Learn MVC Project in 7 days – Day 3

Marla Sukesh, 27 Jul 2015

In this article we will take our MVC 5 Project into next level by introducing Data Entry Screen and Database Communication.

Download Day_3.zip

Introduction

We are assuming that you have read previous days before coming here. In day 2 we have completed a project which had a grid with List of Employees.
In day 3 we will take it to the next level by introducing data access layer and data entry screen.

Complete Series

- Day 1
- Day 2
- Day 3
- Day 4
- Day 5
- Day 6
- Day 7
- Bonus Day 1
- Bonus Day 2

We are pleased to announce that this article is now available as hard copy book you can get the same from www.amazon.com and www.flipkart.com

Agenda

Data Access Layer

A real time project is incomplete without Database. In our project we didn't spoke database layer yet. First Lab of Day 3 will be all about database and database layer.

Here we will use Sql Server and Entity Framework for creating Database and Database Access layer respectively.

What is Entity Framework in simple words?

It's an ORM tool. ORM stands for Object Relational Mapping.

In RDBMS world, we speak in terms of Tables and Columns whereas in .net world (which is an object oriented world), we speak in terms of Classes, objects and properties.

When we think about any data driven application we end up with following two things.

- Write code for communicating with database (called Data Access Layer or Database logic)
- Write code for mapping Database data to object oriented data or vice versa.

ORM is a tool which will automate these two things. Entity framework is Microsoft ORM tool.

What is Code First Approach?

In Entity framework we can follow one of these three approaches

- Database First approach – Create database with tables, columns, relations etc. and Entity framework will generates corresponding Model classes (Business entities) and Data Access Layer code.
- Model First approach – In this approach Model classes and relationship between them will be defined manually using Model designer in Visual studio and Entity Framework will generate Data Access Layer and Databa
- Code First approach – In this approach manually POCO classes will be created. Relationship between those classes will be defined by means of code.When application executes for the first time Entity framework will ge column and relations automatically in the database server.

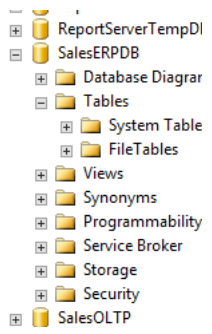
What is mean by POCO classes?

POCO stands for "Plain Old CLR objects". POCO classes means simple .Net classes we create. In our previous example Employee class was simply a POCO class.

Lab 8 – Add Data Access layer to the project

Step 1– Create Database

Connect to the Sql Server and create new Database called "SalesERPDB".



```
ReportServerTempDI
SalesERPDB
    Database Diagrar
    Tables
        System Table
        FileTables
    Views
    Synonyms
    Programmability
    Service Broker
    Storage
    Security
SalesOLTP
```

