ASP.Net Web Forms Course

الدرس الأول

مقدمة إلى ASP.Net

المحتويات :-

- ما هى ASP.Net ؟
- ماذا تحتاج من أداوات لتعمل مع ASP.Net ؟
- ماذا تحتاج أن تعرف من لغات وتقنيات آخرى للعمل مع ASP.Net ؟
- ASP.Net من وجهة نظر فنية.
- كيف تعمل ASP.Net ؟
- كيف تحصل على أدوات العمل ؟
- التعرف على واجهة البرنامج .
- التمرين الآول
- شرح لصفحة ASPX وما يصاحبها من ملفات .
- التمرين الثانى (Entity Framework)

ما هى ASP.Net ؟

مقدمة :

تطوير مواقع الإنترنت أمرٌ هام هذة الأيام , وقد لاقى إهتمام الكثيرين , فأصبح فى تطور مستمر , ولعل من أقوى أوادت تطوير مواقع الإنترنت هى ASP .Net . والتى تقدمها شركة مايكر-وسوفت كبيئة رائعة , توفر العديد من المزايا التى تتيح لك كمبرمج أن تنشأ موقع غنى تستطيع به إرضاء الكثيرين . و ما أوصيك بالتركيز عليه هو "أن تفهم ما وراء المشهد " .

ما هى ASP .Net ؟

ASP. Net ليست لغة برمجة, ولكنها بيئة برمجية لإنشاء وبناء مواقع إنترنت. وهى أداة رائعة لديها ما يجعلك دائماً منبهراً , بل ومغرماً بها .فقد قدمت لك ما يوفر الوقت والجهد فى بناء موقع جذاب, وليس هذا فحسب , بل قدمت لك المزيد من التحكم فى موقعك من حيث البناء والإختبار والنشر للعالم الخارجى وكذلك الإدارة فى أى مرحلة كنت .

حيث تقدم لك مايكروسوفت طريقتين فى بناء مواقع الإنترنت من خلال ASP.Net وهما :-

- **ASP.Net Web Forms**
- **ASP.Net MVC**

سأقدم لك فيما مختصراً لوصف كلاً منهما :-

ASP.Net Web Forms

أحضرت مايكروسوفت من خلالها طريقة العمل مع تطبيقات سطح المكتب (Desk top) إلى تطبيقات الإنترنت, فتجد أن Web Forms تعتمد على العديد من الأدوات المعروفة بإسم Server Controls وكذلك التفاعل من خلال الأحداث والمعروفة بإسم Event Driven . وكثير من الآمور الفنية التى ستتعرف عليها بعد قليل إن شاء الله .

ASP.Net MVC

فى تفاديها لمشاكل Web Forms أحضرت مايكروسوفت من خلال MVC , إمكانية العمل مع مبادئ تصميم البرمجيات , ولعل أهم مبدأ ركزت عليه هو SOC , وهو فصل الإهتمامات , إختصاراً لـ Separation of Concern , حيث تمثل ذلك فى الفصل بين Model-View-Controller, وكذلك وفرت سهولة تطبيق أنماط التصميم والمعروفة بإسم Design-Patterns , أيضاً وفرت لك المزيد من التحكم فى أكواد HTML ومرونة فى إجراء الإختبارات والمعروفة بإسم Unit Testing .

ملحوظة :- لمزيد من المعلومات حول MVC توجد دورة آخرى مكونة من 15 درساً تتحدث عن MVC .

قد تكون قد رأيت أنه هناك ما يعرف بـ ASP.Net Web Page , فماذا عنه ؟ . ASP.Net Web Pages مخصص لمبتديئ البرمجة, أو كما وصفتهم مايكروسوفت , الطلاب والهواة , وذلك لإنها بيئة لبناء مواقع متناهية الصغر , كصفحة أو صفحتين , حيث تريد مايكروسوفت من خلاله توفير بيئة مرنة للذين ليس لديهم خبرة فى البرمجيات وكذلك فى ASP.Net .

ماذا تحتاج من أداوات لتعمل مع ASP.Net ؟

يمكنك التعامل معها من خلال Visual Studio أو Visual Web Developer Express وكلها أدوات تقدم مايكروسوفت , وفى النهاية تحصل على تطبيق مكون من صفحات ذات إمتداد aspx ,تعمل عبر الشبكة سواءً كانت شبكة داخلية أو شبكة خارجية (شبكة الإنترنت).

فنجد أن مايكروسوفت أنشأت بيئة التطوير هذة بغرض توفير جميع أداوات تطوير مواقع الإنترنت داخل آداة واحدة .فنجد أن Vs أو VWD به دعم للعمل مع SQLو CSS و JS وغيرهم.

ولإن تطبيقات ASP.Net تعمل عبر الشبكة , فإنه لابد من خادم يتولى إدارة الطلبات القادمة إلى التطبيق من كافة الأجهزة المتصلة بهذة الشبكة ,بحيث يوجه الطلب القادم إلى التطبيق ليتم تنفيذ طلبه ومن ثم يقدم له الناتج على هيئة أكود Html يتم عرضها فى المتصفح .

IIS: Internet Information services

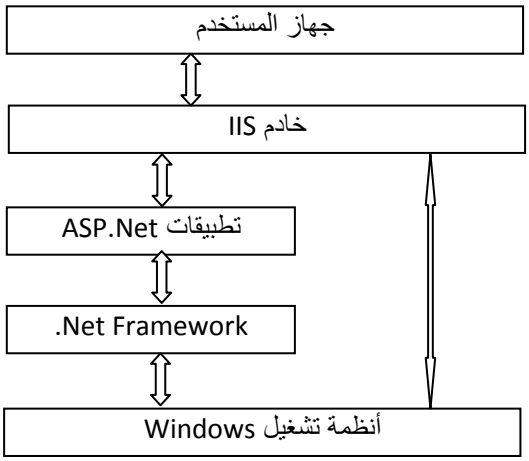
كان الآمر فى بدايته يتطلب وجود IIS والذى يعمل كتطبيق يمثل الخادم الذى سيتولى عملية الإدارة التى تحدثنا عنها , ولكن الآن لا يشترط وجود خادم IIS عند التطوير وذلك لإن VWD أو VS مزودوان بخادمان مدمجان معهما وهما : IIS Express و Cassini ويسمى أيضاً Local-host . ولكن يشترط وجود خادم IIS عند نشر الموقع سواءً كان على جهازك أو على جهاز آخر .

نذكر أنه يمكنك تشغيل تطبيقات ASP.Net على خادم Apache الخاص بـ PHP أيضاً , ولكن بإستخدام ما يعرف بـ Mono Project . كذلك إفتراضياً لابد من وجود نظام تشغيل Windows للعمل مع ASP.Net . ولكن يمكن تشغيل ASP.Net على نظام تشغيل Linux بإستخدام Mono Project .

وسنرى إن شاء الله فى الدرس القادم كيفية تثبيت IIS على جهازك الشخصى , والتفاعل معه .

هندسة صفحة ASP. Net

تامل هذة الصورة :



بدايةً قبل التبحر فى تفاصيل أكثر تعقيداً , الرسم الذى أمامك يمثل الشكل البسيط لبيئة العمل , ولكن يمكن أن يدور الحديث عنه من جهتين ,الجهة الأولى , انه رسم توضيحى لمتطلبات العمل لإنشاء وتطوير تطبيقات ASP. Net . و الجهة الأخرى, أن هذا الشكل يمثل موقعك بعد النشر .

الجهة الأولى :

,ومنه فإنك ترى أن نظام التشغيل الذى يلزمك لتستطيع العمل على ASP. Net هو النظام الخاص بـ Microsoft المعروف بإسم Windows , ويتم تثبيت .Net framework وكذلك الأداة التى تمكنك من إنشاء تطبيقات ASP.Net وهى Visual Studio , أو طبعة أخرى تسمح بإنشاء مثل هذة التطبيقات , نجد أيضاً أنه يجب تثبيت برنامج الخادم IIS . على هذا النظام .

الجهة الأخرى:

نجد من الرسم أن المستخدم يقوم بطلب صفحة من تطبيق ASP. Net وحيث ان هذا التطبيق مثبت على خادم IIS فإن هذا الخادم هو من يقوم بفحص هذا الطلب ليقوم بعد ذلك لتسليمه إلى .Net framework المثبت على نظام التشغيل windows .ليصلك فى النهاية مخرجات الصفحة على هيئة HTML .

