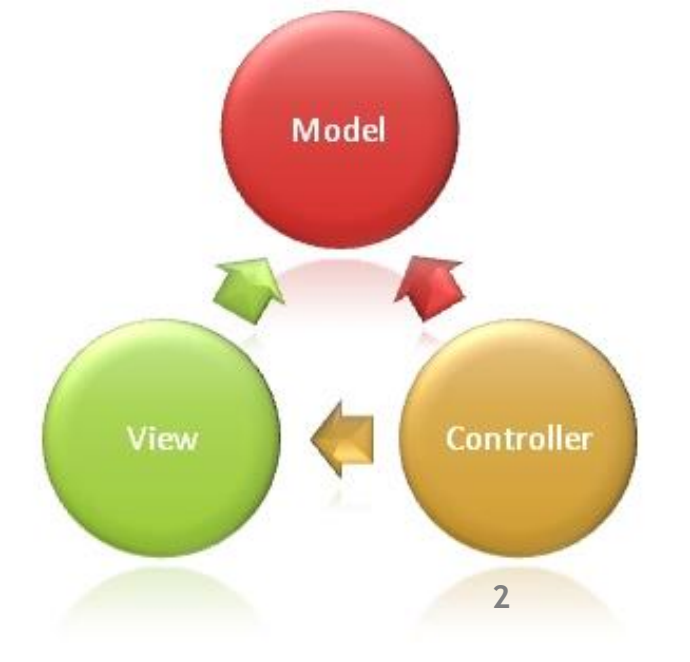
جامعة حماه
الكلية التطبيقية بحماه

MVC (Model View Controller) *

MVC (Model View Controller) *

- هو مبدأ أو نموذج معماري architectural pattern
- يستخدم للتعامل مع واجهات المستخدم
- يعتمد نمط MVC على مبدأ فصل الطبقات
- يقسم الى ثلاثة اقسام كالتالي



* model:

- هو جوهر التطبيق فهي عباره عن مجموعة البيانات data في التطبيق(قائمة من سجلات قاعدة البيانات)
- يكون مسؤولًا عن استرجاع وتخزين البيانات من قاعدة البيانات.

مثلاً لو لدينا تطبيق لعرض موديلات السيارات. كل المعلومات عن السيارة مثل(الماركة، اللون ..وغيرها) تعتبر بيانات ويتم تخزينها في class .

View :

- عبارة عن الواجهة الظاهرة لمستخدم التطبيق.
- مسؤول عن عرض البيانات مثل (عرض سجلات قاعدة البيانات)
- غالباً يكون محتواه ناتج من البيانات التي يجلبها الـ Model-

* controller :

- وحدة تحكم تعالج المدخلات مثل (معالجة سجلات قاعدة البيانات)
- تعتبر مسؤولة عن معالجة تفاعلات المستخدم مع التطبيق حيث يقوم المتحكم بقراءة البيانات من الـ (Viewمدخلات المستخدم) وإرسال المدخلات إلى الـ Model
- يعتبر الرابط أو حلقة الوصل بين ال (model , view) أي بين البيانات والواجهات .

يقوم بتحديث ال model حين يُدخل المستخدم بيانات جديدة إلى ال view .

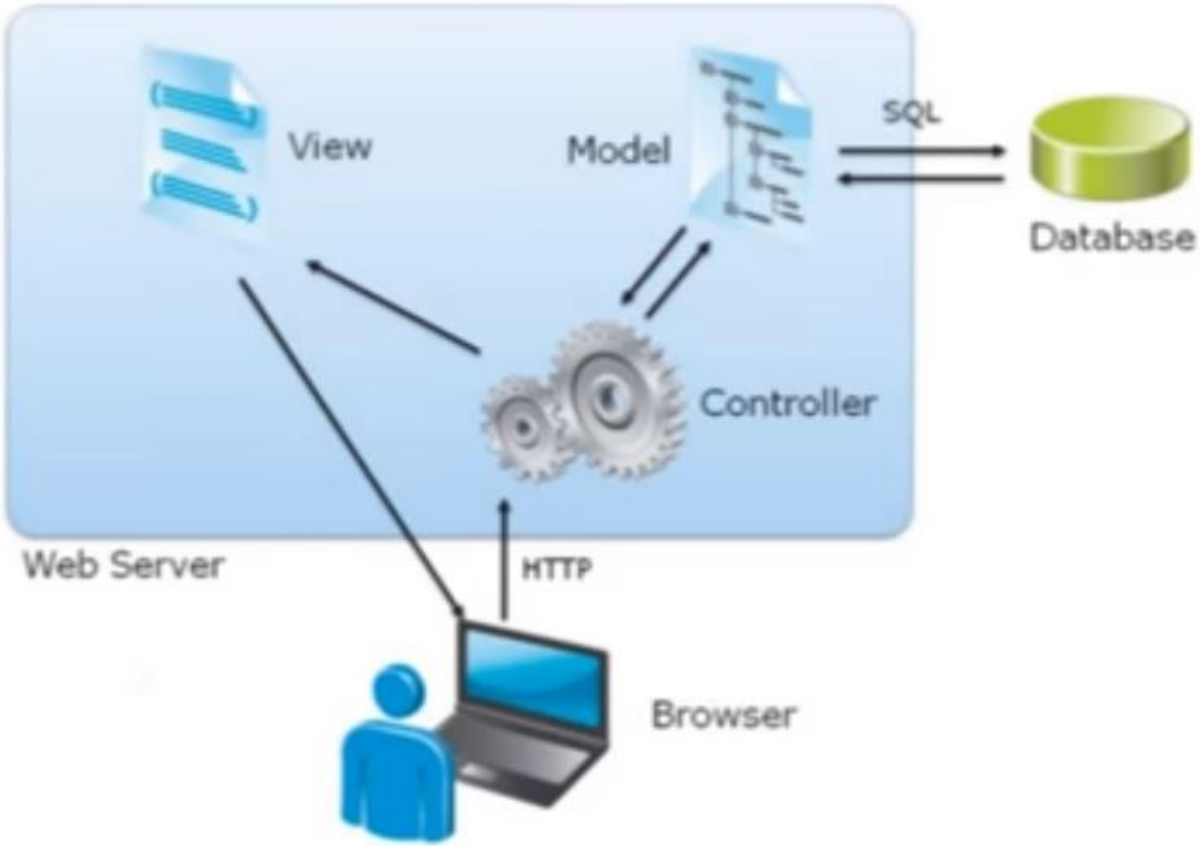
* عندما يقوم المستخدم بفتح المتصفح وطلب صفحة موجودة على ال web server يقوم المتصفح عن طريق بروتوكول http بالبحث عن الصفحة في السيرفر
* بفرض أن المستخدم قام بالبحث عن صفحة employees list
* هذا يعني أن controller يكون بهذا الشكل

