


Správa paměti v prostředí .NET

Martin Dybal

Microsoft MSP

<http://sense4code.com/>

Rozdělení dat v paměti

- Instrukční paměť
 - Stack
 - By default 1MB
 - LIFO
 - Heap
 - Rozdělen na generace
 - Alokace pomocí new
 - Spravován Garbage Collectorem
- 

Datové typy

Value types

- bool
- byte
- char
- decimal
- double
- enum
- float
- int
- long
- sbyte
- short
- struct
- uint
- ulong
- ushort

Reference types

- Array
- class
- interface
- delegat
- string

Datové typy

- Mýty
 - Hodnotové typy jsou na stacku a referenční na Haldě
 - Přiřazení hodnoty se liší u referenčních a hodnotových typů
- Struktura
 - Hodnotový datový typ
 - Jsou menší, nemají takovou režii jako referenční typy
- Demo

Heap

- Code heap
- Small object heap
 - Klasický heap
- Large object heap
 - Objekty větší než 85KB
- Process heap

Garbage collector(GC)

- John McCarthy 1959
- Vyhledává již nepotřebné objekty
- Process CLR



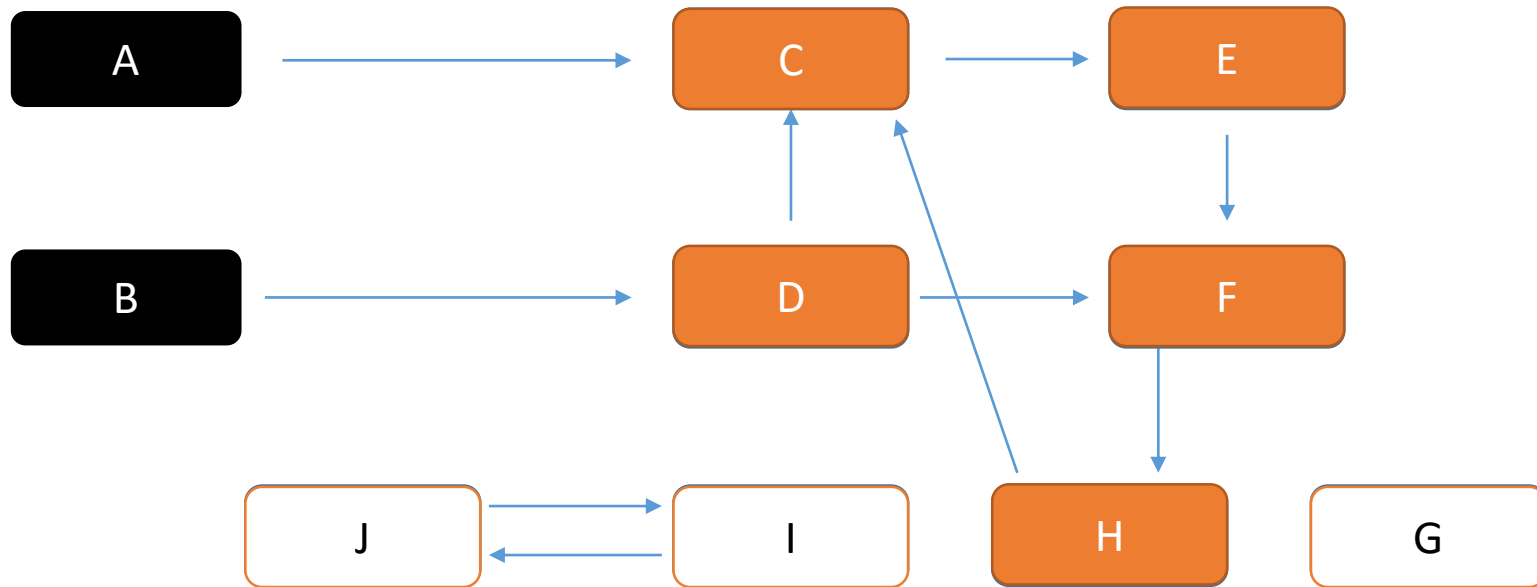
Reference counting GC

- Výhody
 - Dokáže objekty rychleji čistit
 - Nepozastavuje běh programu, vhodné pro RT systémy
- Nevýhody
 - Vysoká režie
 - Alokace, přiřazení, dealokaci
 - Paměťová zátěž
 - Musí si pamatovat počet reference na každý objekt
 - Neumí řešit cyklickou referenci objektů