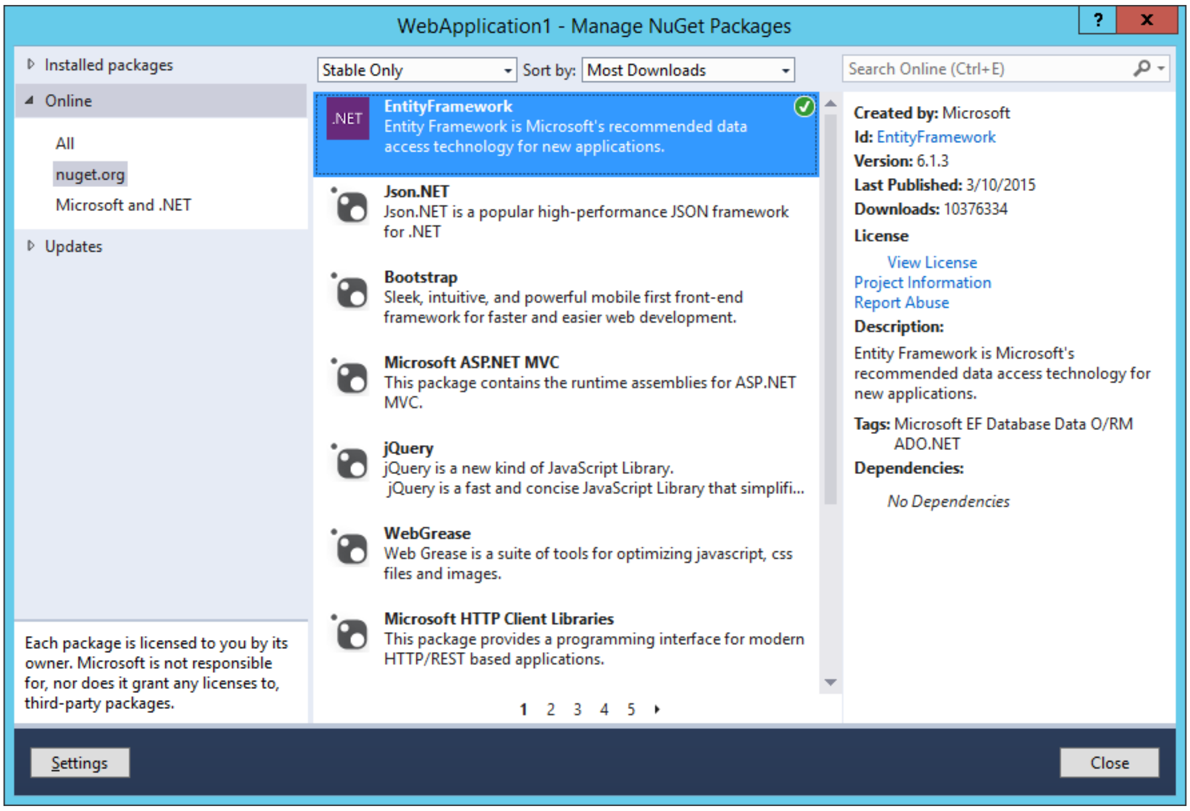
Step 2 – Create ConnectionString

Open Web.config file and inside Configuration section add following section

```
<connectionStrings>
<add connectionString="Data Source=(local);Initial Catalog=SalesERPDB;Integrated Security=True"

        name="SalesERPDAL"

        providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Step 3– Add Entity Framework reference

Right click the project >> Manage Nuget packages. Search for Entity Framework and click install.



WebApplication1 - Manage NuGet Packages

EntityFramework
Entity Framework is Microsoft's recommended data access technology for new applications.

Json.NET
Json.NET is a popular high-performance JSON framework for .NET

Bootstrap
Sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.

Microsoft ASP.NET MVC
This package contains the runtime assemblies for ASP.NET MVC.

jQuery
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifi...

WebGrease
Web Grease is a suite of tools for optimizing javascript, css files and images.

Microsoft HTTP Client Libraries
This package provides a programming interface for modern HTTP/REST based applications.

Created by: Microsoft
Id: EntityFramework
Version: 6.1.3
Last Published: 3/10/2015
Downloads: 10376334
License
View License
Project Information
Report Abuse
Description:
Entity Framework is Microsoft's recommended data access technology for new applications.
Tags: Microsoft EF Database Data O/RM ADO.NET
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Step 4 – Create Data Access layer.

- Create a new folder called "DataAccessLayer" in the root folder and inside it create a new class called "SalesERPDAL"
- Put using statement at the top as follows.

-
  ```
  using System.Data.Entity;
  ```
- Inherit "SalesERPDAL" from DbContext

```
public class SalesERPDAL: DbContext
{
}
```

Step 5 – Create primary key field for employee class

Open Employee class and put using statement at the topas follows.

```
using System.ComponentModel.DataAnnotations;
```

Add EmployeeId property in Employee class and mark it as Key attribute.

```
public class Employee
{
    [Key]
    public int EmployeeId  { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int Salary { get; set; }
}
```

Step 6 – Define mapping

Put following using statement in the top for "SalesERPDAL" class

```
using WebApplication1.Models;
```

Override OnModelCreating method in SalesERPDAL class as follows.

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.Entity<employee>().ToTable("TblEmployee");
    base.OnModelCreating(modelBuilder);
}
</employee>
```

Note: In above code snippet "TblEmployee" represents the table name. It automatically get created in runtime.

Step 7 – Create property to hold Employees in Database

Create a new property called Employee in "SalesERPDAL" class as follows

```
public DbSet<employee> Employees{get;set;}
</employee>
```

DbSet will represent all the employees that can be queried from the database.

Step 8– Change Business Layer Code and get data from Database

Open EmployeeBusinessLayer class.Put using statement in the top.

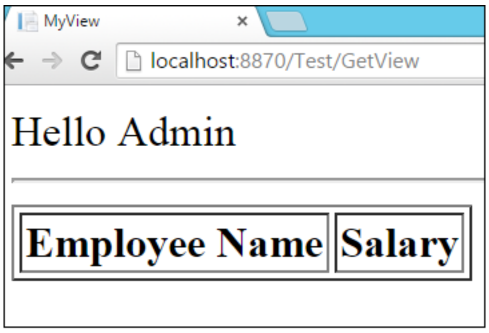
```
using WebApplication1.DataAccessLayer;
```

Now change GetEmployees method class as follows.

```
public List<employee> GetEmployees()
{
    SalesERPDAL salesDal = new SalesERPDAL();
    return salesDal.Employees.ToList();
}
</employee>
```

Step 9 – Execute and Test

Press F5 and execute the application.

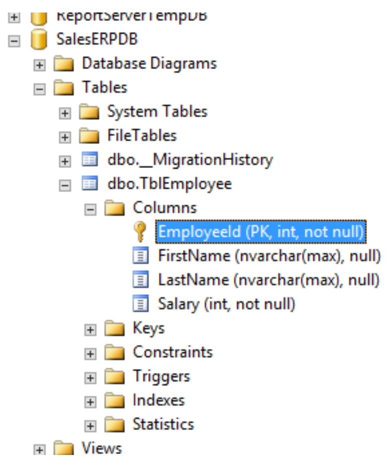


Hello Admin

Employee Name	Salary

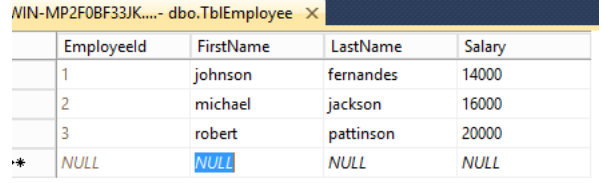
Right now we don't have any employees in the database so we will see a blank grid.

Check the database. Now we have a table called TblEmployee with all the columns.



Step 9 – Insert Test Data

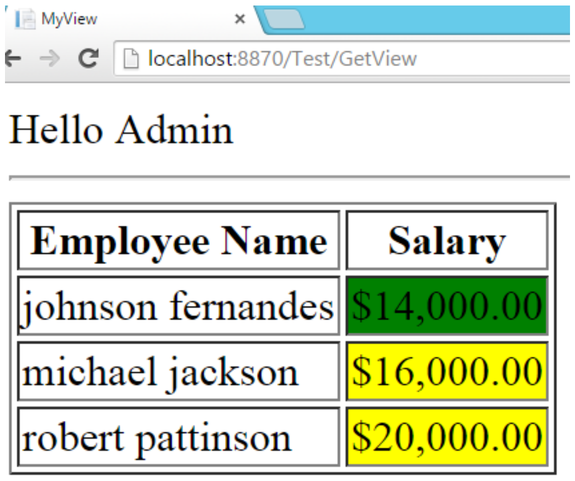
Add some dummy data to TblEmployee table.


WIN-MP2F0BF33JK....- dbo.TblEmployee

	EmployeeId	FirstName	LastName	Salary
	1	johnson	fernandes	14000
	2	michael	jackson	16000
	3	robert	pattinson	20000
*	NULL	NULL	NULL	NULL

Step 10 – Execute and test the application

Press F5 and run the application again.


