# In ASP.NET MVC Passing Results of LINQ to View

**OmarIsaid**, 10 Dec 2016

5.00 (1 vote) Rate this:

Solution for passing data from View to controller, contains useful concepts for beginner developers

## Introduction

This tip/trick shows a simple way for passing data between Controller and View. The current tip/trick shows how to pass the result of LINQ query to View, and show its results using `foreach` loop.

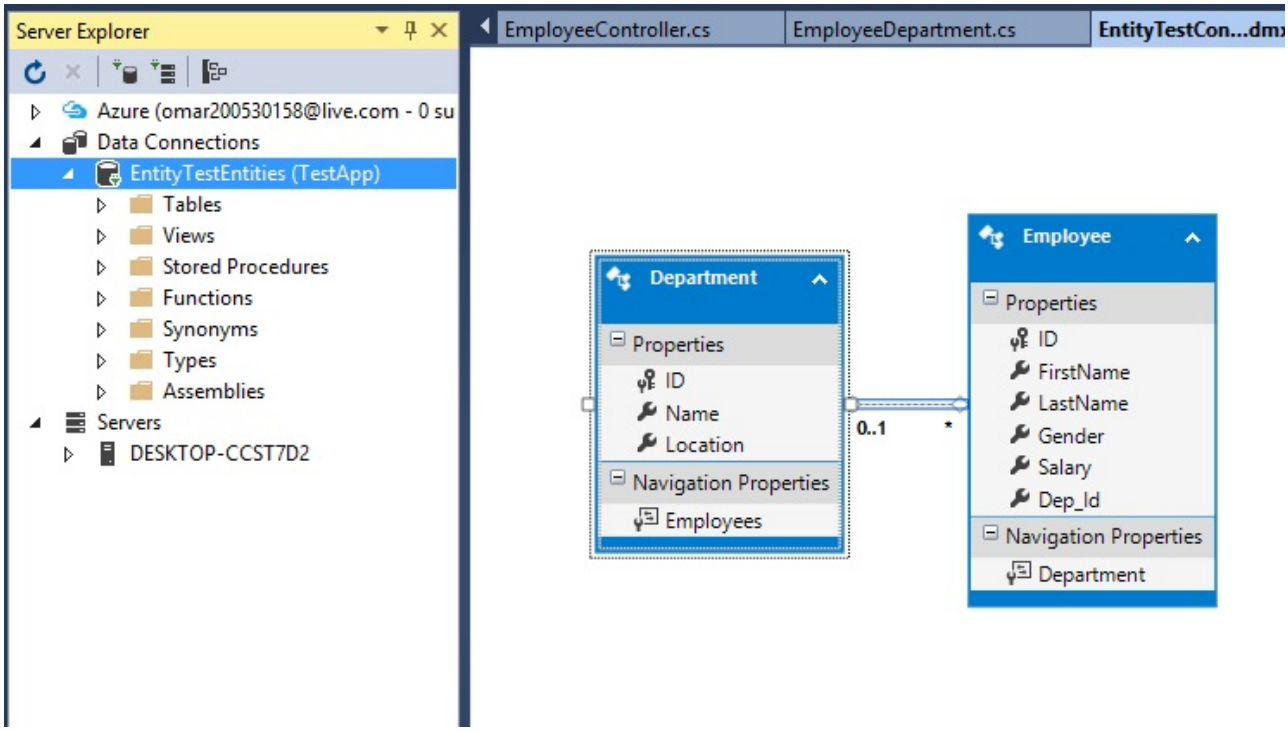
## Background

ASP.NET MVC is an advanced technology from Micrososft for developing web sites, that it provides full separation between View, Controller, and Model layers.

## Using the Code

This solution shows few concepts related to ASP.NET can be a good example for beginners in this field. This solution has ideas and solutions related to LINQ , Entity Framework.

First, add item ADO Entity Data Model, and create models on this file using the database.




Above is the ADO Entity Model after generating the models from the database, the model consists of two tables that have one to many relation. The `department` has many relations with `Employee`. The ADO Entity Model automatically generates the layers of the `Model` classes.

In this solution, the `employee`s of a certain `department` will be returned, the `ViewModel` will be created to show the properties of `Employees` of `department` and properties of the related `department`.

The `ViewModel` contains property of type `IList`, to receive the results of inner `Join` query of LINQ, and display its contents on `View`.

Hide Copy Code

```
public class EmployeeDepartment
{
 public string departmentName {get;set;}
 public string departmentLocation {get;set;}
 public string FirstName {get;set;}
 public string LastName {get;set;}
 public IList EmployeeDepartment MenuItems{get;set;}
}
```

The controller is responsible for processing the data that is retrieved from the model and passing it to View. So, the controller is the heart of ASP.NET MVC. Passing values from controller to View can be binding controller to view, or using properties using `ViewBag`, `ViewData`, and `TempData`.

