Way to Use Multiple Models in a view in ASP.NET MVC

For ASP.NET MVC beginner, it is very common problems that they faces in programming world: That how they can use multiple Models in their ASP.NET MVC application's view. Some of beginners is familiar with some of way to use multiple Models in an ASP.NET MVC application. But, May be they are not know about all the way to use Multiple Models in ASP.NET MVC. Here, in this article I am going to share all the possible ways to use multiple Models in a view. I am assuming that you will familiar with C# and ASP.NET MVC. If you are new to C# and ASP.NET MVC, you can look for different articles to learn ASP.NET MVC. Still, if you will face any issue you can put comment .below article in case you need any detail

:There are several ways to use multiple models in a view in ASP.NET MVC application, Following are the list

ViewModel .1

PartialView .2

ViewData .3

ViewBag .4

TempData .5

Tuple .6

.Let's start this article, we have following Models, which we will use to show data at our View

```
public class Employee
  {
      public string EmpCode { get; set; }
      public string EmpName { get; set; }
      public int DepCode { get; set; }
      public List<course> Courses { get; set; }
  }

  public class Department
  {
      public int DepCode { get; set; }
      public string DepName { get; set; }
  }

  public class Course
  {
      public int CourseCode { get; set; }
      public string CourseName { get; set; }
  }
```

For only demo purpose, we will create some sample data, which is hard coded in a class file under the Models folder named as Data.cs file, which will have the implementation of methods to get hardcoded data for .application in order to keep it convenient

.Following is the code for that we have to get all the temporary Data from Data.cs file

```
public class Data
    {
        public List<course> GetAllCourse()
        {
            return new List<course>{
                new Course{CourseCode=1, CourseName="B.Tech"},
                new Course{CourseCode=2, CourseName="M.C.A."},
                new Course{CourseCode=3, CourseName="B.C.A."},
                new Course{CourseCode=4, CourseName="M.Tech"}
            };
        }

        public List<department> GetAllDepartment()
        {
            return new List<department>
            {
                new Department{ DepCode=1,DepName="Computer Science & Engg."},
                new Department{ DepCode=2,DepName="Electronics"},
                new Department{ DepCode=3,DepName="Mechanical Engg."},
                new Department{ DepCode=4,DepName="Information Technology"}
            };
        }

        public List<employee> GetAllEmployee()
        {
            return new List<employee>
            {
                new Employee{ EmpCode="emp1",EmpName="Suresh Kumar",Courses=
                    new List<course>{new Course{CourseCode=2, CourseName="M.C.A."},
                                     new Course{CourseCode=3, CourseName="B.C.A."}
                    },
                    DepCode=1
                },
                new Employee{ EmpCode="emp2",EmpName="Rajesh Kumar",Courses=
                    new List<course>{new Course{CourseCode=1, CourseName="B.Tech"},
                                     new Course{CourseCode=4, CourseName="M.Tech"}
                    },
                    DepCode=4
                },
                new Employee{ EmpCode="emp3",EmpName="Mahesh Kumar",Courses=
                    new List<course>{new Course{CourseCode=1, CourseName="B.Tech"},
                                     new Course{CourseCode=4, CourseName="M.Tech"}
                    },
                    DepCode=3
                },
                new Employee{ EmpCode="emp4",EmpName="Kamlesh Kumar",Courses=
                    new List<course>{new Course{CourseCode=2, CourseName="M.C.A."},
                                     new Course{CourseCode=4, CourseName="M.Tech"}
                    },
                    DepCode=2
                }
            };
        }
    }
```

Now, we have a view in Views/Home folder named Employee.cshtml, to show the information of employee, we are create links at index.cshtml file to access the employee list. Each links will access by one of a way to use .multiple Models. Here are the sample code

```
<h3>We suggest the following :</h3>
@Html.ActionLink("Employee list using ViewModel"," ViewModelDemo")
@Html.ActionLink("Employee list using PartialView", "PartialViewDemo")
@Html.ActionLink("Employee list using ViewData", "ViewDataDemo")
@Html.ActionLink("Employee list using ViewBag", "ViewBagDemo")
@Html.ActionLink("Employee list using TempData", "TempDataDemo")
@Html.ActionLink("Employee list using Tuple", "TupleDemo")
```