MyView

localhost:8870/Test/GetView

Hello Admin

Employee Name	Salary
johnson fernandes	$14,000.00
michael jackson	$16,000.00
robert pattinson	$20,000.00

Here we go :)

Talk on Lab 8

What is DbSet?

DbSet simply represent the collection of all the entities that can be queried from the database. When we write a Linq query again DbSet object it internally converted to query and fired against database.

In our case “Employees” is a DbSet which will hold all the “Employee” entitieswhich can be queried from database. Every time we try to access “Employees” it gets all records in the “TblEmployee” table and convert it to “Empl

How connection string and data access layer is connected?

Mapping will be done based on name. In our example ConnectionString Name and Data Access Layer class name is same that is “SalesERPDAL”, hence automatically mapped.

Can we change the ConnectionString name?

Yes, in that case we have to define a constructor in Data Access Layer class as follows.

```
public SalesERPDAL():base("NewName")
{
}
```

Organize everything

Just to make everything organized and meaningful let’s do couple of changes.

Step 1 - Rename

- “TestController” to “EmployeeController”
- GetView action method to Index
- Test folder (inside Views folder) to Employee
- and “MyView” view to “Index”

Step 2 – Remove UserName property from EmployeeListViewModel

Step 3 – Remove UserName from View

Open Views/Employee.Index.cshtml View and remove username from it.

In simple words, remove following block of code.

```
 Hello @Model.UserName
<hr />
```

Step 2 – Change Index Action Method in EmployeeController

Accordingly Change the code in Index action in EmployeeController as follows.

```
public ActionResult Index()
{
    ……
    ……
    ……
    employeeListViewModel.Employees = empViewModels;
    //employeeListViewModel.UserName = "Admin";-->Remove this line -->Change1
    return View("Index", employeeListViewModel);//-->Change View Name -->Change 2
}
```

Now at the time of execution URL will “**…./Employee/Index”**

Lab 9 – Create Data Entry Screen

Step 1 – Create action method

Create an action method called “AddNew” in EmployeeController as follows

```
public ActionResult AddNew()
{
    return View("CreateEmployee");
}
```

Step 2 – Create View

Create a view called “CreateEmployee” inside View/Employee folder as follows.

```
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
    <head>
      <meta name="viewport" content="width=device-width" />
      <title>CreateEmployee</title>
    </head>
    <body>
      <div>
         <form action="/Employee/SaveEmployee" method="post">
            First Name: <input type="text" id="TxtFName" name="FirstName" value="" /><br />
            Last Name: <input type="text" id="TxtLName" name="LastName" value="" /><br />
            Salary: <input type="text" id="TxtSalary" name="Salary" value="" /><br />
            <input type="submit" name="BtnSave" value="Save Employee" />
            <input type="button" name="BtnReset" value="Reset" />
         </form>
      </div>
    </body>
</html>
```

