КАК СДЕЛАТЬ CLR-ФУНКЦИЮ ДЛЯ MS SQL

By **devSonia** | **2014-01-28** | **Крафт** | **C#**, **CLR**, **MS SQL**, **функции**

Все знают, что для MS SQL можно писать CLR-примочки. Но не все знают, как. Чтобы не лазить и не искать каждый раз, выкладываю сюда краткий мануал по тому, как это сделать.

Общий алгоритм такой:

- Создаем проект «Библиотека классов» в MS Visual Studio.
- Создаем класс, в котором будем реализовывать наши методы. Число классов в одной сборке не ограничено.
- Для работы с типами данных, используемых сервером, подключим namespace System.Data.SqlTypes, так же потребуется Microsoft.SqlServer.Server для определения атрибутов.
- Пропишем методы (будущие процедуры и функции). Ставим ему атрибуты, которые нужны для превращения метода в CLR-процедуру или функцию. О них подробнее ниже.
- Реализуем метод. При реализации, а также при описании параметров и возвращаемого значения руководствуемся следующим сопоставлением типов:

Тип MS SQL	Тип .Net
BIT	System.Boolean
TINYINT	System.Int16
INT	System.Int32
BIGINT	System.Int64
SMALLMONEY	System.Decimal
MONEY	System.Decimal
DECIMAL	System.Decimal
NUMERIC	System.Decimal
REAL/FLOAT(24)	System.Single
FLOAT/FLOAT(53)	System.Double
CHAR	System.String
NCHAR	System.String
VARCHAR	System.String

NVARCHAR	System.String
TEXT	System.String
NTEXT	System.String
XML	System.Xml.Linq.XElement
SMALLDATETIME	System.DateTime
DATETIME	System.DateTime
DATETIME2	System.DateTime
DATETIMEOFFSET	System.DateTimeOffset
DATE	System.DateTime
TIME	System.TimeSpan
BINARY(50)	System.Data.Linq.Binary
VARBINARY(50)	System.Data.Linq.Binary
VARBINARY(MAX)	System.Data.Linq.Binary
IMAGE	System.Data.Linq.Binary
TIMESTAMP	System.Data.Linq.Binary
UNIQUEIDENTIFIER	System.Guid
SQL_VARIANT	System.Object

- Компилируем сборку.
- Подключаем сборку в MS SQL:

```
alter database DemoCLR set trustworthy on
go

create assembly DemoCLR
from 'ПутьКСборке\ИмяСборки.dll'
with permission_set = unsafe
go
```

- Создаем «оболочки» для своих методов.

Теперь подробнее о каждом типе методов. Через CLR можно подключать процедуры, скалярные, табличные и агрегатные функции.
а) Скалярная функция:

```
namespace DemoClr
{
// Класс, в котором будет скалярная функция
public class ScalarFunction
{
// Этот атрибут необходимо указать, чтобы функцию можно было подключить в MS SQL
[SqlFunctionAttribute()]
// Параметры и возвращаемый тип следует брать из пространства System.Data.SqlTypes
public static SqlDouble Add(SqlDouble A, SqlDouble B)
{
return A + B;
```

```
}
}
}
```

```
-- Определяем имя будущей функции, указываем типы (см. таблицу соответствия) параметров
create function dbo.my_clr_add(@A float,@b float)
-- и возвращаемого значения
returns float
as
-- указываем, что на самом деле наша функция внешняя, лежит в подключенной сборке DemoClr
-- в классе ScalarFunction (относящемуся так же к namespace DemoClr, что указано с помощью
-- квадратных скобок), и имеет имя Add.
external name DemoCLR.[DemoClr.ScalarFunction].[Add]
go
```

Пример работы с функцией:

```
select dbo.my_clr_add(1,2)

----------------------
3

(строк обработано: 1)
```

б) Функция, возвращающая табличное значение.

```
namespace DemoClr
{
// Класс, в котором будет реализована наша функция
public class TableValuedFunction
{
// Атрибут, аналогичный предыдущему случаю, но для табличной функции
// необходимо указать специальный метод для "заполнения" таблицы
[SqlFunction(FillRowMethodName = "GenerateIntervalFillRow")]

// Сама табличная функция возвращает IEnumerable (именно его элементы и будут ходить
// в метод-заполнитель), типы параметров желательно использовать вновь из
// System.Data.SqlTypes
public static IEnumerable GenerateInterval(SqlInt32 From, SqlInt32 To)
{
int[] items = new int[To.Value - From.Value + 1];
for (int i = From.Value; i <= To.Value; i++)
items[i - From.Value] = i;
return items;
}

//Этот служебный метод нужен для получения одной строки набора данных и приведения её
public static void GenerateIntervalFillRow(object o, out SqlInt32 item)
{
item = new SqlInt32((int)o);
}
}
}
```

```
-- Определяем имя будущей функции, указываем типы (см. таблицу соответствия) параметров
create function dbo.my_clr_interval(@A int,@b int)
-- и описываем выходную таблицу
```

```
returns table (ID int)
as
-- указываем, что на самом деле наша функция внешняя, лежит в подключенной сборке DemoClr
-- в классе TableValuedFunction (относящемуся так же к namespace DemoClr, что указано с
помощью
-- квадратных скобок), и имеет имя GenerateInterval.
external name DemoCLR.[DemoClr.TableValuedFunction].GenerateInterval
go
```

Пример работы с функцией:

```
select * from dbo.my_clr_interval(5,10)

ID
-----------
5
6
7
8
9
10

(строк обработано: 6)
```

Про агрегатные функции и хранимые процедуры — чуть позже.

‹ Разница характеров

Веселая и оригинальная реклама mercedes benz «Курица» ›