株式会社フレクシェ

FLEXSCHE
スクリプティング勉強会

2008年11月14日

株式会社フレクシェ
浦野幹夫　先田力哉

本日のプログラム

13:30	FLEXSCHEにおけるアドイン開発の概要	10分	浦野
13:40	スクリプティングとWSCGenの解説	10分	浦野
13:50	FLEXSCHEのアドインコーディングの基礎	50分	浦野
14:40	休憩		
14:50	FLEXSCHEスクリプティングフォームの紹介	60分	浦野
15:50	外部メソッドと外部ファンクション	30分	先田
16:20	休憩		
16:30	グループ演習・質疑応答・ディスカッション		
18:00	懇親会		

本日の進め方

- 一緒にPCを操作しながらお聞きください
- 質問がある方は
  前を向いて手を上げてください
- サポートが必要な方は
  後ろを向いて手を上げてください

随時対応いたします

- お互いに協力しあいながら理解を深めてください

株式会社フレクシェ

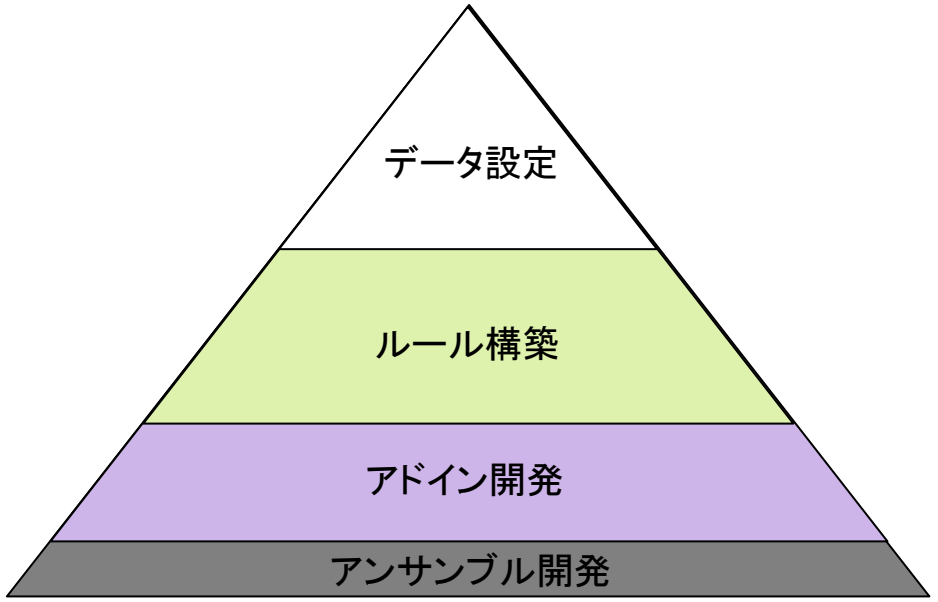
FLEXSCHEにおける アドイン開発の概要

FLEXSCHEのアドイン開発

- 「柔軟性」の要因のひとつ
  - より使いやすく
  - リスク回避の手段
- さまざまな言語・開発環境で作成
  - Visual C++
    高速＆なんでもできる
  - Visual Basic
    親和性が高い
  - C#.Net / VB.Net
    新技術との適応性
  - スクリプト(VBScript / JScript)
    お手軽