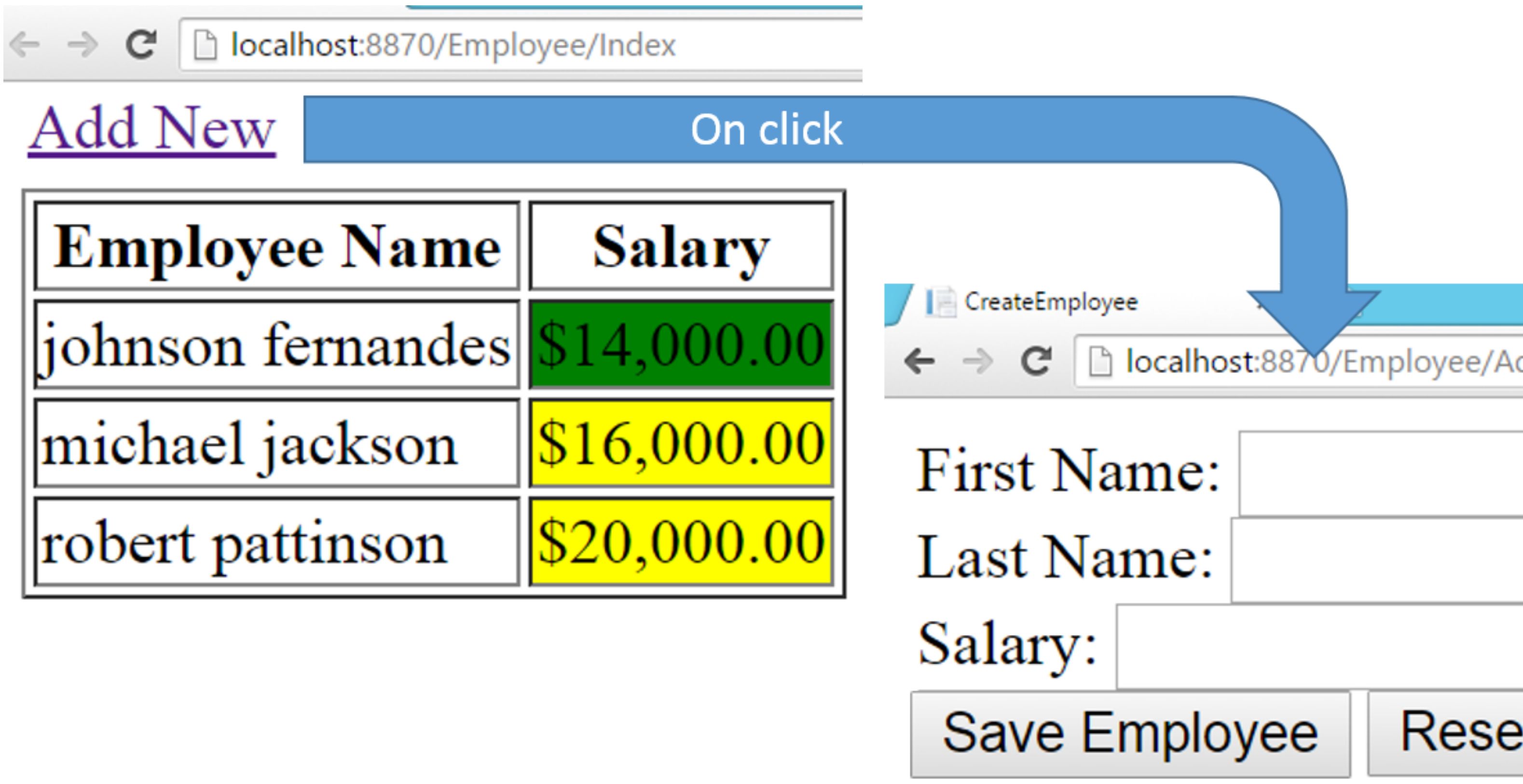
Step 3 – Create a link in Index View

Open Index.cshtml and add a hyperlink pointing to AddNew Action URL.

```
<ahref="/Employee/AddNew">Add New</a>
```

Step 4 –Execute and Test the application

Press F5 and execute the application



Employee Name	Salary
johnson fernandes	$14,000.00
michael jackson	$16,000.00
robert pattinson	$20,000.00

Talk on Lab 9

What is the purpose of form tag?

In day 1 of the series we have understood that “Web world won’t follow Event driven programming model. It follows request-response model. End user make the request and server sendsresponse.” Form tag is one of the way
button inside form tag gets clicked, a request will be sent to the URL specified in action attribute.

What is method attribute in Form tag?

It decides the type of request. Request may be one of the following four types - get, post, put and delete.

As per the web standards we should use–

- Get - > When we want to get something
- Post -> When we want to create something
- Put -> When we want to update something
- Delete -> when we want to delete something.

How making request using Form tag is different from making request via browser address bar or hyperlink?

When request is made with the help of Form tag, values of all the input controls are sent with the request for processing.

What about checkbox, radio buttons and Dropdowns? Will values of this controls also sent?

Yes, All input controls (input type=text, type=radio, type=checkbox) and also dropdowns (which represented as “Select” element).

How values will be sent to server?

When request is of type Get, Put or Delete, values will be sent as Query string parameters.

When it's a post request values will be sent as posted data.

What is the purpose of name attribute in input controls?

As discussed before values of all input controls will be sent along with request when submit button is clicked. It makes server receive more than one value at a time. To distinguish each value separately while sending every va
"name" attribute.

Does name and id attribute serves same purpose?

No, as per last question "name" attribute will be used internally by HTML when the request is being sent whereas "id" attribute will be used by developers inside JavaScript for some dynamic stuffs.

What is the difference between "input type=submit" and "input type=button"?

Submit button will be specially used when we want to make request to the server whereas simple button will be used to perform some custom client side actions. Simple button won't do anything by its own.

Lab 10 – Get posted data in Server side/Controllers

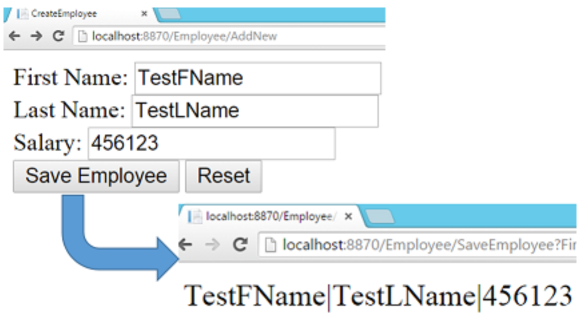
Step 1 – Create SaveEmployee Action method

Inside Employee Controller create an action method called SaveEmployee as follows.

```
public string SaveEmployee(Employee e)
{
   return e.FirstName + "|"+ e.LastName+"|"+e.Salary;
}
```

Step 2 – Execute and Test

Press F5 and execute the application.


CreateEmployee
localhost:8870/Employee/AddNew
Save Employee
Reset
localhost:8870/Employee/
localhost:8870/Employee/SaveEmployee?Fir

First Name: TestFName
Last Name: TestLName
Salary: 456123

TestFName|TestLName|456123

Talk on Lab 10

How Textbox values are updated in Employee object inside action method?

In Asp.Net MVC there is a concept called as Model Binder.

- Model Binder will executes automatically whenever a request is made to an action method containing parameter.
- Model binder will iterate though all primitive parameters of a method and then it will compare name of the parameter with each key in the incoming data (Incoming data means either posted data or query string).Whe
assigned to the parameter.
- After that Model binder will iterate through each and every property of each and every class parameter and compare each property name with each key in incoming data.When match is found, corresponding incoming

What will happen when two parameters are specified, one as "Employee e" and second as "string FirstName"?

FirstName will be updated in both primitive FirstName variable and e.FirstName property.

Will Model Binder workwith composition relationship?

Yes it will, but in that case name of the control should be given accordingly.

Example

Let say we have Customer class and Address class as follows

```
public class Customer
{
    public string FName{get;set;}
    public Address address{get;set;}
}
public class Address
{
    public string CityName{get;set;}
    public string StateName{get;set;}
}
```

In this case Html should look like this

```
...
...
...
<input type="text" name="FName">
<input type="text" name="address.CityName">
<input type="text" name="address.StateName">
...
...
...
```

Lab 11 – Reset and Cancel button

Step 1 – Start withReset and Cancel button

Add a Reset and Cancel button as follows

```
...

...

...

<input type="submit" name="BtnSubmit" value="Save Employee" />

<input type="button" name="BtnReset" value="Reset" onclick="ResetForm();" />
```

```
<input type="submit" name="BtnSubmit" value="Cancel" />
```

Note: Save button and Cancel button have same "Name" attribute value that is "BtnSubmit".

