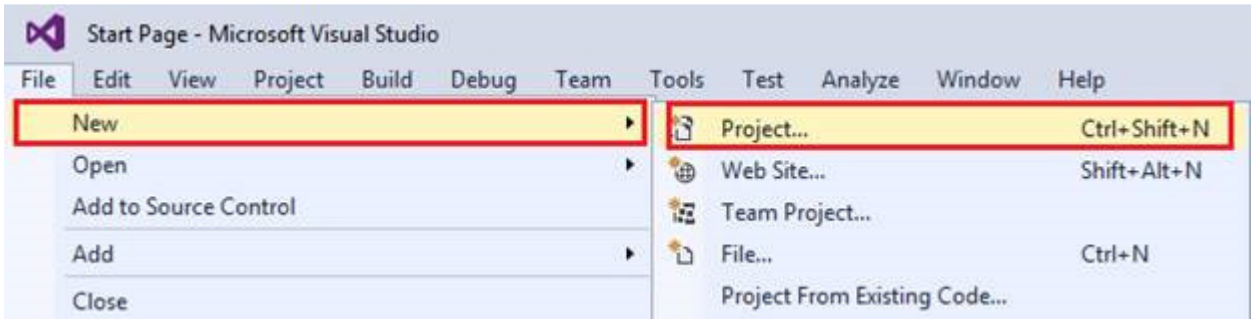
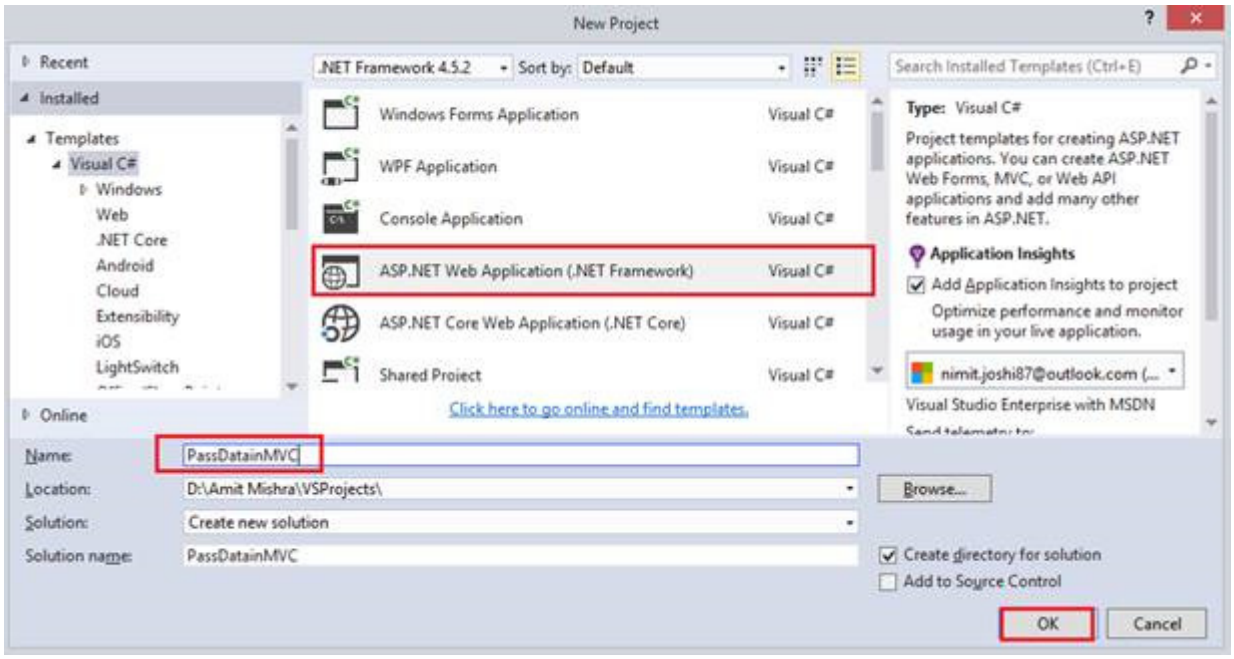
ASP.NET MVC - Passing Data from Controller to View

