

# SQL Server интеграция с .NET Использование SQLCLR

# SQL Server 2008

## Интеграция с .NET

### *Что даёт интеграция с .NET*

- Улучшенная модель программирования
- Улучшенная надежность и безопасность
- Возможность определять типы данных и статистические функции
- Упрощение процесса разработки в результате стандартизации среды
- Возможность повышения производительности и масштабируемости

### *Обратите внимание*

- Когда следует использовать процедуры CLR
- Когда не следует использовать процедуры CLR

# Основные понятия

- **Библиотека классов** - множество predetermined классов, которые могут быть использованы для реализации полной функциональности.
- **Пространство имён** - группы объектов, выполняющих сходные функции.
- **Сборки** - группировка программы средой .NET
- **Домен приложения** - второй уровень группировки программ в среде .NET (домен приложения состоит из группы сборок, загружаемых вместе)

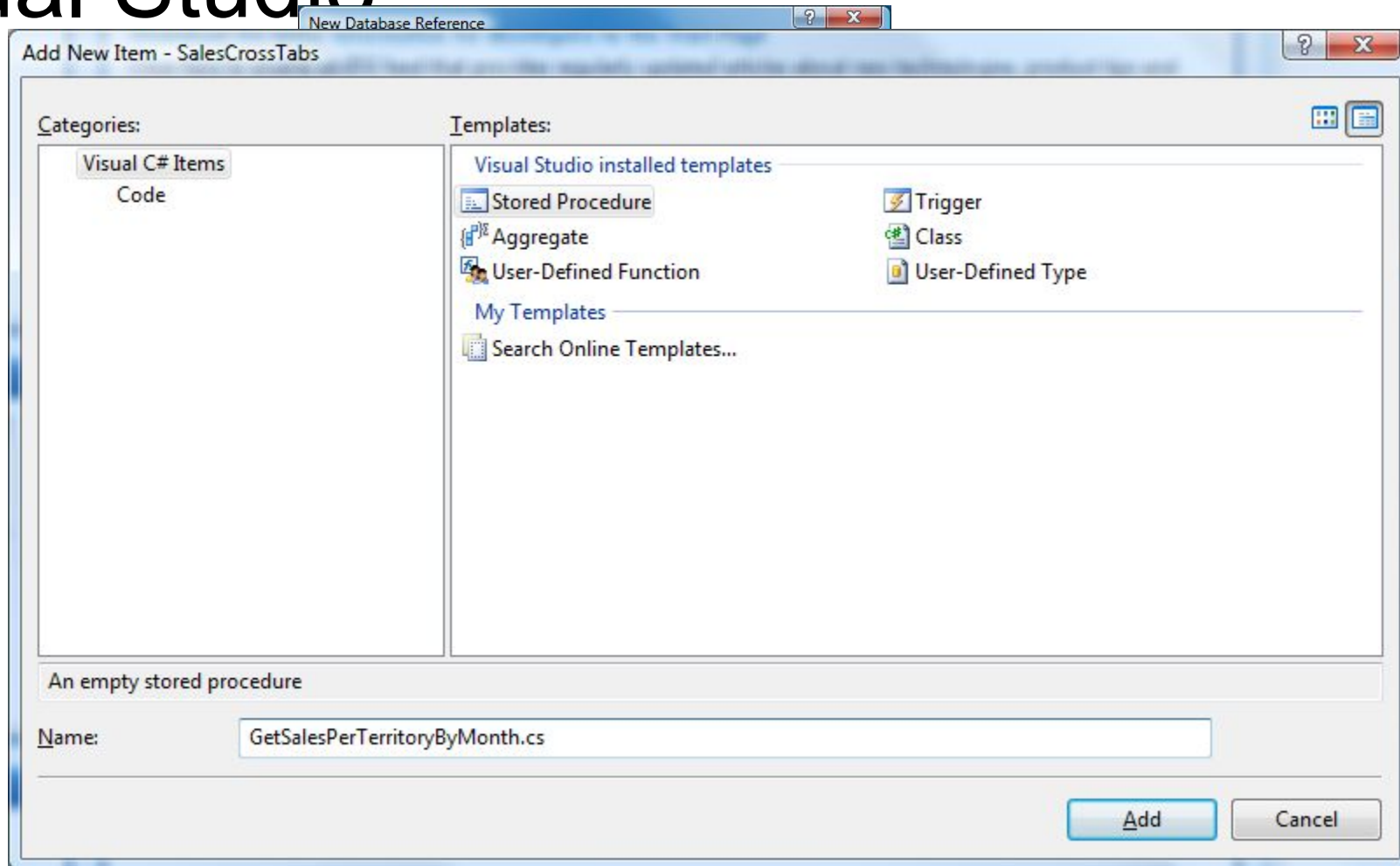
# Сценарий реализации

- Создать сущность SQL сервера, которая бы позволила запускаться CLR коду.
- Написать код (на C# или Visual Basic .NET), который выполняет действия с объектами базы данных.
- Скомпилировать написанный код в сборку с помощью CLR компилятора.
- Загрузить сборку в SQL сервер.
- Создать объект в БД и точку входа в сборку, используя Data Definition Language (DDL).

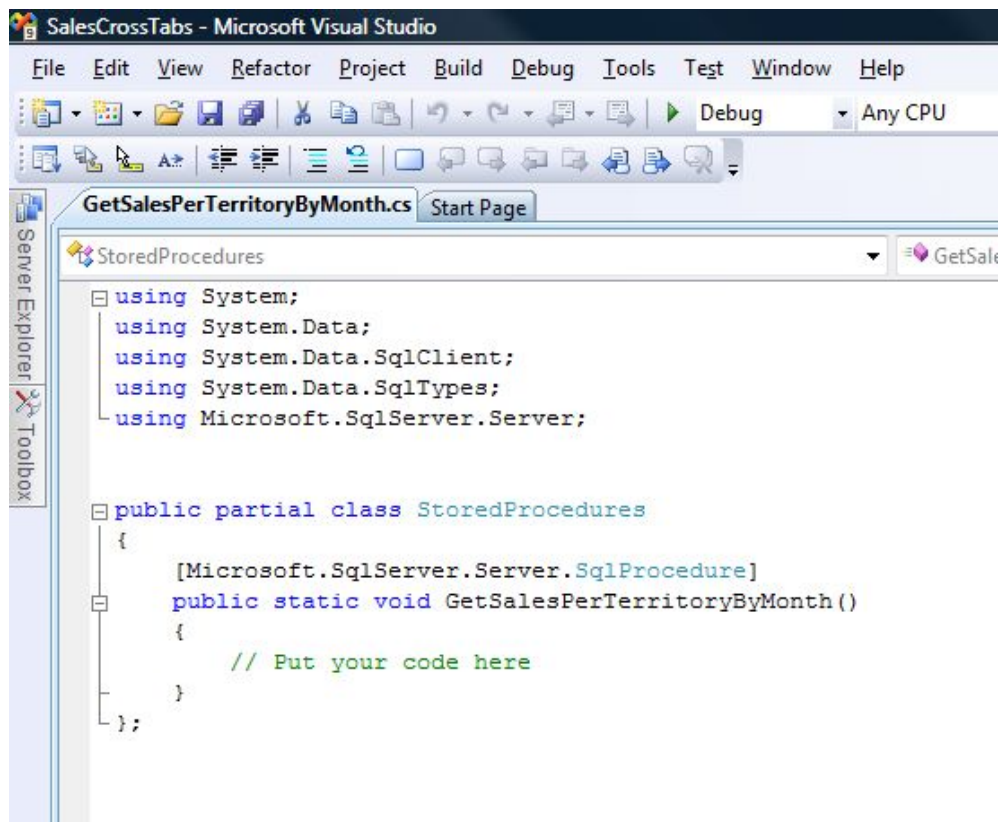
# Объекты, которые можно создавать с помощью SQLCLR

- хранимые процедуры;
- функции определенные пользователем, которые возвращают единичное значение;
- функции определенные пользователем, которые возвращают таблицу и могут быть вызваны из FROM, JOIN, APPLY
- триггеры(DML, DDL и триггеры по логину)
- агрегаты определённые пользователем
- типы определённые пользователем.

