```
LinksPlatform's Platform Reflection Sigil Class Library
./Platform.Reflection.Sigil/DelegateHelpers.cs
   using System;
   using System.Collections.Generic;
   using Sigil;
   using Platform. Exceptions;
4
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Reflection.Sigil
        public static class DelegateHelpers
10
11
            public static TDelegate Compile<TDelegate>(Action<Emit<TDelegate>> emitCode)
12
13
                var @delegate = default(TDelegate);
14
15
                try
16
                    var emiter = Emit<TDelegate>.NewDynamicMethod();
                    emitCode(emiter);
                    @delegate = emiter.CreateDelegate();
19
                }
20
21
                catch (Exception exception)
22
                    exception.Ignore();
23
                finally
25
26
                    if (EqualityComparer<TDelegate>.Default.Equals(@delegate, default))
27
2.8
                         var factory = new NotSupportedExceptionDelegateFactory<TDelegate>();
29
                         @delegate = factory.Create();
31
32
                return @delegate;
33
            }
34
       }
35
   }
./Platform.Reflection.Sigil/EmitExtensions.cs
   using System;
   using Sigil;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Reflection.Sigil
        public static class EmitExtensions
9
            public static Emit<TDelegate> LoadConstantOne<TDelegate>(this Emit<TDelegate> emiter,
10
                Type constantType)
                if (constantType == typeof(float))
                {
13
                    emiter.LoadConstant(1F);
                else if (constantType == typeof(double))
16
17
                     emiter.LoadConstant(1D);
                }
19
                else if (constantType == typeof(long))
20
                    emiter.LoadConstant(1L);
22
23
                else if (constantType == typeof(ulong))
24
                    emiter.LoadConstant(1UL);
26
2.7
                else if (constantType == typeof(int))
                {
29
                    emiter.LoadConstant(1);
30
3.1
                else if (constantType == typeof(uint))
32
33
                    emiter.LoadConstant(1U);
34
35
                else if (constantType == typeof(short))
37
                    emiter.LoadConstant(1);
```

```
emiter.Convert<short>();
    }
    else if (constantType == typeof(ushort))
        emiter.LoadConstant(1);
        emiter.Convert<ushort>();
    else if (constantType == typeof(sbyte))
        emiter.LoadConstant(1);
        emiter.Convert<sbyte>();
    else if (constantType == typeof(byte))
        emiter.LoadConstant(1);
        emiter.Convert<byte>();
    }
    else
    {
        throw new NotSupportedException();
    return emiter;
}
public static Emit<TDelegate> LoadConstant<TDelegate>(this Emit<TDelegate> emiter, Type
   constantType, object constantValue)
    if (constantType == typeof(float))
    {
        emiter.LoadConstant((float)constantValue);
    }
    else if (constantType == typeof(double))
        emiter.LoadConstant((double)constantValue);
    else if (constantType == typeof(long))
        emiter.LoadConstant((long)constantValue);
    }
    else if (constantType == typeof(ulong))
        emiter.LoadConstant((ulong)constantValue);
    else if (constantType == typeof(int))
        emiter.LoadConstant((int)constantValue);
    else if (constantType == typeof(uint))
        emiter.LoadConstant((uint)constantValue);
    }
    else if (constantType == typeof(short))
        emiter.LoadConstant((short)constantValue);
        emiter.Convert<short>();
    }
    else if (constantType == typeof(ushort))
        emiter.LoadConstant((ushort)constantValue);
        emiter.Convert<ushort>();
    else if (constantType == typeof(sbyte))
        emiter.LoadConstant((sbyte)constantValue);
        emiter.Convert<sbyte>();
    else if (constantType == typeof(byte))
        emiter.LoadConstant((byte)constantValue);
        emiter.Convert<byte>();
    }
    else
        throw new NotSupportedException();
    return emiter;
}
```

3.9

41

44 45

46 47

48

49 50

51 52 53

55

57

58 59

60

61 62

63

64

66

67

69 70 71

72

73 74

75

76

77 78

79 80

81

83 84

85 86

87

88

90

91

93

94 95

97 98

99 100

101

102

104 105

106

107

108 109

110

111 112

113

115

```
public static Emit<TDelegate> Increment<TDelegate>(this Emit<TDelegate> emiter, Type
116
                 valueType)
117
                 emiter.LoadConstantOne(valueType);
118
                 emiter.Add();
119
                 return emiter;
121
122
             public static Emit<TDelegate> Decrement<TDelegate>(this Emit<TDelegate> emiter, Type
123
                 valueType)
124
                 emiter.LoadConstantOne(valueType);
                 emiter.Subtract();
                 return emiter;
127
             }
128
129
             public static Emit<TDelegate> LoadArguments<TDelegate>(this Emit<TDelegate> emiter,
130
                 params ushort[] arguments)
131
                 for (var i = 0; i < arguments.Length; i++)</pre>
132
133
                      emiter.LoadArgument(arguments[i]);
135
                 return emiter;
             }
137
138
             public static Emit<TDelegate> CompareGreaterThan<TDelegate>(this Emit<TDelegate> emiter,
139
                 bool isSigned)
140
                 if (isSigned)
141
142
                      emiter.CompareGreaterThan();
143
                 }
144
145
                 else
                 {
146
                      emiter.UnsignedCompareGreaterThan();
147
148
                 return emiter;
149
             }
150
151
             public static Emit<TDelegate> CompareLessThan<TDelegate>(this Emit<TDelegate> emiter,
152
                 bool isSigned)
                 if (isSigned)
154
                 {
155
                      emiter.CompareLessThan();
156
                 }
157
                 else
158
                 {
                      emiter.UnsignedCompareLessThan();
160
161
                 return emiter;
162
             }
163
164
             public static Emit<TDelegate> BranchIfGreaterOrEqual<TDelegate>(this Emit<TDelegate>
165
                 emiter, bool isSigned, Label label)
166
                 if (isSigned)
167
                 {
168
                      emiter.BranchIfGreaterOrEqual(label);
169
                 }
170
                 else
171
                 {
172
                      emiter.UnsignedBranchIfGreaterOrEqual(label);
                 }
174
                 return emiter;
175
             }
176
177
             public static Emit<TDelegate> BranchIfLessOrEqual<TDelegate>(this Emit<TDelegate>
178
                 emiter, bool isSigned, Label label)
179
                 if (isSigned)
180
                 {
                      emiter.BranchIfLessOrEqual(label);
182
                 }
183
                 else
185
                      emiter.UnsignedBranchIfLessOrEqual(label);
186
```

```
187
                  return emiter;
188
189
        }
    }
191
./Platform.Reflection.Sigil/NotSupportedExceptionDelegateFactory.cs
    using System;
using Sigil;
using Platform.Interfaces;
    #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
    namespace Platform.Reflection.Sigil
         public class NotSupportedExceptionDelegateFactory<TDelegate> : IFactory<TDelegate>
10
             public TDelegate Create()
11
^{12}
                  var emiter = Emit<TDelegate>.NewDynamicMethod();
13
                  emiter.NewObject<NotSupportedException>();
14
                  emiter.Throw();
                  return emiter.CreateDelegate();
17
         }
18
    }
19
./Platform.Reflection.Sigil.Tests/InlineTests.cs
    using Xunit;
using SigilVNext = Sigil;
    using PlatformReflectionSigil = Platform.Reflection.Sigil;
    namespace Platform.Reflection.Sigil.Tests
 5
 6
         public static class InlineTests
             [Fact]
 9
10
             public static void SimpleInlineTest()
11
                  //SigilVNext.Disassembler
12
13
14
             private static void InnerMethod()
15
             }
17
18
             private static void OuterMethod()
19
20
21
                  InnerMethod();
^{22}
         }
23
    }
24
```

Index

- ./Platform.Reflection.Sigil.Tests/InlineTests.cs, 4
 ./Platform.Reflection.Sigil/DelegateHelpers.cs, 1
 ./Platform.Reflection.Sigil/EmitExtensions.cs, 1
 ./Platform.Reflection.Sigil/NotSupportedExceptionDelegateFactory.cs, 4