Rozhraní IDisposable a blok using

V prostředí .NET se o řízenou paměť stará Garbage Collector. Nicméně u vnějších zdrojů, jako jsou soubory nebo síťová připojení, musí programátor zajistit jejich uvolnění manuálně. K tomu slouží standardizovaný vzor pomocí rozhraní **IDisposable**.

1. Co je IDisposable?

Jedná se o rozhraní obsahující jedinou metodu: `Dispose()`. Pokud třída toto rozhraní implementuje, signalizuje tím, že drží zdroje, které by měly být po ukončení práce explicitně uvolněny.

```
public class MujZapisovac : IDisposable {
    public void Dispose() {
        // Logika pro uvolnění neřízených zdrojů (např. zavření souboru)
    }
}
```

—

2. Problém bez uvolnění

Pokud zapomenete zavolat `Dispose()`, zdroj zůstane v operačním systému „otevřený“ nebo „zamčený“.

- **Příklad:** Otevřete soubor pro zápis. Pokud ho neuzavřete, žádná jiná aplikace (ani váš program při dalším spuštění) k němu nemusí získat přístup, dokud proces aplikace úplně neskončí.

—

3. Konstrukce using

Zajištění, aby se `Dispose()` zavolal i v případě, že v kódu dojde k chybě (výjimce), je pomocí bloků `try-finally` pracné. Jazyk C# proto nabízí klíčové slovo **using**, které tento proces automatizuje.

Klasický blok using

Zaručuje, že metoda `Dispose()` bude zavolána ihned poté, co program opustí složené závorky.

```
using (var soubor = new StreamWriter("denik.txt")) {
    soubor.WriteLine("Ahoj světe");
} // Zde se automaticky volá soubor.Dispose()
```

Using deklarace (C# 8.0+)

Moderní a stručnější zápis. Objekt se uvolní automaticky na konci bloku (např. na konci metody), ve kterém byl deklarován.

```
public void ZapiseData() {
    using var soubor = new StreamWriter("denik.txt");
    soubor.WriteLine("Nová data");
} // soubor.Dispose() se zavolá automaticky zde
```

—

4. Vztah ke Garbage Collectoru

GC neví, že váš objekt drží zámek na soubor. On vidí pouze malý objekt v paměti.

- Pokud zavoláte `Dispose()`, uvolníte neřízený zdroj **okamžitě**.
- Pokud na to zapomenete, zdroj se uvolní až tehdy, kdy GC objekt úplně odstraní z paměti (což může trvat velmi dlouho).

—

5. Kdy IDisposable implementovat?

Vlastní třídu byste měli implementovat jako `IDisposable`, pokud:

```
1. Přímo využívá neřízené zdroje (práce s Win32 API, ukazatele).
2. Obsahuje jako své členy (fields) jiné objekty, které jsou samy o sobě
`IDisposable` (např. `SqlConnection`, `HttpClient`, `FileStream`).
```

Související články:

- Garbage Collector
- Stack vs. Heap (Zásobník vs. Halda)
- Zpracování výjimek v .NET

Tagy: programming dot-net csharp memory-management resources idisposable