スクリプティングは
アドイン開発の方法のひとつ

モデルの柔軟性
ルールの柔軟性
開発の柔軟性



FLEXSCHEの「柔軟性」を支えるヒエラルキー

アドインの種類

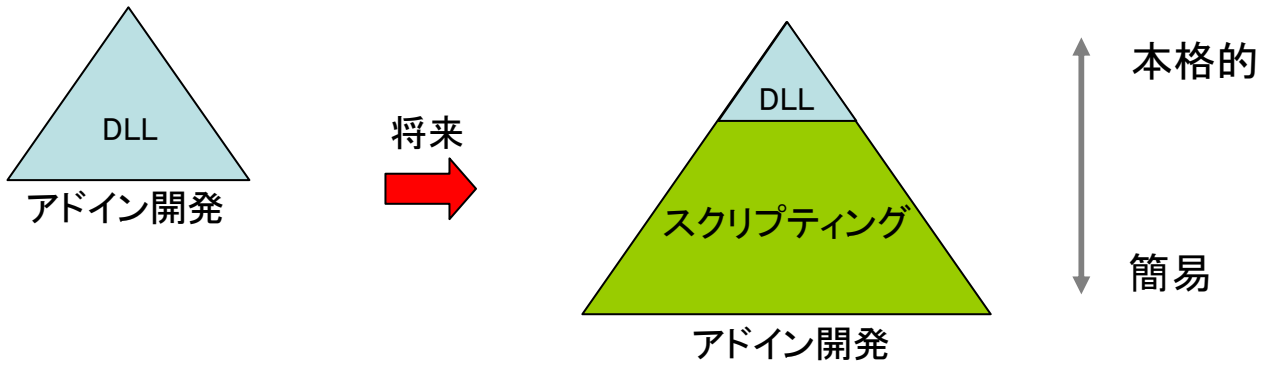
- アドイン
  - GUIからさまざまなタイミングで呼び出す
- 外部メソッド
  - スケジューリングルールへ組み込む
- 外部ファンクション
  - takt計算式から呼び出す
- アンサンブル
  - アドインの集まりで統合された機能体系を実現
    例）GPアンサンブル

株式会社フレクシェ

スクリプティングと WSCGenの解説

FLEXSCHEにおけるスクリプティング

- WSC(Windows Script Components)を利用して
  テキストファイルでCOMモジュールを実装
- AppDataフォルダに配置
- VBScriptとJScript
- スクリプト雛形生成ツール(WSCGen)を利用することで
  記述量を最低限に



スクリプティングの長所と短所

長所

お手軽

- ●その場でインクリメンタルなコーディングができる
  （コンパイルやFLEXSCHEの再起動が不要）
- ●特別な開発環境を必要としない
- ●WSCGenを利用することで、効率よく開発できる
- ●FLEXSCHEスクリプティングフォーム(FSF)によりフォームを手早く作成できる

短所

大規模開発に不向き

- ●インテリセンス機能を利用できない
- ●実行速度が比較的遅く、大量データの処理には向かない
- ●本格的な開発で求められるライブラリなどを利用できない
- ●型チェックが無いのでランタイムエラーが発生しやすい

VBScriptとJScriptの違い

	VBScript	JScript

コード記述	Function Foo(Arg1,Arg2)
	  …
	  Foo = <retval>
	End Function
	
	Sub Foo(Arg1,Arg2)
	  …
	End Sub
		function Foo(arg1,arg2)
		{
		  …;
		  return <retval>;
		}
		
		function Foo(arg1,arg2)
		{
		  …;
		}

メッセージ表示	MsgBox message	alert(message);
代入	<variable> = <value> set <var> = <object>	<var> = <value/object>;

※WSCGenの利用を前提として

VBScriptの注意事項

if文中の条件式はすべて評価される

言語仕様のクセが強い

誤）
```
if opRec.IsBound and opRec.Frozen then
  …
end if
```

正）
```
if opRec.IsBound then
  if opRec.Frozen then
    …
  end if
end if
```

Function/Subは戻り値受け取りの有無によって呼び出し方が異なる

```
v = Foo( 1, 2 )
if Foo( 1, 2 ) then …
Foo 1, 2
```

※あるいはcall文を使う

ステートメントの途中で改行するときは行末に'_'を

```
form.AddControl "btnAdd", "button:追加(&A)", _
  "align_right:listA/width"
```

JScriptの注意事項

処理の最後にApplication/Projectの解放とガベージコレクションが必要

```
project = null;
CollectGarbage();
```

※ ただしWSCGen利用時はフレームワーク側で隠蔽するのでケア不要

日時(DATE)型を直接演算できないので、明示的変換が必要

```
ISDLUtility.TimeToMS( time )
ISDLUtility.TimeFromMS( ms )
```

※ } DATE値 ⇔ Dateオブジェクト

```
VBDateFromJDate( jsdate )
VBDateToJDate( time )
```

} DATE値 ⇔ ミリセカンド値

※ WSCGenが生成する雛形では大域変数sdlutilが定義済みなので(後述)、実際には

```
sdlutil.TimeToMS( time )
sdlutil.TimeFromMS( ms )
```

