How to pass model data from controller to view?

1) **ProjectName → Models →FileName.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace PartialViewWork.Models
{
    public class Employee
    {
        public int eno { get; set; }
        public String ename { get; set; }
        public int esal { get; set; }
    }
}
```

2) **ProjectName→Controllers→ControllerFile**

```
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using PartialViewWork.Models;

namespace PartialViewWork.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/

        public ActionResult Index()
        {
            var e1 = new Employee();
            e1.eno = 1001;
            e1.ename = "ramu";
            e1.esal = 59000;

            ViewBag.edet = e1;

            return View();
        }

    }
}
```

3) **Projectname →Views →Home →Index.cshtml**

```
@{
    Layout = null;
}
```

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
        <div>
            <h1> Model data from controller to view </h1>

            @ViewBag.edet.eno
            @ViewBag.edet.ename
            @ViewBag.edet.esal


        </div>
    <!--  or  -->
    @*  antoher method    *@
    <div>
        @{
          var v1 = ViewBag.edet;
         }
        <h1>
        @v1.eno
        @v1.ename
        @v1.esal
       </h1>

    </div>
</body>
</html>
```

Without using viewbag/viewData we can send model data from controller to view

Using object as parameter

---

1) **Model file same as above**
2) **Controller file without using ViewBag**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using PartialViewWork.Models;

namespace PartialViewWork.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/
```

```
        public ActionResult Index()
        {
            var e1 = new Employee();
            e1.eno = 1001;
            e1.ename = "ramu";