Hide Copy Code

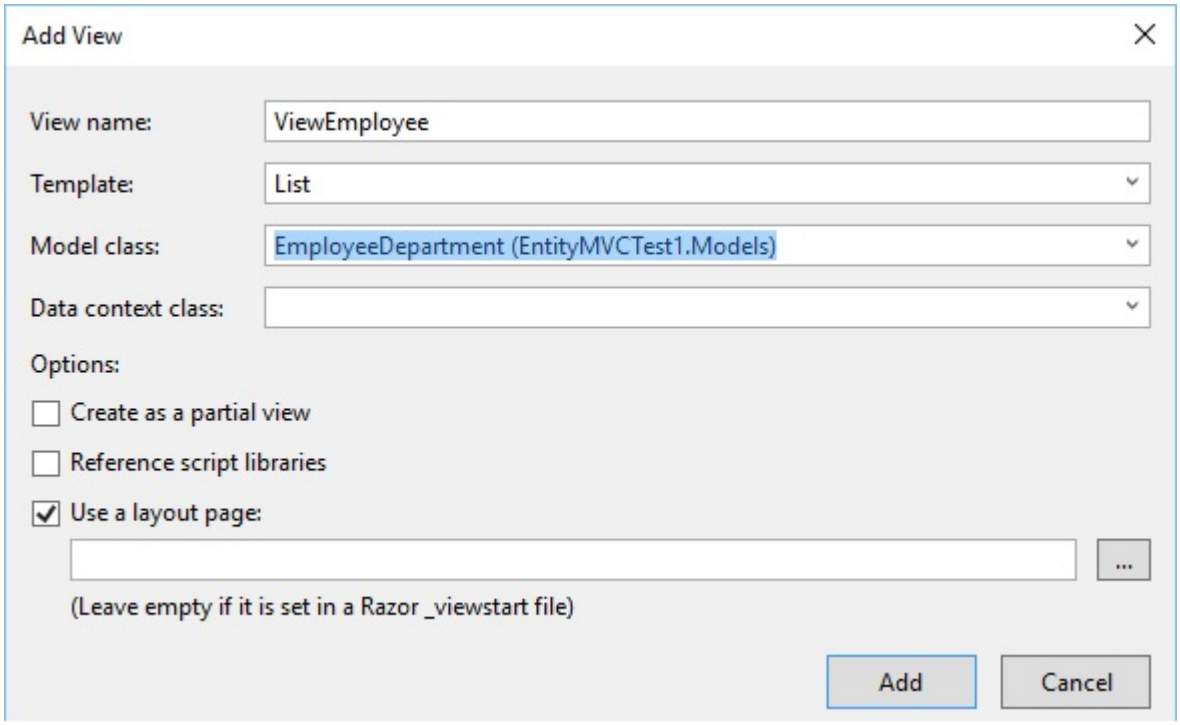
```
public class EmployeeController : Controller
    {
     // GET: Employee
     public ActionResult Index()
     {
         return View();
     }
        public ViewResult ViewEmployee(int DepartmentID)
        {
             EntityTestEntities context = new EntityTestEntities();
             EmployeeDepartment empDepartment = new EmployeeDepartment();

            empDepartment.MenuItems = (from employee in context.Employees
                                       join department in context.Departments
```

```
                                    on employee.Dep_Id equals department.ID
                                    where employee.Dep_Id == DepartmentID
                                    select new EmployeeDepartment()
                                   {
                                        FirstName=employee.FirstName,
                                       LastName=employee.LastName,
                                       departmentName=department.Location,
                                       departmentLocation=department.Name

                                    }).ToList();
            return View(empDepartment);
        }
    }
```

The controller `EmployeeController` uses the inner join query to retrieve the employees of a certain `department`, the search uses the `Id` of the `department` that is passed using URL and received by the action `ViewEmployee`. After that, a view is created for the `ViewEmployee`, that is of type `List` and the `Model` class is `ViewModel` that is `DepartmentEmployee` class.