Employees/list

* ال controller يأتي بالبيانات من قائمة الموظفين

ثم يقوم بعرضها في الview ضمن جدول او حسب وسوم الhtml

MVC كيف تعمل

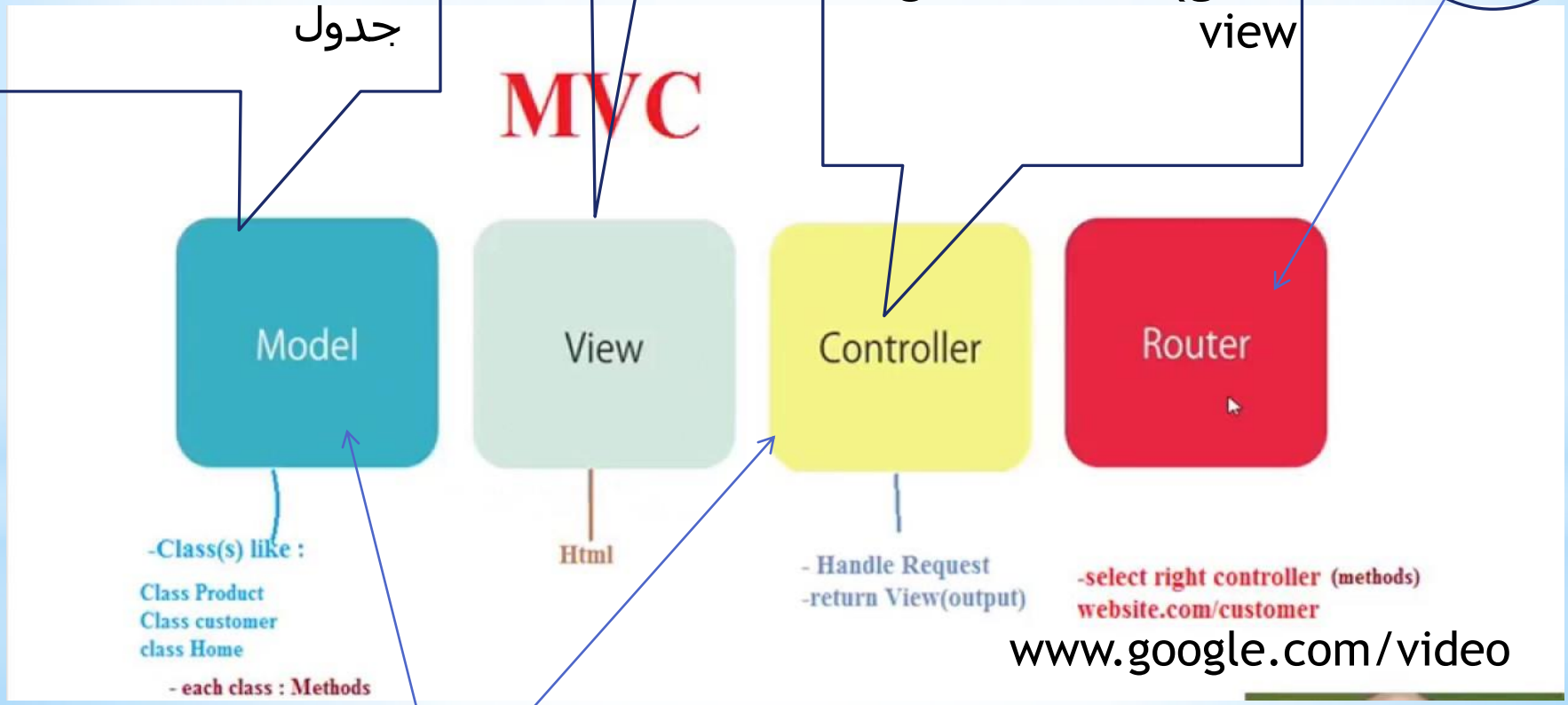

View
Model
SQL
Database
Controller
Web Server
HTTP
Browser

هي عبارة عن Class.
اضع الكلاسات التي احتاجها
واضع ضمن الكلاس المتغيرات (حقول الجدول)
ممكن ان لا تكون جدول

نكتب به كود C#
ينفذ عند العميل

عبارة عن method
كل classيرتبط ب method
الخاصة به وظيفته تنفيذ الأكواد (إضافة ،حذف تعديل) يعيد قيمة من view

