Net Micro Framework での Standard 機能実装について

標準機能となる項目は

修正対象フォルダ

CLR\Libraries\CorLib

関数一例 : corlib\_native\_System\_GC.cpp

HRESULT Library\_corlib\_native\_System\_GC::AnyPendingFinalizers\_\_\_STATIC\_\_BOOLEAN( CLR\_RT\_StackFrame& stack )

解析？

Library\_corlib\_native : lib実装ルール？

System : C# での System;

GC : System のGCクラス

AnyPendingFinalizers : メソッド名(\_ctor ならコンストラクタ？)

\_\_\_ : \_\*3 , メソッドの戻り値と引数を設定していく際の始まり？

\_\_ : 間を \_ \*2, でまとめていく？

STATIC : メソッドのタイプ？

BOOLEAN : 型情報？

SZARRAY\_～ : 配列(～ : U1, I4, …)

関数内の実装

{

// Start

NATIVE\_PROFILE\_CLR\_CORE();

TINYCLR\_HEADER();

// 実体部分

stack.SetResult\_Boolean( g\_CLR\_RT\_ExecutionEngine.m\_finalizersPending.IsEmpty() == false

|| (g\_CLR\_RT\_ExecutionEngine.m\_finalizerThread != NULL && !g\_CLR\_RT\_ExecutionEngine.m\_finalizerThread->CanThreadBeReused()));

// End

TINYCLR\_NOCLEANUP\_NOLABEL();

}

メソッドの場合、

NATIVE\_PROFILE\_CLR\_CORE();

の呼び出しのみでOK?。

corlib\_native.cpp ファイル

#include "CorLib.h"

static const CLR\_RT\_MethodHandler method\_lookup[] =

{

：

Library\_corlib\_native\_System\_GC::AnyPendingFinalizers\_\_\_STATIC\_\_BOOLEAN,

NULL,

Library\_corlib\_native\_System\_GC::SuppressFinalize\_\_\_STATIC\_\_VOID\_\_OBJECT,

Library\_corlib\_native\_System\_GC::ReRegisterForFinalize\_\_\_STATIC\_\_VOID\_\_OBJECT,

：

};

const CLR\_RT\_NativeAssemblyData g\_CLR\_AssemblyNative\_mscorlib =

{

"mscorlib",

0x4EB3CA01,

method\_lookup

};

corlib\_native.h ファイル

#ifndef \_CORLIB\_NATIVE\_H\_

#define \_CORLIB\_NATIVE\_H\_

:

struct Library\_corlib\_native\_System\_GC

{

TINYCLR\_NATIVE\_DECLARE(AnyPendingFinalizers\_\_\_STATIC\_\_BOOLEAN);

TINYCLR\_NATIVE\_DECLARE(SuppressFinalize\_\_\_STATIC\_\_VOID\_\_OBJECT);

TINYCLR\_NATIVE\_DECLARE(ReRegisterForFinalize\_\_\_STATIC\_\_VOID\_\_OBJECT);

//--//

};

:

extern const CLR\_RT\_NativeAssemblyData g\_CLR\_AssemblyNative\_mscorlib;

#endif // \_CORLIB\_NATIVE\_H\_

既存ライブラリを使用可能に

.NET Standard 2.0 では、既存の .NET Framework 向けライブラリを（主に .NET Core 上で）実行できるように、FCL 互換のファサードアセンブリが追加されています。

これらは [TypeForwardedTo] 属性が大量に宣言されており、実行時に本来の FCL のアセンブリの型に転送することで、TypeLoadException などで落ちないようにする役割を持ちます。

具体的には、.NET Standard のリポジトリによれば、以下のアセンブリです。

mscorlib

System

System.ComponentModel.Composition

System.Core

System.Data

System.Drawing

System.IO.Compression

System.IO.Compression.FileSystem

System.Net.Http

System.Numerics

System.Runtime.Serialization

System.Web

System.Xml.Linq

System.Xml

<https://github.com/dotnet/coreclr/issues/1784>

<https://blogs.msdn.microsoft.com/visualstudio_jpn/2017/01/30/whatnetdevelopersoughttoknowtostartin2017/>

.Net Standard のドキュメント

<https://github.com/dotnet/standard/blob/master/docs/versions/netstandard2.0.md>

* どのNamespace のメソッドの対応数が記載されている。