.So far, we have created basic code which we will be using in all scenarios. Further, we will learn each scenario one by one

Using Multiple Models at View using ViewModel

.ViewModel is a pattern that can be used to have multiple models as a single class. It contains properties of entities exactly need to be used in a view

We have three models (classes) named as Employee, Course and Department. All are different models but we need to use all three models in a view. We will create a ViewModel by adding a new folder called ViewModels and :in class file called EmployeeModels.cs and write the following code as shown below

```
public class ViewModelDemo
    {
        public List<Employee> allEmployees { get; set; }
        public List<Course> allCourses { get; set; }
        public List<Department> allDepartments { get; set; }
    }
```

:In the HomeController, add the following code

```
public ActionResult ViewModelDemo()
        {
            Data _dummyData = new Data();
            ViewModelDemo vmDemo = new ViewModelDemo();
            vmDemo.allCourses = _dummyData.GetAllCourse();
            vmDemo.allDepartments = _dummyData.GetAllDepartment();
            vmDemo.allEmployees = _dummyData.GetAllEmployee();
            return View(vmDemo);
        }
```

ViewModelDemo is an action method that will return a view with Model ViewModelDemo which has lists of allCourses, Employees and Departments. We need to add a View to Show these data, to do that: Right click on ViewModelDemo to add a view is called ViewModelDemo. ViewModelDemo.cshtml view will use ViewModelDemo as Model as shown below: this code is written in ViewModelDemo.cshtml

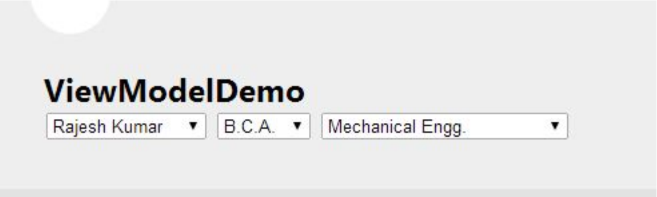
```
@model MvcApplication4.Models.ViewModelDemo
@{
    ViewBag.Title = "ViewModelDemo";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>ViewModelDemo</h2>
<select>
@*Iterating Employee ViewModel *@
@foreach (var item in Model.allEmployees)
    {
        <option>@item.EmpName</option>
    }
</select>

<select>
@*Iterating Courses ViewModel *@
@foreach (var item in Model.allCourses)
    {
        <option>@item.CourseName</option>
    }
</select>

<select>
@*Iterating Department ViewModel *@
@foreach (var item in Model.allDepartments)
    {
        <option>@item.DepName</option>
    }
</select>
```

.Here are the demo screen shot how it will look like



ViewModelDemo

Rajesh Kumar ▾ B.C.A. ▾ Mechanical Engg. ▾

Using Multiple Models at View using PartialView

.Partial view is like UserControl in ASP.NET. It is used where you need to share the same code (Razor and HTML code) in more than one view

.In Home controller, create a new Action Method called PartialViewDemo and create a will return a view having the list of all Employees only. Following are the Code to create this Action

```
public ActionResult PartialViewDemo()
        {
            Data _dummyData = new Data();
            List<Employee> allEmployees = _dummyData.GetAllEmployee();
            return View(allEmployees);
        }
```

.Add a view named as PartialViewDemo. Here are the sample code for PartialViewDemo.cshtml file

```
<h2>PartialViewDemo Show Courses</h2>
<select id="ddlEmployeeCourse">
@*Iterating Employee ViewModel *@

@foreach (var emp in Model)
{
        <option value="@emp.EmpCode">@emp.EmpName</option>
}

</select>

        <h4>Courses Of Selected Employeee</h4>
        <div id="CoursesForEmp">
    </div>


<h2>PartialViewDemo Show Departments</h2>
<select id="ddlEmployeeDept">
@*Iterating Employee ViewModel *@

@foreach (var emp in Model)
{
        <option value="@emp.EmpCode">@emp.EmpName</option>
}

</select>


        <h4>Departments Of Selected Employeee</h4>
        <div id="DepartmentForEmp">

    </div>
```