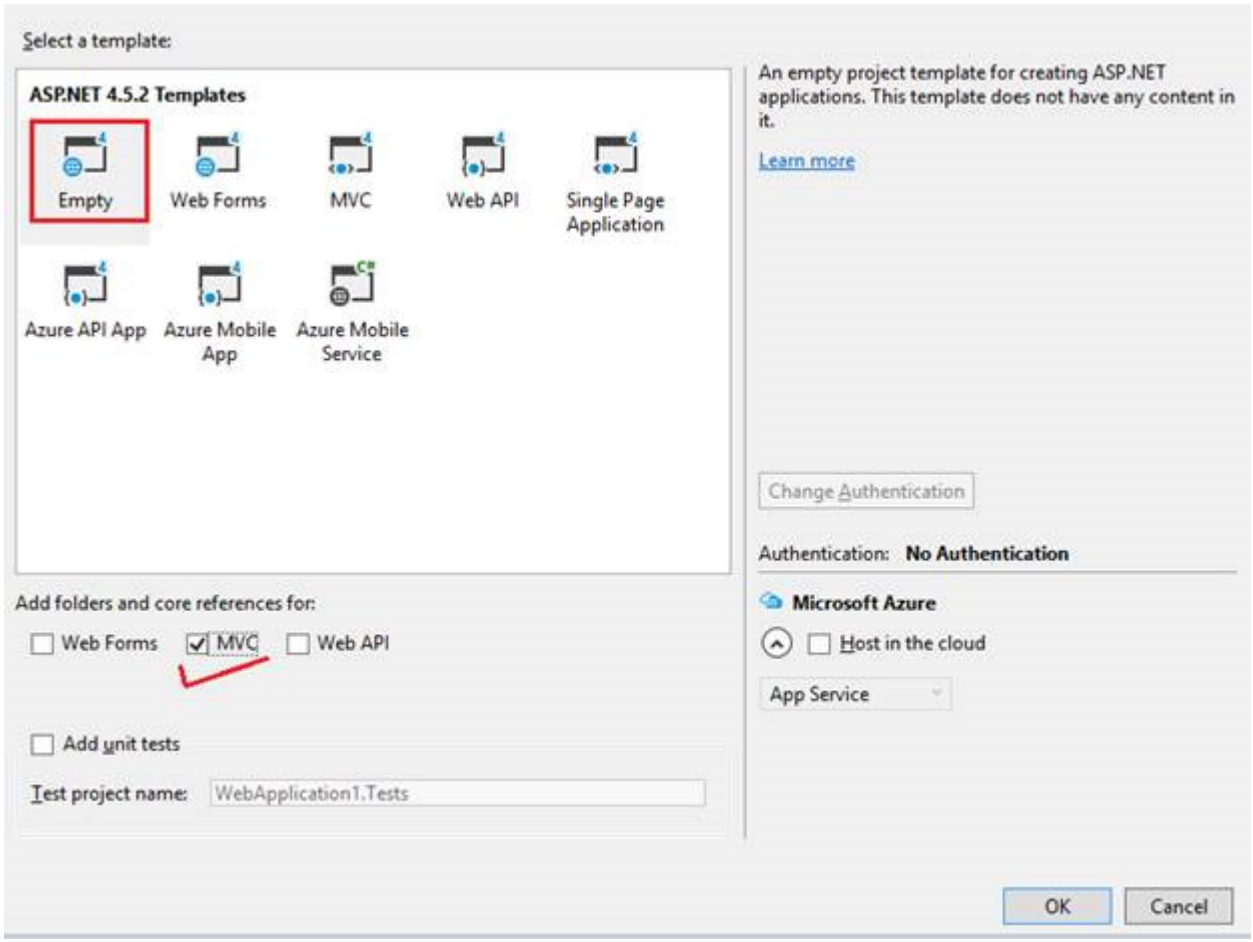
There are many ways to pass Model class data to the View, using a Controller class. In this article, I'm showing how to pass the data from Controller to View by using a simple example. Thus, let's start and flip to your Visual Studio 2015.



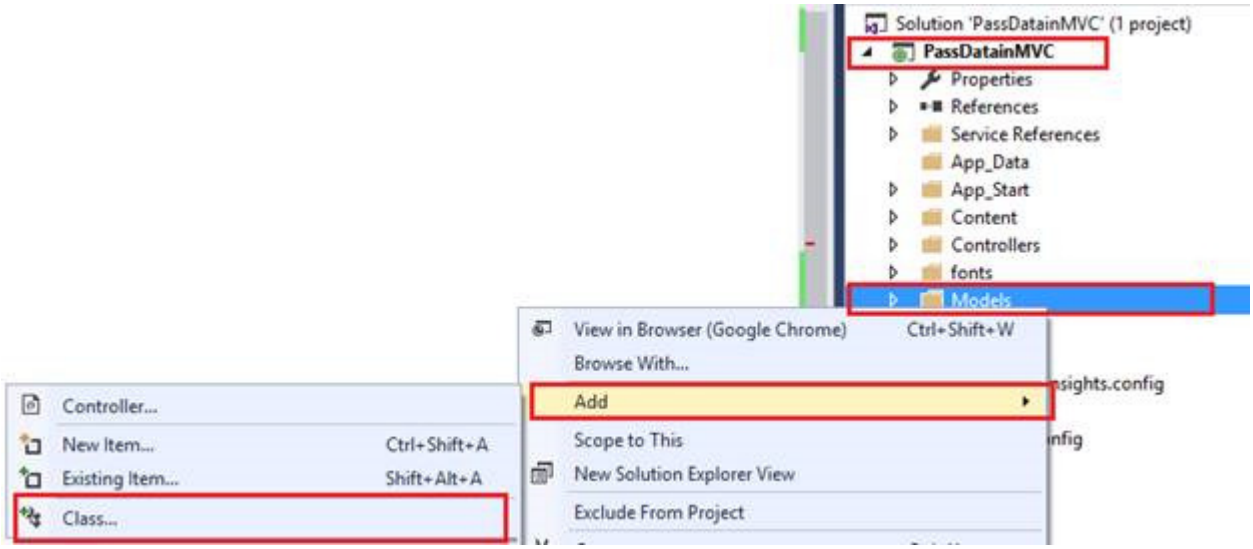
Give a name to your empty ASP.NET Web Application and click OK button.