ماذا تحتاج أن تعرف من لغات وتقنيات آخرى للعمل مع ASP.Net ؟

لبناء موقع كامل بالعديد من الوظائف , لابد من توافر المزيد من اللغات والتقنيات التى تشترك وتتفاعل مع بعضها البعض لأداء مهمة , فقد تكتب كود بلغة C# ليتفاعل مع قاعدة بيانات من نوع SQL , فتضع فى الصفحة آداوات مكتوبة بلغة HTML وعند التفاعل مع هذة الأدوات يتم إستخدام مكتبة أكواد JQuery وهى المكتبة التى تم إنشائها بواسطة لغةJavaScript لإرسال طلب للخادم لتنفيذ الأكواد التى تمت كتابتها بلغة C# لتعود بعدة صفوف من البيانات من جدول فى قاعدة البيانات , لتقوم Jquery بعرضها داخل الصفحة ليظهر لك على هيئة جدول منسق بشكل جذاب تم ضبط آلوانه بإستخدام CSS ___ أرأيت كيف تكاتف الجميع لإداء مهمة واحدة؟

وفيما يلى سنقوم بعرض نبذة مبسطة عن اللغات والتقنيات التى يمكن إستخدامها داخل موقع تم تطويره بإستخدام ASP.Net , ولاحظ أنها نبذة مختصرة لإن التفصيل سيأتى فى وقته إن شاء الله :-

Html

هى لغة عبارة عن مجموعة من الرموز نستخدمها لتصميم صفحات إنترنت .

مثال :- إستخدام <img> لوضع صورة داخل الصفحة وبعضاً من الروابط وغيرها من الأدوات ,مثال Table, Div, button , Textbox.

CSS

لغة تستخدم لتنسيق صفحات الإنترنت . ويتم تنفيذها بواسطة المتصفح .
مثال :- ضبط الألوان وأحجام الخطوط والصور وغير ذلك من التنسيقات .

C#\vb.net

هى لغة برمجة يتم تنفيذها على الخادم (Server) ونستخدمها لآداء المهام المعقدة , ويتم تنفيذها بواسطة Net framework .
مثال:- الإتصال بقاعدة البيانات , أو ربما إرسال بريد إلكترونى وما شابه ذلك .

JavaScript

هى لغة يتم تنفيذها على جهاز المستخدم (Client). ويتم تنفيذها بواسطة المتصفح .
مثال :- إظهار حقل وإخفائه , أو ربما تكبير صورة وتصغيرها , أو إيقاف متحرك وتحريكه مره آخرى , أو ربما الإتصال بالخادم بدون تحديث كامل للصفحة .

JQuery

هى مكتبة تم إنشائها بواسطة JavaScript والهدف منها تبسيط العمل مع JavaScript حيث قدمت لك الكثير ولعل من أبرز ما قدمته سهولة الإتصال بالخادم من خلال الـ client والتحديث الجزئى للصفحة .
مثال :- الإتصال بـ Web Service , أو ربما الإتصال بقاعدة بيانات وإحضار نتائج وعرضها داخل Grid .

Web Service

هى خدمة يقدمها موقع ما ,موقعك أو موقع آخر ,وهى عبارة عن مهمة يتم تنفيذها .
حيث يتم الإتصال بهذة الخدمة من أى مكان ,لتقدم لمن يطلب نتائج يمكن أن يستخدمها فى موقعه .ومنها ما هو مدفوع ومنها ما هو بدون مقابل .
مثال:- خدمات الطقس , أو خدمات إرسال SMS ومعرفة أماكن المتصلين بالتفصيل وما شابه ذلك.

SQL Databases

قواعد بيانات من نوع SQL .

SQL Server

إدارة الطلبات القادمة إلى قاعدة البيانات من تخزين أو إستعلام .

SQL Management System

نظام لإدارة قواعد البيانات من إنشاء وتعديل وحماية والمزيد من أمور التحكم فى قواعد البيانات .

ADO.Net

مجموعة من الـ classes الهدف منها الإتصال بقواعد البيانات والتى تحتوى بداخلها على ما يعرف بـ LINQ و Entity Framework .
مثال :- الإتصال بالقاعدة وإحضار بيانات ثم تخزينها فى الذاكرة ومن ثم فصل الإتصال

LINQ

لغة إستعلامات متكاملة, الهدف منها تبسيط الإستعلام . تشبه فى شكلهاTransact-SQL ولكن يتم تنفيذها ضمن اكواد C#/vb.net . وما وراء المشهد هنا , أن الإستعلامات المكتوبة بـLINQ يتم تحويلها إلى جمل Transact-SQL وذلك فى حالة إجراء الإستعلام مع قاعدة بيانات , ولهذاتم إنشاء LINQ to SQL و LINQ to Entity

مثال :- إجراء إستعلام داخل ملف xml أو داخل قاعدة بيانات , أو ربما Collections .

Entity Framework

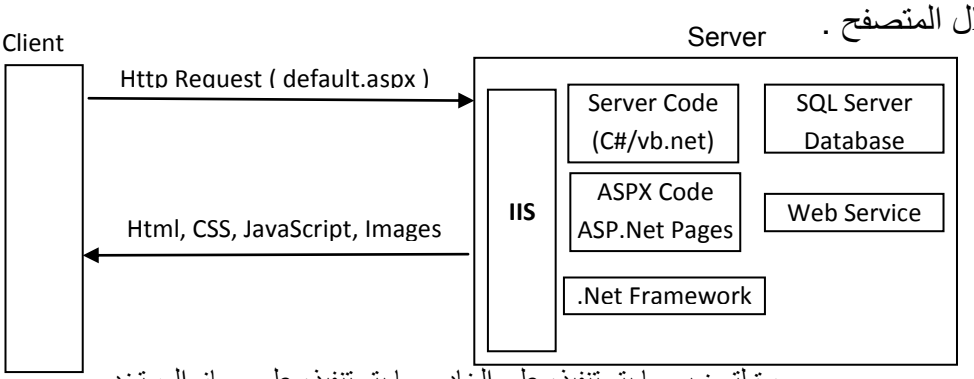
كلمة "Entity" تترجم فى العربية إلى " كيان " , وكلمة Entity هنا تطلق على Class والتى فى حد ذاتها تمثل جدول فى قاعدة البيانات . وتتلخص Ef فى أنها تمثيل لقاعدة البيانات وجداولها فى مجموعة من الـ Classes , حيث توفر لك إمكانية العمل مع قاعدة البيانات وجداولها من خلال مجموعة من الـ Classes الموجودة فى الذاكرة , وفى باطن الأمر يتم إنشاء مجموعة من الآوامر التى يتم إرسالها إلى قاعدة البيانات ,هذة الأوامر تتمثل فى جمل Transact-SQL , أى الجمل العادية التى نعرفها مثال Update, Insert , Select وغيرهم .

وتقع EF ضمن ما يعرف بأنظمة ORM , حيث تقوم أنماط (Patterns) تعرف بـMapping, والمقصود بـ Mapping هو تمثيل جدول موجود فى قاعدة بيانات بـ Class موجودة فى الذاكرة .

Ajax

هى تقنية تجمع فى عملها بين JavaScript وبين XML ,حيث تأتى كلمة Ajax إختصاراً لـ Asynchronous JavaScript and XML, والغرض منها يتلخص فى ثلاث كلمات "تحديث جزئى للصفحة ".وسيأتى تفصيلاً بعد قليل ,والتطبيق سيكون فى درس مستقلاً عنها إن شاء الله.

تأمل الشكل التالى والهدف منه توضيح ما يوجد ويتم تنفيذه على الخادم وما يتم إرساله إلى جهاز المستخدم الذى بدوره يتصفح التطبيق عبر الشبكة .
فى البداية , يجب أن نعرف أن كل الملفات تتواجد بالفعل ضمن ملفات التطبيق على الخادم , ولكن ما نقصده هنا , أنه عند طلب المستخدم لصفحة ما , يتم الإستجابة لطلبه بإرسال كلاً من HTML , CSS, JavaScript, Images وكلها معده ليتم تنفيذها على جهاز المستخدم , أى من خلال المتصفح .



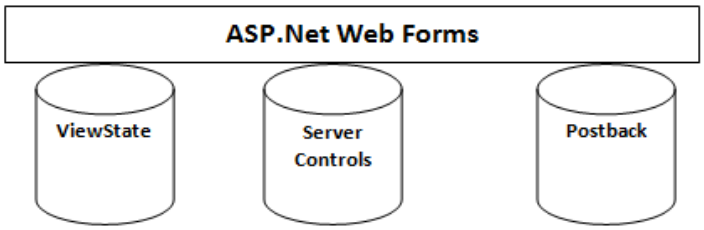
صورة لتوضيح ما يتم تنفيذه على الخادم وما يتم تنفيذه على جهاز المستخدم

ونذكر هنا أن كل ما تم ذكره بالأعلى مختصراً , سيأتى إن شاء الله تفصيلاً له فى مكانه فى الشرح .

