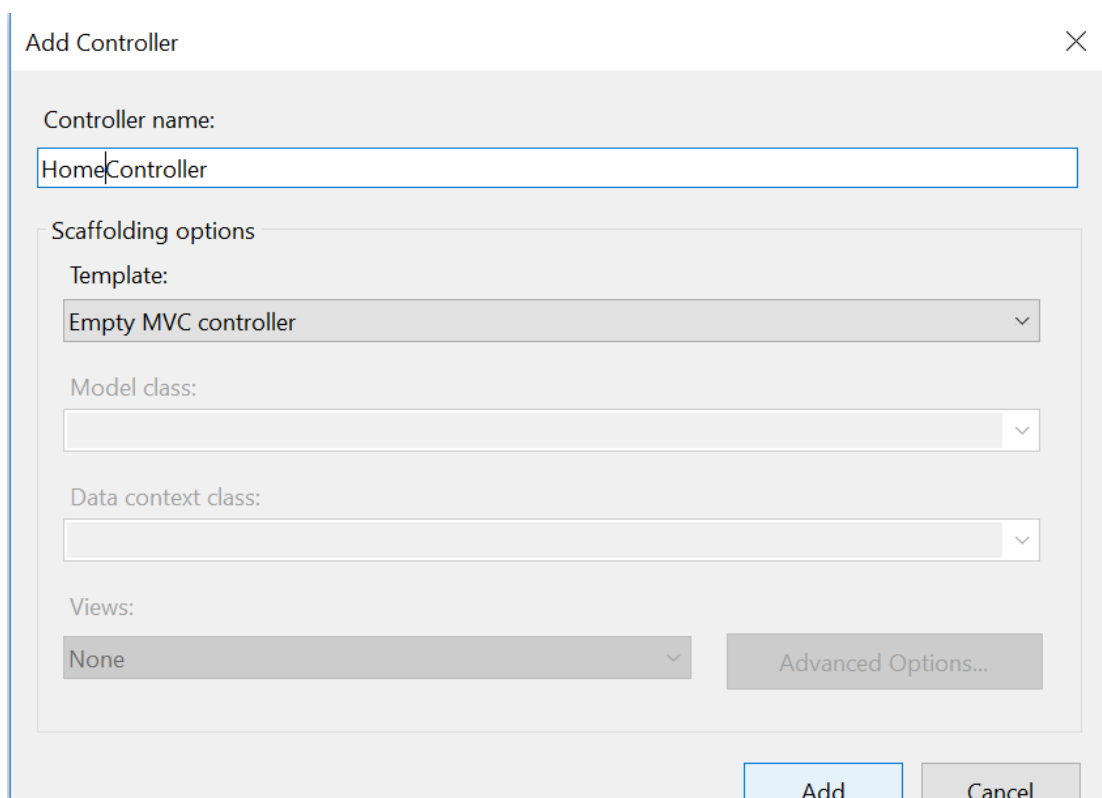


اليوم الرابع

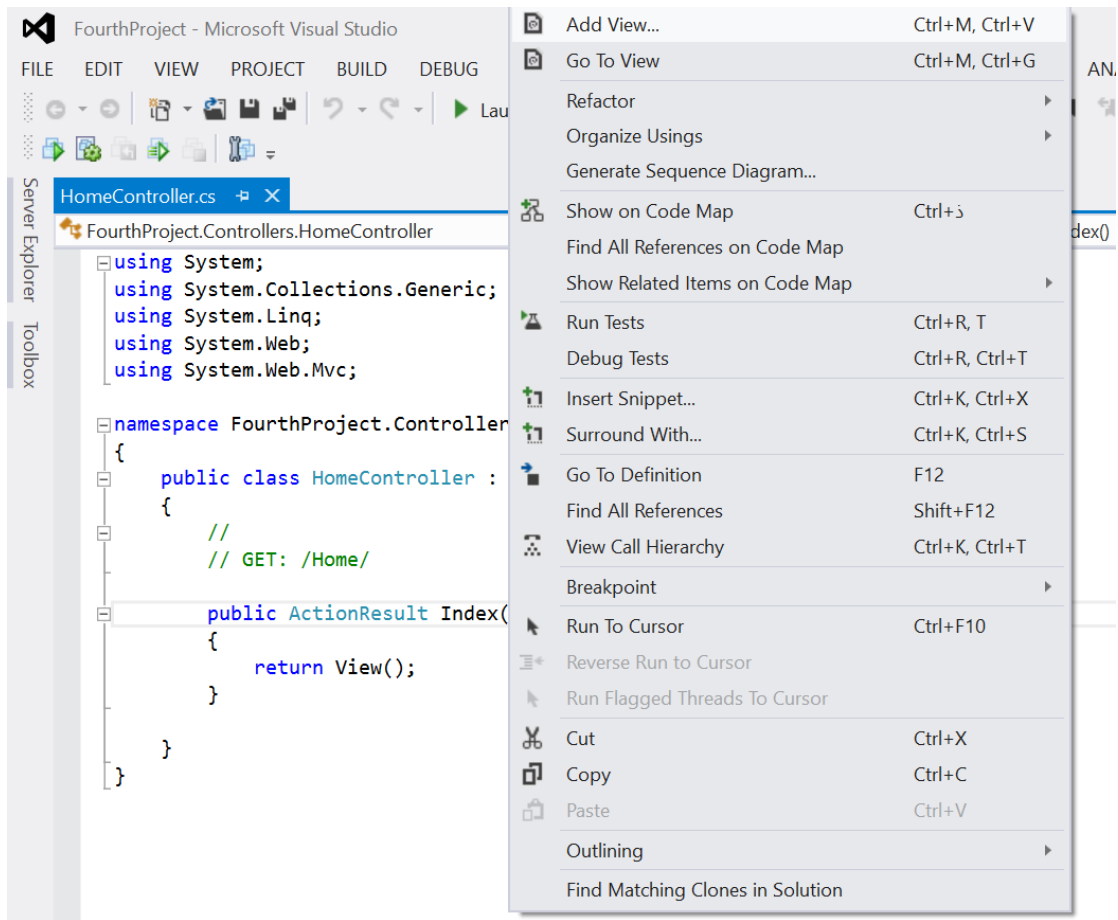
في اليوم الثالث أخذنا مثال حول الدرجات خصوصا في الفيو إذا لاحظت كثيرا أستخدمنا (HTML Helper) ما هذه ؟ وكيف نستخدمها ؟ في هذا الدرس سنحاول شرحها بالاضافة الى مواضيع اخرى .

ربما تسأل إن سبق لك وعملت على (Web Form) كيف يمكنني إضافة العناصر (Controls) مثل (Text Box) أو (Button) ... الخ في MVC ! هناك بالسحب والاسقاط (Drag and Drop) نستطيع استخدام أي عنصر- لكن هنا في MVC الوضع نوعا ما مختلف بعض الشيء- . حيث مايكروسوفت سهلت عليك استخدام العناصر من خلال استخدام أداة جميلة وسهلة الاستخدام سميتها (HTML Hepler) التي تمكننا من استخدام عناصر (HTML) مثل الروابط ومربعات النصوص ... الخ , الميزة الاخرى في هذه أنها لا تحتوي على أحداث أو (View State) وهذا ما يجعل عناصرها خفيفة وسريعة .

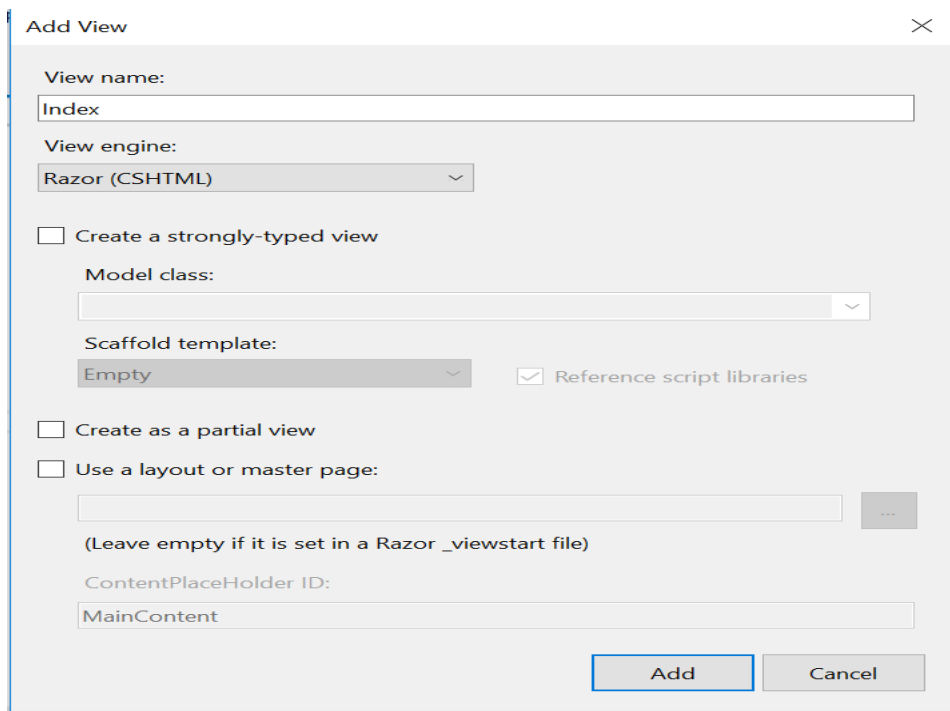
لننشئ مشروع جديد بأسم (FourthProject) , نضيف داخله كونترولر جديد بأسم (HomeController) :



سوف نضيف (View) جديد من خلال (Right Click) على الاكشن (Index) :



نتركه على اسمه الافتراضي (Index) :



الآن سوف نستخدم (HTML Helper) داخل الفيو كالتالي :

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

```

}

<!DOCTYPE html>

<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Index</title>
</head>
<body>
  <div>
    <h3>Welcome </h3>
    <p>Fourth day with MVC , To Learn more about MVC ....</p>
    @Html.ActionLink("Click Here", "About")
  </div>
</body>
</html>

```

لقد عملنا لنك الى اكشن آخر هو (About) , الان نفذ المشروع .



Welcome
 Fourth day with MVC , To Learn more about MVC
[Click Here](#)

لو نقرت على (Click Here) سوف تظهر لنا رسالة خطأ تقول أن الاكشن (About) غير موجود .

لنتكلم أكثر عن (HTML.ActionLink()) , هذا الميثود فيه الكثير من المعاملات (Parameters) أبرزها :

```

@Html.ActionLink(

```

▲ 1 of 10 ▼ (extension) MvcHtmlString HtmlHelper.ActionLink(**string linkText**, string actionName)
 Returns an anchor element (a element) that contains the virtual path of the specified action.
linkText: The inner text of the anchor element.

- ❖ LinkText – عنوان اللنك كما رأينا بالمثال السابق (Click Here) .
- ❖ Action Name – أسم الاكشن الذي ننتقل له .
- ❖ Route Value – لتمرير قيمة الاكشن الاخر .
- ❖ Controller Name – أسم الكونترولر الذي سننتقل للأكشن الذي بداخله .
- ❖ htmlAttributes – اضافة خصائص html للرباط .
- ❖ protocol – بروتوكول الرباط مثلا (http او https) .
- ❖ Hostname – دومين المضيف مثلا (www.w3school.com) .

في المعاملات اعلاه لم نرى معامل خاص بالصور التي تعمل كرابط , في مثل هذه الحالة سوف نستخدم ميثود آخر (Url.Action) .

لو عدنا الى المثال السابق , أضف فيه مجلد جديد وسميه (Images) وضع فيه اي صورة :

```

@{
  Layout = null;
}

```

```

<!DOCTYPE html>

<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Index</title>
</head>
<body>
  <div>
    <h3>Welcome </h3>
    <p>Fourth day with MVC , To Learn more about MVC ....</p>
    @Html.ActionLink("Click Here", "About")
    <hr />
    <p>Do you have problem ? </p>
    <a href="@Url.Action("ContactUs")"></a>
  </div>
</body>
</html>

```

وعند التنفيذ تكون كالتالي :



Welcome

Fourth day with MVC , To Learn more about MVC

[Click Here](#)

If you have any question



ماذا لو أردنا أن نعمل فورم خاصة بتسجيل جديد بالموقع ؟ نستطيع بسهولة أن نستخدم الميثود (html) ونختار العناصر التي نريد إضافتها كما في الصورة :



Create New User ...

Register

First Name : *

Last Name : *

Password : *

Confirm Password : *

Profile :

Receive News Letter?

لعمل الفورم اعلاه , نضيف أكشن جديد داخل الكونترولر (Home) نسماه (Register) :

```

public ActionResult Register()
{
    return View();
}

```

ونضيف فيو جديد بأسم (Register) , داخل هذا الفيو نضيف الكود التالي :

```

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Register</title>
</head>
<body>
    <div>
        <h3>Create New User ...</h3>
        @using (Html.BeginForm())
        {
            <fieldset>
                <legend>Register</legend>
                <p>
                    <label for="FirstName">First Name :</label><br />
                    @Html.TextBox("FirstName")
                    @Html.ValidationMessage("FirstName", "*")
                </p>
                <p>
                    <label for="LastName">Last Name :</label><br />
                    @Html.TextBox("LastName")
                    @Html.ValidationMessage("LastName", "*")
                </p>
                <p>
                    <label for="Password">Password :</label><br />
                    @Html.Password("Password")
                    @Html.ValidationMessage("Password", "*")
                </p>
                <p>
                    <label for="Password">Confirm Password :</label><br />
                    @Html.Password("ConfirmPassword")
                    @Html.ValidationMessage("ConfirmPassword", "*")
                </p>
                <p>
                    <label for="Profile">Profile :</label><br />
                    @Html.TextArea("Profile")
                </p>
                <p>
                    @Html.CheckBox("ReceiveNewsLetter")
                    <label for="ReceiveNewsLetter" style="display:inline">Receive
News Letter?</label>
                </p>
                <p>
                    <input type="submit" value="Register" />
                </p>
            </fieldset>
        }
    </div>
</body>
</html>

```

بالنسبة للـ (Validation) سنتناوله فيما بعد لكن المهم عندنا (HTML Helper) .

العيب في Html Helper أنها تحوي عدد محدد من العناصر , ولتجاوز هذا العيب سوف نقوم بصنع أي عنصر نريده بسهولة .

على سبيل المثال سنحاول أنشاء عنصر (Submit) أضف مجلد جديد الى المشروع بأسم (Helpers) وداخله كلاس جديد بأسم (SubmitButtonHelper) :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace Helpers
{
    public static class SubmitButtonHelper
    {
        public static IHtmlString SubmitButton(this HtmlHelper helper, string buttontext)
        {
            string value = String.Format("<input type=\"submit\" value=\"{0}\" />", buttontext);
            return new HtmlString(value);
        }
    }
}
```

الكود أعلاه قمنا فيه بالتعديل على الميثود (SubmitButton) الموجوده في (Html Helper) .

وحتى نستخدمه بالفيو نستدعي :

```
@using FourthProject.Helpers
```

بعدها نستدعي العنصر الجديد بسهولة في الفيو كما في الصورة أدناه :



```
@Html.SubmitButton("Submit")
```

مشكلة الطريقة أعلاه أنها غير مرنة , لو أردنا إضافة خصائص إضافية لعناصر (HTML Helper) عند انشائها , فأننا سوف نستخدم كلاس آخر يسمى (TagBuilder Class) , وهذا الكلاس يحوي على الميثود التالية :

- AddCssClass() – هذا يمكننا من إضافة خاصية (css Class) لوسم العنصر .
- GenerateId() – إضافة خاصية ال (ID) لوسم العنصر .
- mergeAttribute() – إضافة مجموعة من الخصائص للوسم .
- SetInnerText() – لإضافة (Inner Text) للوسم .

- ToString()- لاستخدام الوسم حيث نحدد ما نريده لإنشاء وسم جديد مثلا وسم بداية او اغلاق الوسم .

كذلك أن هذا الكلاس فيه أربع خصائص مهمة وهي :

- Attributes -يمثل جميع خصائص الوسم .
- IdAttributeDotReplacement - يمثل الرمز المستخدم من قبل الميثود (GenerateId) .
- InnerHtml - المحتويات الداخلية للوسم .
- TagName -يمثل اسم الوسم .

مثلا لو أردنا إضافة وسم الصورة , فأنا سوف نضيف كلاس جديد بأسم (ImageHelper) داخل مجلد (Helpers) , ونضيف فيه الكود التالي :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace Helpers
{
    public static class ImageHelper
    {
        public static IHtmlString CustomImage(this HtmlHelper
htmlHelper,string src, string alt, int width, int height)
        {
            var imageTag = new TagBuilder("image");
            imageTag.MergeAttribute("src", src);
            imageTag.MergeAttribute("alt", alt);
            imageTag.MergeAttribute("width", width.ToString());
            imageTag.MergeAttribute("height", height.ToString());

            return new
HtmlString(imageTag.ToString(TagRenderMode.SelfClosing));
        }
    }
}
```

في الكود أعلاه عرفنا متغير من نوع (TagBuilder) واضفنا الخصائص له التي نريدها للوسم , الان نذهب الى الفيو (Index) ونستخدم هذا الوسم بداخله .

```
@Html.CustomImage("../Images/Home.jpg", "Home", 70,85)
```

والتنفيذ يكون كالتالي :

Welcome

Fourth day with MVC , To Learn more about MVC

[Click Here](#)

Do you have a problem ?



هناك كلاس آخر أسمه (HTMLTextWriter) هذا مشابه لـ (TagBuilder) لكن فيه ميثود إضافية ومن هذه الميثود :

- ❖ AddAttribute() – إضافة خصائص للوسم .
- ❖ AddStyleAttribute() – إضافة خصائص الستايل للوسم .
- ❖ RenderBeginTag() – استدعاء وسم البداية .
- ❖ RenderEndTag() – استدعاء وسم الاغلاق .
- ❖ Write() – لكتابة نص .
- ❖ Writeline() – كتابة سطر جديد .

نلتق في اليوم الخامس ان شاء الله