

ユーザー独自開発機能の CommonMPへの組み込み方法（サンプル）

1. 概要

CommonMPは ユーザーが開発したツール等の独自機能を 組み込むことができます。
ここでは、例として CommonMPの画面に固有のメニューを追加し、そのメニューから、独自に作成したプログラム(実行体)を起動する場合を例に取り、その組み込み方法を説明します。

2. 独自開発処理の追加手順

独自開発処理を、画面のメニューから呼び出すようにするには、下記の手順で、CommonMP への組み込みを行います。

- ①画面から呼び出された時に行う処理(独自開発処理)を作成します。
- ②上記処理を起動するためのモジュールを作成します。(DLLとして提供)
(作成方法は、4. 2章で述べます。)
- ③上記モジュールをCommonMPに登録します。(登録方法は4. 3章で述べます。)
- ④ Menu.xml を修正して、画面上にメニューを表示できるようにします。
(Menu.xml の修正方法は、4. 4章で述べます。)

以下、サンプルを用いて その方法を説明します。

3. サンプル機能の概要

先ず、組み込むサンプル機能の説明を行います。

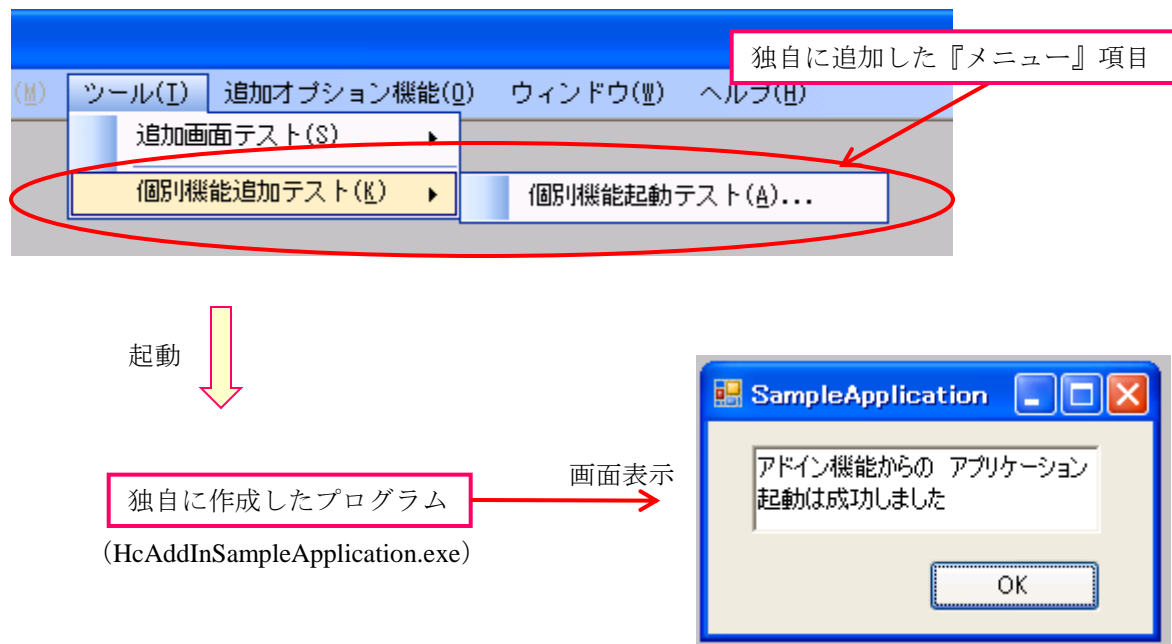


図 3. 1 サンプルの動作概要図

図3. 1に 本サンプルの 動作の概要を示します。

CommonMPに追加したメニューの「個別機能起動テスト」を選択すると、サンプルプログラム (HcAddInSampleApplication.exe)を起動し、そのプログラムが独自の画面を表示します。サンプルとしての機能は画面表示だけで、 表示された画面の『OK』ボタンを押下すると、サンプルプログラムは、終了します。

以下に、本サンプルプログラムを、CommonMPへ組み込む手順を 順に説明していきます。

4. サンプルプログラム組み込みの説明

4. 1 サンプルプログラム（ユーザー独自開発機能処理）の作成

先ず、ユーザーで追加したい機能処理を作成します。

本サンプルのプログラムソースは、下記ディレクトリに格納されています。

¥CommonMP¥Source¥HYMCO¥AddInTools¥HSViewer¥AddInDotNetScreen¥
AddinDotNetJobSample¥HcAddInSampleApplication¥

ディレクトリ内の『HcAddInSampleApplication.csproj』をマイクロソフト社の Visual Studio C# で開くと、プログラムのコンパイル、デバッグ実行が行えます。

図4. 1にサンプルプログラムのコンパイル／実行の概要を示します。プロジェクトファイルを、マイクロソフト社のVisual Studio を用いて開くと、プログラム開発ソリューションが表示されます。 此处で、Visual Studio 上からビルドを行うと、「HcAddInSampleApplication.exe」という実行体を作成されます。

