Entity Framework CRUD Operations Using Stored Procedures

## Introduction

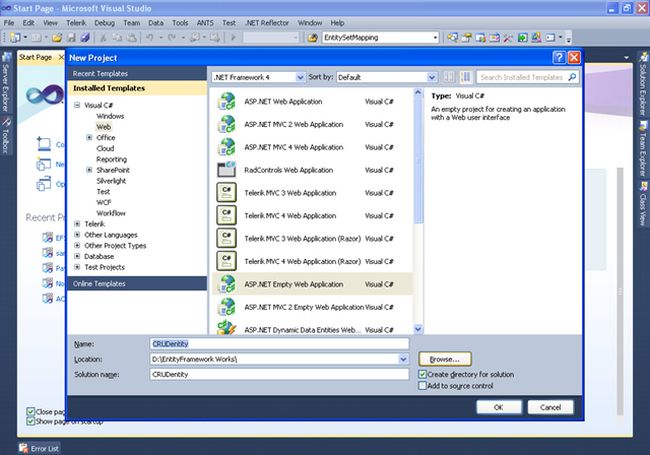
In this article I would like to share something regarding Entity Framework, how we can implement CRUD operations using Stored Procedures in Entity Framework.

In this explanation there are two ways of implementing CRUD operations:

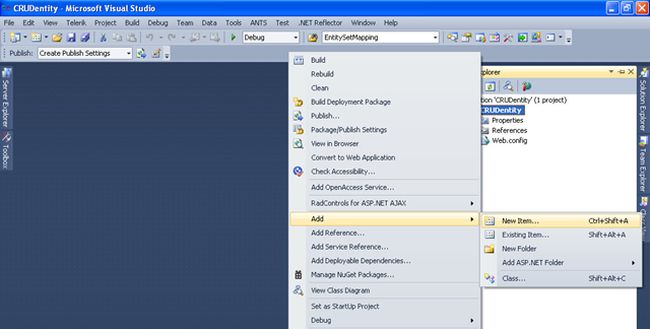
1. By calling Stored Procedures using ExecuteStoreCommand and ExecuteStoreQuery, without mapping to the Model.
2. By mapping Stored Procedures to the Model.

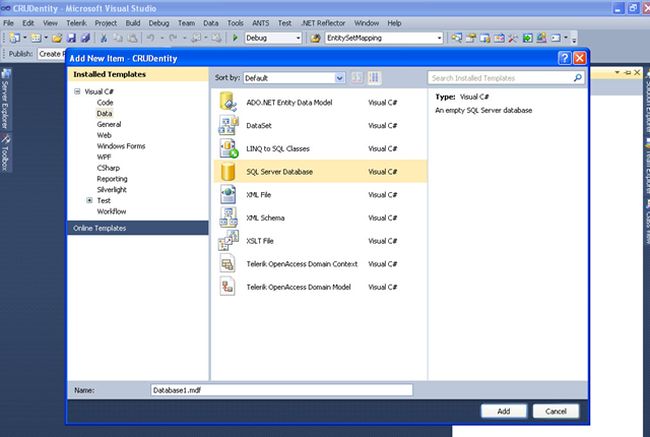
OK, first we will see how we can map Stored Procedures to the Model to implement CRUD operations.

Create an empty web application from VS2010, see:

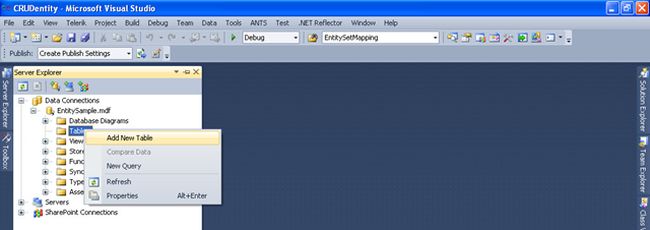


First let us create a sample table. Since I do not have SQL Server installed in my machine, I am adding a SQL Server database as in the following:



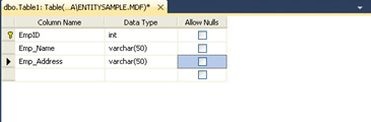


Here you can use the name Database1.mdf depending on your naming convention. Now in Server Explorer, you will see your database, we will add a table and some Stored Procedures as follows:



Add the required columns and save the table with the desired name. The most important aspect before you start working on Entity Framework is to have a Primary Key in your table.

