اليوم الثاني

Model :

في الطبقات (NTier) كان لدينا ثلاث طبقات واحدة تمثل العرض (Presentation) والثانية للتعامل مع قاعدة البيانات (Data Access) والثالثة كانت وسطية بين طبقة العرض وطبقة البيانات تسمى بطبقة (Business Logic) . الطبقة الاخيرة تستلم الطلب من العرض ثم ترسله الى طبقة البيانات التي بدورها تنفذه وترجع النتائج لطبقة الوسطى ومن ثم ترسل هذه الطبقة النتائج الى طبقة العرض .نفس هذه الطبقة في MVC تسمى بـــ (Model) حيث ان البيانات التي تنفذ في الكونترولر تخزن بالمودل ويقرأها الفيو منه فرقها عن طبقة العمل (Business Logic) أنها تتعامل مباشرة مع قاعدة البيانات .

قبل التعمق أكثر بالمودل , هناك شي مهم رأينا باليوم الاول ولم نتكلم عنه وهو Razor , حيث عند انشاء المشروع كان لدينا خيارين اما (ASPX) أو (Razor) . هذين الخيارين هما محركات العرض , بمعنى آخر هما عبارة عن طريقة لكتابة الكود الذي يتنفذ على السيرفر (مثل كود C#) داخل الصفحة .

في الـ Aspx كنا نكتب كود السي شارب داخل الصفحة بالشكل التالي :

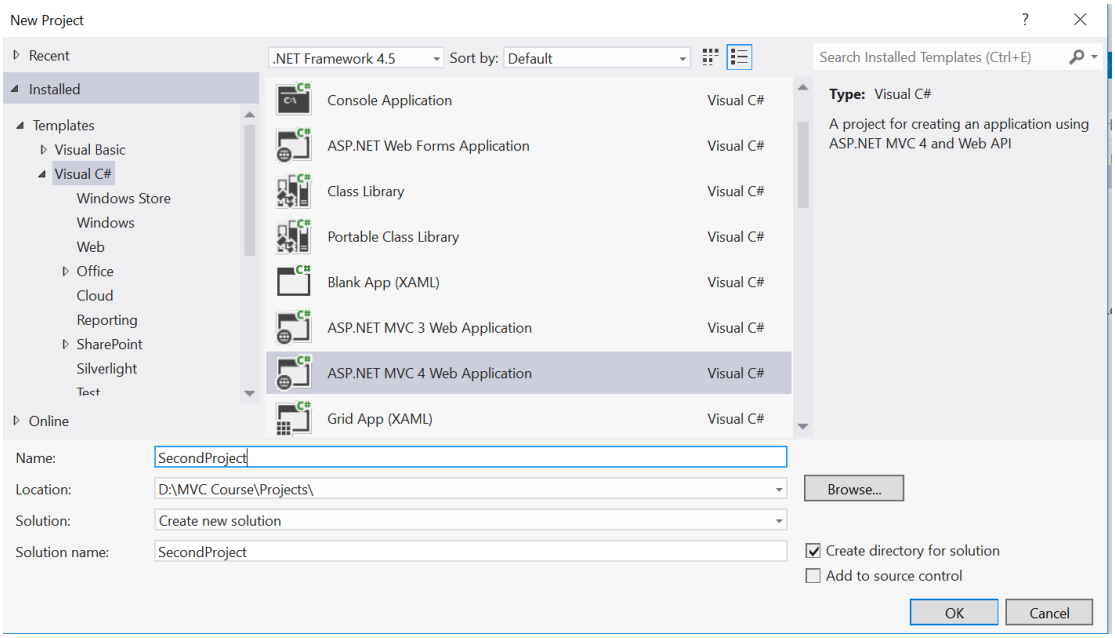
```
<% Code %>
```

أما في Razor نكتبه بالشكل التالي :