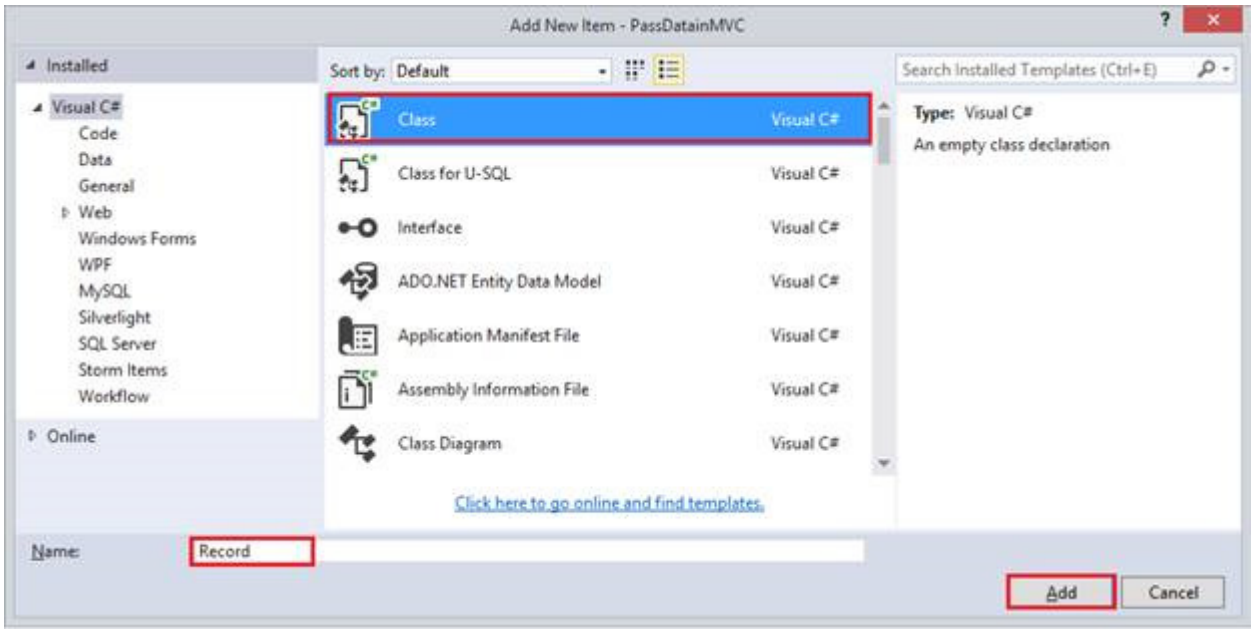
I don't want to use a full MVC based solution. The article is based on a simple example, so just select an Empty template.



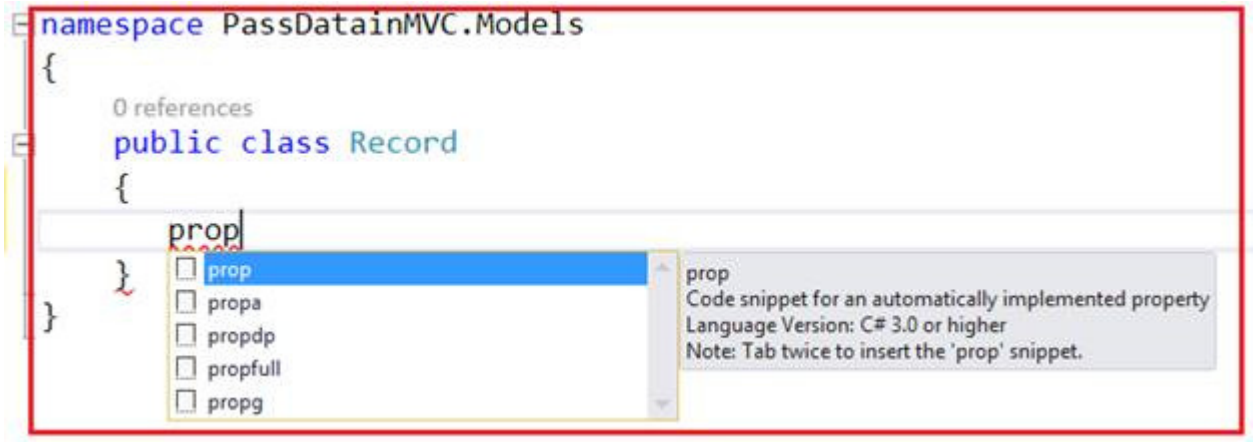
Add a Model class by right clicking on Models folder under Solution.



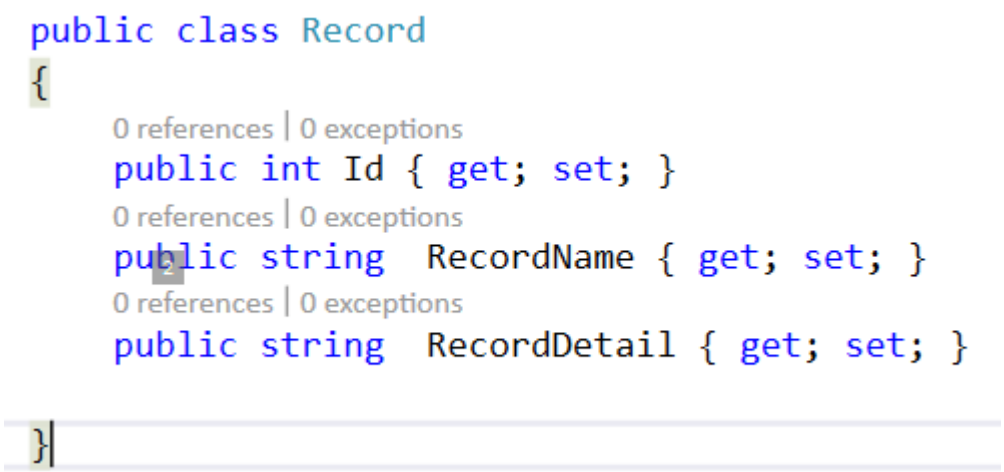
Select class and give a name to your class.



Just type a few words of the code snippet for automatically implementing property and press tab button available on the keyboard two times, which is a good practice, when the code is long.



