

ユーザー独自開発画面の CommonMPへの組み込み方法（サンプル）

1. 概要

CommonMPは標準的に評価用の画面（例えば、一般グラフ表示、ハイト／ハイドログラフ表示等）が準備されています。しかしながら、場合によっては、ユーザーが独自に開発した画面を表示したい場合も有ります。

この要求に応えるため、CommonMPでは、ユーザーが固有な画面（以後、独自画面と呼びます）を作成し、その画面をCommonMP上から表示できるフレームワークを準備しています。具体的には、独自画面を組み込むために必要な親クラスを準備しておりますので、ユーザーはそのクラスから派生して 製作する事になります。

ここでは、例として 簡単な画面を作成し、CommonMP上のメニューから呼び出して表示する場合を例に取り、その組み込み方法を説明します。

尚、独自画面の位置組み込み方法を述べる前に、前提となる考え方を理解しておく必要があります。

1) 独自画面は、基本的に 特定の演算要素／モデルに依存した画面表示を目指したのではなく、色々なモデルが出力したファイルデータを 読み込んで表示する汎用的ツール画面として位置づけられます。

2) CommonMPから見て、独自画面は、CommonMPの演算制御等とは無関係なタイミングで動作する、一つの独立した「業務機能」として見なされます。

2. 独自画面の追加手順

2. 1 追加手順概要

独自画面を CommonMP 上のメニューから呼び出して表示するようにするためには、下記の手順で CommonMP への組み込みを行います。

- ①独自画面を及び、独自画面を表示するためのモジュールを作成します。（DLLとして提供）
- ②上記モジュールを CommonMP へ登録します。
- ③Menu.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。

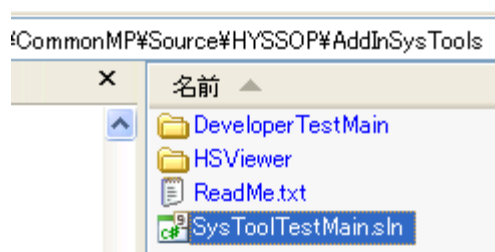
<注意>

基本的に、独自画面と演算モデルを 連動させて動作させることはできません。もし、独自画面に演算結果を表示する場合には、演算終了後に演算の結果として作成されたファイルを読み込む等の操作が必要となります。

2. 2 デイバグ環境

マイクロソフト社 Visual Studio C# 上から、新しく開発する独自画面のデイバグが行える 開発環境を用意しました。

まず、¥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。 このソリューションに 2. 1で述べた 独自画面及びそれを起動するモジュール開発用のソリューションを追加して デイバグ機能を活用しながら、開発を行います。 サンプルとして、HysAddinDotNetScreenSample.csproj が予め登録されています。 以下、本サンプルに従って、説明を行っていきます。