作成された実行体をダブルクリック等により起動すると、図に示すような画面が開きます。画面上の「OK」ボタンを押下すると、プログラムは終了します。

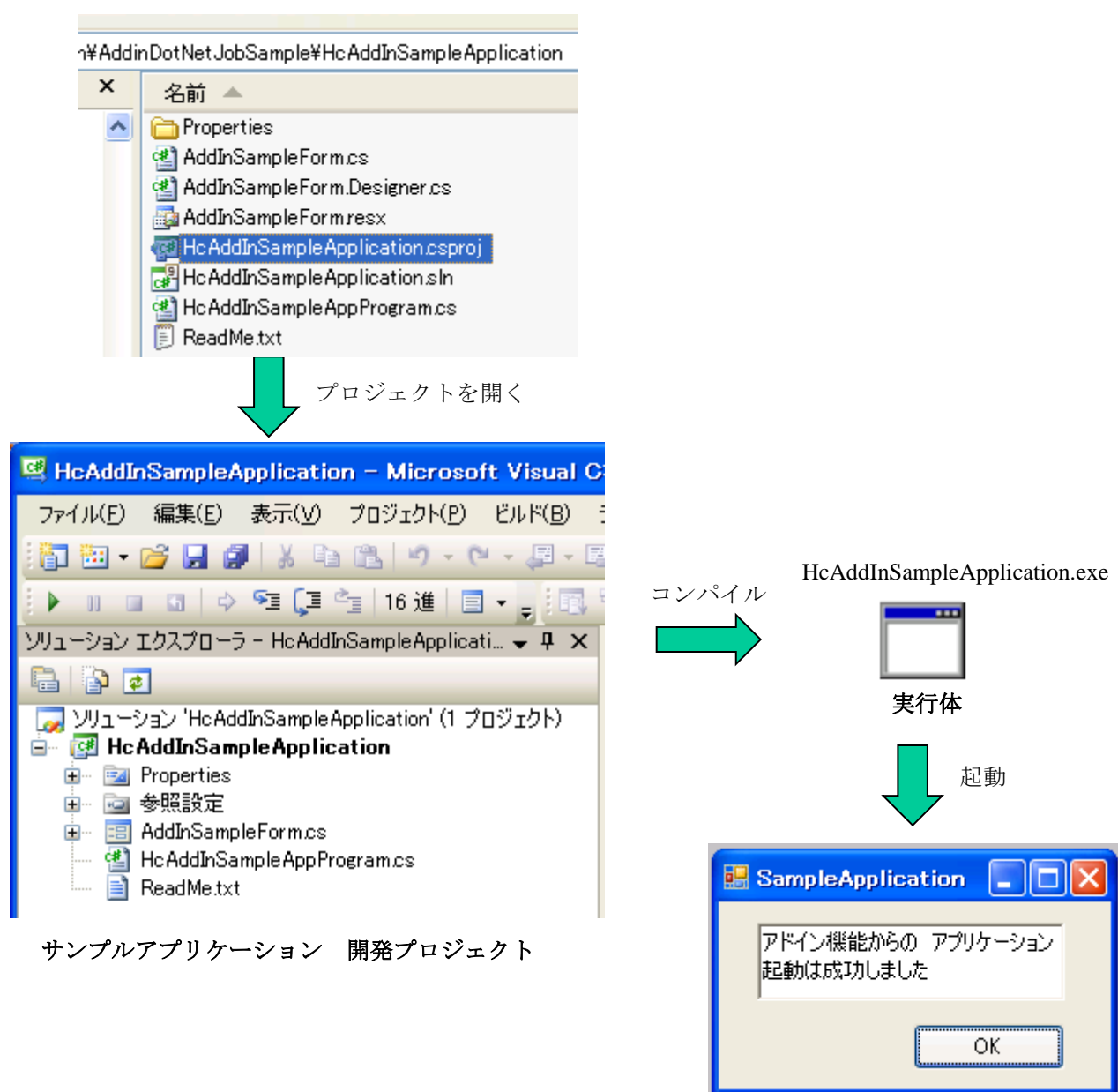


図4. 1 ユーザー独自処理プログラムの作成

4. 2 サンプルプログラム（ユーザー独自開発機能処理）を
起動するためのモジュールを作成

① 作成すべきモジュール概要

4. 1では、ユーザーが独自に作成したプログラム『HcAddInSampleApplication.exe』を、ダブルクリック等の操作により 手動で起動していました。ここでは、CommonMPからの命令で プログラムが起動できるように モジュールを作成します。

図4. 2. 1に ここで作成するモジュールの動作概念を示します。 オペレーターが、画面上のメニューを押下するとメニューに登録してある モジュール識別名称とコールバック番号（登録方法は後述）を CommonMP本体にイベントとして通知します。 CommonMP本体は、受け取ったイベント内の モジュール識別名称を元に、 そのイベントをどのモジュールに通知するかを判断し、指定されたモジュールにコールバック番号をイベントとして通知します。

