Что нового в net10 libraries: strings, collections, options validation

Забиякин М.В (team lead mkb.ru)

String: normalization api

- Normalizations
- String normalization

Символ	é	é
Unicode	U+00E9	U+0065 + U+0301
Длина	1	2

- NFC, NFD, NFKC, NFKD

Issue

- ReadOnlySpan<char>
- Normalize ReadOnlySpan<char>
- ReadOnlySpan<char> надо приводить к string

  issue - open Jun 19, 2023
  [API Proposal]: `ReadOnlySpan<char>` overloads for `StringNormalizationExtensions` · Issue #87757 · dotnet/runtime

Что предложили:

API Proposal

```
namespace System;

// Doesn't throw on the operation even when having wrong character sequence.

public static class StringNormalizationExtensions {
+    public static bool IsNormalized(this ReadOnlySpan<char> source, NormalizationForm normalizationForm = NormalizationFor
+    public static OperationStatus TryNormalize(this ReadOnlySpan<char> source, Span<char> destination, out int charsWritte
}
```

Original Proposal - Alternative Proposal

```
namespace System;

// throws on wrong sequence as String.Normalize does.

public static class StringNormalizationExtensions {
+    public static bool IsNormalized(this ReadOnlySpan<char> strInput, NormalizationForm normalizationForm = NormalizationF
+    public static bool TryNormalize(this ReadOnlySpan<char> strInput, Span<char> destination, out int charsWritten, Normal
}
```

Итог:

- Pr - merged Dec 10, 2024
  Normalization APIs using the spans by tarekgh · Pull Request #110465 · dotnet/runtime · GitHub
- Добавили GetNormalizedLength

```
1  namespace System;
2
3  public static partial class StringNormalizationExtensions
4  {
5      public static bool IsNormalized(this ReadOnlySpan<char> source, NormalizationForm normalizationForm = NormalizationForm.FormC);
6      public static bool TryNormalize(this ReadOnlySpan<char> source, Span<char> destination, out int charsWritten,
7          NormalizationForm normalizationForm = NormalizationForm.FormC);
8      public static int GetNormalizedLength(this ReadOnlySpan<char> source, NormalizationForm normalizationForm = NormalizationForm.FormC);
9  }
```

String: convert hex-string(utf8)

- Utf-8
- Hex-string (#FF0000, 0x7FF4, MD5)

```
1   ReadOnlySpan<byte> hexUtf8 = "48656C6C6F2C20576F726C64"u8; // "Hello, World" в hex
2
3   // -> лишняя аллокация: создаём string из utf8-байтов
4   string hexString = Encoding.UTF8.GetString(hexUtf8);
5
6   // используем существующее строковое API (или собственный парсер)
7   byte[] data = Convert.FromHexString(hexString);
8
9   Console.WriteLine(Encoding.UTF8.GetString(data)); // Hello, World
```

Issue

- Open Mar 31 2025
  [API Proposal]: Convert.{From/To}HexString(utf8)

```
1   ReadOnlySpan<byte> hexUtf8 = "48656C6C6F2C20576F726C64"u8;
2
3   // Простая и эффективная версия (без создания string)
4   byte[] data = Convert.FromHexString(hexUtf8);
5   Console.WriteLine(Encoding.UTF8.GetString(data)); // Hello, World
```

- Pr – merged Jul 25 2025
  Expose UTF-8 support from FromHexString, TryToHexString, and TryToHexStringLower on the Convert class

Итог

- Не нужно строить/парсить string
- Поддержка работы с буферами utf8
- Поддержка выбора регистра
- Семантика OperationStatus/ Try паттерн

```
1   namespace System;
2
3   public static class Convert
4   {
5           public static byte[] FromHexString(ReadOnlySpan<byte> utf8Source);
6           public static OperationStatus FromHexString(ReadOnlySpan<byte> utf8Source, Span<byte> destination, out int bytesConsumed,
7               out int bytesWritten);
8           public static bool TryToHexString(ReadOnlySpan<byte> source, Span<byte> utf8Destination, out int bytesWritten);
9           public static bool TryToHexStringLower(ReadOnlySpan<byte> source, Span<byte> utf8Destination, out int bytesWritten);
10  }
```

Libraries: Collections

- OrderedDictionary<TKey, TValue>
- Second lookup operation. 253 line

src/libraries/System.Text.Json/src/System/Text/Json/Nodes/JsonObject.cs

```
@@ -248,17 +248,21 @@
248 248
249 249             OrderedDictionary<string, JsonNode?> dict = Dictionary;
250 250
251     -           if (dict.TryGetValue(propertyName, out JsonNode? replacedValue))
    251 +           if (!dict.TryAdd(propertyName, value))
252 252             {
    253 +               int index = dict.IndexOf(propertyName);
```