عامل مساعد لشكل الرابط



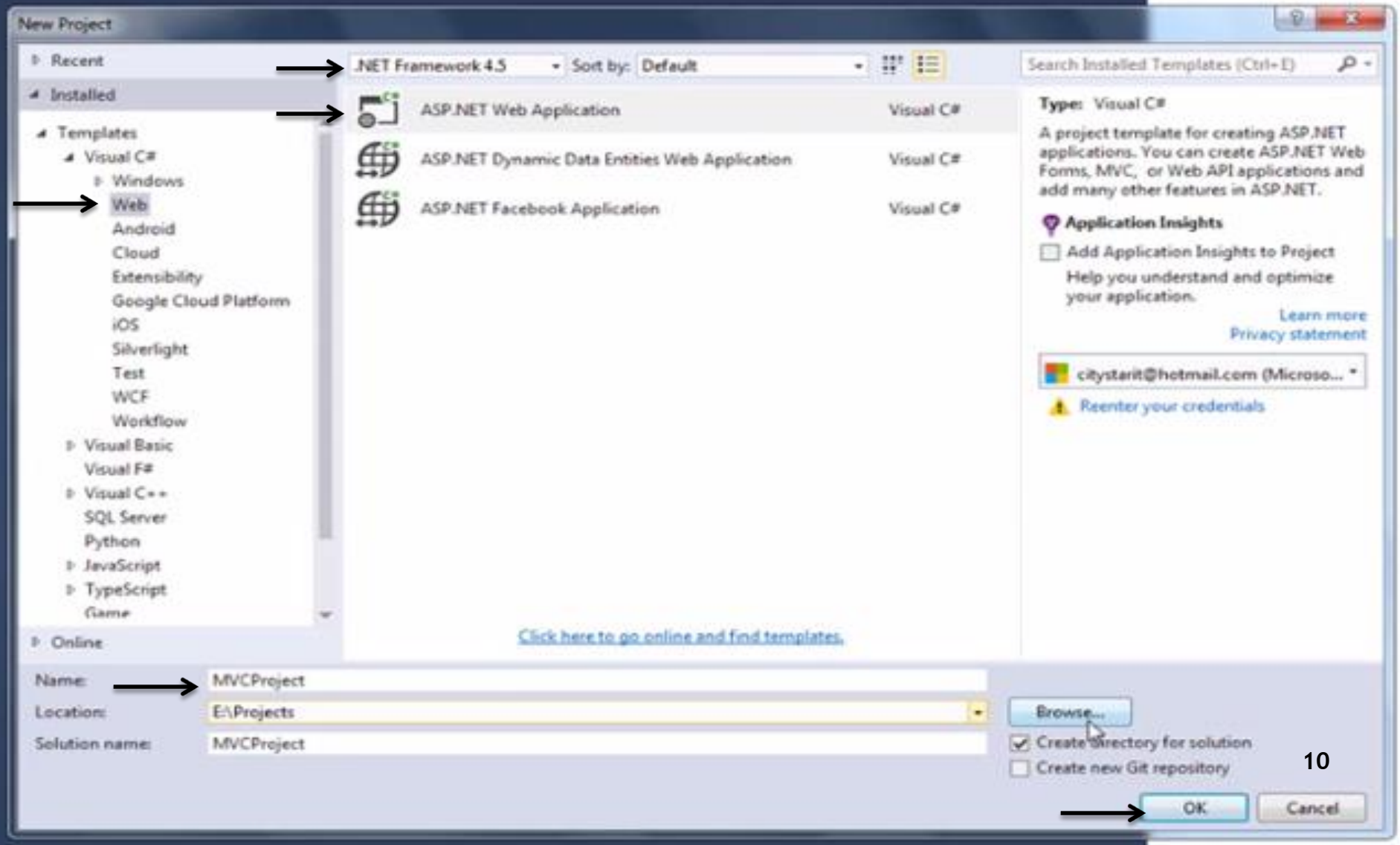
www.google.com/video

اكواد c# واحيانا استعلامات

* ميزاتMVC

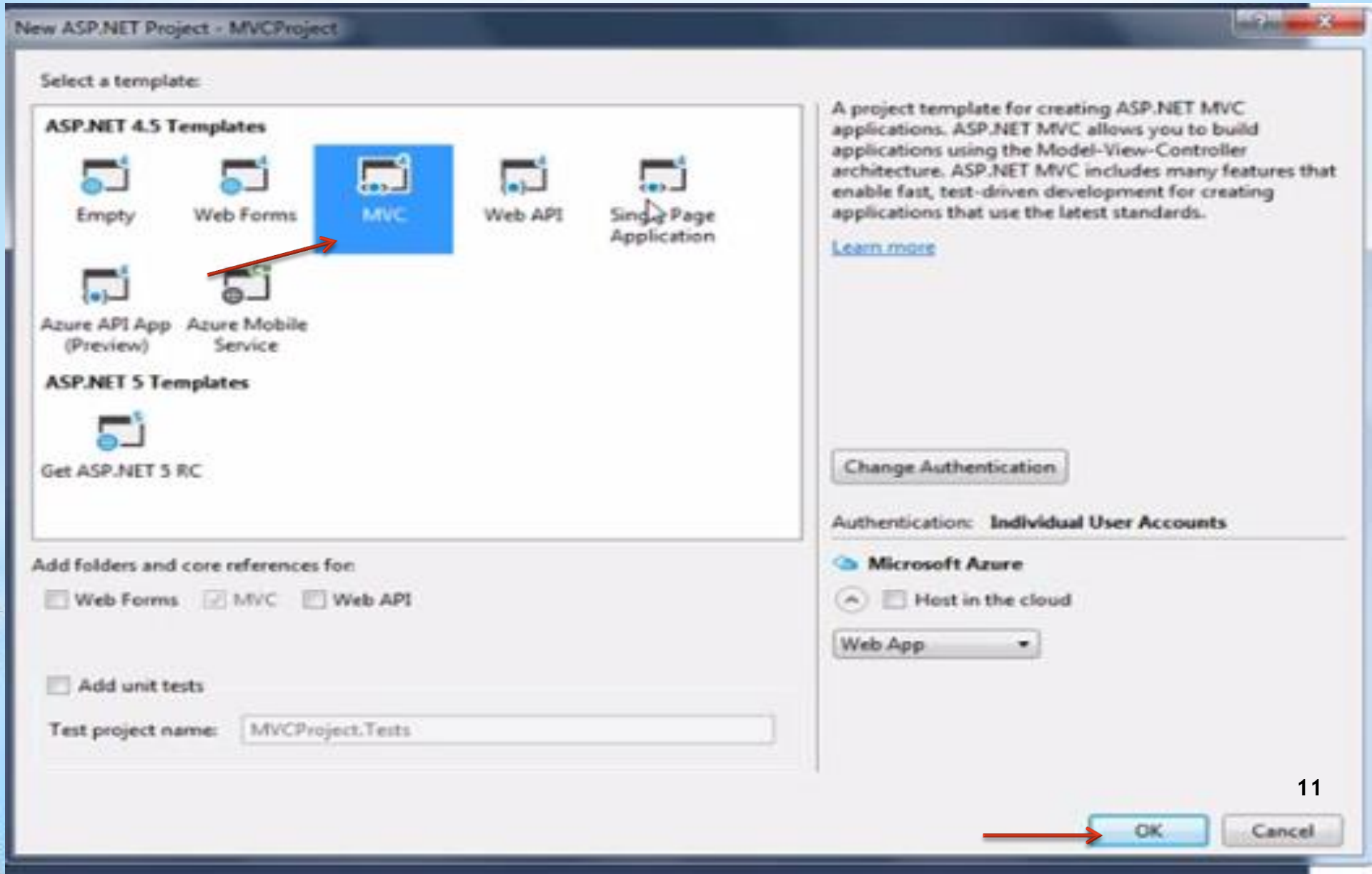
- يسهل MVC إدارة التطبيقات المعقدة عن طريق خاصية فصل الطبقات لتتمكن من التركيز على جانب واحد.
- سهولة اكتشاف الأخطاء.
- سهولة اختبار التطبيق كذلك يبسط MVCعمل فريق التطوير يمكن لأكثر من مطور العمل على الـview، الـ controllerو الـ business logic بشكل متوازٍ.
- يحوي MVC كافة الميزات الموجودة في
- ASP.Net Web Forms مثل الصفحات الرئيسية Master Pages، الأمن والمصادقة Authentication ولكن بنموذج أخف، أكثر مرونة وقابلية للتطوير والاختبار.

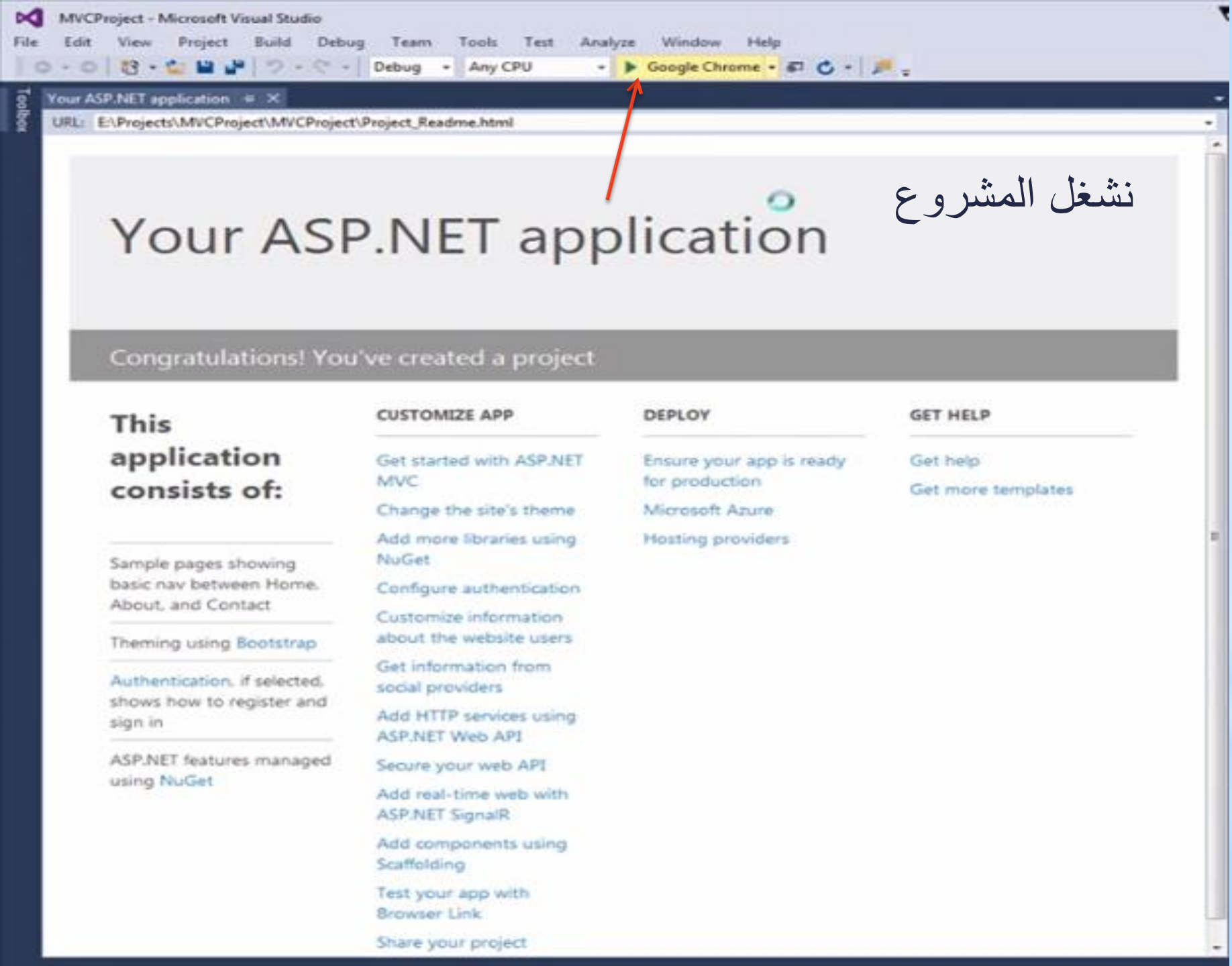
Microsoft Visual Studio 2015>File> New >Project>web > ASP.NET Web Application