ユーザーは、イベントを受け取ってから処理を実装する必要があります。

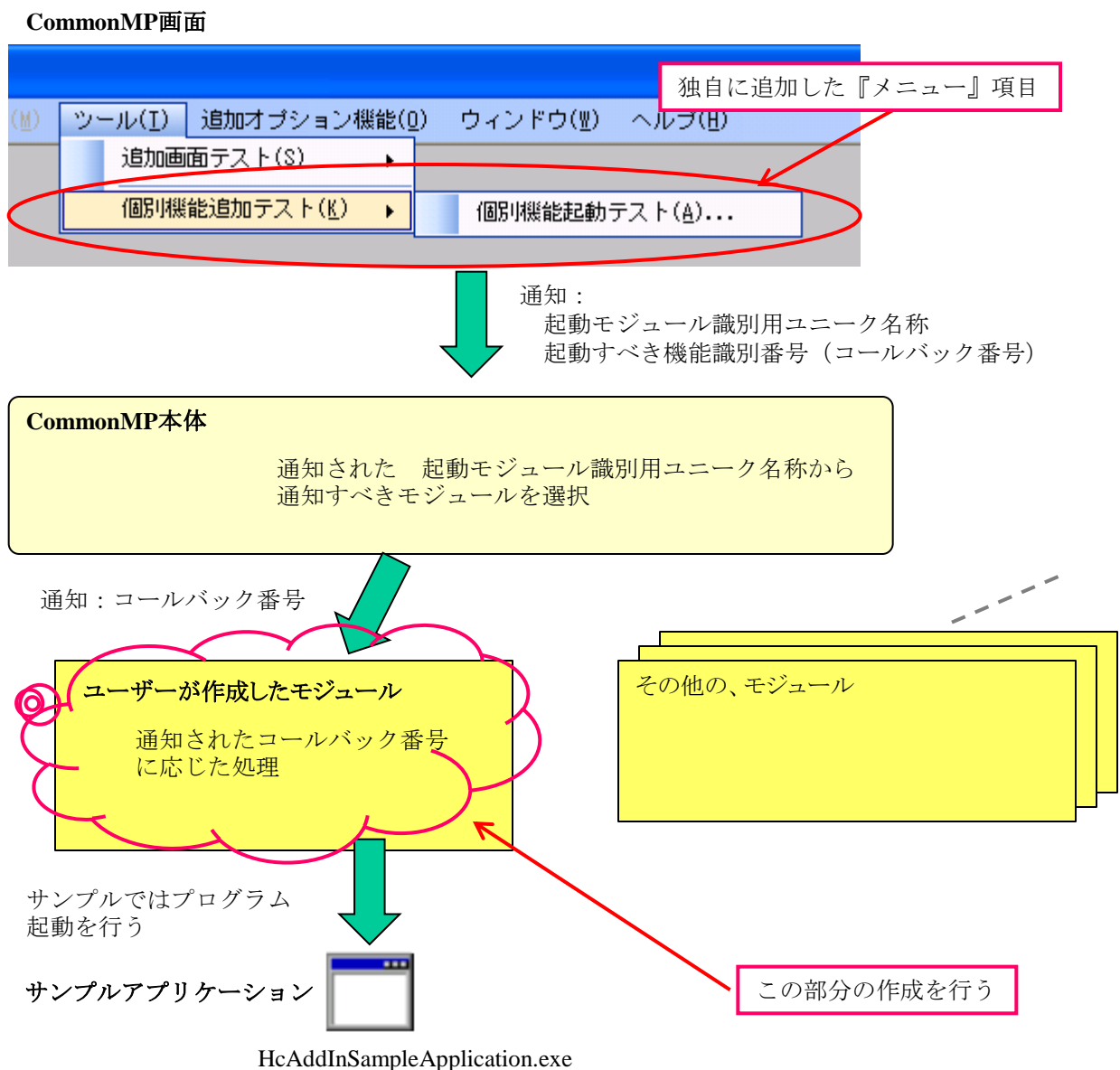


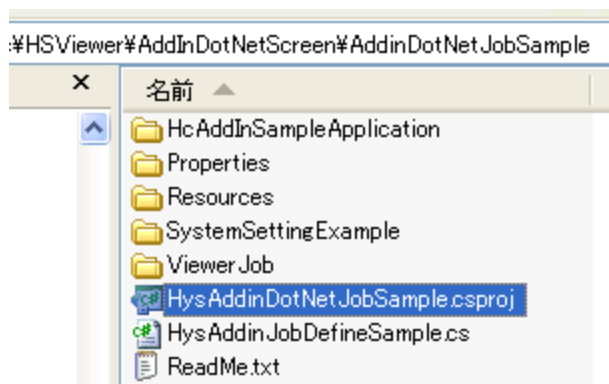
図 4. 2. 1 メニュー押下から サンプルアプリケーション動作までの動作概念
及び、作成すべきモジュール