# Запуск проекта SQL Server в Visual Studio



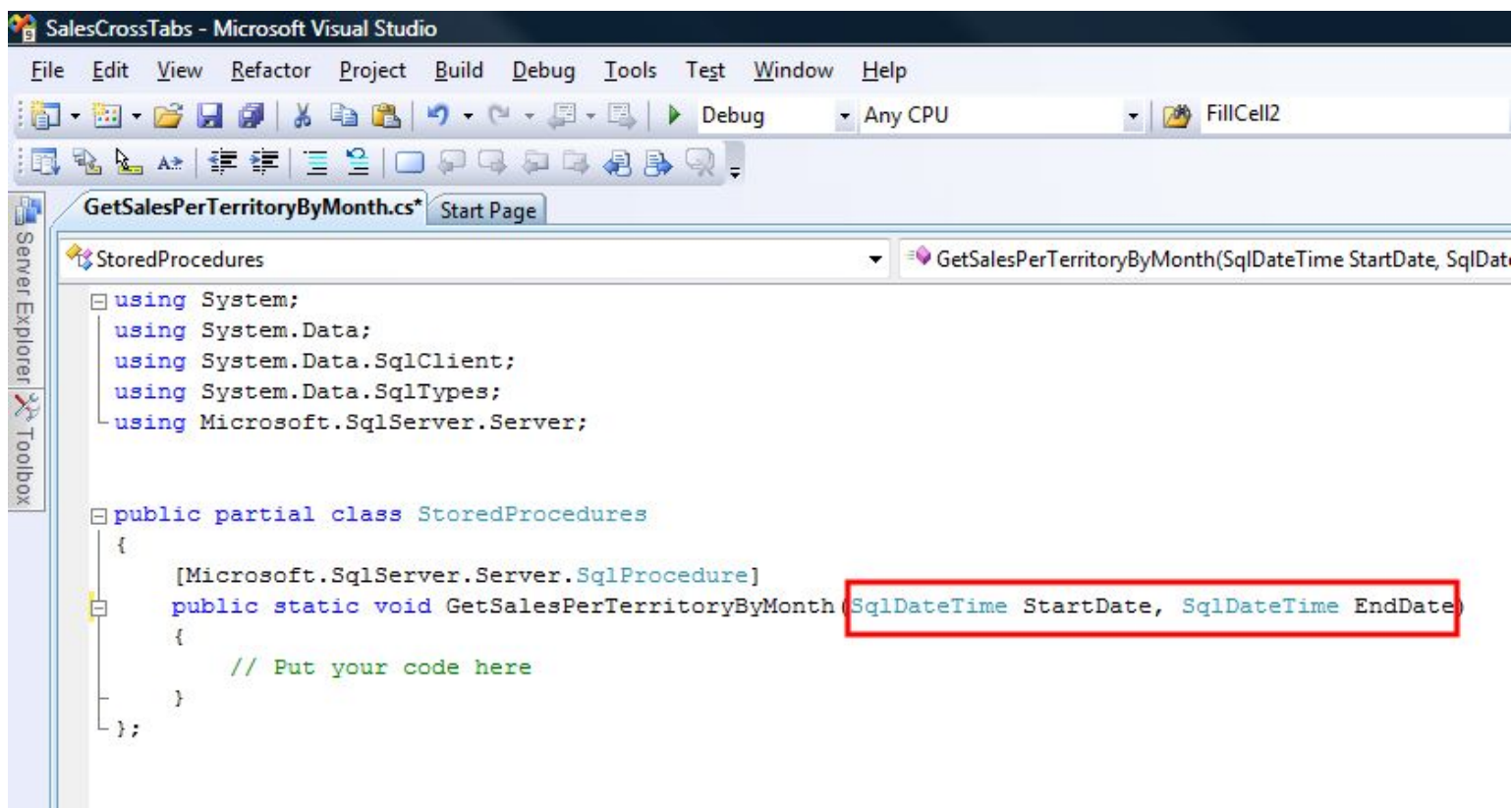
# Внутренняя структура хранимой процедуры



```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;

public partial class StoredProcedures
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void GetSalesPerTerritoryByMonth()
    {
        // Put your code here
    }
};
```