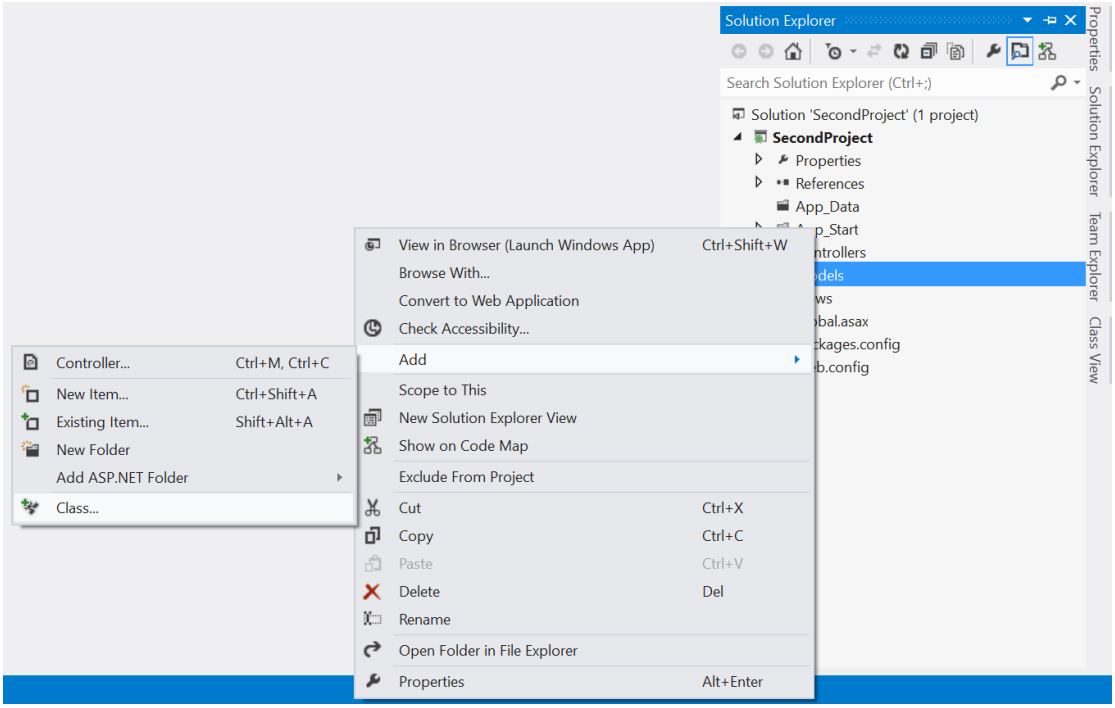
```
@{ Code }
```

دعونا نأخذ مثال عن المودل :

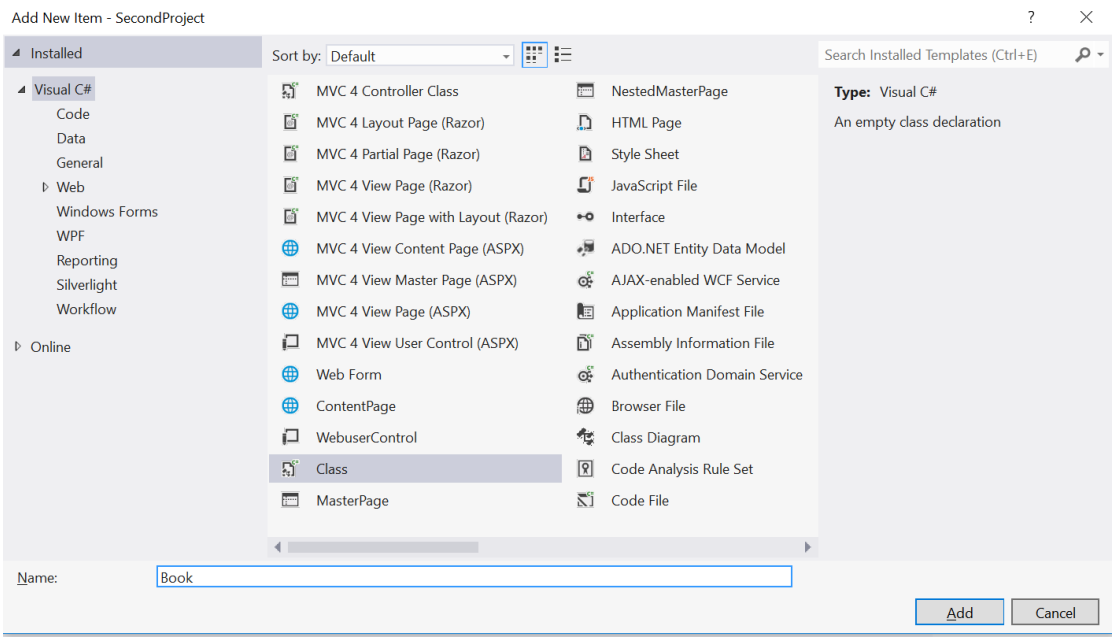
ننشئ مشروع جديد بأسم (SecondProject) :



ثم نضيف مودل جديد من خلال اضافة كلاس للمجلد (Models) :



نسميه (Book) :



لنضيف فيه بعض الخصائص :

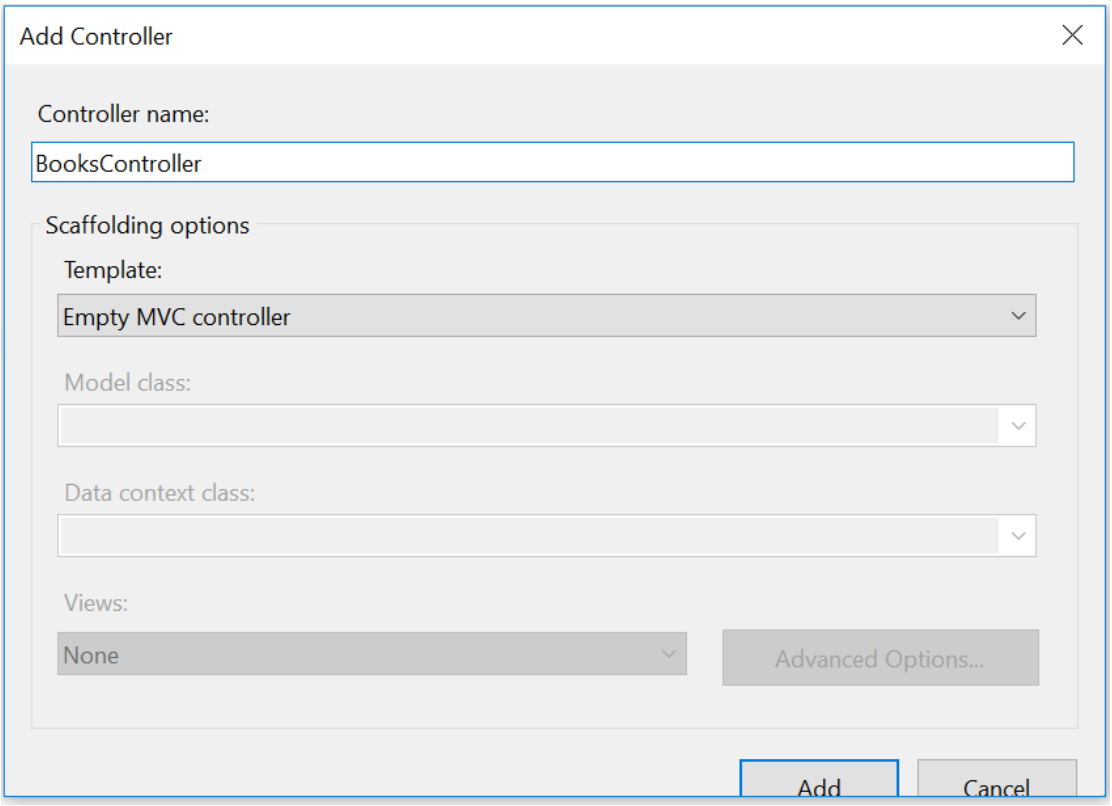
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace SecondProject.Models
{
    public class Book
    {
        public int Id {get;set;}
        public string Title { get; set; }
```