② サンプルプログラムのコンパイルとデバッグ

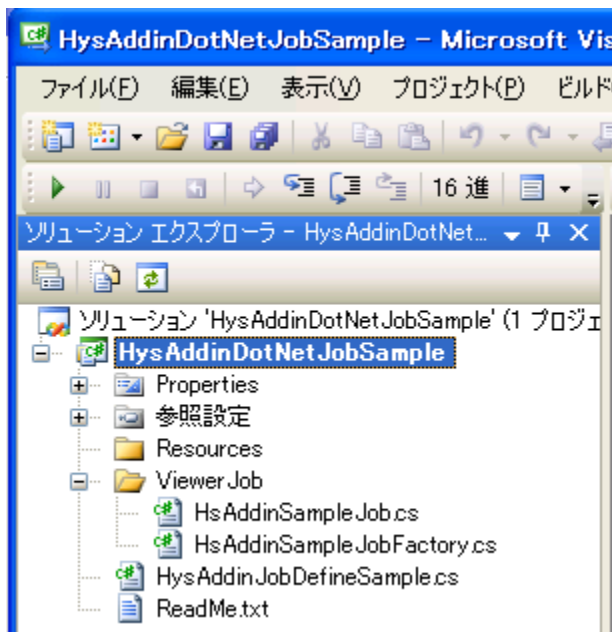
サンプルのプログラムソースは、下記ディレクトリに格納されています。

¥CommonMP¥Source¥HYMCO¥AddInTools¥HSViewer¥AddInDotNetScreen¥
AddinDotNetJobSample¥

ディレクトリ内の『 HysAddinDotNetJobSample.csproj 』を マイクロソフト社の Visual Studio C# で開くと、プログラムのコンパイルが行えます。コンパイルにより、「HysAddinDotNetJobSample.dll」 というDLLが生成されます。



プロジェクトを開く



コンパイル



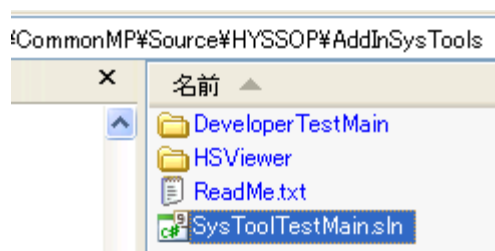
HysAddinDotNetJobSample.dll

図 4. 2. 2 DLL作成

しかしながら、DLL単体では、マイクロソフト社 Visual Studio C# 上から、デバッグを行う事ができません。そこで、DLLをデバッグする開発環境を用意しました。

先ず、¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥ 下の「SysToolTestMain.sln」をマイクロソフト社 Visual Studio C# から開きます。

すると、DLLを起動する為のメインプログラム用ソリューションが開かれます。このソリューション内に 作成する HysAddinDotNetJobSample.csproj が取り込まれており、デバッガー上から、ソース上に ブレークポイントを設定して、デバッグすることが可能となります。



プロジェクトを開く

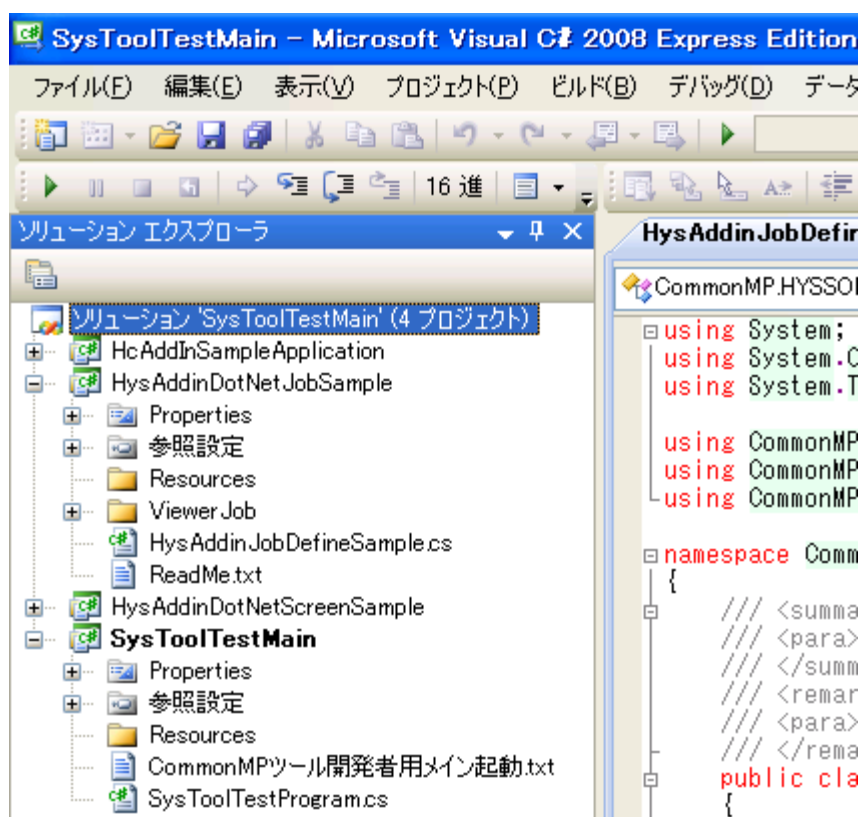


図4. 2. 3 DLLデバッグ用 ソリューション立ち上げ

③ 作成するモジュールクラス

イベントを受け取ったモジュールは、通知されたコールバック番号に従った処理を行います。独自メニューを複数登録した場合、オペレータが画面上から選択したメニューに応じたコールバック番号(後述)が通知されるため、メニューに応じた処理を実装できます。

