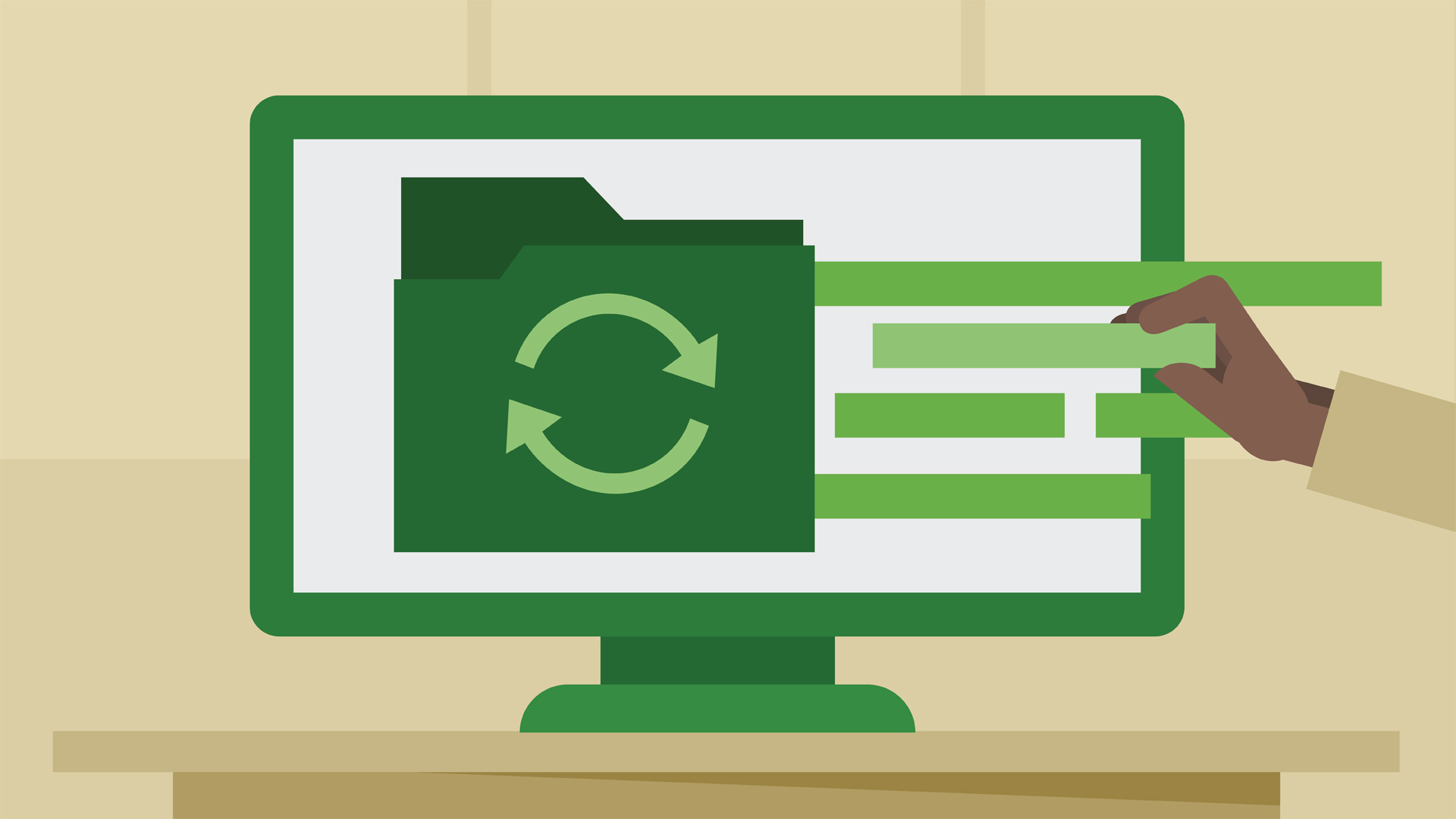
Manejo de memoria





Programación 3 visual

**Manejo de memoria**

**Introducción**

Uno de los principales problemas a la hora de desarrollar software cuando se utiliza múltiples procesos o threads es el manejo de memoria, ya que si no se utiliza correctamente puede causar problemas de rendimiento en el programa que se esté desarrollando también consumir muchos recursos del dispositivo en que se ejecuta y puede causar que la aplicación deje de funcionar o se congele.

