Master Detail CRUD Operations using EF and ASP.net MVC 3

Introduction

In this post I discuss about how we can perform Master-Detail CRUD operation using Entity Framework (Code First) and ASP.net MVC 3. Here I have used JSON (json2.js) for data passing, Ajax for posting and DataTables (datatables.js) for manipulating detail Record.

Creating Master Detail CRUD Application

Create Sample Solution

>> Open VS 2010

>> Create **ASP.net MVC 3** Project named "MasterDetail"

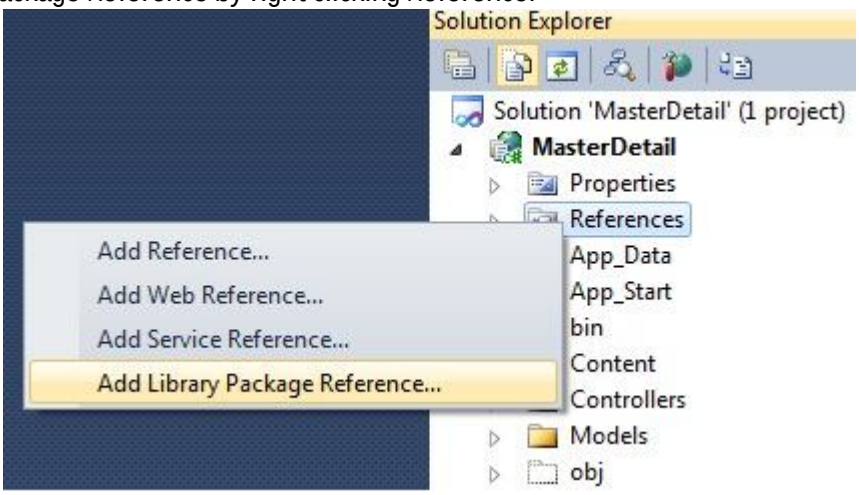
Here I have used

JSON for passing data view to controller

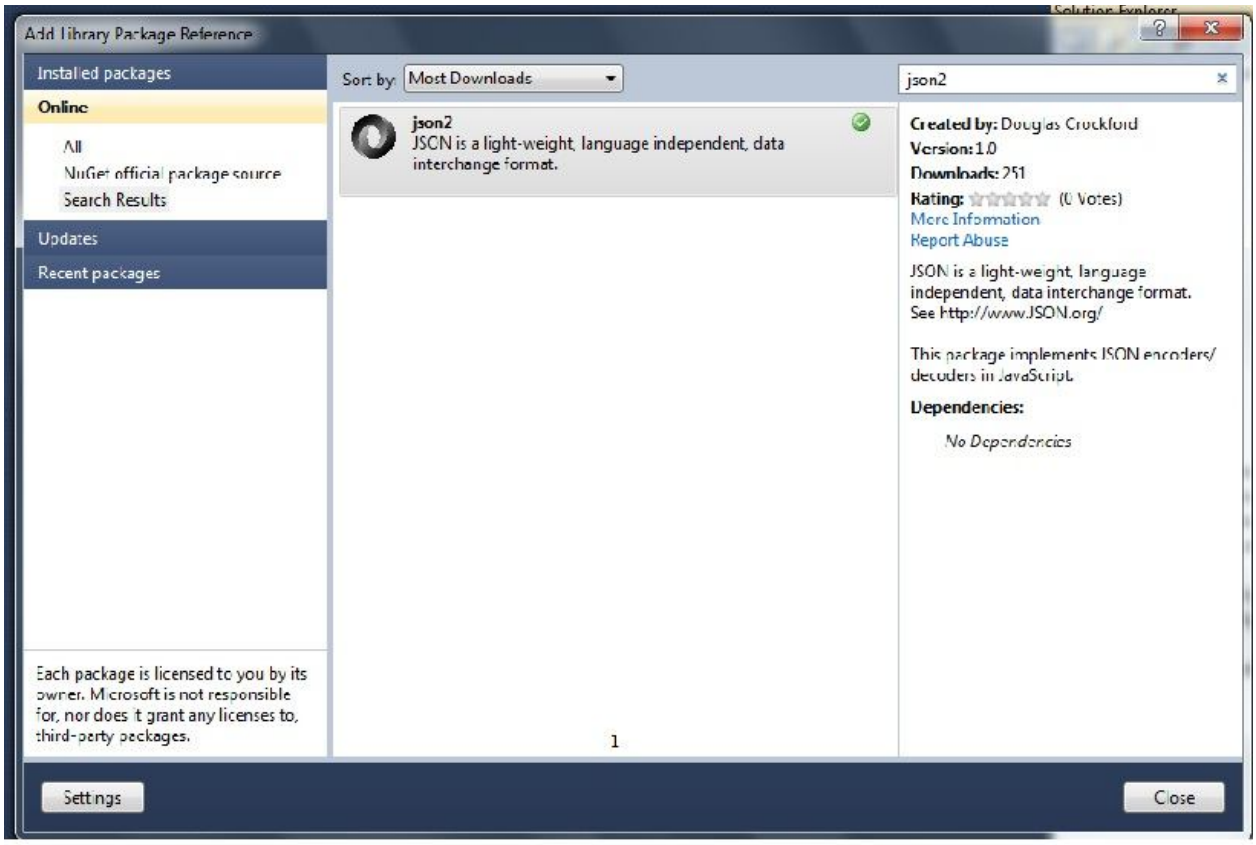
Data Tables for manipulating details record.