Now my table looks as in the following, on which we are going to perform CRUD operations.



OK now let's create Stored Procedures for Insert, Update, Delete, and Select operations.

#### Insert Stored Procedure

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/522611/Entity-Framework-CRUD-Operations-Using-Stored-Proc)

CreatePROCEDURE dbo.InsertEmployee

(

@ID int,

@EmpName varchar(50),

@EmpAddress varchar(50)

)

AS

Begin

insert into Employee(EmpID,Emp\_Name,Emp\_Address)values(@ID,@EmpName,@EmpAddress)

END

#### Delete Stored Procedure

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/522611/Entity-Framework-CRUD-Operations-Using-Stored-Proc)

Create PROCEDURE dbo.deleteEmp

(

@ID int

)

As

Begin

delete from Employee where EmpID=@ID

End

#### Select

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/522611/Entity-Framework-CRUD-Operations-Using-Stored-Proc)

Create PROCEDURE dbo.SelectEmployee

As

Begin

select \* from Employee

End

#### Update

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/522611/Entity-Framework-CRUD-Operations-Using-Stored-Proc)

Create PROCEDURE dbo.UpdateEmployee

(@ID int,

@EmpName varchar(50),

@EmpAddress varchar(50))

As

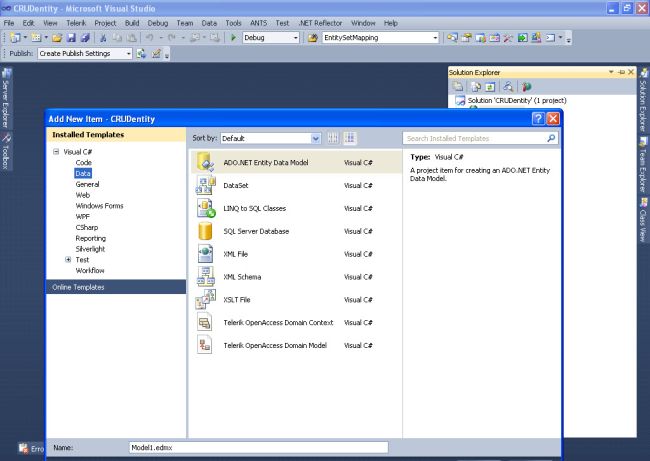
Begin

update Employee set Emp\_Name=@EmpName,Emp\_Address=@EmpAddress where EmpID=@ID

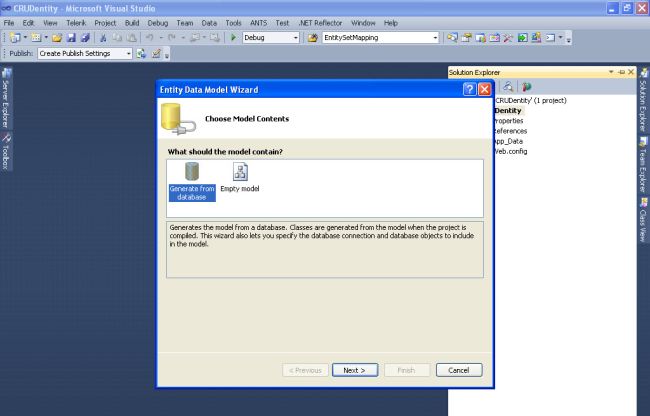
End

We are finished with our database. Now let us create a sample page and add an Entity Model to our application.

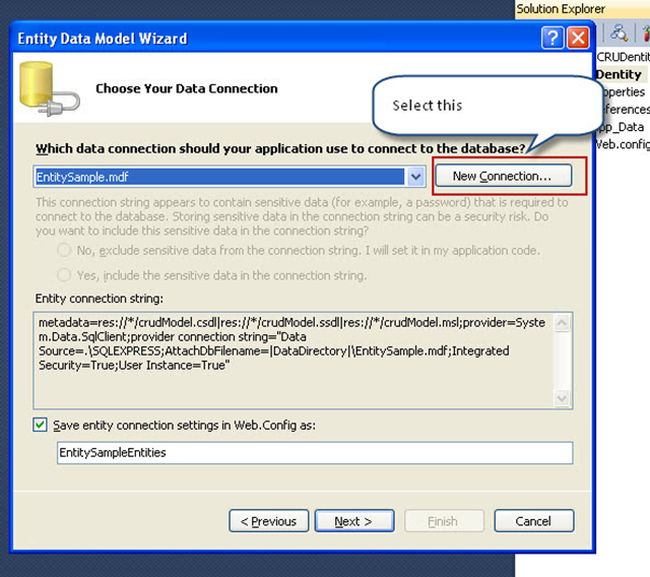
Adding an Entity Model to your application:

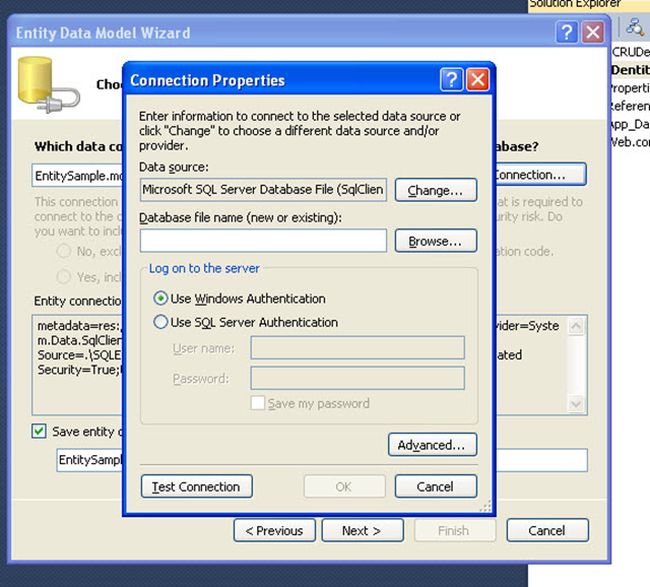


After adding a Model you will immediately have this Entity Data Model Wizard where you have to select Generate from the database and click on Next:



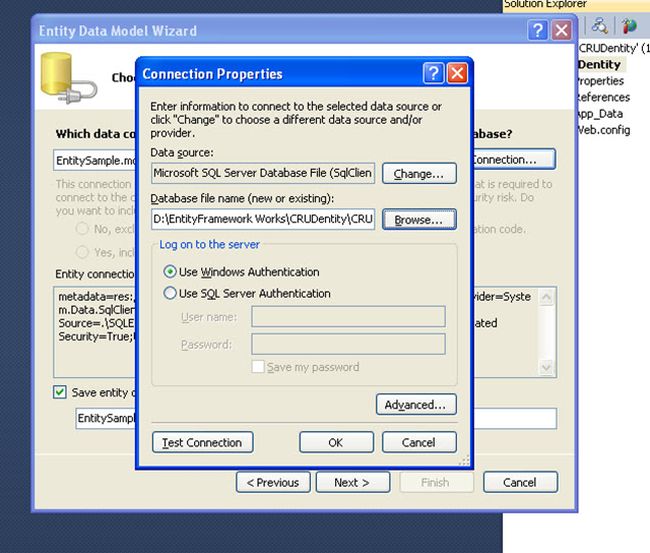
Select New Connection from Choose your data:

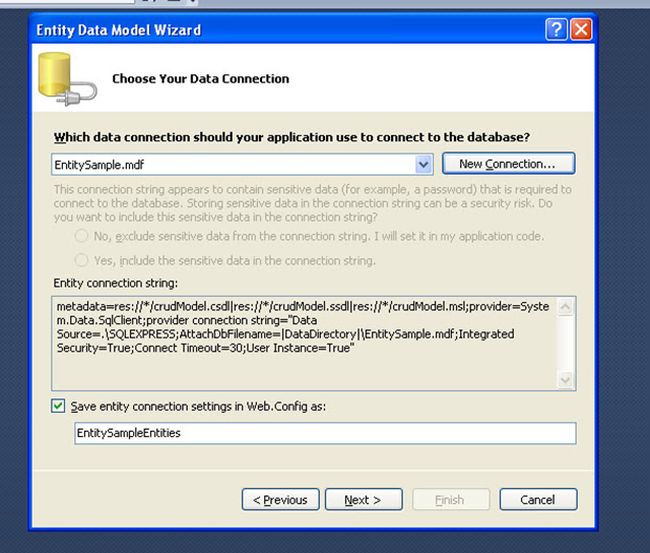




Here on the Data Source you will have various sources which you will see by clicking on Change, as I have created my database in my application I will use Microsoft SQL Server Database File (SqlClient), if anyone is using SQL Server you can change that to SQL Server from the options available.