.Now, we will try to create some javascript function and some more Action and partial View to use other model at this View. Following are the sample code to create javascript function

```
function getDepartmentTable(selectedEmpCode) {
    $.ajax({
        // Get Department PartialView
        url: "/Home/DepartmentPartialDemo",
        type: 'GET',
        data: { EmpCode: selectedEmpCode },
        success: function (data) {
            jQuery("#DepartmentForEmp").html(data);
        },
        error: function (error) {
            alert("Error: Please try again.");
        }
    });
}


function getCourseTable(selectedEmpCode) {
    $.ajax({
        // Get Course PartialView
        url: "/Home/CoursePartialDemo",
        type: 'GET',
        data: { EmpCode: selectedEmpCode },
        success: function (data) {
            jQuery("#CoursesForEmp").html(data);
        },
        error: function (error) {
            alert("Error: Please try again.");
        }
    });
}


jQuery(document).ready(function () {
    jQuery("#ddlEmployeeCourse").change(function (index) {
        var selectedEmpCode = $(this).val();
        getCourseTable(selectedEmpCode);
    });
    jQuery("#ddlEmployeeDept").change(function (index) {
        var selectedEmpCode = $(this).val();
        getDepartmentTable(selectedEmpCode);
    });

});
```

As in above we have called two Action Method by javascript, So we need to create that in HomeController. Following are sample code

```
public ActionResult DepartmentPartialDemo(string EmpCode)
        {
            Data _dummyData = new Data();

            var _depCode = _dummyData.GetAllEmployee().FindAll(emp => emp.EmpCode == EmpCode).FirstOrDefault();


            IEnumerable<Department> cDepartment = _dummyData.GetAllDepartment().FindAll(dep => dep.DepCode == _depCode.DepCode);
            return PartialView("DepartmentPartialView", cDepartment);
        }


public ActionResult CoursePartialDemo(string EmpCode)
        {
            Data _dummyData = new Data();

            Employee _Emp = _dummyData.GetAllEmployee().FindAll(emp => emp.EmpCode == EmpCode).FirstOrDefault();

             List<Course> cCourses = new List<Course>();
            foreach(Course courses in _Emp.Courses){
                 cCourses.Add( _dummyData.GetAllCourse().FindAll(course => course.CourseCode == courses.CourseCode).FirstOrDefault());
            }

            return PartialView("CoursePartialView", cCourses);
        }
```

As, we have defined two new PartialView in above Action Methods, So, we need to create these Partial View in to the Shared folder by right clicking on DepartmentPartialDemo action method, give it name as :DepartmentPartialView. And similarly, for CoursePartialDemo Action Method create another partial View CoursePartialView. Following, are the sample code of both partial views

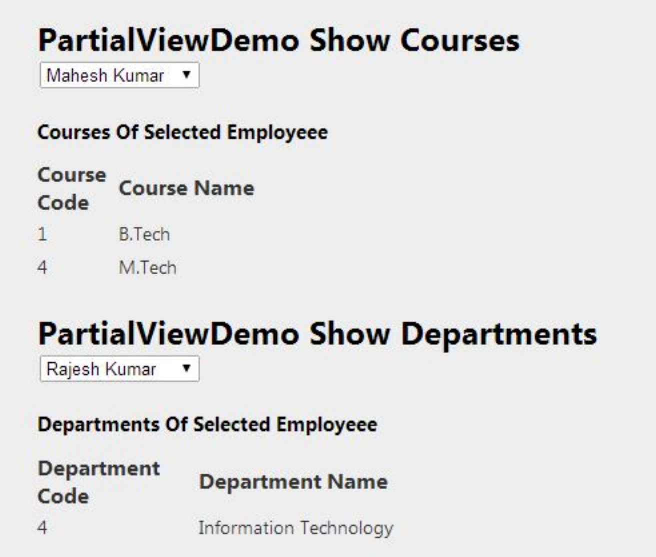
In DepartmentPartialView.cshtml

```
@model  IEnumerable <MvcApplication4.Models.Department>
 <table>
         <tr>
         <th  style="width:120px;">Department Code</th>
         <th>Department Name </th>
         </tr>
    @foreach (var item in Model)
    {
          <tr>
             <td>@item.DepCode </td>
             <td>@item.DepName </td>
         </tr>
    }
 </table>
```