```
        public string Author { get; set; }
    }
}
```

حتى الان عملنا كلاس مهمته تمرير البيانات من الكونترولر الى الفيو .

لنعمل كونترولر (مثلما تعلمنا باليوم الاول) بأسم BooksController :



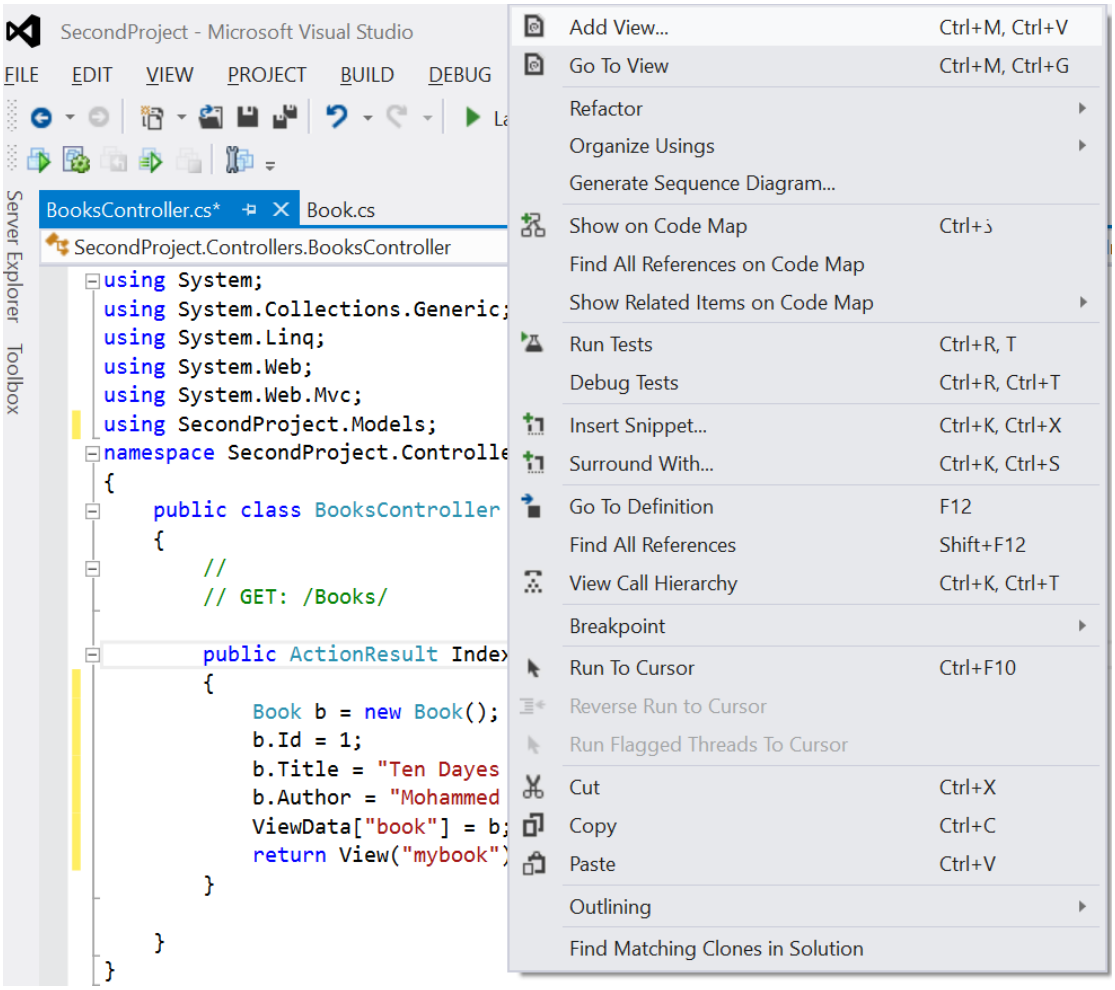
أول خطوة نعملها داخل الكونترولر هي ان نستدعي NameSpace الخاصة بالمودل :