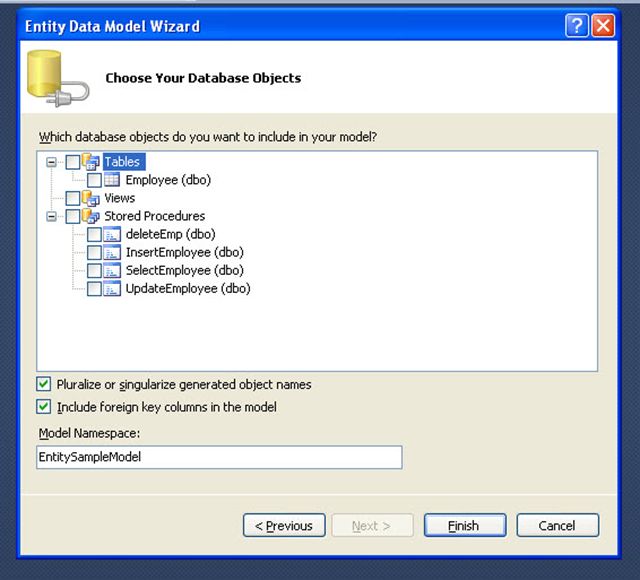
Since I am using a Microsoft SQL Server Database File (SqlClient) I will browse for my Database file and click on "OK".

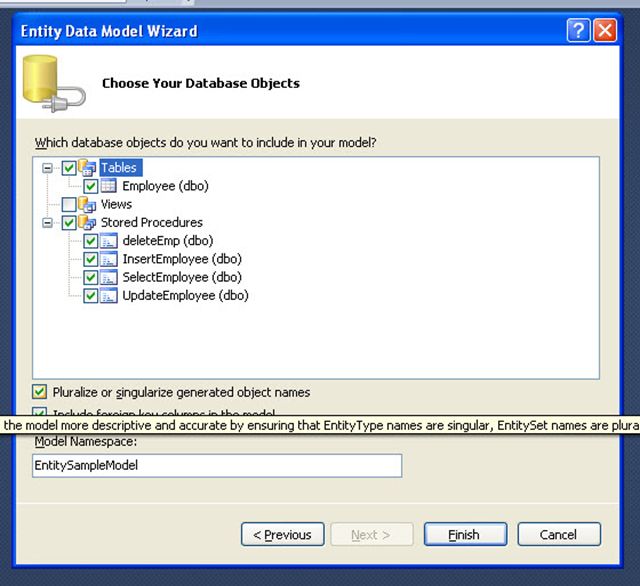




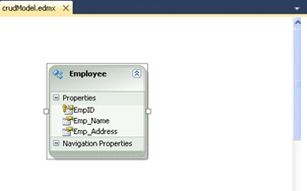
Here you will see my Database file and also the connection settings in Web.Config will be saved with the name EntitySampleEntities. Click Next where you will find all your tables and Stored Procedures that you have created. Select the required one. Since I created only one table and 4 Stored Procedures I will select them.

#### Initial Window





Click on "Finish" after you are finished, then you will see your model with the tables you added and if there are any relations it will also map them. As of now I created just one table that will be shown as follows:



Now we are finished with creating the database and adding it to an Entity Model. Now we will see how to perform CRUD operations without mapping the Stored Procedures to the model.

I also included some LINQ queries wherever needed, for example to auto-generate Employee ID and binding the drop-down list.

Create a web page and add the following design to that page:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/522611/Entity-Framework-CRUD-Operations-Using-Stored-Proc)

<%@ Page Language="C#" AutoEventWireup="true"

CodeBehind="crud.aspx.cs" Inherits="CRUDentity.crud" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<center>

<h3>

Display data in gridview using Entity Framework with out Mapping Stored Procedure

to Model

</h3>

<div style="width: 800px; margin: 0 auto; float: left;">

<asp:GridView ID="grdEmployess" runat="server" BackColor="White" BorderColor="#999999"

DataKeyNames="EmpID" BorderStyle="None" BorderWidth="1px" CellPadding="3"GridLines="Vertical">