Step 2 – define ResetForm function

In Head section of Html add a script tag and inside that create a JavaScript function called ResetForm as follows.

```
<script>
    function ResetForm() {
        document.getElementById('TxtFName').value = "";
        document.getElementById('TxtLName').value = "";
        document.getElementById('TxtSalary').value = "";
    }
</script>
```

Step 3 – Implement cancel click in EmplyeeController's SaveEmployee action method.

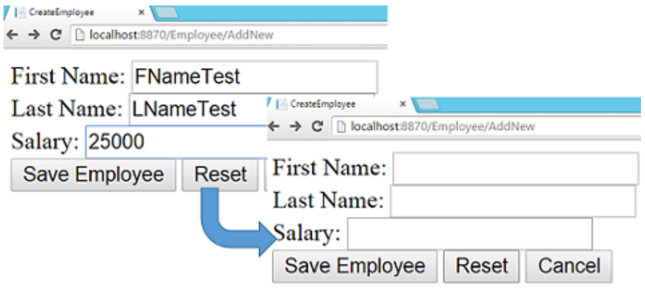
Change SaveEmployee action method as following

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        case "Save Employee":
            return Content(e.FirstName + "|" + e.LastName + "|" + e.Salary);
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

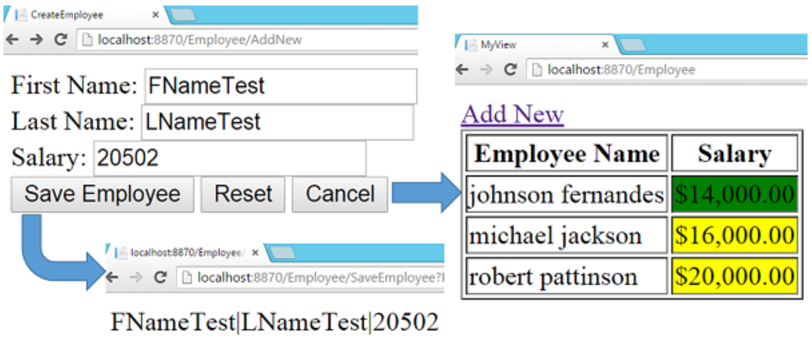
Step 4 – Execute the application.

Press F5 and execute the application. Navigate to the AddNew screen by clicking "Add New" link.

Step 5 – Test Reset functionality



Step 6 – Test Save and Cancel functionality



Employee Name	Salary
johnson fernandes	$14,000.00
michael jackson	$16,000.00
robert pattinson	$20,000.00

```
FNameTest|LNameTest|20502
```

Talk on Lab 11

Why same name is given to both Save and Cancel button?

We know that, as soon as submit button is clicked, a request is sent to the server.Along with the request values of all the input controls will be sent.

Submit button is also an input button. Hence value of the submit button (which is responsible for the request) will be sent too.

When Save button will be clicked, value of Save button that is "Save Employee" will be sent with request and when Cancel button is clicked, value of Cancel button that is "Cancel" will sent with request.

In Action method, Model Binder will do remaining work. It will update the parameter values with values in input data (coming with request)

What are the other ways to implement multiple submit buttons?

There are many ways. I would like to discuss three of them.

1. Hidden Form element

Step 1 – Create a hidden form element in View as follows.

```
<form action="/Employee/CancelSave" id="CancelForm" method="get" style="display:none">

</form>
```

<form action="/Employee/CancelSave" id="CancelForm" method="get" style="display:none"> </form>

Step 2 – Change Submit button to normal button and post above form with the help of JavaScript.

```
<input type="button" name="BtnSubmit" value="Cancel" onclick="document.getElementById('CancelForm').submit()" />
```

2. Change action URL dynamically using JavaScript

```
<form action="" method="post" id="EmployeeForm" >
...
...
<input type="submit" name="BtnSubmit" value="Save Employee" onclick="document.getElementById('EmployeeForm').action = '/Employee/SaveEmployee'" />
...
```

```
<input type="submit" name="BtnSubmit" value="Cancel" onclick="document.getElementById('EmployeeForm').action = '/Employee/CancelSave'" />
</form>
```

3. Ajax

Instead of submit button use simple input button and onclick of it make pure Ajax request using jQuery or any other library.

Why we have not used input type=reset for implementing Reset functionality?

Input type=reset control won't clear the values, it just set the value to default value of a control. Example:

```
<input type="text" name="FName" value="Sukesh">
```

In above example default value of control is "Sukesh".

If we use input type=reset for implementing Reset functionality then by default "Sukesh" will be set in the textbox every time "reset" button is clicked.

What if names are not matching with property names of the classes?

This is a very common question during interviews.

Let say we have HTML as follows

```
First Name: <input type="text" id="TxtFName" name="FName" value="" /><br />
Last Name: <input type="text" id="TxtLName" name="LName" value="" /><br />
Salary: <input type="text" id="TxtSalary" name="Salary" value="" /><br />
```

Now our Model class contain property names as FirstName, LastName and Salary. Hence default model binder won't work here.

In this situation we have following three solutions

- Inside action method, retrieve posted values using Request.Form syntax and manually construct the Model object as follows.

```
public ActionResult SaveEmployee()
{
        Employee e = new Employee();
        e.FirstName = Request.Form["FName"];
        e.LastName = Request.Form["LName"];
        e.Salary = int.Parse(Request.Form["Salary"])
...
...
}
```

- Use parameter names and Creates Model object manually as follows.

```
public ActionResult SaveEmployee(string FName, string LName, int Salary)
{
    Employee e = new Employee();
    e.FirstName = FName;
    e.LastName = LName;
    e.Salary = Salary;
...
...
}
```

- Create Custom Model Binder and replace default model binder as follows.