プロジェクトを開く

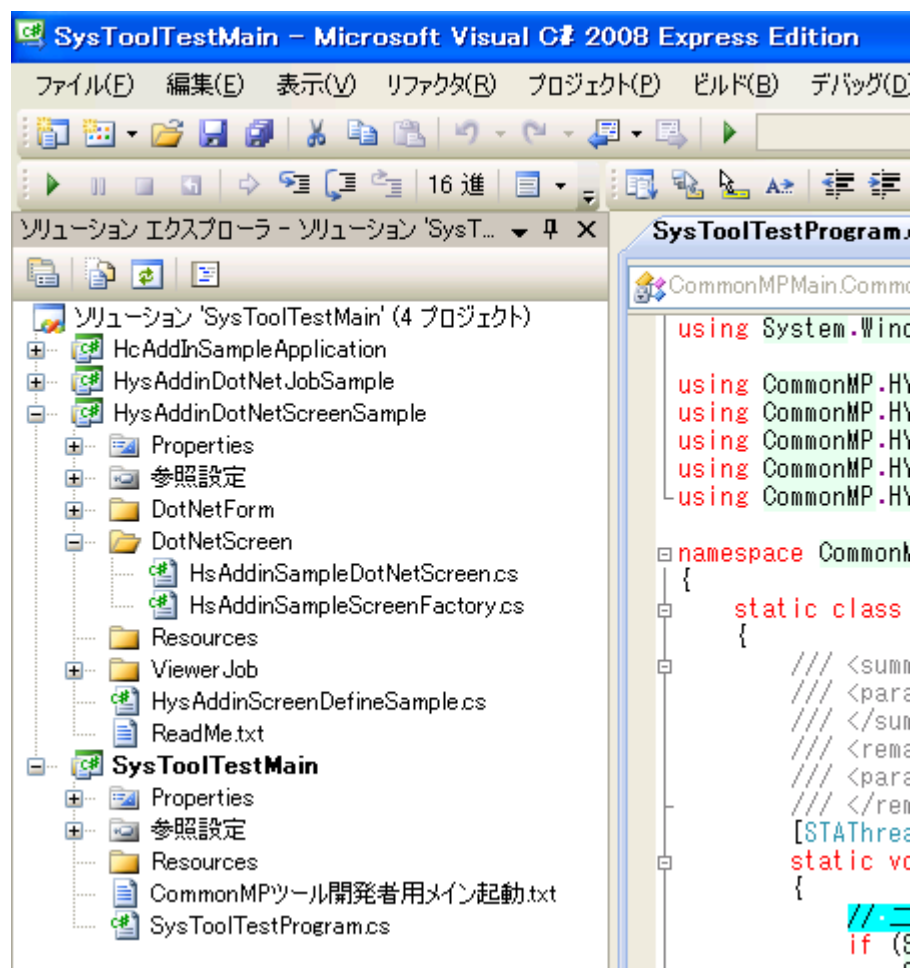


図 2. 1 デイバグ用 ソリューション立ち上げ

3. サンプル機能の概要

まず、サンプルの機能説明を行います。

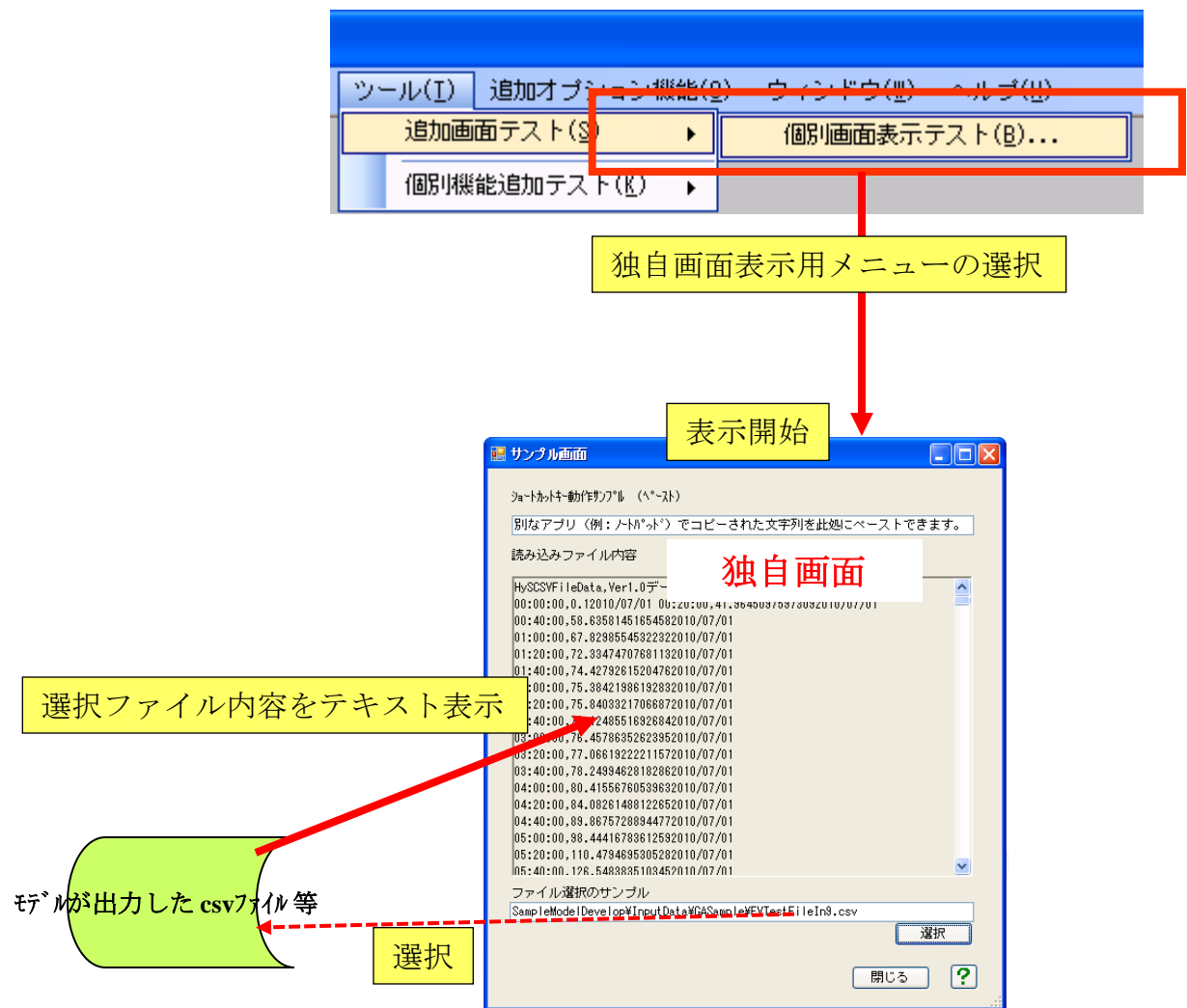


図 2. 2 サンプル機能の概要

図2. 2に示すように、独自画面はメニューの「追加画面テスト」－「個別画面表示テスト」から呼び出されます。ここで、モデルが出力したCSVファイルを読み込み、テキスト表示します。

独自画面作成サンプルを開発する Visual Studio 用のプロジェクトは、
CommonMP¥Source¥HYSSOP¥AddInSysTools¥ HSVIEWER¥AddInDotNetScreen¥
AddinDotNetScreenSample¥ 下に

「HysAddinDotNetScreenSample.csproj」として用意されています。

(前述の テスト用メインに組み込まれています。)

図2. 3に示すように CommonMPへ独自画面を組み込む場合には、図中「 」で囲った部分を作成する必要があります。「表示を制御する為の一種の業務処理」は、複数の画面を持つ事が可能です。但し、ここでの業務処理は、画面からのメニュー入力に限られる為、CommonMP本体の制御を行ったりはできません。