<AlternatingRowStyle BackColor="#DCDCDC" />

<EmptyDataTemplate>

No record to show

</EmptyDataTemplate>

<FooterStyle BackColor="#CCCCCC" ForeColor="Black" />

<HeaderStyle BackColor="#000084" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#999999" ForeColor="Black" HorizontalAlign="Center" />

<RowStyle BackColor="#EEEEEE" ForeColor="Black" />

<SelectedRowStyle BackColor="#008A8C" Font-Bold="True" ForeColor="White" />

<SortedAscendingCellStyle BackColor="#F1F1F1" />

<SortedAscendingHeaderStyle BackColor="#0000A9" />

<SortedDescendingCellStyle BackColor="#CAC9C9" />

<SortedDescendingHeaderStyle BackColor="#000065" />

</asp:GridView>

</div>

<br />

<div style="width: 800px; margin: 0 auto; float: left;">

<h3>

Insert Data to table using Entity Framework with out Mapping Stored Procedures to

Model</h3>

<table>

<tr>

<td>

Employee ID :

</td>

<td>

<asp:TextBox ID="txtEmpID" ReadOnly="true" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td>

Employee Name :

</td>

<td>

<asp:TextBox ID="txtEmployeeName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rqrdEmployeeName" runat="server" ErrorMessage="\*"

ControlToValidate="txtEmployeeName" ToolTip="Employee name required"ValidationGroup="g"

Display="Dynamic" ForeColor="Red"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td>

Employee Address :

</td>

<td>

<asp:TextBox ID="txtAddress" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rqrdAddress" runat="server"

ErrorMessage="\*"ControlToValidate="txtAddress"

ToolTip="Address required" ValidationGroup="g"

Display="Dynamic" ForeColor="Red"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td>

</td>

</tr>

<tr align="center">

<td colspan="3">

<asp:Button ID="btnInsert" runat="server" Text="Insert"

ValidationGroup="g"OnClick="btnInsert\_Click" />

</td>

</tr>

</table>

</div>

<br />

<div style="width: 800px; margin: 0 auto; float: left;">

<h3>

Edit and Update data using storedprocedure With out mapping it to Model</h3>

<table>

<tr>

<td>

Select Employee ID :

</td>

<td>

<asp:DropDownList ID="ddleditEmpID" runat="server"

AutoPostBack="true"

OnSelectedIndexChanged="ddleditEmpID\_SelectedIndexChanged">

</asp:DropDownList>

</td>

</tr>

<tr>

<td>

Employee Name :

</td>

<td>

<asp:TextBox ID="txtedtEmployeeName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rqrdedtEmpName" runat="server"

ErrorMessage="\*"ControlToValidate="txtedtEmployeeName"

ToolTip="Employee name required" ValidationGroup="g1"

Display="Dynamic"ForeColor="Red"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td>

Employee Address :

</td>

<td>

<asp:TextBox ID="txtedtEmpAddress" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rqrdedtEmpAddress" runat="server" ErrorMessage="\*"

ControlToValidate="txtedtEmpAddress" ToolTip="Address required" ValidationGroup="g1"

Display="Dynamic" ForeColor="Red"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td>

</td>

</tr>

<tr align="center">

<td colspan="4">

<asp:Button ID="btnUpdate" runat="server" Text="Update" OnClick="btnUpdate\_Click" />

</td>

</tr>

</table>

</div>

<br />

<div style="width: 800px; margin: 0 auto; float: left;">

<h3>

Delete data using storedprocedure With out mapping it to Model</h3>

<table>

<tr>

<td>

Select Employee ID to Delete :

</td>

<td>

<asp:DropDownList ID="ddlEmpID" runat="server">

</asp:DropDownList>

</td>

</tr>

<tr>

<td>

</td>

</tr>

<tr align="center">

<td colspan="2">

<asp:Button ID="btnDelete" runat="server"

ValidationGroup="g1" Text="Delete"OnClick="btnDelete\_Click" />

</td>

</tr>

</table>

</div>

</center>

</div>

</form>

</body>

</html>

***Yourpage.aspx.cs***

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/522611/Entity-Framework-CRUD-Operations-Using-Stored-Proc)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

namespace CRUDentity

{

public partial class crud : System.Web.UI.Page

{

EntitySampleEntities entities = new EntitySampleEntities();

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

checkMax();

loadGrid();

bindDDL();

}

}

protected void btnInsert\_Click(object sender, EventArgs e)

{

Page.Validate("g");

if (Page.IsValid)