Step 1 – Create Custom Model Binder

```
public class MyEmployeeModelBinder : DefaultModelBinder
{
    protected override object CreateModel(ControllerContext controllerContext, ModelBindingContext bindingContext, Type modelType)
    {
        Employee e = new Employee();
        e.FirstName = controllerContext.RequestContext.HttpContext.Request.Form["FName"];
        e.LastName = controllerContext.RequestContext.HttpContext.Request.Form["LName"];
        e.Salary = int.Parse(controllerContext.RequestContext.HttpContext.Request.Form["Salary"]);
        return e;
    }
}
```

Step 2- Replace default model binder with this new model binder

```
public ActionResult SaveEmployee([ModelBinder(typeof(MyEmployeeModelBinder))]Employee e, string BtnSubmit)
{
    ......
```

What does RedirecttToFunction do?

It generates RedirectToRouteResultJust like ViewResult and ContentResult (discussed in Day 1), RedirectToRouteResult is a child of ActionResult.It represents the redirect response. When browser receives RedirectToRouteResul

Note: Here browser is responsible for new request hence URL will get updated to new URL.

What is EmptyResult?

One more child of ActionResult. When browser receives EmptyResult as a response it simply displays blank white screens. It simply represents "No Result".

In our example this situation won't happen. Just to make sure that all code paths returns a value EmptyResult statement was written.

Note: When ActionMethod return type is Void, it is equivalent to EmptyResult

Lab 12 – Save records in database and update Grid

Step 1 – Create SaveEmployee in EmployeeBusinessLayer as follows

```
public Employee SaveEmployee(Employee e)
{
    SalesERPDAL salesDal = new SalesERPDAL();
    salesDal.Employees.Add(e);
    salesDal.SaveChanges();
    return e;
}
```

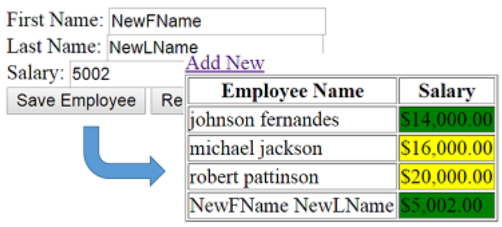
Step 2 – Change SaveEmployee Action method

In EmployeeController change the SaveEmployee action method code as follows.

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        case "Save Employee":
            EmployeeBusinessLayer empBal = new EmployeeBusinessLayer();
            empBal.SaveEmployee(e);
            return RedirectToAction("Index");
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

Step 3 – Execute and Test

Press F5 and execute the application. Navigate to Data entry screen and put some valid values.



Employee Name	Salary
johnson fernandes	$14,000.00
michael jackson	$16,000.00
robert pattinson	$20,000.00
NewFName NewLName	$5,002.00

Lab 13 – Add Server side Validation

In Lab 10 we have seen basic functionality of Model Binder. Let understand a little more about same.

- Model binder updates the Employee object with the posted data.
- But this is not the only functionality performed by Model Binder. Model Binder also updates ModelState. ModelState encapsulates the state of the Model.
    - It have a property called IsValid which determines whether the Model (that is Employee object) gets successfully updated or not.Model won't update if any of the server side validation fails.
    - It holds validation errors.
      Example:ModelState["FirstName "].Errors will contain all errors related to First Name
    - It holds the incoming value(Posted data or queryString data)
    - It holds the incoming value(Posted data or queryString data)

In Asp.net MVC we use DataAnnotations to perform server side validations.

Before we get into Data Annotation lets understand few more things about Model Binder

How Model Binder work with primitive datatypes

When Action method contain primitive type parameter, Model Binder will compare name of the parameter with each key in the incoming data (Incoming data means either posted data or query string).

- When match is found, corresponding incoming data will be assigned to the parameter.
- When match is not found, parameter will be assigned with default value. (Default value – For integer it is 0 (zero), for string it is null etc.)
- In case assignment is not possible because of datatype mismatch exception will be thrown.

How Model Binder work with classes

When parameter is a Class parameter, Model Binder will iterate through all properties of all the class and compare each property name with each key in incoming data.

- When match is found,
    - If corresponding incoming value is empty, then
        - Null value will be assigned to property. If null assignment is not possible, default value will be set and ModelState.IsValid will be set to false.
        - If null assignment is possible but will be considered as invalid value because of the validation attached to the property then null be assigned as value and ModelState.IsValid will be set to false.
    - If corresponding incoming value is non empty,
        - In case assignment is not possible because of datatype mismatch or Server side validation failure null value will be assigned and ModelState.IsValid will be set to false.
            - If null assignment is not possible, default value will be set
- When match is not found, parameter will be assigned with default value. (Default value – For integer it is 0 (zero), for string it is null etc.)In this case ModelState.IsValid will remain unaffected.

Let's understand same by adding validation feature to our on-going project.

Step 1 – Decorate Properties with DataAnnotations

Open Employee class from Model folder and decorate FirstName and LastName property with DataAnnotation attribute as follows.

```
public class Employee
{
...
...
    [Required(ErrorMessage="Enter First Name")]
    public string FirstName { get; set; }

    [StringLength(5,ErrorMessage="Last Name length should not be greater than 5")]
    public string LastName { get; set; }
...
...
}
```

Step 2 – Change SaveEmployee Action method

Open EmplyeeController and Change SaveEmployee Action method as follows.

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        case "Save Employee":
            if (ModelState.IsValid)
            {
                EmployeeBusinessLayer empBal = new EmployeeBusinessLayer();
                empBal.SaveEmployee(e);
                return RedirectToAction("Index");
            }
            else
            {
                return View("CreateEmployee ");
            }
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

Note: As you can see, When ModelState.IsValid is false response of SaveEmployee button click is ViewResult pointing to "CreateEmployee" view.

Step 3 – Display Error in the View

Change HTML in the "Views/Index/CreateEmployee.cshtml" to following.