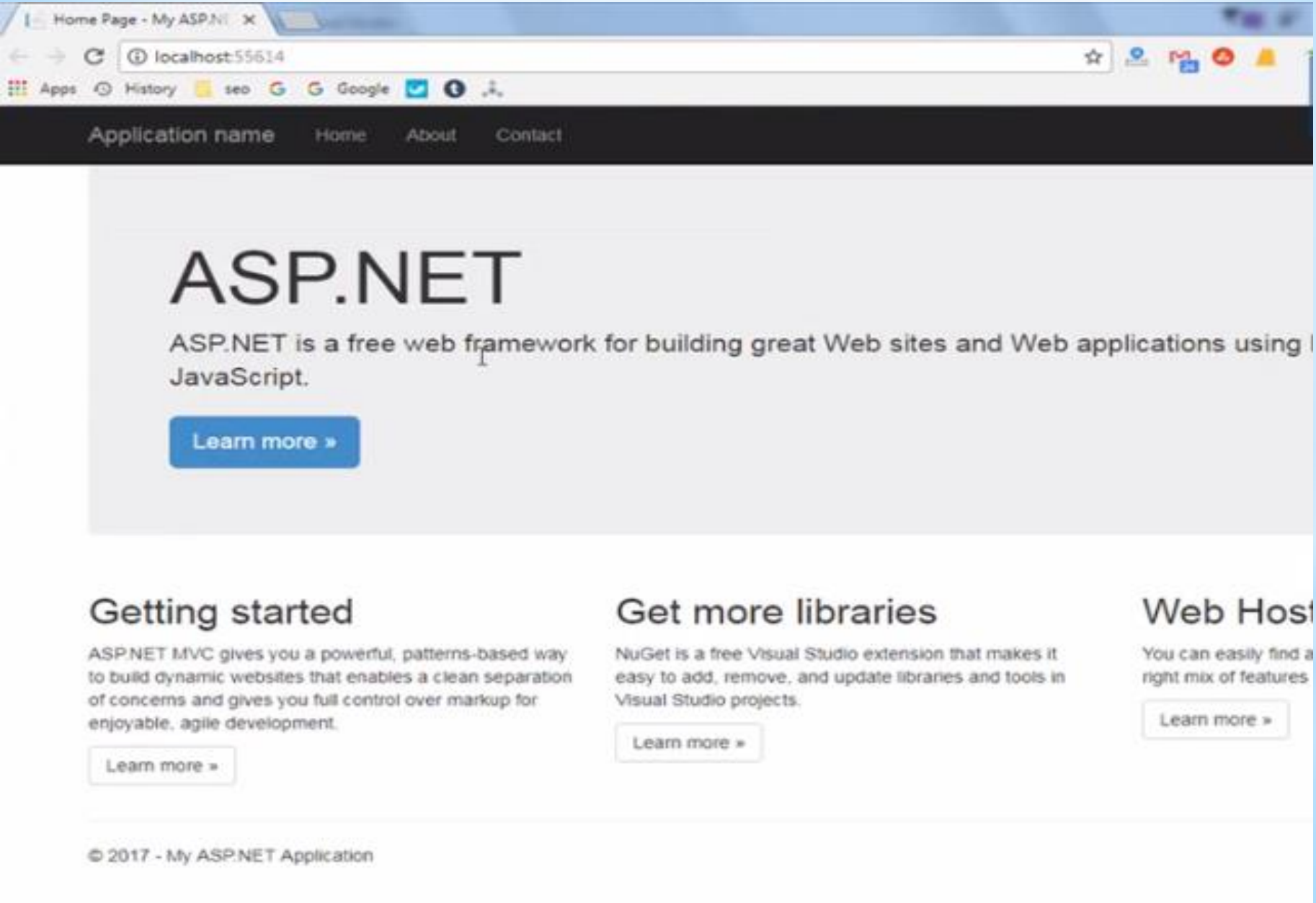
MVC >> ok

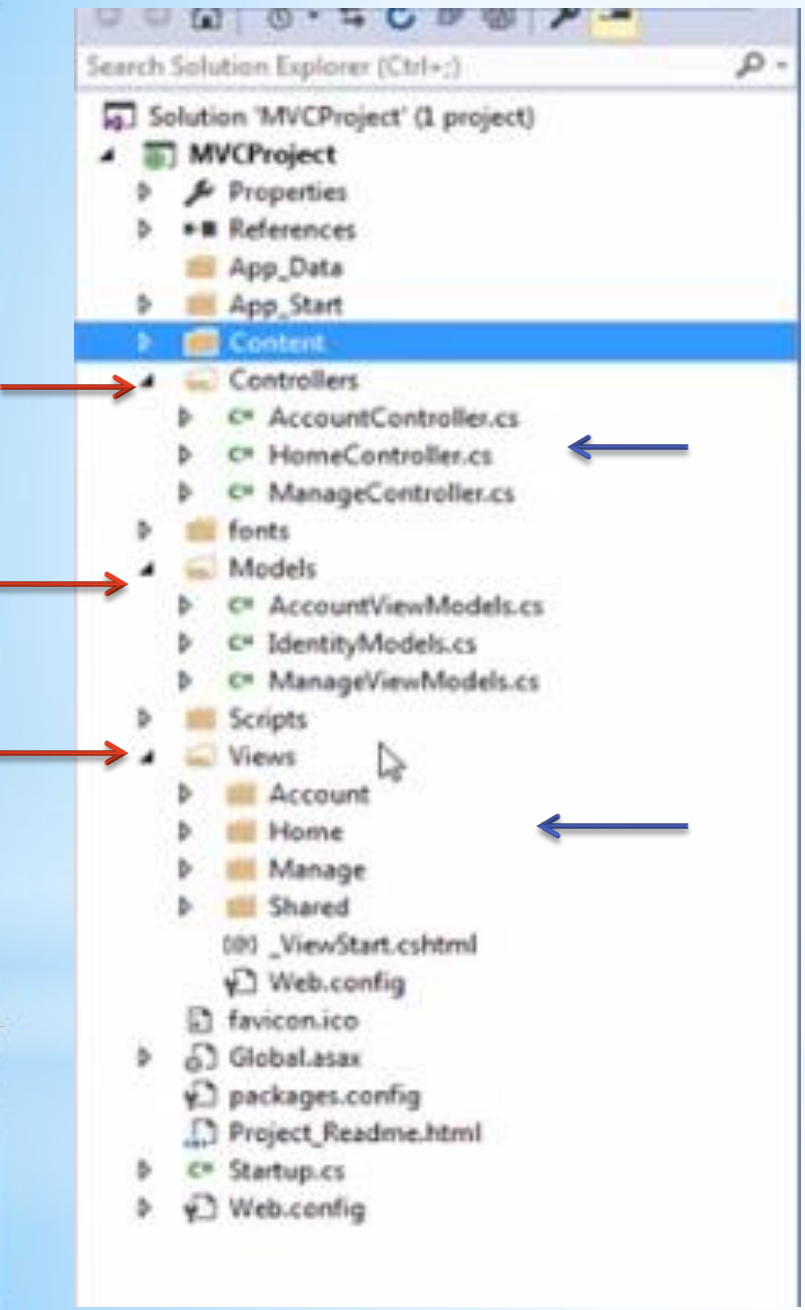
نختار*


New ASP.NET Project - MVCProject
Select a template:
ASP.NET 4.5 Templates
Empty
Web Forms
MVC
Web API
Single Page Application
Azure API App (Preview)
Azure Mobile Service
ASP.NET 5 Templates
Get ASP.NET 5 RC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.
Learn more
Change Authentication
Authentication: Individual User Accounts
Microsoft Azure
Host in the cloud
Web App
Add folders and core references for:
Web Forms
MVC
Web API
Add unit tests
Test project name: MVCProject.Tests
OK
Cancel