ここで、ユーザーが独自に作成するモジュールは CommonMPフレームワーク上で動作する必要がある為、CommonMPが提供する親クラスの派生クラスとして作成する必要があります。図4. 3 に クラスの派生関係を示します。

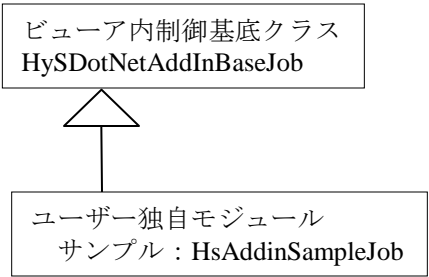


図 4. 2. 4 ユーザーが作成するクラスの派生関係

図4. 2. 4 に示す様に、ユーザー独自モジュールを CommonMPが提供する HySDotNetAddInBaseJob クラスの派生として作成することで、メニューイベント受け取りや、対応モジュールの選択等の処理は親クラス側で行われ、派生クラス側では、通知されたイベントに対する処理だけを実装する事に専念できます。具体的には、下記メソッドをオーバーライドして実装する事になります。

1) メニューコールバック処理 (MenuCallBack(コールバック番号)) (実装が必須)

オペレーターが画面のメニューを押下した時に呼ばれるメソッドです。引数で与えられたコールバック番号を元に、どのメニューが呼ばれたかを判断して、それに応じた処理を行います。本メソッドの実装は 必須です。

2) 初期化処理 (Initialize) (必要に応じて実装)

ユーザー独自モジュールが最初に立ち上がった時に行う処理です。特別に行う処理が無ければ、本メソッドの実装は不要です。

3) 終了処理 (Terminate) (必要に応じて実装)

画面が終了する時に、呼ばれる処理です。特別に行う処理が無ければ、本メソッドの実装は不要です。

4) システム終了判断 (ExitOK) (必要に応じて実装)

