STUDY4
為 學 習 而 生

Build School

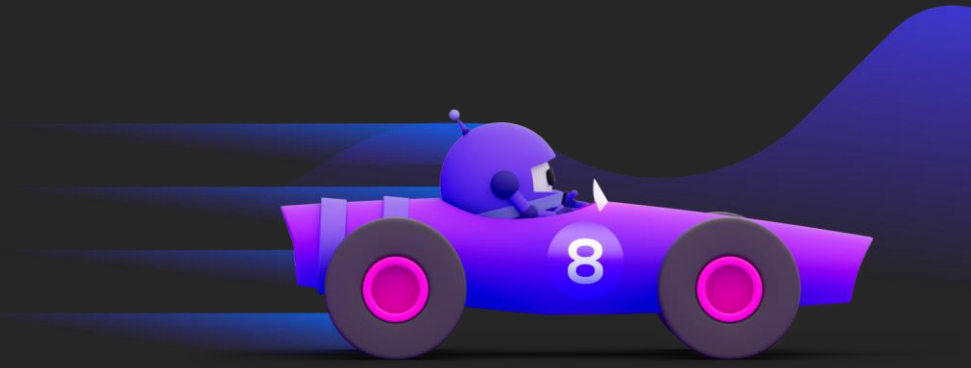
.NET Conf
TAIWAN

STUDY4
為 學 習 而 生

Build School

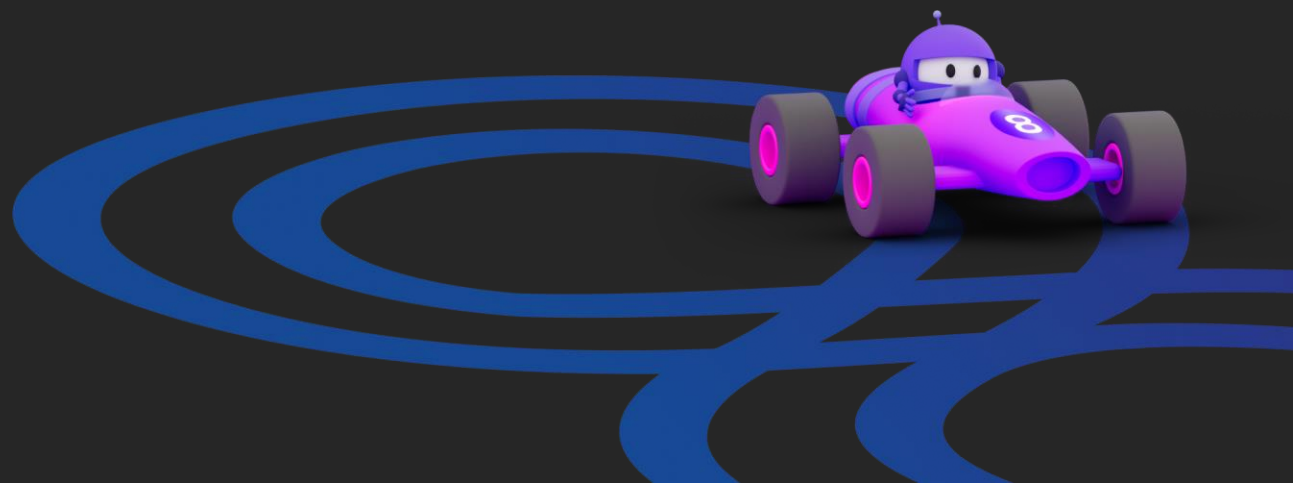
每年改一版 到底煩不煩

Bill Chung



去年的C#

已經配不上今年的你

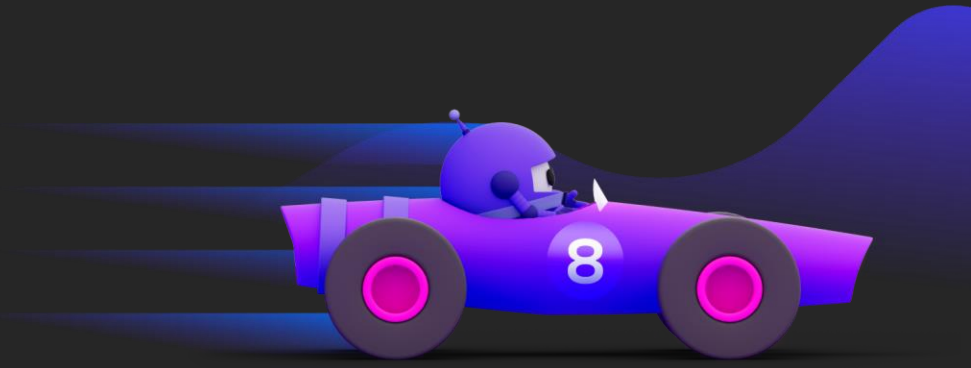


深有所感

- value tuple
- patterns matching
- generic math
- ……

about me

- Bill Chung
- 海角點部落
- https://github.com/billchungiii
- https://skilltree.my/
- https://www.build-school.com/