MVCProject - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug
Any CPU
Google Chrome
Your ASP.NET application
URL: E:\Projects\MVCProject\MVCProject\Project_Readme.html
Toolbox
نشغل المشروع
Your ASP.NET application
Congratulations! You've created a project
This application consists of:
Sample pages showing basic nav between Home, About, and Contact
Theming using Bootstrap
Authentication, if selected, shows how to register and sign in
ASP.NET features managed using NuGet
CUSTOMIZE APP
Get started with ASP.NET MVC
Change the site's theme
Add more libraries using NuGet
Configure authentication
Customize information about the website users
Get information from social providers
Add HTTP services using ASP.NET Web API
Secure your web API
Add real-time web with ASP.NET SignalR
Add components using Scaffolding
Test your app with Browser Link
Share your project
DEPLOY
Ensure your app is ready for production
Microsoft Azure
Hosting providers
GET HELP
Get help
Get more templates

* تم فتح المشروع كامل



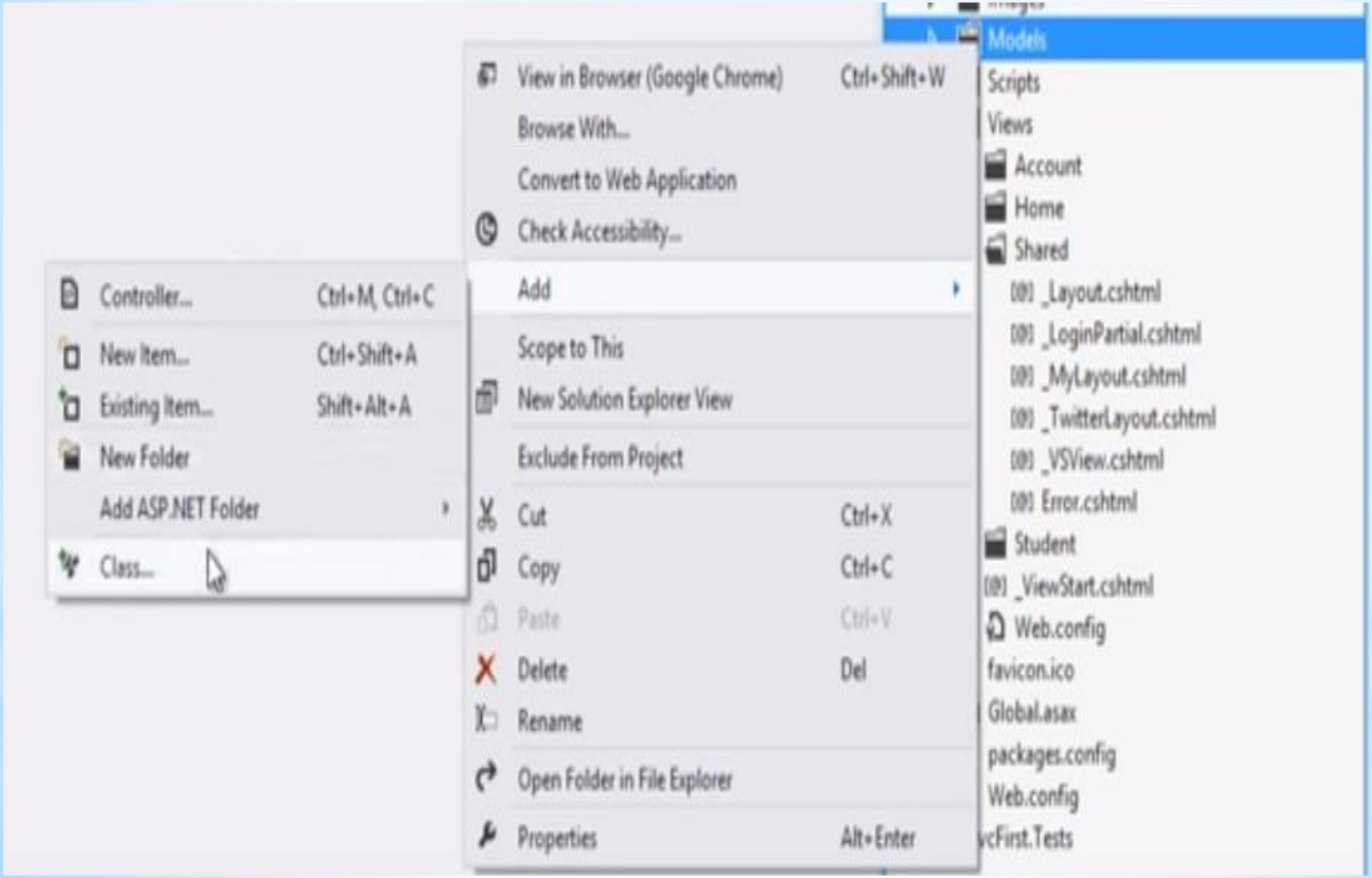


* ثلاث مجلدات تكون MVC مثلاً:

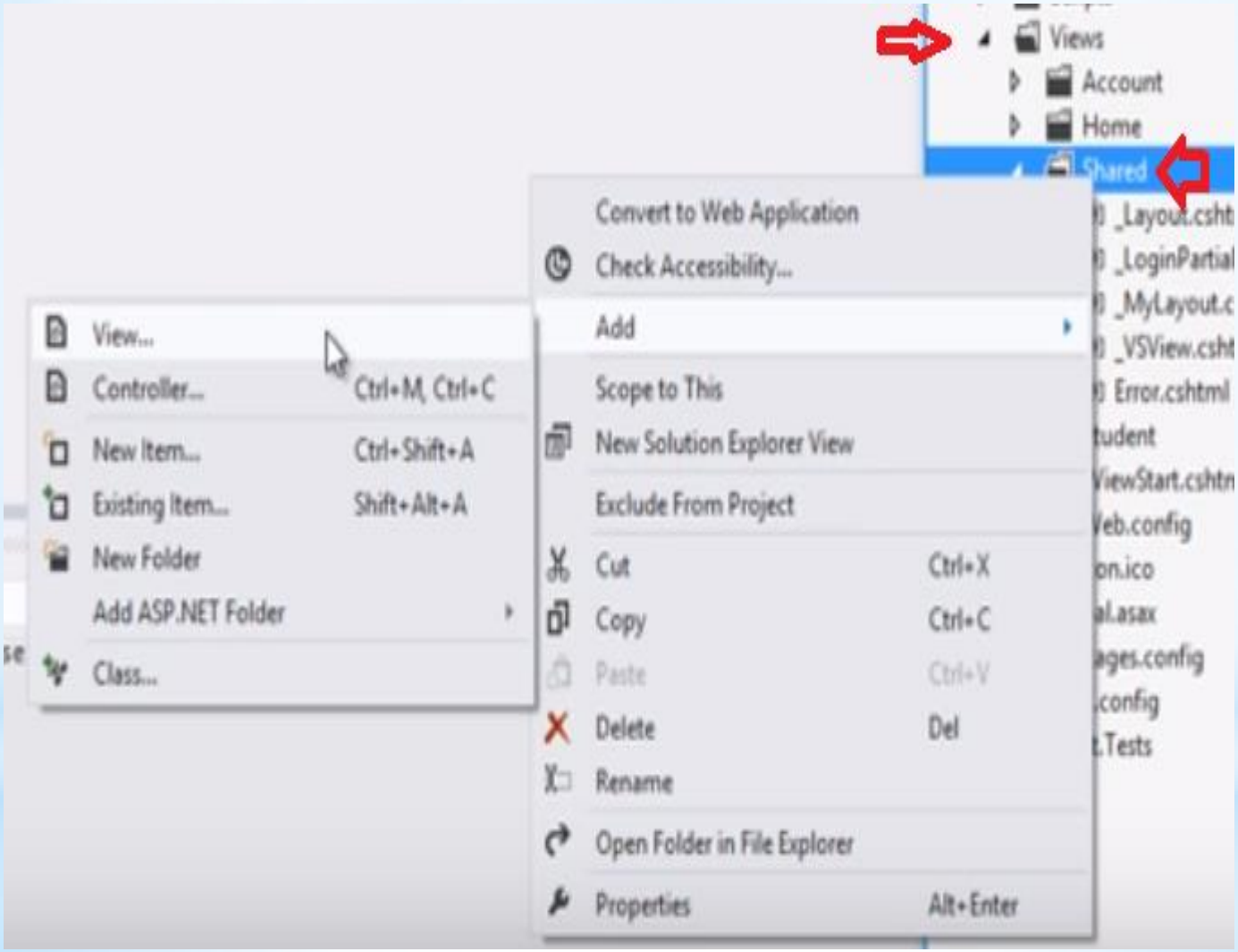
home مرتبط وموجود ب controller

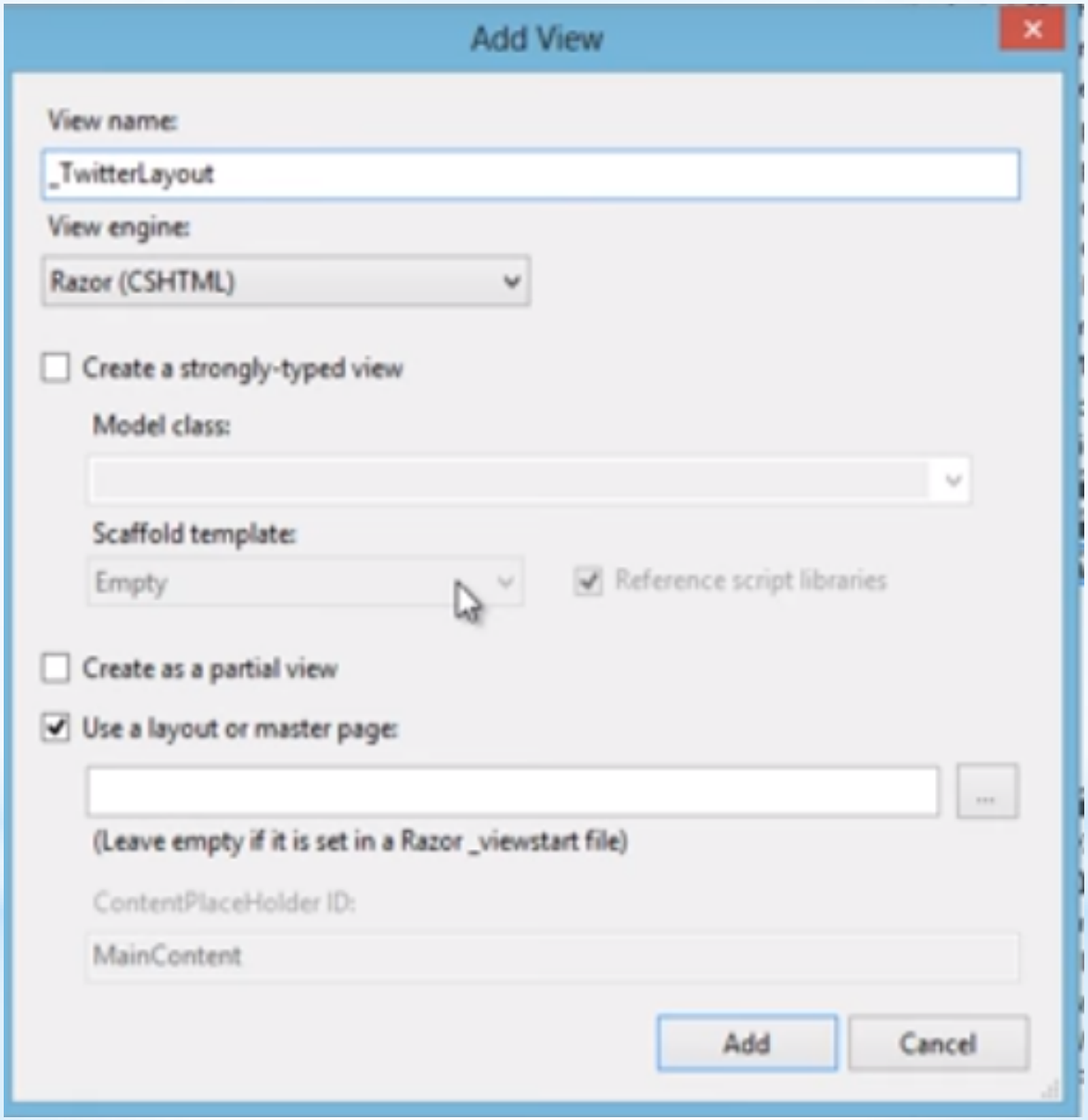
كل view مرتبط ومتصل مع controller

* هنا ننشأ model

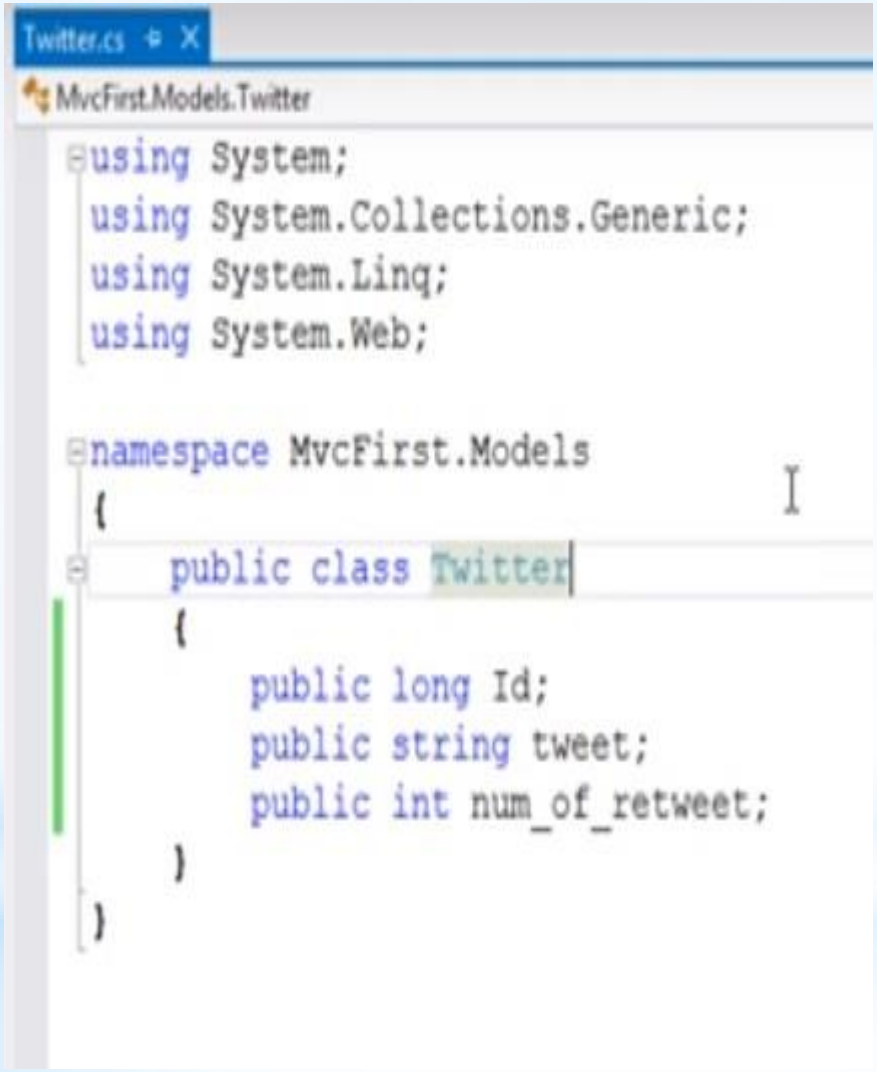


*ننشأ view

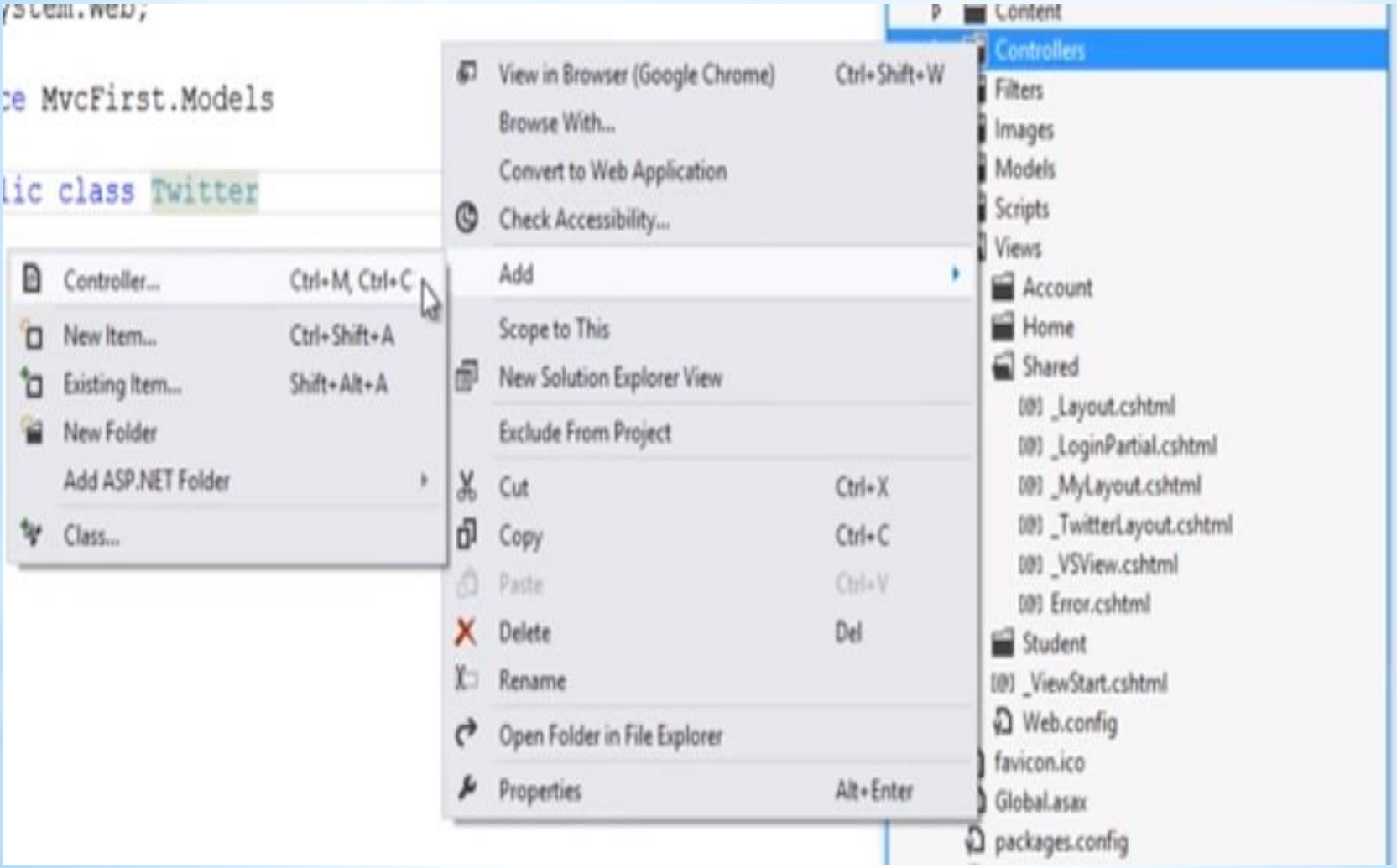

Views
Account
Home
Shared
_Layout.csht
_LoginPartial
_MyLayout.c
_VSView.csht
Error.cshtml
tudent
ViewStart.cshtn
Veb.config
on.ico
al.asax
ages.config
config
t.Tests
Convert to Web Application
Check Accessibility...
Add
Scope to This
New Solution Explorer View
Exclude From Project
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Rename
Open Folder in File Explorer
Properties Alt+Enter
View...
Controller... Ctrl+M, Ctrl+C
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Folder
Add ASP.NET Folder
Class...


Add View
View name:
_TwitterLayout
View engine:
Razor (CSHTML)
Create a strongly-typed view
Model class:
Scaffold template:
Empty
Reference script libraries
Create as a partial view
Use a layout or master page:
...
(Leave empty if it is set in a Razor _viewstart file)
ContentPlaceHolder ID:
MainContent
Add
Cancel