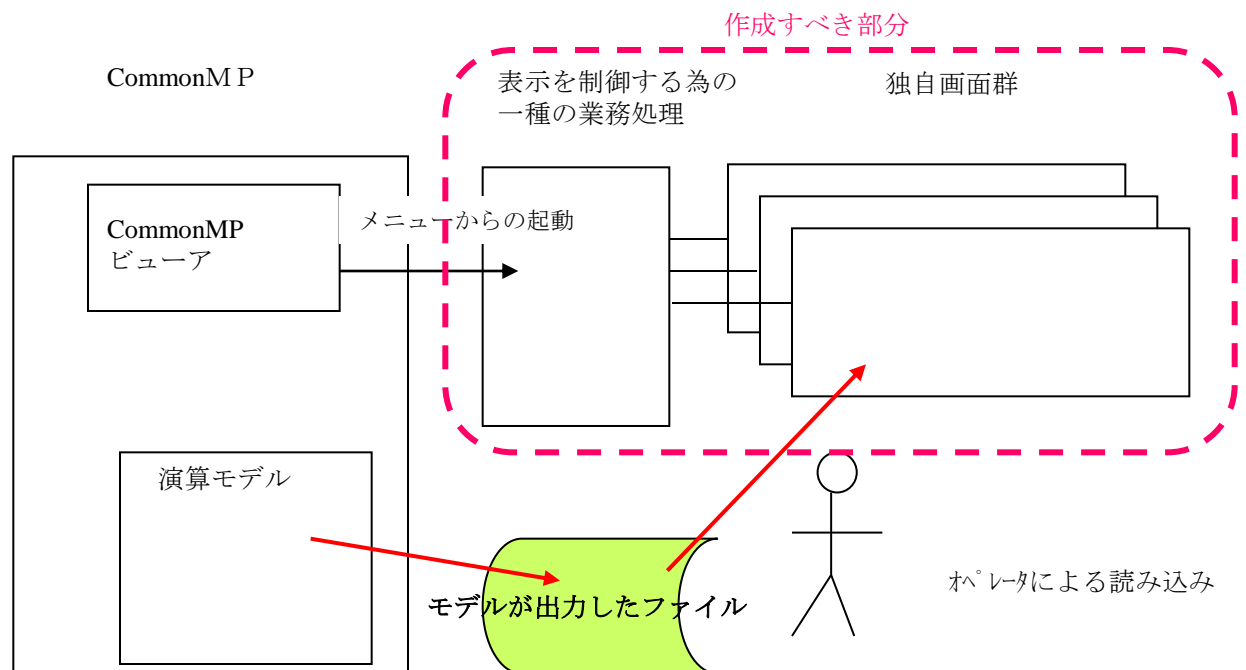


図 2. 3 独自画面組み込みの為作成する部分

3. オペレーター操作と画面表示に至る処理概要

独自画面開発に当たっては、画面のメニューからどのような処理で独自画面が表示されるかを、理解しておく必要があります。図3. 1に オペレーターがメニューを選択してから、独自画面が表示されるまでの、処理概要を示します。

オペレーターが画面メニューを押下すると、押下したイベントがビューア内部(業務)処理に通知されます。ビューア内部処理は、イベントの内容に従って、処理を開始します。画面表示要求であれば、画面生成ファクトリに独自画面を生成させ、画面を表示させます。尚、ビューア内部処理は、ビューアが立ち上がった時に、ビューア内部処理用ファクトリクラスにより生成されます。

図3. 1内に示す 以下のクラスは、一つのDLLとして提供される必要があります。

- ・ビューア内部処理用ファクトリ
- ・ビューア内部処理
- ・画面生成ファクトリ
- ・独自画面
- ・.net ベース画面フォーム

4. 関連クラス体系とモデル開発者が作成するクラス

サンプルとして作成する各クラスの派生関係を 図4. 1に示します。
尚、サンプルのソースは CommonMPをインストールしたディレクトリ下の
¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥HSViewer¥AddInDotNetScreen
¥AddinDotNetScreenSample¥ 下に存在します。