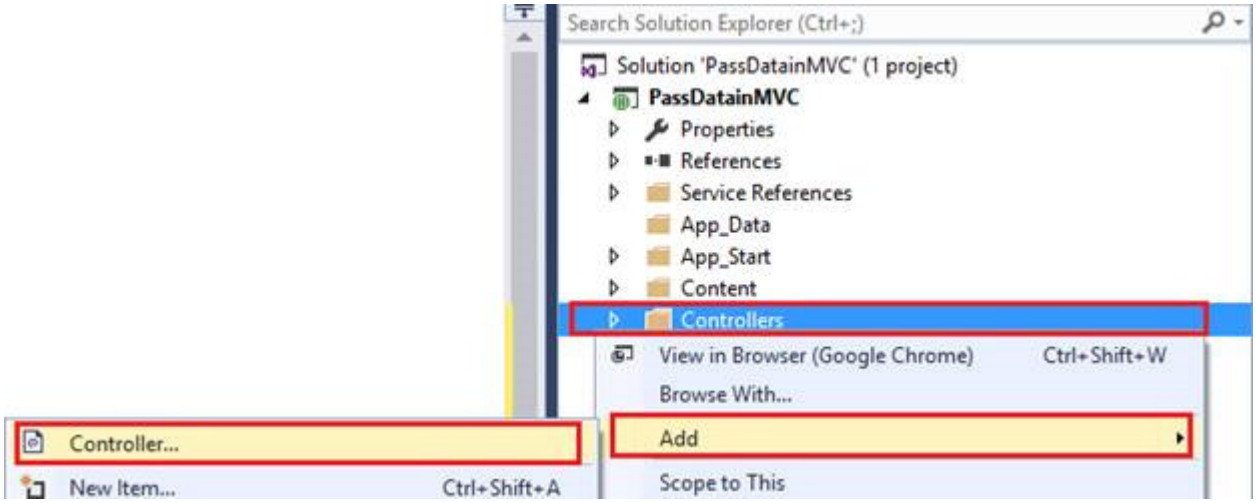
The public Record model class contains only three properties, which we want to pass to Views.



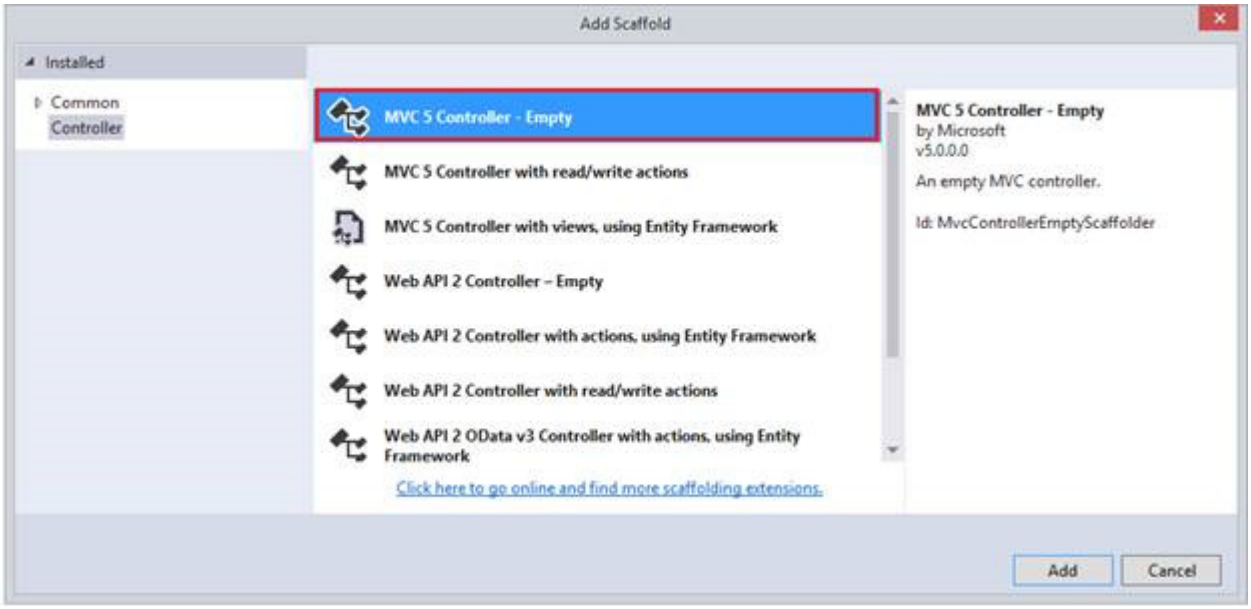

```
1.  public class Record
2.      {
3.          public int Id { get; set; }
4.          public string  RecordName { get; set; }
5.          public string  RecordDetail { get; set; }
```

```
6.
7.      }
```

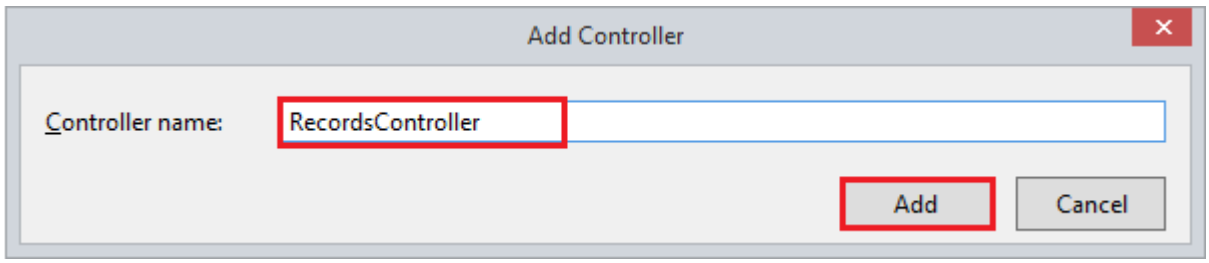
Add a controller class by right clicking on Controllers folder.



Select an Empty Controller from the list of Scaffold.



