Abstract   
In this article Soyuj describes different type of triggers, more specially CLR triggers, and their uses. It covers examples for creating T-SQL as well as CLR Triggers. by [Soyuj Kumar Sahoo](http://aspalliance.com/author.aspx?uId=67242)  
[Feedback](http://aspalliance.com/feedback.aspx?uId=190698)  
Average Rating: http://aspalliance.com/images/1x1.gifhttp://aspalliance.com/images/1x1.gif  
**Views (Total / Last 10 Days):** 65897/ 542

**Article Contents**:

* [Introduction](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page1)
* [What is Trigger?](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page2)
* [Types of Triggers](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page3)
* [DML Triggers](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page4)
* [DDL Triggers](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page5)
* [CLR Triggers](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page6)
* [Steps for Creating CLR Trigger](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page7)
* [Checking the triggering action](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page8)
* [Conclusion](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#Page9)

[](http://a.lakequincy.com/c.ashx?channel=1&format=7&pageid=C38DF2D9-AF6A-52D3-EA17-7B21925D8003&publisher=1&ypos=358&zone=4&url=http%3a%2f%2faspalliance.com%2f1273_CLR_Triggers_for_SQL_Server_2005.all&country=VN&userguid=5ef54fdd-0e95-463a-ae2b-0d3c76468d39&placement=4989&creative=3846)

# Introduction

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

As I discussed in my [previous](http://aspalliance.com/1157) articles, now Microsoft SQL Server 2005 is integrated with Microsoft .NET Framework Common Language Runtime (CLR), so we can use any .NET Framework language to create database objects. The CLR provides the execution environment for all the server side objects that are created using a .NET language. This means the database developers can now perform tasks that were impossible or difficult to achieve with T-SQL alone. Especially when working with large amounts of server code, developers can easily organize and maintain their code investments.

This article covers different types of triggers supported by Microsoft SQL Server 2005 with basic ideas about them with an example. It then describes the step-by-step approach to create a CLR trigger (a DML type).

# What is Trigger?

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

A trigger is a Database object just like a stored procedure or we can say it is a special kind of Stored procedure which fires after (/before) a specified language event executes. More specifically, it is for the object which is attached to a Table or View or Database schemas for tracking the operations on them. The main difference between a trigger and a stored procedure is that the former is attached to a table or view and is fired only when an INSERT, UPDATE, and/or DELETE occurs, while a stored procedure executes at any time when it is called.

# Types of Triggers

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

There are some added types in SQL Server 2005 for triggering actions:

1.    DML Triggers

·         AFTER Triggers

·         INSTEAD OF Triggers

2.    DDL Triggers

3.    CLR Triggers

# DML Triggers

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

These triggers are fired when a Data Manipulation Language (DML) event takes place. These are attached to a Table or View and are fired only when an INSERT, UPDATE and/or DELETE event occurs. The trigger and the statement that fires it are treated as a single transaction. Using this we can cascade changes in related tables, can do check operations for satisfying some rules and can get noticed through firing Mails. We can even execute multiple triggering actions by creating multiple Triggers of same action type on a table. We have to specify the modification action(s) at the Table level that fires the trigger when it is created.

# AFTER Triggers

As the name specifies, AFTER triggers are executed after the action of the INSERT, UPDATE, or DELETE statement is performed. This is the only option available in earlier versions on Microsoft SQL Server. AFTER triggers can be specified on tables only. Here is a sample trigger creation statement on the Users table.

Listing 1 (AFTER Trigger example)

------ Creating a DML trigger in T-SQL -------

SET NOCOUNT ON

CREATE TABLE UserTable (User\_ID int IDENTITY, User\_Name varchar(30), Type varchar(10))

go

CREATE TRIGGER tr\_User\_INSERT

ON UserTable

FOR INSERT

AS

PRINT GETDATE()

Go

INSERT UserTable (User\_Name, Type) VALUES ('James', 'ADMIN')

------ Result ---------------

Apr 30 2007  7:04AM

# INSTEAD OF Triggers

INSTEAD OF triggers are executed in place of the usual triggering action. INSTEAD OF triggers can also be defined on views with one or more base tables, where they can extend the types of updates a view can support.

