

LinksPlatform's Platform.Unsafe Class Library

./ByteArrayExtensions.cs

```
1 using System.Runtime.InteropServices;
2 using Platform.Exceptions;
3 using Platform.Collections;
4
5 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7 namespace Platform.Unsafe
8 {
9     public static class ByteArrayExtensions
10     {
11         public static TTStruct ToStructure<TTStruct>(this byte[] bytes)
12             where TTStruct : struct
13         {
14             Ensure.Always.ArgumentNotEmpty(bytes, nameof(bytes));
15             var structureSize = Structure<TTStruct>.Size;
16             Ensure.Always.ArgumentMeetsCriteria(array => array.Length == structureSize, bytes,
17                 ↪ nameof(bytes), "Bytes array should be the same length as struct size.");
18             var pointer = Marshal.AllocHGlobal(structureSize);
19             Marshal.Copy(bytes, 0, pointer, structureSize);
20             var structure = Marshal.PtrToStructure<TTStruct>(pointer);
21             Marshal.FreeHGlobal(pointer);
22             return structure;
23         }
24     }
25 }
```

./IntPtr.cs

```
1 using System;
2 using System.Reflection;
3 using System.Runtime.InteropServices;
4 using Platform.Reflection;
5 using Platform.Reflection.Sigil;
6
7 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9 namespace Platform.Unsafe
10 {
11     public static class IntPtr<T>
12     {
13         public static readonly Func<IntPtr, T> GetValue;
14         public static readonly Action<IntPtr, T> SetValue;
15
16         static IntPtr()
17         {
18             GetValue = CompileGetValueDelegate();
19             SetValue = CompileSetValueDelegate();
20         }
21
22         static private Func<IntPtr, T> CompileGetValueDelegate()
23         {
24             return DelegateHelpers.Compile<Func<IntPtr, T>>(emitter =>
25             {
26                 if (CachedTypeInfo<T>.IsNumeric)
27                 {
28                     emitter.LoadArgument(0);
29                     emitter.LoadIndirect<T>();
30                     emitter.Return();
31                 }
32                 else
33                 {
34                     emitter.LoadArguments(0);
35                     emitter.Call(typeof(Marshal).GetGenericMethod(nameof(Marshal.PtrToStructure),
36                         ↪ Types<T>.Array, Types<IntPtr, Type, bool>.Array));
37                     emitter.Return();
38                 }
39             });
40         }
41
42         static private Action<IntPtr, T> CompileSetValueDelegate()
43         {
44             return DelegateHelpers.Compile<Action<IntPtr, T>>(emitter =>
45             {
46                 if (CachedTypeInfo<T>.IsNumeric)
47                 {
48                     emitter.LoadArguments(0, 1);
49                     emitter.StoreIndirect<T>();
50                     emitter.Return();
51                 }
52             });
53         }
54     }
55 }
```

```

50     }
51     else
52     {
53         emitter.LoadArguments(0, 1);
54         emitter.LoadConstant(true);
55         emitter.Call(typeof(Marshal).GetTypeInfo().GetMethod(nameof(Marshal.Structure_
        ↳ IntPtr), Types<object, IntPtr,
        ↳ bool>.Array));
56         emitter.Return();
57     }
58     });
59 }
60 }
61 }

```

./IntPtrExtensions.cs

```

1  using System;
2  using System.Runtime.CompilerServices;
3  using Platform.Numbers;
4
5  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7  namespace Platform.Unsafe
8  {
9      public static class IntPtrExtensions
10     {
11         [MethodImpl(MethodImplOptions.AggressiveInlining)]
12         public static TElement GetValue<TElement>(this IntPtr pointer) =>
13             ↳ IntPtr<TElement>.GetValue(pointer);
14
15         [MethodImpl(MethodImplOptions.AggressiveInlining)]
16         public static void SetValue<TElement>(this IntPtr pointer, TElement value) =>
17             ↳ IntPtr<TElement>.SetValue(pointer, value);
18
19         [MethodImpl(MethodImplOptions.AggressiveInlining)]
20         public static IntPtr GetElement(this IntPtr pointer, int elementSize, int index) =>
21             ↳ pointer + (elementSize * index);
22
23         [MethodImpl(MethodImplOptions.AggressiveInlining)]
24         public static unsafe IntPtr GetElement(this IntPtr pointer, long elementSize, long
25             ↳ index) => new IntPtr((byte*)pointer.ToPointer() + (elementSize * index));
26
27         [MethodImpl(MethodImplOptions.AggressiveInlining)]
28         public static IntPtr GetElement<TIndex>(this IntPtr pointer, int elementSize, TIndex
29             ↳ index) => pointer.GetElement((long)elementSize, (Integer)(Integer<TIndex>)index);
30     }
31 }

```

./MemoryBlock.cs

```

1  using System.Collections.Concurrent;
2  using System.Threading.Tasks;
3
4  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6  namespace Platform.Unsafe
7  {
8      public static unsafe class MemoryBlock
9      {
10         public static void Zero(void* pointer, long capacity)
11         {
12             var unlongs = capacity / sizeof(ulong);
13             Parallel.ForEach(Partitioner.Create(0, unlongs), range =>
14             {
15                 for (long i = range.Item1; i < range.Item2; i++)
16                 {
17                     *((ulong*)pointer + i) = 0;
18                 }
19             });
20             for (var i = unlongs * sizeof(ulong); i < capacity; i++)
21             {
22                 *((byte*)pointer + i) = 0;
23             }
24         }
25     }
26 }

```

./Structure.cs

```
1 using System.Runtime.InteropServices;
2
3 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
5 namespace Platform.Unsafe
6 {
7     public static class Structure<TStruct>
8         where TStruct : struct
9     {
10         /// <summary>
11         /// <para>
12         /// Returns the size of an unmanaged type in bytes.
13         /// This method do this without throwing exceptions for generic types as <see
14         /// → cref="Marshal.SizeOf{T}" /> and <see cref="Marshal.SizeOf(System.Type)" /> do.
15         /// </para>
16         /// <para>
17         /// Возвращает размер неуправляемого типа в байтах.
18         /// Этот метод делает это без выбрасывания исключений для универсальных типов, как это
19         /// → делают<see cref = "Marshal.SizeOf {T}" /> и < see cref = "Marshal.SizeOf
20         /// → (System.Type)" />.
21         /// </para>
22         /// </summary>
23         /// <remarks>
24         /// <para>
25         /// Based on proposed solution at https://stackoverflow.com/a/18167584/710069
26         /// For actual differences in .NET code see:
27         /// https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L202
28         /// → https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L219-L222
29         /// Note that this behaviour can be changed in future versions of .NET
30         /// </para>
31         /// <para>
32         /// На основе предложенного решения https://stackoverflow.com/a/18167584/710069
33         /// Фактические различия в коде .NET:
34         /// https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L202
35         /// → https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L219-L222
36         /// Обратите внимание, что это поведение может быть изменено в будущих версиях.NET
37         /// </para>
38         /// </remarks>
39         public static int Size { get; } = Marshal.SizeOf(default(TStruct));
40     }
41 }
```

./StructureExtensions.cs

```
1 using System.Runtime.InteropServices;
2
3 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
5 namespace Platform.Unsafe
6 {
7     public static class StructureExtensions
8     {
9         public static byte[] ToBytes<TStruct>(this TStruct obj)
10             where TStruct : struct
11         {
12             var structureSize = Structure<TStruct>.Size;
13             var bytes = new byte[structureSize];
14             var pointer = Marshal.AllocHGlobal(structureSize);
15             Marshal.StructureToPtr(obj, pointer, true);
16             Marshal.Copy(pointer, bytes, 0, structureSize);
17             Marshal.FreeHGlobal(pointer);
18             return bytes;
19         }
20     }
21 }
```

Index

./ByteArrayExtensions.cs, 1
./IntPtr.cs, 1
./IntPtrExtensions.cs, 2
./MemoryBlock.cs, 2
./Structure.cs, 2
./StructureExtensions.cs, 3