```
using SecondProject.Models;
```

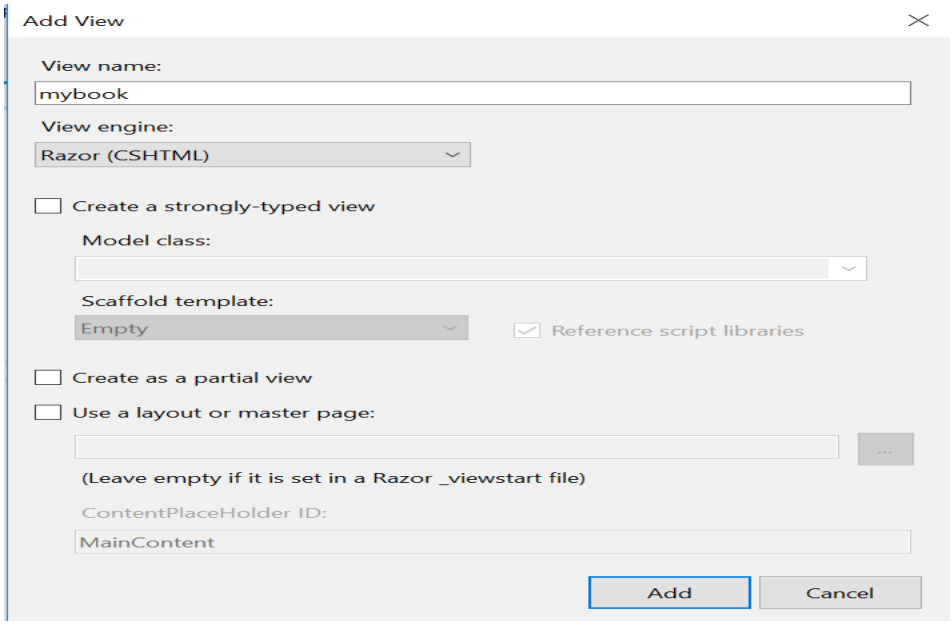
الان داخل الاكشن (Index) سوف نضيف الكود التالي :

```
public ActionResult Index()
        {
            Book b = new Book();
            b.Id = 1;
            b.Title = "Ten Dayes with MVC ";
            b.Author = "Mohammed Alsaady";
            ViewData["book"] = b;
            return View("mybook");
        }
```

في الكود أعلاه عرفنا Object من نوع المودل (book) وملأناه بالبيانات , ثم حملنا هذا الاوبجكت في View Data وأرجعنا الفيو (mybook) الذي سنقوم بأنشاءه الان من خلال Right Click على الاكشن Index ثم Add View :



لا تنسى ان تسميه : mybook مثلما كتبناه بالكود السابق :



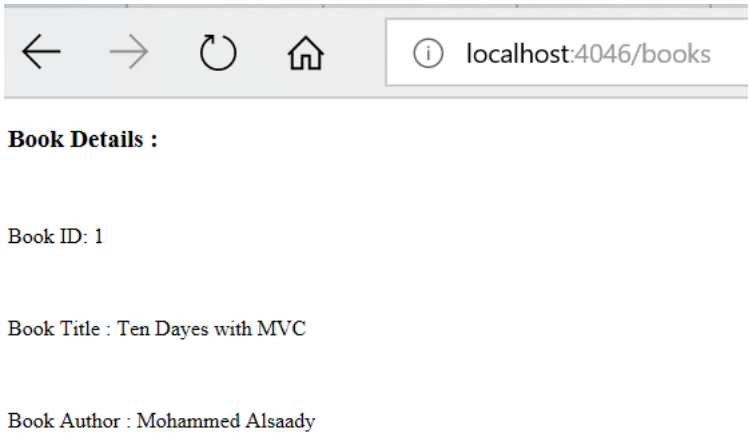
الان داخل الفيو سوف نعمل object من المودل book , حتى نعرض محتوياته التي ارسلت من الكونترولر خلال view data :

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>mybook</title>
</head>
<body>
    <div>
        @{
            SecondProject.Models.Book kitab =
(SecondProject.Models.Book)ViewData["book"];
            }
        <h3>Book Details : </h3><br />
        <p>Book ID: @kitab.Id.ToString()</p><br />
        <p>Book Title : @kitab.Title</p><br />
        <p>Book Author : @kitab.Author </p>
    </div>
</body>
</html>
```

عند التنفيذ :



لو لاحظت بالكود السابق عند كتابة كود السي شارب في الفيو قد أستخدمنا (@{}) لاننا حددنا محرك العرض في بداية انشاء المشروع Razor , كذلك أستخدمنا (@) وهذه نفس السابقة لكن اذا كان لدينا سطر برمجي واحد نستطيع استخدمها بدل من كتابة (@{}) .

ماهي الـــ View Data ؟ عبارة عن حاوية أو قاموس تحمل كل انواع البيانات التي تمرر من الكونترولر الى الفيو .

لدينا نوع آخر مشابه جدا لـــــ View Data تسمى View Bag , لا تختلف كثيرا عن الفيو داتا فقط بالصيغة (Syntax) وانها تستطيع التعامل مع الميزات الدينامكية الجديدة في C# 4 , لناخذ مثال عليها , لو رجعنا الى الكونترولر ومررنا البيانات من خلال View Bag :

```
  public ActionResult Index()
        {
            Book b = new Book();
            b.Id = 1;
            b.Title = "Ten Dayes with MVC ";
            b.Author = "Mohammed Alsaady";
            ViewData["book"] = b;
            ViewBag.book = b;
            return View("mybook");
        }
```

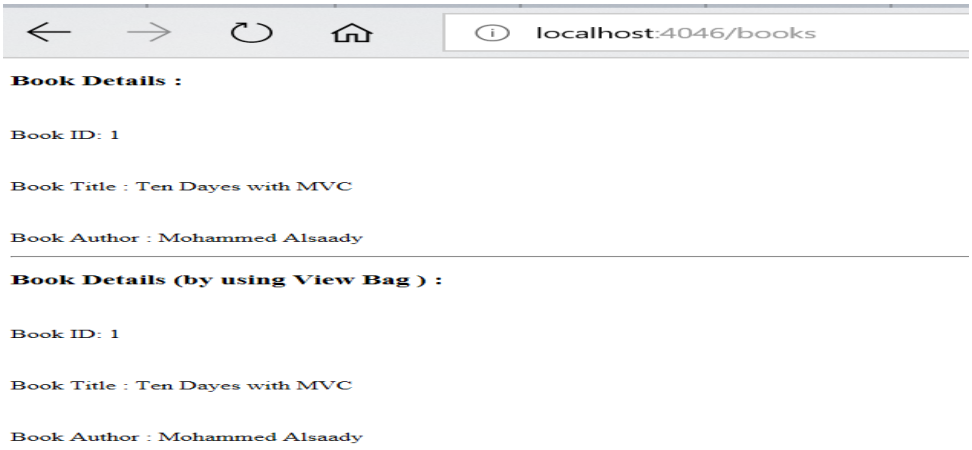
لاحظ صـيغة view Data و View Bag (بأمكانك اسـتخدام اي واحدة) , اما بالفيو حتى نسـتدعيها سـوف نكتب الكود التالي :