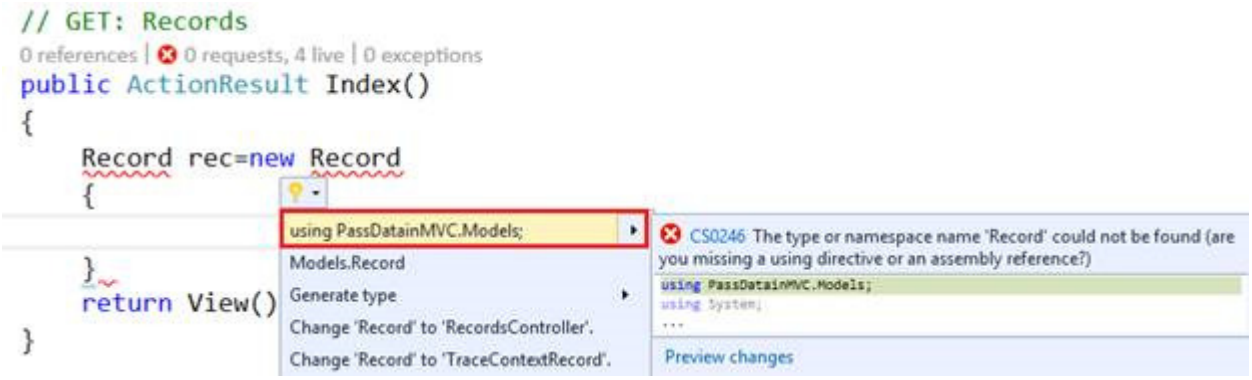
Give a meaningful name to your Controller.



Controller is ready now. By using ViewBag, we can pass the data from Controller to View.

ViewBag – ViewBag gets the dynamic view data dictionary.

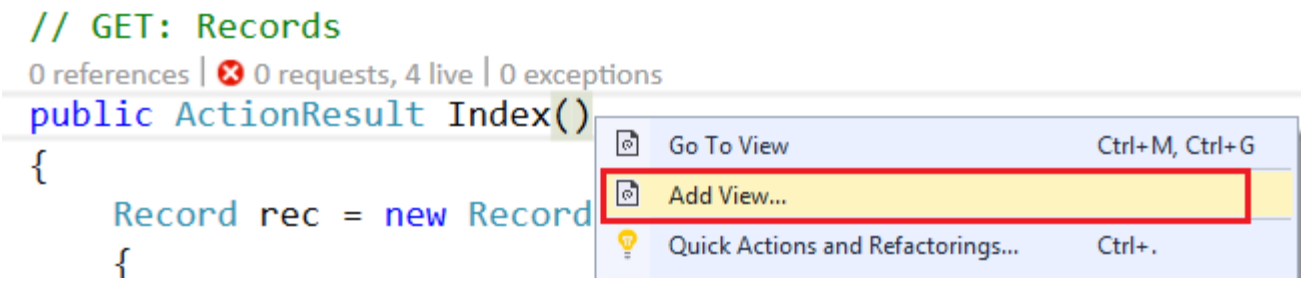
Create an object of your Model class and resolve it by namespace.



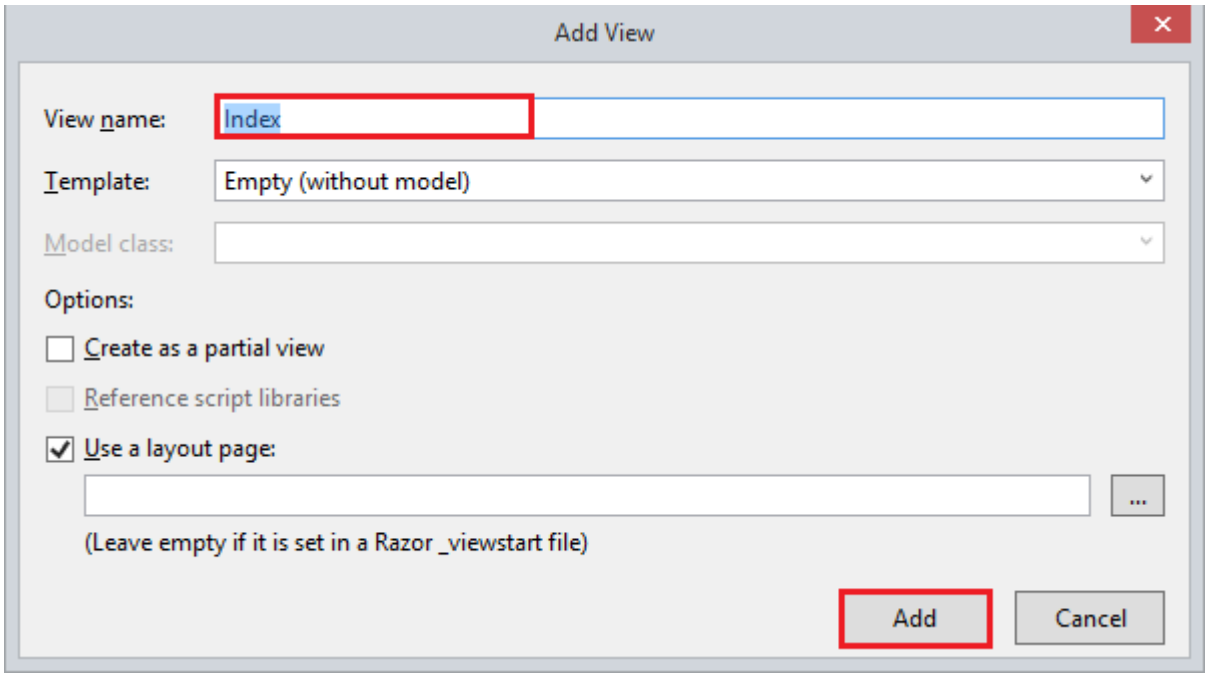
I have created an object of model class and assigned it to in ViewBag data dictionary. Assign the value for the property based on model class.

```
1.  public ActionResult Index()
2.          {
3.              Record rec = new Record
4.              {
5.                  Id = 101,
6.                  RecordName = "Bouchers",
7.                  RecordDetail = "The basic stocks"
8.              };
9.              ViewBag.Message = rec;
10.             return View();
11.         }
```

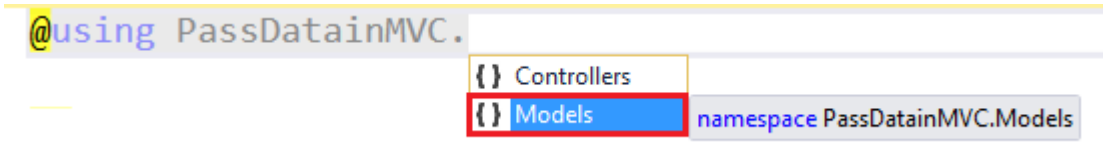
Add a View for an Index Action by right clicking on it.