と表記すればよい。

親和性が低い

存在しないプロパティの取得がエラーとして検出されない

```
project.NotExistingMethod();          エラーが指摘される
project.NotExistingProperty = v;      エラーが指摘される
v = project.NotExistingProperty;      エラーが指摘されない
```

JScriptでの日時の扱い

- FLEXSCHEから取得した日時(Date値)を処理するためにはDateオブジェクトに変換

```
var v = VBDateToJDate( project.BasisTime );
v.setMonth( v.getMonth() + 15 );
var ctrl = form.AddControl( "dt", "datetime" );
ctrl.Value = VBDateFromJDate( v );
```

15ヶ月進める

FSFの日時コントロール

- 比較するだけならばミリセカンド値に変換するのが妥当

```
ms1 = sdlutil.TimeToMS( project.SchedulingStartTime );
ms2 = sdlutil.TimeToMS( opRec.ManufactureStartTime );
if( ms1 > ms2 ) {…}
```

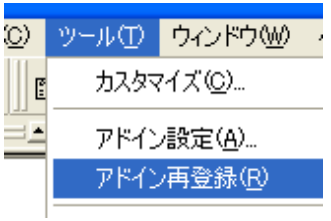
- 変数で一時的にそっと保持するだけであれば変換は不要

```
var v = project.BasisTime;
var ctrl = form.AddControl( "dt", "datetime" );
ctrl.Value = v;
```

※ `alert(v)` もOK

スクリプティング共通の注意事項

- スクリプトをテキストエディタ上で変更すれば、速やかに挙動に反映される

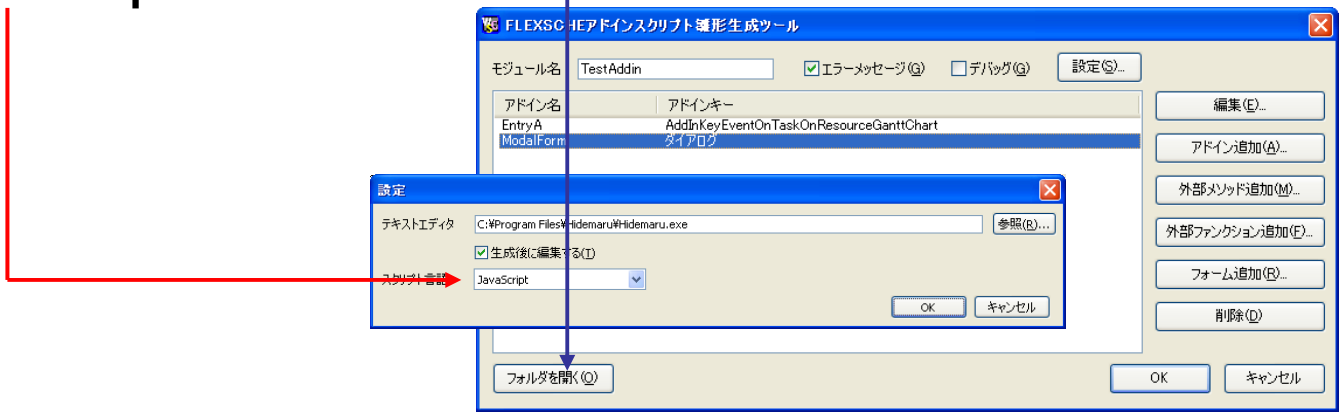


例外）

  - SelfRegistration内を変更した場合は「アドイン再登録」を実行しなくてはならない。
  - 非debugモードのFSFモードレスフォームのハンドラの変更はリビルドボタンを押さないと反映されない

スクリプト雛形生成ツール(WSCGen)

- VBScriptまたはJScriptのスクリプトの雛形
  - アドイン
  - 外部メソッド
  - 外部ファンクション
  - アドインから利用するフォーム(FSF)
- AppDataフォルダに生成
- VBScriptとJScriptに対応



