntimeBinder.Binder.GetMembe
                                                , new CSharpArgumentInfo[] { CSharpArg
            DynamicExpression getUsersExpression = Expression.Dynamic(binderGetUsers, type

            CallSiteBinder binderGetCount = Microsoft.CSharp.RuntimeBinder.Binder.GetMembe
                                                , new CSharpArgumentInfo[] { CSharpArg
            DynamicExpression getCountExpression = Expression.Dynamic(binderGetCount, type

            Expression<Func<DynamicDataSet, object>> dynamicLambdaExp = Expression.Lambda<

            Func<DynamicDataSet, object> dynamicLambdaExpFunc = dynamicLambdaExp.Compile()

            dynamic dynamicDataSet = new DynamicDataSet(PrepareDataSet());
            int? count = dynamicLambdaExpFunc.Invoke(dynamicDataSet) as int?;
        }

        public static DataSet PrepareDataSet()
        {
            DataTable dataTable = new DataTable("Users");
            dataTable.Columns.Add("Name", typeof(string));
            dataTable.Columns.Add("Age", typeof(int));
            DataRow dataRow = dataTable.NewRow();
            dataRow[0] = "Davis";
            dataRow[1] = "31";
            dataTable.Rows.Add(dataRow);
            DataSet dataSet = new DataSet();
            dataSet.Tables.Add(dataTable);
```

```
            return dataSet;
        }
    }
```