Tracing GC

- Výhody
 - Pokud nedochází ke collectingu, tak nijak neomezuje běh
 - Dokáže vyřešit cyklickou referenci objektů
 - Částečně řeší fragmentaci heapu
- Nevýhody
 - Musí pozastavit běh programu
 - Odklizení je nedeterministické

Garbage collecting



~~Mark~~
Mark

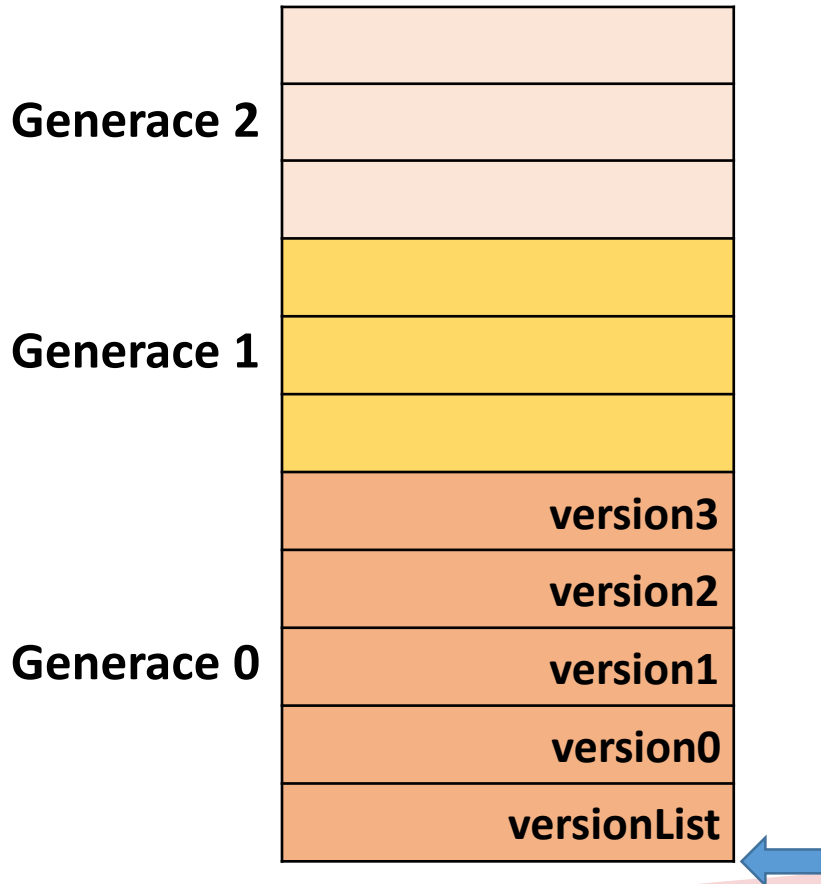
GC Roots

- Lokální proměnné
 - GCHandles
 - Pin
 - Static
 - Registry
 - F-Reachable queue
- 