            e1.esal = 59000;

  //          ViewBag.edet = e1;

            return View(e1);    //object as paratmeter without using ViewBag
        }

    }
}
```

3) View File (Projectname →Views→Home→index.cshtml)

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
        <div>
            <h1> Model data from controller to view </h1>
            @Model.eno
            @Model.ename
            @Model.esal
            </div>
      <!--  or  -->
    @*  antoher method    *@

    @{
        var k = Model;
    }
    <h1>
        @k.eno
        @k.ename
        @k.esal
    </h1>
</body>
</html>
```

Passing Collection Of Objects:using viewBag & ViewData

Model file: (ProjectName (PassingObject)→Models →Emp.cs)

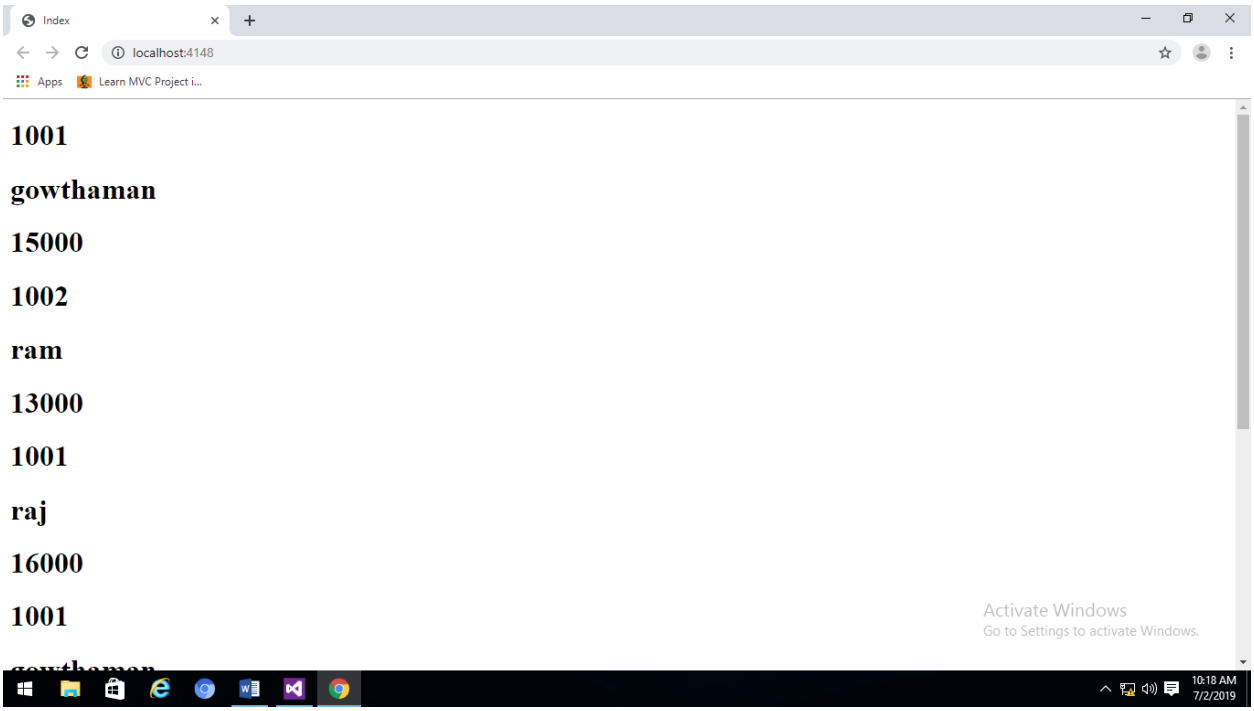
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace PassingObject.Models
{
    public class Emp
    {
        public int eno { get; set; }
        public String ename { get; set; }
        public int esal { get; set; }
    }
}
```

Controller File:(PassingObject→Controllers→HomeController)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using PassingObject.Models;
namespace PassingObject.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            var em1 = new Emp();
            em1.eno = 1001;
            em1.ename = "gowthaman";
            em1.esal = 15000;

            var em2 = new Emp();
            em2.eno = 1002;
            em2.ename = "ram";
            em2.esal = 13000;

            var em3 = new Emp();
            em3.eno = 1001;
            em3.ename = "raj";
            em3.esal = 16000;

            var allemp=new List<Emp>();
            allemp.Add(em1);
            allemp.Add(em2);
            allemp.Add(em3);

            ViewBag.retobj = allemp;
            ViewData["colobj"] = allemp;

            return View();
        }
    }
}
```