# DDL Triggers

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

DDL triggers are new to SQL Server 2005. This type of triggers, like regular triggers, fire stored procedures in response to an event. They fire in response to a variety of Data Definition Language (DDL) events. These events are specified by the T-SQL statements that are start with the keywords CREATE, ALTER, and DROP. Certain stored procedures that perform DDL-like operations can also fire this. These are used for administrative tasks like auditing and regulating database operations.

# CLR Triggers

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

A CLR triggers can be any of the above, e.g. can be a DDL or DML one or can also be an AFTER or INSTEAD OF trigger. Here we need to execute one or more methods written in managed codes that are members of an assembly created in the .Net framework. Again, that assembly must be deployed in SQL Server 2005 using CREATE assembly statement.

The Microsoft.SqlServer.Server Namespace contains the required classes and enumerations for this objective.

# Steps for Creating CLR Trigger

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

The following are required steps for creating a CLR-Trigger of DML (After trigger) type for Insert action. This database Object is executed as the result of a user action against a table i.e. for an INSERT statement.

·         Creating a .NET class of triggering action

·         Making assembly(.DLL) from that Class

·         Enabling CLR environment in that database.

·         Registering the assembly in SQL Server

·         Creating CLR Trigger using that assembly

## 1. Creating a .NET class

Here we can use any managed language that is supported by .Net Framework such as C++, C#, VB, J#, JScript or XAML, etc. As I am with VB, this managed code is in Visual Basic. Let us discuss the objective of this entity. According to the above example of "tr\_User\_INSERT" trigger, we have the UserTable for holding the user details. There is a field "Type" which explains the user role (ADMIN, End User, Register User etc.).  Our objective is to check the role of the inserted User for ADMIN type and then do the action as we set.

Open the notepad, copy the following codes and save it as MyFirstAssembly.vb.

Listing 2 (.NET Class of Trigger type)

*Imports System.Data*

*Imports System.Data.SqlClient*

*Imports Microsoft.SqlServer.Server*

*Partial Public Class MyFirstClrTrigger*

*<Microsoft.SqlServer.Server.SqlTrigger(Name*

*= "checkUserRole", Target*

*= "UserTable", Event*

*= "FOR INSERT")> \_*

*Public Shared Sub checkUserRole()*

*Dim triggContext As SqlTriggerContext = SqlContext.TriggerContext()*

*Dim userType As String = String.Empty*

*If triggContext.TriggerAction = TriggerAction.Insert Then*

*Using connection As New SqlConnection("context connection=true")*

*connection.Open()*

*Dim sqlComm As New SqlCommand*

*Dim sqlPipe As SqlPipe = SqlContext.Pipe()*

*sqlComm.Connection = connection*

*sqlComm.CommandText = "SELECT Type from INSERTED"*

*userType = sqlComm.ExecuteScalar.ToString()*

*If userType.ToUpper = "ADMIN" Then*

*sqlPipe.Send("Hello !!! You have the Admin role.")*

*sqlPipe.Send("We can use e-mail codes here to inform.")*

*End If*

*End Using*

*End If*

*End Sub*

*End Class*

Let us go into the codes. There are two major Namespaces used, System.Data.SqlClient and Microsoft.SqlServer.Server. Microsoft.SqlServer.Server provides the SqlTriggerAttribute Class, which is used to mark a method definition in an assembly as a trigger in SQL Server. The Sqltrigger attribute requires some parameters to set the created trigger properties, such as Name - the name of the Trigger, Target - the table or view to which the trigger applies (in case of DML type) and Event - the event to fire the trigger for (e.g. FOR INSERT, DELETE and/or UPDATE or INSTEAD OF etc.). Again, that method must be a Static (Shared in VB) one; here checkUserRole() is the target method.

The SqlTriggerContext class provides the required triggering properties of the Trigger for doing action. TriggerAction property of this Class is the global enumeration TriggerAction type of Microsoft.SqlServer.Server namespace which indicates what action fired the Trigger.

Here we use the INSERTED table, which is automatically created and managed by SQL Server 2005. This is used to set the conditions of DML trigger action. There is also another called DELETED, used in case of delete action. The CLR triggers can access the Inserted or Deleted tables through SqlCommand object using context connection.

## 2. Making assembly (.DLL)

Now we have created the MyFirstAssembly.vb file with MyFirstClrTrigger class, so we need to compile the class to create an assembly. Here I use the VB compiler "vbc.exe," found in .Net   Framework library. Use the following DOS command for creating assembly.

