**Triggers**

**A code is executed on events like update insert delete**

**You don’t have to execute by yourself again and again**

**For e.g**

**Table 1 has a trigger type for insertion**

**When every insertion will go on in this table the trigger will execute.**

**What inside the trigger**

**SQL**

**Code.**

**STORED PROCEDURE**

**Line of code execute not automatically you have to execute it by yourself**

**Mostly stored procedure calls inside the application layer where you are writing code for the application.**

**You have an idea of a function**

**That if you create a function, therefore function has to execute..**

**When?**

**Whenever we can call that function.**

**Advantages? of using stored procedures? Instead, write the stored procedure functionality over the application.**

**Request and Response calculation.**

**Transfer the responsibility to server.**

**Let if I wanted to insert something into the database table.**

**I can use stored procedure (Execution is on server side not on application side)**

**Let**

**Create the stored procedure name it (abc)**

**Passing the parameters**

**For the use of SP**

**You can connect the database**

**When the connection is successful**

**Then you can use sql command to call the SP**

**Abc(12546,’abcdef’,’H.R’)**

-- ================================================

-- Template generated from Template Explorer using:

-- Create Procedure (New Menu).SQL

--

-- Use the Specify Values for Template Parameters

-- command (Ctrl-Shift-M) to fill in the parameter

-- values below.

--

-- This block of comments will not be included in

-- the definition of the procedure.

-- ================================================

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE <Procedure\_Name, sysname, ProcedureName>

-- Add the parameters for the stored procedure here

<@Param1, sysname, @p1> <Datatype\_For\_Param1, , int> = <Default\_Value\_For\_Param1, , 0>,

<@Param2, sysname, @p2> <Datatype\_For\_Param2, , int> = <Default\_Value\_For\_Param2, , 0>

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>

END

GO

CREATE PROCEDURE PROC\_NAME

(@FIRST\_NAME string, @LAST\_NAME string, @AGE int)

AS

BEGIN

SET NOCOUNT ON;

//if (@empid > 3)

//SELECT \* from EMPLOYEE where EMP\_ID = @empid

Insert into Employee111() values (@FIRST\_NAME, @LAST\_NAME, @AGE)

//Else

PRINT 'Done'

END

EMPISERTION(5)

CREATE PROCEDURE EMPISERTION

(@empid INT)

AS

DECLARE

@tem String;

BEGIN

SET NOCOUNT ON;

if (@empid > 3)

begin

SELECT @tem = EMP\_Name from EMPLOYEE where EMP\_ID = @empid;

If (@tem = ‘’)

Begin

Insert into emp values()

End

PRINT @tem;

end

Else

begin

PRINT 'NO RECORD FOUND'

end

END

ALTER PROCEDURE

[dbo].[EMPINSERTION]

(@empid INT)

AS

BEGIN

SET NOCOUNT ON;

if (@empid > 3)

SELECT \* from EMPLOYEE where EMP\_ID = @empid

Else

PRINT 'NO RECORD FOUND'

END

1. **namespace** Storeprocedure
2. {
3. **public**  **class** Common
4. {
6. SqlConnection sqlCon=**null**;
7. String SqlconString=ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
8. **public** **void** Test(**string** firstName,**string** lastName,**int** age)
9. {
10. **using**(sqlCon=**new** SqlConnection(SqlconString))
11. {
12. sqlCon.Open();
13. SqlCommand sql\_cmnd = **new** SqlCommand("PROC\_NAME", sqlCon);
14. sql\_cmnd.CommandType = CommandType.StoredProcedur;

sql\_cmnd.Parameters.AddWithValue("@FIRST\_NAME", SqlDbType.NVarChar).Value=firstName;

sql\_cmnd.Parameters.AddWithValue("@LAST\_NAME", SqlDbType.NVarChar).Value=lastName;

sql\_cmnd.Parameters.AddWithValue("@AGE", SqlDbType.Int).Value = age;

1. sql\_cmnd.ExecuteNonQuery();
2. sqlCon.Close();
3. }
4. }
5. }
6. }