{

var ietsParameterID = new SqlParameter("@ID", System.Data.SqlDbType.Int);

ietsParameterID.Value = Convert.ToInt16(txtEmpID.Text);

var ietsParameterEmpName = new SqlParameter("@EmpName", txtEmployeeName.Text);

var ietsParameterEmpAddress = new SqlParameter("@EmpAddress", txtAddress.Text);

entities.ExecuteStoreCommand("InsertEmployee @ID,@EmpName,@EmpAddress",

ietsParameterID, ietsParameterEmpName, ietsParameterEmpAddress);

loadGrid();

checkMax();

bindDDL();

txtAddress.Text = string.Empty;

txtEmployeeName.Text = string.Empty;

}

}

public void checkMax()

{

int? maxEmpID = entities.Employees.Max(q => (int?)q.EmpID);

if (maxEmpID != null)

{

maxEmpID = maxEmpID + 1;

txtEmpID.Text = maxEmpID.ToString();

}

else

{

maxEmpID = 1000;

txtEmpID.Text = maxEmpID.ToString();

}

}

public void loadGrid()

{

var selectData = entities.ExecuteStoreQuery<Employee>("SelectEmployee").ToList();

grdEmployess.DataSource = selectData;

grdEmployess.DataBind();

}

public void bindDDL()

{

var display = from e in entities.Employees select new { e.EmpID };

ddlEmpID.DataSource = display.ToList();

ddlEmpID.DataTextField = "EmpID";

ddlEmpID.DataValueField = "EmpID";

ddlEmpID.DataBind();

ddlEmpID.Items.Insert(0, "--Select--");

ddleditEmpID.DataSource = display.ToList();

ddleditEmpID.DataTextField = "EmpID";

ddleditEmpID.DataValueField = "EmpID";

ddleditEmpID.DataBind();

ddleditEmpID.Items.Insert(0, "--Select--");

}

protected void btnDelete\_Click(object sender, EventArgs e)

{

if (ddlEmpID.SelectedItem.Text != "--Select--")

{

var ietsParameterID = new SqlParameter("@ID", ddlEmpID.SelectedItem.Text);

entities.ExecuteStoreCommand("deleteEmp @ID", ietsParameterID);

loadGrid();

checkMax();

bindDDL();

}

}

protected void btnUpdate\_Click(object sender, EventArgs e)

{

Page.Validate("g1");

if (Page.IsValid)

{

if (ddleditEmpID.SelectedItem.Text != "--Select--")

{

var ietsParameterID = new SqlParameter("@ID", System.Data.SqlDbType.Int);

ietsParameterID.Value = Convert.ToInt16(ddleditEmpID.SelectedItem.Text);

var ietsParameterEmpName = new SqlParameter("@EmpName", txtedtEmployeeName.Text);

var ietsParameterEmpAddress = new SqlParameter("@EmpAddress", txtedtEmpAddress.Text);

entities.ExecuteStoreCommand("UpdateEmployee @ID,@EmpName,@EmpAddress",

ietsParameterID, ietsParameterEmpName, ietsParameterEmpAddress);

loadGrid();

}

}

}

protected void ddleditEmpID\_SelectedIndexChanged(object sender, EventArgs e)

{

if (ddleditEmpID.SelectedItem.Text != "--Select--")

{

int id = Convert.ToInt16(ddleditEmpID.SelectedValue.ToString());

var display = from e1 in entities.Employees

where e1.EmpID.Equals(id)

select new { e1.Emp\_Name, e1.Emp\_Address };

foreach (var v in display)

{

txtedtEmployeeName.Text = v.Emp\_Name;

txtedtEmpAddress.Text = v.Emp\_Address;