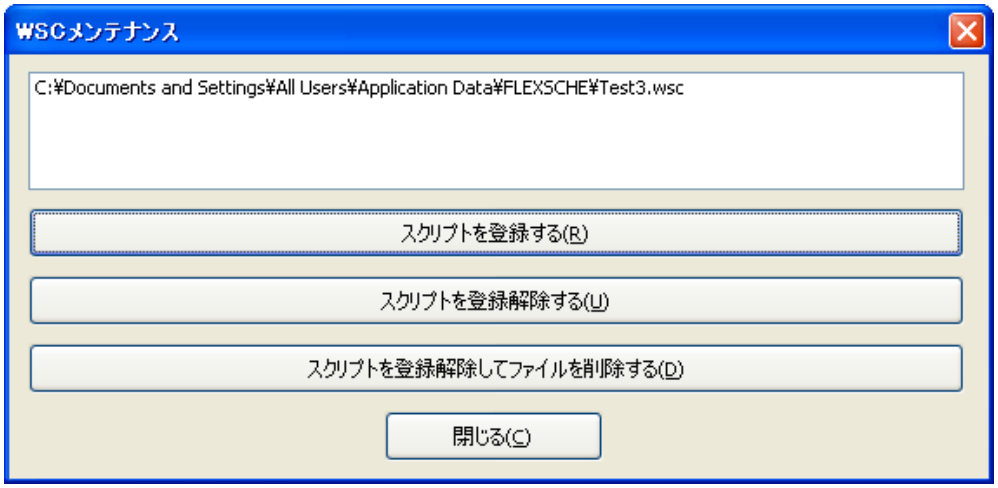
スクリプト雛形の構造

コーディング作業の大部分は実装部分の編集

実際に生成して編集してみます

WSCGenのWSC登録機能

- 既存のWSCファイルの配置作業に利用（特にVista）
- アイコンまたはウィンドウに対象ファイルをドロップ



WSCGen利用時の注意事項

- 不要なスクリプトファイルの削除時には必ず登録解除
  - さもないとレジストリがじわじわと肥大化
  - 上書きはOK
- こまめにバックアップをとる

FLEXSCHEへのアクセス

Core

定数の前に”SData.”を付加

例）SData.SDRStatusManufactureStarted

GUI

定数の前に”FLEXSCHE.”を付加

例）FLEXSCHE. FSVTypeTimeChart

SDLib

定数の前に”SDLib.”を付加
SDLUtilityは大域変数”sdlutil”にインスタンス生成済み

例）sdlutil.KeyState(SDLib. SDLVKControl)

スクリプトのデバッグ

- エラーメッセージ
  - 行番号によるエラー発生位置の特定（「エラーメッセージ」をチェック）
- メッセージボックスの利用
  - “MsgBox” (VBScript) / “alert” (JScript)
- メッセージパネルの利用
  - env.MessagePanel.AddLine <メッセージ>
- デバッガによるトレースと値の参照
  - MS Script Editor (MS-Office付属) / MS Visual Studio
  - デバッグ開始 “Stop” (VBScript) / “debugger” (JScript)

株式会社フレクシェ

FLEXSCHEの
アドインコーディングの基礎

アドインキー一覧

AddInKeyMenuExternalDS
AddInKeyMenuView
AddInKeyMenuEdit
AddInKeyMenuSchedule
AddInKeyMenuTimeChart
AddInKeyMenuCtrlView
AddInKeyMenuTool
AddInKeyMenuWindow
AddInKeyMenuHelp
AddInKeyMenuSequenceChart

AddInKeyHookStartup
AddInKeyHookAfterLoadingProject
AddInKeyHookBeforeSavingProject
AddInKeyHookAddInsRegistered
AddInKeyHookAfterSavingProject
AddInKeyHookBeforeLoadingData
AddInKeyHookAfterLoadingData
AddInKeyHookBeforeSavingData
AddInKeyHookAfterSavingData
AddInKeyHookBeforeClosingProject
AddInKeyHookAfterClosingProject
AddInKeyHookTerminate

AddInKeyHookLoadingCtrlView
AddInKeyHookLoadingCtrlPanel
AddInKeyHookSavingCtrlView
AddInKeyHookSavingCtrlPanel
AddInKeyHookLoadingCtrlPane
AddInKeyHookSavingCtrlPane
AddInKeyHookClosingTimeChart
AddInKeyHookClosingCtrlView

