الفصل الأول
التعرف على Visual Basic داخل تقنية .NET

ربما تتساءل عن بيئة عمل **.NET Framework** وعن علاقتها بموضوع الكتاب وهو **Visual Basic 2012** وعن الأهداف التى من أجلها تم تطوير هذه التقنية. وهذا ما سنحاول سريعاً التعرف عليه من خلال هذا الفصل.

بانتهاء هذا الفصل ستتعرف على:

- لمحة عن تطور **Visual basic**.
- ما هى تقنية ***.NET***؟
- **Visual Basic** وبيئة **.NET Framework**..
- نموذج **.NET Framework** التنفيذى.

لمحة عن تطور Visual Basic

كانت Visual Basic 6 برنامجاً مستقلاً ، أما Visual Basic 2012 فهي عبارةعن لغة في نظام تطوير أكبر، فإذا عدنا إلي الوراء للتعرف علي أصول Visual Basic فقد كانت Basic لغة برمجية يتم استخدامها منذ عشرين عاماً كجزء من MS-Dos. وفي عام ١٩٨٥ تحولت Basic إلي Visual Basic كما أصبحت جزءاً من أداة إنشاء التطبيقات الخاصة بالـ Windows.أما بخصوص برنامج Visual Basic 6فهو ليس مجرد لغة –بل كان يمكنه إنشاء النماذج الخاص به – علي سبيل المثال – أما بالنسبة لـ Visual Basic 2012 فيملك أداة إنتاج نماذج جديدة توفر له وسيلة جديدة للتفاعل مع نظام التشغيل Windows تلك هي بيئة .NET Framework. التي تحيط بلغة Visual Basic 2012.

عندما تنظر إلى Visual Basicفمن المحتمل أن تفكر فى تطبيقات Windows الخاصة بـ Microsoft. فعلى مدى خمسة عشر عاما استخدم المطورون Ruby Forms Engine الخاصة ببرنامج Visual Basic. لكتابة تطبيقات العمل الشائعة، وعندما كان يتم تعريفبرنامج كبرنامج Visual Basic، فهذا معناه أنه تطبيق Windows بشكل افتراضى.

لكن زاد مجال Visual Basic واستخدامها مع ظهور Visual Basic.NET و Visual Basic.NET هى عبارة عن لغة مثل C++ أو COBOL أو Java والتى يمكنك استخدامها لكتابة أى نوع من التطبيقات مدعم من قبل (Application programming interface) API. والآن عندما يتم تعريف برنامج كبرنامج Visual Basic، فيجب أن تطرح الأسئلة التالية : ما نوع البرنامج ؟ هل هو تطبيق Windows أو موقع ويب؟ هل هو خدمة Windows أو خدمة ويب XML؟

يمكن القول أنه مازلت هناك حاجة كبيرة لإنشاء تطبيقات Windows – والتى يطلق عليها الآن تطبيقات Windows Forms. فبالرغم من أن تطبيقات الويب تزداد الحاجة

إليها يوماً بعد يوم، فإن بيئة العمل الفعالة لأى تطبيق Windows ليست مألوفة فحسب، بل لها أيضا العديد من المزايا التى من الصعب أن تتوفر بالنسبة لأى نوع تطبيق آخر.
فى بيئة .NET.أصبحت اللغة البرمجية مجرد وسيلة للتفاعل مع بيئة العمل وبالتالى مع نظام تشغيل Windows فجميع البرامج تحتاج الى مجموعة من القواعد المحددة (بالنسبة لاتخاذ القراروعمليات التكرار وما شابه) داخل البرامج للحصول على التطبيق المطلوب توفر لغة Visual Basic مثل هذه المجموعة من القواعد فى حين توفر بيئة العمل الكائنات والأحداث التى سيتم التعامل معها.