>> let add JSON and DataTables js file on our project using following way.

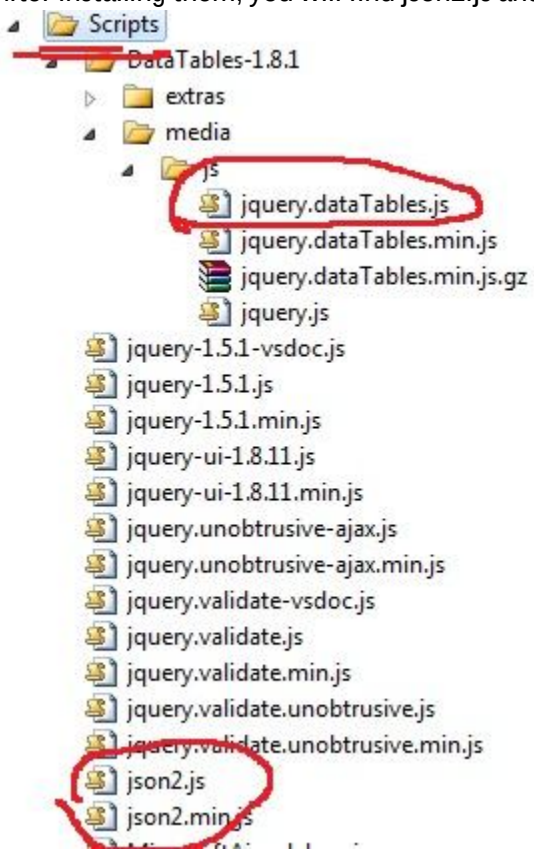
- Select Add Library Package Reference by right clicking Reference.



- Add Library Package Manager Window will appear; from the window search json2 & DataTables and install them.



- After installing them, you will find json2.js and datatables.js on script folder.



Now our solution is ready for further work.