AddInKeyEnsembleSetup
AddInKeyEnsembleLoadData
AddInKeyEnsembleSaveData
AddInKeyEnsembleGenerateOperations
AddInKeyEnsembleReschedule
AddInKeyEnsembleUnassignOperations
AddInKeyEnsembleEditSettings
AddInKeyEnsembleLoadSettings
AddInKeyEnsembleSaveSettings
AddInKeyEnsembleVerifySchedule
AddInKeyEnsembleCreateNewProject
AddInKeyEnsembleTerminate

AddInKeyHookCommand
AddInKeyHookAfterCommand

AddInKeyHookBeforeMovingOperations
AddInKeyHookAfterMovingOperations

AddInKeyHookBeforeMovingOperation
AddInKeyHookBeforeEveryMovingOperation
AddInKeyHookAfterEveryMovingOperation
AddInKeyHookAfterMovingOperation
AddInKeyHookAfterLevelingOperations
AddInKeyHookAfterMovingOperation2
AddInKeyHookBeforeMovingFreeCalendar
AddInKeyHookBeforeEveryMovingFreeCalendar
AddInKeyHookAfterEveryMovingFreeCalendar
AddInKeyHookAfterMovingFreeCalendar

AddInKeyHookBeforeResizingOperation
AddInKeyHookBeforeEveryResizingOperation
AddInKeyHookAfterEveryResizingOperation
AddInKeyHookAfterResizingOperation

AddInKeyEventOnTaskOnResourceGanttChart
AddInKeyEventOnResourceOnResourceGanttChart
AddInKeyEventOnItemOnInventoryChart
AddInKeyEventOnResourceOnLoadChart
AddInKeyEventOnOrderOnOrderGanttChart
AddInKeyEventOnOperationOnOrderGanttChart
AddInKeyEventOnBodyOnResourceGanttChart
AddInKeyEventOnBodyOnInventoryChart
AddInKeyEventOnBodyOnLoadChart
AddInKeyEventOnFreeCalendarOnResourceGanttChart
AddInKeyEventOnCornerOnTimeChart
AddInKeyEventOnTopOnTimeChart
AddInKeyEventOnBodyOnOrderGanttChart
AddInKeyEventOnTaskOnSequenceChart
AddInKeyEventOnCornerOnSequenceChart

AddInKeyEventCommandHasBeenExecuted
AddInKeyEventDropFile
AddInKeyEventKeyDown
AddInKeyEventTimer
AddInKeyDispatchMessage
AddInKeyEventTimeChartSettingsUpdated
AddInKeyEventFSEventFired
AddInKeyEventMessageToProject

AddInKeyEventOnRecordOnTreePanel
AddInKeyEventOnSnapshotOnTreePanel
AddInKeyEventOnRecordOnTableLeft
AddInKeyEventOnDataSetOnTreePanel
AddInKeyEventOnOperationOnOperationPanel