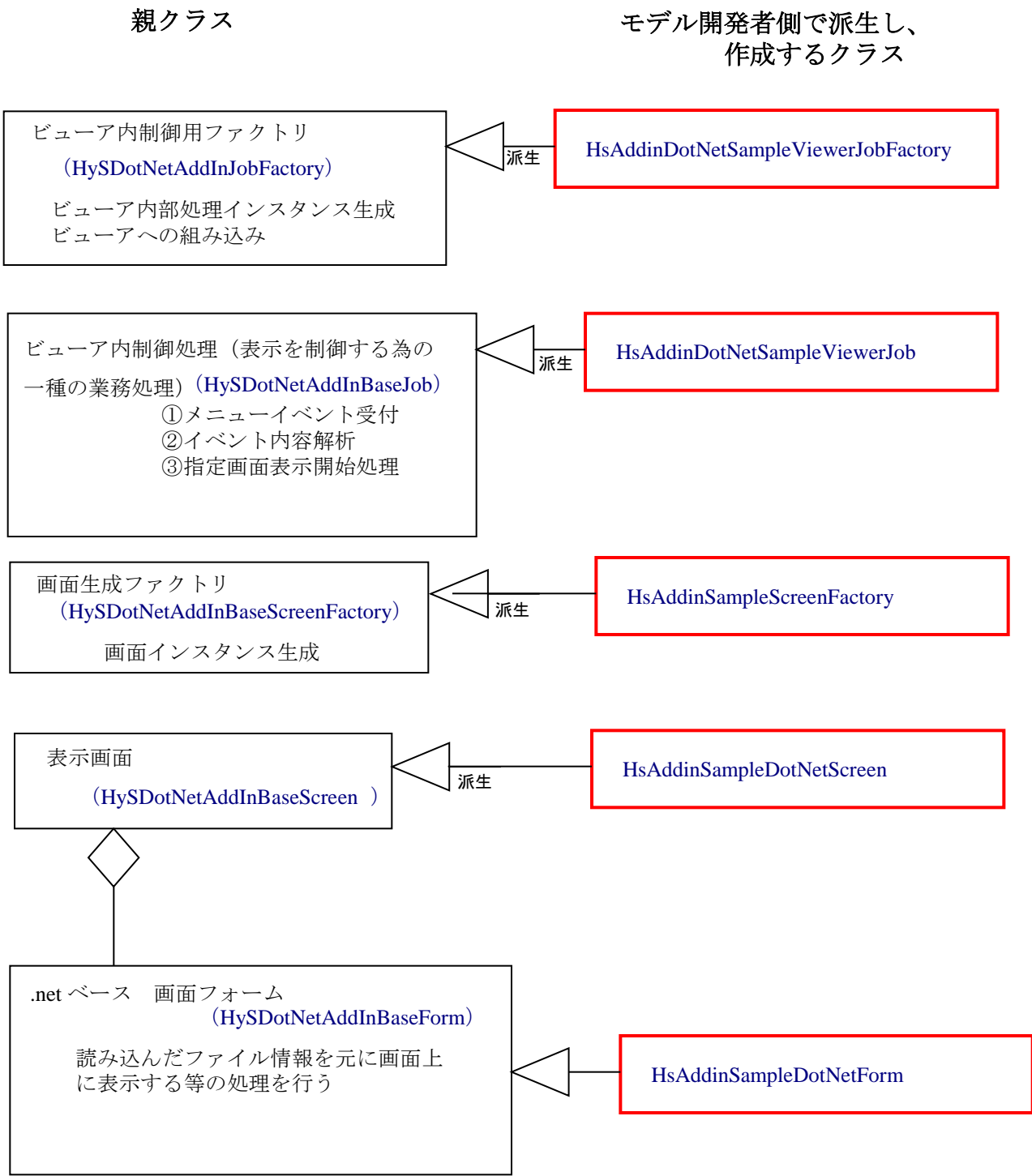


図4. 1 関連するクラスと開発者が作成すべきクラス

以下、クラス毎に、開発者が実装する処理を述べていきます。

4.1 ビューア内部処理用ファクトリ (HsAddinDotNetSampleViewerJob)

ビューア内部処理用ファクトリを用いて、CommonMPは、 独自画面表示用のビューア内処理（表示を制御する一種の業務処理）を システム内に組み込みます。本ファクトリは、CommonMP.dicon 内に定義しておく必要があります。実装すべきメソッドを、図4. 2に示します。

システム起動時

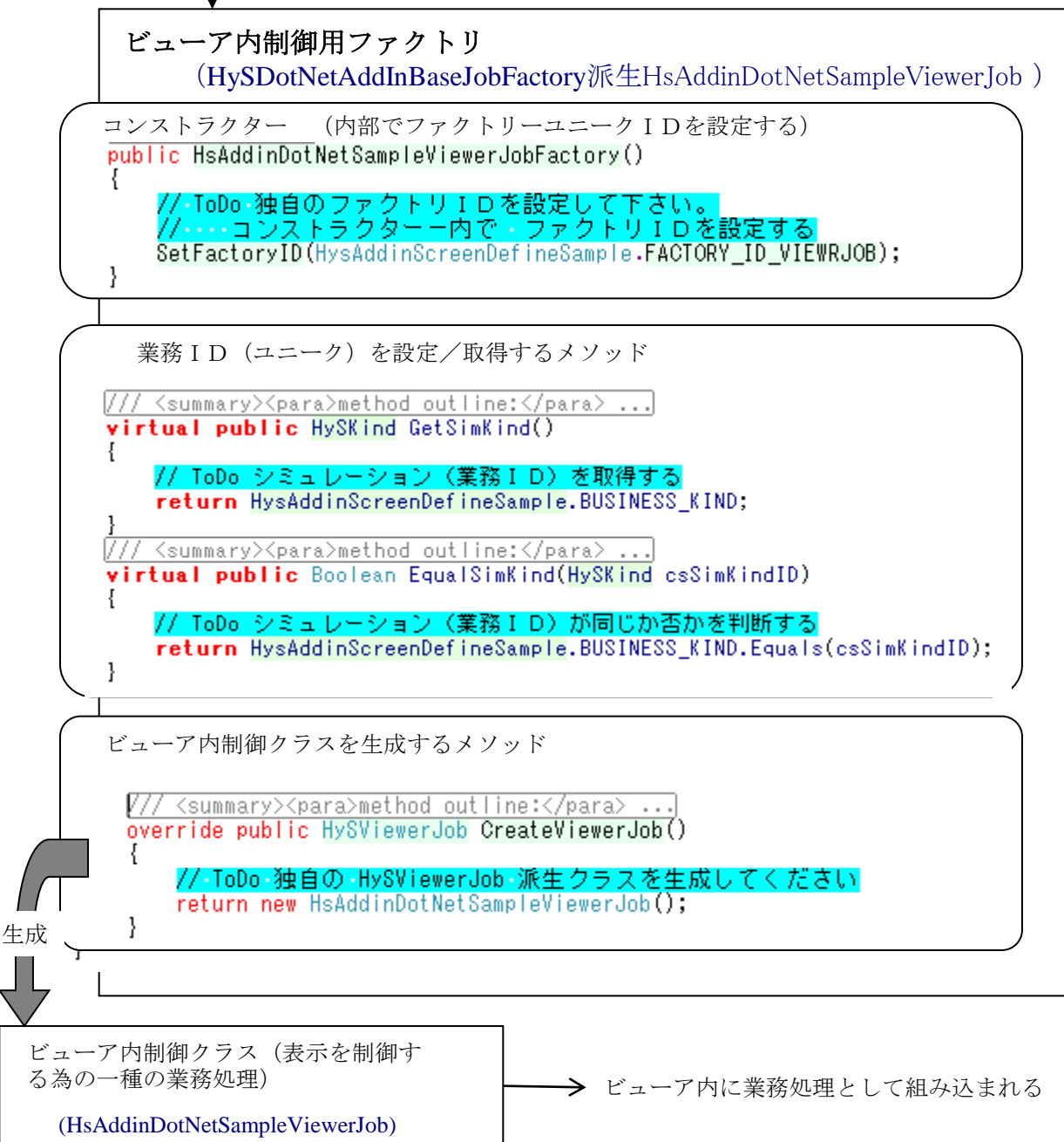


図4. 2 ビューア内部処理用ファクトリで実装すべきメソッド

4.2 ビューア内制御クラス (HsAddinDotNetSampleViewerJob)