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>mybook</title>
</head>
<body>
    <div>
        @{
            SecondProject.Models.Book kitab =
(SecondProject.Models.Book)ViewData["book"];
            SecondProject.Models.Book kitab_viewbag =
(SecondProject.Models.Book)ViewBag.book;
            }
        <h3>Book Details(by using View Data : </h3><br />
        <p>Book ID: @kitab.Id.ToString()</p><br />
        <p>Book Title : @kitab.Title</p><br />
        <p>Book Author : @kitab.Author </p>
        <hr />
        <h3>Book Details (by using View Bag )  : </h3><br />
        <p>Book ID: @kitab_viewbag.Id.ToString()</p><br />
        <p>Book Title : @kitab_viewbag.Title</p><br />
        <p>Book Author : @kitab_viewbag.Author </p>
    </div>
</body>
</html>
```

عند التنفيذ :



اذا تلاحظ في شريط المستفح كتبنا فقط اسم الكونترولر ولم نكتب معاه الاكشن ! لانه بصورة افتراضية سوف ينفذ الاكشن الذي اسمه Index .

طيب سؤال يطرح نفسه اذا كانت View Bag مشابه للـــــ View Data فهل نستطيع تمرير بيانات من الكونترولر من خلال View Dataواستلامها في الفيو بـ View Bag ؟ نعم , نفس المثال السابق سوف نضيف الكود التالي الكونترولر :

```
public ActionResult Index()
        {
            Book b = new Book();
            b.Id = 1;
            b.Title = "Ten Dayes with MVC ";
            b.Author = "Mohammed Alsaady";
            ViewData["book"] = b;
            ViewBag.book = b;
            ViewData["b"] = b;
            return View("mybook");
        }
```

لاحظ في الكود اعلاه قد مررنا الاوبجكت من خلال View Data[“b”]

اما في الفيو :

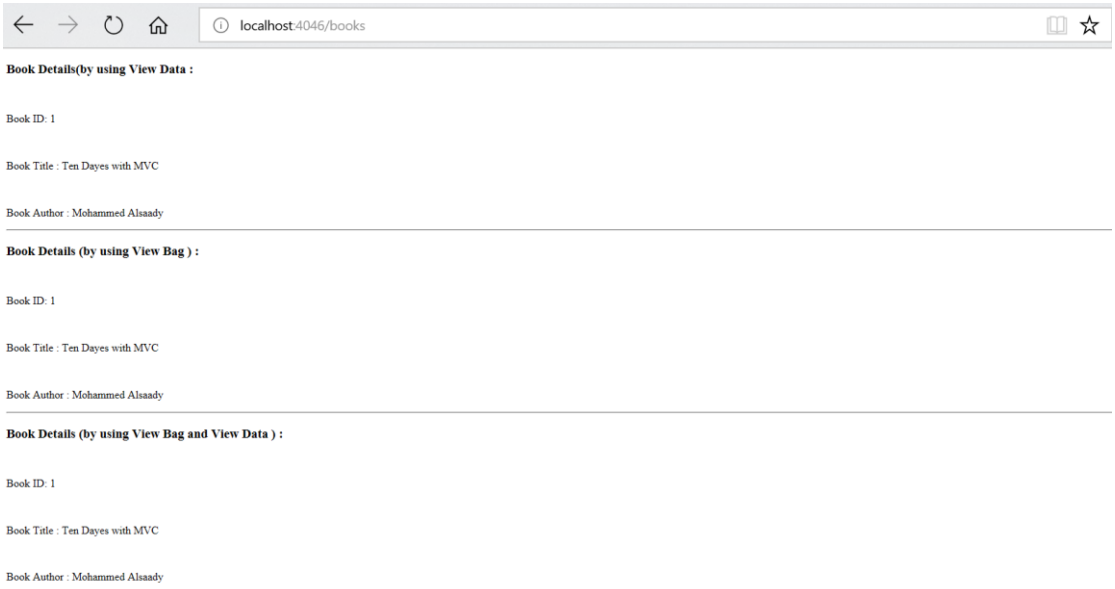
```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>mybook</title>
</head>
<body>
    <div>
        @{
            SecondProject.Models.Book kitab =
(SecondProject.Models.Book)ViewData["book"];
            SecondProject.Models.Book kitab_viewbag =
(SecondProject.Models.Book)ViewBag.book;
            SecondProject.Models.Book Kitab_viewdata
=(SecondProject.Models.Book)ViewBag.b;
```