画面が終了する時に、Terminate 後に呼ばれ、終了処理が完了したか否かを返します。特別にシステム終了時に行う処理が無ければ、本メソッドの実装は不要です。

図4. 2. 5に CommonMP画面と メソッドの関係を示します。

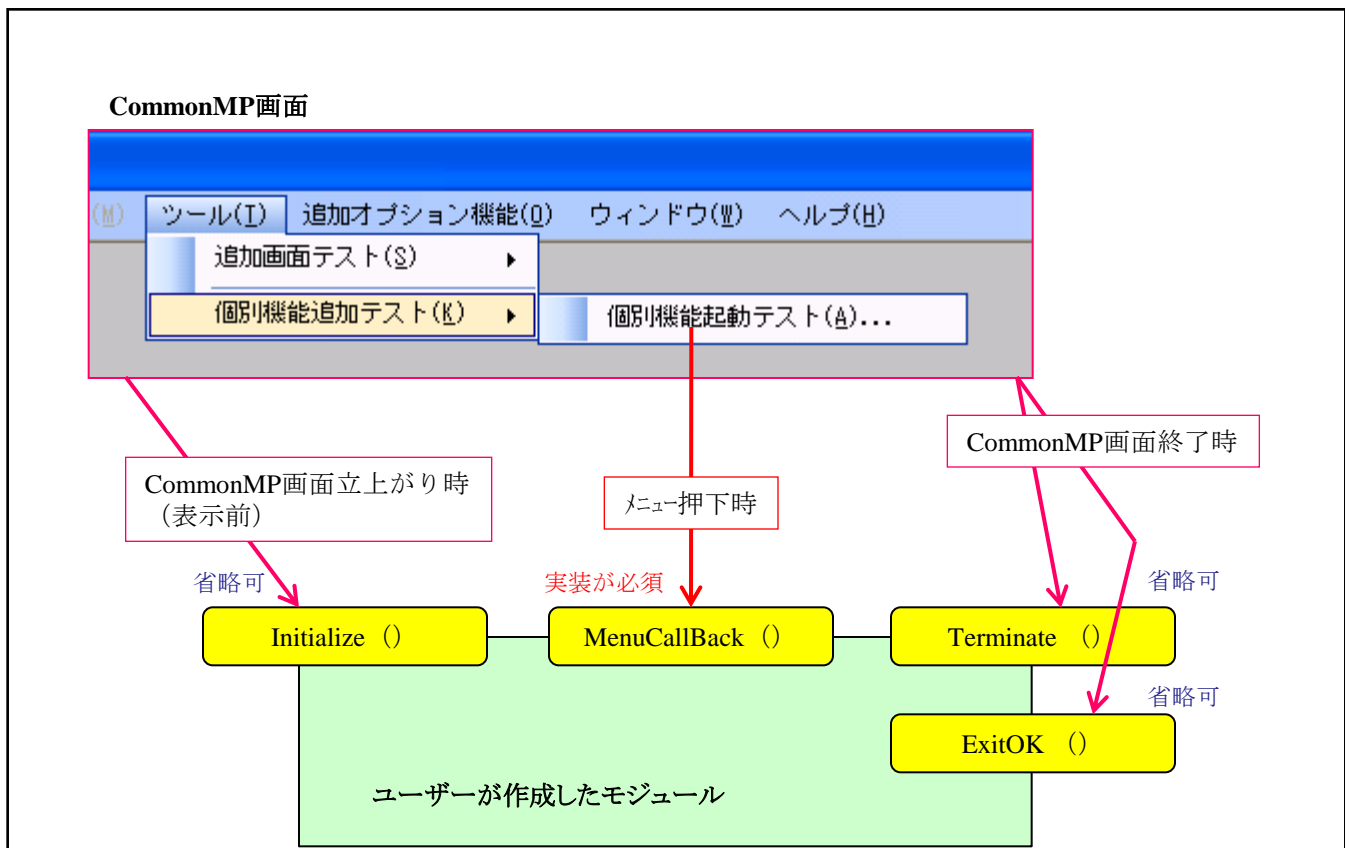


図 4. 2. 5 実装するメソッド

サンプル (HsAddInSampleJob) では、下記処理を実装しています。

① MenuCallBack (ICallBackNo)

引数 ICallBackNo のコールバック番号値が 199877 (プログラム上では、HysAddInJobDefineSample.SAMPLE_DO_SOMETHINGとして 定義されている) の時に、HcAddInSampleApplication.exe を起動しています。 また、コールバック番号が 1998776 (プログラム上では、 HysAddInJobDefineSample.SAMPLE_DISP_MANUAL として定義されている) の時に、本解説を表示します。

② Terminate ()

HcAddInSampleApplication.exe が立ち上がっている場合には、それを強制終了させています。

③ Initialize () 、 ExitOK () 何も処理していません。(省略可能です。)

④ ファクトリークラス

CommonMPでは、あるクラスのインスタンス(実体)を 生成する(new)ために、ファクトリークラスを準備する必要があります。(要素モデルと同様)

図4. 2. 6に ファクトリークラスの動作概要を示します。

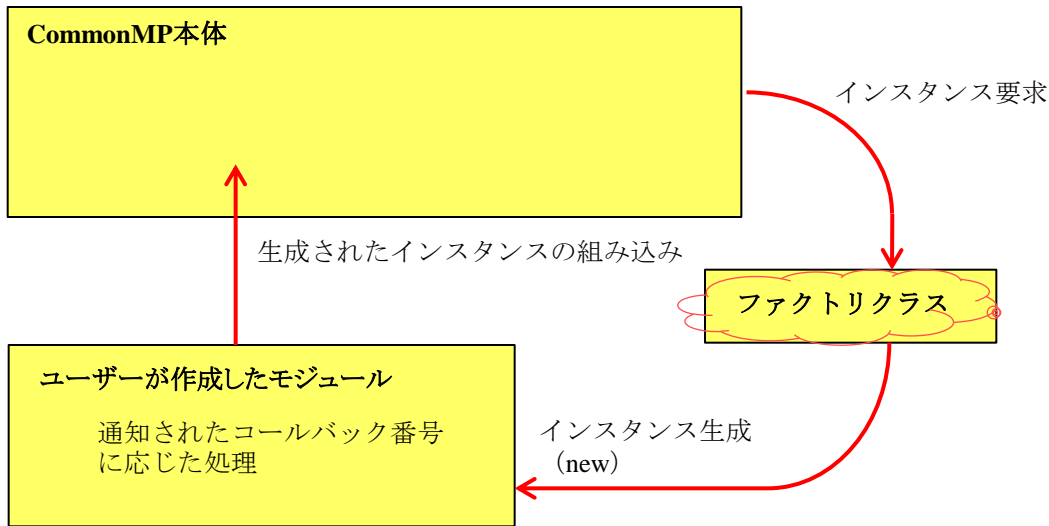


図 4. 2. 6 ファクトリークラスの働き

・ ファクトリークラスの派生関係

ユーザーが作成するファクトリークラスは、アドイン用のファクトリ親クラス (HySDotNetAddInBaseJobFactory)から派生させて下さい。(図4. 2. 7参照)