التعريف بـ ASP.Net بشكل فنى

تقف ASP. Net على ثلاثة أعمدة هما:-

- Postbacks
- ViewState
- Server Controls



سنعطى تعريفات شبه مختصرة لإن التفصيل سيأتى في النقاط المتتالية إن شاء الله :

- **PostBacks**
  - رحلة طلبك لتحديث صفحة ما, بداية من الضغط على Button (مجرد مثال) لتذهب إلى الخادم,ومن ثم العودة بالتحديثات,ليتم عرض الصفحة المحدثة فى متصفح المستخدم . والتعريف المختصر : "رحلة الطلب من المتصفح إلى الخادم ثم العكس ", ستجدها فى الإنجليزية بإسم "Round Trip" والترجمة "ذهابا وإياباً".
- **ViewState**
  - حالة التطبيق , كما ذكرنا فى تعريف PostBack , فإن هناك ذهاباً وإياباً, وإليك هذه الحقائق :الصفحة بمثابة عمل Instance من Class, يتم إنشاء و تدمير هذا الـ Instance فى كل مرة تبدأ هذة الرحلة (Postback) وبالتالى كل الخصائص التى تم ضبطها فى هذا الـ Instance قد ذهبت مع الريح, ويتم إعادة إنشائه مرة أخرى ليتم إستخدامه فى العودة للصفحة , وبالطبع ستكون قيم خصائصه هى القيم الإبتدائية ,وليست القيم الجديدة التى قام المستخدم بإختيارها , لذا وجب علينا أن نحتفظ بالقيم الجديدة فى مكان ما, لنعيد إنشاء الصفحة بهذة القيم لتظهر وكأن عملية التدمير وإعادة الإنشاء لم تتم .ومن أجل هذا الغرض جاءت ViewState .
- **Server Controls**
  - هى أداوت صممت خصيصاً لتتفاعل مع الخادم .والأمر الذى يميزها أنها Server Controls هى الخاصية Runat , ولاشك أنها تتمتع بالعمل تلقائياً مع ماتم تعريفه مسبقاً بـ ViewState, أى قابلة لحفظ حالتها بإستخدام ViewState. ولا نغفل ذكر أمراً جيداً , وهو ,إن من هذة الأداوت من يوفر لك الوقت والجهد وكذلك الكفاءة , كأدوات للتحقق من صحة البيانات المدخلة. تلك الأدوات التى تقوم بتوليد أكواد مساعدة لها صممت بواسطة JavaScript , لتقوم بعمل كان يحتاج منك الكثير والكثير .وأيضاً لا ننسى أداوت ما يعرف بـ Membership . فنجد أن ASP.Net توفر لك أدوات تستطيع من خلالها ان تزود موقعك بمزيد من الخصوصية كتسجيل المستخدم دخوله للموقع ,وكذلك تسجيل عضو جديد وأشياء من هذا القبيل. وسيأتى تفصيلاً لكيفية عمل هذة الأدوات فيما بعد .

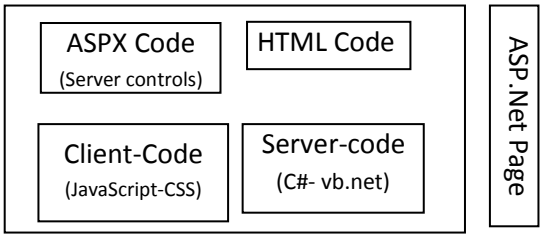
صفحة ASP.NET وكيف تعمل ؟

فعلى هذة الثلاثة تم بناء ASP.Net , وكأنها مجموعة من الأدوات تدعم كلاً من PostBack و ViewState . هذا هو كل شئ , فعند تصميم صفحة ما فى موقع . نجد الأمر قد بات واقعا محسوساً , فهيا لنرى مما تتكون الصفحة أو بالأحرى الموقع , لإن الموقع الواحد يتكون من صفحات, وهذة الصفحات تم إنشاؤها بإستخدام هذة البيئة (ASP.Net) , فنجد أن الصفحة الواحدة ماهى إلا ملف بإمتداد .aspx , له تركيبته الخاصة والتى تميزه عن ملفات أخرى مشابه له .وسنتعرف على هذة التركيبة فيما بعد إن شاء الله .

ولكن لعل من المهم الآن أن تعرف أن محتويات هذة الصفحة تتكون من كود يسمى aspx Code. وهو عبارة عن أداوت تم إنشاؤها لتشكل فى النهاية كامل الصفحة, وهذة الأداوت يطلق عليها إسم Server- Controls والترجمة أداوت الخادم وقد تقدم ذكرها . وهذا لا ينفى إحتواء الصفحة على أدوات أخرى والتى يطلق عليها HTML Controls. ومن الجدير بالذكر الآن أن تضع فى ذهنك أن المتصفح (Web Browser) الذى هو المسئول الأول والأخير عن عرض صفحات الإنترنت , لايفهم إلا لغة واحدة وهى HTML . ومثال على هذة المتصفحات مثل (Internet Explorer Firefox , chrome , Opera, Safari, وغيرهم).

ألا يوجد هناك تضارب!! , كيف نستخدم أداوت Server Control وهى aspx Code والتى لا يفهما المتصفح ! . فكيف سيتم عرضها للمستخدم ؟!. فى الحقيقة هذة الأدوات يتم تحويلها إلى HTML فى وقت التشغيل(Run-time). وهذة العملية يطلق عليها إسم Rendering .

ولاحظ أيضاً أنه قد يوجد فى الصفحة أيضاً أكواد #C أو Vb.net للتفاعل مع الخادم , أو ربما كود JavaScript لإعطاء ردة فعل سريعة إستجابة ً لإختيارات المستخدم. أو قد يوجد هناك أكواد CSS لتنسيق عناصر الصفحة ونقصد بالعناصر هنا المكونات. ولكن ليس بالضرورة أن يتواجد الكل فى صفحة واحدة, فقد يتواجد كلاً منهم فى صفحة مستقلة وهذا الأفضل .



وإليك التفصيل من وجهة نظر فنية :-

ما هى HTML ؟

هى إختصار (Hypertext Markup Language). وهى لغة مكونة من رموز تتشكل بها صفحات الإنترنت , فالخانة التى تراها وتستطيع أن تكتب فيها إسمك او عنوانك أو حتى بريدك الإلكترونى تسمىTextbox , والرمز أو الأداة التى تمثل الرابط الذى تراه وتسطيع الضغط عليه يذهب بك إلى مكان آخر سواء صفحة أخرى او موقع آخر أو تنتقل إلى مكان آخر فى نفس الصفحة يسمى anchor .فعند الحاجة لإظهار رابط للمستخدم يذهب به إلى محرك البحث Google فإننا نستخدم التالى :

```
<a  href="http://www.google.com" title= "go to google">Google</a>
```

فهنا نجد ان كلمة Google ستظهر للمستخدم وكلمة "go to google" ستظهر عندما يقف بالمؤشر على الرابط. وبالفعل سيذهب بالمستخدم إلى الرابط المرفق بالأعلى وهو محرك البحث جوجل وذلك عند الضغط على الرابط . وهذا بعض الشئ من قدرات HTML.

فنأتى هنا لتعريف لهذة اللغة (HTML) فى جملة واحدة : "هى لغة تصميم صفحات إنترنت". ولكن حين نتكلم عن صفحة إنترنت مصممة فقط بـHTML , فإننا نتحدث عن صفحة ثابتة المحتوى ولا تتغير, يطلق على هذا النوع من الصفحات Static Pages .ونقصد هنا بإنها لا تتغير أى أن محتواها لا يتغير ,فسيكون المحتوى هو نفس المحتوى الذى سيراه المستخدم عند كل زيارة إلا إذا قام المصمم بتغير محتوى الصفحة من جديد.

أما النوع الآخر من الصفحات هو Dynamic Pages وهو الذى جاءت من أجله ASP. Net , حيث يستطيع فيها المستخدم التحكم فى شكل الصفحة , ويمكنه أيضاً التعامل مع قواعد البيانات لإرسال وإستقبال بيانات من وإلى الخادم .وتوفير المزيد من الحماية لخصوصية الموقع , والعمل على توفير قدر كبير من العمليات الإدارية لصفحات الموقع من فحص وإختبار وتتبع الأخطاء وإصلاحها والتعامل مع الذاكرة و غيرها المزايا الكثير الكثير التى تقدمها لك ASP.Net . ويتم هذا التفاعل من خلال إرسال وإستقبال طلب للخادم يعرف فى الإنجليزية بإسم HTTP Request.

