Span<T>, Memory<T>

Petr Novák / novakpe@fel.cvut.cz / 2021-04-27

Obsah

1 Úvod

Span<T> nevytváří nový objekt / pole, tedy nealokuje žádnou paměť. Jde v podstatě pouze o vytvoření jakoby virtuálního okna na část již existující paměti / pole.

2 Použití Span<T> a ReadOnlySpan<T>

Nejprve vytvoříme pole, které je vytvořeno skutečně v paměti, tedy zabírá určitou část paměti:

```
byte[] array = new byte[] { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 };
```

Pokud je potřeba pracovat pouze s částí tohoto pole a nevytvářen nové lze použít následující:

```
Span<byte> part = new Span<byte>(array, 2, 5);
```

Vytvoří se pouze okno do původního pole a to od druhého prvku do pátého prvku. Nové pole „part“ bude obsahovat prvky 6, 8, 10, 12, 14. První (index „0“) prvek pole „part“ je třetí (index „2“) prvek pole „array“ obsahující hodnotu „6“. „part“ tedy nevytváří žádný nový objekt / pole, ale v podstatě pouze ukazatel na část původního pole.
Prvky v původním poli „array“ lze číst a měnit zcela podle potřeby i přes nové pole „part“. Například:

```
„array[3] = 123” je zcela stejné jako “part[1] = 123“
```

Oboje změní hodnotu „8“ na hodnotu „123“.
Pod-část pole z již existujícího Span<T> lze vytvořit pomocí Span.Slice(…):

```
// z pole „part“ vytvoří nové pod-pole, pouze s prvky „8, 10“
Span<byte> first = part.Slice(1, 2);
// z pole „part“ vytvoří nové pod-pole, začínající třetího prvku, tedy „10, 12, 14“
Span<byte> second = part.Slice(2);
```

V některých případech je vhodné použít `ReadOnlySpan<T>`:

```
// vytvoří pod-pole určené pouze pro čtení, nelze modifikovat
ReadOnlySpan<byte> test = new Span<byte>(array, 2, 5);
```

Toto může být vhodné v několika případech:

- Chránit vytvořenou pod-část podle proti změně
- Pro objekty jako „String“ které nelze měnit (modifikací se vytváří nový)

3 Použití Memory<T> a ReadOnlyMemory<T>

Jde o obdobu Span<T>

4 Další možnosti

Span<T> obsahuje některé další užitečné metody:

Clear() – Vymazaní celého pole
Fill() – Vyplní prvky pole požadovanou hodnotou
BinarySearch() – Hledání hodnoty „int pos = `part.BinarySearch((byte)3);`“

Poznámky:

- Span<T> a ReadOnlySpan<T> nelze použít pro „await“ a „yield“.
- Memory<T> a ReadOnlyMemory<T> lze použít pro „await“ a „yield“.
- Span<T> a ReadOnlySpan<T> jsou umístěny na zásobníku (nikoli na spravované haldě) a proto jsou pouze „structura“, nemůže být zabaleno do „objekt“.
- Memory<T> a ReadOnlyMemory<T> jsou umístěny na spravované haldě (nikoli na zásobníku) a jsou tedy „class“, mohou být zabaleny do „objekt“.