In CoursePartialView.cshtml

```
@model  IEnumerable <MvcApplication4.Models.Course>
 <table>
         <tr>
         <th  style="width:60px;">Course Code</th>
         <th>Course Name </th>
         </tr>
    @foreach (var item in Model)
    {
          <tr>
             <td>@item.CourseCode </td>
             <td>@item.CourseName </td>
         </tr>
    }
 </table>
```

.Now, after running application the screen shot will looks like



PartialViewDemo Show Courses

Courses Of Selected Employeee

Course Code	Course Name
1	B.Tech
4	M.Tech

PartialViewDemo Show Departments

Departments Of Selected Employeee

Department Code	Department Name
4	Information Technology

Using Multiple Models at View using ViewData

The very important features and use of ViewData is used to pass data from a controller to a view. We can be define ViewData is a dictionary of objects that is a type of ViewDataDictionary class. In ASP.NET MVC, ViewData is defined as property in ControllerBase class. To Use the values stored in ViewData, It is require typecasting to their datatype in view. The values in ViewData are accessible using a key. Let's see in details how can we use .ViewData to pass multiple Models in a View in ASP.NET MVC

.First, we will create a new Action in HomeController named ViewDataDemo, Following are the sample code

```
public ActionResult ViewDataDemo()
        {
            Data _dummyData = new Data();
            ViewData["Employees"] = _dummyData.GetAllEmployee();
            ViewData["Departments"] = _dummyData.GetAllDepartment();
            ViewData["Courses"] = _dummyData.GetAllCourse();
            return View();
        }
```

.Here ViewDataDemo action method will be passing all three models to its view using ViewData. Now Add a View named ViewDataDemo and place following code in ViewDataDemo.cs.html

```
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ViewDataDemo</title>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
    <script>
        $(document).ready(function(){
            ShowData();
        })

        function ShowData(){
            getEmployeeTable();
            getCourseTable();
            getDepartmentTable();
        }

        //Create table to display Course detail
        function getCourseTable() {
            $("#CoursesForEmp").empty();
            $("#CoursesForEmp").append("<table id='tblCourses'><tr><th  style='width:120px;'>Course Code </th><th >Course Name </th> </tr> </table>");

            //Get all Courses with the help of model
            //(ViewData["Courses"]), and convert data into json format.
            var allCourses = @Html.Raw(Json.Encode(ViewData["Courses"]));

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

                $("#tblCourses").append
                        (" <tr> <td >"  +
                        allCourses[i].CourseCode
                            + " </td> <td >"+
                            allCourses[i].CourseName+" </td> </tr>");

            }
        }

        //Create table to display Employee detail
        function getEmployeeTable() {
            $("#EmpolyeeList").empty();
            $("#EmpolyeeList").append(" <table id='tblEmployee' > <tr> <th  style='width:120px;'> Employee Code </th> <th >Employee Name </th> </tr> </table>");

            //Get all Employees with the help of model
            //(ViewData["Employees"]), and convert data into json format.
            var allEmployees = @Html.Raw(Json.Encode(ViewData["Employees"]));

            for (var i = 0; i  < allEmployees.length; i++) {
                $("#tblEmployee").append(" <tr>" +
                "<td >"  + allEmployees[i].EmpCode +
                    " </td> <td >"+
            allEmployees[i].EmpName+" </td> </tr>");

            }
        }

        //Create table to display Department detail
        function getDepartmentTable() {
            $("#DepartmentForEmp").empty();
            $("#DepartmentForEmp").append(" <table id='tblDepartment' > <tr> <th  style='width:120px;'> Department Code </th> <th >Department Name </th> </tr> </table>");

            //Get all departments with the help of model
            //(ViewData["Departments"]), and convert data into json format.
            var allDepartments = @Html.Raw(Json.Encode(ViewData["Departments"]));

            for (var i = 0; i  < allDepartments.length; i++) {
                $("#tblDepartment").append(" <tr>" +
                        "<td >"  + allDepartments[i].DepCode +
                            " </td> <td >"+
                    allDepartments[i].DepName+" </td> </tr>");

            }
        }


    </script>
</head>
<body>
    <div>

  <h4>Courses List Multiple Models using ViewData</h4>
        <div id="CoursesForEmp">
        </div>

        <h4>Employees List Multiple Models using ViewData</h4>
        <div id="EmpolyeeList">
        </div>

        <h4>Departments List Multiple Models using ViewData</h4>
        <div id="DepartmentForEmp">

        </div>

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

Using Multiple Models at View using ViewBag

Like ViewData, ViewBag is also used to pass data from a controller to a view. It is a dynamic property which comes in ControllerBase class that is why it doesn’t require typecasting for datatype. To call multiple Models using :ViewBag Create an Action Method ViewBagDemo in HomeController, Following are the sample code