```
            }
        <h3>Book Details(by using View Data : </h3><br />
        <p>Book ID: @kitab.Id.ToString()</p><br />
        <p>Book Title : @kitab.Title</p><br />
        <p>Book Author : @kitab.Author </p>
        <hr />
        <h3>Book Details (by using View Bag )  : </h3><br />
        <p>Book ID: @kitab_viewbag.Id.ToString()</p><br />
        <p>Book Title : @kitab_viewbag.Title</p><br />
        <p>Book Author : @kitab_viewbag.Author </p>
        <hr />
        <h3>Book Details (by using View Bag and View Data )  : </h3><br />
        <p>Book ID: @Kitab_viewdata.Id.ToString()</p><br />
        <p>Book Title : @Kitab_viewdata.Title</p><br />
        <p>Book Author : @Kitab_viewdata.Author </p>
    </div>
</body>
</html>
```

هنا استلمنا (ViewData) على شكل View Bag . عند التنفيذ :



حتى الان فأن View Data وView Bag خيارات جيدة وســـهلة لتمرير البيانات الى الفيو من الكونترولر , لكن ! لكن ماذا ؟ لا ينصـح بهم !!! لنراجع المثال السـابق لقد قمنا فيه بتمرير Object من نوع المودل (Book) خلال View Data وView Bag من الكونترولر الى الفيو, وفي الفيو قمنا بتحويل View Data أو View bag الى النوع Book :

```
SecondProject.Models.Book Kitab_viewdata
=(SecondProject.Models.Book)ViewBag.b
```

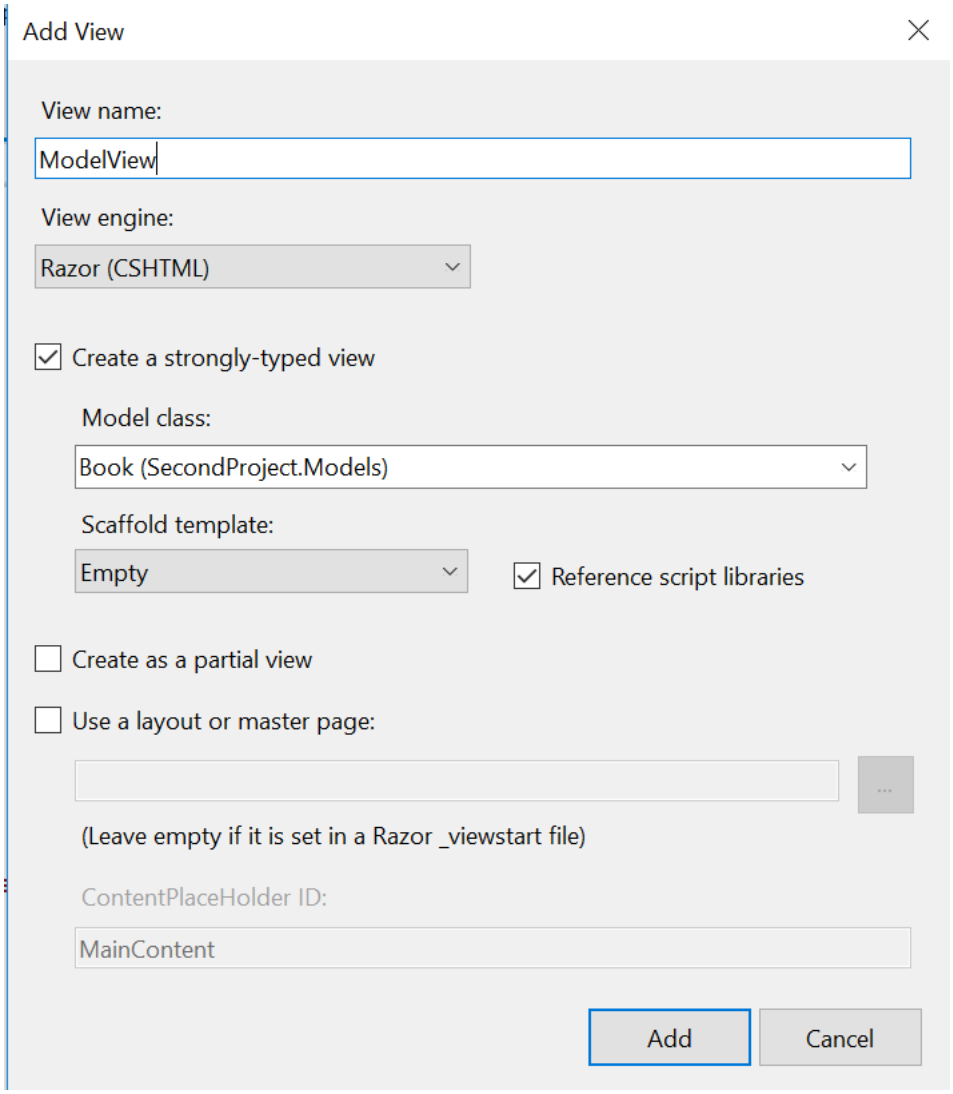
ماذا لو اخطأت بنوع التحويل ؟ هذا واحد من ســلبيات اســتخدام View Data وView Bag , للعلم أن الكونترولر ليس له شــأن بالتحويل وتبقى هذه مشــكلة الفيو ! طيب ماذا لو كان الشــخص الذي عمل الكونترولر غير الشخص الذي يعمل على الفيو ؟!

أذا المشــكلة بالتحويل ولتجاوزها ســوف نقوم بتمرير البيانات من الكونترولر الى الفيوعن طريق المودل (على أعتبار ان البيانات هي من نوع المودل لذلك لسنا مضطرين الى تحويلها) .

لنأخذ مثال ونرى , ننشـــئ أكشـــن جديد داخل الكونترولر بأســـم (BookAction) ونأخذ أوبجكت من المودل ونملأه بالبيانات ثم نرجع الفيو ومعه الاوبجكت :