Pero al mismo tiempo tener un buen manejo de memoria en una aplicación puede ayudar a consumir menos recursos del dispositivo y al mismo tiempo ayudar a optimizar procesos. En este ensayo veremos cómo manejar la memoria de manera óptima utilizando las distintas herramientas que .NET nos brinda.

**Garbage collector**

El garbage collector es un gestor de memoria que libera la memoria que no se utiliza de manera automática, esto permite hacer optimizar el uso de los recursos del dispositivo, esto sin que el usuario de cuenta.

La manera en la que recolecta la basura no lo hace todo completo sin no de manera parcial o también llamada generaciones las cuales son tres: la generación 0 se encuentran los objetos de vida corta como variables locales, si dicha variable sigue usándose entre objetos pasará a generación 1, y cuando ya no se use se libera la memoria, en caso de que dichas variables continua en el main de la aplicación pasa a generación 3 ya que en este punto se encuentran los datos y variables de larga vida y cuando ya no se utilicen se libera la memoria. Todo esto ocurre en el garbage collector en background.

Cabe destacar que en el garbage collector se puede utilizar en hilos/thread, también con objetos estáticos, objetos de caché y variable globales.

Aunque también es posible que manipular algunos parámetros del garbage collector aunque no es muy recomendable, pero es posible hacer uso de buenas prácticas para aumentar su eficiencia.

**Manipulación del garbage collector**

Aunque no sea muy recomendable su manipulación, si es posible, pero esto solamente es conveniente realizarlo cuando sabemos a ciencia cierta que un conjunto de recursos costosos que se han estado utilizando no se van a utilizar más y por tanto deberían ser liberados. El método a utilizar para forzar al GC a liberar recursos es System.GC.Collect().

La clase System.GC es la que implementa el recolector de basura. Algunos métodos útiles son:

* GC.MaxGeneration(), que devuelve el numeral de la máxima generación en el GC soportada por el montículo.
* GC.WaitForPendingFinalizers(): Este método suspende el hilo hasta que se procesen los finalizadores pendientes.
* GC.GetGeneration(): Que permite manipular las distintas generaciones del GC

A pesar de que la gestión de la memoria es un proceso bastante automático en .NET, es muy conveniente entender cómo funciona entre bastidores porque sí que hay ciertas prácticas que permiten sacar provecho de su funcionamiento. Por ejemplo:

* Limitar el uso de objetos estáticos, ya que constituyen raíces que no se pueden recolectar.
* Utilizar la localidad de referencia en la medida de lo posible, es decir: declarar los objetos lo más cerca posible de su primer uso. Esto evitará la creación de objetos mucho antes de su uso.
* Tratar de concentrar las operaciones sobre un objeto en una porción pequeña de código, pues el GC se basa en la premisa de que el uso de un objeto está concentrado en unas pocas líneas de código y después no se vuelve a utilizar (nuevamente, localidad de referencia).
* Usar convenientemente Close() y Dispose() para liberar recursos como conexiones, buffers, streams, etc. que ya no se van a utilizar.

**Conclusión**

En conclusión puedo decir que a pesar de que .net sea uno de los pocos entornos que contienen contienen garbage collector, se puede optimizar el uso de la memoria también y a pesar de que se puede manipular el garbage collector lo considero algo arriesgado.

Pero algo que podemos hacer, no solo en el entorno de .net sino en todos los entornos y es hacer un buen uso de sobre el manejo de memoria como las variables para que no se acumule el cache uso de funciones para no repetir código o los otros consejos que mencione.

**Referencias**

*youtube (7 abr. 2017) Exploring NET's Memory Management — A Trip Down Memory Lane. Obtenido de youtube:*

[*https://www.youtube.com/watch?v=9FEfy9y0fFQ*](https://www.youtube.com/watch?v=9FEfy9y0fFQ)