```
public ActionResult ViewBagDemo()
    {
        Data _dummyData = new Data();
        ViewBag.Employees = _dummyData.GetAllEmployee();
        ViewBag.Departments = _dummyData.GetAllDepartment();
        ViewBag.Courses = _dummyData.GetAllCourse();
        return View();
 }
```

:Here ViewBagDemo action method will be passing data to view (ViewBagDemo.cshtml) file using ViewBag. Place, following code at ViewBagDemo.cshtml to use Multiple Models at View using ViewBag

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ViewBagDemo</title>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
    <script>
        $(document).ready(function(){
            ShowData();
        })

        function ShowData(){
            getEmployeeTable();
            getCourseTable();
            getDepartmentTable();
        }

        //Create table to display Course detail
        function getCourseTable() {
            $("#CoursesForEmp").empty();
            $("#CoursesForEmp").append("<table id='tblCourses'><tr><th  style='width:120px;'>Course Code </th><th >Course Name </th> </tr> </table>");

            //Get all Courses with the help of model
            //(ViewData.Courses), and convert data into json format.
            var allCourses = @Html.Raw(Json.Encode(ViewBag.Courses));
            for (var i = 0; i  < allCourses.length; i++) {

                $("#tblCourses").append
                        (" <tr> <td >"  +
                        allCourses[i].CourseCode
                            + " </td> <td >"+
                            allCourses[i].CourseName+" </td> </tr>");

            }
        }

        //Create table to display Employee detail
        function getEmployeeTable() {
            $("#EmpolyeeList").empty();
            $("#EmpolyeeList").append(" <table id='tblEmployee' > <tr> <th  style='width:120px;'> Employee Code </th> <th >Employee Name </th> </tr> </table>");

            //Get all Employees with the help of model
            //(ViewData.Employees), and convert data into json format.
            var allEmployees = @Html.Raw(Json.Encode(ViewBag.Employees));

            for (var i = 0; i  < allEmployees.length; i++) {
                $("#tblEmployee").append(" <tr>" +
                "<td >"  + allEmployees[i].EmpCode +
                    " </td> <td >"+
            allEmployees[i].EmpName+" </td> </tr>");

            }
        }

        //Create table to display Department detail
        function getDepartmentTable() {
            $("#DepartmentForEmp").empty();
            $("#DepartmentForEmp").append(" <table id='tblDepartment' > <tr> <th  style='width:120px;'> Department Code </th> <th >Department Name </th> </tr> </table>");

            //Get all departments with the help of model
            //(ViewData.Departments), and convert data into json format.
            var allDepartments = @Html.Raw(Json.Encode(ViewBag.Departments));

            for (var i = 0; i  < allDepartments.length; i++) {
                $("#tblDepartment").append(" <tr>" +
                        "<td >"  + allDepartments[i].DepCode +
                            " </td> <td >"+
                    allDepartments[i].DepName+" </td> </tr>");

            }
        }
```

```
    </script>
</head>
<body>
    <div>
        <h4>Courses List Multiple Models using ViewBag</h4>
        <div id="CoursesForEmp">
        </div>

        <h4>Employees List Multiple Models using ViewBag</h4>
        <div id="EmpolyeeList">
        </div>

        <h4>Departments List Multiple Models using ViewBag</h4>
        <div id="DepartmentForEmp">

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

Using Multiple Models at View using TempData

Like ViewData, TempData is a dictionary of objects that is a type of TempDataDictionary class. The data is stored temporary, So we need to call Keep method to keep the data in TempData. . TempData is similar to ViewData but the difference is that it allow us to send and receive the data from one controller to another controller and from one action to another action. It's all about possible because it internally uses session variables. TempData is .defined as property in ControllerBase class. Values stored in TempData require typecasting to datatype in view. The values in TempData are accessible using a key

:To use Multiple Models using TempData, we will create an Action in HomeController named TempDataDemo. Following are the sample code