إصدارات Visual Studio

توجد بيئة تطوير Visual Studio فى عدد من الإصداراتالأساسية كالتالى :

- **Visual Studio Professional Edition**: يعتبر هذا الإصدار شائعاً بصورة كبيرة بالنسبة للمطور المنفرد أو لتطوير تطبيقات متوسطة الحجم، بما فى ذلك إنشاء تطبيقات Windows 8 إلا أنه لا يدعم استخدام MSDN وإنما يوجد منه إصدار آخر لذلك يسمى Visual Studio Professional 2012 with MSDN.
- **Visual Studio Premium**: ويحتوى على جميع السمات الموجودة بالإصدار Professional بأشكاله المختلفة بالإضافة إلى العديد من أدوات اختبار الكود الإضافية والمتنوعة.
- **Visual Studio Ultimate**: وهو أعلى إصدارات Visual Studio 2012 على الإطلاق، فهو يحتوى على جميع السمات والأدوات الموجودة فى كلٍ من الإصدارين السابقين بالإضافة إلى العديد من أدوات التتبع الذكية للكود وتصحيح ما به من أخطاء وكذلك اختبار كفاءة الويب. وهذا هو الإصدار المستخدم فى هذا الكتاب.
- **Team System**: لقد تم تصميم هذا الإصدار لفرق البرمجة فى الشركات الكبرى.وهو يتضمن أدوات تصميم على نطاق كبير مثل التطوير المعتمد على عمليات الاختبار و Team Foundation Server.

ما هى تقنية .NET؟

أرادت مايكروسوفت من خلال تقنية **.NET** توفير بنية تحتية مشتركة لجميع المطورين والمستخدمين على حد سواء تهدف أساساً إلى تحويل نظرة البرمجة والبرمجيات من مفهوم الحاسبات الشخصية المستقلة أو الشبكات الصغيرة إلى مفهوم إتاحة البيانات وعرضها فى جميع أنحاء العالم من خلال شبكة الإنترنت. وقد أطلقت مايكروسوفت فى البداية على هذا النطاق الجديد المصطلح **Next Generation Windows Services (NGWS)** ثم عادت وأسمته **.NET**..

يحتوى **.NET Framework** (نطاق **.NET**) حقيقةً على وفرة هائلة من التصنيفات والبنية التحتية والأدوات التى تمكن المطورين من تطوير تطبيقات عالية الدقة والجودة بسهولة منقطعة النظير مقارنةً بالإصدارات السابقة، حيث يحتوى هذا النطاق على عدد من لغات البرمجة القوية والشهيرة مثل **Visual C++ 2012** ولغة **Visual C# 2012** إلى جانب لغة **Visual Basic 2012** موضوع هذا الكتاب، بالإضافة إلى تضمين **ASP.NET** و**ADO.NET** كعناصر أساسية داخل النطاق.

وقديماً كنت فى حاجة إلى استخدام وإتقان لغة واحدة من لغات البرمجة إذا أردت تطوير تطبيق من التطبيقات يعمل على الأجهزة الشخصية أو من خلال خادم إحدى الشبكات. أما الآن فالأمر تغير إلى حد ما، حيث يلزمك التعرف على العديد من التقنيات والمهارات مثل صفحات الخادم النشطة **Active Server Pages (ASP)** ومكونات **COM** و**HTML** و**XML** و**VBScript** و**J Script** وغيرها من التقنيات الأخرى إذا أردت تطوير تطبيق متعدد الأطراف **n-tier Application**. ومن حسن الحظ تضمين جميع هذه التقنيات داخل نطاق **.NET** (**.NET Framework**). كما يمكنك داخل **.NET Framework** (نطاق **.NET**). اختيار اللغة التى تشعر تجاهها بالراحة كما يمكنك إنشاء وتطوير تطبيقك باستخدام أكثر من لغة، حيث تتيح **.NET** التداخل بين اللغات.

وإذا كان الموطن الطبيعى للبرامج الآن هو الحاسبات الشخصية، فإن استخدام .NET فى تطوير التطبيقات والبرامج يعمل على نقل موطن هذه التطبيقات أو تلك البرامج إلى أعالى البحار ليكون داخل الويب وهو ما يعنى إتاحتها للعالم بأثره. والهدف من هذا الكتاب أن تتعرف على الكيفية التى يتعامل بها Visual Basic 2012 مع بيئة عمل .NET Framework الخاصة بنظام Windows.