Creating Model

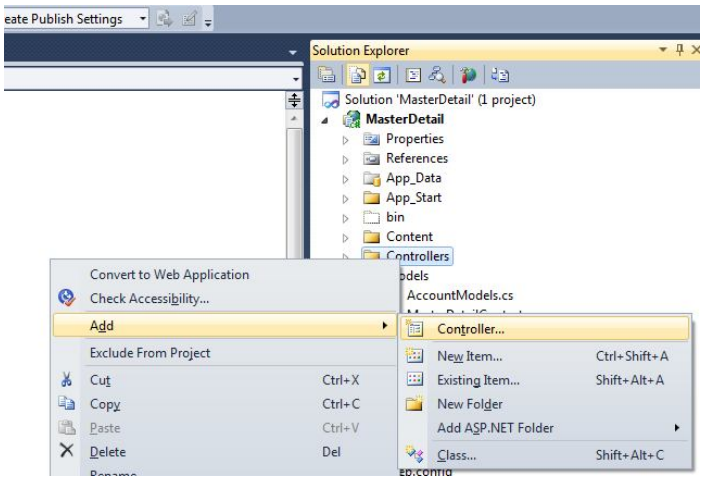
Here we have considered we have two entities SalesMain and SalesSub(One to many relation). One salesMain has multiple sales sub records.

```
public class SalesMain
    {

        [Key]
        public int SalesId { get; set; }
        public string ReferenceNo { get; set; }
        public DateTime SalesDate { get; set; }
        public string SalesPerson { get; set; }

        public virtual ICollection<SalesSub> SalesSubs { get; set; }
    }


public class SalesSub
    {

        [Key, Column(Order = 0)]
        public int SalesId { get; set; }

        [Key, Column(Order = 1)]
        public string ItemName { get; set; }

        public int Qty { get; set; }
        public decimal UnitPrice { get; set; }


        public virtual SalesMain SalesMain { get; set; }
    }
```

>> Now build your project/ Press f5.

Creating Controller, Context and Views

>> Right Click on controller Folder and Select Add >> Controller

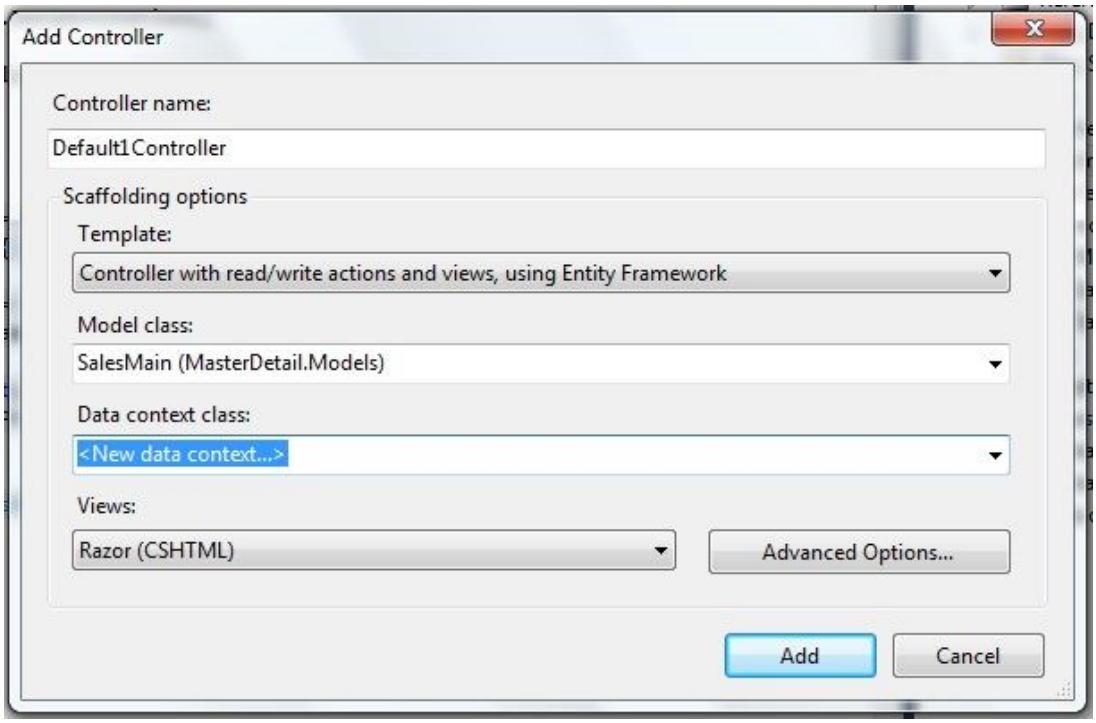


>> Name it SalesController

>> Select “SalesMain (MasterDetail.Models)” as a Model Class

>> Select <new data Context> and give its name “MasterDetail.Models.MasterDetailContext”