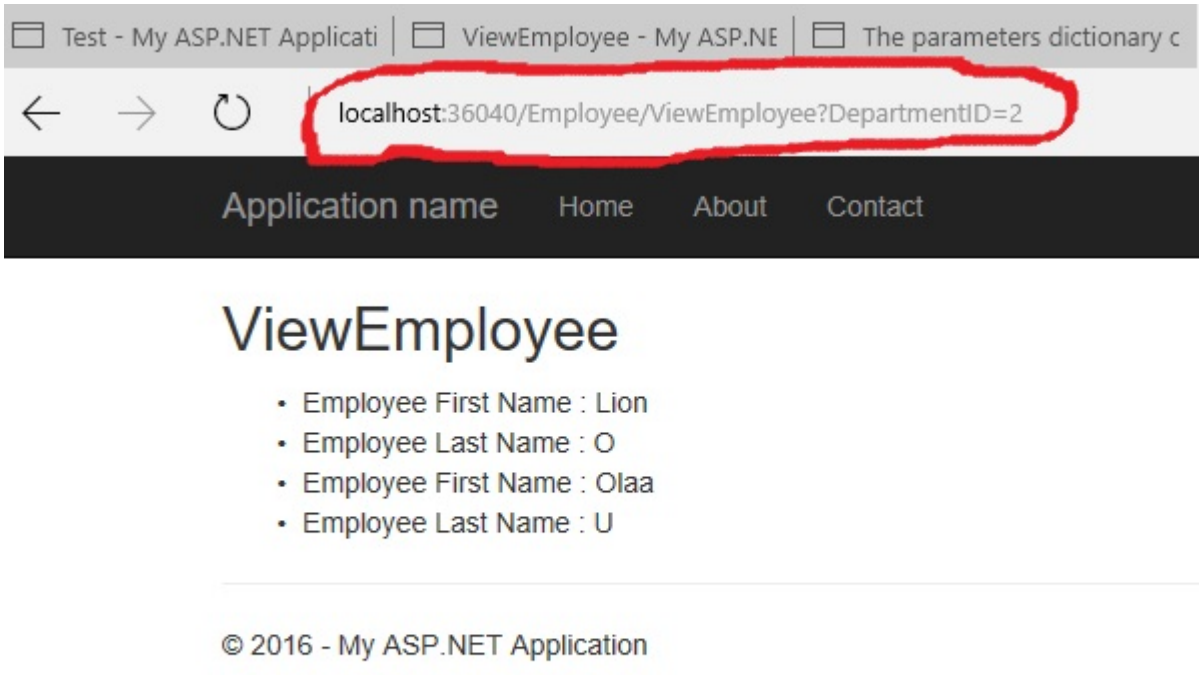

Hide Copy Code

```
    @model EntityMVCTest1.Models.EmployeeDepartment
    @{
     ViewBag.Title = "ViewEmployee";
    }

<h2>ViewEmployee</h2>

<ul>
    <li>@foreach (var item in Model.MenuItems) {</li>
    <li>Employee First Name : @item.FirstName</li>
    <li>Employee Last Name : @item.LastName</li>
    <li>}</li>
</ul>

<p>The markup at ViewEmployee.cshtml starts by identifying the Model of the view,
that is the EmployeeDepartment.After that the Markup uses the object passed by the action of
controller to iterate the value through the foreach loop</p>
```




# License



# Share

TWITTER FACEBOOK

# About the Author

**OmarIsaid**
Software Developer (Junior)
Jordan

No Biography provided

# You may also be interested in...


A Solution Blueprint for DevOps

TransparentSplash Control

Multiple Models in a View in ASP.NET MVC 4 / MVC 5

A Beginner's Tutorial On Understanding and Implementing Dependency Injection in ASP.NET Core

SharpDOM, View Engine for ASP.NET MVC

How to Choose the Best Way to Pass Multiple Models in ASP.NET MVC


# Comments and Discussions



### Some thoughts...

**Ed Bouras 13-Dec-16 6:05**

Omar - thanks for writing this article. I understand this is a beginner's article, so I am offering these suggestions as a supplement to your information.

It's nice to provide a basic understanding of the relationship between the controller and view but for anyone who is a beginner there are a few simple techniques you can follow to help develop better habits from the start. Again, this is not a criticism of the article, just supplementary suggestions

1) Consider removing any data operations from the controller itself and place it into classes that have the sole responsibility for operating on the data. Best, place the data logic in a separate assembly so you are not tempted to run the LINQ directly from the web project (but that may be a little more advanced). I know it is tempting to put it in the controller- especially when the EF model is in the web project - but this is a habit that will quickly balloon out of control with even medium-sized applications and cause a maintenance and scalability nightmare down the road

2) Consider using a database view when just displaying data in a certain format (like your EmployeeDepartment class). That way you can simplify the LINQ down to just referencing something like context.vEmployeeDepartments (the DB view) instead of doing a join in LINQ and filling a .NET class disconnected from the DB context itself.

3) Always employ a "using" block when working with IDisposable objects like an EF database context (EntityTestEntities in this

code).[^using Statement]

Well, I lied... I do have one criticism of the article: Omar, you really should show how you use the model in the view to display the first and last name information. If a beginner is following along with you that is one glaring omission that will cause frustration.

Sign In · View Thread

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin