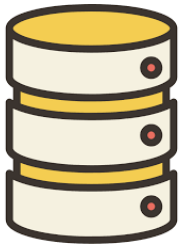


دليل المبرمج 2.1

ASP.Net core mvc

.NET Core 2.1



database first entity
framework



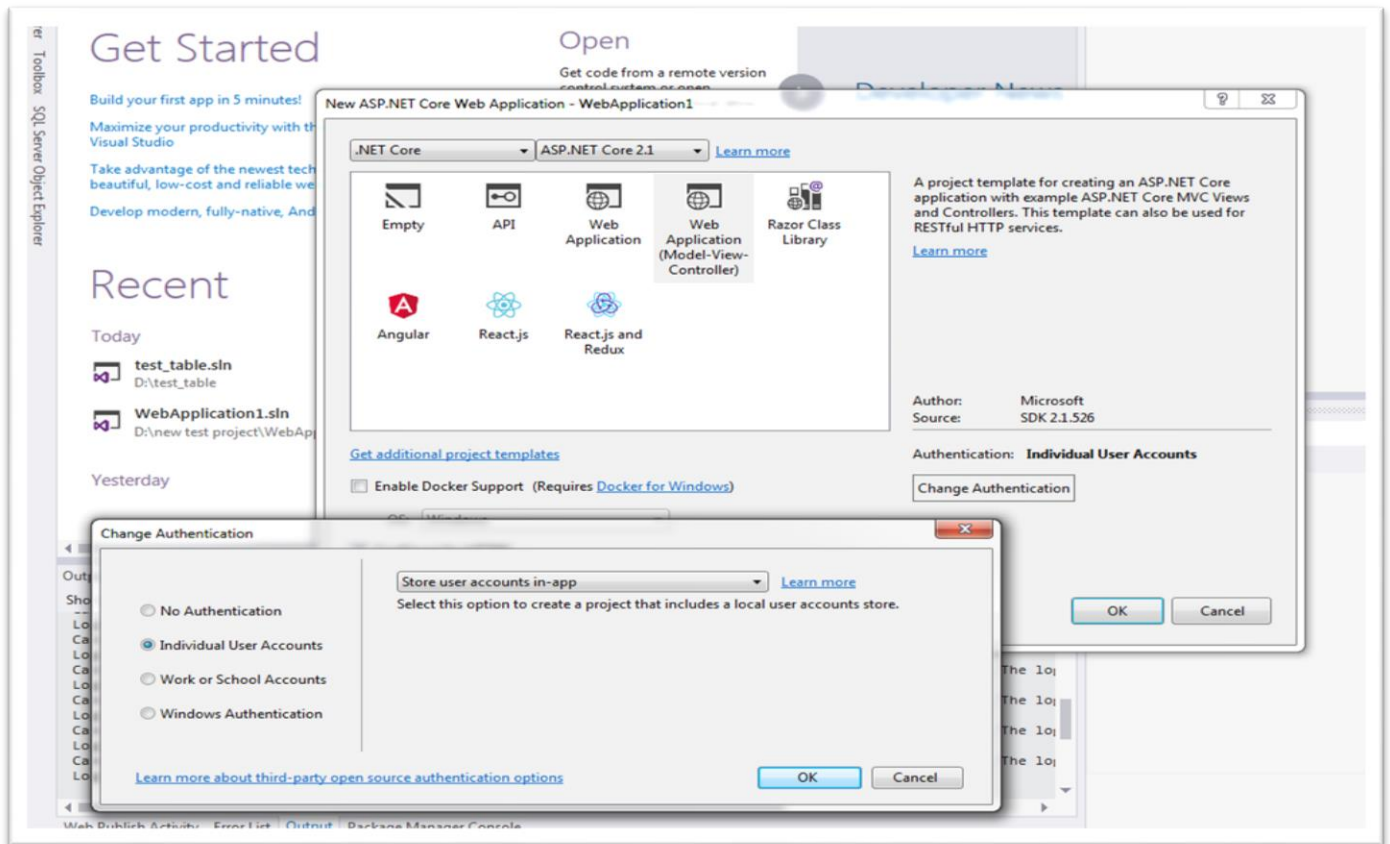
AHMED SABRY - [011-584-217-98]

دليل المبرمج 2.1

انشاء مشروع جديد



- أ- من قائمة file نختار new ثم نختار project ، او عن طريق الاختصار Ctrl+shift+n
- ب- نختار Mvc
- ج- نضغط change authentication ، ثم نختار individual user authentication
- د- نضغط ok



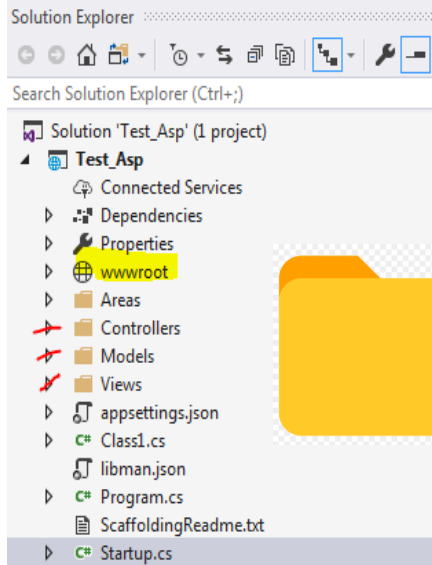
ملحوظه: عند القيام بالخطوة رقم ج سيتم انشاء فولدر باسم Areas

دليل المبرمج 2.1



wwwroot

داخل قائمة الـ solution يوجد ملف **wwwroot** : يحتوى على ملفات الصور والتنسيقات مثل css و js



☑ مجلد **Areas** : خاص بعمل Identity وهو المسؤول عن عمل تسجيل او اشتراك للموقع ، هذا المجلد تم انشاءه عندما قمنا بعمل change authentication

← مجلد الـ **models** خاص بالكلاسات (قواعد البيانات)

← مجلد الـ **Controllers** خاص بالتحكم بطريقة عرض الموديل فى الـ view

← مجلد الـ **view** خاص بالصفحات التى سيتم عرضها للمستخدم

☑ مجلد **Shared** موجود داخل مجلد الـ view يحتوى على صفحة **_Layout.cshtml** وهى المسؤولة عن اضافة اجزاء ثابتة فى كل صفحات الموقع مثل navbar و footer على سبيل المثال

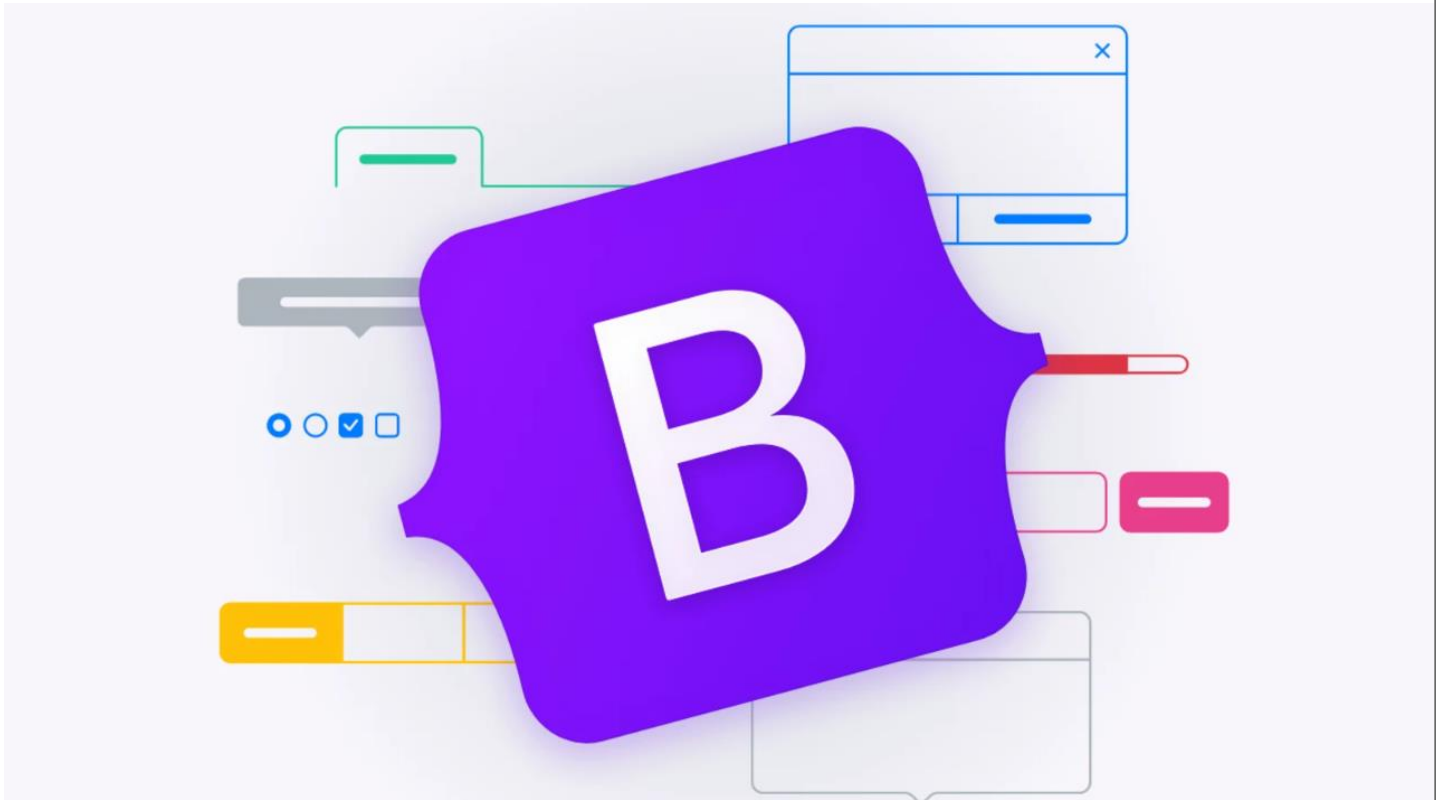
دليل المبرمج 2.1

إضافة المكتبات الى المشروع

Install Bootstrap

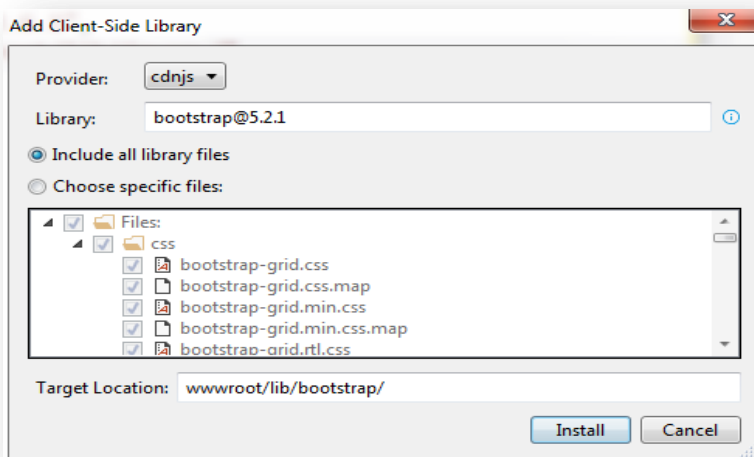
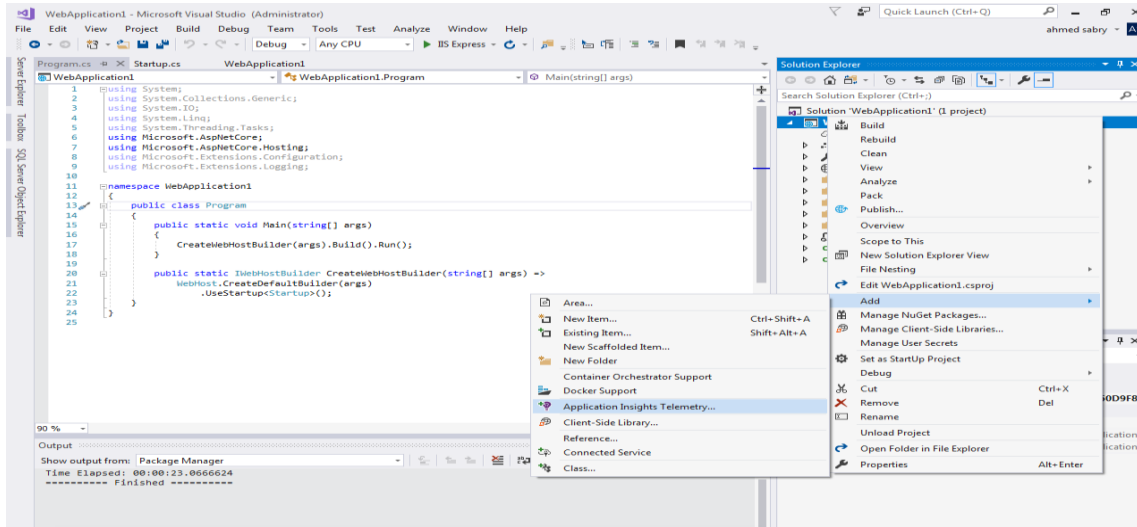


هي مكتبة تحتوى على كلاسات "CSS" جاهزة ، ومكونات HTML بتنسيقات جاهزة – تساعد في بناء تصميم صفحات الويب، وهي تفيد بشكل أساسي في بناء مواقع متناسقة "responsive" مع مختلف أحجام وقياسات الشاشات، بما في ذلك شاشات الهواتف والأجهزة اللوحية، وهي مكتبة مجانية .



دليل المبرمج 2.1

من قائمة FILE نختار Add ثم نختار client-side-library



نختار CDN وهي اختصار لـ Content Delivey Network باختصار أي شبكة توصيل والمحتوى، وهي مجموعة من الخوادم المتزامنة و الموزعة على شبكة الأنترنت، تحتوي على مكتبات مثل مكتبة بوتستراب و مكتبة jquery و مكتبة popper.js (...)

ثم نكتب bootstrap ثم نضغط tab

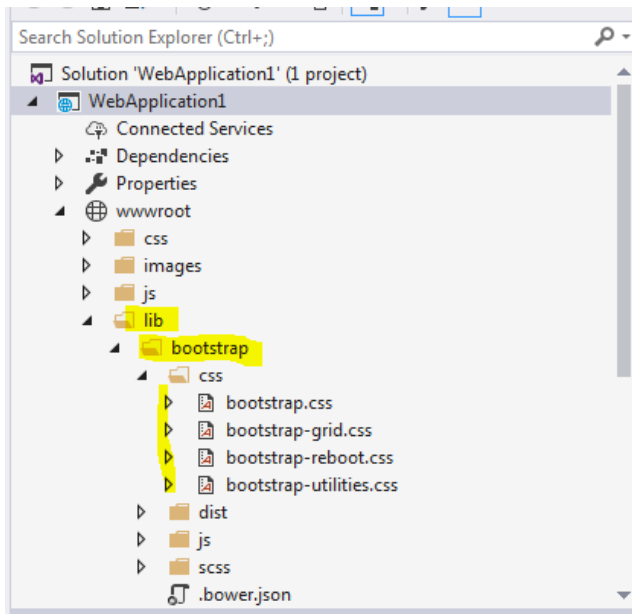
bootstrap@5.2.1

ملاحظة : بعد الضغط على tab سيكتب تلقائيا علامة @ ثم رقم اخر اصدار من المكتبة ، ويمكن تعديل الرقم الذى يلى رمز @ برقم اصدار سابق للمكتبة

داخل الـ SOLUTION سنجد ملف Lib بداخله مجلد bootstrap

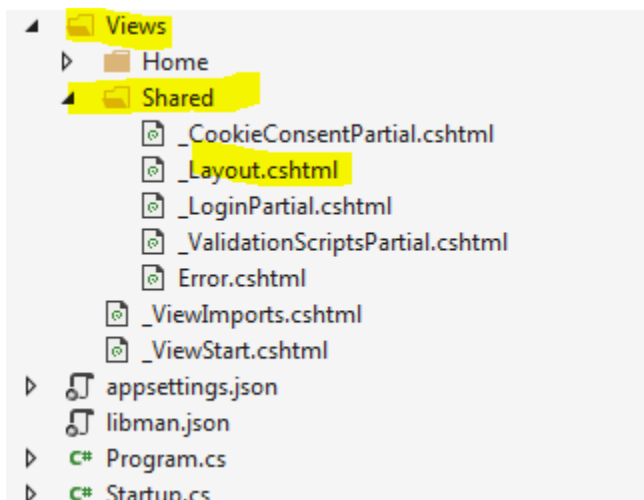
AHMED SABRY - [011-584-217-98]

دليل المبرمج 2.1



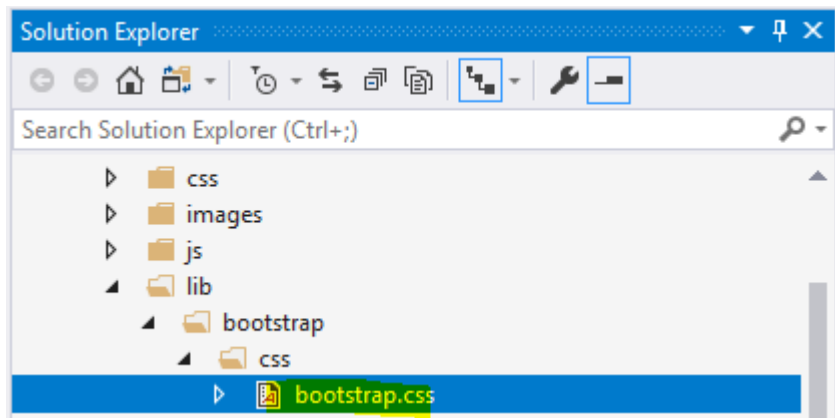
طريقة استدعاء المكتبة داخل ال VIEW

من VIEW نختار Shared ثم `Layout.cshtml` ثم نقوم بفتحه



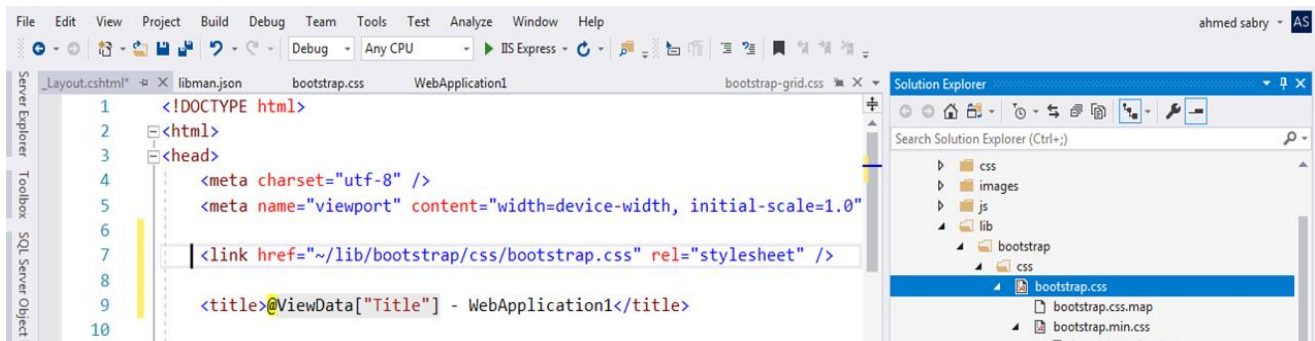
دليل المبرمج 2.1

من قائمة الـ SOLUTION نختار lib ثم bootstrap ثم css ثم
نسحب ملف bootstrap.css داخل وسم الـ <head>



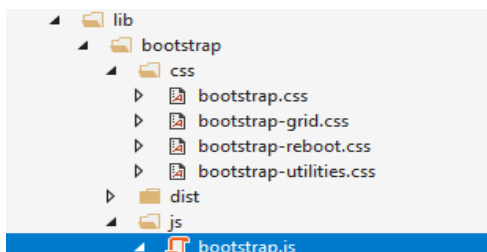
بعد السحب والافلات سنجد كتابة السطر التالي

```
<link href="~/lib/bootstrap/css/bootstrap.css" rel="stylesheet" />
```



bootstrap الخاصة بالـ js طريقة اضافة ملفات الـ

نسحب ملف bootstrap.js فى صفحة الـ _Layout قبل اغلاق وسم الـ body



```
<script src="~/lib/bootstrap/js/bootstrap.js"></script>

</body>
</html>
```

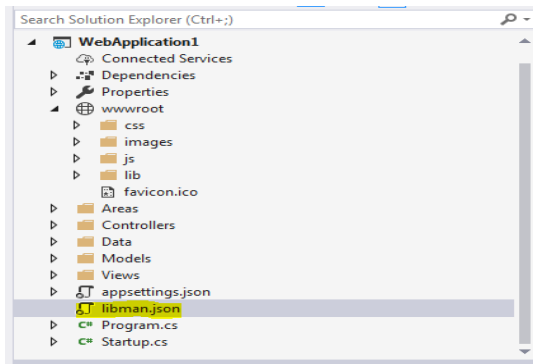
دليل المبرمج 2.1

Install jquery

jQuery هي عبارة عن مكتبة تحوي مجموعة من الدوال خاصة بال JavaScript، ومهمة هذه المكتبة هي اختصار العمليات التي تحتاج عدداً كبيراً من الأسطر البرمجية إلى مجموعة توابع تُستدعى بسطر برمجي واحد.



بعد ان اضفنا CDN سنجد ملف تم انشاءه باسم `libman.json` ، واذا قمنا بفتحه سنجد بداخله مكتبة البوت ستراب التي قمنا بعمل اضافته لها .



دليل المبرمج 2.1

صورة من الملف بعد فتحه

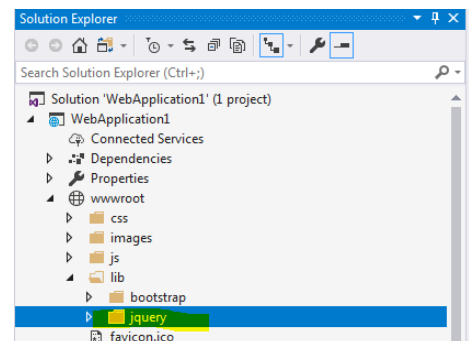
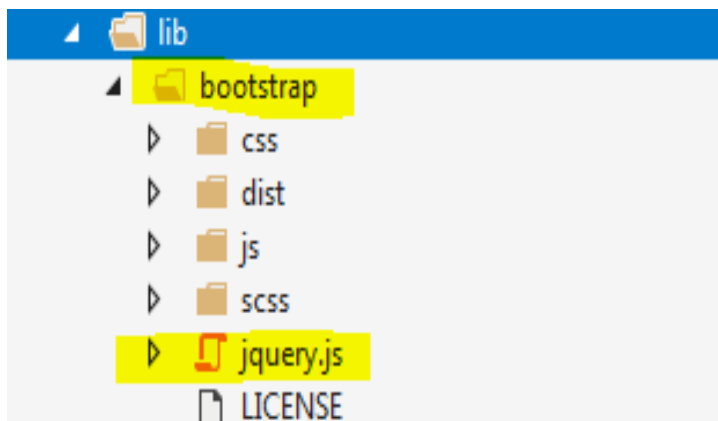
```
"libraries": [
  {
    "library": "bootstrap@5.2.1",
    "destination": "wwwroot/lib/bootstrap/"
  }
]
```

يمكن ان نضيف مكتبة اخري من خلال ملف ال libman.json عن طريق الكود

نضيف علامة comma " و " بعد ال curly brackets ثم نضيف المكتبة بين curly brackets هكذا

```
"libraries": [
  {
    "library": "bootstrap@5.2.1",
    "destination": "wwwroot/lib/bootstrap/"
  },
  {
    "provider": "cdnjs",
    "library": "jquery@3.6.1",
    "destination": "wwwroot/lib/jquery/"
  }
]
```

ثم نقوم بالحفظ CTRL+S ، سنجد اضافة ملف الجى كويرى داخل المسار التالى lib
jquery.js ← bootstrap ←



دليل المبرمج 2.1

طريقة اختبار المكتبة

```
<script>
  $(document).ready(function () {
    alert("مجاني تعليمي موقع")
  }
);
</script>
```

Install font-awesome



font-awesome@6.2.0

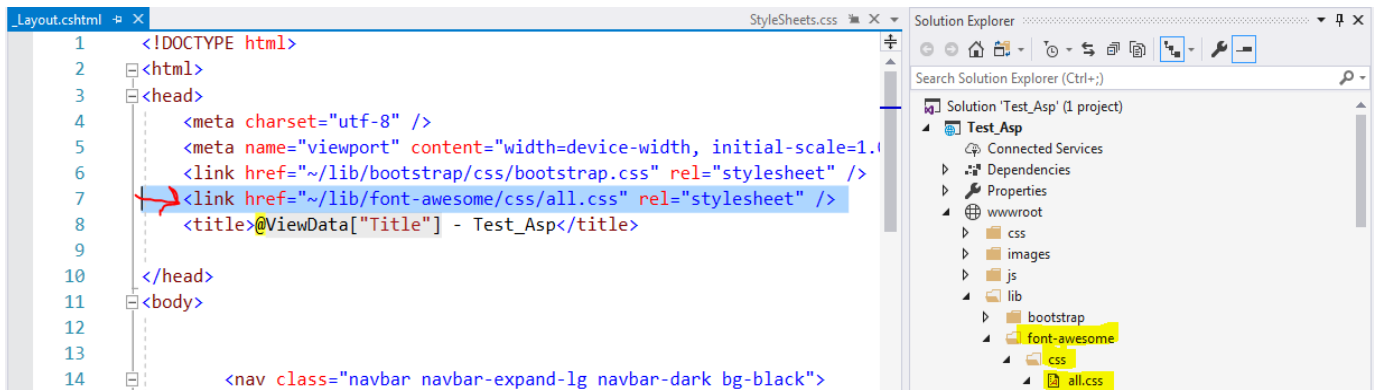
هي مكتبة لاضافة ايقونات في موقعك فهي تتعامل مع هذه الايقونات كأنها خط
و يمكن اعطائها تنسيق حجم خط و لون خط و غير ذلك



للممكن اضافة المكتبة من خلال طريقة client-side-library او من
خلال ملف libman.json

دليل المبرمج 2.1

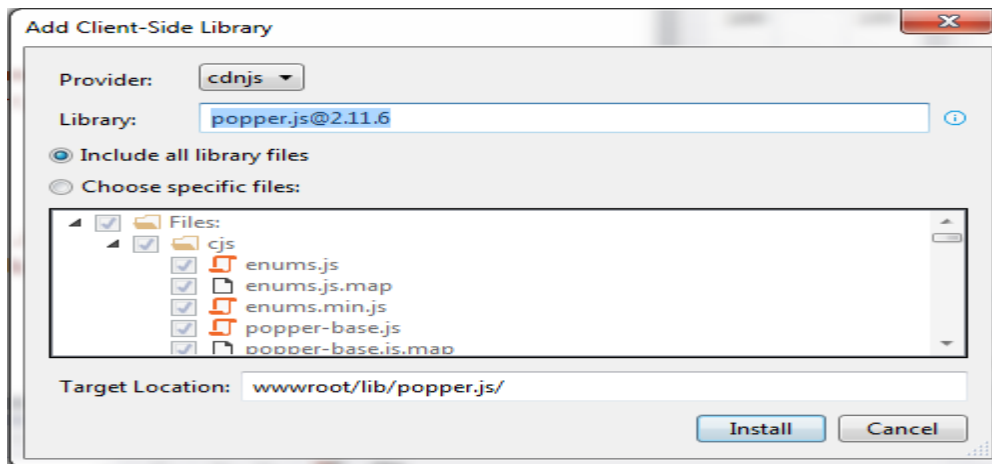
☑ من قائمة الـ SOLUTION نختار **wwwroot** ثم **lib** ثم **font-awesome** ثم **css** ثم نسحب ملف **all.css** فوق وسم اغلاق الـ **</head>**



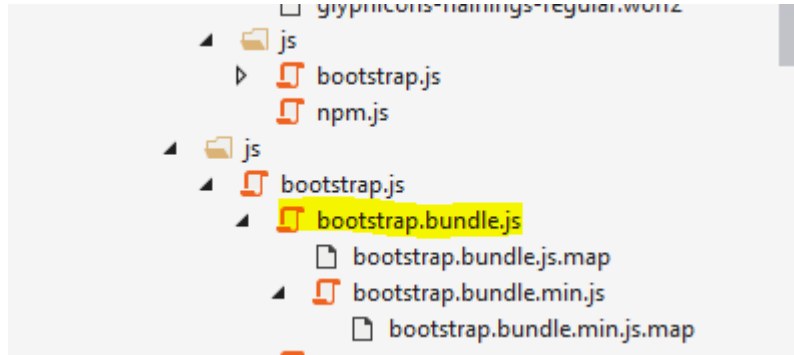
Install popper

هذه المكتبة خاصة بتشغيل بعض الادوات مثل Dropdown داخل الـ Nav bar

[popper.js@2.11.6](#)



دليل المبرمج 2.1

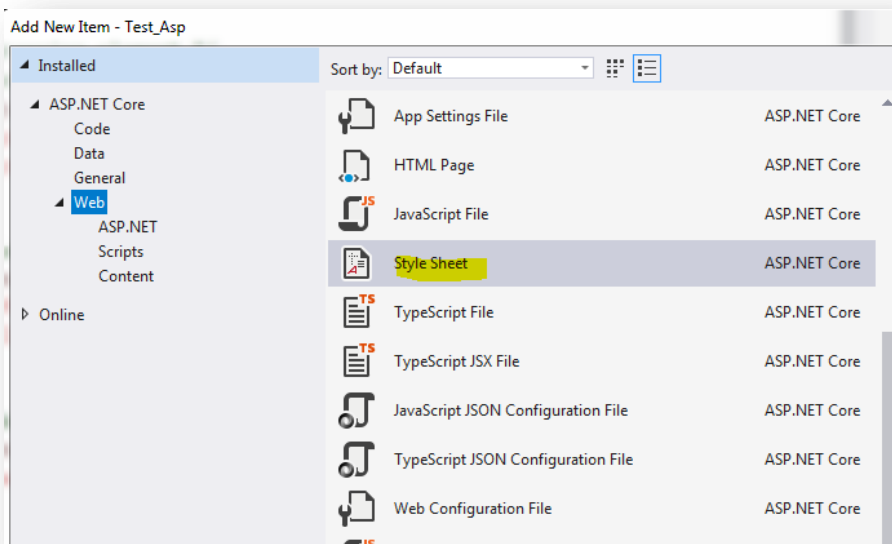


نسحب الملف bootstrap.bundle.js ونضعه قبل اغلاق وسم الـ body

```
<script src="~/lib/bootstrap/js/bootstrap.bundle.js"></script>
</body>
</html>
```

طريقة اضافة ملف CSS خارجي

من قائمة الـ SOLUTION نختار **wwwroot** ثم نضغط كليك لمين ثم **Add** ثم **new folder** ونقوم بعمل **rename** لل فولدر باسم **css** ثم نضغط كليك لمين على الفولدر ثم **add** ثم **new item** ثم نختار **web** ثم نختار **StyleSheet**



دليل المبرمج 2.1

طريقة اضافته في ال view

من قائمة ال SOLUTION نسحب ملف StyleSheet داخل وسم ال <head>

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.
  <title>Document</title>
  <link href="/lib/bootstrap/css/bootstrap.css" rel="stylesheet" />
  <link href="/style/StyleSheet.css" rel="stylesheet" />
</head>
```

صفحة ال _Layout.cshtml هي صفحة تحتوى على الجزء الثابت في كل الصفحات على سبيل المثال اذا كان هناك navbar او footer نريد تضمينه في جميع الصفحات نضعه في ال ال _Layout.cshtml

اما في وسم ال body لصفحة ال _Layout.cshtml فهو الجزء متغير من صفحة الى اخرى لذلك نكتب @RenderBody()

```
<body>
  @RenderBody()
</body>
```

طريقة ازالة ال-Layout من صفحة معينة

نكتب: Layout = null;

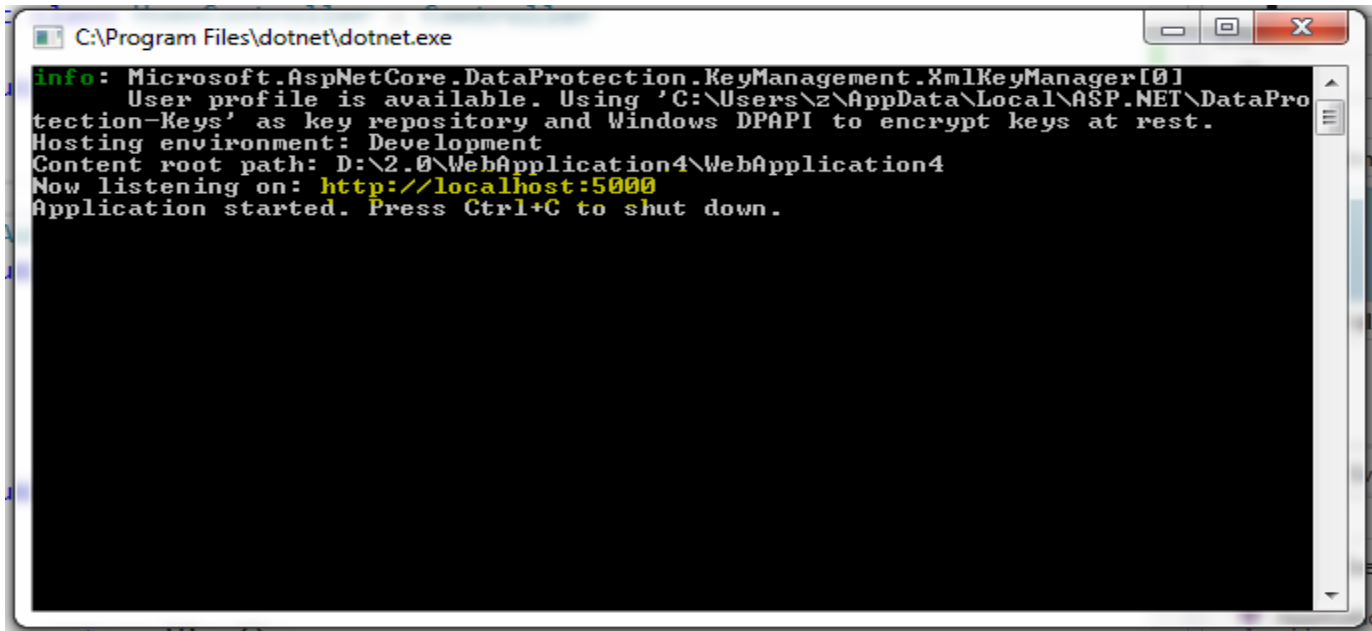
```
@{
  ViewData["Title"] = "tiger";
  Layout = null;
}
```

دليل المبرمج 2.1

طريقة تشغيل المشروع

نضغط على F5

سيتم فتح ملف كنسول dotnet.exe

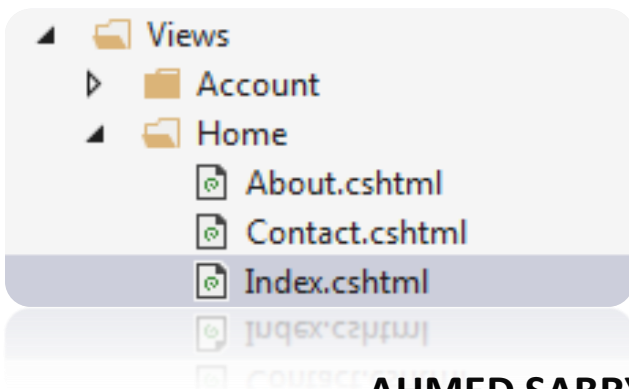


```

C:\Program Files\dotnet\dotnet.exe
info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[0]
      User profile is available. Using 'C:\Users\z\AppData\Local\ASP.NET\DataPro
tection-Keys' as key repository and Windows DPAPI to encrypt keys at rest.
Hosting environment: Development
Content root path: D:\2.0\WebApplication4\WebApplication4
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
  
```

سنجد رابط localhost وهو بمثابة رابط محلي لاستضافة الموقع يمكن ان نكتب هذا العنوان في المتصفح ليتم الدخول على الـ localhost
طريقة اخرى

من قائمة solution نختار فولدر View ← Home ← Index.cshtml
ثم نقوم بالضغط على w + shift + ctrl



دليل المبرمج 2.1

design patterns (mvc)

MVC: هو اختصار لـ model view controller وهو (خدمة - Service) (او نمط في التصميم يعمل على تجنب المشاكل الموجودة بكثرة في التصميم يعتمد النموذج على عزل منطق العمل عن واجهة الاستخدام محققًا بذلك استقلالية لكل منهما في التطوير، الفحص والصيانة).

ملف الـ Startup.cs

هو ملف يحتوي اضافة الخدمات وشكل الـ

داخل الـ solution يوجد ملف باسم Startup.cs لو قمنا بفتحه سنجد دالة تدعى ConfigureServices تحتوي هذه الدالة على اضافة خدمة الـ mvc

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```

model

يحتوي مجلد الـ model على

- ① الكلاسات التي تمثل قواعد البيانات ، والكلاس يعد بمثابة table
- ② كلاس Dbcontext بمثابة Dataset بداخله مجموعة من (كلاسات) بمثابة جداول

دليل المبرمج 2.1

controller

هو كلاس يحتوى على

① Dbcontext : بمثابة Dataset بمعنى انها تمثل مجموعة جداول قاعدة البيانات

② اكشن او اكثر لكل صفحة view مربطة بالاكشن ،

وظيفة الاكشن : من خلال الاكشن تتحكم فى كلاسات الـ dbcontext ، وننظم البيانات الموجودة بداخله وتتحكم فى طريقة العرض داخل صفحة الـ View ويمكن ان تستقبل بارمتر من صفحة الـ View الخاصة بها

على سبيل المثال تحويل كلاس الموديل الى list او لاجراء عمليات الاستعلام مثل الاضافة والحذف والتعديل

view

هى الصفحات التى تظهر للمستخدم ويمكن استدعاء عناصر الموديل داخل الـ view عن طريق تقنية Razor تسمح لنا بكتابة بكتابة html داخل سى شارب

دليل المبرمج 2.1

Razor



ال Razor هي طريقة لكتابة سى شارب داخل Html

طريقة اضافة تعليق فى ال Razor

@* -----writeComments----- *

طريقة كتابة كود داخل Scope ال Razor

نقوم بكتابة @ ثم نقوم بفتح curly brackets ثم نقوم بغلق ال curly brackets

```
@{
}
```

مثال لكتابة كود سى شارب داخل وسم HTML

```
<ul>
@for (int i = 0; i < 10; i++) {
<li>@i</li>
}
</ul>
```

Tag helper

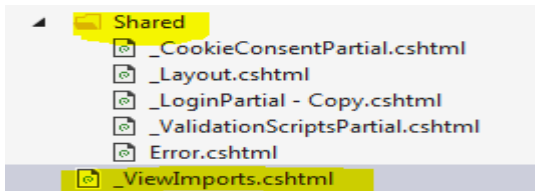
هي طريقة لكتابة او تضمين عناصر #c داخل Html tag

```
<form class="nav-link"
asp-controller="اسم الكنترول"
asp-action="اسم الاكشن">
method ="post"
</form >
```

طريقة اضافة مفهوم ال Tag helper فى المشروع

من Views ← Shared ← ViewImports.cshtml عند فتح الملف
سنجد ان ال TagHelper موجودة على مستوى المشروع بالكامل لذلك يمكن
استخدامها فى اى صفحة من صفحات ال view

دليل المبرمج 2.1



```
@using Test_Asp
@using Test_Asp.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

ViewData & ViewBag

ViewData

تستخدم ViewData لتعريف متغير داخل Razor
تعريف متغير في صفحة الـ view واعطاه قيمة بواسطة ViewData

```
@{
    ViewData["x"] = "y"; //var x as string =y
    ViewData["m"] = 5; //var m as int = 5
}
```

طريقة استدعاء المتغير داخل وسم HTML نكتب @ ثم المتغير داخل الوسم

```
<h2>@ViewData["x"]</h2>
<h3>@ViewData["m"]</h3>
```

ViewBag.varname

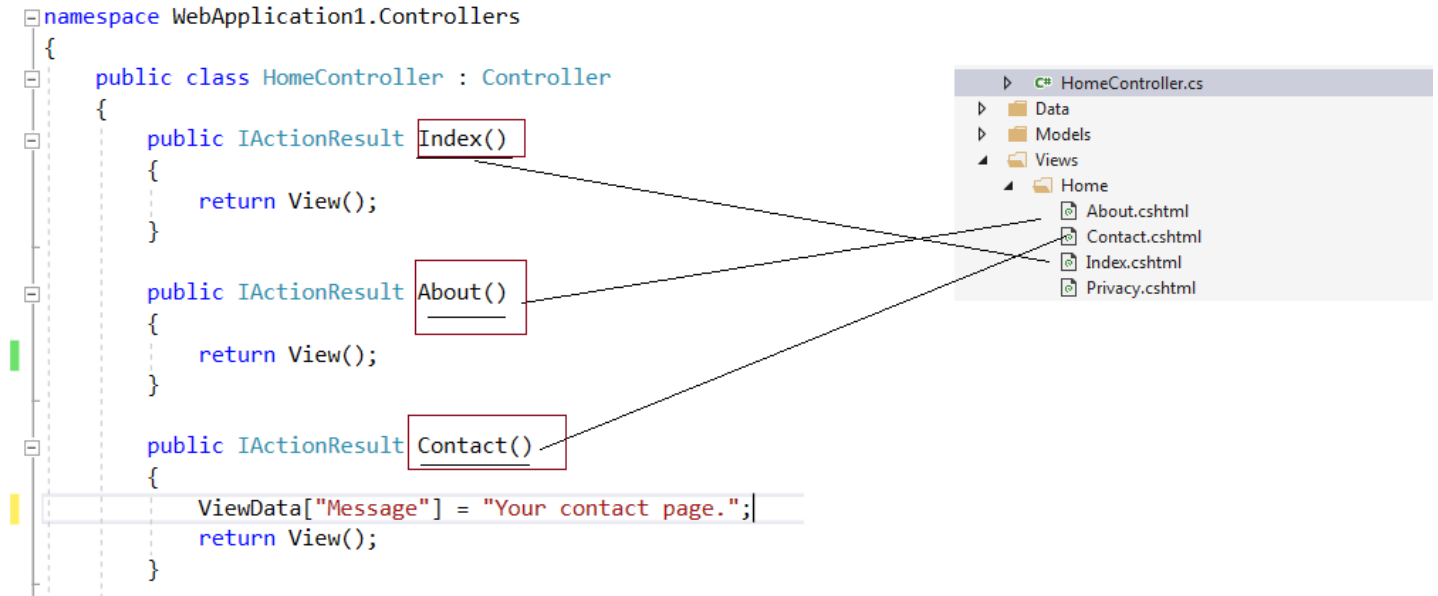
```
@{
    ViewBag.x = "mohamed";
}
<h1> @ViewBag.x </h1>
ActionResult
```

الاكشن : هو بمثابة اتفاق يمثل نتيجة عرض الدالة الخاصة بالاكشن .

يحتوى الكنترول اكشن او اكثر لكل صفحة view اما ان تكون للعرض فقط او لارسال بيانات من الموديل الى الـ view والعكس صحيح

دليل المبرمج 2.1

شكل الكنترول من الداخل



صيغة كتابة الاكشن

```

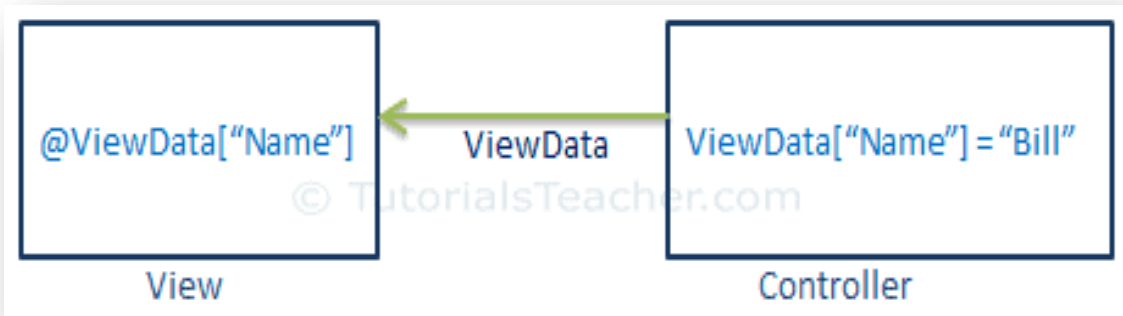
Public IActionResult view name()
{
    Return (view);
}

```

view name : تستبدل باسم الصفحة المراد التعامل معها حيث ان الاكشن يعد بمثابة method خاصة بالصفحة

دليل المبرمج 2.1

Sending Data from control to view



```

public IActionResult About()
{
    ViewData["Message"] = "Your application page.";
    ViewData["show_Message"] = 5;
    return View();
}
  
```

```

<h2>@ViewData["Message"]</h2>
<h3>@ViewData["show_Message"]</h3>
  
```

```

public IActionResult About()
{
    ViewBag.x = "ahmed";
    return View();
}
  
```

```

<h1> @ViewBag.x </h1>
  
```

دليل المبرمج 2.1

ملاحظات :

الـ ViewData تحتاج الى تحديد نوع المتغير عند استقبال القيمة
الـ ViewData يمكن ان ترجع قيمة فارغة

الـ ViewBag لاتحتاج الى تحديد نوع المتغير عند استقبال القيمة
الـ ViewBag لا يمكن ان ترجع قيمة فارغة
مثال تطبيقي للتوضيح (١)

```
public IActionResult About()
{
    IList<string> studentList = new List<string>();
    studentList.Add("ahmed");
    studentList.Add("mohamed");
    studentList.Add("omar");
    ViewData["students"] = studentList;
    return View();
}
```

```
<ul>
    @foreach (var std in (IList<string>) ViewData["students"])
    عند استخدام (ViewData) يلزم تعريف نوع المتغير //
    {
        <li>
            @std
        </li>
    }
</ul>
```

دليل المبرمج 2.1

Or

```
<ul>
@foreach (var std in ViewData["students"] as IList<string>)
{
<li>
@std
</li>
}
</ul>
```

مثال تطبيقي للتوضيح (٢)

```
public IActionResult About()
{
    IList<string> studentList = new List<string>();
    studentList.Add("ahmed");
    studentList.Add("mohamed");
    studentList.Add("omar");
    ViewBag.item_name = studentList;
    return View();
}
```

```
<ul>
    @foreach (var std in ViewBag.item_name)
    // عند استخدام (ViewBag) لا نحتاج الى تعريف نوع المتغير
    {
        <li>
            @std
        </li>
    }
</ul>
```

دليل المبرمج 2.1

routing

الـ routing هو طريقة تستخدم لعمل لينك مربوط بصفحة view بحيث يتم التوجه الى الصفحة التي ترغب في عرضها عن طريق اللينك

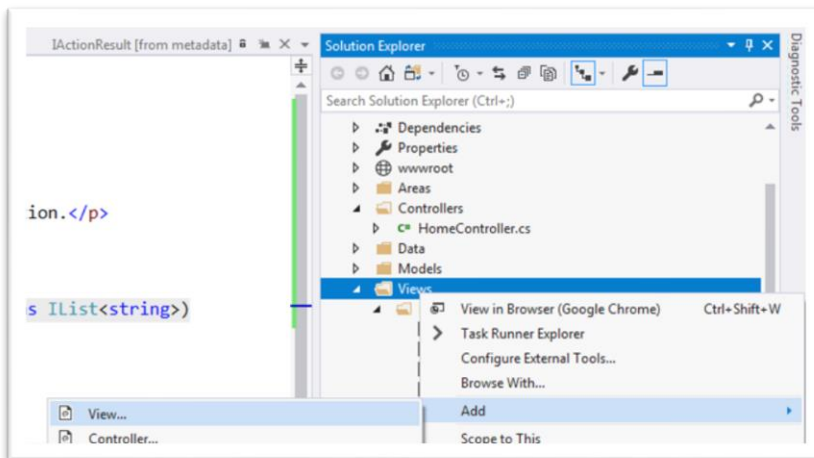
```
<a class="nav-link"
asp-controller="اسم الكنترول"
asp-action="اسم الاكشن">
send mail</a>
```

ملاحظة :

عند انشاء البروجيكت بشكل افتراضى يتم انشاء كنترول باسم Home ويوجد مجلد داخل مجلد الـ view بنفس اسم الكنترول يوجد بداخله اربع صفحات (Index-Privacy -Contact -About)

طريقة اضافة view للكنترول

من solution نختار مجلد view ثم الفولدر الذى يحتوى على الصفحات (Home) ثم add ثم نختار view



دليل المبرمج 2.1

ثم نختار اسم ال view على سبيل المثال نكتب جوجل ثم نضغط Add

من قائمة solution نختار `_Layout.cshtml` ، سنجد في ال navbar
الينك تم انشاءه تلقائيا

```
<li><a
asp-controller="Home"
asp-action="Contact">Contact</a>
</li>
```

طريقة عرض الurl في الrouting

في ملف ال `Startup.cs` يوجد template يحتوى على شكل عرض الurl حيث
يظهر الكنترول ثم الاكشن ثم الid ان وجد ، ويمكن تغير هذه ال template الى
الشكل الذى تريده

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
}
```

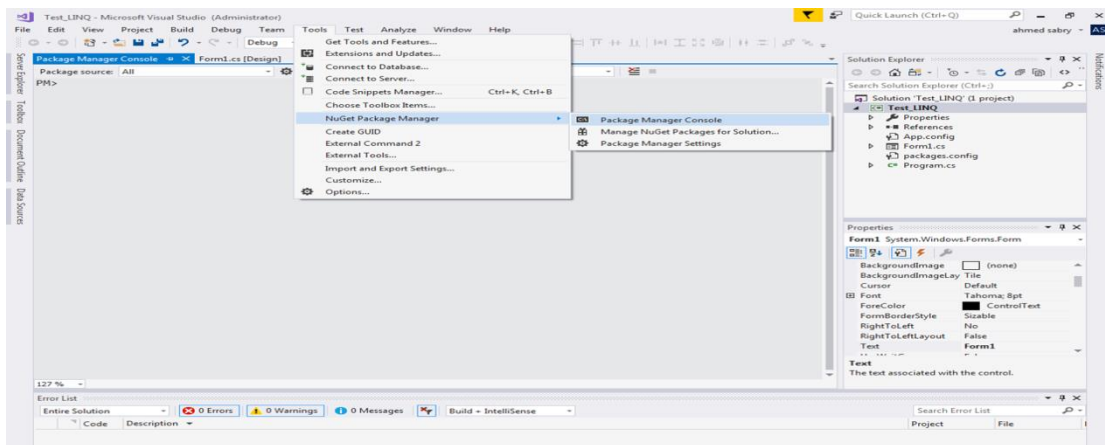

دليل المبرمج 2.1

database first entity framework core

: entity framework

هو اطار عمل يسمح بعمل نسخة من قاعدة البيانات داخل المشروع : حيث يقوم اطار العمل بتحويل قاعدة البيانات (sqlserver) على سبيل المثال الى class وتحويل الجداول الى كلاس ومن ثم يمكن عمل كائن من كلاس قاعدة البيانات ليكون هذا الكائن بمثابة قاعدة البيانات

① من قائمة tools نختار nuget package manager - ثم نختار package manager console



② نكتب فى شاشة الconsole الامر التالى

```
install-package entityframework
```

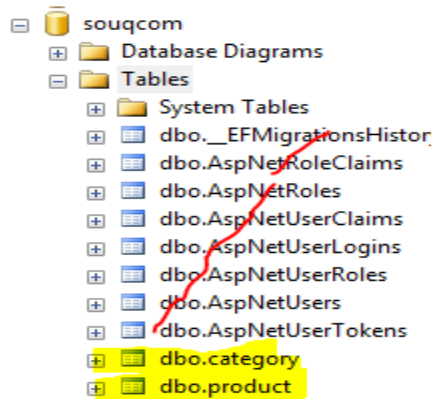
للم وللمسح الشاشة نكتب الامر clear

دليل المبرمج 2.1

Scaffold

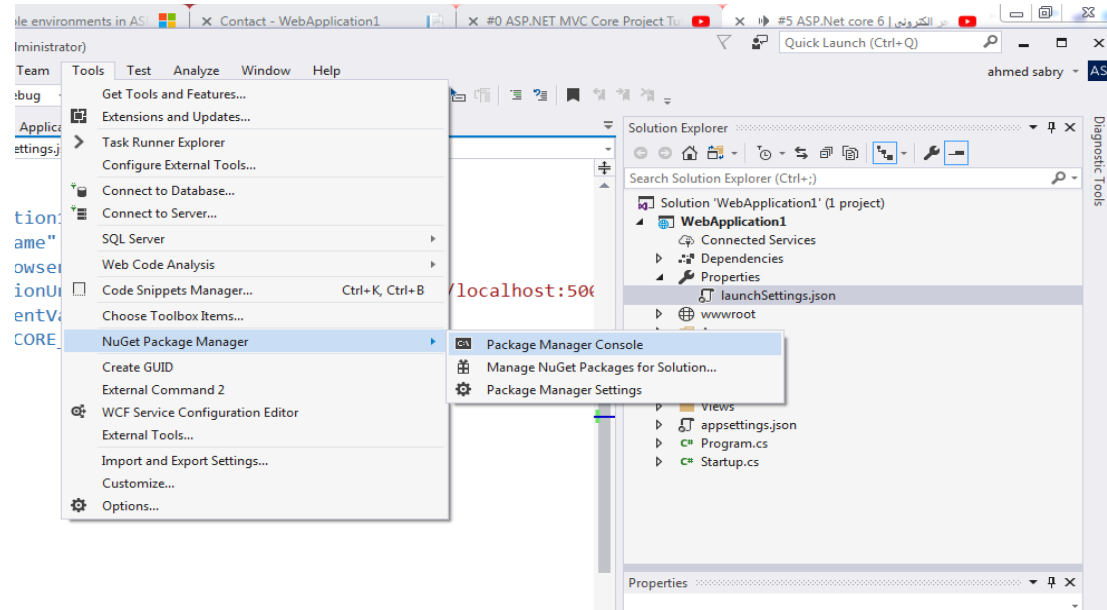
انشاء قاعدة بيانات اولاً داخل الـ sql server ثم تحويلها الى كلاسات داخل الـ model عن طريق Scaffold

على سبيل المثال لدينا فى الـ sql قاعدة بيانات باسم Souqcom تحتوى على جدولين جدول الـ category وجدول الـ product



الخطوات :

نقوم بفتح الـ package console manager



دليل المبرمج 2.1

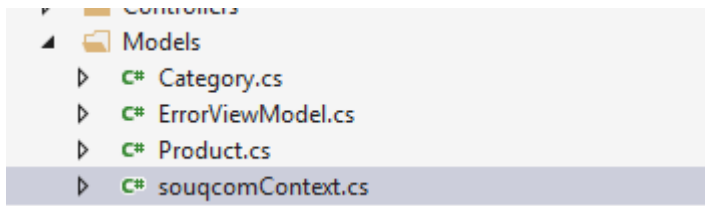
نكتب الامر التالي

```
Scaffold-DbContext "Server = اسم السيرفر;  
Database = اسم قاعدة البيانات  
Trusted_Connection = true ;"  
Microsoft.EntityFrameworkCore.SqlServer -OutputDir  
Models -force
```

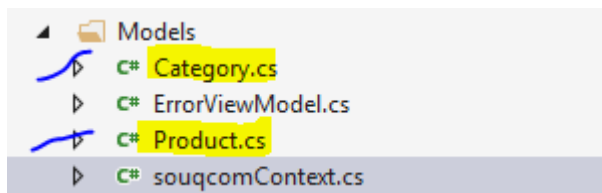
OutputDir Models : وظيفتها حفظ الجدول داخل الـ model

-force : وظيفتها التحديث في حالة وجود بيانات موجودة مسبقا

في الـ solution داخل ملف الـ Model سنجد كلاس تم انشاءه باسم قاعدة البيانات وبجانبه كلمة **Context**



وسنجد كلاس لكل جدول في قاعدة البيانات



اما الكلاس المسمى بنفس اسم قاعدة البيانات وبجانبه كلمة **Context** هو بمثابة dataset وهو يمثل قاعدة البيانات بالكامل

طريقة استخدام كلاس (قاعدة البيانات) في الاكشن

```
public IActionResult Index()  
{  
    souqcomContext db = new souqcomContext();  
    return View();  
}
```

دليل المبرمج 2.1

طريقة استخراج جدول من قاعدة البيانات من الـ object على شكل list ووضعها في متغير

نقوم بتعريف متغير من نوع var ثم يساوى ثم اسم الـ object ثم . ثم اسم الجدول ثم to list ، وبين اقواس الـ return نضع اسم المتغير الذى يحمل البيانات

```
public IActionResult Index()
{
    souqcomContext db = new souqcomContext();
    var reselt = db.Category.ToList();
    return View(reselt);
}
```

طريقة استدعاء جدول الـ Category فى الـ View على شكل List
نقوم بفتح صفحة الـ View الخاصة بالاكشن

نكتب @model ثم list من الجدول الذى نريد عرضه

```
@model List<Category>
```

طريقة عرض عناصر الـ model عن طريق foreach داخل الـ View
يمكن العرض فى جدول لعرض البيانات بشكل افضل

```
@foreach (var items in Model)
{
    <p>@items.Id</p>
    <p>@items.Name</p>
    <p>@items.Price</p>
}
```

دليل المبرمج 2.1

طريقة العرض من اكثر من جدول داخل الـ view

نقوم بعمل كلاس

```
public class Class1
{
    public IEnumerable<Product> Degree_1 { get; set; }
    public IEnumerable<Category> Degree_2 { get; set; }
}
```

يمكن استبدال IEnumerable بـ list : حيث تستخدم الـ IEnumerable للقراءة فقط بينما الـ list تسمح بالاضافة والازالة

```
public IActionResult google()
{
    souqcomContext db = new souqcomContext();
    Class1 cs = new Class1();
    cs.Degree_1 = db.Product.ToList();
    cs.Degree_2 = db.Category.ToList();
    return View(cs);
}
```

دليل المبرمج 2.1

صفحة الـ View

نكتب @model ثم اسم الـ namespace ثم . ثم اسم الكلاس

```
@model Test_Asp.Class1
```

```
@foreach (var items in Model.Degree_1)
{
    <tr>
        <td>@items.Id</td>
        <td>@items.Name</td>
        <td>@items.Description</td>
    </tr>
}
```

```
@foreach (var items in Model.Degree_2)
{
    <tr>
        <td>@item_product.Id</td>
        <td>@item_product.NameProduct</td>
        <td>@item_product.Description</td>
    </tr>
}
```

دليل المبرمج 2.1

الاتصال بقاعدة البيانات

طريقة الاتصال بقاعدة بيانات دون الاتصال بملف json



🔗 نفتح ملف ال Models ثم نختار اسم الكلاس الذى يحنوى على جميع الجداول **Context** في حالة اذا كانت قاعدة البيانات التى قمنا بعمل scaffolded لها تدعى souqcom سيكون اسم الكلاس كالتالى souqcomContext.cs داخل كلاس قاعدة البيانات نقوم بعمل مشيد فارغ 🔗

```
namespace Test_Asp.Models
{
    public partial class souqcomContext : DbContext
    {
        public souqcomContext()
        {
        }
    }
}
```

🔗 سنجد دالة OnConfiguring بداخلها دالة UseSqlServer نقوم بكتابة السيرفر واسم قاعدة البيانات

```
protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        #warning To protect potentially sensitive information in your
connection string, you should move it out of source code. See
http://go.microsoft.com/fwlink/?LinkId=723263 for guidance on
storing connection strings.
        optionsBuilder.UseSqlServer("Server =.\\SQL2008;
Database = writename_Database; Trusted_Connection = true ;");
    }
}
```

دليل المبرمج 2.1

Asp.netcor MVC

طريقة الاتصال بقاعدة بيانات من نوع sqlserver عن طريق جملة اتصال داخل ملف json

١- نقوم بوضع جملة الاتصال في ملف appsettings.json

مثال لسيرفر اسمه .\SQL2008
يضاف || قبل اسم السيرفر

"DefaultConnection": "Server =.\\SQL2008; Database = اسم قاعدة البيانات ; Trusted_Connection = true ;multipleactiveresultsets=true"

appsettings.json

```
{
  "ConnectionStrings": {
    //"DefaultConnection": "Server =SQL8004.site4now.net; Database =
    db_a8bf16_souqcom;User Id=db_a8bf16_souqcom_admin;Password=adminfhmu512256",
    "DefaultConnection": "Server =.\\SQL2008; Database = اسم قاعدة
    البيانات ; Trusted_Connection = true ;multipleactiveresultsets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Install Microsoft.EntityFrameworkCore.Tools

Scaffold

Scaffold-DbContext "Server =.\\SQL2008; Database = اسم قاعدة البيانات ; Trusted_Connection = true ;" Microsoft.EntityFrameworkCore.SqlServer - OutputDir Models -force

دليل المبرمج 2.1

٢. داخل ملف الـ startup.css نقوم بعمل حقن لكلاس قاعدة البيانات ونضع فى دالة `GetConnectionString` معرف الاتصال الذى قمنا بانشاءه فى ملف `appsettings`

Startup.cs

```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<SouqcomContext>(options =>
    options.UseSqlServer(
    Configuration.GetConnectionString("DefaultConnection")));
}
```

Singleton in HomeController

- طريقة عمل Singleton لل object الذى يحتوى على قاعدة البيانات
- نقوم بكتابة اسم الكلاس الذى يوجد داخل ال Models الذى يحتوى على قاعدة البيانات

```
namespace Test_Asp.Controllers
{
    public class HomeController : Controller
    {
        souqcomContext db ;
        public HomeController(souqcomContext _db )
        {
            db = _db;
        }
        public IActionResult Index()
        {
            var reselt = db.Category.ToList();
            return View(reselt);
        }
    }
}
```

فى الكنترول نقوم باستنساخ كائن داخلى من قاعدة البيانات على سبيل المثال نسميه db ثم نقوم بعمل مشيد يستقبل (بارمتر عباره عن اوبجيكت من كلاس قاعدة البيانات) ثم يساوى الاوبجيكت الداخلى من object

ملاحظة : البارمتر فى المشيد سيتم تمريره تلقائيا من خلال services (Singleton) التى قمنا بوضعها فى ملف Startup

دليل المبرمج 2.1

وظيفة هذا المشيد : فى الكنترول يوجد IActionResult نستطيع تعريف object بشكل مباشر دون الحاجة الى استنساخه

مثال بدون عمل Singleton

```
souqcomContext db = new souqcomContext();
var cats = db.Product.ToList();
```

مثال بعد عمل Singleton

```
public IActionResult Index()
{
    var reselt = db.Category.ToList();
    return View(reselt);
}
```

view

```
@model List<Category>
<!DOCTYPE html>

<body>
    <table>
        <thead>
            <tr>
                <th>id</th>
                <th>name</th>
                <th>description</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var items in Model)
            {
                <tr>
                    <td>@items.Id</td>
                    <td>@items.Name</td>
                    <td>@items.Description</td>
                </tr>
            }
        </tbody>
    </table>
```

دليل المبرمج 2.1

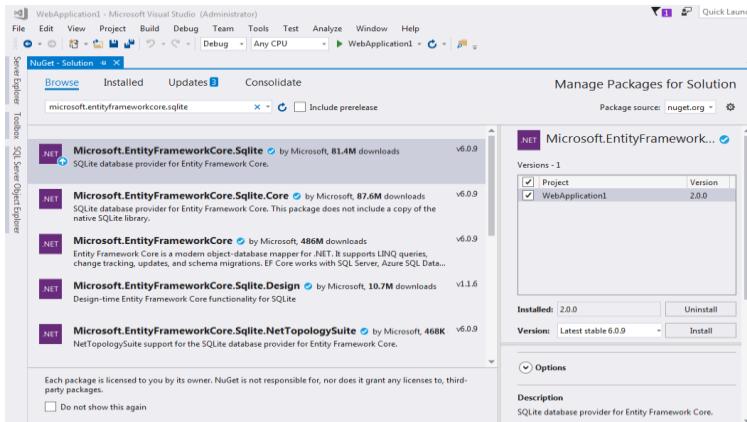
طريقة الاتصال ب sql lite

في ملف Startup داخل دالة الـ ConfigureServices يوجد جملة اتصال بـ sqlserver

```
services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(
Configuration.GetConnectionString("DefaultConnection")));
```

نقوم بالدخول على manage package

نقوم بتحميل microsoft.entityframeworkcore.sqlite ونختار الاصدار المناسب



نقوم بالتعديل على الدالة ونجعل الاتصال من خلال UseSqlite

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<CookiePolicyOptions>(options =>
    {
        // This lambda determines whether user consent for non-essential cookies is needed for a given request.
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });
    services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(
    Configuration.GetConnectionString("DefaultConnection")));
```

بعد التعديل

```
services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlite("Data source= register.db"));
```

بين اقواس الدالة UseSqlite : نكتب Data source ثم علامة يساوى ثم نكتب اى اسم نريده لقاعدة البيانات بشرط ان نتبعه بنقطة ثم db

عند تشغيل البروجيكت سيقوم تلقائيا بانشاء قاعدة بيانات بنفس الاسم وسنجد داخل ملف المشروع

دليل المبرمج 2.1

الاستعلام

LINQ

هي تقنية تساعدنا على الاتصال بقاعدة البيانات وعمل إجراءات مثل الاستعلام على سبيل المثال وهي تشبه الية الado.net ولاكن تختلف عن ال ado من حيث الاختصار فى الكود

البحث: search:

فورم البحث فى ال navbar

```
<form method="get" asp-action="index" class="d-flex" role="search">
  <input class="form-control me-2" type="search" placeholder="Search"
  name="id" aria-label="Search">
  <button class="btn btn-outline-success"
  type="submit">Search</button>
</form>
```

اكشن البحث

```
[HttpGet] //sarch by name
public async Task<IActionResult> index(string id)
{
    var result_qury = from x in db.Category select x;
    if (!string.IsNullOrEmpty(id))
    {
        result_qury = db.Category.Where(x => x.Name.Contains(id));
        ViewBag.query = result_qury.ToList();
    }
    //AsNoTracking موقت تخزين ذاكرة الى نحتاج ولا فقط القراءة نريد بمعنى
    return View(await result_qury.AsNoTracking().ToListAsync());
}
```

تعقيب : Task async تعنى التزامن وفى الretrn نكتب await حتى ينتظر لان كل الاوامر تنفذ فى نفس الوقت لذلك نجعله ينتظر حتى الانتهاء من اخر امر قبل الارجاع

AsNoTracking : لان التخزين موقت للقراءة فقط

(!string.IsNullOrEmpty(id)) : تعنى ان حقل ادخال البحث ليس فارغ

دليل المبرمج 2.1

تحليل :

قمنا بعمل متغير يجلب كل البيانات من الجدول ولاكن في حالة كان بارمتر (حقل الادخال في البحث) فارغ نجل المتغير يحمل استعلام بشرط **where** ويجلب بيانات البحث بدلالة الاسم على سبيل المثال حيث ان **Name.Contains(id)** تعنى ان البارمتر يحتوى على البارمتر

ViewBag.query : هو متغير يحمل قيمة الاستعلام فى حالة لم يكن حقل ادخال البحث فارغ ويمكن تمرير هذا المتغير الى الـ **view**

ملاحظة: **ViewBag.query**: يمكن تغير اسم **query** الى اى اسم تريده

صفحة الـ **view**

```
<h2 class="show_location"> show location </h2>
@{
    var myList = ViewBag.query;

    if (myList != null)
    {
        @foreach (var itemss in (List<Category>)ViewBag.query)
        {
            <iframe src="@itemss.Description" width="600" height="450"
style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-
when-downgrade"></iframe>
            <br>
        }
    }
}
```

فكرة جوجل ماب من خلال **iframe src** نتحقق اذا كان متغير الاستعلام لا يساوى فارغ نقوم في هذه الحالة بعمل **foreach** على هذا المتغير وليس الجدول بالكامل

ثم نقوم بعمل **iframe** داخل حلقة الـ **foreach** فى كل سطر فى الاستعلام سيقوم بانشاء **iframe** وقيمة المصدر الخاص بها يجلب تلقائيا من قاعدة البيانات حيث ان المتغير هو قائمة ليست من الجدول وبالتحديد من سطور الاستعلام فقط وفى كل مرة يتم تبديل الـ **src** بحقل موجود فى قاعدة البيانات يحمل لينك خريطة الخاصة بال **iframe**

دليل المبرمج 2.1

الإضافة

```
public IActionResult Create()
{
    return View();
}
```

```
[HttpPost]
public IActionResult Create(Category ca)
{
    db.Add(ca);
    db.SaveChanges();
    ViewBag.message = ca.Name + "save";
    return View(ca);
}
```

تعقيب :

اكشن الـ create تأخذ في البارمتر object من نوع class الـ table المراد انشاءه. هذا الكائن يحمل البيانات من حقول الـ input الموجودة بالنموذج Db تمثل قاعدة البيانات -- يمكن ان نكتب db.Category.Add (object) لاننا نريد اضافة بيانات هذا الكائن الى جدول الفئة التي تمثله حيث ان ca اصبح كائن يحمل متغيرات يمكن تمريرها الى جدول Category

لحفظ التغيرات في الـ sql يجب كتابة الامر التالي

```
db.SaveChanges();
```

بعد ان اصفنا الكائن الى قاعدة البيانات بالتحديد في جدول الـ Category التي نرسم لها ب(db) يمكن ان نقوم بتعريف متغير يساوي (اسم الفئة التي قمنا بانشاءها) عن طريق كتابة اسم الكائن على سبيل المثال ca او حسب الاسم الذي قمنا باختياره ثم . سنجد جميع حقول الذي يحملها الكائن ca نختار على سبيل المثال الحقل name

دليل المبرمج 2.1

الحذف

```
public IActionResult Delete_news(int id)
{
    var querydelete = db.Category.Find(id);
    db.Category.Remove(querydelete);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

صفحة ال-view

امام كل صنف زرار حذف وفي ال href كتبنا الكنترول ثم اسم الاكشن ثم رقم ال id لكي يحمل زرار الحذف دلالة الحذف

```
<table class="table">
  <thead class="table-dark">
    <tr>
      <th>id</th>
      <th>name</th>
      <th>Price</th>
      @*<th>Description</th>*@
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var items in Model)
    {
      <tr>
        <td>@items.Id</td>
        <td>@items.Name</td>
        <td>@items.Price</td>
        @*<td>@items.Description</td>*@
        <td>
          @* name cntrol/name action from cntrol / name id rows *@
          <a class="btn btn-danger"
href="/Home/Delete_news/@items.Id">Delete</a>
          <a class="btn btn-danger"
href="/Home/ubdate_news/@items.Id">edite</a>
        </td>
      </tr>
    }
  </tbody>
</table>
```

دليل المبرمج 2.1

التعديل

في حدث الدخول على الاكشن نستقبل رقم الid عن طريق الurl ثم نقوم بعمل كائن من نوع الكلاس يحمل نتيجة استعلام الصنف الذي نريد تعديله وفي الreturn نقوم بجلب هذا الكائن حتى يتم ملئ الحقول تلقائيا في صفحة (نموذج) التعديل

```
[HttpGet]
public IActionResult ubdate_news(int id)
{
    Category queryubdate = db.Category.Find(id);
    return View(queryubdate);
}
```

في حدث الخروج من الاكشن

```
[HttpPost]
public IActionResult ubdate_news(int id, Category ubdate_catogrey)
{
    db.Category.Update(ubdate_catogrey);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

تعقيب : الاكشن تحمل متغير للبحث بدلالة المتغير وتحمل object من نوع الجدول لكي يحمل التعديلات المطلوبة

```
<form asp-action="ubdate_news">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input type="hidden" asp-for="Id" />
    <div class="form-group">
        <label asp-for="Name" class="control-label"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Price" class="control-label"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Description" class="control-label"></label>
        <input asp-for="Description" class="form-control" />
        <span asp-validation-for="Description" class="text-danger"></span>
    </div>
    <div class="form-group">
        <input type="submit" value="Save" class="btn btn-default" />
    </div>
</form>
```


دليل المبرمج 2.1

استعلامات native sql

البحث

```
using System.Text; //to use StringBuilder
```

```
public string About()
{
    var posts = db.Category.FromSql("select * from Category").ToList();
    StringBuilder sb = new StringBuilder();

    foreach(var Post in posts)
    {
        sb.Append(Post.Name + "\n");
    }
    return sb.ToString();
}
```

الاضافة

```
public string About()
{
    int result = db.Database.ExecuteSqlCommand("INSERT INTO category
(Name, price) VALUES('شايين صن', '25'); ");

    StringBuilder sb = new StringBuilder();
    if (result > 0) { return "Add Done"; }
    else
        return "not found";
}
```

الحذف

```
public string About()
{
    int result = db.Database.ExecuteSqlCommand("delete from Category where
id =67");
    StringBuilder sb = new StringBuilder();
    if (result > 0) { return "delete"; }
    else
        return "not found";
}
```

التعديل

```
int result = db.Database.ExecuteSqlCommand("UPDATE category SET Name =
' حار زيت فول 'WHERE id = '68'; ");
```

دليل المبرمج 2.1

استبدال استخدام الـ Linq بـ sql native

```
public IActionResult Index()
{
    var reselt = db.Category.ToList();
    return View(reselt);
}
```

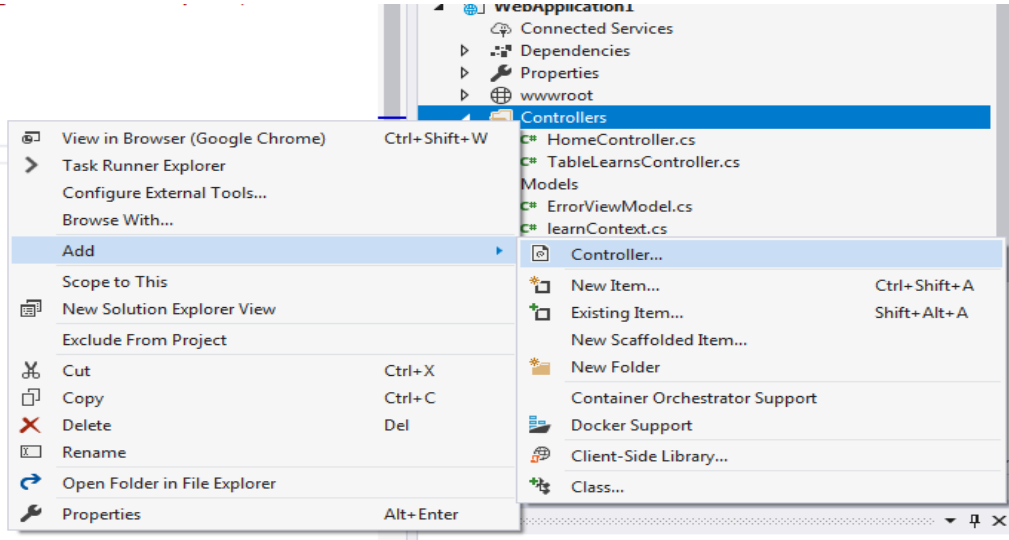
```
public IActionResult Index()
{
    var reselt = db.Category.FromSql("select * from Category").ToList();
    return View(reselt);
}
```

استخدام بارمتر في الاستعلام عن طريق interpolation

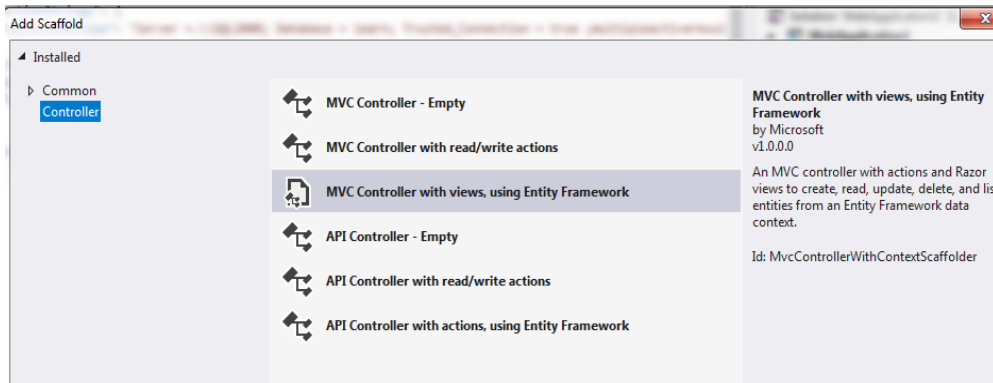
```
public IActionResult Delete_news(int id)
{
    int result = db.Database.ExecuteSqlCommand($"delete from Category
where id = {id}");
    return RedirectToAction("Index");
}
```

دليل المبرمج 2.1

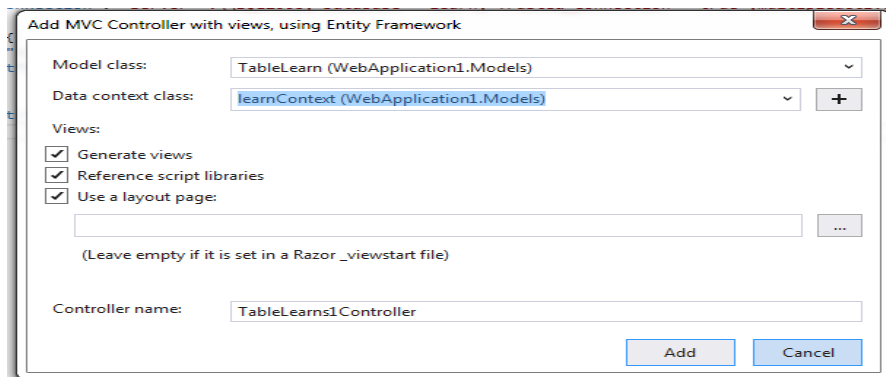
طريقة انشاء كنترول جاهز



نقوم بانشاء كنترول باستخدام Entity framework



Add نختار اسم الكلاس المراد ، ثم نختار اسم الكلاس الذي يحتوي على جميع الجداول ، ثم نضغط



دليل المبرمج 2.1

رفع صورة

في الكلاس المطلوب عمل انشاء سطر جديد له في قاعدة البيانات
نقوم بعمل متغير عام من نوع IFormFile ونقوم باعطاء اسم له (File)
ملحوظة هامة : يجب وضع السمة NotMapped فوق الحقل
ثم اضافة الـ namespace

```
using System.ComponentModel.DataAnnotations.Schema;
```

بشكل افتراضي ، ينشئ EF عمودًا لكل خاصية

تتجاوز السمة [NotMapped] هذا الاصطلاح الافتراضي. يمكنك تطبيق السمة [NotMapped] على خاصية واحدة أو أكثر لا تريد إنشاء عمود مقابل لها في جدول قاعدة البيانات.

```
namespace Test_Asp.Models
{
    public partial class Category
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Price { get; set; }
        public string Description { get; set; }
        public string ImgUrl { get; set; }

        [NotMapped]
        public IFormFile File { get; set; }
    }
}
```

نضغط كلك لمين على wwwroot ونختار new folder ونقوم بتسميته على
سبيل المثال (uploads)

دليل المبرمج 2.1

في الـ view الخاصة بعمل انشاء للصورة

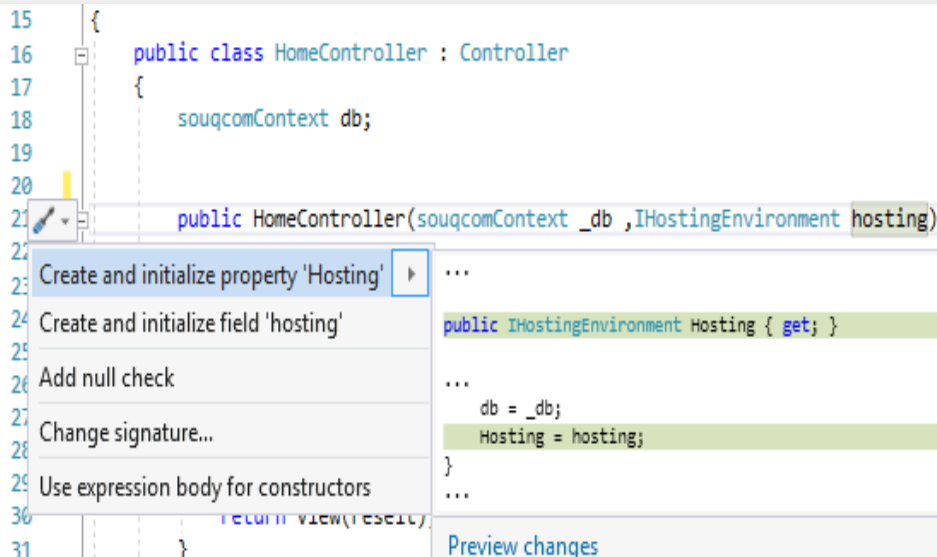
```
<div class="form-group">
<label asp-for="File" class="control-label">selecte your file </label>
<input type="file" asp-for="File" class="form-control" />
@*<span asp-validation-for="File" class="text-danger"></span>*@
</div>
```

نذهب الى المشيد لنضيف **IHostingEnvironment** ثم بارمتر
على سبيل المثال نسميه **hosting**
نضيف الـ **namespace** التالي

```
using Microsoft.AspNetCore.Hosting;

public HomeController(souqcomContext _db ,IHostingEnvironment hosting)
{
    db = _db;
}
```

ثم نقوم بعمل **hosting** **create and initialize [field/orproperty]**



```
15 {
16     public class HomeController : Controller
17     {
18         souqcomContext db;
19
20
21         public HomeController(souqcomContext _db ,IHostingEnvironment hosting)
22         {
23             ...
24             public IHostingEnvironment Hosting { get; }
25             ...
26             db = _db;
27             Hosting = hosting;
28         }
29         ...
30         return view(model);
31     }
}
```

Context menu options:

- Create and initialize property 'Hosting' (selected)
- Create and initialize field 'hosting'
- Add null check
- Change signature...
- Use expression body for constructors

Preview changes

دليل المبرمج 2.1

بعد عمل initialize سيضيف حقل وفي المشيد سيقوم بعمل single لل hosting يمكن استخدامه بدون الحاجة الى استنساخه

```
public IHostingEnvironment Hosting { get; } // للتعامل مع ملفات الروت
public HomeController(souqcomContext _db ,IHostingEnvironment hosting)
{
    db = _db;
    Hosting = hosting;
}
```

فى الاكشن فى وضع الـ HttpPost اى عندما نضغط ارسال

```
public IActionResult Create()
{
    return View();
}

[HttpPost]
public IActionResult Create(Category ca)
{
    if (ModelState.IsValid)
    {
        string fileName = string.Empty;
        if (ca.File != null)
        {
            string uploads = Path.
(Hosting.WebRootPath, "uploads");
            fileName = ca.File.FileName;
            string fulpath = Path.Combine(uploads, fileName);
            ca.File.CopyTo(new FileStream(fulpath, FileMode.Create));
        }
        db.Add(ca);
        db.SaveChanges();
        ViewBag.message = ca.Name + "save";
    }
    return View();
}
```

دليل المبرمج 2.1

تحليل

```
public IActionResult Create(Category ca)
```

الاكشن تستقبل كائن من نوع الكلاس المراد حفظ بياناته

```
if (ModelState.IsValid)
```

في حالة اذا كان قيم النموذج مرتبطة ارتباط منطقي بالنموذج

```
string fileName = string.Empty;
```

قمنا بعمل متغير (fileName) يساوى نص فارغ

```
if (ca.File != null)
```

نتحقق اذا كان هناك ملف تم رفعه

```
string uploads = Path.Combine(Hosting.WebRootPath, "uploads");
```

قمنا بعمل متغير يساوى مسار الملف

لكي يتم قراءة الكلاس Path لا بد من استخدام مجال الاسماء
Namespace الخاص بالتعامل مع الملفات

```
using System.IO; // للتعامل مع الملفات
```

Combine : تستخدم للدمج سنستخدمها لدمج المسار مع اسم الملف المرفوع

Hosting : او اسم الحقل الذي قمنا بانشاءه

```
public IHostingEnvironment Hosting { get; }
```

WebRootPath : WebRootPath, "uploads");
اسم المجلد الذي سيتم الحفظ بداخله

الان المتغير uploads يحمل المسار + اسم المجلد

```
fileName = ca.File.FileName;
```

نعود للمتغير ونجعله يساوى اسم الفايل المرفوع

```
string fulpath = Path.Combine(uploads, fileName);
```

دليل المبرمج 2.1

قمنا بعمل متغير (fulpath) يساوى دمج متغيرين المتغير الاول الذى يحتوى على المسار والمجلد والمتغير الثانى يساوى اسم الفايل المرفوع

```
ca.File.CopyTo(new FileStream(fulpath,
    FileMode.Create));
```

الكائن ثم . ثم اسم حقل الـ IFormFile ثم CopyTo

ثم الدالة new FileStream : تسقبل متغير نصى (اسم المتغير الذى يحتوى على المسار واسم الفولدر) وتستقبل enum تدعى FileMode نختار منها Create

يمكن ان نحفظ اسم الصورة في قاعدة البيانات في الجدول داخل الحقل الذى نريده

```
[HttpPost]
public IActionResult Create(Category ca)
{
    if (ModelState.IsValid)
    {
        string fileName = string.Empty;
        if (ca.File != null)
        {
            string uploads = Path.Combine(Hosting.WebRootPath, "uploads");
            fileName = ca.File.FileName;
            string fulpath = Path.Combine(uploads, fileName);
            ca.File.CopyTo(new FileStream(fulpath, FileMode.Create));
        }

        ca.Description = fileName;
        db.Add(ca);
        db.SaveChanges();
        ViewBag.message = ca.Name + "save";
    }

    return View();
}
```


Send mill gamil

```

public ActionResult SendEmail()
{
    return View();
}
[HttpPost]
public ActionResult SendEmail(/*string receiver,*/ string subject,
string message,string text_email)
{
    try
    {
        if (ModelState.IsValid)
        {
            var senderEmail = new MailAddress("bookfhmt022@gmail.com");
            var receiverEmail = new MailAddress("bookfhmt022@gmail.com"/*,
"Receiver"*/);

            var password = "eqeuwwslwvkps0aq";

            var sub = subject;
            var body = " المرسل: " + "\n" + text_email + "\n" +
" الرسالة عنوان: " + "\n" + message;
            var smtp = new SmtpClient
            {
                Host = "smtp.gmail.com",
                Port = 587,
                EnableSsl = true,
                DeliveryMethod = SmtpDeliveryMethod.Network,
                UseDefaultCredentials = false,
                Credentials = new NetworkCredential(senderEmail.Address,
password)
            };
            using (var mess = new MailMessage(senderEmail, receiverEmail)
            {
                Subject = subject,
                Body = body
            })
            {
                smtp.Send(mess);
            }
            return View();
        }
    }
    catch (Exception)
    {
        ViewBag.Error = "Some Error";
    }
    return View();
}

```

View page

```
@{
    ViewBag.Title = "SendEmail";
}
<h2> SendEmail </h2>
<br>
@if (ViewBag.Error != null)
{
    <h2> @ViewBag.Error </h2>
}
<form method = "post"
action = "SendEmail">
    <div class="container">
        @*<span class = "form-control-static"> Receiver:
        </span>
        <input class = "form-control"
            type= "text"
            name= "receiver"/> <br/>*@
        <span class="form-control-static"> email:</span>
        <input class="form-control"
            type="text"
            name="text_email"/>

        <span class="form-control-static"> Subject:</span>
        <input class="form-control"
            type="text"
            name="subject"/>

        <span class="form-control-static"> Message: </span>
        <input class="form-control"
            type="text"
            name="message"/>

        <input class="btn btn-primary"
            type="submit"
            value="Send"/>
    </div>
</form>
```

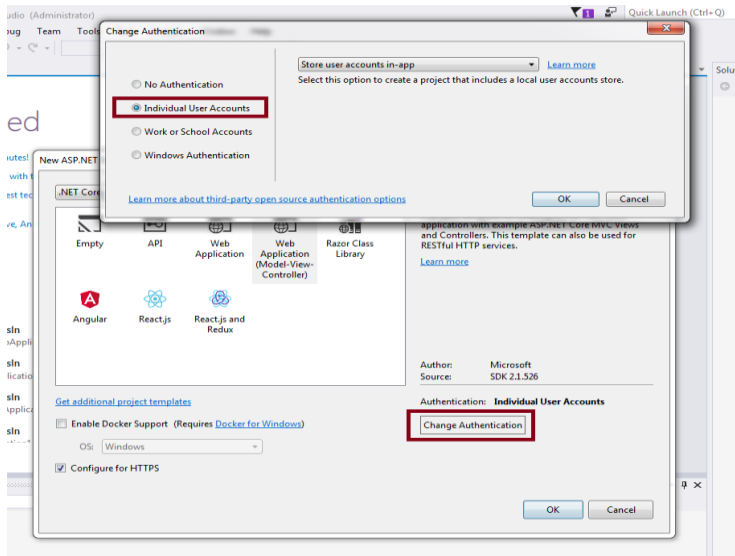
<https://www.youtube.com/watch?v=mte7LroYd74>

دليل المبرمج 2.1

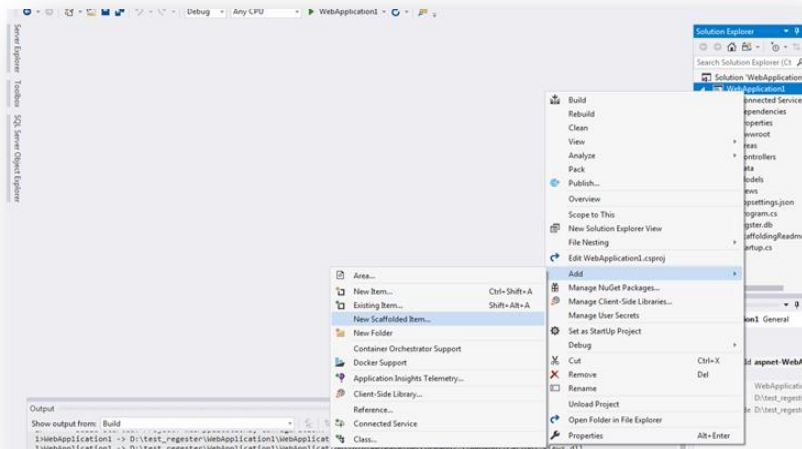
اضافة Identity



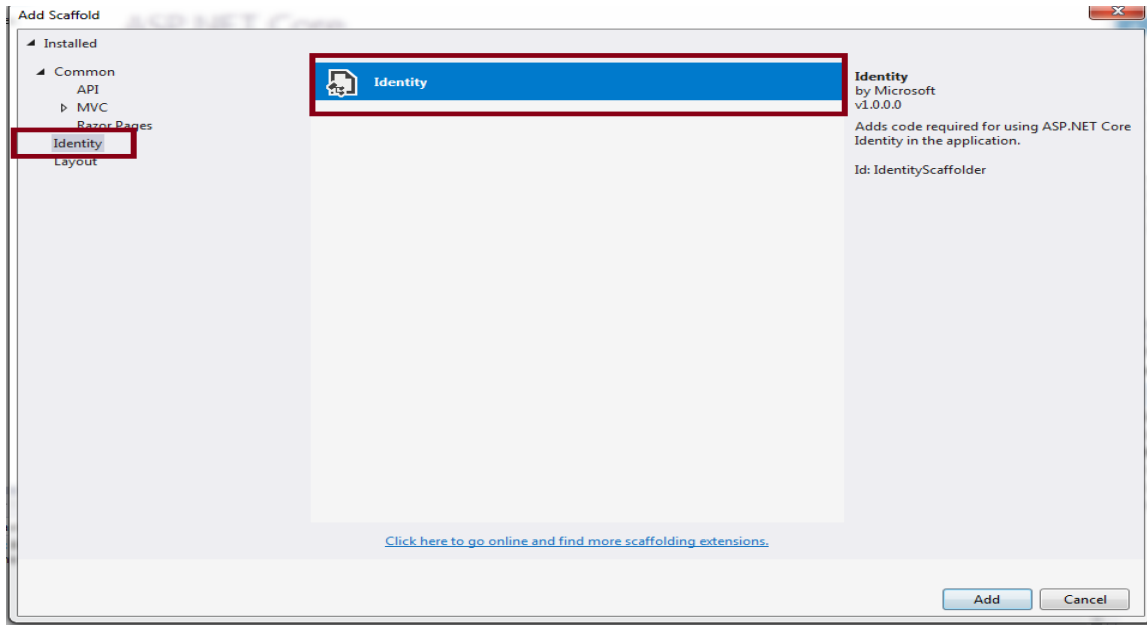
عند انشاء المشروع نقوم بتفعيل خاصية ال authentication
سنقوم على سبيل المثال باختيار حساب شخصي individual user



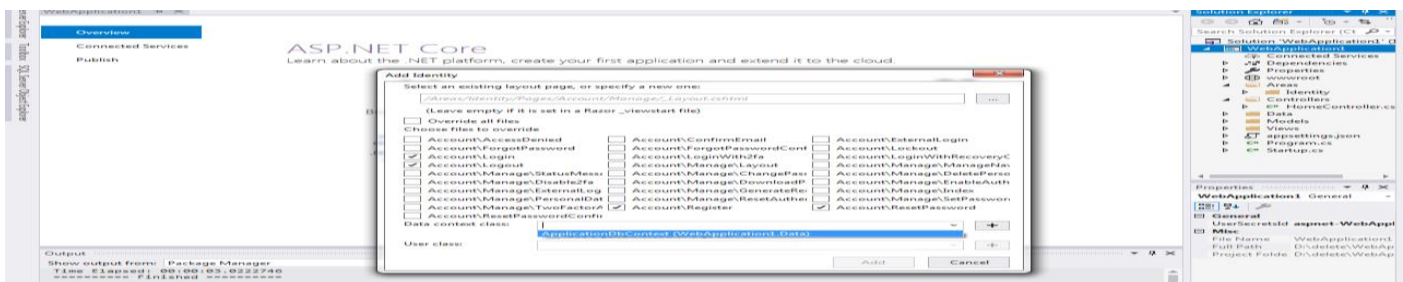
نقوم بالضغط كلك لمين على المشروع ومن Add نختار new scaffolded item



دليل المبرمج 2.1



نختار Identity ثم add ،، تظهر لنا شاشة نختار (add Identity) نقوم بوضع علامة صح امام الاضافات التي نريد تضمينها ثم نختار من الكمبيوتر كلاس ApplicationDbContext



في شاشة الconsole نكتب الكود التالي حتى يتم عمل Apply للـ migration

Update-Database

دليل المبرمج 2.1

تجربة التحقق من دخول مستخدم

إذا لم نكتب الامر Update-Database وقمنا بالدخول بعمل run للتطبيق سيطلب عمل apply migration نقوم بعمل apply

بعد ذلك ندخل على ال nav bar ونقوم بعمل تسجيل دخول ، بعد تسجيل دخول (اشتراك)

ندخل على اكشن الصفحة التي نريدها على سبيل المثال صفحة About ثم نتحقق من ان اسم المستخدم ليس فارغ بمعنى ان المستخدم قام بعمل log in وفي هذه الحالة سنقوم بعمل متغير يحمل اسم المستخدم الذي قام بتسجيل الدخول .

```
public IActionResult About()
{
    if (User.Identity.Name!=null)
    {
        ViewData["Message"] = "welcom \n" + User.Identity.Name;

        //ViewData["Message"] = "Your application description page.";
    }
    return View();
}
```

الان نقوم بالدخول على login سنلاحظ كتابة اسم المستخدم في ال nav bar بالاميل الذي قام بادخاله

WebApplication1 Home About Contact

Hello ahmed [redacted]@yahoo.com! Logout

About

welcom [redacted]@yahoo.com

دليل المبرمج 2.1

طريقة حماية بعض البيانات وعدم ظهورها الا بعد تسجيل دخول

```
@if (User.Identity.Name != null)
{
    <p>بنجاح الدخول تسجيل تم</p>
}
```

Authorize

يمكن ان نمنع الدخول الى هذه الاكشن الا اذا كان هناك مستخدم قام بعمل Login

```
[Authorize]
public IActionResult About()
{
    ViewData["Message"] = "welcom \n" + User.Identity.Name;
    return View();
}
```

تطبيق التأكد من تسجيل الدخول على جميع الاكشن وعمل استثناء لبعض الاكشن

تحليل : فوق كلاس الكنترول نكتب Authorize بمعنى ان كل الاكشن تتطلب تسجيل الدخول

```
namespace Test_Asp.Controllers
{
    [Authorize]
    public class HomeController : Controller
    {
        souqcomContext db;

        public IHostingEnvironment Hosting { get; } //الروت ملفات مع للتعامل
        public HomeController(souqcomContext _db ,IHostingEnvironment hosting)
        {
            db = _db;
            Hosting = hosting;
        }
        [AllowAnonymous]
        public IActionResult Index()
        {
            //Singleton => conectionstring in appsettings.json &services Singleton in
            Startup.cs
            var reselt = db.Category.ToList();
            return View(reselt);
        }
    }
}
```

نقوم بعمل استثناء الاكشن عن طريق AllowAnonymous

دليل المبرمج 2.1

تحليل اهم الجداول التي تم انشاءها بعد عمل apply migration

جدولAspNetUsers: يقوم بتخزين رقم تلقائي للمستخدم ثم اسم للمستخدم : عباره عن الامليل الذي قام بتسجيله

اسم الجدول AspNetUsers	
id	رقم معرف المستخدم
UserName	اسم المستخدم

اسم الجدول AspNetRoles	
id	رقم معرف الصلاحية
Name	اسم الصلاحية

جدولAspNetRoles: يقوم بتخزين رقم العملية في حقل id ثم اسم العملية في حقل Name حيث ان الرقم بمثابة مفتاح او رقم للعملية

اسم الجدول AspNetUserRoles	
UserId	رقم معرف المستخدم
RoleId	رقم معرف الصلاحية

جدول AspNetUserRoles: يقوم بتخزين (الرقم التلقائي الذي تم انشاءه للمستخدم س) ثم اسم مفتاح الصلاحية التي نريد تمريرها للمستخدم س

دليل المبرمج 2.1

Roles

مثال لعمل صلاحيات او اكسس لكل مستخدم بحيث يكون لكل مستخدم Roles خاصه به بعد عمل اشتراك او تسجيل في الموقع ستسجل هذه البيانات في جدولAspNetUsers ، وفي هذا الجدول في حقل الـ id سيتم انشاء رقم معرف لهذا المستخدم وفي جدولAspNetRoles في حقل الـ ID نقوم باعطاء رقم للصلاحيه ونقوم بتسمية هذه الصلاحيه على سبيل المثال ١ ← admin ٢ ← user في جدولAspNetUserRoles نضع الـ id الـ User ونضع ID الصلاحيه

في الاكشن الذاتى نريد فتحها بصلاحيه معينه نقوم بكتابة الامر التالى

```
[Authorize(Roles = "X" )]
```

المتغير X نقوم بتغيره الى اسم الصلاحيه

```
[Authorize(Roles = "Admin" )]
public IActionResult About()
{
    ViewData["Message"] = "Your application description page.";
    return View();
}
```

ويمكن ان نجعل الاكشن متاحة لأكثر من صلاحيه

```
[Authorize(Roles = "Admin,user")]
public IActionResult About()
{
    ViewData["Message"] = "Your application description page.";
    return View();
}
```


دليل المبرمج 2.1

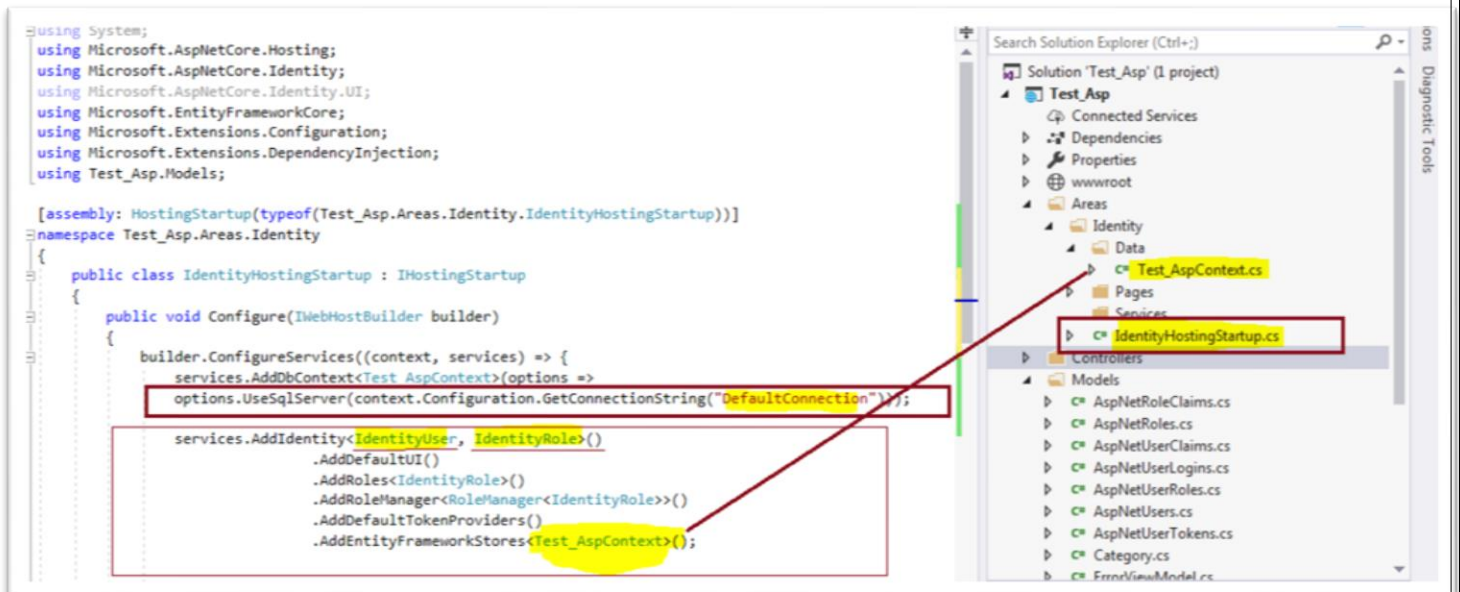
ملحوظة هامة : في اصدار 2.1 netcor. يجب اضافة خدمة ال Roles من اجل ان تعمل طريقة اضافة هذه الخدمة

من قائمة solution ← نختار Areas ثم ← Identity ← نختار ملف IdentityHostingStartup.cs نذهب الى دالة

`public void Configure(IWebHostBuilder builder)`

اسفل نص الاتصال نقوم باضافة الخدمة التي نريدها

```
services.AddIdentity<IdentityUser, IdentityRole>()
    .AddDefaultUI()
    .AddRoles<IdentityRole>()
    .AddRoleManager<RoleManager<IdentityRole>>()
    .AddDefaultTokenProviders()
    .AddEntityFrameworkStores<Test_AspContext>();
```



Test_AspContext : هو اسم الـ اسم الموجود في المسار التالي solution ← Areas ← Identity ←

Test_AspContext ← Data

دليل المبرمج 2.1

طريقة اضافة Session

مفهوم الـ

السيشن هو عبارة عن جلسة بين العميل والسيرفر او هو عبارة عن طريقة لتخزين المعلومات في متغيرات تُسمى (session variables) ويتم نقلها بين صفحات المواقع الإلكترونية المختلفة لتكون متاحة للاستخدام، ولا تخزن على جهاز العميل

يمكن ان نقوم بعمل وقت محدد للسيشن بمعنى عند دخول المستخدم بالاسم والباسورد المخصص له يقوم بعمل سيشن (كمتغير عام على مستوى جميع الصفحات) يحتفظ بمعلوماته وبعد دقيقة على سبيل المثال يحذف السيشن تلقائى فلا يمكن استدعائه الا بعد تسجيل الدخول وانشاءه من جديد

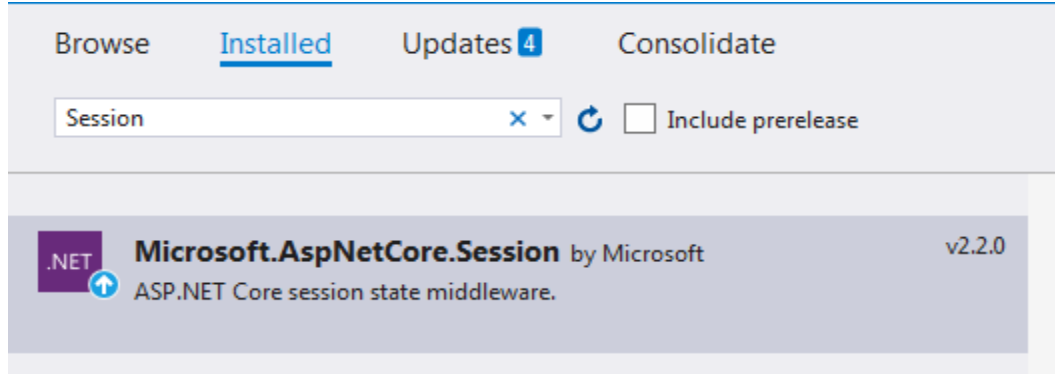
اما الكوكيز (cookies)

او ما يعرف بملفات الارتباط - تقوم بتخزين المعلومات في جهاز المستخدم ، حيث يمكن ان تخزن اختياراته او هواياته او كلمات المرور التى قام بتسجيلها ، وعند دخول العميل مرة اخرى الى الموقع يتم التعرف على العميل ومعرفة اختياراته من قبل السيرفر ومن ثم يقوم بتحميل هذه البيانات تلقائيا

فى الكوكيز يمكن للمستخدم ان يقوم بمسح جميع ملفات الكوكيز من المتصفح اما فى السيشن هو خاص بالسيرفر وبالتى لا يمكن للعميل مسح السيشن

دليل المبرمج 2.1

الخطوة الاولى نقوم بتحميل هذه الحزمة



فى دالة ال **Configure** نكتب

```
app.UseSession(); //i will UseSession
```

وفى دالة ال **ConfigureServices** نكتب

```
//-----Add_UseSession -*****
services.AddSession(options => { options.IdleTimeout =
    TimeSpan.FromMinutes(1); });
```

ملاحظة : يمكن تغير مدة ال **Session** من دقائق لايام

فى نفس الدالة بالتحديد فى ال **Configure** الخاصة بال **CookiePolicyOptions** ،
نقوم يتحويل القيمة **TRUE** الى **false**

```
services.Configure<CookiePolicyOptions>(options =>
{
    // This lambda determines whether user consent for non-essential
    cookies is needed for a given request.
    options.CheckConsentNeeded = context => false; //replace true--to
    false
});
```

نقوم بعمل اكش فى الكنترول ونسميه على سبيل المثال **LOGIN**

دليل المبرمج 2.1

```
public IActionResult login()
{
    return View();
}
```

في صفحة الVIEW نقوم بعمل حقلين للدخول بمثابة بارمتر وفي نهاية الفورم قمنا بعمل متغير يحمل رسالة

```
@{
    ViewData["Title"] = "login";
}
<h2>login</h2>
<h1
    style= "color :red ; font-size:medium ;text-align:center "
    >
    اولا الدخول تسجيل يتوجب الصفحة هذه على للدخول
</h1>
<form asp-action="login">
    <div class="form-group">
        <label class="control-label">username</label>
        <input name="na" class="form-control"/>
    </div>
    <div class="form-group">
        <label for="pa" class="control-label">userpassword</label>
        <input name="pa" class="form-control"/> <br/>
    </div>
    <input type="submit" value="login" class="btn btn-default"/>
    <p style=" color :red">@ViewData["message"]</p>
</form>
```

دليل المبرمج 2.1

في اكشن الكنترول عند الارسال نتحقق من ادخال المستخدم للبارمترات (الاسم والباسورد) ثم نقوم بعمل متغير يحمل نتيجة استعلام البحث ، ثم قمنا بعمل شرط للتحقق من عدد صفوف الاستعلام اذا كانت لا تساوى صفر بمعنى اذا كان هناك مستخدم بهذه البيانات

```
[HttpPost]
public IActionResult login(string na, string pa)
{
    if (!string.IsNullOrEmpty(na) & !string.IsNullOrEmpty(pa) )
    {
        var result_user = db.Category.Where(x => x.Name==(na) &
x.Description==(pa));
        /* ViewBag.result_user*/ var count_result =
result_user.Count();

        if ( count_result !=0 )
        {
            HttpContext.Session.SetString("name", "ahmed");
            HttpContext.Session.SetString("role", "admin");
            return RedirectToAction("Index");
        }
    }
    //else
    ViewData["message"] = " المدير بواسطة الا الصفحة هذه على الدخول يمكن لا ";
    return View();
}
```

في هذه الحالة نقوم بانشاء Session
ثم اعادة التوجه الى الصفحة الرئيسية RedirectToAction
طريقة لانشاء متغير في السيشن يحمل اسم وقيمة على سبيل المثال الاسم :
role والقيمة **admin**

```
HttpContext.Session.SetString("role", "admin");
```

وفي حالة وجود خطأ : نقوم باسناد قيمة للمتغير الذى قمنا بانشاءه مسبقا في
الفورم حتى يتم عرضه في الفورم وفي النهاية نكتب **return**
View();

دليل المبرمج 2.1

التحقق من السيشن قبل الدخول على صفحة معينة

فى اكشن الصفحة المراد منع المستخدم العادى من الدخول اليها الا بعد تسجيل الدخول بواسطة السيشن

نقوم بتعريف متغير من نوع (STRING) يحمل اسم القيمة عن طريق الدالة GETSTRING، ثم نتحقق من قيمة السيشن اذا كانت تساوى القيمة التى نريدها نقوم بعمل RETURN VIEW واذا لم توجد القيمة التى نريدها فى السيشن نقوم بتوجيه الى صفحة تسجيل الدخول

```
public IActionResult Create()
{
    string Session_name_user = HttpContext.Session.GetString("role"); //get - not_set
    if(Session_name_user=="admin")
        return View();
    else
        return RedirectToAction("login");
}
```

طريقة عمل لينك تسجيل خروج يقوم بمسح السيشن

نقوم بعمل دالة ونقوم بمسح السيشن عن طريق الدالة clear
ثم نقوم بعمل اعادة توجيه لصفحة تسجيل الدخول بعد مسح السيشن
ويمكن ان نقوم بعمل لينك للدخول

```
public IActionResult Delete_Session()
{
    HttpContext.Session.Clear();
    return RedirectToAction("login");
}
```

دليل المبرمج 2.1

طريقة حجب لينك فى جميع الصفحات عن طريق الـ Session
طريقة استدعاء الـ Session فى صفحة الـ Layout.cshtml
فوق صفحة الـ View نقوم باستخدام مكتبة السيشن

```
@using Microsoft.AspNetCore.Http;
<!DOCTYPE html>
```

بعد ذلك نقوم بعمل بحجب للينك الذى نريد اخفائه عن المستخدم العادى

```
@if (Context.Session.GetString("role") == "admin")
{
    <li class="nav-item">
        <a class="nav-link" asp-area="" asp-
controller="Home" asp-action="Delete_Session">
            Delete_Session
        </a>
    </li>
}
```

يمكن ان نتحقق اذا كان هناك مستخدم يملك اى صلاحية
يظهر له لينك لتسجيل الخروج

```
@if (Context.Session.GetString("role") != null)
{
    <li class="nav-item">
        <a class="nav-link" asp-area="" asp-controller="Home"
asp-action="Delete_Session">
            log_off
        </a>
    </li>
}
```

واذا لم يكن هناك مستخدم له صلاحية سيظهر لينك لتسجيل الدخول

```
else
{
    <li class="nav-item">
        <a class="nav-link" asp-area="" asp-controller="Home"
asp-action="login">
            log_in
        </a>
    </li>
}
```

دليل المبرمج 2.1

سيناريوهات حذف الجلسة

Action to remove All_session?

```
public IActionResult Delete_Session()  
{  
    HttpContext.Session.Clear();  
    return RedirectToAction("login");  
}
```

by key?remove specific session

```
HttpContext.Session.Remove("role");
```

Reset session value Null

```
Session["your_session"] = null;
```

دليل المبرمج 2.1

Data Annotations

هي طريقة لتقيد الكتابة داخل input حتى نتفادى الاخطاء
عند عمل submit



طريقة استخدامها :

نذهب الى الكلاس الذى نريد تطبيق مفهوم الـ Data Annotations عليه
فى مجال الاسماء نقوم باضافة مكتبة الـ DataAnnotations

```
using System.ComponentModel.DataAnnotations;
//add_DataAnnotations
```

```
[Required]
public string Name { get; set; }
```

فوق الحقل نقوم بكتابة [] بداخل الاقواس نكتب الخاصية التى نريدها

دليل المبرمج 2.1

الخاصية Required :

تفيد بان الحقل يجب ان لا يكون فارغ ولا يمكن ارسال الفورم عندما يكون هذا الحقل فارغ

وفي حالة محاولة المستخدم من الارسال ستظهر رسالة للمستخدم تفيد بان هذا الحقل مطلوب

The Name field is required.

يمكن عمل customize للرسالة الافتراضية

[Required(ErrorMessage = "مطلوب الاسم")]

الخاصية [StringLength(write_number)] :

تحدد عدد الحروف المسموح بادخالها :

```
[StringLength(2)]
public string Price { get; set; }
```

تحليل : قمنا على سبيل المثال بختيار الحد الاقصى لكتابة السعر رقمين او حرفين ، لو قمنا بكتابة اكثر من حرفين ستظهر هذه الرسالة

The field Price must be a string with a maximum length of 2.

يمكن ايضا التعديل على هذه الرسالة

```
[ StringLength (3,ErrorMessage = "عن السعر يزيد لا ان يجب",MinimumLength =2)]
public string Price { get; set; }
```

دليل المبرمج 2.1

الخاصية DataType :

تمكننا من تحديد نوع الحقل على سبيل المثال يمكن تحديد الحقل على انه حقل لادخال باسورد وبالتالي عند كتابة المستخدم فى هذا الحقل ستظهر النصوص على شكل char "نجوم"

```
[DataType(DataType.Password)]
public string Description { get; set; }
```

[EmailAddress] الخاصية

تجعل حقل الادخال مقيد بادخال ايميل فقط بمعنى يجب ان يحتوى حقل الادخال على تنسيق ايميل

```
[EmailAddress]
public string Description { get; set; }
```

phone الخاصية

تجعل الحقل يقبل ارقام فقط ، ولاكن بدون تحجيم لعدد الحروف ، على سبيل المثال لو قمنا بادخال ٠١٢ فقط سيقبل

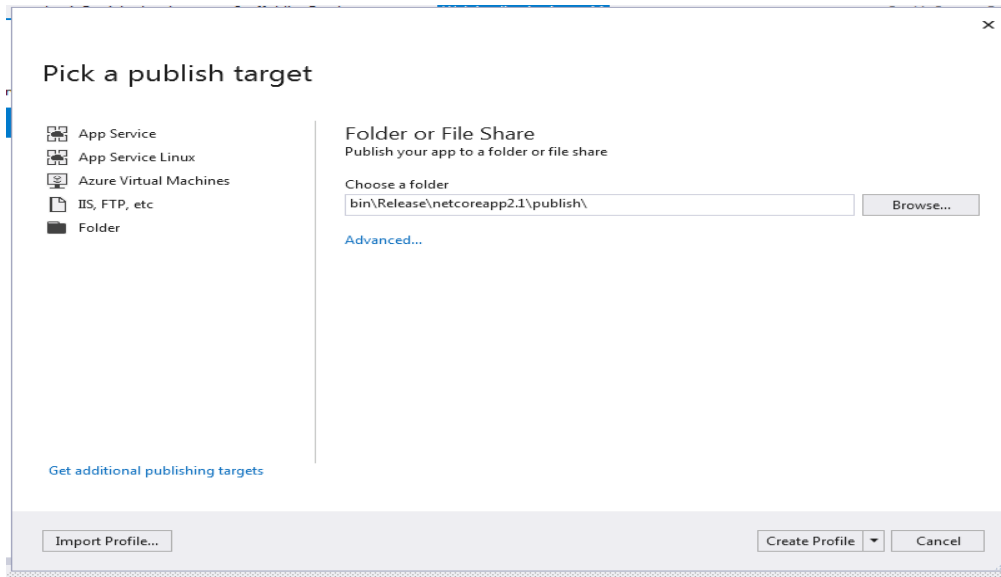
يمكن اضافة الخاصية StringLength بجانب الخاصية phone من اجل تحديد نوع الادخال (رقم) وكذلك عدد الحروف

```
[Phone]
[StringLength(11, ErrorMessage = " ولا 11 عن الرقم يزيد لا ان يجب ", MinimumLength = 11)]
public string Description { get; set; }
```

دليل المبرمج 2.1

طريقة نشر الموقع

نضغط كلك لمين على اسم المشروع ونختار publish ثم نختار folder ثم نحدد مكان الحفظ



بعد ذلك نضغط على configuration ونحدد مكان الحفظ ثم نضغط publish

Publish

Publish your app to Azure or another host. [Learn more](#)

FolderProfile

Publish

[New Profile...](#)
[Actions](#)

Target Location	bin\Release\netcoreapp2.1\publish\	Configure...
Delete Existing Files	False	
Configuration	Release	

Continuous Delivery

Automatically publish your application to Azure with continuous delivery

[Configure](#)

دليل المبرمج 2.1

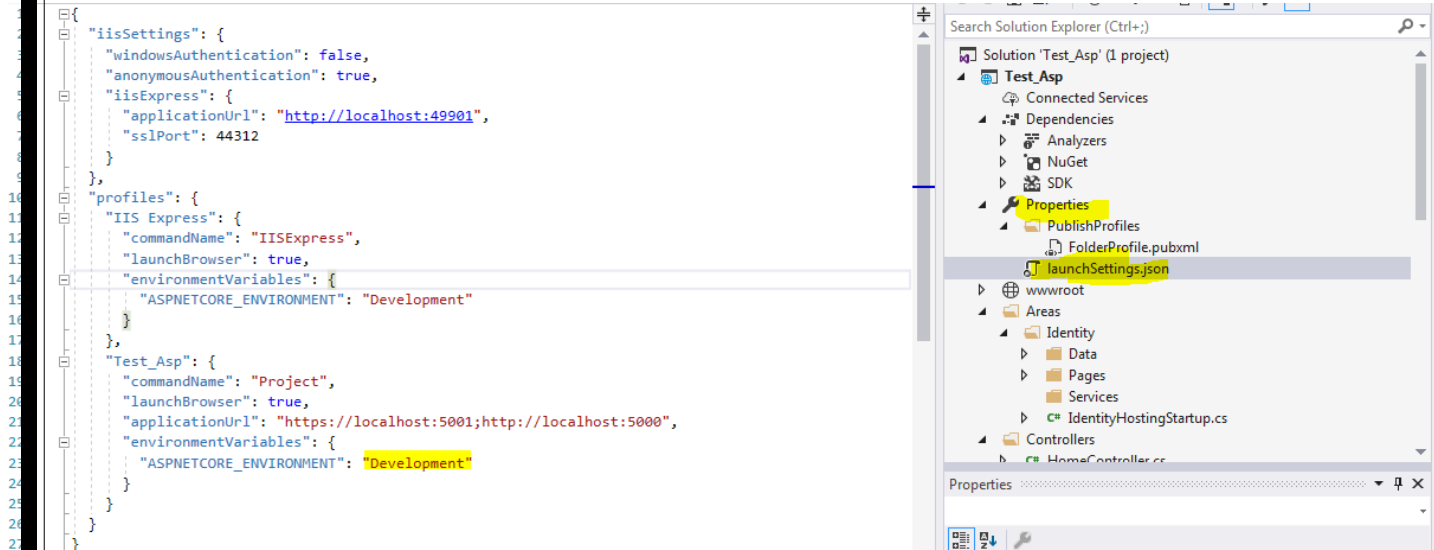
إذا اردنا تضمين قاعدة بيانات **sqlite** على الاستضافة نذهب الى مجلد الملف الاصلى ثم نسخة **copy** من ملف قاعدة البيانات الذى ينتهى بـ **db** ثم نلصقه داخل ملف الـ **publish**

عند الرفع على الاستضافة نقوم بعمل ضغط لجميع ملفات التى تم انتاجها عند عمل **publish** بالاضافة الى قاعدة البيانات التى قمنا بوضعها **pest** داخل مجلد الـ **publish** ثم نقوم بعمل فك الضغط من وحذف ملف الضغط

الان المشروع قابل للنشر والاستخدام

ملاحظات :

في الـ **solution** ← **Properties** ← يوجد ملف **launchSettings.json** خاص بالمود الخاص بالتطبيق هل هو في وضع التطوير ام في مرحلة الانتاج
يمكن تغيير كلمة **Development** الى (انتاج) **production** حتى لا يتم عرض الاخطاء للمستخدم العادى في حالة وجود خطأ



دليل المبرمج 2.1

اضافات :

طريقة العرض عن طريق view model

نلجاء الى هذه الطريقة عندما نريد اضافة حقول ليست موجودة في class واحد

مثال

```
Class Category
namespace Test_Asp.Models
{
    public partial class Category
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Price { get; set; }
        public string Description { get; set; }

        [NotMapped]
        public IFormFile File { get; set; }
    }
}
```

إذا اردنا اضافة حقول ليست موجودة في الكلاس

نقوم بعمل كلاس يحتوى على (class+ public var)

← متغير من نوع الكلاس الذى نريد عرضه care

← بالاضافة الى المتغيرات التى نريد اضافتها phone

على سبيل المثال نريد اضافة متغير لرقم الهاتف

دليل المبرمج 2.1

مثال

```
namespace Test_Asp.Models
{
    public class ClassA
    {
        ✓ public Category cate { get; set; } // لكي يحمل قيم الكلاس
        ✓ public string phone { get; set; }
    }
}
```

في الكنترولر

قمنا بعمل object من الكلاس الاصلى وقمنا باسناد قيم حقوله

ثم قمنا باستنساخ object من الكلاس الجديد الذى يحتوى على (class+var) الذى قمنا بتسميته على سبيل المثال classA

وفى المتغير الذى قمنا بانشاءه من نوع الكلاس (cate) نقوم باسناد قيم الـ object له

ثم قمنا باسناد قيمة للمتغير الذى نريد اضافته (phone)

```
[HttpGet]
public IActionResult Index( )
{
    Category c = new Category();
    c.Id =1;
    c.Name = "ahmed";
    c.Price = "25";
    c.Description = "verygood";
    ClassA class_A = new ClassA()
    {
        cate =c,
        phone = " my phone is : 01158421798"
    };
    return View(class_A);
}
```

دليل المبرمج 2.1

وفي الـ `return View` قمنا بارجاع الـ `view` الجديد من الكلاس الذي يحتوى على اوبجكت الكلاس القديم بالاضافى الى حقل او حقل مثل الحقل `phone` في صفحة العرض نقوم بكتابة `@model` مع مراعاة حالة الاحرف صغيرة ثم اسم الـ `namespace` ثم اسم الكلاس الجديد

```
@model Test_Asp.Models.ClassA
<link href="~/style/StyleSheet.css" rel="stylesheet" />
<h1> @Model.phone</h1>
<p>@Model.cate.Id </p>
<p>@Model.cate.Name </p>
<p>@Model.cate.Price </p>
```

ملاحظات :

بدل من ان نقوم باستنساخ كائن من `class` الـ `Category` ثم نقوم باستناد القيم يدويا

```
Category c = new Category();
c.Id = 1;
```

يمكن ان نقوم بعمل حقل في الفورم تبعت قيم للبارمتر `ca`

```
public IActionResult Index(ClassA ca)
```

```
public IActionResult Index(ClassA ca)
{
    ClassA class_A = new ClassA()
    {
        ✓ cate = ca.cate ,
        phone = " my phone is : 01158421798"
    };
    return View(class_A);
}
```

دليل المبرمج 2.1

وبداخل الكائن **class_A** نقوم بإسناد الحقل الذي نريده من البارمتر (ca.cate) للمتغير ← cate الذي قمنا بإنشائه من نوع class في صفحة الview : قمنا بإنشاء موديل يحمل الكلاس الذي يحتوى على حقل (كلاس اخر) ومتغير

```
@model Test_Asp.Models.ClassA

<link href="~/style/StyleSheet.css" rel="stylesheet" />

<div class="row">
  <div class="col-md-4">
    <form asp-action="test_cate">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="form-group">
        <label asp-for="cate.Id" class="control-label"></label>
        <input asp-for="cate.Id" class="form-control" />
        <span asp-validation-for="cate.Id" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="cate.Name" class="control-label"></label>
        <input asp-for="cate.Name" class="form-control" />
        <span asp-validation-for="cate.Name" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="cate.Price" class="control-label"></label>
        <input asp-for="cate.Price" class="form-control" />
        <span asp-validation-for="cate.Price" class="text-danger"></span>
      </div>
      <div class="form-group">
        <input type="submit" value="Create" class="btn btn-default" />
      </div>
    </form>
  </div>
</div>
```

قمنا بربط الفورم باكشن اسمه (على سبيل المثال test_cate)

قمنا بعمل حقول مرتبطة بمتغيرات الموجودة في الموديل

Cate هو حقل من نوع كلاس الcategory والكلاس ClassA يحتوى على هذا الحقل وبالتالي يمكن الوصول الى متغيرات الcategory عن طريق الحقل Cate

دليل المبرمج 2.1

بعد الضغط على ارسال سيتم التوجه الى الاكشن المربوط بالفورم

```
public IActionResult test_cate(ClassA ca )//ca includ
var nameis (cate) =>type class categorey
{
    ClassA class_A = new ClassA()
    {
        cate= ca.cate,
        phone = "01158421798"
    };
    return View(class_A);
}
```

وفي صفحة الاكشن الذى سيتم الانتقال اليها بعد تمرير المتغيرات نقوم بعرض المتغيرات المرسله

```
@model Test_Asp.Models.ClassA
<link href="~/style/StyleSheet.css" rel="stylesheet" />
<h1> @Model.phone</h1>
<p>@Model.cate.Id </p>
<p>@Model.cate.Name </p>
<p>@Model.cate.Price </p>
```

طريقة للتحكم فى الالوان عند العرض

```
@if (Model.cate.Id > 5)
{<p style="color:red">@Model.cate.Id </p>}
else{<p style="color:dodgerblue">@Model.cate.Id </p>}
<p>@Model.cate.Name </p>
<p>@Model.cate.Price </p>
```

مثال اخر للتلوين بشرط

```
@foreach (var items in Model)
{
    <tr>
        @if (items.Id > 2)
        { <td style="color:red">@items.Id</td> }
        else
        { <td>@items.Id</td> }
    }
```

دليل المبرمج 2.1

عند التحقق من رقم قيمة من نوع "string" يجب ان نقوم اولا بتحويلها الى قيمة رقمية حتى نستطيع قراءة قيمتها

```
@if ( Decimal.Parse(items.Price) > 22)
{
    <td style="color:red">@items.Price</td>
}
else
{
    <td>@items.Price</td>
}
```

اذا كان المتغير الذى نتحقق منه يمكن ان يحتوى على رقم عشرى يجب تحويله الى Decimal وليس Int حتى يتم قراءة الرقم العشرى

يمكن تغيير لون السطر بالكامل عن طريق اسم الكلاس ، بالاعتماد على قيمة خلية بدلالة رقم السطر

```
@{
    int xc = -1;
}

@foreach (var items in Model)
{
    *@ السطر الاول يبدأ م من رقم صفر وفى كل لفة سيزيد واحد
    xc++;
    *@ فى حلقة التكرار فى كل لفة المتغير هيزود على قيمته واحد
    <tr class="pixel">
        @if (items.Id > 2)
        {
            <td style="color:red">@items.Id</td> }
        else
        {
            <td>@items.Id</td>
        }
        <td>@items.Name</td>

        @if (Decimal.Parse(items.Price) > 50)
        {
            <td style="color:red">@items.Price</td>
            <script>
document.getElementsByClassName("pixel")[@xc].style.backgroundColor = "yellow";
            </script>
        }
        else
        {
            <td>@items.Price</td>
        }
    }
}
```

دليل المبرمج 2.1

ملخص الاتصال بقاعدة البيانات

يوجد صيغة باستعمال id و pass
ويوجد صيغة بدون الحاجة الى اسم المستخدم وكلمة المرور

⚙️ .NET Framework Data Provider for SQL Server

Standard Security

```
Server=myServerAddress; Database=myDataBase; User Id=myUsername;  
Password=myPassword;
```

SQL Server 2008

Trusted Connection

```
Server=myServerAddress; Database=myDataBase; Trusted_Connection=True;
```

```
"DefaultConnection": "Server =SQL8002.site4now.net; Database =  
writenameDatabase;User Id=writeid;Password=writepass"
```

```
"DefaultConnection": "Server =.\SQL2008; Database = اسم قاعدة البيانات;  
Trusted_Connection = true ;multipleactiveresultsets=true"
```

```
Scaffold-DbContext "Server = .\SQL2008; Database  
= اسم قاعدة البيانات; Trusted_Connection = true ;"  
Microsoft.EntityFrameworkCore.SqlServer -  
OutputDir Models -force
```

دليل المبرمج 2.1

Buy now with **PayPal**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

<!--https://www.youtube.com/watch?v=XOO_Q6eC4ao&list=PLUC-ItjDJLmN-
6PnzaU5CKf-4tYsyPC2d&index=36-->

<form action="https://www.sandbox.paypal.com/cgi-bin/webscr"
method="post">
<!-- Identify your business so that you can collect the payments. -->
<input type="hidden" name="business" value="bookfhmto23@gmail.com">
<!-- Specify a Buy Now button. -->
<input type="hidden" name="cmd" value="_xclick">
<!-- Specify details about the item that buyers will purchase. -->
<input type="hidden" name="item_name" value="برجر ع الفحم">
<input type="hidden" name="amount" value="2"> <!--pierce-->
<input type="hidden" name="currency_code" value="USD">
<!-- Display the payment button. -->
<input type="image" name="submit" border="0" src="paypal.png" alt="Buy
Now">

</form>

</body>
</html>
```