>> Then automatically it will create Views, Controller and Context Class.



Now we have to modify Our Sales Controller Class and Views.

Modify Sales Controller

Modify existing Create Method by following:

```
[HttpPost]
        public JsonResult Create(SalesMain salesmain)
        {
            try
            {
                if (ModelState.IsValid)
                {

                    // If sales main has SalesID then we can understand we have existing sales
Information
                    // So we need to Perform Update Operation

                    // Perform Update
                    if (salesmain.SalesId > 0)
                    {

                        var CurrentsalesSUb = db.SalesSubs.Where(p => p.SalesId == salesmain.SalesId);

                        foreach (SalesSub ss in CurrentsalesSUb)
                            db.SalesSubs.Remove(ss);

                        foreach (SalesSub ss in salesmain.SalesSubs)
                            db.SalesSubs.Add(ss);

                        db.Entry(salesmain).State = EntityState.Modified;
```

```
                    }
                    //Perform Save
                    else
                    {
                        db.SalesMains.Add(salesmain);
                    }

                    db.SaveChanges();

                    // If Sucess== 1 then Save/Update Successfull else there it has Exception
                    return Json(new { Success = 1, SalesID = salesmain.SalesId, ex="" });
                }
            }
            catch (Exception ex)
            {
                // If Sucess== 0 then Unable to perform Save/Update Operation and send Exception to
View as JSON
                return Json(new { Success = 0, ex = ex.Message.ToString() });
            }

            return Json(new { Success = 0, ex = new Exception("Unable to save").Message.ToString() });
        }
```

Modify Edit Method in following way

```
public ActionResult Edit(int id)
        {
            ViewBag.Title = "Edit";
            SalesMain salesmain = db.SalesMains.Find(id);

            //Call Create View
            return View("Create", salesmain);
        }
```

Delete “Edit method” with Http post because we will use Create method for performing Save and Update operation.

Finally the **sales controller** looks like following.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MasterDetail.Models;
using System.Web.Helpers;
using System.Data.Objects;

namespace MasterDetail.Controllers
{
    public class SalesController : Controller
    {
        private MasterDetailContext db = new MasterDetailContext();

        //
        // GET: /Sales/
        public ViewResult Index()
        {
```

```
            return View(db.SalesMains.ToList());
        }

        //
        // GET: /Sales/Details/5

        public ViewResult Details(int id)
        {
            SalesMain salesmain = db.SalesMains.Find(id);
            return View(salesmain);
        }

        //
        // GET: /Sales/Create

        public ActionResult Create()
        {
            ViewBag.Title = "Create";
            return View();
        }



        // POST: /Sales/Create
        /// <summary>
        /// This method is used for Creating and Updating  Sales Information
        /// (Sales Contains: 1.SalesMain and *SalesSub )
        /// </summary>
        /// <param name="salesmain">
        /// </param>
        /// <returns>
        /// Returns Json data Containing Success Status, New Sales ID and Exeception
        /// </returns>
        [HttpPost]
        public JsonResult Create(SalesMain salesmain)
        {
            try
            {
                if (ModelState.IsValid)
                {

                    // If sales main has SalesID then we can understand we have existing sales
Information
                    // So we need to Perform Update Operation

                    // Perform Update
                    if (salesmain.SalesId > 0)
                    {

                        var CurrentsalesSUb = db.SalesSubs.Where(p => p.SalesId == salesmain.SalesId);