تحويل البرامج إلى خدمات

قديماً كنا نطلق على البرامج أو السمات المتاحة لشريحة من الناس كلمة "خدمة" Service مثل برنامج إظهار رقم الطالب الموجود بشركات الاتصالات فهو عبارة عن خدمة عامة بأجر رمزى معين وكذلك خدمة البريد الإلكترونى المتاحة عبر الويب. وقد أرادت مايكروسوفت من خلال .NET معاملة جميع البرامج كخدمات متاحة لجميع المستخدمين من خلال شبكة الإنترنت. فمثلاً إذا أردت فى هذه الأيام اقتناء أحد برامج تحرير النصوص كبرنامج Word مثلاً، تقوم بشراء البرنامج على قرص مدمج ثم تثبته على حاسبك من خلال هذا القرص المدمج. أما مع استخدام .NET فالأمر مختلف، حيث يمكن إتاحة محرر النصوص كخدمة مشتركة من خلال الإنترنت ومن ثمَّ يمكنك تثبيته على حاسبك أو يمكنك بطريقة أخرى تشغيل المحرر من خلال الإنترنت، على أن تقوم بدفع مقابل يتناسب مع الإمكانات التى تستخدمها فقط.

ولعلك الآن تتساءل، أليس هذا متاحاً داخل الشبكات الصغيرة حيث يمكنك استخدام البرامج المثبتة على الخادم كما لو كانت موجودة على حاسبك المتصل بالشبكة ؟. وكلامك صحيح مائة فى المائة، إلا أن الويب يختلف اختلافاً كبيراً عن الشبكات الصغيرة. فكل شبكة يكون لها نظام تشغيل معين ونموذج مكونات معين وخصائص عديدة مشتركة إلى جانب عدد معين من الحاسبات المتصلة بهذه الشبكة. أما شبكة الإنترنت أو الويب فتتكون من أعداد هائلة من الحاسبات ذات أنظمة التشغيل والخصائص المختلفة. لذا فإن أخذ مبدأ التجانس فى الحسبان سيزيد المشكلة تعقيداً. فإذا وُجدت الخدمة على أحد الأجهزة البعيدة، فإن

الجهاز الذى يقوم باستخدام هذه الخدمة لا يعرف عنها أو عن خصائصها شيئاً، وإنما يتم التأكد من أن كلا الحاسبين يفهم معنى البيانات المرسلة من أحدهما إلى الآخر وهذا يتم من خلال لغة عالمية موحدة تمكن من الاتصال السهل والسريع بين المرسل والمستقبل وهذا يتحقق حقيقةً من خلال مبدأين غاية فى الأهمية هما لغة الترميز الممتدة Extensible Markup Language (XML) والبروتوكول Simple ObjectAccessProtocol (SOAP).

Visual Basic وبيئة .NET Frame Work.

لقد قامت شركة Microsoft بإنشاء بيئة عمل .NET Frame Work.لتسهيل تطوير التطبيقات المعتمدةعلى نظام Windows. ولكن نظراً للفروق الموجودة بين الإصدارين 6.0 و Visual Basic.NET من لغة Visual Basic (حيث تعتبر Visual Basic .NET بمثابة أول إصدار .NET.)، فقد وجد معظم مطورى Visual Basic أن التطوير أصبح أكثر صعوبة . فعلى سبيل المثال، قام الإصدار Visual Basic.NET بتحويل جميع المتغيرات إلى كائنات، الأمر الذى قضى على قدرة المبرمج على تعريف نوع متغير بشكل مباشر.

لكن تطوير التطبيقات داخل بيئة عمل .NET Framework. ليس أكثر صعوبة مما كان عليه فى الإصدار 6.0 من Visual Basic ومن الممكن أن تكون بيئة عمل .NET Framework و Visual Basic 2012من الأدوات الفعالة، وهذا الأمر لا يتطلب سوى اكتشاف كيفية عملها معاً من خلال Integrated DevelopmentEnvironment(IDE)الخاصة بـ Visual Studio.

المزيد عن بيئة عمل Net Framework.