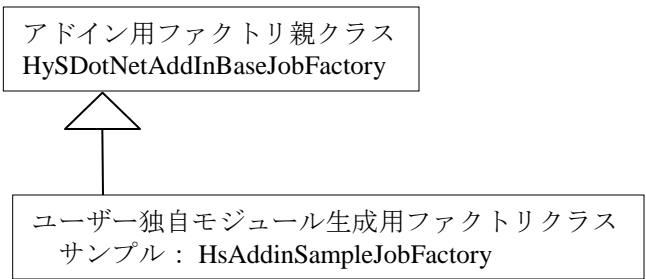


図 4. 2. 7 ユーザーが作成するファクトリークラスの派生関係

- ・ファクトリクラスが実装するメソッド

以下に、ファクトリークラスでユーザーが実装すべきメソッド及び、その実装例を示します。

```

namespace CommonMP.HYSSOP.OptionImpl.HSAddinJob.Sample
{
    /// <summary><para>class outline:</para> ...
    public class HsAddinSampleJobFactory : HySDotNetAddInBaseJobFactory
    {
        /// <summary> ...
        public HsAddinSampleJobFactory()
        {
            ///.コンストラクター内で・ファクトリIDを設定する
            ///.ToDo.独自のファクトリIDを設定して下さい。
            SetFactoryID(HysAddinJobDefineSample.FACTORY_ID_VIEWRJOB);
        }

        /// <summary><para>method outline:</para> ...
        override public HySKind GetSimKind()
        {
            ///.ToDo.シミュレーション(業務ID)を取得する
            return HysAddinJobDefineSample.BUSINESS_KIND;
        }

        /// <summary><para>method outline:</para> ...
        override public Boolean EqualSimKind(HySKind csSimKindID)
        {
            ///.ToDo.シミュレーション(業務ID)が同じか否かを判断する
            return HysAddinJobDefineSample.BUSINESS_KIND.Equals(csSimKindID);
        }

        /// <summary><para>method outline:</para> ...
        override public HySViewerJob CreateViewerJob()
        {
            ///.ToDo.独自の・HySViewerJob.派生クラスを生成してください
            return new HsAddinSampleJob();
        }
    }
}

```

インスタンス生成部分

```

namespace CommonMP.HYSSOP.OptionImpl.HSAddinJob.Sample
{
    /// <summary><para>class outline:</para> ...
    public class HysAddinJobDefineSample
    {
        ///.よく使用する定数等を此处で纏めて定義しておく
        /// <summary>業務種別識別子
        /// システム内でユニークな値としてください(メニューに登録します) </summary>
        static public readonly HySKind BUSINESS_KIND
            = new HySObjectKind("Sample.HSAddin.HysAddinJobSampleBusiProc");
        /// <summary>ビューア内処理ファクトリー
        /// システム内でユニークな値としてください</summary>
        static public readonly HySIdentifier FACTORY_ID_VIEWRJOB
            = new HySID("Sample.HsAddinJobSample.Factory.ViewerJob");
    }
}

```

4. 3 サンプルプログラム（ユーザー独自開発機能処理）を
起動するためのモジュールをCommonMPへ登録

CommonMPからユーザーが作成したモジュールをコールするためには 予め、ユーザー提供モジュールの入ったDLLを、CommonMPへ登録する必要があります。 その為には、次の2つの事を行う必要があります。

① CommonMP.dicon ファイルへの DLL情報登録

CommonMPを構成するソフトウェアモジュールは、¥CommonMP¥Execute¥conf¥ 下の CommonMP.dicon ファイル^(注意:1) に登録します。 CommonMP実行体は、図4. 3. 1に示す様に、 起動時に、CommonMP.dicon ファイルを読み込み、中に記述されたDLLを ロードします。