* نضيف له بيانات

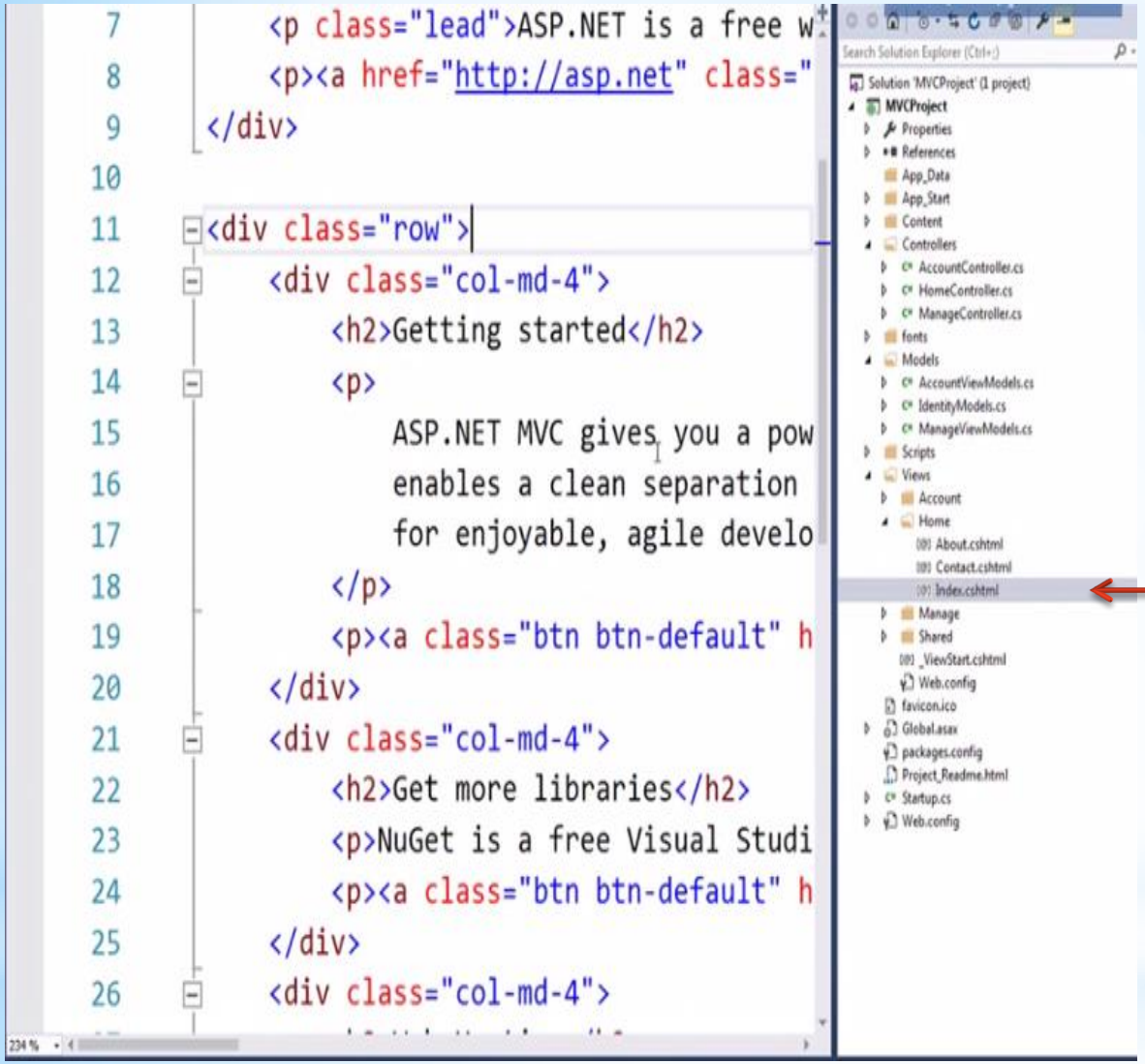


* نقوم بإنشاء controller



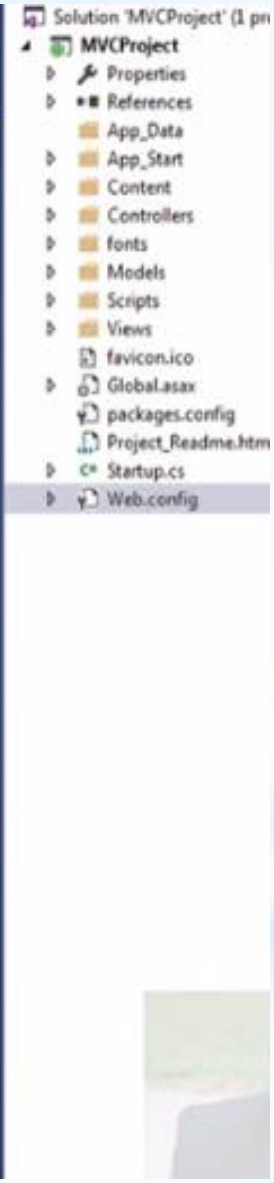
* نضيف أوبجيكت الموديل الى controller
* واضافة قيم ابتدائية
* وبعدها ننشأ view
* من داخل ال controller
* عن طريق الضغط باليمين وبعدها add view

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MvcFirst.Models;

namespace MvcFirst.Controllers
{
    public class TwitterController : Controller
    {

        public ActionResult Index()
        {
            Twitter tobject = new Twitter();
            tobject.Id = 546;
            tobject.tweet = "Welcome to My Twitter App";
            tobject.num_of_retweet = 100;

            return View(tobject);
        }


    }
}
```



* ملفات ال view تكون ملفات html عادية

```
<!--
  For more information on how to configure
  http://go.microsoft.com/fwlink/?LinkId=3
  -->
<configuration>
  <configSections>
    <!-- For more information on Entity Fr
    <section name="entityFramework" type="
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connecti
      providerName="System.Data.SqlClient"
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0
    <add key="webpages:Enabled" value="fal
    <add key="ClientValidationEnabled" val
    <add key="UnobtrusiveJavaScriptEnabled
  </appSettings>
```



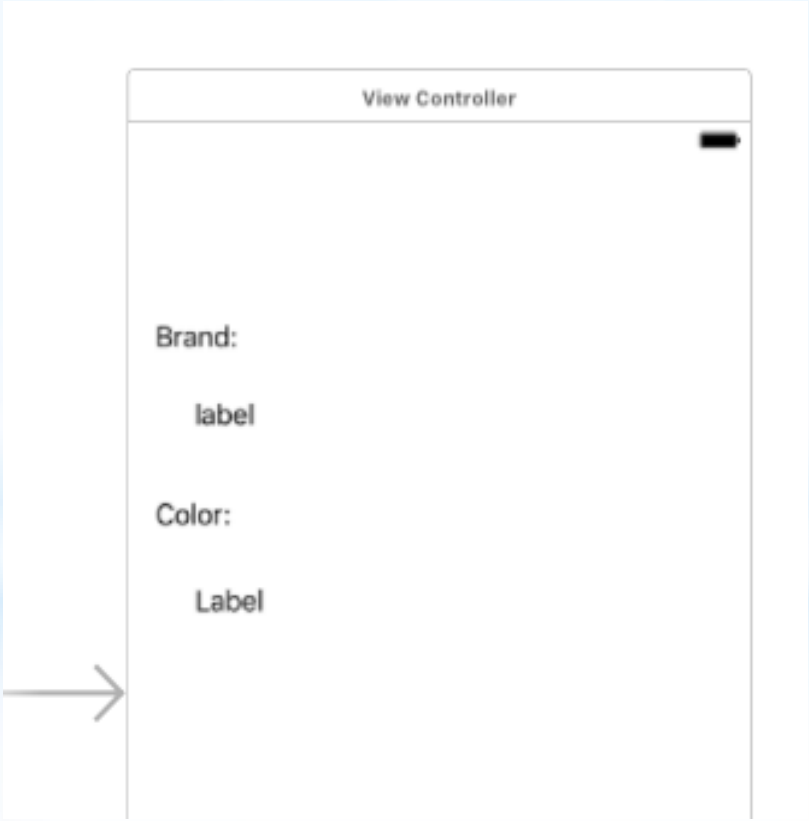
* مثال يستعرض ماركة و لون السيارة

Model: ننشئ class نسميه car ونضع فيه بيانات السيارة (ماركة brand)(لون color)

```
// Model
class Car {
    var Brand: String?
    var Color: String?
}
```

View : هي واجهة المستخدم سننشئها كالتالي :

وننشئ outlet لكل من ال labels كالتالي :




```
// View from the storyboard - both labels
@IBOutlet var Brand: UILabel!
@IBOutlet var Color: UILabel!
```

* **Controller** :وهي الاوامر المستخدمة لربط عناصر الواجهة بالبيانات.
- تم انشاء object اسمه car1 من كلاس car حتى نتمكن من الوصول الى خصائص الكلاس (الماركة و اللون).
- باستخدام ال car1 object
وصلنا الى خاصية brandو وضعنا فيها قيمة lexus
وخاصية color وضعنا فيها قيمة Red
-الان مرحله الربط بين عناصر الواجهة والبيانات
فالأمر Brand.text يشير الى ان نضع في ال labelالموجود في الواجهة النص الموجود في خاصية Car1.Brand
وهو في هذه الحالة lexus نفس الامر لعنصر اللون .

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.

    let Car1 = Car() //object
    Car1.Brand = "Lexus"
    Car1.Color = "Red"

    Brand.text = Car1.Brand
    Color.text = Car1.Color

}
```

ملاحظة : يفضل أن تنشئ ملف منفصل لتضع فيه كلاسات ال model

```swift
import UIKit


// Model
class Car {
    var Brand: String?
    var Color: String?
}


//Controller

class ViewController: UIViewController {


    // View from the storyboard - both labels
    @IBOutlet var Brand: UILabel!
    @IBOutlet var Color: UILabel!


    // Controller continued

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

        let Car1 = Car() //object
        Car1.Brand = "Lexus"
        Car1.Color = "Red"

        Brand.text = Car1.Brand
        Color.text = Car1.Color

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

* نقوم بتشغيل التطبيق لرؤيه النتيجة