**ملحق 1**

**أهمّ الاستثناءات Exceptions**

في هذا الفصل الملحق، سنشرح أهمّ أنواع الاستثناءات، وسنكتفي بأهمّها، وهي بعض تلك الموجودة في فضاء الاسم System.. ويمكنك أن تراجع شرح موضوع معالجة الاستثناء في [الفصل الثاني](file:///C:\\Users\\Administrator\\01-%20%D8%A3%D8%B3%D8%A7%D8%B3%D9%8A%D9%91%D8%A7%D8%AA%20%D8%A7%D9%84%D9%84%D8%BA%D8%A9\\02-%20%D8%A7%D9%84%D9%85%D8%B4%D8%A7%D8%B1%D9%8A%D8%B9\\2-%20%D8%A5%D9%86%D8%B4%D8%A7%D8%A1%20%D8%A2%D9%84%D8%A9%20%D8%AD%D8%A7%D8%B3%D8%A8%D8%A9\\5-%20%D9%85%D8%B9%D8%A7%D9%84%D8%AC%D8%A9%20%D8%A7%D9%84%D8%A7%D8%B3%D8%AA%D8%AB%D9%86%D8%A7%D8%A1%D8%A7%D8%AA.htm).

**خليّة الاستثناء Exception Class**

**استخدام خليّة الاستثناء Exception Class:**

ترث كلّ الاستثناءات خليّة الاستثناءات الرئيسيّة Exception Class.. بمعنى أنّ الاستثناء من النوع Exception ينطلق لو حدث أيّ خطإ في تطبيقك مهما كان نوعه.

إذن فبم تفيدنا كلّ هذه الاستثناءات التي سنشرحها ها هنا؟.. أسمعك تتساءل!

حسنا.. يمكنك اعتبارها عمليّة تنظيميّة، حتّى تحصل على درجة أكبر من الدقّة عند معالجة الخطإ.. انظر للمثال التالي:

**Try**

**…………….**

**' ويمكنك استخدام الجملة التالية لمغادرة مقطع معالجة الاستثناء**

**Exit Try**

**Catch Exc As DivideByZeroException**

**…………….**

**Catch Exc As ArithmeticException**

**…………….**

**Catch Exc As SystemException**

**…………….**

**Catch When أي شرط يناسبك**

**…………….**

**Catch Exc As Exception**

**…………….**

**Finally**

**…………….**

**End Try**

لاحظ أنّ هذا الكود يحاول معالجة الاستثناء الناتج عن القسمة على صفر DivideByZeroException.. إنّ هذا الاستثناء مشتقّ من الخليّة ArithmeticException، التي تمثّل جميع الاستثناءات الناتجة عن العمليّات الحسابيّة..

فإذا لم يحدث استثناء القسمة على صفر، وحدث استثناء حسابيّ آخر، فستتمّ معالجته في مقطع الاستثناء ArithmeticException.

فإذا لم يحدث استثناء حسابيّ، وحدث استثناء في النظام، فإنّ الكود سيستخدم الاستثناء SystemException، (ومنه قد اشتقّت الاستثناءات الحسابيّة).

ويمكنك استخدام مقطع Catch When، لمعالجة الاستثناءات بشروط معينة.. وأخيرا، لو كان هناك أيّ استثناء لا يندرج تحت الاستثناءات السابقة، فستتمّ معالجته تحت الاستثناء العامّ Exception، الذي تندرج تحته كلّ أنواع الاستثناءات.

كما ترى: إنّها مجرّد عمليّة تنظيميّة، للتحكّم في ردّ الفعل المتّخذ عند حدوث أنواع بعينها من الاستثناءات.. فمثلا، يبدو أنّ المبرمج مهتمّ في الكود السابق، باتّخاذ خطوات خاصّة عند حدوث استثناء القسمة على صفر.

هذا مع ملاحظة، أنّ بإمكانك استخدام جملة When في أيّ جملة Catch من الجمل السابقة.. مثل:

**Catch Exc As Exception When أي شرط يناسبك**

كما يمكنك أن تكتب كلمة Catch منفردة كالتالي:

**Try**

**Catch Exc As DivideByZeroException**

**Catch**

**End Try**

**ملحوظة هامّة:**

لا ريب أنّك ما زلت لم تفهم وظيفة المقطع Finally جيّدا، وتتساءل: ولماذا لا أكتب الكود بعد نهاية مقطع الاستثناء كلّه، بدلا من كتابته في جملة Finally؟

هذا يعود في الواقع للكيفيّة التي يعمل بها معالج الاستثناء..