View File : PassingObject→Views→Home→Index.cshtml

```
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        @{
            var e = ViewBag.retobj;
        }

        @foreach(var k in e)
        {
            <h1>@k.eno</h1>
            <h1>@k.ename</h1>
            <h1>@k.esal</h1>
        }
@*=======================using view data dictionay================*@
        @{
            var e1 = ViewData["colobj"];
        }

        @foreach(var k1 in e1 as List<PassingObject.Models.Emp>)   // strong typing
        {
            <h1>@k1.eno</h1>
            <h1>@k1.ename</h1>
            <h1>@k1.esal</h1>
        }
    </div>
</body>
</html>
```


Index
localhost:4148
Apps
Learn MVC Project i...
1001
gowthaman
15000
1002
ram
13000
1001
raj
16000
1001
Activate Windows
Go to Settings to activate Windows.
10:18 AM
7/2/2019

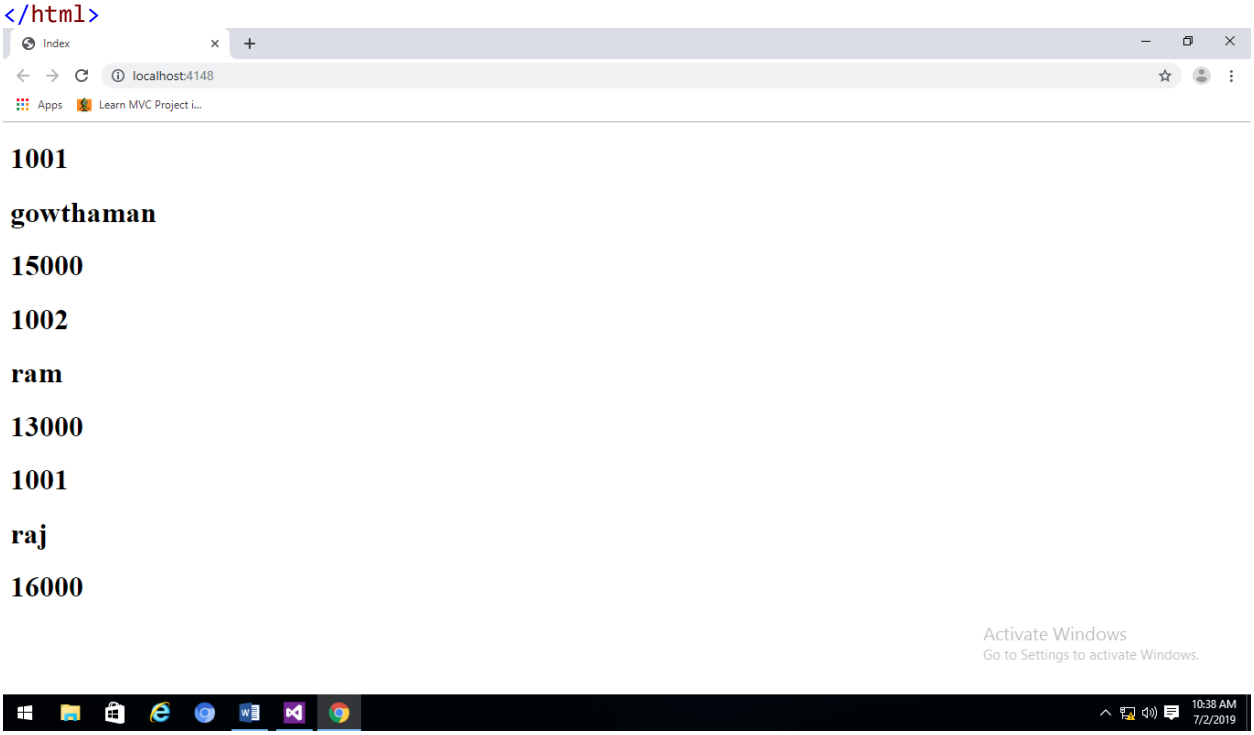
Passing Collection Of Objects as a parameter to the view.(don't use viewData & viewBag)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using PassingObject.Models;
namespace PassingObject.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            var em1 = new Emp();
            em1.eno = 1001;
            em1.ename = "gowthaman";
            em1.esal = 15000;

            var em2 = new Emp();
            em2.eno = 1002;
            em2.ename = "ram";
            em2.esal = 13000;

            var em3 = new Emp();
            em3.eno = 1001;
            em3.ename = "raj";
            em3.esal = 16000;

            var allemp=new List<Emp>();
            allemp.Add(em1);
            allemp.Add(em2);
            allemp.Add(em3);

          //  ViewBag.retobj = allemp;
         //   ViewData["colobj"] = allemp;

            return View(allemp); // passing object as a parameter to the view
        }
    }
}
```

View File:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
```

```
    <div>
        @{
            var e = Model;
        }

        @foreach(var k in e)
        {
            <h1>@k.eno</h1>
            <h1>@k.ename</h1>
            <h1>@k.esal</h1>
        }

    </div>
</body>
</html>
```



How to implement Strong Typing while passing single object to a view?

(passing model data from a controller to view using object as parameter)

Why we are using strong typing in view(ex: index.cshtml) ?

1) Enables Compile time checking of code (i.e: objectname.(show properties))

Ex:1

```
@{
    var e = Model as PassingObject.Models.Emp;  //strong typing(explicit cast)
}
@e.eno                  // automatically property will come when type .

@e.ename

@e.esal
```

Ex:2 using @model declaration in views

(To avoid explicit type casting of the model object)

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        @model PassingObject.Models.Emp


            <h1>@Model.eno</h1>
            <h1>@Model.ename</h1>
            <h1>@Model.esal</h1>