Give a name to it and select Add button.



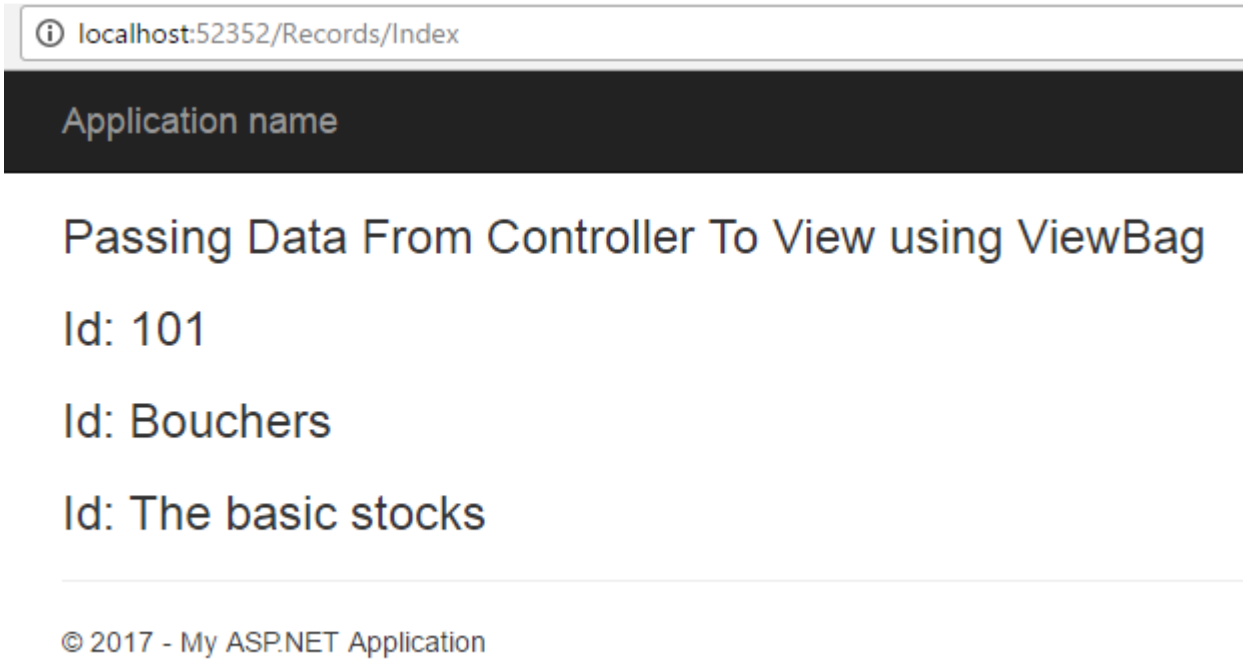
First of all import the model class.



Assign viewbag into a variable and all the properties will be in place, using the variable and Razor block.

```
@using PassDatainMVC.Models

@{
    ViewBag.Title = "Index";
}

<h3>Passing Data From Controller to View using ViewBag</h3>
@{
    var data = ViewBag.Message;
}
<h3>Id: @data.Id</h3>
<h3>RecordName: @data.RecordName</h3>
<h3>RecordDetail: @data.RecordDetail</h3>
```

Build and run your Application. You will get ViewBag Data.



The other way of passing the data from Controller to View is ViewData. Also, a dictionary type object is similar to ViewBag. There are no huge changes in Controller and ViewData contains key-value pairs.

```
0 references
public class RecordsController : Controller
{
    //GET: Records
    0 references | 0 requests, 4 live | 0 exceptions
    public ActionResult Index()
    {
        Record rec = new Record
        {
            Id = 101,
            RecordName = "Bouchers",
            RecordDetail = "The basic stocks"
        };
        ViewData["Message"] = rec;
        return View();
    }
}
```