إنّ بإمكانك أن تكتب مقاطع لمعالجة الاستثناءات، داخل مقطع Try.. وبإمكانك من داخل مقطع Try أيضا، استدعاء إجراءات أخرى، بها مقاطع لمعالجة الاستثناءات.. بدأت تفهم إذن معنى كلمة هيكليّ Structured التي نصف بها معالجة الاستثناء؟

بناء على هذا، لو حدث استثناء، ولم يوجد مقطع Catch يحتوي على نوع الخطإ المناسب، فإنّ التنفيذ ينتقل لمقطع معالجة الخطأ الأعلى (الذي يحتوي على المقطع الحاليّ)، أو يغادر الإجراء الحاليّ إلى الإجراء الذي استدعاه (إذا كان هذا الاستدعاء قد حدث في جملة Try).. ثمّ إلى الإجراء الذي استدعاه.. وهكذا، حتّى يعثر على جملة Catch المناسبة.. (لن يحدث هذا طبعا في حالة استخدام جملة Catch العامة:

**Catch اسم\_الاستثناء As Exception**

فهي تتعامل مع أي نوع من الاستثناءات.

ممّا سبق يتّضح لك ما يلي:

1- معنى أنّ معالجة الخطأ هيكليّة Structured، حيث يتمّ تمرير الاستثناء عبر الإجراءات المختلفة من الداخل للخارج ـ إلى أن يتمّ العثور على المعالج المناسب للاستثناء.. وبهذا تمتلك تحكّما أكبر في معالجة الخطإ وفي أيّ إجراء تتمّ، بحيث يمكنك ـ لو أردت ـ ضمّ كلّ جمل المعالجة في موضع واحد.

2- أنّ جملة Finally حيويّة للغاية، فهي تلزم VB بتنفيذ بعض الكود قبل أن يغادر الإجراء.. وأنت تفهم الآن كيف يمكن مغادرة الإجراء إجباريّا في حالة حدوث استثناء غير معالج.. لهذا فمن المهمّ جدّا أن تستخدم مقطع Finally لإغلاق الملفات التي فتحتها وإنهاء أيّ عمل قد بدأته، حتّى لا تحدث نتائج غير متوقعة.

 والآن تعال نرى أهمّ خصائص ووسائل خليّة الاستثناء، التي ترثها كلّ أنواع الاستثناءات الأخرى.

**خصائص خليّة الاستثناء Exception Class:**

**ملفّ المساعدة HelpLink:**

تقرأ أو تغيّر ملفّ المساعدة الذي يشرح الاستثناء.

**الرسالة Message:**

نصّ يشرح الاستثناء الذي حدث وأسبابه.

**مصدر الاستثناء Source:**

تعيد اسم التطبيق أو الكائن الذي تسبّب في حدوث الاستثناء.

**موقع الهدف TargetSite:**

تعيد اسم الدالة التي تسبّبت في حدوث الاستثناء.

**الاستثناء الداخليّ InnerException:**

تعيد الاستثناء الذي تسبّب في إطلاق الاستثناء الحالي.. يحدث أحيانا أن ينطلق استثناء داخل إحدى الوسائل، فتقوم بتمريره إلى مستدعيها.

**مسار الرّصة StackTrace:**

تعيد نصّا، يمثّل مسار الوسائل التي تمّ استدعاؤها من بعضها، وصولا لتلك التي حدث بها الاستثناء.

**وسائل خليّة الاستثناء Exception Class:**

**اقرإ الاستثناء الأصليّ GetBaseException:**

تعيد الاستثناء الأصليّ، الذي تسبّب في انطلاق سلسلة من الاستثناءات.

**إلى نصّ ToString:**

تعيد نصّا يمثّل الاستثناء.

**بعض أهمّ الاستثناءات**

**الاستثناءات الخاصّة بالنظام:**

 تمثّل الخليّة SystemException الخليّة الرئيسيّة لكلّ أنواع الاستثناءات الموجودة في فضاء الاسم System.. وينطلق الاستثناء SystemException عند حدوث أيّ خطإ في النظام، ممّا سنفصّله في المقاطع التالية.