    </div>
</body>
</html>
```

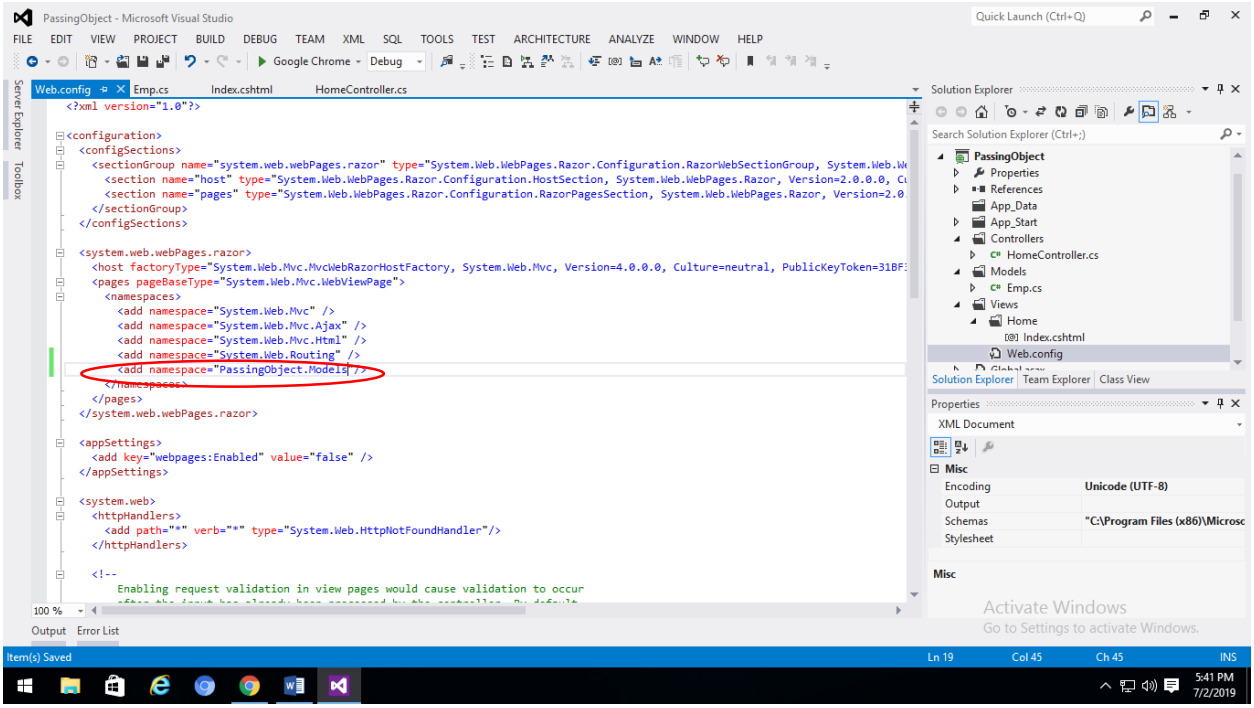
Ex:3 using @using declaration in views

(To avoid specifying a fully qualified name for the model)

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        @using PassingObject.Models
        @model Emp


            <h1>@Model.eno</h1>
            <h1>@Model.ename</h1>
            <h1>@Model.esal</h1>

    </div>
</body>
</html>
```

Ex:4 using namespace in views (To better approach)

Step 1: Views Directory → web.config file.




```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>

        @model Emp

            <h1>@Model.eno</h1>
            <h1>@Model.ename</h1>
            <h1>@Model.esal</h1>

    </div>
</body>
</html>
```

So far you have seen how to implement strong typing while passing a single objects,

Now we see pass collection of objects using @model

Syntax:(in view file)

@model IEnumerable<projectname.Models.ClassName>

Ex: 1(view file→ index.cshtml)

```
<body>
    <div>

        @model IEnumerable<PassingObject.Models.Emp>

        @{
            var ee = Model;
        }
        @foreach(var p in ee)
        {
            <h1>@p.eno</h1>
            <h1>@p.ename</h1>
            <h1>@p.esal</h1>
        }

    </div>
</body>
</html>
Or
        @
   var ee = Model as IEnumerable<Emp>;  // don't forgot web.config file to add namespace
        }
        @foreach(var p in ee)
        {
            <h1>@p.eno</h1>
            <h1>@p.ename</h1>
            <h1>@p.esal</h1>
        }
```

Ex:2 (collection of objects)-another method

```
        @{
            var e = Model as List<PassingObject.Models.Emp>;  // strong typeing
        }

        @foreach(var k in e )
        {
            @k.eno    // whenever user type @k. (automatically show properties(eno) at compile time)
```

```
            @k.ename
            @k.esal
        }
```

Controller file should be

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using PassingObject.Models;
namespace PassingObject.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            var em1 = new Emp();
            em1.eno = 1001;
            em1.ename = "gowthaman";
            em1.esal = 15000;


            var em2 = new Emp();
            em2.eno = 1002;
            em2.ename = "ram";
            em2.esal = 13000;

            var em3 = new Emp();
            em3.eno = 1001;
            em3.ename = "raj";
            em3.esal = 16000;

            var allemp=new List<Emp>();
            allemp.Add(em1);
            allemp.Add(em2);
            allemp.Add(em3);

            return View(allemp); // passing collection of object as a parameter to the
view
        }
    }
}
```