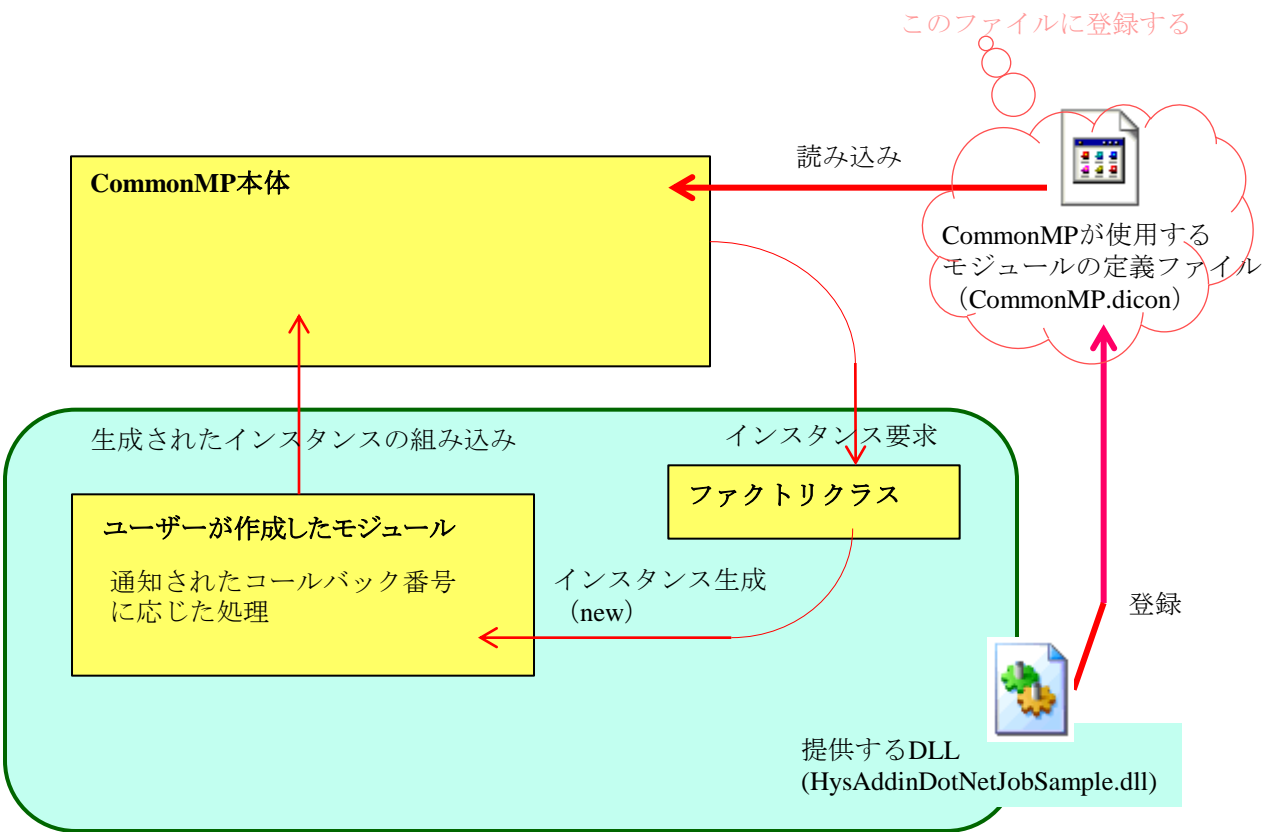


図 4. 3. 1 DLLの登録概念

(注意:1)

¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥ 下の「SysToolTestMain.sln」をから起動して、デバッグを行う場合には、
¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥DeveloperTestMain¥bin¥conf ¥ 下の CommonMP.dicon ファイルにも 登録する必要があります。

CommonMP.dicon ファイルはXML形式となっています。

＜Componentタグ内に、下記属性を記述します

kind = “ViewerJob” :kind 属性は必ず”ViewerJob”として下さい。

package=“ネームスペース名” :package属性は、ファクトリークラスを記述したソースのネームスペース名として下さい

class=“ファクトリクラス名称” :class 属性は、ファクトリクラスの名称を入れて下さい。

dll=“DLL名称” :dll 属性は、提供するDLLの名称を入れて下さい。

サンプルの CommonMP.dicon への登録例を 図4. 3. 2に示します。

CommonMP.dicon ファイル内に追加する内容

```
<component
  kind = "ViewerJob"
  package="CommonMP.HYSSOP.OptionImpl.HSAddinJob.Sample"
  class="HsAddinSampleJobFactory"
  dll="HysAddinDotNetJobSample.dll"
/>
```

必ず”ViewerJob”とする

DLL名称を入れます

開発したソースコード

```
namespace CommonMP.HYSSOP.OptionImpl.HSAddinJob.Sample
{
    /// <summary><para>class outline:</para> ...
    public class HsAddinSampleJobFactory : HysDotNetAddInBaseJobFactory
    {
        /// <summary> ...
        public HsAddinSampleJobFactory()
        {
            //・コンストラクター内で・ファクトリIDを設定する
            //・ToDo・独自のファクトリIDを設定して下さい。
            SetFactoryID(HysAddinJobDefineSample.FACTORY_ID_VIFWRJOB);
        }
    }
}
```

図4. 3. 2 CommonMP.dicon ファイルへの登録例

② DLLファイルのコピー

コンパイルによって作成された DLLファイルを、¥CommonMP¥Execute¥bin 下へコピーします。

4. 4 メニューへの登録

CommonMPのメニュー体系は、CommonMP実行環境下の（例えば、¥CommonMP¥Execute¥）下のconf¥ja¥ 下に Menu.xmlとして 記述されています。
この Menu.xml を変更することで、CommonMPのメニューを変更することが可能です。
図4. 4に 先に作成した独自モジュールを呼び出す様に設定した Menu.xml の例を示します。



図4. 4 Menu.xml ファイルへの登録例

¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥ SysToolTestMain.slnを開いて、サンプルをデバッグ実行するために、

¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥DeveloperTestMain¥bin¥conf¥
下に存在する

CommonMP.dicon

ja¥Menu.xml

に、以上述べてきた設定が行われています。

参照して下さい。

また、変更すべき項目だけを

¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥HSViewer¥AddInDotNetScreen¥

¥AddinDotNetJobSample¥SystemSettingExample

に纏めてあります。

合わせて参照して下さい。

<コールバック番号使用上の注意>

Menu.xmlの コールバック番号は、

<menuitem>

.....

<callback flg="true" no="199877"/>

.....

</menuitem>

コールバック番号

CommonMP本体システム側で使用している番号との競合を防ぐため、100000 より大きな数を使用して下さい。