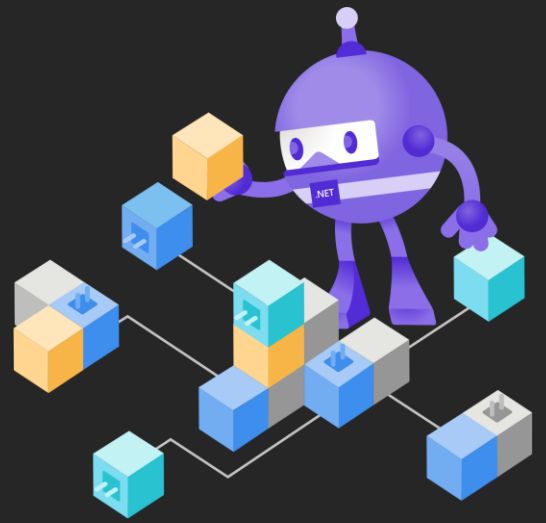


C# 13

- params collection
- all new lock object
- ref struct
- ref & unsafe
- semi-auto property
- indexer
- partial property
- escape sequence
- linq

params collections

- `params` 修飾詞的擴展
- 過去只能使用一維陣列



擴展的型別

- span type
  - Span<T>, ReadOnlySpan<T>
- implement interfaces
  - IEnumerable<T>, ICollection<T>, IList<T>
  - IReadOnlyCollection<T>, IReadOnlyList<T>
- 具有可存取的執行個體Add方法
- 具有可存取的執行個體無參數建構式

params 類型擴展

```
public static void Display<T>(params IEnumerable<T> args)
{
    Console.WriteLine($"params IEnumerable<T>");
    foreach (var arg in args)
    {
        Console.WriteLine(arg);
    }
}

public static void Display<T>(params Span<T> args)
{
    Console.WriteLine($"params Span<T>");
    foreach (var arg in args)
    {
        Console.WriteLine(arg);
    }
}
```

多載解析

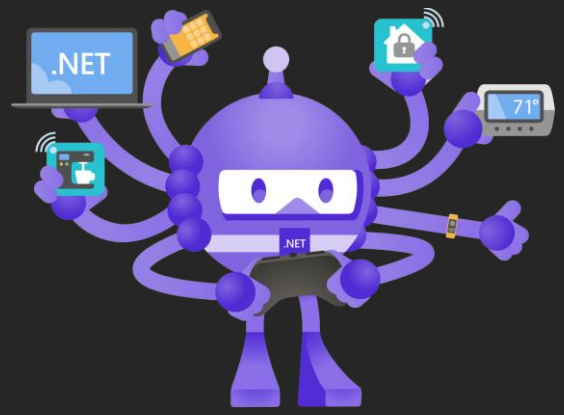
- 多載解析可能會不如預期或衝突
- 尤其在泛型參數未明確化狀況
- 使用 `OverloadResolutionPriorityAttribute (.NET 9)`

OverloadResolutionPriority

```
[OverloadResolutionPriority(1)]
public static void Display<T>(params ReadOnlySpan<T> args)
{
    Console.WriteLine($"params ReadOnlySpan<T>");
    foreach (var arg in args)
    {
        Console.WriteLine(arg);
    }
}
```

all new lock object

- `System.Threading.Lock class`
- 避免 `Sync block index` 同時擔任多種角色導致效能損失



使用實踐

- 執行緒可能產生例外的狀況下要結束鎖定
- 需要 `time out` 的情境

過去的 lock

```
var obj = new object();
lock (obj)
{
    // do something
}
```

現在你可以...

```
var locker = new Lock();

lock (locker)
{   }

locker.Enter();
try
{   }
finally
{
    locker.Exit();
}

using (locker.EnterScope())
{   }
```