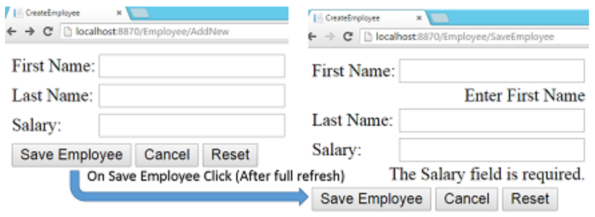
This time we will format our UI a little with the help of "table" tag;

```
<table>
   <tr>
      <td>
         First Name:
      </td>
      <td>
         <input type="text" id="TxtFName" name="FirstName" value="" />
      </td>
   </tr>
   <tr>
      <td colspan="2" align="right">
        @Html.ValidationMessage("FirstName")
      </td>
   </tr>
   <tr>
      <td>
        Last Name:
      </td>
      <td>
         <input type="text" id="TxtLName" name="LastName" value="" />
      </td>
   </tr>
   <tr>
      <td colspan="2" align="right">
         @Html.ValidationMessage("LastName")
      </td>
   </tr>

   <tr>
      <td>
        Salary:
      </td>
      <td>
         <input type="text" id="TxtSalary" name="Salary" value="" />
      </td>
   </tr>
   <tr>
      <td colspan="2" align="right">
        @Html.ValidationMessage("Salary")
      </td>
   </tr>

   <tr>
      <td colspan="2">
         <input type="submit" name="BtnSubmit" value="Save Employee" />
         <input type="submit" name="BtnSubmit" value="Cancel" />
         <input type="button" name="BtnReset" value="Reset" onclick="ResetForm();" />
      </td>
   </tr>
</table>
```

Step 4 – Execute and Test

Press F5 and execute the application. Navigate to "Employee/AddNew" action method and test the application.

Test 1


CreateEmployee
localhost:8870/Employee/AddNew

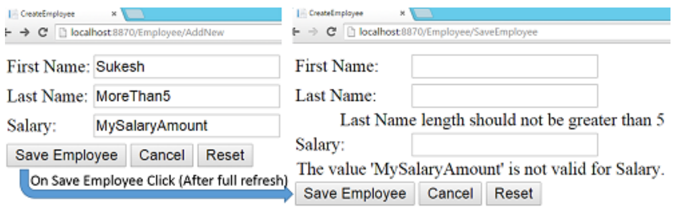
First Name:
Last Name:
Salary:
Save Employee | Cancel | Reset

On Save Employee Click (After full refresh)

CreateEmployee
localhost:8870/Employee/SaveEmployee

First Name:
Enter First Name
Last Name:
Salary:
The Salary field is required.
Save Employee | Cancel | Reset

Test 2


CreateEmployee
localhost:8870/Employee/AddNew

First Name: Sukesh
Last Name: MoreThan5
Salary: MySalaryAmount
Save Employee | Cancel | Reset

On Save Employee Click (After full refresh)

CreateEmployee
localhost:8870/Employee/SaveEmployee

First Name:
Last Name:
Last Name length should not be greater than 5
Salary:
The value 'MySalaryAmount' is not valid for Salary.
Save Employee | Cancel | Reset

Note: You may end up with following error.
"The model backing the 'SalesERPDAL' context has changed since the database was created. Consider using Code First Migrations to update the database."

To remove this error, simply add following statement in Application_Start in Global.asax file.

```
Database.SetInitializer(new DropCreateDatabaseIfModelChanges<SalesERPDAL>());
```

Database class exists inside System.Data.Entity namespace

If you are still getting the same error then, open database in Sql server and just delete __MigrationHistory table.

Soon I will release a new series on Entity Framework where we will learn Entity framework step by step. This series is intended to MVC and we are trying to stick with it.

Talk on lab 13

What does @Html.ValidationMessage do?

- @ means it's a razor code
- Html is an instance of HtmlHelper class available inside view.
- ValidationMessageis a function of HtmlHelper class which displays the error message

How ValidationMessage function works?

ValidationMessage is a function. It executes at runtime. As we discussed earlier, ModelBinder updates the ModelState. ValidationMessage displays the error message from ModelState based on Key.

Example: ValidationMessage("FirstName") displays the error message related to First Name.

Do we have more attributes like required and StringLength?

Yes, here are some

- DataType – Make sure that data is of particular type like email, credit card number, URL etc.
- EnumDataTypeAttribute–Make sure that value exist in an enumeration.
- Range Attribute – Make sure that value is in a particular range.
- Regular expression- Validates the value against a particular regular expression.
- Required – Make sure that value is provided.
- StrengthLength – Validates the string for maximum and minimum number of characters.

How Salary is getting validated?

We have not added any Data Annotation attribute to Salary attribute but still it's getting validated. Reason for that is, Model Binder also considers the datatype of a property while updating model.

In Test 1 – we had kept salary as empty string. Now in this case, as per the Model binderexplanation we had (In Lab 13), ModelState.IsVaid will be false and ModelState will hold validation error related to Salary which will disp
Html.ValidationMessage("Salary")

In Test 2 – Salary data type is mismatched hence validation is failed.

Is that means, integer properties will be compulsory by default?

Yes, Not only integers but all value types because they can't hold null values.

What if we want to have a non-required integer field?

Make it nullable?

```
public int? Salary{get;set;}
```

How to change the validation message specified for Salary?

Default validation support of Salary (because of int datatype) won't allow us to change the validation message. We achieve the same by using our own validation like regular expression, range or Custom Validator.

Why values are getting cleared when validation fails?

Because it's a new request. DataEntry view which was rendered in the beginning and which get rendered later are same from development perspective but are different from request perspective. We will learn how to maintain

Can we explicitly ask Model Binder to execute?

Yes simply remove parameters from action method. It stops default model binder from executing by default.

In this case we can use UpdateModel function as follows.

```
Employee e = new Employee();
UpdateModel<employee>(e);
</employee>
```

Note: UpdateModel won't work with primitive datatypes.

What is the difference between UpdateModel and TryUpdateModel method?