HTTP Request

دعنا نلقى الضوء على HTTP قبل كل شئ ,بدايةً هو إختصار " Hypertext Transfer Protocol " وهو إتفاقية لنقل النصوص المتشعبه , وأتى مع ظهور WWW وهى World Wide Web . ويأتى له تعريف بإنه" المستندات المرتبطة ببعضها على الإنترنت" .فيمكن ان نعطيه تعريف بإنه " وسيلة الإتصال بن جهاز المستخدم و الخادم " ويستخدم بعضاً من الأوامر التى يستطيع ان يفرق بها بين نوعية الطلب القادم إليه ليتفاعل مع كل طلب بما يتوافق معه , ومثال هذة الأوامر مثل GET , POST وغيرهما, ولكن هما ما يتم إستخدامها فى ASP.Net .

GET and POST

فالأول GET , و يستخدم عند طلب الصفحة لأول مرة وهو لعرض القيم الإبتدائية للصفحة , ليمر هذا الطلب بعدد من المراحل , يزداد هذا العدد فى الطلبات المتتالية لنفس لصفحة, ولكن عندها سيكون طلب الصفحة بإستخدام الأمر POST , ويأخذ هذا الأمر طريقه فى الذهاب إلى الخادم عند إعطاء المستخدم الأمر بذلك, ليقوم بتنفيذ حدث ما تم إعداده خصيصاً ليكون بمثابة ردة فعل لما قام المستخدم بفعله , بالطبع لا يعرف المستخدم انه قام بعمل POST, ولن يجد ذلك مكتوباً أمامه. لكنه قام بالضغط على Button , او قام بإختيار شئ ما, يتطلب الذهاب إلى الخادم لتحديث حالة التطبيق .

ملحوظة :-

حالة التطبيق نقصد بها هنا المحتويات وما تم التعديل فيه وأيضاً ما يترتب على هذا التعديل . على سبيل المثال, إختيار المستخدم عرض الهواتف المحمولة من قائمة لديه فى الصفحة, ليظهر له

جدول به كل الهواتف المتاحة. فهل من المنطق ان يختار عرض السيارات ونظل نحن نعرض له قائمة بالهواتف , فكان لابد من تحديث حالة التطبيق ليعرض له التحديثات ,فوجب علينا ان نعود إلى الخادم ليمدنا بالتحديثات ومن ثم عرضها للمستخدم , وتسمى رحلة الذهاب والإياب بـإسم Postback . وبعد هذة الرحلة يتم تحديث حالة التطبيق.

ملخص هذا الكلام: قد عرفت ان الطلب الأول للصفحة الواحدة يكون بإستخدام الأمر GET والأوامر المتتالية لنفس الصفحة من نفس المستخدم تكون بإستخدام الأمر POST .

المستخدم والخادم

لابد أن تفهم جيداً آلية عمل كلاً منها قبل ان تبدأ الغوص بداخل تطوير المواقع بإستخدام ASP.Net , و هذا بدوره سيعطيك فهماً عميقاً إن شاء الله .

هنا نرى أن المستخدم أو ما يطلق عليه جهاز العميل, وبالفعل يطلق عليه المتصفح . هذا المتصفح هو المسئول الأول عن إرسال الطلبات للخادم , وفى بعض الأحيان قد لا يحتاج إلى العودة إلى الخادم . فعند الحاجة إلى تعديل بسيط فى شكل عنصر فى الصفحة كتغير لون أو إخفاء أو حتى إظهار هذا العنصر ,فإن مثل هذة العمليات تنفذ على جهاز العميل وعندها لا حاجة تتطلب العودة إلى الخادم

وقد نعطى مبدأ هام وهو " لا عودة إلى الخادم إلا عند الحاجة الملحة لذلك", وما هى تلك الحاجة التى ستجعلك تتكبد هذا العناء فى رحلة طويلة ,تبدأ من جهاز العميل إلى الخادم ليرحل ومعه كامل بيانات الصفحة ثم يعود أيضاً بكامل الصفحة,وقد تم تحديث جزء فقط من الصفحة والبقية كما هى.

ألا يعتبر هذا إهداراً لموارد النظام من إشغال للمعالج وكذلك الذاكرة .ولكن لماذا نضطر إلى العودة إلى الخادم ؟ , حسناً , سأجيب على هذا السؤال بالتالى: هل تذكر أنك وضعت قاعدة البيانات على خادم فى مكان ما ,فكيف يمكن للمستخدم المقيم فى الجهة الأخرى أن يعرف ما هى رحلات الطيران المتوافر فيها أماكن للحجز عند تعامله مع تطبيق قمت بعمله لشركة طيران عالمية :) . ففى هذة الحالة , وجب أن يرسل المستخدم إلى الخادم طلباً بالبيانات المراد الإستعلام عنها, وهى على سبيل المثال, إسم أو رقم البلد المتجه إليها العميل , وكذلك التى سيغادرها . فيرد عليه الخادم بأرقام الرحلات ومواعيدها والتكلفة المالية. فهنا كانت الحاجة ملحة للخوض الرحلة لتحديث الصفحة .ولمساعدة العميل فى إيجاد ما يريد .ونتمنى له رحلة طيبة :) .

Ajax جاءت لتنقذ الموقف

رجوعاً إلى المثال السابق الخاص بتطبيق شركة الطيران, قد يضطر المستخدم الى الإنتظار لحين عودة الطلب من الخادم , وقد يقف تحميل الصفحة تماما , ولن يستطيع المستخدم حينها التفاعل مع الصفحة لحين العودة, وهذا قد يسبب الملل للمستخدم, مما قد يدعوه لترك الموقع , وبهذا تخسر كل يوماً زبوناً ! .....هل تذكر حين ذكرنا رحيل الطلب بكامل الصفحة من جهاز العميل إلى الخادم ليعود بكامل الصفحة وقد تم تحديث جزء منها فقط . وفى الغالب أنت تقوم بتحديث

جزء فقط ,لإنك إن قمت بتحديث كامل الصفحة فأنت تتحدث عن صفحة جديدة.فدعنا نركز على جزء من الصفحة.

,أليس من المنطقى أن يتم إرسال الجزء المراد تحديثه فقط .ويتم ترك باقى الصفحة كما هو ؟! .بلى , ولكن فى الشكل الطبيعى هذا غير جائز . فكان ظهور Ajax ظهور المنقذ , تلك التقنية التى تعتمد على JavaScript بالعمل مع XML لإرسال وإستقبال HTTP Requests , فى الواقع يعرف بـإسم XMLHTTPRequest . وهو HttpRequest قائم على XML فى عملية الإرسال والإستقبال . والتى تتولى عملية الإرسال والإستقبال هى لغة JavaScript لتقوم بدورها بتحديث عناصر الصفحة التى هى بالفعل HTML بعد عملية الإستقبال .إذاً فهى تقوم بتحديث كود HTML.

هرع الكثير من المبرمجون إلى العمل مع مكتبة Ajax ,التى تقدم العديد من المزايا التى تهدف فى المقام الأول إلى مصطلح "Partial Update " والترجمة هى "التحديث الجزئى " .تشمل هذة المكتبة ما يعرف بـ Namespaces , Classes , Events , والتى تقدم سهولة التعامل مع العديد من المزايا الخاصة بـ ASP. Net مثال ذلك: تحديد الهوية Authentication , وكذلك الإعدات الشخصية Profiles وكذلك التعامل مع خدمات الويب Web Service , وتدعم كذلك العمل مع نظام المجموعات وهو ما يعرف إنجليزياً بـ Roles , ولا ننسى التعامل مع تعدد اللغات وهو ما يعرف بإسم Globalization . وليس هذا فحسب . جاءت كذلك بعض الأدوات المصاحبة لهذة المكتبة وهى ما باتت تعرف بـإسم Ajax Sever Controls ومجموعهم خمسة أداوت :

- ScriptManager
- ScriptManagerProxy
- UpdatePanel
- UpdateProgress,
- Timer

وهذة الأدوات أصبحت جزءاً داخلياً من الإصدارت الحديثة لبيئة التطوير الخاصة بـ ASP.Net .

برع أيضاً الكثير من المطورون فى إستخدام هذة المكتبة فى إعداد الكثير من الأدوات الفعالة, التى تقوم فى الأساس على تقنية Ajax , ومن هنا جاءت لنا أدوات باتت تعرف بإسم Ajax Control Toolkit .وهناك الكثير منها ,حتى إن بإمكانك أن تقوم بعمل أداتك الخاصة التى تفى بإحتياجاتك . هذا فقط بعد أن تتعلم كيف تعمل هذة المكتبة وكيف تستخدمها.

كانت هذة نبذة سريعة على Ajax ولن تحتاج إلا لمعرفة وفهم المصطلحات التى ذكرت من قبل وكيفية التعامل أيضاً مع المكتبة والأدوات معاً .

