1. تمهيد

لا شك أن هدف أي مبرمج هو بلوغ درجة الاحتراف و بناء تطبيقات قوية و متينة من جميع النواحي، فالاعتناء بمظهر واجهة البرنامج و جعلها جذابة و سهلة الاستعمال للمستخدم أمر حيوي كما أن ملاحقة الأخطاء البرمجية (Debugging) و جعل التطبيق متين و خالي تماما من الأخطاء لا يقل أهمية عن الأول، لكن هل فكرت بالجانب الخاص بمصادر النظام و خاصة سرعة التنفيذ؟

إن من بين الأمور الأساسية التي تجعل البرنامج ينال رضا المستعمل هو سرعة التنفيذ، لذلك عليك الحرص على ذلك باختيار أسرع طرق التنفيذ من بين الخيارات المتوفرة لمعالجة مقطع من الكود. لكن كيف السبيل إلى المقارنة بين سرعات التنفيذ لمختلف الحلول المتوفرة؟

هذا السؤال سنجيب عليه إن شاء الله بين أسطر هذا المقال. لذلك شدوا الأحزمة سننطلق...

2. لمحة عن الفئة الجديدة `Stopwatch`

في سعيها الدءوب لتحسين و تطوير بيئة ال .NET، تضع لنا Microsoft ال Framework .NET 2.0 الذي يحتوي في مجمله على تحسينات و فئات جديدة جعلت منه غنيا مقارنة بسابقيه بحيث أصبح المبرمج يمتلك مجموعة جديدة كليا من الفئات و الأدوات التي جعلت من البرمجة متعة حقيقية.

من بين التحسينات التي أضافتها Microsoft على ال Framework .Net مجموعة جديدة من الفئات و من بينها الفئة `Stopwatch` التابعة لمجال الأسماء `System.Diagnostics`

رغم بساطتها إلا أنها ذات فائدة كبيرة للمبرمج فبواسطتها يمكنك احتساب وقت تنفيذ مقطع من الشيفرة بطريقة متناهية الدقة (دقة تصل إلى 10^{-7} جزء من الثانية) و من خلال 3 أسطر فقط!

2.أ. طريقة عمل الفئة

الفئة تمتلك مجموعة من الخصائص و الطرق تمكننا من حساب وقت التنفيذ وذلك بإتباع النهج التالي:

1 تهيئة الفئة باستعمال الطريقة `Reset()`.
2 إعطاء الأمر ببدء احتساب الوقت باستعمال الطريقة `Start()` قبل بداية مقطع الكود المعني مباشرة.
3 إعطاء الأمر بتوقيف عملية الاحتساب باستعمال الطريقة `Stop()` بعد مقطع الكود.
4 معرفة الوقت المستنفذ من الخاصية `Elapsed` أو `ElapsedMilliseconds` أو `ElapsedTicks`.

2.ب. الطرق(Methods)

أحببت أن أختصر و أعرج على أهم الطرق التي نحتاجها.

الطريقة	الشرح
`Reset()`	تهيئة نسخة الفئة الحالية لعملية جديدة بحذف قيمة الوقت المحتسب سابقا (إعطاء القيمة 0 (أو -null- Nothing) للخاصية `ElapsedXXX`) .
`Start()`	إعطاء الأمر ببداية الاحتساب. في حالة عدم تهيئة نسخة الفئة من قبل فإن عملية الاحتساب ستكون تراكمية أي أن القيمة الجديدة ستضاف إلى القيمة السابقة المخزنة في الخاصية `ElapsedXXX`.
`Stop()`	توقيف عملية الاحتساب و جعل الخاصية `IsRunning` تحمل القيمة `false`.

الجدول 1.2: بعض طرق الفئة `Stopwatch`.

ملاحظة: `ElapsedXXX` تشير إلى الخصائص ال `Elapsed` أو `ElapsedTicks` أو `ElapsedMilliseconds`