

Tehnici Avansate de Programare

Gestionarea resurselor și ecosistemul NuGet

Petru Rebeja, Marius Apetrii

11 Martie 2021

Facultatea de Matematică

Universitatea Alexandru Ioan Cuza, Iași

Introdurre

Recapitulare

- Clase abstracte vs. interfețe.
- Acuplare și Coeziune.
- Fluxul de lucru GitHub

Despre ce vom discuta azi

- Garbage collector \neq IDisposable
- Ecosistemul NuGet

Gestiunea resurselor în .net

Platforma .net separă resursele în două categorii:

1. Resurse gestionate automat (*managed resources*)
2. Resurse gestionate manual (*unmanaged resources*)

Resurse gestionate automat

- Resursele gestionate automat sunt resursele care se află sub controlul platformei `.net`.
- Pot fi dealocate automat de Garbage Collector.

Resursele gestionate manual sunt resursele din afara platformei dar cu care avem nevoie să interacționăm.

Exemple:

- Ferestre,
- Fișiere,
- Conexiuni la baza de date
- Etc.

Resursele utilizate de aplicații trebuie gestionate eficient.

De ce contează asta?

- Gestiunea resurselor are impact direct asupra aplicațiilor.
- O gestiune necorespunzătoare poate avea efecte precum:
 - Costuri ridicate de infrastructură,
 - Costuri ridicate de întreținere,
 - Timp de răspuns întârziat => satisfacția scăzută a utilizatorilor.

Resursele unei aplicații .net sunt gestionate astfel:

- Garbage Collector se ocupă de curățarea memoriei alocată resurselor gestionate automat,
- Șablonul de eliminare (*dispose pattern*).

Garbage Collector

Garbage Collector¹ (GC)

- Este responsabil de gestiunea eficientă a memoriei.
- Elimină necesitatea de a elibera manual memoria ocupată de resursele aplicației.
- Oferă siguranța la acces (un obiect nu poate citi conținutul altuia).

¹<https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals>

Generații de obiecte¹

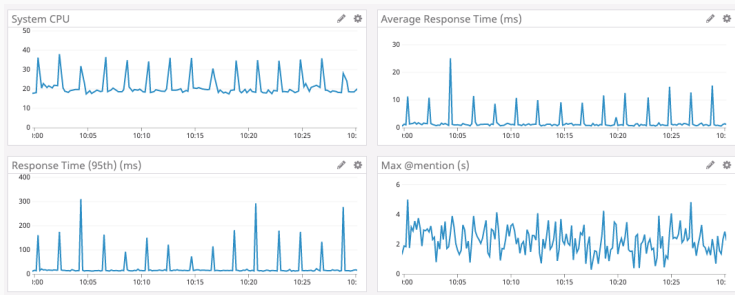
Memoria heap este organizată în 3 generații:

1. **Generația 0** — conține obiecte cu termen de viață foarte scurt. Marea majoritate a obiectelor trebuie să fie în generația 0.
2. **Generația 1** — conține obiecte cu termen de viață scurt și/sau de dimensiuni mari. Este o zonă de trecere de la generația 0 la generația 2.
3. **Generația 2** — conține obiecte cu termen de viață lung. Ex: date statice ale aplicației.

- La crearea unui obiect nou, acesta este pus în Generația 0.
- La următoarea execuție a GC se verifică:
 - Dacă obiectul mai are referințe active, atunci este mutat în Generația 1,
 - Altfel, memoria alocată obiectului este eliberată.
- Procesul se repetă pentru fiecare generație în parte.

- Când GC începe verificarea memoriei, toate firele de execuție a aplicației sunt suspendate.
- Excepție: firul care declanșează GC nu este suspendat.

Efectul execuției GC asupra procesorului și impactul asupra timpului de răspuns al aplicației Discord².



²<https://blog.discordapp.com/why-discord-is-switching-from-go-to-rust-a190bbca2b1f>

Dispose pattern

- Este folosit pentru a dealoca memoria resurselor gestionate automat.
- Este o implementare a metodei `System.IDisposable.Dispose()`

³<https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/implementing-dispose>

IDisposable

```
public interface IDisposable
{
    void Dispose();
}
```

Dispose pattern³

```
public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}
```

Dispose pattern³

```
protected virtual void Dispose(bool disposing)
{
    if(this.disposed)
        return;

    if(disposing)
    {
        // Dispose managed resources
    }

    // Dispose unmanaged resources
    this.disposed = true;
}
```

Dispose pattern³

```
~DisposableResource()  
{  
    Dispose(false);  
}
```

Cum utilizăm obiecte IDisposable

Cum utilizăm obiecte IDisposable

- Obiectele care implementează IDisposable trebuie dealocate prin apelarea metodei `Dispose()`.
- Apelarea se poate face în două moduri:
 - Cu instrucțiunea `using` sau,
 - Într-un block `try/finally`.

Blocul try/finally

```
DisposableResource res = null;
try
{
    res = new DisposableResource();
    // Do something with the resource
}
finally
{
    if(res != null) res.Dispose();
}
```

- Apelează automat metoda `Dispose()` la părăsirea contextului curent.
- Este echivalentă cu blocul `try/finally`.

```
using(var res = new DisposableResource())  
{  
    // Do something with the resource  
}
```

Ecosistemul NuGet

Ce este NuGet?

- Aplicație utilitară pentru integrarea bibliotecilor oferite de terți în soluție/proiect.
- Facilitează instalarea pachetelor și a pachetelor de care acestea depind.
- Facilitează ștergerea și actualizarea pachetelor.

Ce este un pachet NuGet?

- O arhivă cu extensia schimbată din `.zip` în `.nupkg`.
- Conține bibliotecile necesare (fișierele `.dll`) și metadate.

- Cele publice se găsesc pe <https://www.nuget.org/>.
- Pentru pachetele private sunt necesare servere dedicate.

Înceiere

- Resurse gestionate automat vs resurse gestionate manual,
- Organizarea în 3 generații,
- Gestionarea ineficientă a resurselor are impact negativ asupra:
 - experienței utilizatorilor,
 - bugetului.

Este indicat să folosim NuGet pentru adăugarea de biblioteci în soluție/proiect.

Mulțumesc pentru atenție!