ننتقل إلى جزء هام جداً وهو معرفة خط سير الطلب من جهاز المستخدم إلى جهاز الخادم , وما هى المراحل التى يمر بها الطلب حتى يعود إلينا مرة أخرى بما أرسلناه من أجله .

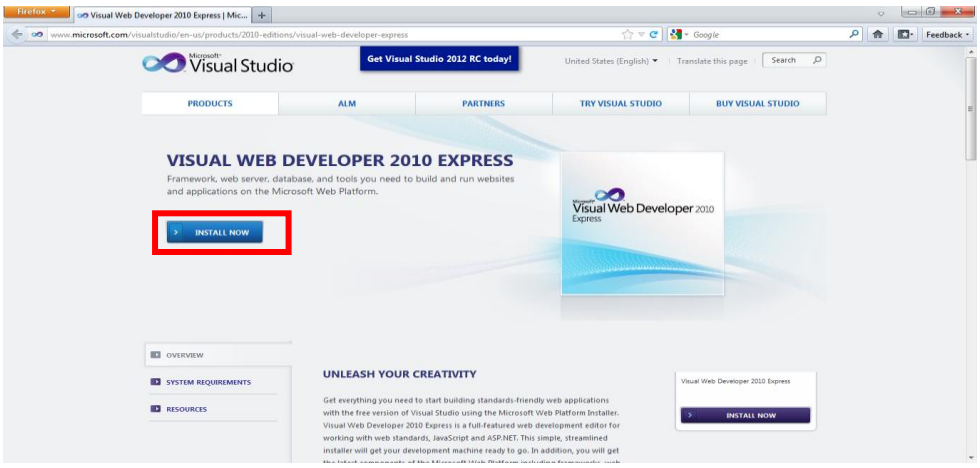
على الرغم من أن الحديث السابق يعد أمراً نظرياً , قد يساعد قليل من الحفظ وكثيراً من الفهم على إستيعابه , ولكنه كان ضرورياً , ولكن نكتفى بهذا القدر وننتقل إلى الجزء العملى .

كيف تحصل على أدوات العمل

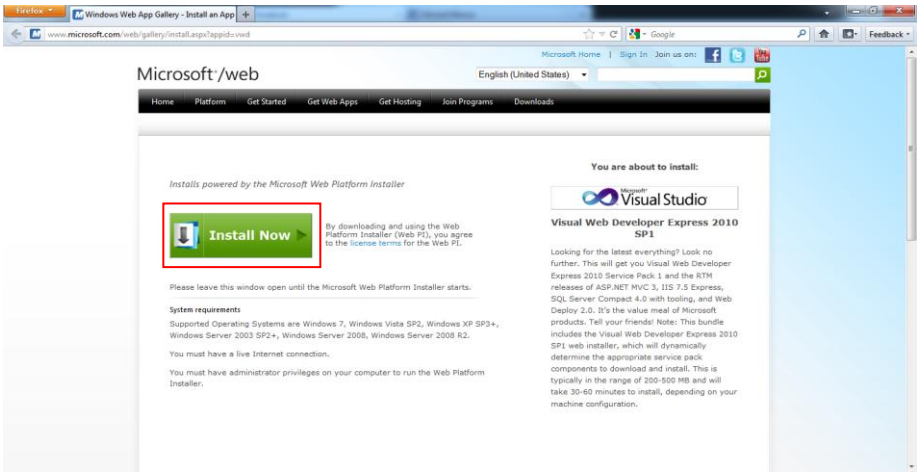
بالنسبة لى سأقوم بالعمل مع Visual Web Developer Express 2010 (VWD), وهو النسخة المجانية التى قدمتها مايكروسوفت للطلاب والدارسين ,وتستطيع أن تحصل عليها من هذا الرابط :-

www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-web-developer-express

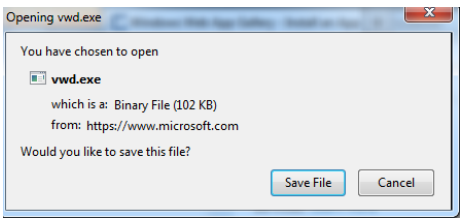
وعبر هذا الرابط تجد الصفحة أمامك كالشكل التالى :-



كما ترى فى الصورة التى أمامك , يمكنك تثبيت البرنامج من خلال الضغط على Install لتظهر لك النافذة التالية والتى بها أيضاً Install Now كما ترى :-



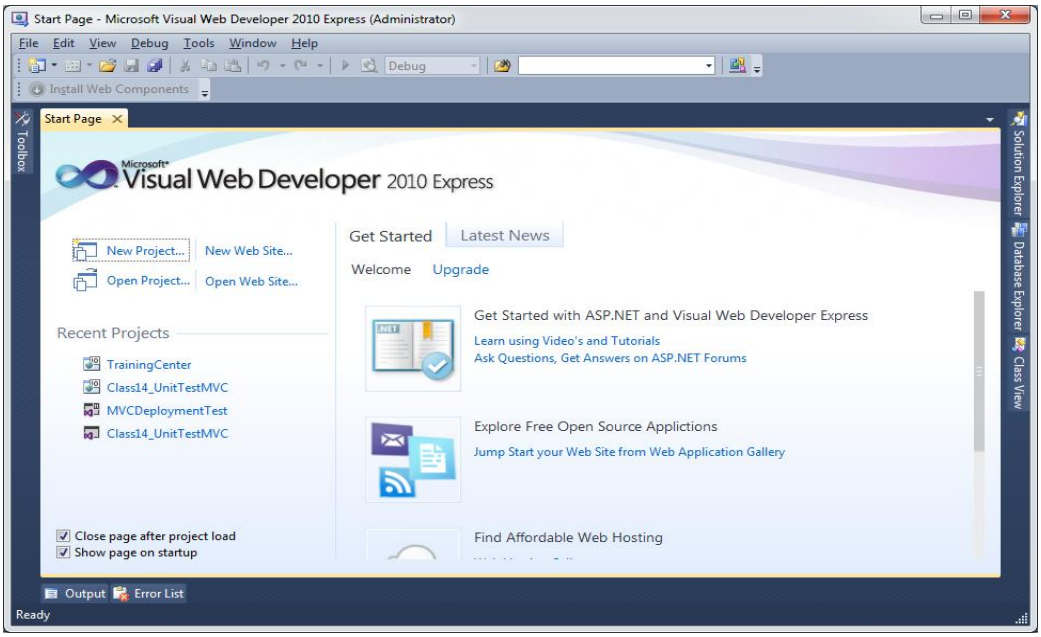
لتجد النافذة التالية :-



بعد الضغط على Save سيتم تحميل برنامج Web Platform Installer والذى يسمح لك بتثبيت ASP.Net , ومنه سنقوم بإختيار Visual Web Developer- Express , ومن ثم Install لتتبع الخطوات بعدها , وبالطبع بعد التثبيت ستجد البرنامج قد تم إدراجه ضمن قائمة Start ويمكن تشغيله من هناك .

من المفترض أنه قد أصبح البرنامج لديك الآن , فهيا لنتعرف على واجهة البرنامج بشكل سريع , لندخل فى أول تمرين لنا .

التعرف على واجهة البرنامج



إليك تعريفات مختصرة لكل نافذة تراها أمامك :-

إسم النافذة	وظيفتها
Solution Explorer	نافذة تحتوى على ملفات موقعك ,ومن خلالها تستطيع التنقل بين ملفات الموقع من صفحات وملفات مساعدة .
Data base Explorer	تستطيع من خلالها التفاعل مع قاعدة البيانات كإضافة جدول أو حذفه او التعديل فى أعمدته أو صفوفه.
Class View	تستطيع من خلالها تصفح محتويات الـ Classes الموجودة فى التطبيق الحالى لترى ما بها من دوال وخصائص ووراثة .
Toolbox	تستطيع من خلالها رؤية الأدوات المتاحة والتى يقدمها لك VWD .
Output	تستطيع من خلالها معرفة حالة مخرجات عمليات البناء والنشر لملفات التطبيق , سواءً تم بنجاح أم هناك مشكلة .
Error List	ترى من خلالها الأخطاء والتحذيرات الموجودة داخل الكود .