                        foreach (SalesSub ss in CurrentsalesSUb)
                            db.SalesSubs.Remove(ss);

                        foreach (SalesSub ss in salesmain.SalesSubs)
                            db.SalesSubs.Add(ss);

                        db.Entry(salesmain).State = EntityState.Modified;
                    }
                    //Perform Save
                    else
                    {
                        db.SalesMains.Add(salesmain);
```

```
                    }

                    db.SaveChanges();

                    // If Sucess== 1 then Save/Update Successfull else there it has Exception
                    return Json(new { Success = 1, SalesID = salesmain.SalesId, ex="" });
                }
            }
            catch (Exception ex)
            {
                // If Sucess== 0 then Unable to perform Save/Update Operation and send Exception to
View as JSON
                return Json(new { Success = 0, ex = ex.Message.ToString() });
            }

            return Json(new { Success = 0, ex = new Exception("Unable to save").Message.ToString() });
        }

        //
        // GET: /Sales/Edit/5
        public ActionResult Edit(int id)
        {
            ViewBag.Title = "Edit";
            SalesMain salesmain = db.SalesMains.Find(id);

            //Call Create View
            return View("Create", salesmain);
        }




        // GET: /Sales/Delete/5
        public ActionResult Delete(int id)
        {
            SalesMain salesmain = db.SalesMains.Find(id);
            return View(salesmain);
        }



        // POST: /Sales/Delete/5
        [HttpPost, ActionName("Delete")]
        public ActionResult DeleteConfirmed(int id)
        {
            SalesMain salesmain = db.SalesMains.Find(id);
            db.SalesMains.Remove(salesmain);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

Modifying Create View

Add following *.js and *.css file.

```
@*This is for jquery*@
<script src="../../Scripts/jquery-1.5.1.js" type="text/javascript"></script>
@*This is for jquery UI, for Calender control*@
<script src="../../Scripts/jquery-ui-1.8.11.js" type="text/javascript"></script>

@*This is for JSON*@
<script src="../../Scripts/json2.js" type="text/javascript"></script>

@*These are for DataTables*@
<script src="../../Scripts/DataTables-1.8.1/media/js/jquery.dataTables.js"
type="text/javascript"></script>
<script src="../../Scripts/DataTables-1.8.1/extras/TableTools/media/js/TableTools.js"
type="text/javascript"></script>
<script src="../../Scripts/DataTables-1.8.1/extras/TableTools/media/js/ZeroClipboard.js"
type="text/javascript"></script>

@*These are for styling Control*@
<link href="../../Content/DataTables-1.8.1/extras/TableTools/media/css/TableTools.css"
rel="stylesheet" type="text/css" />
<link href="../../Content/DataTables-1.8.1/extras/TableTools/media/css/TableTools_JUI.css"
rel="stylesheet" type="text/css" />
<link href="../../Content/themes/base/jquery.ui.all.css" rel="stylesheet" type="text/css" />
```

Add html table for manipulating list of data.

After adding html table, we have converted it to DataTable so that we can easily add/delete item, read item more easily.

```
<table class="tbl" id="tbl">
       <thead>
       <tr>
       <th>ItemName</th> <th>Quantity</th><th>Unit Price</th>
       </tr>
       </thead>

       <tbody>
       @if (Model != null)
       {
           foreach (var item in Model.SalesSubs)
           {
           <tr>
           <td>
           @Html.DisplayTextFor(i => item.ItemName)
           </td>
           <td>
           @Html.DisplayTextFor(i => item.Qty)
           </td>
           <td>
           @Html.DisplayTextFor(i => item.UnitPrice)
           </td>

           </tr>
           }
       }

       </tbody>

       </table>
```