```
public ActionResult BookAction()
        {
            Book b = new Book();
            b.Id = 2;
            b.Title = "C# 5.0 Pocket Reference ";
            b.Author = "Joseph Albahari ";
            return View("ModelView",b);
        }
```

الان ننشئ فيو جديد من خلال Right Click على BookAction ونختار Add View ونسميه : ModelView :



نؤشـــر على (Create a strongly-typed view) يعني ســـوف نجعل الفيو من نوع المودل , ونختار من القائمة (Book SecondProject.Models) .

سنرى ان السطر التالي موجود في أعلا الفيو الجديد :

```
@model SecondProject.Models.Book
```

نكتب كود الفيو سوف نتعامل مع المودل مباشرة ولا نحتاج ان نعمل التحويل :

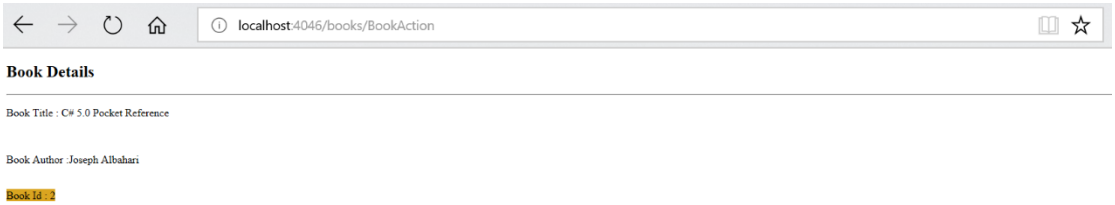
```
@model SecondProject.Models.Book

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ModelView</title>
</head>
<body>
    <div>
        <h2> Book Details </h2><hr />
        <p>Book Title : @Model.Title</p><br />
        <p>Book Author :@Model.Author</p><br />
        @if (Model.Id == 2)
        {
            <span style="background-color:goldenrod">
                Book Id : @Model.Id.ToString()
            </span>
        }
    </div>
</body>
</html>
```

هل ترى في الكود اعلاه اننا لم نتسخدم لا Data View ولا View Bag وهذا يعتبر الاصح ان نجعل الفيو من نوع المودل لذلك بعدها لا نحتاج ان نعمل اوبجكت ونحوله , السلبية الوحيدة في هذا الشي اننا لا يمكن ان نعمل نوع الفيو من أكثر من مودل!.



لو أردنا اضافة بيانات جديدة للفيو مثلا مستخدم قد قام بتسجيل الدخول , بهذه الحالة لدينا خيارين :

الاول : ان نضيف خصائص جديدة الى المودل (book) للمستخدم مثلا (UserName) وهذا غير منطقي كون المودل خاص بالكتب وليس له علاقة بالمستخدمين الذين سيكون لهم مودل خاص بهم , وقد قلنا سابقا ان الفيو لا يمكن ان يكون من نوع أكثرمن مودل .

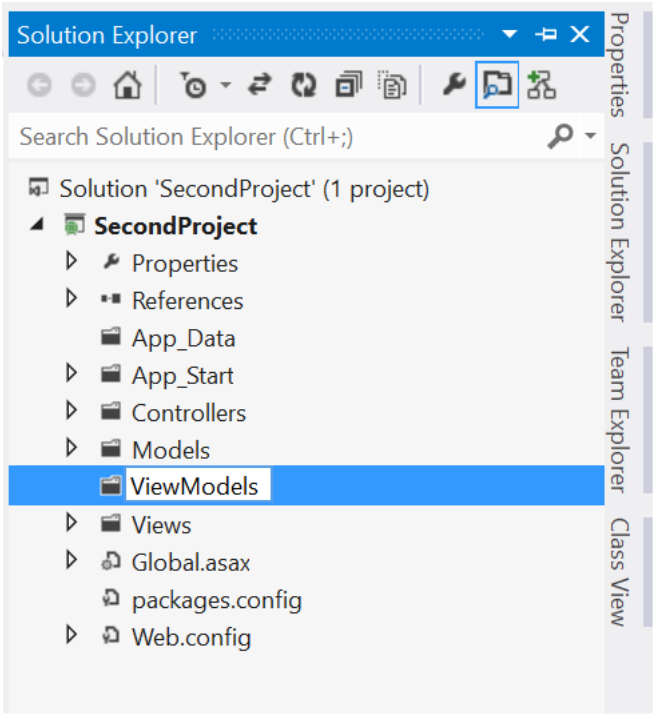
ثانياً : ان نستخدم الـ View Data أو View Bag وقد أسلفنا سابقا الى سلبيات أستخدامهما .

إذا ما الحل ؟ وضعت لنا مايكروسوفت شي لحل هذه المشكلة و (ViewModel) وهي عبارة عن حاوية بيانات خاصة بالفيو . وهي تنشئ أعتمادا على الفيو , عكس المودل الذي ينشئ حسب العمل او قاعدة البيانات .

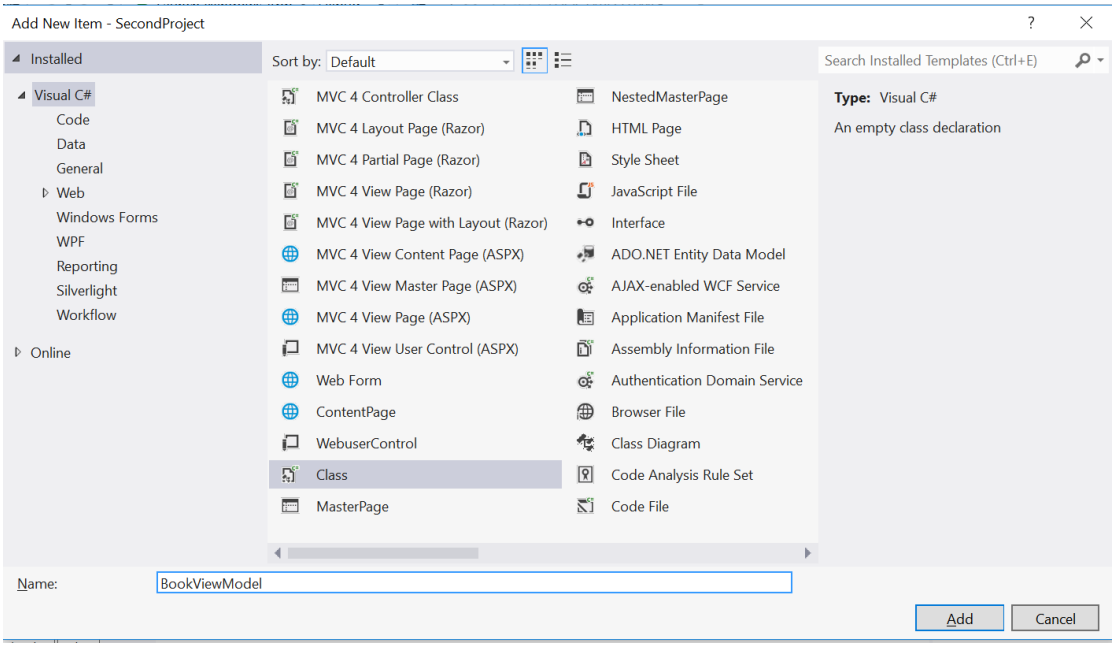
وفكرة عمل View Model هي بعدما يستلم الكونترولر طلب المستخدم يقوم بمعالجته ويستدعي اكثر من مودل حسب نوع الطلب ثم يقرر أي فيو مناسب للأجابة , بعد ذلك يقوم الكونترلر بأنشاء View Model Object من البيانات الراجعة من المودل أعتمادا متطلبات الفيو . بعدها يقوم الكونترولر بتمرير الفيو مودل الى الفيو . في هذه الحالة سيكون الفيو مكتوب بشكل قوي من نوع المودل فيو .

الكلام اعلاه ربما فيه صعوبه نوعا ما لنأخذ مثال عليه :

نعود الى مشروعنا (SecondProject) وننشئ داخله مجلد خاص بافيو مودل بأسم (View Models) :



الان نضيف داخل هذا المجلد كلاس بأسم (BookViewModel) .



نكتب الشفرة التالية :

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

namespace SecondProject.ViewModels
{
    public class BookViewModel
    {
```

```
        public string Id { get; set; }
        public string AboutBook { get; set; }
        public string IdColor { get; set; }
        public string UserName { get; set; }
    }
}
```

