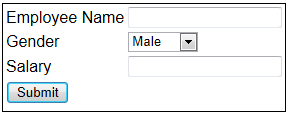
In [Part 6](http://csharp-video-tutorials.blogspot.com/2012/10/sql-injection-prevention-part-6.html), we have discussed that sql injection can be prevented by using parameterized queries or stored procedures. We have also seen how to call a stored procedure with input parameters. In this part, we will learn about calling a stored procedure with output parameters.  
  
Our example, will be based on **tblEmployees**. The script to create this table is shown below. The important point to note here is that, **EmployeeId**is marked as an identity column. When inserting a row into the table, we need not provide a value for the **EmployeeId** column. If you want to learn more about identity column, please watch - [Part 7 of SQL Server : Identity Column in SQL Server.](http://csharp-video-tutorials.blogspot.com/2012/08/identity-column-in-sql-server-part-7.html)  
Create Table tblEmployees  
(  
 EmployeeId int identity primary key,  
 Name nvarchar(50),  
 Gender nvarchar(10),  
 Salary int  
)   
  
   
  
**Script to insert sample data. Notice, that in the insert statement we are not providing a value for EmployeeId Column.**  
Insert into tblEmployees values('Mike','Male',5000)  
Insert into tblEmployees values('Pam','Female',3500)  
Insert into tblEmployees values('John','Male',2350)  
Insert into tblEmployees values('Sara','Female',5700)  
Insert into tblEmployees values('Steve','Male',4890)  
Insert into tblEmployees values('Sana','Female',4500)  
  
**1.** **spAddEmployee** stored procedure inserts a row into tblEmployees tables.   
**2.** @Name, @Gender and @Salary are **input** parameters.  
**3.** @EmployeeId is an **output** parameter  
**4.** The stored procedure has got only 2 lines of code with in the body. The first line inserts a row into the **tblEmployees**table. The second line, gets the **auto generated identity value** of the  **EmployeeId**column.  
**5.** This procedure, will later be called by a dot net application.  
Create Procedure spAddEmployee    
@Name nvarchar(50),    
@Gender nvarchar(20),    
@Salary int,    
@EmployeeId int Out    
as    
Begin    
 Insert into tblEmployees values(@Name, @Gender, @Salary)    
 Select @EmployeeId = SCOPE\_IDENTITY()    
End   
  
   
  
**At this point, we have done everything that is required for our demo, from a database perspective. Now let's flip to visual studio. Create an asp.net web application. Copy and Paste the following HTML onto a webform.**  
<table style="border: 1px solid black; font-family:Arial">  
    <tr>  
        <td>  
            Employee Name  
        </td>  
        <td>  
            <asp:TextBox ID="txtEmployeeName" runat="server"></asp:TextBox>  
        </td>  
    </tr>          
    <tr>  
        <td>  
            Gender  
        </td>  
        <td>  
            <asp:DropDownList ID="ddlGender" runat="server">  
                <asp:ListItem>Male</asp:ListItem>  
                <asp:ListItem>Female</asp:ListItem>  
            </asp:DropDownList>  
        </td>  
    </tr>   
    <tr>  
        <td>  
            Salary  
        </td>  
        <td>  
            <asp:TextBox ID="txtSalary" runat="server"></asp:TextBox>  
        </td>  
    </tr>         
    <tr>  
        <td colspan="2">  
            <asp:Button ID="btnSubmit" runat="server" Text="Submit"   
                onclick="btnSubmit\_Click" />  
        </td>  
    </tr>            
    <tr>  
        <td colspan="2">  
            <asp:Label ID="lblMessage" runat="server"></asp:Label>  
        </td>  
    </tr>    
</table>  
  
**The design of the webform, should be as shown below.**  
   
  
**Copy and paste the following code in the code behind page.**  
protected void btnSubmit\_Click(object sender, EventArgs e)  
{  
    //Read the connection string from Web.Config file  
    string ConnectionString = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
    using (SqlConnection con = new SqlConnection(ConnectionString))  
    {  
        //Create the SqlCommand object  
        SqlCommand cmd = new SqlCommand("spAddEmployee", con);  
        //Specify that the SqlCommand is a stored procedure  
        cmd.CommandType = System.Data.CommandType.StoredProcedure;  
  
        //Add the input parameters to the command object  
        cmd.Parameters.AddWithValue("@Name", txtEmployeeName.Text);  
        cmd.Parameters.AddWithValue("@Gender", ddlGender.SelectedValue);  
        cmd.Parameters.AddWithValue("@Salary", txtSalary.Text);  
  
        //Add the output parameter to the command object  
        SqlParameter outPutParameter = new SqlParameter();  
        outPutParameter.ParameterName = "@EmployeeId";  
        outPutParameter.SqlDbType = System.Data.SqlDbType.Int;  
        outPutParameter.Direction = System.Data.ParameterDirection.Output;  
        cmd.Parameters.Add(outPutParameter);  
  
        //Open the connection and execute the query  
        con.Open();  
        cmd.ExecuteNonQuery();  
                  
        //Retrieve the value of the output parameter  
        string EmployeeId = outPutParameter.Value.ToString();  
        lblMessage.Text = "Employee Id = " + EmployeeId;  
    }  
}  
  
**Note:** Please make sure to add the following **using**declarations at the top of the code behind page.  
using System.Data.SqlClient;  
using System.Configuration;  
  
**Now, run the application. Fill in the employee details and click Submit. The Employee row gets added to the database, and the generated EmployeeId is shown on the screen.**