

C#13に入らなかった機能

2024/12/09 C# Tokyo

@tanaka_733

自己紹介 @tanaka_733

- New Relic株式会社
テクニカルサポート 部部長
- **C# Tokyo**運営メンバー
- Microsoft MVP for Developer Technologies
- 最近の趣味
 - 草ソフトボール
 - ゴルフ
 - ピアノ

今日のテーマ

- C# 13が.NET 9とあわせてリリースされました
- C# 13では採用されなかった（GAされなかった）機能をのぞいてみましょう
- 「採用されなかった」の基準が難しいので、LDM NotesやIssues、++C++; の情報を参考にいくつかピックアップしています
- と思って調べたいたら、C# 13でどこまで実装されたのか結局よくわからん！？という状態になってしまいました...

今日のテーマ

- fieldプロパティ
- ファーストクラスSpan
- Extensions Member
- null条件アクセスにおけるincrement/decrement演算子

fieldプロパティ

- プロパティ宣言でバックリングフィールドをfieldとして参照
- ステータス
 - C# 13でpreviewとして実装
 - [Issue 仕様](#)
 - [ドキュメント](#)

```
cs 📄 コピー  
  
{ get => field; set => field = value; }
```

```
cs 📄 コピー  
  
{ get => field; set => throw new InvalidOperationException(); }
```

```
cs 📄 コピー  
  
{ get => overriddenValue; set => field = value; }
```

```
cs 📄 コピー  
  
{  
    get;  
    set  
    {  
        if (field == value) return;  
        field = value;  
        OnXyzChanged();  
    }  
}
```

式本体のプロパティと `get` アクセサのみを持つプロパティでは、以下も使用できます `field`。

```
cs 📄 コピー  
  
public string LazilyComputed => field ??= Compute();
```

```
cs 📄 コピー  
  
public string LazilyComputed { get => field ??= Compute(); }
```

ファーストクラスSpan

- C# 7.2でSpan<T>、ReadOnlySpan<T>が導入
 - コンパイラ的には単なる構造体
 - Span<T>、ReadOnlySpan<T>、T[]をまとめて扱いたい
- ステータス
 - [提案されている仕様](#)
 - 一部導入済み？
 - コレクション式でSpanへの解決を優先 (C# 12)
 - C# 13でのいくつかの機能はSpanを特別扱いしたわけではないがSpanの使い勝手が向上している (次のページで)
 - **拡張メソッドのレシーバーの解決 (未実装?)**

ファーストクラスSpan (C# 13)

- params にコレクション式で表現できるものが指定可能

```
static void M3(params ReadOnlySpan<char> a)
{ }
```

- イテレーター・asyncメソッド内でrefとunsafeが指定可能
- ジェネリック型制約にallows ref structが指定可能に
 - 標準のAction、Funcにも追加

```
void M4(Func<ReadOnlySpan<char>, char> selector)
{ }
```

ファーストクラスSpan (未実装?)

- こういう拡張メソッドを使えるようにしたい

```
ReadOnlySpan<int> intSpan = [1,2,3,4];  
//これはOK  
intSpan.StartsWith('a');
```



```
int[] intArray = [1,2,3,4];  
//コンパイラエラー  
intArray.StartsWith('a');
```

0 個の参照

```
public static class MemoryExtensions
```

```
{
```

2 個の参照

```
    public static bool StartsWith<T>(this ReadOnlySpan<T> span, T value) where T : IEquatable<T>  
    {  
        return true;  
    }  
}
```


Extensions Members

- メソッド以外のメンバーも拡張できるように
- スタータス
 - 未実装だが最近活発的に議論中
 - [LDM Notes 提案された仕様](#)

```
public static class Enumerable
{
    extension(IEnumerable source) // extension members for IEnumerable
    {
        public bool IsEmpty { get { ... } }
    }
    extension<TSource>(IEnumerable<TSource> source) // extension members for IEnumerable<TSource>
    {
        public IEnumerable<T> Where(Func<TSource, bool> predicate) { ... }
        public IEnumerable<TResult> Select<TResult>(Func<TSource, TResult> selector) { ... }
    }
    extension<TElement>(IEnumerable<TElement>) // static extension members for IEnumerable<TElement>
    where TElement : INumber<TElement>
    {
        public static IEnumerable<TElement> operator +(IEnumerable<TElement> first, IEnumerable<TElement> second) {
        }
    }
}
```

null条件アクセスにおけるincrement/decrement演算子

- [LDM Note](#) にて実装しない方針
- 右のサンプルで「議論中」と記載のnull条件アクセスは議論中だがあまり動きがなさそう

```
class C
{
    5 個の参照
    int F;
    0 個の参照
    void M(C? c)
    {
        c?.F++;
        c?.F--;
        ++c?.F;
        --c?.F;
        //これは議論中
        c?.F += 1;
    }
}
```