```
public ActionResult TempDataDemo()
        {
            Data _dummyData = new Data();
            // TempData demo uses dummyData to get List<Employees> only one time
            // for subsequent request to get List<Employees> it will use TempData
            TempData["Employees"] = _dummyData.GetAllEmployee();

            // This will keep Employees data untill next request is served
            TempData.Keep("Employees");

            TempData["Departments"] = _dummyData.GetAllDepartment();
            TempData.Keep("Departments");

            TempData["Courses"] = _dummyData.GetAllCourse();
            TempData.Keep("Courses");

            return View();
        }
```

:Now, we will create a View for this Action named TempDataDemo.cshtml. Following are the sample code to use Multiple Models at View via TempData

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>TempDataDemo</title>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
    <script>
        $(document).ready(function(){
            ShowData();
        })

        function ShowData(){
            getEmployeeTable();
            getCourseTable();
            getDepartmentTable();
        }

        //Create table to display Course detail
        function getCourseTable() {
            $("#CoursesForEmp").empty();
            $("#CoursesForEmp").append("<table id='tblCourses'><tr><th  style='width:120px;'>Course Code </th><th >Course Name </th> </tr> </table>");

            //Get all Courses with the help of model
            //(TempData["Courses"]), and convert data into json format.
            var allCourses = @Html.Raw(Json.Encode(TempData["Courses"]));

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

                $("#tblCourses").append
                        (" <tr> <td >"  +
                        allCourses[i].CourseCode
                            + " </td> <td >"+
                            allCourses[i].CourseName+" </td> </tr>");

            }
        }

        //Create table to display Employee detail
        function getEmployeeTable() {
            $("#EmpolyeeList").empty();
            $("#EmpolyeeList").append(" <table id='tblEmployee' > <tr> <th  style='width:120px;'> Employee Code </th> <th >Employee Name </th> </tr> </table>");

            //Get all Employees with the help of model
            //(TempData["Employees"]), and convert data into json format.
            var allEmployees = @Html.Raw(Json.Encode(TempData["Employees"]));

            for (var i = 0; i  < allEmployees.length; i++) {
                $("#tblEmployee").append(" <tr>" +
                "<td >"  + allEmployees[i].EmpCode +
                    " </td> <td >"+
            allEmployees[i].EmpName+" </td> </tr>");

            }
        }

        //Create table to display Department detail
        function getDepartmentTable() {
            $("#DepartmentForEmp").empty();
            $("#DepartmentForEmp").append(" <table id='tblDepartment' > <tr> <th  style='width:120px;'> Department Code </th> <th >Department Name </th> </tr> </table>");

            //Get all departments with the help of model
            //(TempData["Departments"]), and convert data into json format.
            var allDepartments = @Html.Raw(Json.Encode(TempData["Departments"]));

            for (var i = 0; i  < allDepartments.length; i++) {
                $("#tblDepartment").append(" <tr>" +
                        "<td >"  + allDepartments[i].DepCode +
                            " </td> <td >"+
                    allDepartments[i].DepName+" </td> </tr>");

            }
        }


    </script>
</head>
<body>
    <div>
        <h4>Courses List Multiple Models using TempData</h4>
        <div id="CoursesForEmp">
        </div>

        <h4>Employees List Multiple Models using TempData</h4>
        <div id="EmpolyeeList">
        </div>

        <h4>Departments List Multiple Models using TempData</h4>
        <div id="DepartmentForEmp">

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

Using Multiple Models in View using Tuple

A tuple is a data structure that has a specific number and sequence of elements. An example of a tuple is a data structure with three elements (known as a 3-tuple or triple) that is used to store an identifier such as a person's name in the first element, a year in the second element, and the person's income for that year in the third element. The .NET Framework directly supports tuples with one to seven elements. In addition, you can .create tuples of eight or more elements by nesting tuple objects in the Rest property of a Tuple<T1, T2, T3, T4, T5, T6, T7, TRest> object

:Passing multiple Models in View using Tuple is very easy, to do this let’s create an Action in HomeController. Following are sample code