تعد بيئة عمل .NET Framework. كمفهوم بمثابة مجموعة من البرامج المتصلة ببعضها البعض والتي تم تطويرها من قبل شركة Microsoft والتي تعمل على الربط بين الأشخاص والنظم والأجهزة التي يستخدمونها بالمعلومات التي يحتاجون إليها. وتعد بيئة عمل .NET.

Framework هي بيئة التطوير التي تقوم بكل ما سبق من خلال Visual Basic. وتعد Visual Basic جزء من بيئة .NET Framework.

تتضمن التطبيقات المصممة بصورة جيدة الطبقات التالية:

- تتضمن الأجهزة التابعة في بيئة .NET. أجهزة مثل التليفونات المحمولة وأجهزة PDA وأجهزة الكمبيوتر الشخصية التي تعمل بنظام Windows أو برنامج تصفح الويب على أي نظام تشغيل.
- تعتمد وحدات الخدمة Serversفي بيئة .NET. على Windows Server أو SQL Server. ويعد نظام تشغيل وحدة الخدمة أقل مرونة من نظام تشغيل الجهاز التابع في عالم .NET. وأحيانًا يتم استخدام وحدات خدمة مثل BizTalk أو SharePoint Services. بوجه عام، توفر وحدات الخدمة خدمات متعددة.
- يوجد بين وحدات الخدمة والأجهزة التابعة خدمات ويب XML أو غيرها من الأدوات الخاصة بالاتصال. وتعد خدمات ويب XML أداة يمكنها العمل على نظم مختلفة لنقل معلومات من وحدات خدمة إلى أجهزة تابعة أو من أجهزة تابعة إلى أجهزة أخرى أو حتى بين الخدمات بعضها البعض.

أما عن أدوات المطور، فتتمثل في Visual Basic و Visual Studio. وتعد Visual Basic هي اللغة، في حين تعد Visual Studio هي الأداة. ويملك المطور أدوات أخرى مثل C++ أو C#أو F#أو Java script.

تعتبر الكيفية التي تتفاعل بها Visual Basic مع أجزاء Visual Studioأمرًا مهمًا للغاية أيضًا؛ فهذا يشكل ماهية عمل البرامج. فالبرنامج سيستخدم الخدمات الموفرة من قبل بيئة عمل .NET Framework. عن طريق الأدوات الخاصة باللغة. ويعد أمر التفاعل هذا أساسيًا لكل شئ . وهو الأمر الذي تحتاج إلى التركيز عليه في تخطيطك.

تمثل المصطلحات التى تواجه الكثير من المبرمجين عند الانتقال إلى بيئة عمل

NETFramework.بعض الصعوبات والارتباك. لتجنب الوقوع فى الارتباك الناشئ عن هذه المصطلحات نورد فيما يلى أهم المصطلحات المستخدمة فى التطوير باستخدام بيئة عمل:

- ***Visual Basic 2012***: ونقصد بها لغة البرمجة التى هى موضوع هذا الكتاب، حيث لم يعد بإمكانك تشغيل لغة Visual Basic أو تحميلها ككيان منفصل كما كان يحدث فى Visual Basic 6 أو الإصدارات التى قبله. وهى تعتبر إحدى لغات البرمجة التى تعمل تحت بيئة NET Frame Work..
- ***NET Framework.***: عبارة عن الطبقة الموجودة بين اللغة (لغة Visual Basicفى هذه الحالة) ونظام التشغيل (إصدار Windows 98 أو /ME2000 أوXP أو 2003Server أو Windows 7 أو Windows 8 أو أى إصدارٍ آخر). وتعمل طبقة NET. Framework على توفير الإمكانيات الوظيفية المطلوبة اعتماداً على أسلوب عمل نظام Windows الذى تعمل عليه،وكذلك مجموعات الملفات الخاصة بإمكانيات وظيفية أخرى (مثل الوصول إلى قاعدة البيانات والعمليات الحسابية).
- ***Visual Studio2012***: عبارة عن الأداة التى يمكنك استخدامها لإنشاء أى نوع من التطبيقات باستخدام أى لغة برمجية متوافقة. حلت بيئة تطوير Visual Studio محل Visual Basic6.0 والذى كان يعتبر فيما سبق جزءاً من مجموعة برامج Visual Studio (والتى كان يحمل كل مكون بها رقم الإصدار 6.0).