実装すべきメソッドを、図4.3に示します。

ビューア内制御

(HySDotNetAddInBaseJob派生HsAddinDotNetSampleViewerJob)

立ち上がり／終了処理群 : ビューア内処理の立ち上がりと終了処理を行います。
特別な事を行わない場合、**return true;** とします。

```
/// <summary><para>method outline:</para> ...  
public override Boolean Initialize()...  
/// <summary><para>method outline:</para> ...  
public override Boolean Terminate()...  
/// <summary><para>method outline:</para> ...  
public override Boolean ExitOK()...
```

メニュー選択イベント

引数で与えられた、コールバック番号によって処理を分岐する

```
/// <summary><para>method outline:</para> ...  
public override long MenuCallBack(long lCallBackNo, Boolean bCheckOnOff)  
{  
    long lRtn = 0;  
  
    if (lCallBackNo == HysAddinScreenDefineSample.SAMPLE_DISP_SCREEN)  
    {  
        // サンプル画面表示開始  
        lRtn = this.DispSomeScreen();  
    }  
    else if (lCallBackNo == HysAddinScreenDefineSample.SAMPLE_DO_SOMETHING)  
    {  
        // 何らかの処理  
        lRtn = this.DoSomething();  
    }  
    else  
    {  
        lRtn = -1;  
    }  
  
    return lRtn;  
}
```

表示処理本体（サンプルでは、本処理を MenuCallBack内に入れず、
別メソッドに分離している。）

```
/// <summary><para>method outline:</para> ...  
protected virtual long DispSomeScreen()  
{  
    HySID csID = null;  
    csID = new HySID("ArbitraryCode"); // 画面にはユニークな値を一意に設定  
  
    // 画面表示  
    HySDotNetAddInBaseScreen csDotNetScreen  
        = this.CmdShowScreen((HySObjectKind)HysAddinScreenDefineSample.SAMPLE_SCREEN, csID)  
        as HySDotNetAddInBaseScreen;  
    if (csDotNetScreen != null)  
    {  
        // 表示に成功ならば  
        csDotNetScreen.ActivateForm(); // 表示した画面を最前面に持ってくる  
    }  
  
    return 0;  
}
```

図4.3 ビューア内制御クラスで実装すべきメソッド

4.3 画面生成ファクトリ (HsAddinSampleScreenFactory)

実装すべきメソッドを、図4.4に示します。

画面生成ファクトリ

(HySDotNetAddInBaseScreenFactory派生HsAddinSampleScreenFactory)

コンストラクタ

```
public HsAddinSampleScreenFactory()
{
    SetFactoryID(HysAddinScreenDefineSample.FACTORY_ID_SCREEN); // ←必須
}
```

業務ID (ユニーク) を設定/取得するメソッド

```
/// <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 HysAddinScreenDefineSample.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 HysAddinScreenDefineSample.BUSINESS_KIND.Equals(csSimKindID);
}
```

画面を生成するメソッド

```
/// <summary><para>method outline:</para> ...
virtual public HySScreen CreateScreen(HySKind csSimulatorKind,
                                       HyIdentifier csID, HySKind csScreenKind)
{
    HySScreen csScreen = null;
    if (csSimulatorKind.Equals(HysAddinScreenDefineSample.BUSINESS_KIND) == true)
    {
        if (HysAddinScreenDefineSample.SAMPLE_SCREEN.Equals(csScreenKind) == true)
        { // SAMPLE_SCREENを作成する必要があるならば、
          csScreen = new HsAddinSampleDotNetScreen();
        }
        //else-if (HysAddinScreenDefineSample.XXXXXX_SCREEN.Equals(csScreenKind) == true)
        //{//別指定する_SCREENを作成する必要があるならば
        //}
        else
        {
            サンプルでは、1種類の画面しか対応していないが、複数の画面を
            生成する事も可能。
        }
    }
    return csScreen;
}
```

図4.4 画面生成ファクトリで実装すべきメソッド

4.4 表示画面 (HsAddinSampleDotNetScreen)

実装すべきメソッドを、図4.5に示します。

表示画面

(HySDotNetAddInBaseScreen派生HsAddinSampleDotNetScreen)

初期化／終了メソッド

```

/// <summary> ...
public override void Initialize()
{
    //>>>> ●必須● Formを生成する ●必須●>>>>
    m_csSampleForm = new HsAddinSampleDotNetForm(this);
    base.SetViewForm(m_csSampleForm);

    // 画面サイズ(もしも、プログラムから変更したい場合)
    //m_csSampleForm.Width = DEFAULT_WIDTH;
    //m_csSampleForm.Height = DEFAULT_HEIGHT;

    // 画面表示位置
    m_csSampleForm.SetDesktopLocation(350, 40);

    // 各種設定
    m_csSampleForm.WindowState = FormWindowState.Normal;
    m_csSampleForm.StartPosition = FormStartPosition.Manual;
    m_csSampleForm.Text = "サンプル画面";

    //>>>> ●必須● 親クラスにFormを設定する ●必須●>>>>
    base.SetViewForm(m_csSampleForm);
    ////////////////////////////////////////
}
/// <summary> ...
public override void Terminate()...