# Добавление параметров



```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;

public partial class StoredProcedures
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void GetSalesPerTerritoryByMonth(SqlDateTime StartDate, SqlDateTime EndDate)
    {
        // Put your code here
    }
};
```



# Объект SQLPipe

- **Метод Send()**

```
command.CommandText = "SELECT * FROM  
Sales.SalesOrderHeader"; SqlDataReader reader =  
command.ExecuteReader();  
SqlConnection.Pipe.Send(reader);
```

- **Метод ExecuteAndSend()**

```
SqlCommand command = new SqlCommand("SELECT  
VendorID, AccountNumber, Name FROM Purchasing.Vendor  
WHERE CreditRating <= @rating", connection);  
command.Parameters.AddWithValue("@rating", rating);  
SqlConnection.Pipe.ExecuteAndSend(command);
```

# Тестирование процедуры CLR

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows the Object Explorer for the AdventureWorks database, with the procedure `dbo.GetSalesPerTerritoryByMonth` highlighted. To the left, the Output window shows the build and deployment logs, indicating a successful process. To the right, a portion of another window is visible, showing file paths related to the assembly.

**Output Window:**

```
Show output from: Build

----- Build started: Proj
C:\Windows\Microsoft.NET\F

Compile complete -- 0 erro
SalesCrossTabs -> d:\VSPro
Drop assembly: SalesCrossT
----- Deploy started: Pro
Deploying file: SalesCross'
Deploying file: SalesCross'
Deploying file: GetSalesPe
Deploying file: Properties
===== Build: 1 succee
===== Deploy: 1 succee
```

**Object Explorer:**

- AdventureWorks
- Database Diagrams
- Tables
- Views
- Synonyms
- Programmability
  - Stored Procedures
    - System Stored Procedures
    - dbo.AddEmp
    - dbo.GetSalesPerTerritoryByMonth**
    - dbo.uspGetBillOfMaterials
    - dbo.uspGetEmployeeManagers
    - dbo.uspGetManagerEmployeees
    - dbo.uspGetWhereUsedProductID
    - dbo.uspLogError
    - dbo.uspPrintError
    - HumanResources.uspUpdateEmployeeHireInfo
    - HumanResources.uspUpdateEmployeeLogin
    - HumanResources.uspUpdateEmployeePersonalInfo
- Functions
- Database Triggers

**File Path Window:**

```
reference:C:\Windows\Micr
...
:sCrossTabs.dll
:sCrossTabs.pdb
:tSalesPerTerritoryByMonth
ties\AssemblyInfo.cs ...
```