عندما تريد كتابة برنامج جديد فى بيئةNET. ، يجب أن تقوم أولاً بتشغيل بيئة تطوير Visual Studio 2012 ثم تقوم بتحديد نوع البرنامج الذى تريد كتابته باللغة البرمجية التى ترغب فى استخدامها. فمثلا لإنشاء برنامج Visual Formsيفضل استخدام لغة Visual Basic ، ولكتابة تطبيق لـ Smart Device يفضل استخدام لغة # C.

- *تطبيق* ***Windows Forms*** : هو المصطلح الجديد لتطبيق Visual Basic القياسى. ويشير هذاالمصطلح الى تطبيق تمت كتابته باستخدام بيئة عمل NET. Framework وله واجهة استخدام Windows.

- *تطبيق* ***Web Forms***: مصطلح يطلق على تطبيق له واجهة صفحة ويب مكتوبة باستخدام بيئةعمل **.NET Framework**. ويشبه إنشاء تطبيق **Web Forms** إنشاء تطبيق **Windows Forms**.
- *خدمات الويب* : عبارة عن مجموعات ملفات الفئات المكتوبة باستخدام معيار متعارف عليه من قبل الأشخاص أنفسهم الذين قاموا بتعريف المعايير بالنسبة لشبكة الويب . ويتم استخدام خدمات الويب للتفاعل بين النظم المختلفة.

خلاصة القول،من السهل على بيئة عمل **.NET From work**. التعامل مع كل الجوانب الخاصة بنظام **Windows**. فتوفر **.NET Framework**. على وجه التحديد – اسماً برمجياً لكل كائن وكل حدث يمكن لـ **Windows** التحكم فيه . ويمكن أن يستخدم المبرمج ذلك الاسم للإشارة الى أى شيء له كود فى نظام التشغيل.

نموذج .NET Framework. التنفيذى

حينما ترغب فى إنشاء أحد التطبيقات باستخدام بيئة تطوير **Visual Studio2012**، يتم إجراء الخطوات الآتية:

١. يتم أولاً كتابة الكود باستخدام اللغات المتعددة التى يدعمها **.NET**.

٢. بعد ذلك يتم ترجمة هذا الكود من خلال مترجم **.NET**. الذى يقوم بتحويل الكود المصدرى للُّغة المستخدمة إلى تنسيق جديد يسمى تنسيق اللغة الوسيطة **Intermediate Language (IL)** أو لغة مايكروسوفت الوسيطة **Microsoft Intermediate Language (MSIL)** حيث يطلق على الملف الناتج من عملية التحويل والذى يحتوى على تنسيق اللغة الوسيطة "الملف التنفيذى المحمول" **Portable Executable (PE)**.

٣. لتنفيذ التطبيق، يتم تحويل تنسيق اللغة الوسيطة إلى كود أصلى**Native Code** وذلك داخل الحاسب المستخدم لتشغيل التطبيق باستخدام مترجم يسمى **Just-in-Time (JIT)**.

ولعلك تلاحظ أننا لم نقم باستخدام المفسرات **Interpreters** أثناء الدورة بالكامل وإنما يتم ترجمة جميع الكود داخل بيئة التشغيل المشتركة **Common Language Runtime (CLR)**التى تحتوى على مجموعة الملفات الأساسية المكونة لتطبيقات **NET.**، وهذا يؤدى بالطبع إلى تشغيل التطبيقات بكفاءة أعلى لأن المفسرات تتسبب فى بطء التنفيذ.