```

ビューア設定メソッド

```

/// <summary> ...
public override void SetViewer(HySViewer csViewer)
{
    //>>>>> ●必須● ・オーバーライド下場合には ・親クラス処理をコール ・●必須●>>>>>
    base.SetViewer(csViewer);

    //便利の為、Jobクラスをキャストしておく
    m_csSampleJob = m_csVJob as HsAddinDotNetSampleViewerJob;

    //必要ならば ・業務固有の独自処理を行う

    //独自処理が無くて、且つ ・Jobをキャストしておく必要が無いならば、本メソッドは不要
}

```

ビューアを設定するタイミングで、Jobクラスのインスタンスが確定される。Jobを意識する必要がなければ、本メソッドの実装は不要。

図4.5 表示画面で実装すべきメソッド

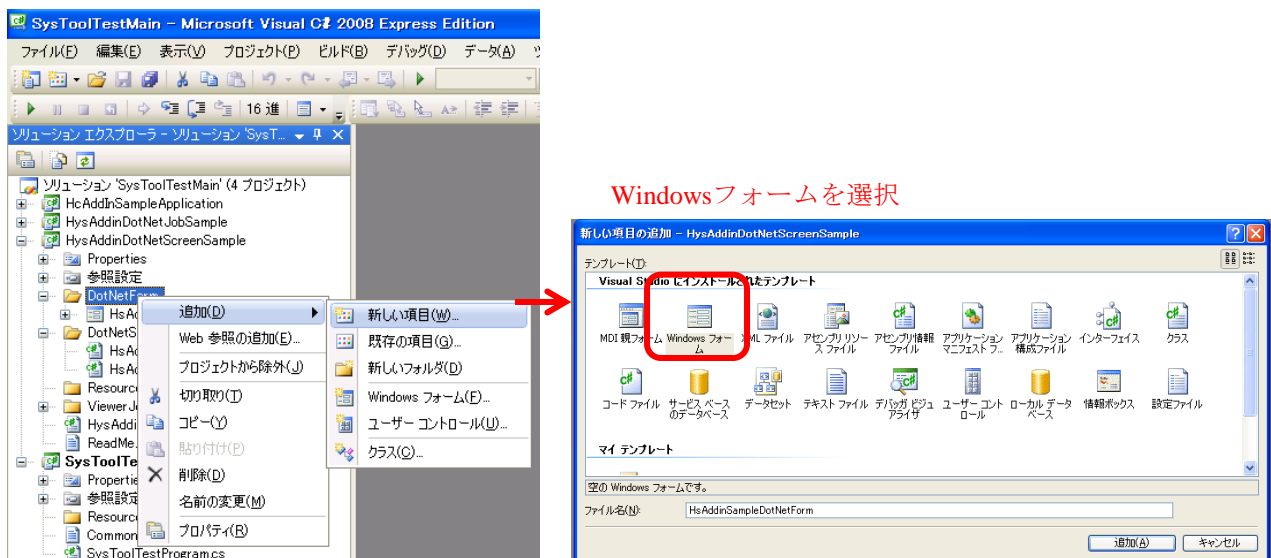
4.5 .net ベース 画面フォーム (HsAddinSampleDotNetForm)

フォームは、マイクロソフト社 Visual Studio 2008 C# を用いて、新規作成します。

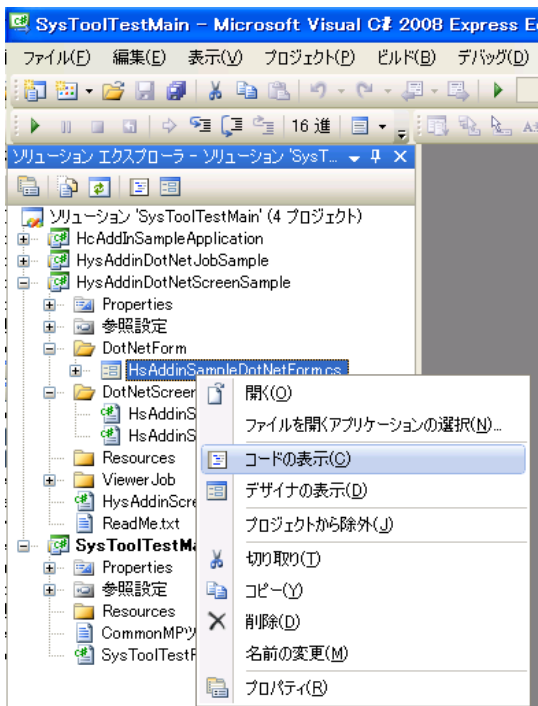
図 4. 6 に示すように、Visual Studio のポップアップメニューの「追加」→「新しい項目」を選択し、新しい項目追加のダイアログを開きます。ここで、Windows フォームを選択して 新しく作成するフォーム名称を入力し 「追加」ボタンを押下します。すると、プロジェクトにフォームが生成されます。

ここで作成された Windows フォームは、CommonMP に対応したフォームではない為、ソースコードを開き、派生関係等を修正します。

4. 1～4. 4 で述べた各クラスは、.net 上で作成した画面を CommonMP へ組み込むためのフレームワークを動作させるために存在し、実際の画面の主な機能は、.net ベース画面フォームにコーディングされます。



(1) 空のフォームを作成する



(2) 作成したフォームのコードを編集する

図4. 6 Windowsフォームの新規作成

図4. 7に コードの修正例を示します。 この例では、 クラスの派生関係を変更し、引数付きのコンストラクターを追加しています。

また、フォームクラスは必ず、画面イベントとして 画面のアクティブ化、非アクティブ化イベント時に動作するメソッドを実装する必要があります。

図4. 8に示すように 画面全体のプロパティ画面から そのメソッドを設定できます。 この時、作成したメソッド内で、必ず親クラスのイベントメソッドをコールして下さい。(図4. 8内ソード参照)