This is simple html table then we used following jQuery for convert it to datatable.

```
$(document).ready(function () {

        // here i have used datatables.js (jQuery Data Table)
        $('.tbl').dataTable({
            "sDom": 'T<"clear">lfrtip',
            "oTableTools": {
                "aButtons": [],
                "sRowSelect": "single"
            },
            "bLengthChange": false,
            "bFilter": false,
            "bSort": false,
            "bInfo": false
        });


        var oTable = $('.tbl').dataTable();
    });
```

Adding new row to Table

Following code shows how to read from text boxes and then add it to datatable.

```
function Add() {
        // Adding item to table
        $('.tbl').dataTable().fnAddData([$('#ItemName').val(), $('#Qty').val(),
$('#UnitPrice').val()]);

        // Making Editable text empty
        $('#ItemName').val("")
        $('#Qty').val("")
        $('#UnitPrice').val("")


     }
```

Delete selected row from Table

Following code shows how to remove selected item from datatable.

```
// This function is used fro
    // delete selected row from Detail Table
    // set deleted item to Edit text Boxes
    function DeleteRow() {

        // Here I have used DataTables.TableTools plugin for getting selected row items
        var oTT = TableTools.fnGetInstance('tbl'); // Get Table instance
        var sRow = oTT.fnGetSelected(); // Get Selected Item From Table


        // Set deleted row item to editable text boxes
        $('#ItemName').val($.trim(sRow[0].cells[0].innerHTML.toString()));
        $('#Qty').val(jQuery.trim(sRow[0].cells[1].innerHTML.toString()));
        $('#UnitPrice').val($.trim(sRow[0].cells[2].innerHTML.toString()));

        $('.tbl').dataTable().fnDeleteRow(sRow[0]);

    }
```

Save/Posting Data to sales Controller

Here we have two steps

1. Read view data and create JSON object
2. Ajax post

```
function Sales_save() {
     // Step 1: Read View Data and Create JSON Object

        // Creating SalesSub Json Object
        var salessub = {"SalesId":"", "ItemName":"","Qty":"","UnitPrice":""};


        // Creating SalesMain Json Object
        var salesmain = { "SalesId":"","ReferenceNo": "", "SalesDate": "", "SalesPerson": "",
"SalesSubs":[] };

        // Set Sales Main Value
        salesmain.SalesId = $("#SalesId").val();
        salesmain.ReferenceNo = $("#ReferenceNo").val();
        salesmain.SalesDate = $("#SalesDate").val();
        salesmain.SalesPerson = $("#SalesPerson").val();


        // Getting Table Data from where we will fetch Sales Sub Record
        var oTable = $('.tbl').dataTable().fnGetData();



        for (var i = 0; i < oTable.length; i++)
        {

            // IF This view is for edit then it will read SalesId from Hidden field
            if ($('h2').text() == "Edit")
            {
                salessub.SalesId = $('#SalesId').val();
            }
            else
            {
                salessub.SalesId = 0;
            }

            // Set SalesSub individual Value
            salessub.ItemName = oTable[i][0];
            salessub.Qty = oTable[i][1];
            salessub.UnitPrice = oTable[i][2];
            // adding to SalesMain.SalesSub List Item
            salesmain.SalesSubs.push(salessub);
            salessub = { "ItemName": "", "Qty": "", "UnitPrice": "" };


        }
        // Step 1: Ends Here


        // Set 2: Ajax Post
        // Here i have used ajax post for saving/updating information
        $.ajax({
            url: '/Sales/Create',
            data: JSON.stringify(salesmain),
            type: 'POST',
```

```
            contentType: 'application/json;',
            dataType: 'json',
            success: function (result) {

                if (result.Success == "1") {
                    window.location.href = "/Sales/index";
                }
                else {
                    alert(result.ex);
                }
            }
        });


    }
```

Summery

Here I have discussed about Sales Controller and only create view. Index, details and Delete views are skipped because they are as usual. Edit view has deleted because here we have used Create view for performing both create and Edit operation.

If you want to know more about datatable and JSON then you can check following links.

http://www.datatables.net/

http://json.org/