ماذا فعلنا ؟ اننا بالبداية قمنا بجعل جميع البيانات التي سـوف ترسـل للفيو من نفس النوع وهو (String) لذلك جعلنا (Id) من هذا النوع , وسنقوم بالتحويل داخل الكونترولر حتى نجنب الفيو من ان يقوم بأي تحويل للبيانات . ثم بعدها قمنا بتعريف خاصـية (AboutBook) التي سـوف تجمع عنوان الكتاب والمؤلف , وقمنا بأضـافة خاصية جديدة خاصة بلون خلفية (Id) .

الان لنرجع الى الكونترولر ونكتب فيه اكشن جديد :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SecondProject.Models;
using SecondProject.ViewModels;
namespace SecondProject.Controllers
{
    public class BooksController : Controller
    {
        //
        // GET: /Books/

        public ActionResult Index()
        {
            Book b = new Book();
            b.Id = 1;
            b.Title = "Ten Dayes with MVC ";
            b.Author = "Mohammed Alsaady";
            ViewData["book"] = b;
            ViewBag.book = b;
            ViewData["b"] = b;
            return View("mybook");
        }
        public ActionResult BookAction()
        {
            Book b = new Book();
            b.Id = 2;
            b.Title = "C# 5.0 Pocket Reference ";
            b.Author = "Joseph Albahari ";
            return View("ModelView",b);
        }
        public ActionResult BookViewModel()
        {
            Book b = new Book();
            b.Id = 3;
            b.Title = "Windows Server 2016 Unleashed";
            b.Author = "Rand Morimoto ";
            //----------------
            BookViewModel bViewModel = new BookViewModel();
            bViewModel.Id = b.Id.ToString();
            bViewModel.AboutBook = "Title Of Book :" + b.Title + " , By : " +
b.Author;
            if (b.Id > 2)
            {
```

```
                bViewModel.IdColor = "aqua";
            }
            else
            {
                bViewModel.IdColor = "goldenrod";
            }
            bViewModel.UserName = "Mohammed";
            return View("BookView",bViewModel);
        }

    }
}
```

ماذا فعلنا ؟ قمنا بالبداية بملأ أوبجكت المودل بالبيانات ثم جعلنا الفيو مودل تأخذ البيانات من هذا الاوبجكت بعد تحويلها جميعها الى نوع واحد (String) ثم ارسلناها الى الفيو (BookView) .

الان ننشئ هذا الفيو من خلال Right Click على BookViewModel بأسم : (BookView) .

```
@model SecondProject.ViewModels.BookViewModel
@using SecondProject.ViewModels
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>BookView</title>
</head>
<body>
    <div>
        <h5>Welcome @Model.UserName</h5>
        <hr />
        <p>About Book : @Model.AboutBook</p><br />
        <span style="background-color:@Model.IdColor">
            Id Of Book : @Model.Id
        </span>
    </div>
</body>
</html>
```

و الاستعراض كالتالي :

نلتق باليوم الثالث .