TryUpdateModel will be same as UpdateModel except one added advantage.

UpdateModel will throw an exception if Model adaption fails because of any reason. In case of UpdateModel function ModelState.IsValid will not be of any use.

TryUpdateModel is exactly same as keeping Employee object as function parameter. If updating fails ModelState.IsValid will be false;

What about client side validation?

It should be done manually unless and until we are using HTML Helper classes.

We are going to talk about both manual client side validation and automatic client side validation with the help of HTML helper classes in day 4.

Can we attach more than one DataAnnotation attribute to same property?

Yes we can. In that case both validations will fire.

Lab 14 – Custom Server side validation

Step 1 – Create Custom Validation

Open Employee.cs file and create a new class Called FirstNameValidation inside it as follows.

```
public class FirstNameValidation:ValidationAttribute
{
    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
    {
        if (value == null) // Checking for Empty Value
        {
            return new ValidationResult("Please Provide First Name");
        }
        else
        {
            if (value.ToString().Contains("@"))
            {
                return new ValidationResult("First Name should Not contain @");
            }
        }
        return ValidationResult.Success;
    }
}
```

Note: Creating multiple classes inside single file is never consider as good practice. So in your sample I recommend you to create a new folder called "Validations" in root location and create a new class inside it.

Step 2- Attach it to First Name

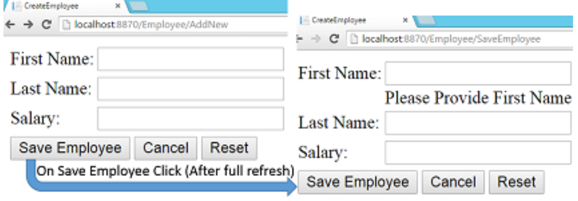
Open Employee class and remove the default "Required" attribute from FirstName property and attach FirstNameValidation as follows.

```
[FirstNameValidation]
public string FirstName { get; set; }
```

Step 3 – Execute and Test

Press F5. Navigate to "Employee/AddNew" action.

Test 1



CreateEmployee
localhost:8870/Employee/AddNew

First Name:	
Last Name:	
Salary:	

Save Employee | Cancel | Reset

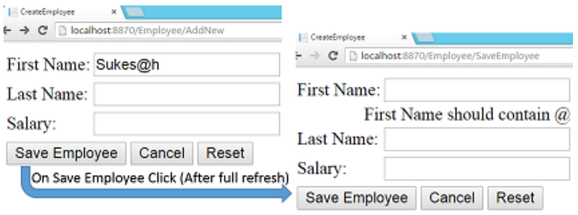
On Save Employee Click (After full refresh)

CreateEmployee
localhost:8870/Employee/SaveEmployee

First Name:	
	Please Provide First Name
Last Name:	
Salary:	

Save Employee | Cancel | Reset

Test 2



CreateEmployee
localhost:8870/Employee/AddNew

First Name:	Sukes@h
Last Name:	
Salary:	

Save Employee | Cancel | Reset

On Save Employee Click (After full refresh)

CreateEmployee
localhost:8870/Employee/SaveEmployee

First Name:	
	First Name should contain @
Last Name:	
Salary:	

Save Employee | Cancel | Reset

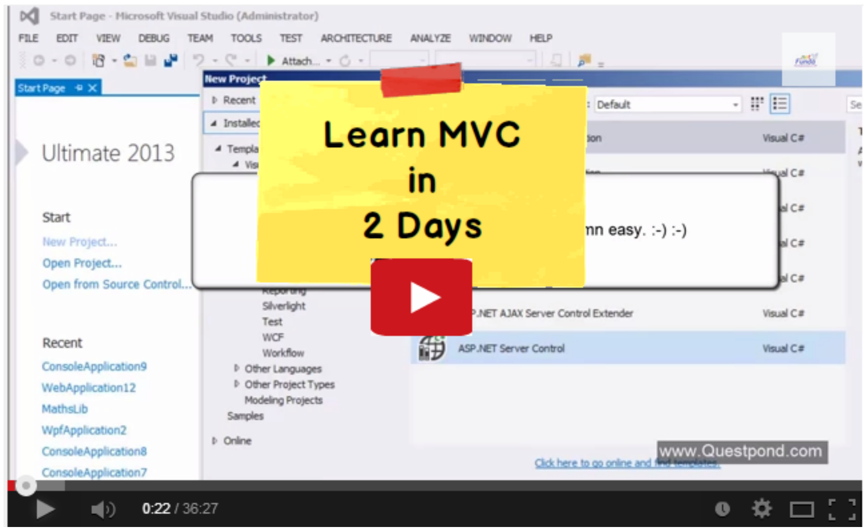
Conclusion

Here we complete our day 3. In Day 4 we will take our project into next version. Here is the agenda in Day 4

- Implement client side validation
- Understand HTML helper
- Implement Authentication
- Adding Footers with partial Views
- Create consistent layout with Master Pages
- Custom Request filtering

Connect us on Facebook, LinkedIn or twitter to stay updated about new releases.

For Offline Technical trainings in Mumbai visit StepByStepSchools.Net
For Online Trainings visit JustCompile.com



License

About the Author

Marla Sukesh

Instructor / Trainer Train IT
India

Learning is fun but teaching is awesome.

Code re-usability is my passion ,Teaching and learning is my hobby, Becoming an successful entrepreneur is my goal.

By profession I am a Corporate Trainer.
I do trainings on WCF, MVC, Business Intelligence, Design Patterns, HTML 5, jQuery, JSON and many more Microsoft and non-Micrsoft technologiees.

Find my profile here

My sites

- **justcompile.com**
- **www.sukesh-marla.com**

@Twitter
@Facebook

Comments and Discussions

207 messages have been posted for this article Visit **https://www.codeproject.com/Articles/986730/Learn-MVC-Project-in-days-Day** to post and view comments on this article, or click **here** to get a print view with me