**الاستثناءات الخاصّة بالتطبيقات والبرامج:**

ترث هذه الاستثناءات الاستثناء SystemException.. وهي تضمّ:

**AppDomainUnloadedException:**

يحدث عند محاولة التعامل مع تطبيق غير محمّل UnLoaded (تمّ انتهاؤه، أو لم يبدأ بعد).

**CannotUnloadAppDomainException:**

يحدث عندما تفشل محاولة إغلاق unload أحد التطبيقات application domain.

**BadImageFormatException**:

يحدث عند محاولة تشغيل ملفّ تنفيذي executable file غير صالح للتشغيل.

**ContextMarshalException**:

يحدث عندما تفشل محاولة التعامل مع كائن خارج حدود العمليّة الحاليّة.

**ExecutionEngineException**:

يحدث عندما يحدث خطأ داخليّ في محرّك التنفيذ الخاصّ باللغة CLR's Execution engine.

**InvalidProgramException**:

يحدث هذا الاستثناء إذا كان هناك خطأ في مترجم الكود compiler، ممّا يؤدّى إلى ترجمة البرنامج بطريقة خاطئة.. بخلاف ذلك، لا يمكن أن تتسبّب أنت في حدوث هذا الاستثناء.

**NotSupportedException**:

يحدث عندما تحاول أداء وظيفة معيّنة غير متاحة بواسطة البرنامج.

**UnauthorizedAccessException**:

يحدث عند محاولة التعامل بدون تصريح مع عنصر محميّ من عناصر النظام.

**ApplicationException**:

يمثّل الخليّة الرئيسيّة Base class، للاستثناءات التي تحدث في التطبيق.

**الاستثناءات الخاصّة بفشل التحميل:**

**TypeLoadException**:

هذه هي الخليّة الرئيسيّة للاستثناءات التي تحدث عند فشل تحميل كائن معيّن.

**DllNotFoundException**:

يحدث عندما لا يتمّ العثور على ملفّ .dll معيّن.

**EntryPointNotFoundException**:

يحدث عندما يفشل استدعاء إحدى الدوال من ملفّ  .dll معيّن.

**الاستثناءات الخاصّة بالذاكرة:**

ترث هذه الاستثناءات الاستثناء SystemException.. وهي تضمّ:

**OutOfMemoryException**:

يحدث عندما يعجز التطبيق عن حجز المساحة المطلوبة في الذاكرة.

**StackOverflowException**:

يحدث عندما يستنفد البرنامج كلّ مساحة الرصّة stack المخصّصة له ـ في الغالب بسبب إجراء ارتدادي Recursive Procedure غير مكتوب جيّدا.

**الاستثناءات الخاصّة بالتعامل مع الكائنات:**

ترث هذه الاستثناءات الاستثناء SystemException.. وهي تضمّ:

**InvalidCastException**:

يحدث حينما تحاول إجراء عمليّة تحويل غير ممكنة بين نوعين من الكائنات Invalid conversion of types.

**NullReferenceException**:

يحدث عندما تحاول أن تتعامل مع وسيلة غير مشتركة Non\_shared، بينما الكائن يشير إلى القيمة Nothing، بمعنى أنّه لا يشير إلى أيّ نسخة Instance من الخليّة.

**TypeUnloadedException**:

يحدث عند محاولة التعامل مع خليّة غير محمّلة Unloaded class.

**TypeInitializationException**:

يحدث عند حدوث خطئ في حدث إنشاء الكائن New.

**InvalidOperationException**:

هذه هي الخليّة الرئيسيّة للاستثناء ObjectDisposedException.

**ObjectDisposedException**:

يحدث عند محاولة استخدام كائن تمّ التخلّص منه Disposed.

**الاستثناءات الخاصّة بالدوالّ**

**الاستثناءات الخاصّة باستدعاء الوسائل المحميّة Protected:**

**MemberAccessException**:

هذه هي الخليّة الرئيسيّة، لكلّ الاستثناءات الخاصّة بفشل استدعاء حقول ووسائل وخصائص الكائنات.

**FieldAccessException**:

يحدث عندما تحاول التعامل مع حقل (متغيّر) field خاصّ Private أو محميّ protected في إحدى الخلايا Classes، من خارج النطاق Domain المسموح لك بالتعامل مع المتغيّر من خلاله.