}

}

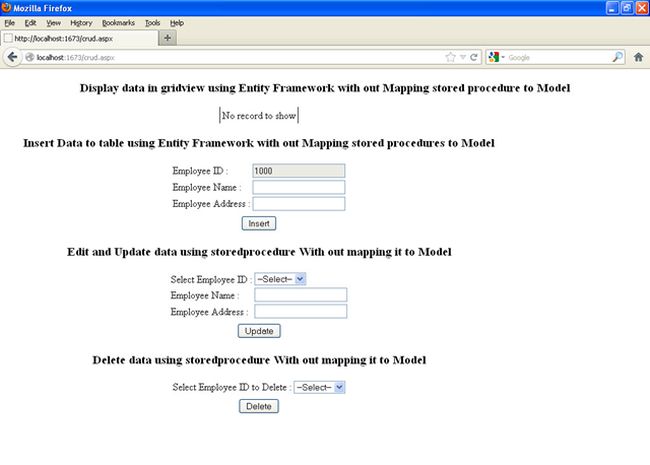
}

}

}

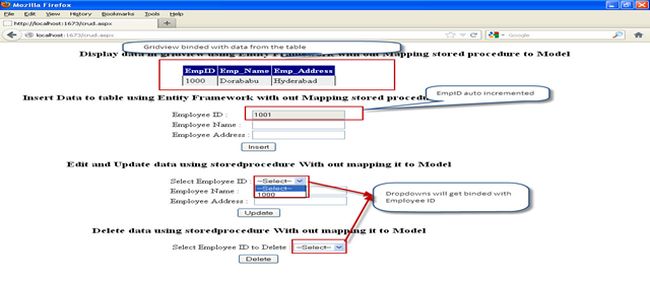
***Sample screenshots***

When you first run the application:



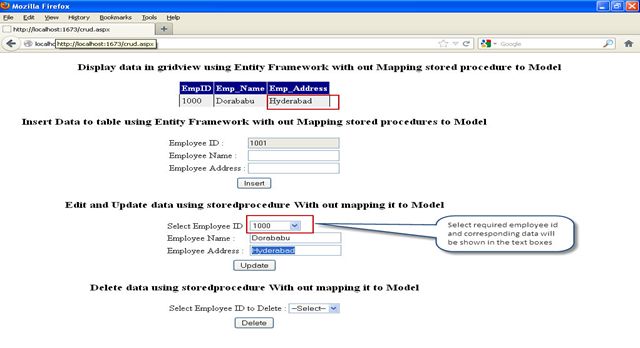
Since there are no records in the table you will see the grid view is empty. Also you will see the Employee ID is read only, to avoid duplicates I make this one if you want you can remove that and do whatever you need to.

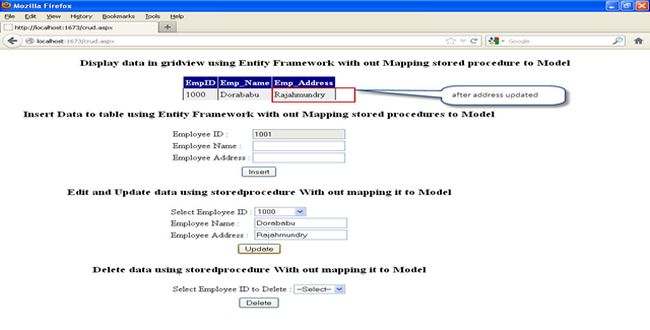
Now we will see what happens after submitting data:



Now we will edit the record see here I will change the Employee Address initially it is Hyderabad I will change it to some other. To do that select the Employee ID that you need to edit and update. Since here I have only one Employee I will do for that.

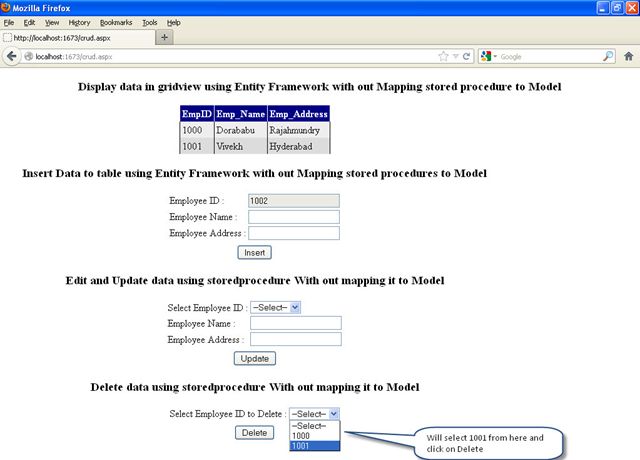
#### Before editing Employee Address





Let's do the deleting; for this I will add another employee to the table as shown and then will delete it.

#### Before delete



#### After Delete



That's it, this is how we can do basic CRUD operations using Entity Framework without mapping Stored Procedures to the Model.

Wait for the next one on how we can implement CRUD operations using Entity Framework by mapping Stored Procedures to the Model.