Comment on line R253 — Resolved

```
    254 +               Debug.Assert(index >= 0);
    255 +               JsonNode? replacedValue = dict.GetAt(index).Value;
    256 +
```

Issue

Perfomance issue – opened Sep 17 2024

[Add an OrderedDictionary.TryAdd overload that returns the int-based index of an entry if the key already exists]

API Proposal

```
namespace System.Collections.Generic;

public partial class OrderedDictionary<TKey, TValue>
{
    public bool TryAdd(TKey key, TValue value);
+   public bool TryAdd(TKey key, TValue value, out int index);
}
```

Итог

- Pr – merged Nov 1 2024
  Add TryAdd and TryGetValue overloads with out int index to OrderedDictionary
- Добавили два метода

```
1   namespace System.Collections.Generic;
2
3   public partial class OrderedDictionary<TKey, TValue>
4   {
5       // Existing
6       // public bool TryAdd(TKey key, TValue value);
7       // public bool TryGetValue(TKey key, [MaybeNullWhen(false)] out TValue value);
8       public bool TryAdd(TKey key, TValue value, out int index);
9       public bool TryGetValue(TKey key, [MaybeNullWhen(false)] out TValue value, out int index);
10  }
```

Итог

- В этом же pr обновили JsonObject

src/libraries/System.Text.Json/src/System/Text/Json/Nodes/JsonObject.cs

```
@@ -248,9 +248,17 @@ internal void SetItem(string propertyName, JsonNode? value)
248 248
249 249                OrderedDictionary<string, JsonNode?> dict = Dictionary;
250 250
251     -              if (!dict.TryAdd(propertyName, value))
    251 +              if (
    252 + #if NET10_0_OR_GREATER
    253 +                  !dict.TryAdd(propertyName, value, out int index)
    254 + #else
    255 +                  !dict.TryAdd(propertyName, value)
    256 + #endif
    257 +                  )
252 258                {
    259 + #if !NET10_0_OR_GREATER
253 260                    int index = dict.IndexOf(propertyName);
    261 + #endif
254 262                    Debug.Assert(index >= 0);
255 263                    JsonNode? replacedValue = dict.GetAt(index).Value;
```

Замеры

Около 10-20% прирост производительности JsonObject

Method	Toolchain	Count	Mean	Error	StdDev	Ratio	RatioSD
AddAndUpdate	**main**	**1**	**126.1 ns**	**2.12 ns**	**1.77 ns**	**1.00**	**0.02**
AddAndUpdate	pr	1	112.7 ns	1.79 ns	1.68 ns	0.89	0.02
AddAndUpdate	**main**	**10**	**821.4 ns**	**11.77 ns**	**9.83 ns**	**1.00**	**0.02**
AddAndUpdate	pr	10	668.0 ns	13.07 ns	10.92 ns	0.81	0.02
AddAndUpdate	**main**	**50**	**3,886.8 ns**	**77.38 ns**	**163.21 ns**	**1.00**	**0.06**
AddAndUpdate	pr	50	3,014.6 ns	46.28 ns	41.02 ns	0.78	0.03

Libraries: ValidationContext

```
1   using System.ComponentModel.DataAnnotations;
2
3   var user = new User { Name = null };
4
5   var context = new ValidationContext(user)
6   {
7       MemberName = nameof(User.Name)
8   };
9
10  var results = new List<ValidationResult>();
11  Validator.TryValidateProperty(user.Name, context, results);
12
13  Console.WriteLine(results[0].ErrorMessage);
14
15  class User
16  {
17      [Required]
18      public string Name { get; set; }
19  }
```

Self-contained application

- Deployment size
- Trimming
- Reflection problem

C# Copy

```
void PrintMethodNames(Type type)
{
    foreach (var method in type.GetMethods())
    {
        Console.WriteLine(method.Name);
    }
}

// Called somewhere in the app
PrintMethodNames(typeof(DateTime));
```

Issue

- ValidationContext trim-unsafe
- Open Mar 5 2025
  [API Proposal]: Provide trim-safe overload for ValidationContext construction

API Proposal

```
namespace System.ComponentModel.DataAnnotations;

public sealed class ValidationContext
{
    public ValidationContext(object instance, string displayName, IServiceProvider? serviceProvider = null, IDictionary<obj
}
```

API Usage

```
// Trim-safe initializtion of ValidationContext
var validationContext = new ValidationContext(42, "MyMagicNumber", null, null);
```

Итого

- Pr – merged Mar 14 2025
  Add trim-safe ValidationContext constructor
- Возможность создать ValidationContext с явным displayName
- Избежать рефлексии
- Trim-safe

Спасибо за внимание!

Полезные материалы

- What's new in .NET 10 runtime | Microsoft Learn
- Understanding trim analysis - .NET | Microsoft Learn
- Нормализация Unicode / Хабр