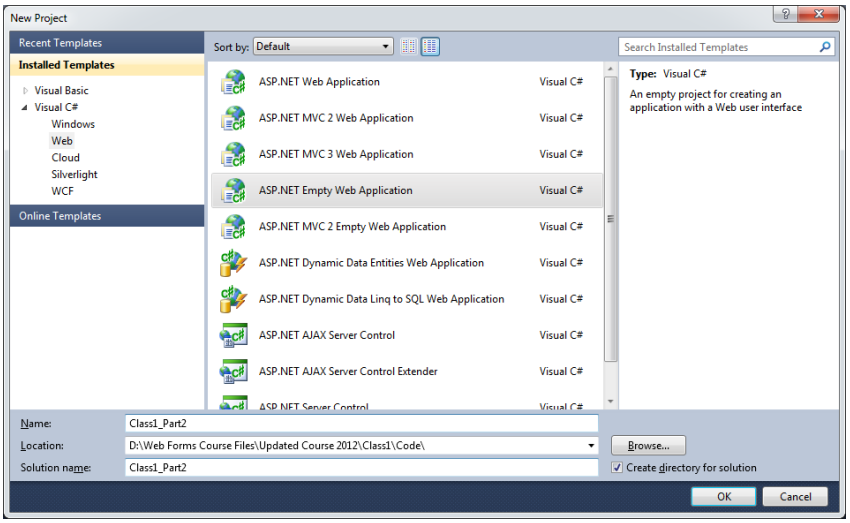
ملحوظة :- لمزيد من المعلومات حول واجهة البرنامج , يمكنك مشاهدة الملف المرئى Class1_Part1 فى مجلد Video .

التمرين الأول

Hello World

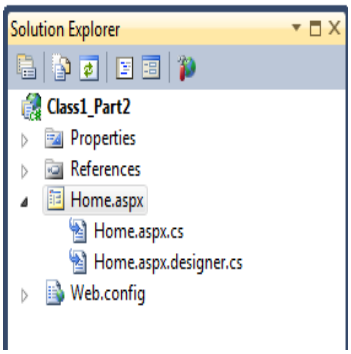
من خلال هذا التطبيق سنقوم بإستعراض محتويات صفحة aspx من الداخل وما يرتبط بها من ملف Code-Behind .فإليك خطوات إنشاء هذا التطبيق :-

1 -بعد تشغيل برنامج Visual Web Developer ,والذهاب إلى قائمة File لنقوم بإختيار New Project ومن ثم نختار ASP.Net Empty Application .



هنا نقوم بإضافة مشروع فارغ لنتعرف على كيفية أنشاء تطبيق من الصفر , فبعد إنشاء المشروع قم بإضافة صفحة aspx من خلال Add New Item ولتكن بإسم Home.aspx , لترى ماذا بها .

فى البداية لننظر إلى ملفات التطبيق :-



هنا تجد أننا قد أضفنا بالفعل الصفحة Home.aspx , يصاحبها ملف Home.aspx.cs , والملف Home.aspx.designer.cs . فهيا لنأخذ تفصيلاً عن هذة الملفات على حده :-

Home.aspx

يمثل هذا الملف صفحة aspx , ويحتوى على أكواد تصميم الصفحة , وهذا هو محتوى الصفحة :-

```
<%@  Page Language="C#" AutoEventWireup="true" CodeBehind="Home.aspx.cs" Inherits="Class1_Part2.Home" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Welcome to Home Page </title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

ولنأخذ أول سطر فى هذة الصفحة وهو :

```
<%@  Page Language="C#" AutoEventWireup="true" CodeBehind="Home.aspx.cs" Inherits="Class1_Part2.Home" %>
```

<% %>	تسمى هذة العلامات Server Tags , ويتم إستخدامها لإدراج Server-Code بداخل أكواد HTML .حيث يمكن أن تجد منها أشكالاً كثيراً مثال :- <% %> وتستخدم لإدراج Server-Side Code . <%@ %> وتستخدم لإدراج تعليمات للصفحة , فإما لتعريف الصفحة أو إدراج namespace أو Control أو غير ذلك كثير . <%= %> وتستخدم كأنها Response.Write وتتوقع بعد علامة = أن يأتى Object يحمل قيمة, ولا يصح أن تكون void. <%# %> وتستخدم فى عمليات Binding, أى عمليات ربط الآدوات بمصدر البيانات وهو فى الغالب قاعدة بيانات. <%$ %> وتستخدم لإدراج تعبيرات, كإحضار قيم من ملف Web.Config أو ربما ملف Resources أو قد تستخدمها مع إحضار قيم من Routing. <%-- --%> وتستخدم لإدراج تعليقات . وقد كتبت مقالة على الإنترنت بعنوان " Inline Server Tags سلسلة دروس " إبحث عنها فى جوجل .
@ Page	يسمى هذا Page Directive , وبه نعطى تعليمات للصفحة ببعض الإعدادات التى يجب مراعاتها .

هنا نحدد أن لغة أكواد هذة الصفحة مكتوباً بلغة C# .	`Language="C#"`

تساعد هذة الخاصية فى تعرف الصفحة على أحداثها , حيث يقوم بالبحث عن كل حدث يبدأ بكلمة Page ثم _ ثم إسم الحدث , أى على هذا الشكل Page_EventName , ولهذا تتعرف الصفحة على Page_Load وهو حدث تحميل الصفحة.
 	
فإذا كانت هذة الخاصية تحمل القيمة false , فهناك مشكلة , لإن الصفحة لن تتعرف على الأحداث المعرفة بداخل ملف code-behind , وتحتاج حينها أن تخبر الصفحة بإسماء الأحداث ومثال على ذلك أن تقوم بعمل التالى :-
 	
`public partial class _Default : System.Web.UI.Page`
`{`
`public _Default()`
`{`
`this.Load += new EventHandler(PageLoading);`
`}`
`protected void PageLoading (object sender, EventArgs e)`
`{`
`Response.Write("Hello World! ");`
`}`
`}`
	`AutoEventWireup="true"`

هنا نحدد إسم ملف الكود الخلفى (Code-Behind) الذى يصاحب صفحة aspx والهدف من وجوده هو إحتواء أكواد الصفحة وهو بالطبع Server-Code , ومعنى Server-Code أنه أكواد ستعمل على الخادم مثال C# أو Vb.net.	`CodeBehind="Home.aspx.cs"`
هل تعلم أن صفحة aspx يتم تحويلها إلى Class وقت التنفيذ ,فيتم جعل هذا الـ Class ترث من ملف Code-behind ,والذى يتمثل هنا فى Home Class .لمزيد من المعلومات عن هذا الموضوع , يمكنك الجوع إلى مقالة كتبتها على مدونتى ,بعنوان "مقدمة تعريفية بـ ASP.Net الجزء الخامس: ASP.Net Compilation "	`Inherits="Class1_Part2.Home"`

ولاحظ أن Page Directive والذى تم شرحه بالأعلى لا يتم إرساله إلى المتصفح .

بعد ذلك تجد فى الصفحة أنه قد بدأها بعنصر يسمى Html وهى لبداية الصفحة التى سيتم إرسالها إلى المتصفح ,ثم بداخله بدأ يكتب الكود الذى نستخدمه لتصميم شكل الصفحة وما تحتويه من عناصر , فعلى سبيل المثال , بدأ برأس الصفحة Head وحدد بداخله عنوان الصفحة Title والذى يظهر فى الشريط العلوى للصفحة ,تبعه وجود العنصر Body والذى بداخله كامل محتويات الصفحة والتى تظهر للمستخدم فى المتصفح , ولإن الصفحة ستحتوى

على Server Controls وجب أن يكون بداخل العنصر Body عنصراً آخر يسمى form بشرط أن يحمل الخاصية runat=server حيث نحدد بها أن العنصر form هو Server Control , ومن هذا فإن أى عنصر يحتوى على runat=Server فإنه عنصر أو آداة تعمل على الخادم , فيمكن التحكم به والتفاعل معه عبر أكواد الخادم والتى ستكون فى الغالب داخل ملف Code-behind مكتوبةً بـ C# أو Vb.net .

ومن هذا نستخلص التالى :-

- كل صفحة aspx لابد وأن تحتوى على Form واحد فقط يحمل الخاصية runat , وفى حالة وجود أكثر من واحد فإنك ستقابل رسالة خطأ توقف العمل .
- كل أداة Server Control أى تحمل الخاصية runat لابد وأن تتواجد ضمن Form يحمل الخاصية runat أيضاً .

داخل العنصر form تجد عنصر يسمى Div ووجوده بمثابة حاوياً أو إطاراً (Container) ليحمل بداخله عناصر آخرى , وهو للتنظيم ليس أكثر , و لا مشكلة فى وجوده من عدمه .

أيضاً وجب التنبيه على أن كل عنصر يتم فتحه لابد من غلقه , مثال , <form> لابد أن يتبعها </form>.

Home.aspx.cs

هذا الملف الذى تحدثنا عنه بالأعلى وعرفنا أن إسمه هو Code-behind , وما هو إلا Class تقوم بالوراثة من System.Web.UI.Page وتأتى على مثل هذا الشكل :-

```
namespace Class1_Part2
{
    public partial class Home : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

وبداخلها يتم كتابة الكود الخاص بالصفحة ,فقد يكون هذا الكود للإتصال بقواعد البيانات وقد يكون لإرسال بريد الكترونى أو غير ذلك من المهام التى تتطلب تنفيذها على الخادم .