```
public ActionResult TupleDemo()
        {
            Data _dummyData = new Data();
            var allToupleData=new Tuple<List<Employee> ,List<Department> ,List<Course> (
                _dummyData.GetAllEmployee(),_dummyData.GetAllDepartment(),_dummyData.GetAllCourse());
            return View(allToupleData);
        }
```

.Here, we had defined a tuple which is having lists of Employees, Departments and Courses. After fill dummy data, we are passing this tuple to the View

:Now, we need to add a View named as TupleDemo. Following are the sample code to use Multiple Models using Tuple at a View

```
@using MvcApplication4.Models;
@model  Tuple<List<Employee>, List<Department>, List<Course>>
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ToupleDemo</title>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
    <script>
        $(document).ready(function(){
            ShowData();
        })

        function ShowData(){
            getEmployeeTable();
            getCourseTable();
            getDepartmentTable();
        }

        //Create table to display Course detail
        function getCourseTable() {
            $("#CoursesForEmp").empty();
            $("#CoursesForEmp").append("<table id='tblCourses'><tr><th  style='width:120px;'>Course Code </th><th >Course Name </th> </tr> </table>");

```

```
          //Get all Courses with the help of model
          //(Model.Item3), and convert data into json format.
          var allCourses = @Html.Raw(Json.Encode(Model.Item3));

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

              $("#tblCourses").append
                      (" <tr> <td >"  +
                      allCourses[i].CourseCode
                        + " </td> <td >"+
                        allCourses[i].CourseName+" </td> </tr>");

          }
        }

        //Create table to display Employee detail
        function getEmployeeTable() {
          $("#EmpolyeeList").empty();
          $("#EmpolyeeList").append(" <table id='tblEmployee' > <tr> <th  style='width:120px;'> Employee Code </th> <th >Employee Name </th> </tr> </table>");

          //Get all Employees with the help of model
          //(Model.Item1), and convert data into json format.
          var allEmployees = @Html.Raw(Json.Encode(Model.Item1));

          for (var i = 0; i  < allEmployees.length; i++) {
              $("#tblEmployee").append(" <tr>" +
              "<td >"  + allEmployees[i].EmpCode +
                " </td> <td >"+
          allEmployees[i].EmpName+" </td> </tr>");

          }
        }

        //Create table to display Department detail
        function getDepartmentTable() {
          $("#DepartmentForEmp").empty();
          $("#DepartmentForEmp").append(" <table id='tblDepartment' > <tr> <th  style='width:120px;'> Department Code </th> <th >Department Name </th> </tr> </table>");