# Интеграция с CLR в SQL Server

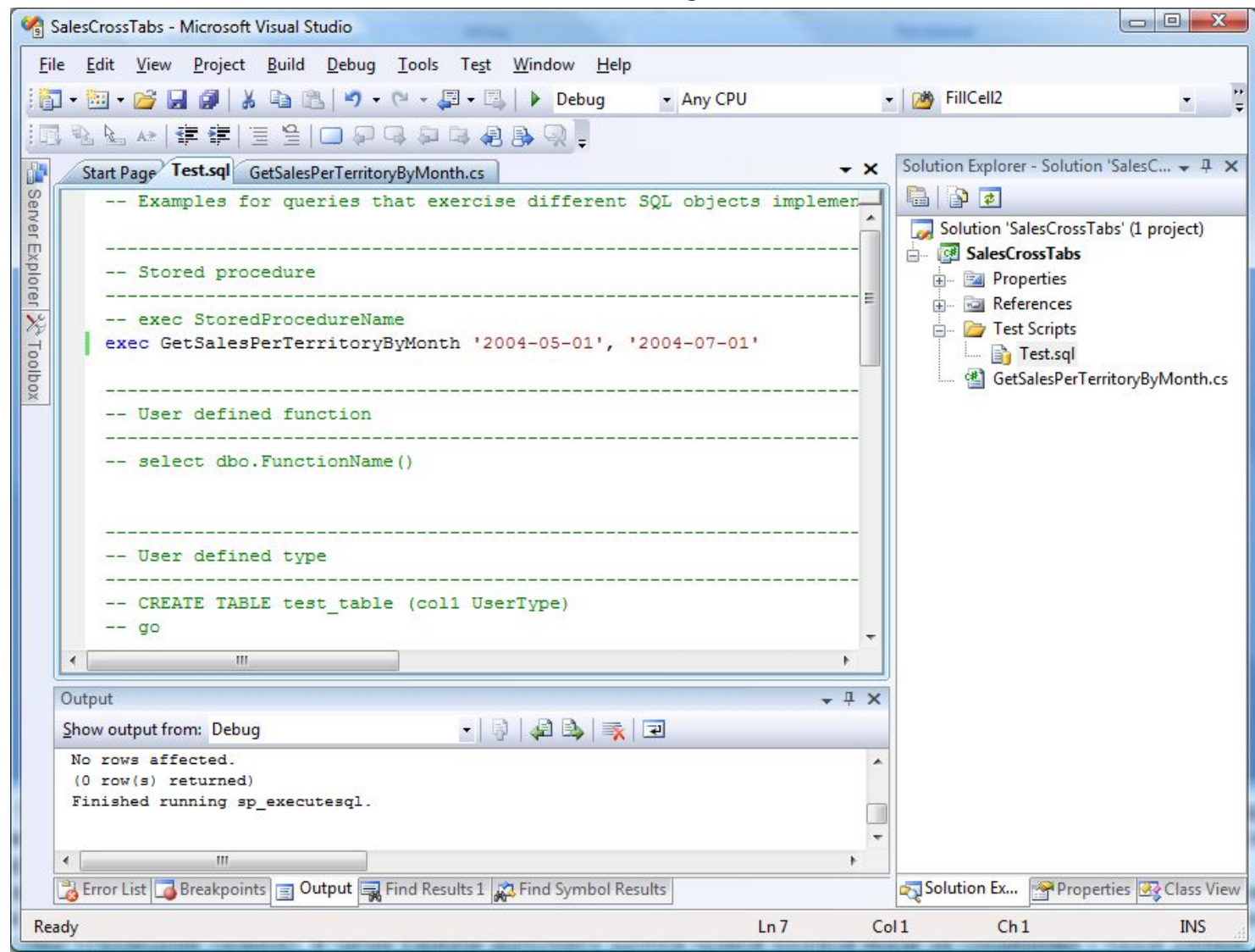
## Messages

Msg 6263, Level 16, State 1, Line 1

Execution of user code in the .NET Framework is disabled. Enable "clr enabled" configuration option.

```
1 exec sp_configure 'clr enabled', 1
2 reconfigure
```

# Отладка процедуры



# Развертывание процедуры

Window Help

```
CREATE ASSEMBLY assembly_name
[ AUTHORIZATION owner_name ]
FROM { <client_assembly_specifier> | <assembly_bits> [ ,...n ] }
[ WITH PERMISSION_SET = { SAFE | EXTERNAL_ACCESS | UNSAFE } ]
[ ; ]
```

```
1 create procedure GetSalesPerTerritoryByMonth
2 @startDate datetime,
3 @endDate datetime
4 as
5 external name SalesCrossTabs.StoredProcedures.
   GetSalesPerTerritoryByMonth
```

# Литература

<http://msdn.microsoft.com/ru-ru/library/ms131102.aspx>

<http://blogs.msdn.com/alexejs/archive/2009/05/12/clr.aspx>

<http://blogs.msdn.com/alexejs/archive/2009/05/15/p20090515.aspx>

<http://blogs.msdn.com/alexejs/archive/2009/05/11/0-9-8-7-6-5-5-6.aspx>

[http://blogs.msdn.com/alexejs/archive/2009/05/20/p20090520\\_5F00\\_1.aspx](http://blogs.msdn.com/alexejs/archive/2009/05/20/p20090520_5F00_1.aspx)

<http://scherbinin.blog.ru/58761875.html>

<http://www.gotdotnet.ru/LearnDotNet/CSharp/510966.aspx>

<http://www.osp.ru/text/print/302/380182.html>

<http://win2008.ru/2009/03/11/dbd-nyuans-v-sozdanii-triggerov-clr/>