كذلك نجد بداخله أحداث الصفحة والآدوات التى بداخلها , فقد تجد حدث خاص بالآداة GridView وآخرللأداة DropDownList إضافة إلى أحداث الصفحة كحدث Page_Load وحدث Page_Preinit وسيأتى إن شاء الله الحديث عن دورة حياة الصفحة بكل الأحداث والدوال التابعة لها .

كذلك يمكن ضبط خصائص ودوال الصفحة من خلال الخاصية Page , والتى تمثل الصفحة الحالية , أو قد تستخدم الكلمة المحجوزة this والتى تشير إلى الـClass التى نحن بداخلها.

ولاحظ أن class Home يسبقها كلمة Partial والتى بدورها تسمح للـClass أن يتم تجزئتها على أكثر من ملف . وهذا ما سنلاحظه فى الملف القادم .

Home.aspx.designer.cs

كما يتضح من الاسم أن هذا الملف هو للتصميم , ولكن تصميم ماذا ؟ _____ تصميم أداوات الخادم المعروفة بإسم Server Controls , فما رأيك أن تقوم بسحب Button من شريط الأدوات إلى اصفحة ثم تذهب إلى هذا الملف مرة آخرى , ألن يكون المحتوى على مثل هذا الشكل :-

```
namespace Class1_Part2 {


    public partial class Home {

        /// <summary> ...
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        /// <summary> ...
        protected global::System.Web.UI.WebControls.Button Button1;
    }
}
```

ألا ترى أنه لا يحتوى إلا على نسخة من Button و Form وهما عنصر من أدوات الخادم , أى يحملان الخاصية runat وكذلك معّرف الهوية من خلال الخاصية (ID) .

ولاحظ أن هذا الملف لا دخل لك فيه , إنما هو من عملية compilation , ولكن هو يفسر ما تحدثنا عنه , وهو أن الصفحة وأداواتها أثناء وقت التنفيذ ما هى إلا Instance من Class , وسيأتى حديثاً مطولاً حول الجملة التى قد تكون غامضة الآن إن شاء الله .

ملحوظة:-

لمزيد من المعلومات فى هذة النقطة يمكنك الآن مشاهدة ملف الفيديو بعنوان Class1_Part2 .

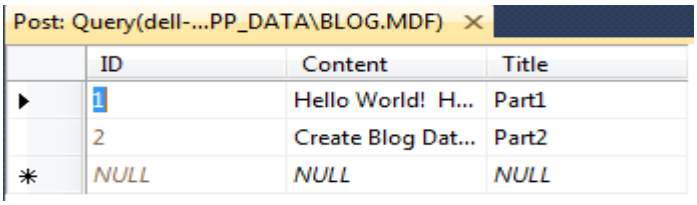
التمرين الثانى

هنا سنقوم بإنشاء تطبيق يعمل مع قواعد البيانات , وسنقوم بالإتصال بالقاعدة من خلال إستخدام Entity Framework. فهيا للخطوات :-

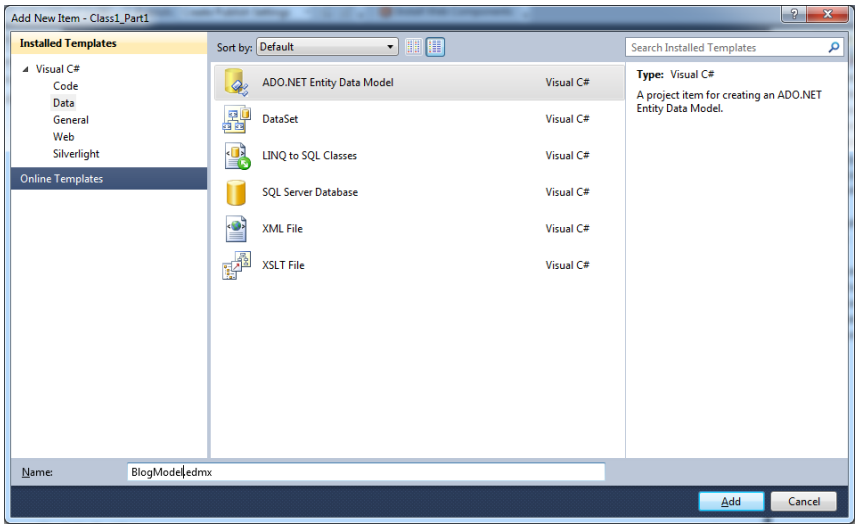
1- قم بشتغيل Visual Web Developer , ومن قائمة File , نقوم بإختيار New Project .
2- ومن ثم يمكنك تحديد ASP.Net Web Application وقم بتسميته BlogApp .
3- قم بالوقوف على مجلد App_Data , ومن ثم R.C ثم إختر من القائمة Add New Item لإضافة SQL Database File وليكن إسمها Blog.mdf .
4- قم بإلضغط مرتين على القاعدة لتظهر أمامك نافذة Data base Explorer ومنها تستطيع أن تنشأ New Table, وليكن إسمه Post وإجعله موافقاً الشكل التالى :-

Primary Key	Column Name	Data Type
Yes	ID	Int
	Content	Nvarchar(2000)
	Title	Nvarchar(200)

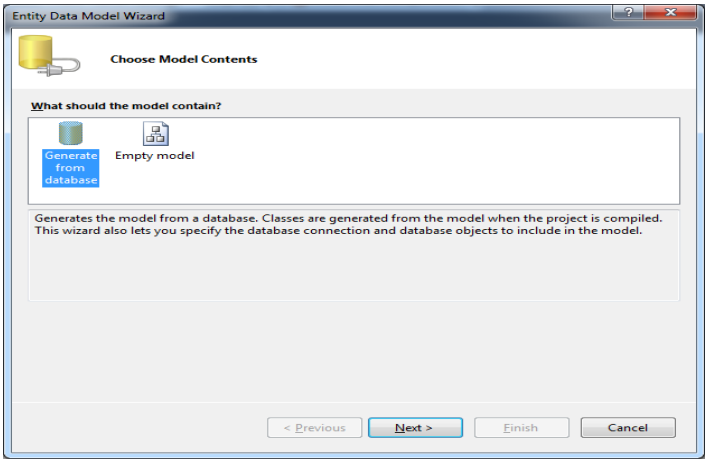
5 -بعد ذلك يمكنك الوقوف على الجدول فى نافذة Database Explorer و من ثم إختيار R.C ومن نختار من القائمة Show Table Data , لنقوم بملئ بعض البيانات يدوياً.



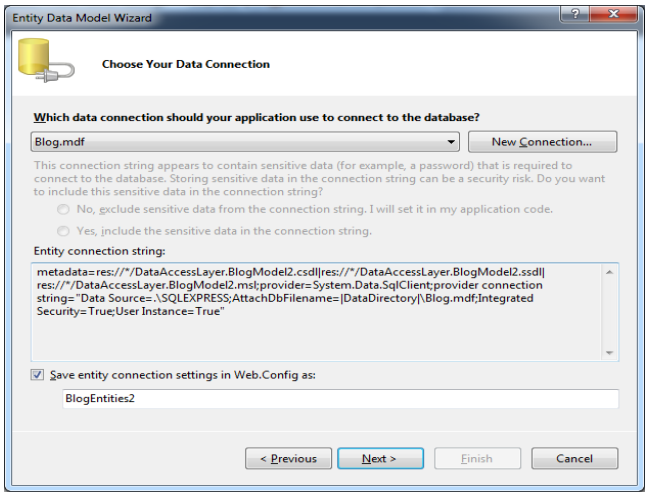
6 -نتوجه الآن إلى نافذة Solution Explorer ومنها نقوم بإنشاء New Folder ونقوم بتسميته DataAccessLayer, وبه سنقوم بإضافة Entity Data Model وذلك من خلال الوقوف عليه وإختيارها من Add New Item , وليكن إسمها BlogModel



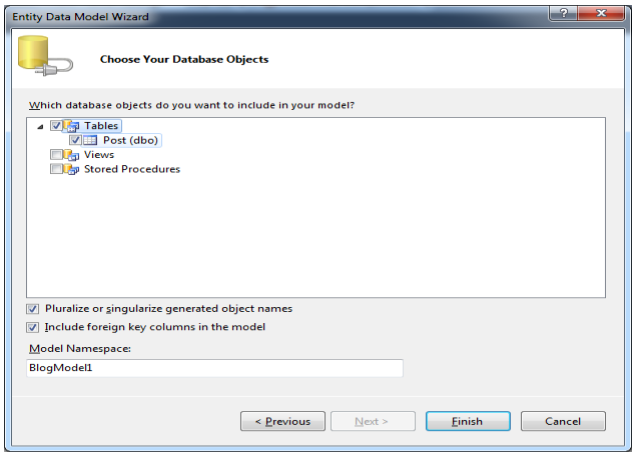
7 -لتظهر أمامك النافذة التالية والتى نحدد من خلالها أننا نريد إنشاء Model ناتج عن قاعدة بيانات تم إنشائها سابقاً , وسنقوم بإختيار Generate From database :-