**MethodAccessException**:

يحدث عندما تحاول التعامل مع وسيلة Method خاصّة private أو محميّة protected في إحدى الخلايا، من خارج النطاق Domain المسموح لك بالتعامل معها من خلاله.

**الاستثناءات الخاصّة باستدعاء الوسائل غير الموجودة:**

**MissingMemberException**:

هذه هي الخليّة الرئيسيّة للاستثناءات التي تحدث عندما تحاول استدعاء عضو Member غير موجود في كائن معيّن.

**MissingFieldException**:

يحدث عندما تحاول أن تتعامل مع حقل (متغيّر) غير موجود في الكائن.

**MissingMethodException**:

يحدث عندما تحاول التعامل مع وسيلة غير موجودة في الكائن.

**الاستثناءات الخاصّة بمعاملات الدوالّ:**

ترث هذه الاستثناءات الاستثناء SystemException.. وهي تضمّ:

**FormatException**:

يحدث عندما يكون معامل الدالة غير منسّق بطريقة صحيحة incorrectly formatted.

**ArgumentException**:

يمثّل الخليّة الرئيسيّة للاستثناءات التالية:

**ArgumentNullException**:

يحدث عند تمرير القيمة Nothing كمعامل لدالة تنتظر مرجعا صالحا لكائن Valid object reference.

**ArgumentOutOfRangeException**:

يحدث عند إرسال قيمة خارج المدى المسموح به لمعاملات الدالّة.

**DuplicateWaitObjectException**:

يحدث عندما يظهر كائن أكثر من مرّة في مصفوفة الكائنات المتزامنة Synchronization objects ([راجع الفصل رقم 14](file:///03-%20%D9%85%D9%88%D8%A7%D8%B6%D9%8A%D8%B9%20%D9%85%D8%AA%D9%82%D8%AF%D9%91%D9%85%D8%A9/14-%20%D8%A7%D9%84%D9%85%D9%87%D8%A7%D9%85%D9%91%20%D8%A7%D9%84%D9%85%D8%AA%D8%B9%D8%AF%D9%91%D8%AF%D8%A9/13-%20%D8%A5%D9%8A%D9%82%D8%A7%D9%81%20%D8%A7%D9%84%D8%B9%D9%85%D9%84%D9%8A%D9%91%D8%A9%20%D8%A7%D9%86%D8%AA%D8%B8%D8%A7%D8%B1%D8%A7%20%D9%84%D8%AD%D8%AF%D8%AB%20%D9%85%D8%A7.htm)).

**الاستثناءات الخاصّة بالمصفوفات:**

ترث هذه الاستثناءات الاستثناء SystemException.. وهي تضمّ:

**ArrayTypeMismatchException**:

يحدث عندما تحاول وضع قيمة من نوع غير مناسب في إحدى خانات المصفوفة.

**IndexOutOfRangeException**:

يحدث عندما تحاول التعامل مع خانة غير موجودة في المصفوفة.

**RankException**:

يحدث عندما تحاول تمرير مصفوفة كمعامل لدالة، بينما عدد أبعاد المصفوفة غير مناسب لمعامل الدالة.

**الاستثناءات الخاصّة بالعمليّات الحسابيّة:**

ترث هذه الاستثناءات الاستثناء SystemException.. وهي تضمّ:

**ArithmeticException**:

يمثّل الخليّة الرئيسيّة للاستثناءات الحسابيّة، حيث يحدث هذا الاستثناء عند حدوث أيّ خطإ حسابيّ.. وترث باقي الاستثناءات الحسابيّة هذه الخليّة.

**DivideByZeroException**:

يحدث عندما تقسم عددًا صحيحا Integer على صفر.. لاحظ أنّ قسمة الأعداد العشريّة على صفر لا ترمي هذا الاستثناء، بل ينتج عنها القيمة "ما لا نهاية".

**NotFiniteNumberException**:

يحدث عندما تحاول استخدام رقم غير صالح ( ما لا نهاية الموجبة PositiveInfinity، ما لا نهاية السالبة NegativeInfinity، ليس رقما NaN) في عمليّة حسابيّة تتطلّب رقما مقبولا.

**OverflowException**:

يحدث عندما يتجاوز ناتج عمليّة حسابيّة معيّنة، الحدود الرقميّة المسموح بها.