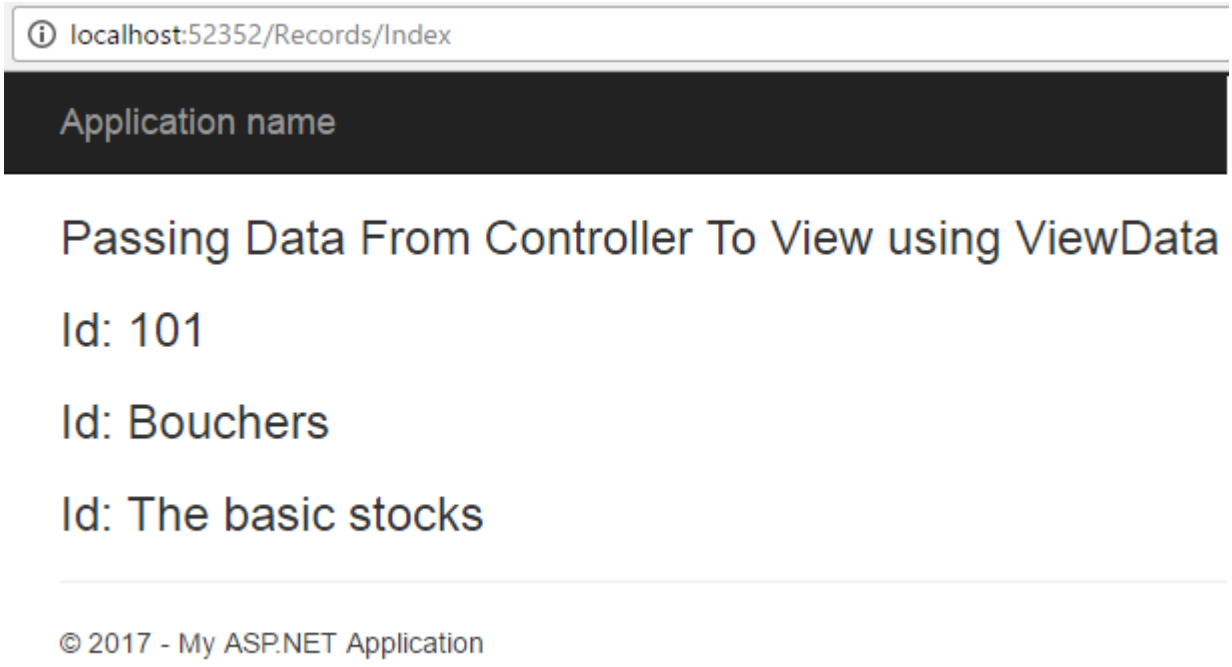
```
1.  public ActionResult Index()
2.          {
3.              Record rec = new Record
4.              {
5.                  Id = 101,
6.                  RecordName = "Bouchers",
7.                  RecordDetail = "The basic stocks"
8.              };
9.              ViewData["Message"] = rec;
10.             return View();
11.         }
```

Access your model class when you are using ViewData, as shown below.

```
@using PassDatainMVC.Models
@{
    ViewBag.Title = "Index";
}

<h3>Passing Data From Controller To View using ViewData</h3>
@{
    var data = (Record)ViewData["Message"];
}
<h3>Id: @data.Id</h3>
<h3>RecordName: @data.RecordName</h3>
<h3>RecordDetail: @data.RecordDetail</h3>
```

```
1.  @using PassDatainMVC.Models
2.  @{
3.      ViewBag.Title = "Index";
4.  }
5.  <h3>Passing Data From Controller To View using ViewData</h3>
6.  @{
7.      var data = (Record)ViewData["Message"];
8.  }
9.  <h3>Id: @data.Id</h3>
10. <h3>RecordName: @data.RecordName</h3>
11. <h3>RecordDetail: @data.RecordDetail</h3>
```

Save and Run your project. You will get the expected result from the ViewData.



The other way of passing the data from Controller to View can be by passing an object of the model class to the View. Erase the code of ViewData and pass the object of model class in return view.

```
0 references
public class RecordsController : Controller
{
    //GET: Records
    0 references | 0 requests, 4 live | 0 exceptions
    public ActionResult Index()
    {
        Record rec = new Record
        {
            Id = 101,
            RecordName = "Bouchers",
            RecordDetail = "The basic stocks"
        };

        return View(rec);
    }
}
```

```
1.  public ActionResult Index()
2.          {
3.              Record rec = new Record
4.              {
5.                  Id = 101,
6.                  RecordName = "Bouchers",
7.                  RecordDetail = "The basic stocks"
8.              };
9.              return View(rec);
10.         }
```

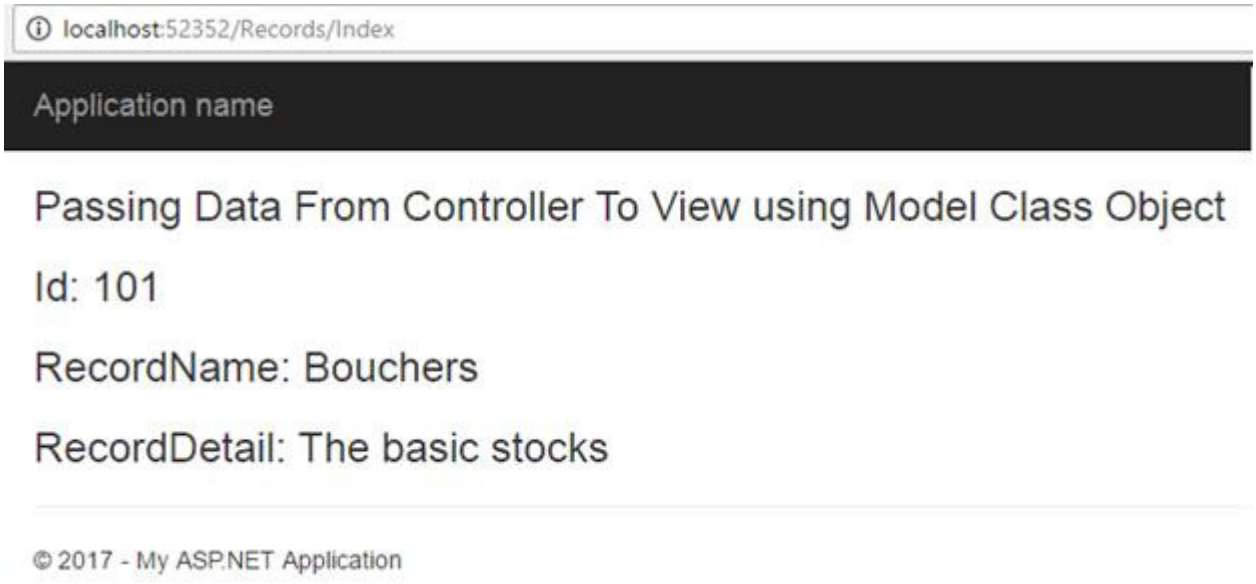
Import the binding object of model class at the top of Index View and access the properties by @Model.

```
@using PassDatainMVC.Models

@model PassDatainMVC.Models.Record

@{
    ViewBag.Title = "Index";
}

<h3>Passing Data From Controller To View using Model Class Object</h3>

<h3>Id: @Model.Id</h3>
<h3>RecordName: @Model.RecordName</h3>
<h3>RecordDetail: @Model.RecordDetail</h3>
```

```
1.  @using PassDatainMVC.Models
2.  @model PassDatainMVC.Models.Record
3.  @{
4.      ViewBag.Title = "Index";
5.  }
6.  <h3>Passing Data From Controller To View using Model Class Object</h3>
```