*Listing 3 (compile the .VB file)*

*C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>vbc*

*/t:library /out:F:\CLR\_Trigger\_Test\CLR\_Assembly\MyFirstAssembly*

*F:\CLR\_Trigger\_Test\MyFirstClrTrigger.vb*

This command creates a DLL called MyFirstAssembly.dll at out path F:\CLR\_Trigger\_Test\CLR\_Assembly\.

## 3. Enabling CLR environment

By default, the CLR functionality is trued off in SQL Server; so we need to enable it. Use the following T-SQL codes to enable the CLR functionality.

Listing 4 (enable CLR action)

*------ Enabling CLR action in database -------*

*sp\_configure 'clr enabled', 1*

*Go*

*RECONFIGURE with Override*

*Go*

## 4. Registering the assembly

Our next step is to create an assembly in the Database, based on the compiled DLL (MyFirstAssembly.dll). The following T-SQL codes are useful regarding this objective.

Listing 5 (registering assembly)

------ Registering an Assembly -------

Create Assembly UserAssembly

From 'F:\CLR\_Trigger\_Test\CLR\_Assembly\MyFirstAssembly.dll'

With Permission\_Set=Safe

Go

For more information about this check my [previous article](http://aspalliance.com/1081_CLR_Integration_in_SQL_Server_2005).

## 5. Creating CLR Trigger

Now we will create an extended Trigger using CREATE Trigger statement of T-SQL. There is a new clause named EXTERNAL NAME in SQL Server 2005, which allows us to reference a method of the Registered assembly. By doing so we set the triggering action of our Trigger using that managed code method of the assembly.

## Listing 6 (creating a Trigger)

------ Creating a DML trigger in CLR -------

Create Trigger tr\_User\_CheckRole

on UserTable

For INSERT

AS

External Name UserAssembly.MyFirstClrTrigger.checkUserRole

Go

Here we use the checkUserRole() shared method of our MyFirstClrTrigger class of the registered assembly UserAssembly.

Checking the triggering action

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

Finally, we need to check our Trigger. Let us test the trigger using the same INSERT Statement that we used before.

Listing 7 (Insert into UserTable)

----- Checking the DB objects.----------------

INSERT UserTable (User\_Name, Type) VALUES ('James', 'ADMIN')

Go

----- Output Message -----

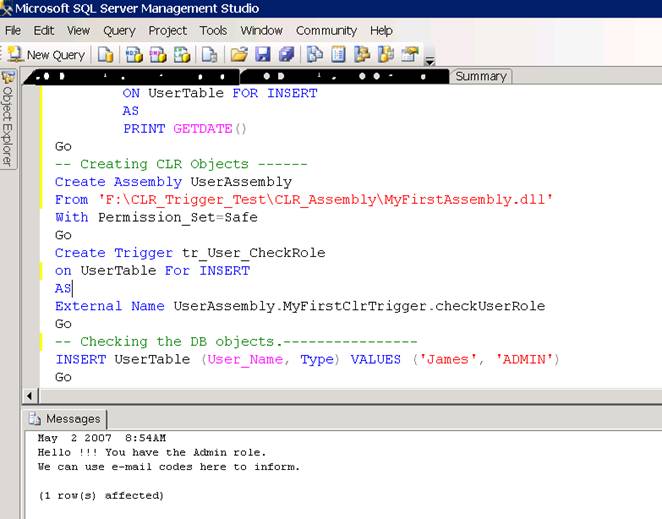
May 2 2007  8:54AM

Hello !!! You have the Admin role.

We can use e-mail codes here to inform.

The output is the result of two triggering actions; one is of T-SQL type and another one is of CLR type. Both are attached to the UserTable for the same type of action (i.e. after Insert statement).  The following image is the output result window of SQL Server Management studio.

## 



Conclusion

[ [Back To Top](http://aspalliance.com/1273_CLR_Triggers_for_SQL_Server_2005.all#top) ]

Now we have our CLR Trigger in our database with source code stored externally. Hence, externally stored code objects can be more secure than the previous one created by T-SQL. Also now we have the power and expressiveness of .NET language for our database objects. Above all, now we have a brief idea about all types of Triggers supported by SQL Server 2005.

By

Soyuj Kumar