Generace objektů

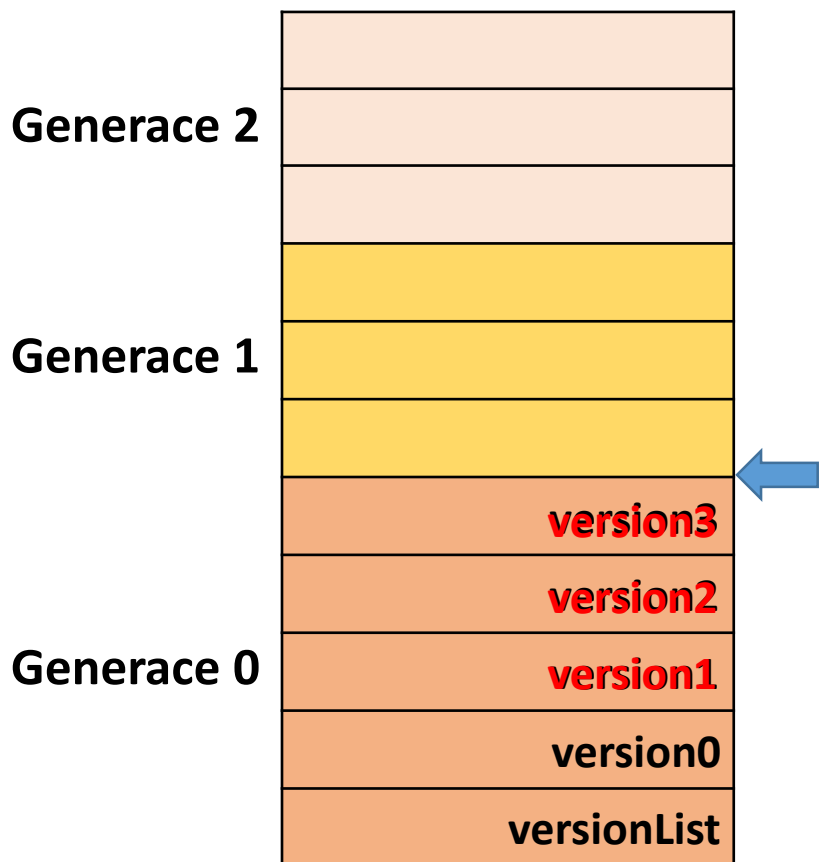
- Generace 0
- Generace 1
- Generace 2

Generace objektů



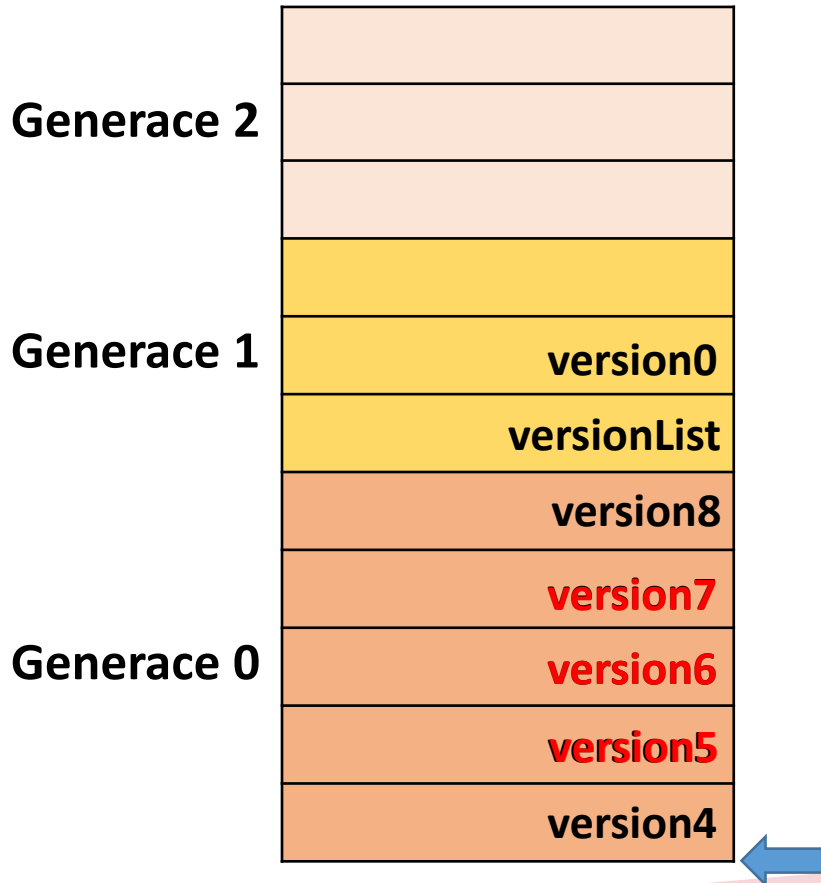
```
private static List<Version> versionList;  
0 references  
private static void Main()  
{  
    versionList = new List<Version>();  
    for (int i = 0; i < 100; i++)  
    {  
        var version = new Version();  
        if (i % 4 == 0)  
        {  
            versionList.Add(version);  
        }  
    }  
}
```