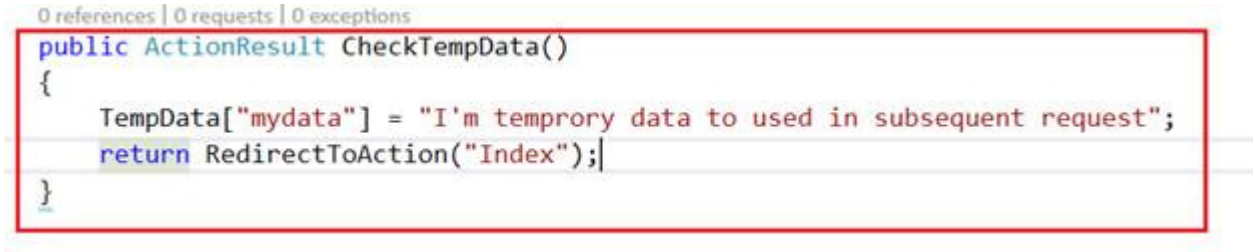
```
7.
8.  <h3>Id: @Model.Id</h3>
9.  <h3>RecordName: @Model.RecordName</h3>
10. <h3>RecordDetail: @Model.RecordDetail</h3>
```

The expected result from the Model class Object is given.



The other way of passing the data is TempData, which is responsible for storing temporary data.

After completing the subsequent request; the TempData will be cleared out. TempData is derived from TempDataDictinory. It is very useful to transfer non-sensitive data from one action method to another action method. As you can see in the screenshot given below, I have created an Action method for checking TempData. TempData transfers the data from the CheckTempData method to Index method.

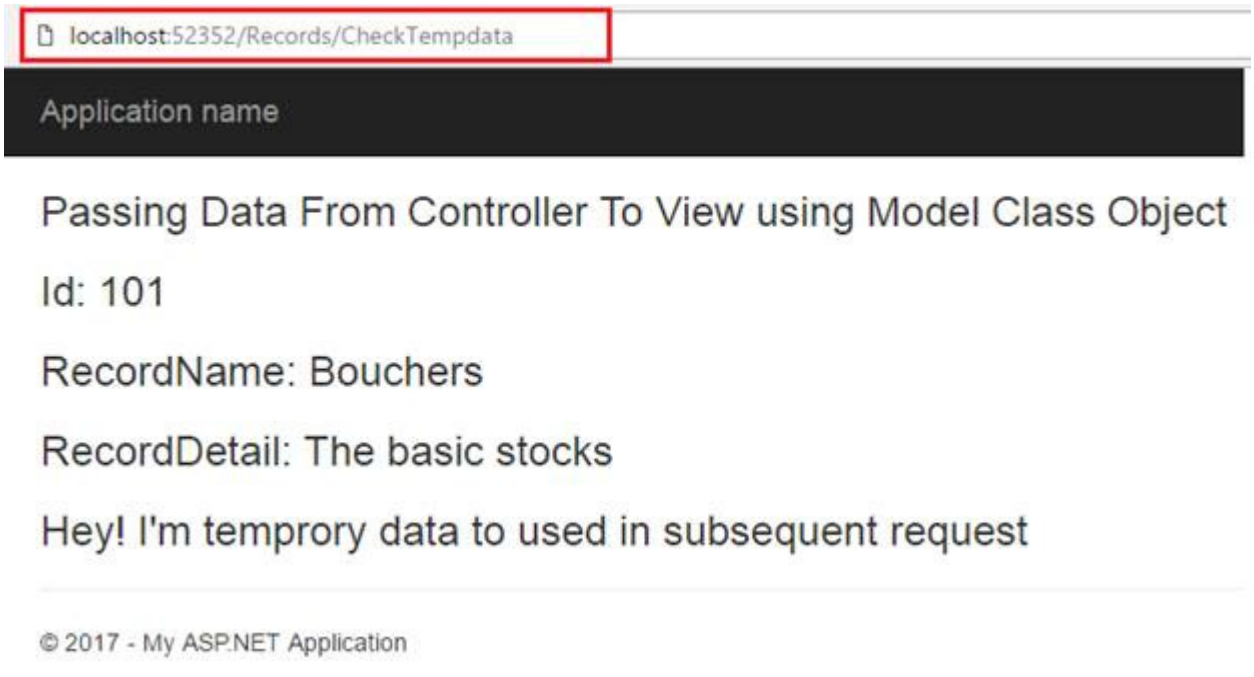



```
1.  public ActionResult CheckTempData()
2.          {
3.              TempData["data"] = "I'm temprory data to used in subsequent request";
4.              return RedirectToAction("Index");
5.          }
```

Acccess TempData in Index.Chtml view is given.

```
1.  <h3>Hey! @TempData["data"]</h3>
```

Run the Application and call the respective action method. TempData uses an internal session to store the data.


localhost:52352/Records/CheckTempdata
Application name
Passing Data From Controller To View using Model Class Object
Id: 101
RecordName: Bouchers
RecordDetail: The basic stocks
Hey! I'm temprory data to used in subsequent request
© 2017 - My ASP.NET Application