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,
                 ↪  nameof(bytes), "Bytes array should be the same length as struct size.");
17              var pointer = Marshal.AllocHGlobal(structureSize);
18              Marshal.Copy(bytes, 0, pointer, structureSize);
19              var structure = Marshal.PtrToStructure<TTStruct>(pointer);
20              Marshal.FreeHGlobal(pointer);
21              return structure;
22          }
23      }
24  }
```

./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>>(emiter =>
25              {
26                  if (CachedTypeInfo<T>.IsNumeric)
27                  {
28                      emiter.LoadArgument(0);
29                      emiter.LoadIndirect<T>();
30                      emiter.Return();
31                  }
32                  else
33                  {
34                      emiter.LoadArguments(0);
35                      emiter.Call(typeof(Marshal).GetGenericMethod(nameof(Marshal.PtrToStructure),
                         ↪  Types<T>.Array, Types<IntPtr, Type, bool>.Array));
36                      emiter.Return();
37                  }
38              });
39          }
40
41          static private Action<IntPtr, T> CompileSetValueDelegate()
42          {
43              return DelegateHelpers.Compile<Action<IntPtr, T>>(emiter =>
44              {
45                  if (CachedTypeInfo<T>.IsNumeric)
46                  {
47                      emiter.LoadArguments(0, 1);
48                      emiter.StoreIndirect<T>();
49                      emiter.Return();
```

```
50                  }
51                  else
52                  {
53                      emiter.LoadArguments(0, 1);
54                      emiter.LoadConstant(true);
55                      emiter.Call(typeof(Marshal).GetTypeInfo().GetMethod(nameof(Marshal.Structure↵
    ↪   ToPtr), Types<object, IntPtr,
    ↪   bool>.Array));
56                      emiter.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) =>
    ↪   IntPtr<TElement>.GetValue(pointer);
13
14         [MethodImpl(MethodImplOptions.AggressiveInlining)]
15         public static void SetValue<TElement>(this IntPtr pointer, TElement value) =>
    ↪   IntPtr<TElement>.SetValue(pointer, value);
16
17         [MethodImpl(MethodImplOptions.AggressiveInlining)]
18         public static IntPtr GetElement(this IntPtr pointer, int elementSize, int index) =>
    ↪   pointer + (elementSize * index);
19
20         [MethodImpl(MethodImplOptions.AggressiveInlining)]
21         public static unsafe IntPtr GetElement(this IntPtr pointer, long elementSize, long
    ↪   index) => new IntPtr((byte*)pointer.ToPointer() + (elementSize * index));
22
23         [MethodImpl(MethodImplOptions.AggressiveInlining)]
24         public static IntPtr GetElement<TIndex>(this IntPtr pointer, int elementSize, TIndex
    ↪   index) => pointer.GetElement((long)elementSize, (Integer)(Integer<TIndex>)index);
25     }
26  }
```

./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 ulongs = capacity / sizeof(ulong);
13             Parallel.ForEach(Partitioner.Create(0, ulongs), range =>
14             {
15                 for (long i = range.Item1; i < range.Item2; i++)
16                 {
17                     *((ulong*)pointer + i) = 0;
18                 }
19             });
20             for (var i = ulongs * 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.
14          /// </para>
15          /// <para>
16          /// Возвращает размер неуправляемого типа в байтах.
17          /// Этот метод делает это без выбрасывания исключений для универсальных типов, как это
17          ↪ делают <see cref="Marshal.SizeOf{T}()"/> и <see cref="Marshal.SizeOf(System.Type)"/>.
18          /// </para>
19          /// </summary>
20          /// <remarks>
21          /// <para>
22          /// Based on proposed solution at https://stackoverflow.com/a/18167584/710069
23          /// For actual differences in .NET code see:
24          /// https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95
24          ↪ d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L202
25          /// https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95
25          ↪ d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L219-L222
26          /// Note that this behaviour can be changed in future versions of .NET
27          /// </para>
28          /// <para>
29          /// На основе предложенного решения https://stackoverflow.com/a/18167584/710069
30          /// Фактические различия в коде .NET:
31          /// https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95
31          ↪ d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L202
32          /// https://github.com/Microsoft/referencesource/blob/f82e13c3820cd04553c21bf6da01262b95
32          ↪ d9bd43/mscorlib/system/runtime/interopservices/marshal.cs#L219-L222
33          /// Обратите внимание, что это поведение может быть изменено в будущих версиях.NET
34          /// </para>
35          /// </remarks>
36          public static int Size { get; } = Marshal.SizeOf(default(TStruct));
37      }
38  }


./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