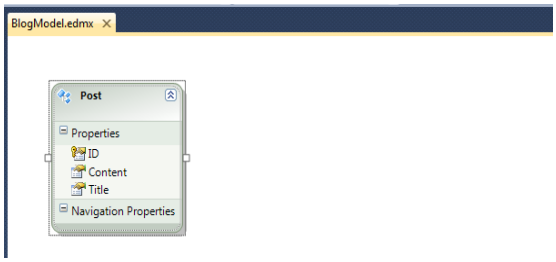
8 -لتظهر أمامك بعد ذلك النافذة التالية والتى تحدد فيها إسم القاعدة التى تريد العمل معها:-



9 -وبعد الضغط على Next سنقوم بإختيار الجدول الذى نريد العمل معه :-



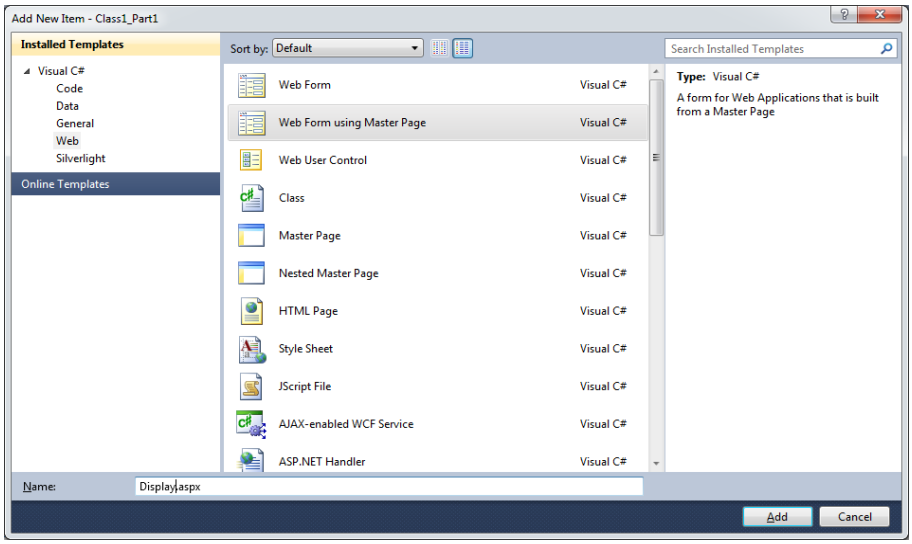
10 - وبعد الضغط على Finish , فقد أنهيت إنشائها EF Model , لتظهر أمامك كما ترى ويمكنك إستخدامها بعد ذلك :-



يجب عليك أن تعرف أن ما فعلته مع Entity Framework فى الخطوات السابقة, هو تمثيل الجدول الذى قمنا بتحديده وهو هنا Post بـــ Class , أى قامت بإنشاء class يمثل الجدول الموجود فى قاعدة البيانات. وذلك بهدف سهولة التعامل معه حيث سيكون هذا التعامل فى الذاكرة , إضافة إلى هذا , تحويل كل حقول الجدول إلى خصائص داخل ال Class كما ترى بالشكل . كيف تقوم بتحويل البيانات من Class إلى جدول , وما هو السر وراء هذا الأمر, وما هى مكونات ملفات entity , وما هى الإنماط التى تستخدمهاentity , وماذا يقصد بـ ORM ؟ هذا ما سنراه إن شاء الله , فى الدرس الخاص بذلك .

إلى هذا الحد , أضفنا قاعدة البيانات , وأضفنا (, entity Framework) EF والتى ستعمل كـ Data Access Layer .

11 - بعد ذلك يمكننا إنشاء New Folder , بإسم Posts ونضع به صفحة بإسم Display ,على أمل أن نعرض بها مجموعة المواضيع المتواجدة داخل جدول Post.
مع مراعاة أن الصفحة Display.aspx ستكون متفرعة من MasterPage :-



12 - سيكون محتوى صفحة Details.aspx على الشكل التالى :-

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"
    CodeBehind="Display.aspx.cs" Inherits="Class1_Part1.Posts.Display" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <asp:Repeater runat="server" ID="PostsRepeater">
        <ItemTemplate>
            <h1>
                <%# Eval("Title") %></h1>
            <p>
                <%# Eval("Content").ToString().Substring(0,50) %></p>
            <a href='<%# Eval("ID" ,"details.aspx?id={0}") %>'>More</a>
        </ItemTemplate>
        <SeparatorTemplate>
            <hr />
        </SeparatorTemplate>
    </asp:Repeater>
</asp:Content>
```

هنا فى هذة الصفحة نقوم بإستخدام Repeater , لعرض المواضيع بداخله , وبداخلها إستخدمنا دالة Eval والتى تعمل مع وجود مصدر بيانات , حيث نمرر لها أسماء الأعمدة التى نريد عرضها , فعلى سبيل المثال قمنا بتمرير Title لعرض العنوان . و كما ترى هذة العلامات <% %> يأتى بداخلها كود Server-Code , وهذا هو الهدف من هذة العلامات , حيث يتم إستخدامها فى إدراج Server-Code ضمن أكواد Html .ولاحظ أننا أيضاً سنقوم بعرض 50 حرفاً من محتوى الموضوع بإستخدام الدالة Substring.

13 - إلى هنا يأتى دور إضافة أكواد الإتصال بـ Entity Model والحصول على بيانات ليتم عرضها داخل لاصفحة Display.aspx , فنحتاج الآن أن نذهب إلى ملف Code-behind , لإضافة بعض الكود بداخل حدث Page_Load :-

Display.aspx.cs

Class1_Part1.Posts.Display | Page_Load(object sender,

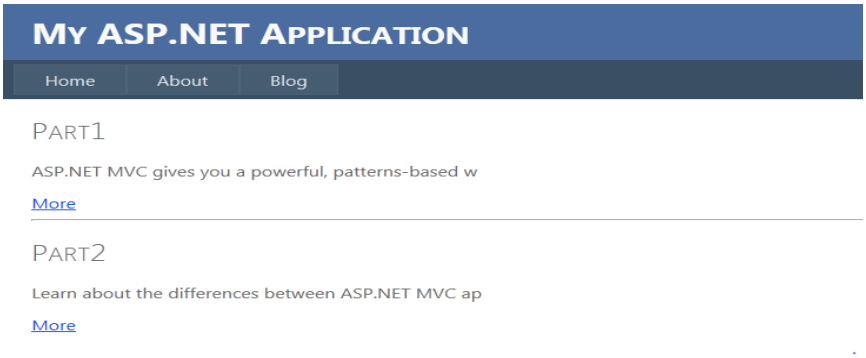
```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Class1_Part1.DataAccessLayer;
namespace Class1_Part1.Posts
{
    public partial class Display : System.Web.UI.Page
    {
        BlogEntities1 postentity;
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                postentity = new BlogEntities1();
                PostsRepeater.DataSource = postentity.Posts;
                PostsRepeater.DataBind();
            }
        }
    }
}
```

14 - نحتاج هنا أن نضع رابطاً للصفحة Display.aspx داخل صفحة Site.Master , ولهذا نذهب إلى صفحة site.Master وبها نضع الرابط المظلل بالأسفل :-

```
<asp:Menu ID="NavigationMenu" runat="server" CssClass="menu"
EnableViewState="false" IncludeStyleBlock="false" Orientation="Horizontal">
    <Items>
      <asp:MenuItem NavigateUrl="~/Default.aspx" Text="Home"/>
      <asp:MenuItem NavigateUrl="~/About.aspx" Text="About"/>
      <asp:MenuItem NavigateUrl="~/Posts/Display.aspx" Text="Blog"/>
  </Items>
</asp:Menu>
```

هذا الرابط بدوره سيذهب إلى صفحة Display.aspx بداخل مجلد Posts ولاحظ أن نص الرابط سيكون كلمة Blog .

15 - جاء الآن دور تشغيل التطبيق ومعاينة العمل , فبعد الضغط على Blog التى أمامك فى الشكل التالى سيظهر لك ما ترى :-



ينقص هذا المشروع التنسيقات , وستراها إن شاء الله فى الدرس الخاص بـ CSS .