図4. 9に .net ベース画面フォームクラスが実装すべきメソッドを示します。

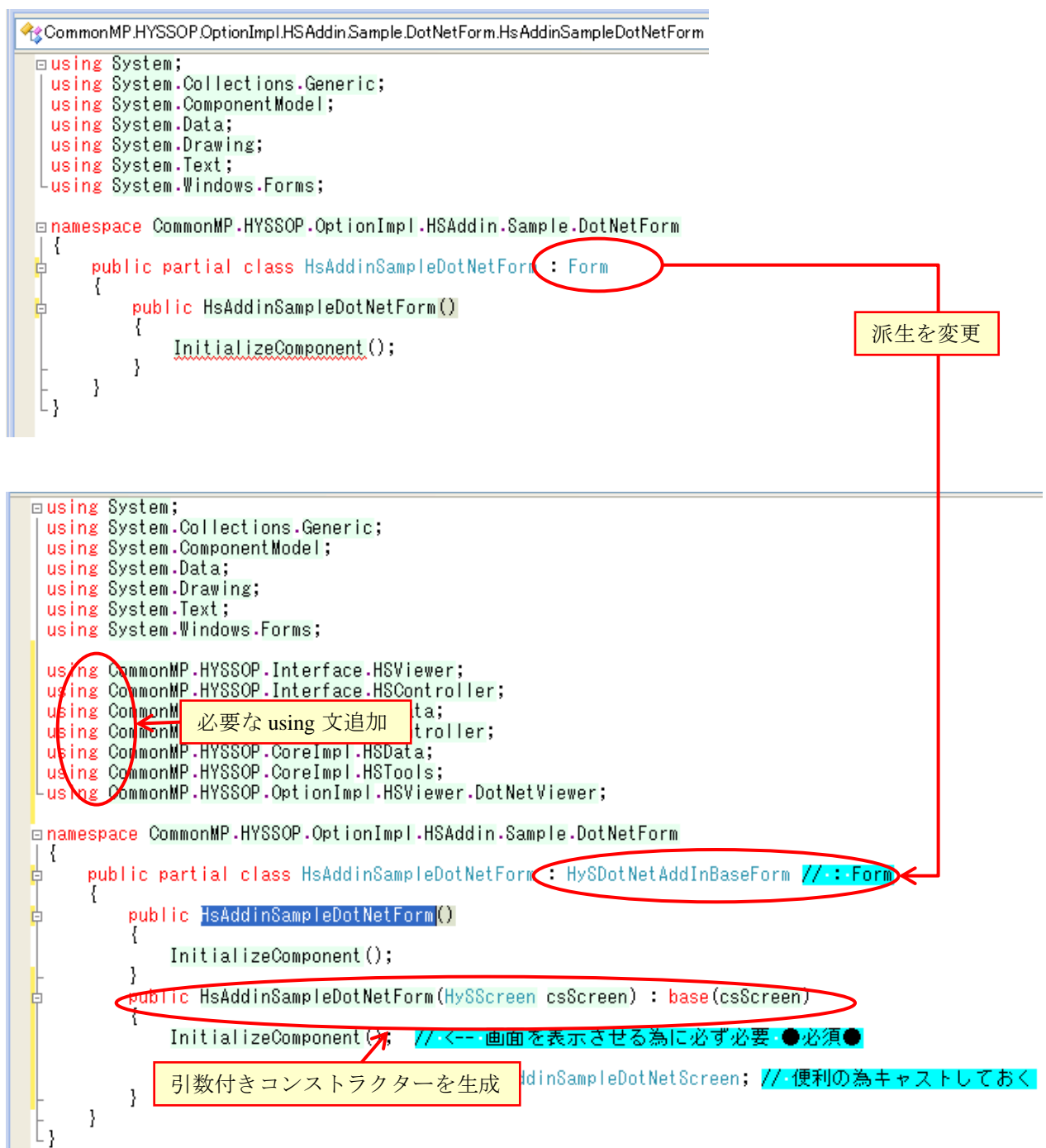


図4. 7 コードの修正例

4.5 .net ベース 表示画面フォーム (HsAddinSampleDotNetForm)

実装すべきメソッドを、図4. 4に示します。

.netベース表示画面フォーム

(HySDotNetAddInBaseForm派生 HsAddinSampleDotNetForm)

引数付きコンストラクタ

```
/// <summary><para>method outline:</para> ...  
public HsAddinSampleDotNetForm(HySScreen csScreen) : base(csScreen)  
{  
    InitializeComponent(); //<-- 画面を表示させる為に必ず必要・●必須●  
    m_csSampleScreen = csScreen as HsAddinSampleDotNetScreen; //・便利の為  
}
```

画面のアクティブ化、非アクティブ化に対応するイベントメソッド

```
/// <summary> ...  
private void HsAddinSampleDotNetForm_Activated(object sender, EventArgs e)...  
/// <summary> ...  
private void HsAddinSampleDotNetForm_Deactivate(object sender, EventArgs e)...
```

画面のカットアンドペースト等に対応するイベントを処理するメソッド

```
/// <summary><para>method outline:</para> ...  
public override void NoticeUndoEvent()...  
/// <summary><para>method outline:</para> ...  
public override void NoticeRedoEvent()...  
/// <summary><para>method outline:</para> ...  
public override void NoticeCutEvent()...  
/// <summary><para>method outline:</para> ...  
public override void NoticeCopyEvent()...  
/// <summary><para>method outline:</para> ...  
public override void NoticePasteEvent()...  
/// <summary><para>method outline:</para> ...  
public override void NoticeDeleteEvent()...  
/// <summary><para>method outline:</para> ...  
public override void NoticeAllSelectEvent()...
```

カットアンドペースト等に対応していなければ、何もしないで
return を行う。

図4. 9 .net ベース画面フォームで実装すべきメソッド

5. 独自画面サンプルの CommonMPへの登録

CommonMPからユーザーが作成したDLLは下記の手順で、CommonMPへ登録します。

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5. 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。