Generace objektů



```
private static List<Version> versionList;  
0 references  
private static void Main()  
{  
    versionList = new List<Version>();  
    for (int i = 0; i < 100; i++)  
    {  
        var version = new Version();  
        if (i % 4 == 0)  
        {  
            versionList.Add(version);  
        }  
    }  
}
```

Generace objektů




```
private static List<Version> versionList;  
0 references  
private static void Main()  
{  
    versionList = new List<Version>();  
    for (int i = 0; i < 100; i++)  
    {  
        var version = new Version();  
        if (i % 4 == 0)  
        {  
            versionList.Add(version);  
        }  
    }  
}
```


Spuštění garbage collecting

- Zaplněna generace 0
- System wide memory pressure
- ~~GC.Collect()~~


Finalization

- Finalize queue
 - F-reachable queue
 - ResourceWrapper, IDispoze
- 

.NET Tips

- `String.Empty`
 - Dispose Pattern
 - `WeakReference`
 - Large object heap compacting
- 

Dispose Pattern

- Není přesně dáno kdy se objekt odklidí
 - IDisposable
 - Destruktor
 - Using
 - CA2000 , CA1816
 - GC.SuppressFinalize(this)
- 

Dispose Pattern

```
public class DisposableClass : IDisposable
{
    private bool disposed = false;

    0 references
    public void Dispose() // NOT virtual
    {
        Dispose(true);
        GC.SuppressFinalize(this); // Prevent finalizer from running.
    }

    2 references
    protected virtual void Dispose(bool disposing)
    {
        if (!disposed)
        {
            disposed = true;
            if (disposing)
            {
                // Call Dispose() on other objects owned by this instance.
                // You can reference other finalizable objects here.
                Console.WriteLine(" Disposing everything including other finalizable objects");
            }
            // Release unmanaged resources owned by (just) this object.
            Console.WriteLine(" Disposing internals of this object");
        }
    }

    0 references
    ~DisposableClass()
    {
        Dispose(false);
    }
}
```

String.Empty

- Je efektivnější než ""

WeakReference



Memory profiling

- Proč je object v paměti?
- Co zabírá tolik paměti?
- Memory leak



Memory Leak

- Chyba správy paměti
- Objekt uložený v paměti nepřístupný z programu



Memory Optimize Memory Traffic

- Příliš mnoho alokací
- Časté spouštění Garbage collectingu




Large object heap compacting

- Od .Net 4.5.1
- Velmi časově náročné

```
GCSettings.LargeObjectHeapCompactionMode =  
    GCLargeObjectHeapCompactionMode.CompactOnce;  
GC.Collect();
```

Verze GC v .NET

- Mobile
 - Workstation
 - Server
- 

Background GC

- Workstation .NET 4.0
- Server .NET 4.5
- Lze vypnout
 - `<gcConcurrent enabled="False">`

SustainedLowLatency

- `GCSettings.LatencyMode = GCLatencyMode.SustainedLowLatency;`

Large object support

- Možnost vytvářet objekty větší než 2GB
 - Pouze 64bitové aplikace
 - `<gcAllowVeryLargeObjects enabled="true"/>`
- 