逾時處理

```
if (locker.TryEnter(TimeSpan.FromSeconds(1)))
{
    try
    {
        // do something
    }
    finally
    {
        locker.Exit();
    }
}
else
{
    // 處理逾時
}
```

ref struct -- 介面與泛型約束

- 允許 `ref struct` 實作介面
- `allows ref struct`
- 現有泛型介面與類別支援 `allows ref struct`
  - `IEnumerable<T>`
  - `Action` 家族
  - `Func` 家族

實作介面

```
public ref struct RefStruct : IComparable<RefStruct>, IEquatable<RefStruct>
{
    // .. 略
    public int CompareTo(RefStruct other)
    {
        return _number.CompareTo(other._number);
    }

    public bool Equals(RefStruct other)
    {
        return _number.Equals(other._number);
    }
}
```

泛型約束

```
public static bool IsBigger<T>(this T source , T other) where T : IComparable<T>,
                                                                 allows ref struct
{
    return source.CompareTo(other) > 0;
}
```

ref & unsafe

- 允許在非同步/迭代器中使用 `ref` 或 `unsafe`

非同步

```
async static Task Main(string[] args)
{
    await Task.Delay(100);
    ref int source = ref GetValue();
    source *= 2;
    Console.WriteLine(numbers[0]);
}
```

迭代器

```
static IEnumerable<int> GetNumbers()
{
    ref int x = ref numbers[0];
    yield return 1;
    yield return 2;
    yield return 3;
}
```

semi-auto property

- 使用 `field` 關鍵字
- 目前為 `preview`

field 關鍵字

```
public int Age
{
    get;
    set
    {
        if (field != value)
        {
            field = value;
            OnPropertyChanged(nameof(Age));
        }
    }
}
```

indexer

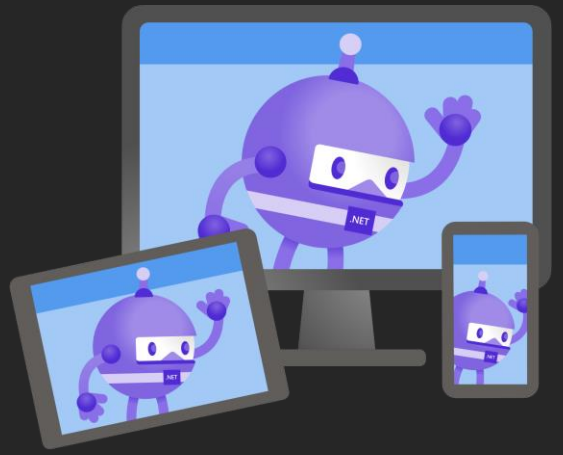
- 可以在初始化設定式使用 ^ 運算子

物件初始化使用 ^

```
var obj = new MyClass()
{
    Number =
    {
        [0] = 99,
        [^1] = 100
    }
};
```

partial property

- 分離宣告和實作，便於管理大型程式碼
- 支援 `source generator`



分離宣告與實作

```
public partial class Person
{
    // 宣告與定義
    public partial string Name { get; set; }
}

public partial class Person
{
    // 實作
    private string _name;
    public partial string Name
    {
        get => _name;
        set =>_name = value;
    }
}
```

escape sequence

- 使用 `\e` 替代 `\u001b`
- 主要是為了支援更簡易的終端機輸出

ESC 的過去與未來

```
// 以前你要這樣寫
Console.WriteLine("\u001b[1;33mThis is a bold yellow text\u001b[0m");

// 現在你可以這樣寫
Console.WriteLine("\e[1;33mThis is a bold yellow text\e[0m");
```

linq

- CountBy method
- AggregateBy method
- Index method

CountBy

```
var oldCountByCity = persons.GroupBy(p => p.City)
                            .Select(g => new { City = g.Key, Count = g.Count() });

var newCountByCity = persons.CountBy(p => p.City);
```

AggregateBy

```
var oldAggregateByCity = persons.GroupBy(p => p.City)
                                .Select(g => new { City = g.Key, Age = g.Sum(p => p.Age) });

var newAggregateByCity = persons.AggregateBy(p => p.City, 0, (acc, p) => acc + p.Age);
```

Index

```
var oldIndex = persons.Select((p, index) => (p, index));

var newIndex = persons.Index();
```

STUDY4 為學習而生 Build School

謝謝各位
願 C# 與你同在

範例程式碼

https://github.com/billchungiii/CS13_NewFeatures