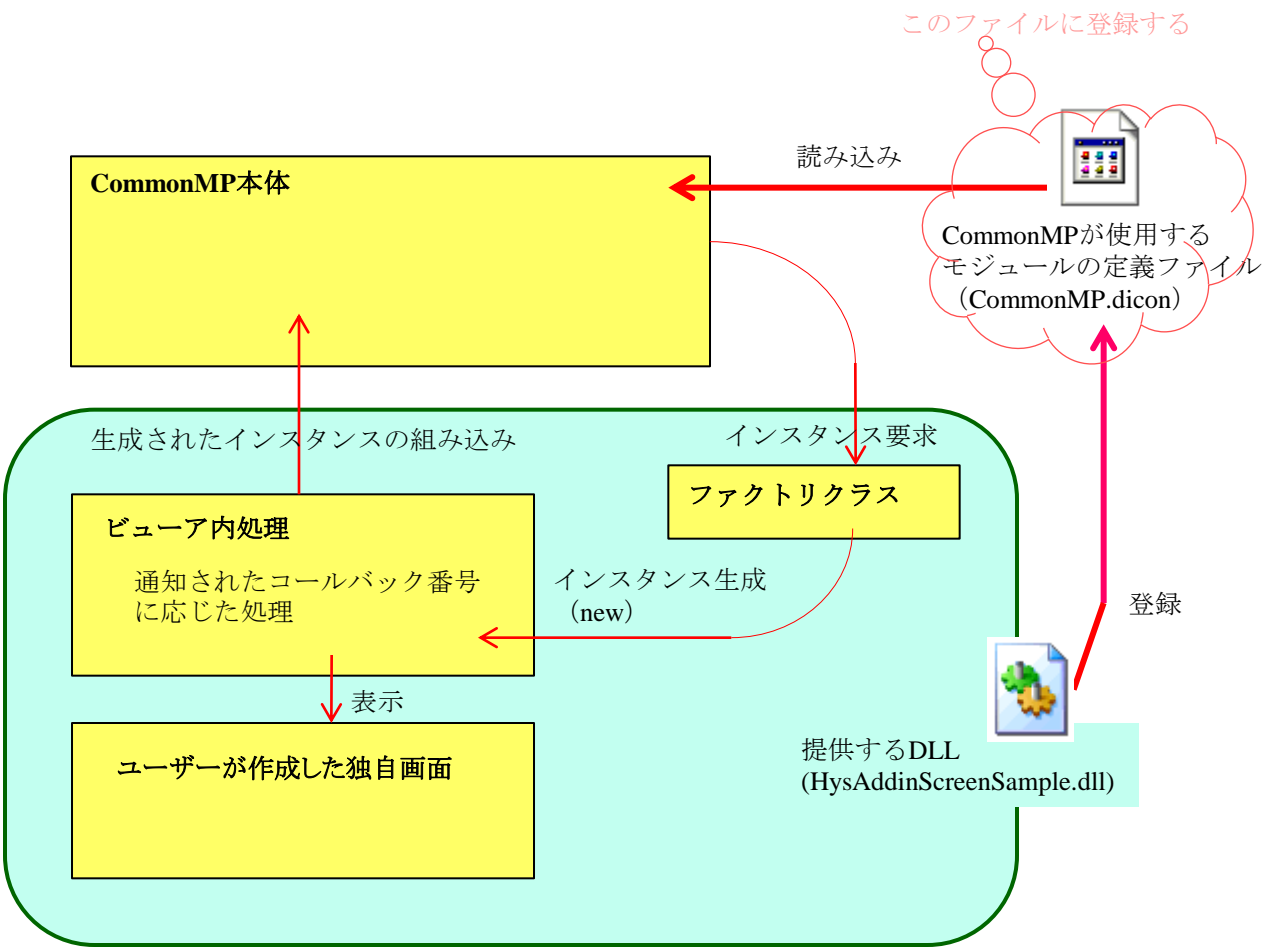


図 5. 1 DLLの登録概念

(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。ここに次の2つのファクトリークラスを登録します。

(1)ビューア内処理用ファクトリクラスの登録

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、ビューア内処理用ファクトリークラスを記述した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。

(2)スクリーンファクトリクラスの登録

Component タグ内に下記属性を記述します
kind="ViewerJob" :kind 属性は必ず "Screen" として下さい。
package="ネームスペース名" :package属性は、スクリーンファクトリークラスを記述した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。

CommonMP.dicon ファイル内に追加する内容の例

```
<component
  kind = "ViewerJob"
  package="CommonMP.HYSSOP.OptionImpl.HSAddin.Sample"
  class="HsAddinDotNetSampleViewerJobFactory"
  dll="HysAddinScreenSample.dll"/>
```

```
<component
  kind = "Screen"
  package="CommonMP.HYSSOP.OptionImpl.HSAddin.Sample"
  class="HsAddinSampleScreenFactory"
  dll="HysAddinScreenSample.dll"/>
```

必ず"Screen"とする

DLL名称

開発したソースコード

```
namespace CommonMP.HYSSOP.OptionImpl.HSAddin.Sample
{
    /// <summary> ...
    public class HsAddinSampleScreenFactory : HySDotNetAddInBaseScreenFactory
    {
```

図 5. 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。

6. メニューへの組み込み

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6. 1に Menu.xml の設定例を示します。



図6. 1 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¥

¥AddinDotNetScreenSample¥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。

7. 補足

(1) 国際対応

CommoMP本体は、国際対応として、言語に関係する外部定義ファイル（メニュー定義ファイル等）をデフォルト用とローカル言語用とで格納位置を別けて管理しています。デフォルトは英語です。

日本語は、ローカル言語としての扱いとなり、そのため、日本語OSでCommonMPを実行した場合は、ローカル定義フォルダ“ja”（日本語用）の定義ファイルが読み込まれ実行されます。定義ファイルの変更時は、動作環境をご確認の上、対応するフォルダ内の定義ファイルを修正して下さい。

例) メニュー定義ファイルの格納位置

¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥DeveloperTestMain¥bin¥conf¥
下に存在するファイル

Menu.xml	・・・	デフォルト用メニュー定義ファイル（英語）
ja¥Menu.xml	・・・	日本語用メニュー定義ファイル

※) 日本語OSでの実行時は、有効となる定義ファイルja¥Menu.xmlを修正する。