اللغة الوسيطة IL

اللغة الوسيطة **Intermediate Language (IL)** عبارة عن لغة آلة قامت مايكروسوفت بإنشائها وتحتوى على مجموعة من التعليمات والأوامر التى لا تعتمد على نوع المعالج المستخدم والتى يتم تحويلها إلى كود أصلى**Native Code**. وهذه التعليمات تحتوى على ما يلى:

- العمليات الحسابية والمنطقية
- عبارات التحكم
- الوصول المباشر للذاكرة
- احتواء الاستثناءات (الأخطاء)

كما أن اللغة الوسيطة تمتاز بالميزات التالية:

- يمكنك تشغيل اللغة الوسيطة على أى معالج.
- يمكنك تشغيل اللغة الوسيطة على أى نظام تشغيل.
- تحتوى اللغة الوسيطة على المزيد من الأمان فى الوصول إلى الكود.
- تتيح اللغة الوسيطة تشغيل أكثر من تطبيق داخل نفس مساحة الذاكرة.

ترجمة اللغة الوسيطة إلى الكود الأصلى

لا يمكنك تنفيذ كود اللغة الوسيطة إلا بعد تحويله إلى كود أصلى **Native Code** وذلك كما يلى:

١. يتم إرفاق كود يسمى **Stub code** مع كل دالة داخل نوع التصنيف الذى يجرى تحميله.

٢. حينما يتم استدعاء إحدى الدوال، يقوم الكود المصاحب للدالة بتوجيه تنفيذ البرنامج إلى المترجم JIT الذى يقوم بالتحويل من اللغة الوسيطة إلى الكود الأصلى
٣. يتم استبدال الكود المصاحب للدالة بعنوان الدالة داخل الكود الأصلى.
٤. إذا تم استدعاء نفس الدالة مرةً أخرى، يتم تنفيذ الكود الأصلى من الذاكرة مباشرةً دون الحاجة إلى استخدام المترجم JIT مرةً أخرى.

وكما ترى من اسم المترجم Just-in-Time (JIT) فإن عملية الترجمة تتم فقط وقت الحاجة ولا يوجد مفسر Interpreter داخل العملية.

تنفيذ التطبيق

بمجرد الانتهاء من تحويل كود اللغة الوسيطة إلى الكود الأصلى، يتم مباشرةً تنفيذ التطبيق وهنا تظهر عملية تجميع نفايات الذاكرة أو ما يسمى Garbage Collection، حيث يقوم البرنامج أثناء تنفيذه باستخدام عدد من الموارد مثل سجلات قاعدة البيانات ووصلات الشبكة وهكذا، ويكون التحكم فى هذه الموارد داخل البرنامج أمرٌ بالغ التعقيد، لذا يتم التحكم فى هذه الموارد باتباع الخطوات الآتية:

١. يتم حجز جزء من الذاكرة بقدر النوع الذى يمثل المورد.
٢. يتم استهلال المورد، أى تشغيل حالته الابتدائية وتهيئته للاستخدام.
٣. يتم الوصول إلى حالة من تصنيف المورد أو بمعنى آخر استخدامه لأداء الوظيفة المنوطة به.
٤. يتم تحديث حالة المورد دلالةً على إمكانية تدميره لعدم الحاجة إليه بعد الانتهاء من استخدامه.
٥. يتم تحرير جزء الذاكرة الذى يستخدمه المورد.

إذا لم يتم تحرير الذاكرة المستخدمة من قِبَل المورد أو تمت محاولة الوصول إلى المورد بعد تحرير ذاكرته، فى كلتا الحالتين يصبح البرنامج غير ثابت ويتصرف تصرفات غريبة. وهنا يأتى دور مجمع النفايات Garbage Collector الذى يعد أحد العناصر الأساسية داخل CLR

والذى يقوم نيابةً عن المبرمج بالتحكم فى تحرير الذاكرة والتعامل مع الموارد. ولكن لا يستطيع مجمع النفايات التعرف على وقت تنفيذ الخطوة الرابعة فى الخطوات السابقة. فعلى سبيل المثال إذا أراد المبرمج إنهاء الاتصال بالشبكة، فأنَّى لمجمع النفايات أن يعرف ذلك ؟. حقيقةً يجب أن يوضح المبرمج ذلك من خلال دالة الهدم **Destructor** الخاصة بحالة التصنيف أو من خلال الدالة **()Close** مثلاً بحيث يعرف المجمع فقط رغبة المبرمج فى إنهاء استخدام المورد.