          //Get all departments with the help of model
          //(Model.Item2), and convert data into json format.
          var allDepartments = @Html.Raw(Json.Encode(Model.Item2));

          for (var i = 0; i  < allDepartments.length; i++) {
              $("#tblDepartment").append(" <tr>" +
                      "<td >"  + allDepartments[i].DepCode +
                        " </td> <td >"+
                  allDepartments[i].DepName+" </td> </tr>");

          }
        }


    </script>
</head>
<body>
    <div>
        <h4>Courses List Multiple Models using Tuple</h4>
        <div id="CoursesForEmp">
        </div>

        <h4>Employees List Multiple Models using Tuple</h4>
        <div id="EmpolyeeList">
        </div>

        <h4>Departments List Multiple Models using Tuple</h4>
        <div id="DepartmentForEmp">

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

Summary:I above article, we learned about passing multiple models in a view using various ways. Please, read another article How to Choose the Best Way to Pass Multiple Models. I hope you will like this article the demo .source code is available here to download

!...Keen to hear from you

.If you have any questions related to what's mentioned in the article or need help with any issue, ask it, I would love to here from you. Please MakeUseOf Contact and i will be more than happy to help

Tweet Linkedin • Google • :Connect with the author

About the author

Anil Sharma is Chief Editor of dotnet-stuff.com. He's a software professional and loves to work with Microsoft .Net. He's usually writes articles about .Net related technologies and here to shares his experiences, personal notes, Tutorials, Examples, Problems & Solutions, Code Snippets, Reference Manual and Resources with !!!...C#, Asp.Net, Linq , Ajax, MVC, Entity Framework, WCF, SQL Server, jQuery, Visual Studio and much more

Follow @anilrightnow

Previous ←

→ Next

مواضيع شبيهة

Way to Use Multiple Models in a view in ASP.NET MVC

... For ASP.NET MVC beginner, it is very common problems that they faces in programming world: That how

Understanding Best Way to Use Multiple Models in ASP.NET MVC

...It is depend upon our requirement to use multiple Models in ASP.NET MVC application, in this article

Introduction to Razor Pages in ASP.NET Core

...Razor Pages is a new feature of ASP.NET Core MVC that makes coding page-focused scenarios easier and

أضف تعليق

تعليق | استعراض

اسم *

البريد الإلكتروني *

تعليق *

□ أبلغني عند اضافة تعليقات جديدة

حفظ التعليقات

Recent Articles

- Get Table information (Column Name, Data Type, Size) in SQL Server
- How to remove query string from URL using JQuery
- Drop Function if exist – SQL Server
- ?What is .NET Standard
- ?What is ASP.NET Core
- ?What is .NET Core
- Understanding File uploads in ASP.NET Core
- Preventing Cross-Site Request Forgery (XSRF/CSRF) Attacks in ASP.NET Core
- Introduction to Razor Pages in ASP.NET Core
- Getting started with Razor Pages in ASP.NET Core

Browse By Category

4 Dependency Injection 8 C Sharp 3 ASP.NET Web Page 12 ASP.NET vNext 29 ASP.NET MVC 9 ASP.NET Core 12 ASP.NET 20 Angular JS 1 Ajax 2 ADO.NET 6 Net Framework.

3 Web API 1 Visual Studio 3 SQL Server 4 OOPS 3 LINQ 2 Knockout JS 3 jQuery 2 JavaScript 1 HTML5 2 Entity Framework 5 Design Patterns

Most Popular Articles

- Way to Use Multiple Models in a view in ASP.NET MVC
- How to Render different Layout in ASP.NET MVC
- Understanding File uploads in ASP.NET Core
- Understanding URL Rewriting and URL Attribute Routing in ASP.NET MVC (MVC5) with Examples
- CRUD Operations using ADO.Net and C# in ASP.Net
- Understanding Loose Coupling and Tight Coupling
- How to remove query string from URL using JQuery
- Understand ASPX and Razor View Engine
- Preventing Cross-Site Request Forgery (XSRF/CSRF) Attacks in ASP.NET Core
- How to Persist Data with TempData