AddInKeyEventOnOperationOnOperationViewer_GUIPlus

具体的な説明は開発者マニュアルの「アドインキー」にあります

アドインキーを使い分けを
理解することが重要

メニューへのアドイン

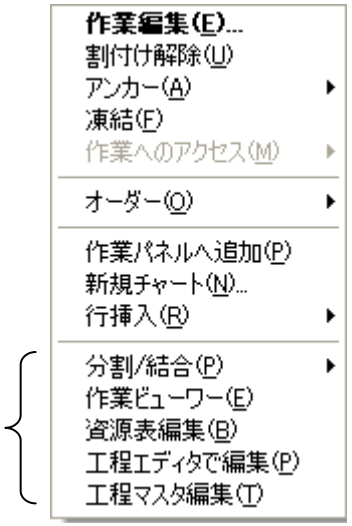
- メニューバー

  メニューアドイン　AddInKeyMenu***

- ポップアップメニュー

  一部のイベントアドインのサブキーとして

  AddInSubKeyEvent_ContextMenuを指定



最下セパレータの下にアドインのメニュー項目

addIn.MenuStringに与えた文字列

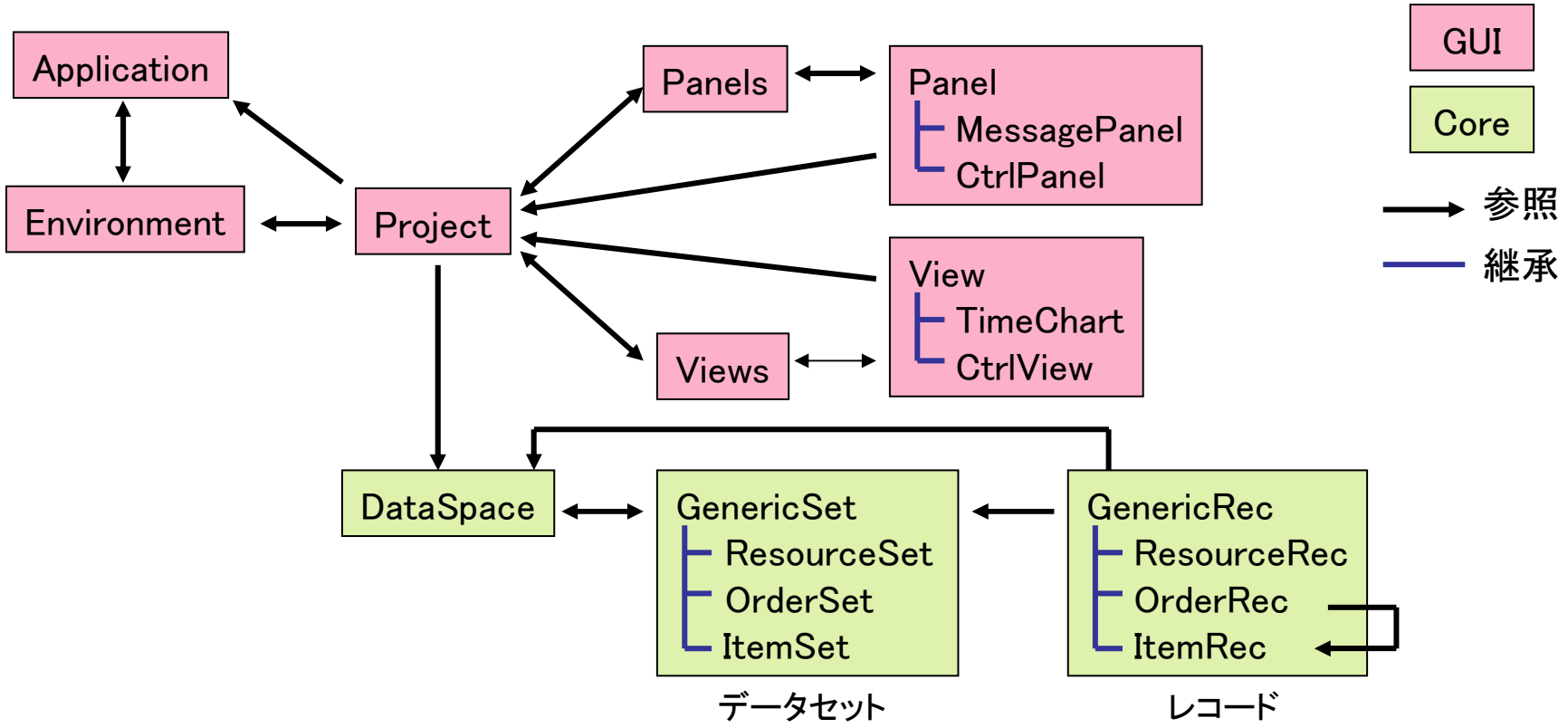
アドインエントリーの登録と実装

- アドインエントリーの登録
  - アドインキー / サブキー / メニュー文字列 / 順序
- エントリーの実装
- UIエントリーの実装
- パラメタの取得

例 : AddInKeyEventOnTaskOnResourceGanttChart

オブジェクト間の関係

主なオブジェクト間の参照関係



これらの関係を理解・把握(＝概念モデルの構築)するとコーディングが非常に楽になります

レコードデータの反復処理

データセットの取得とインデックスによるアクセス

```
set sdSpace = project.DataSpace
set resSet = sdSpace.ResourceSet
c = resSet.CountOfRecords
for i = 0 to c - 1
  set resRec = resSet.Record( i )
  if resRec.IsBound then
    …
  end if
next
```

インデックスは”0-based”

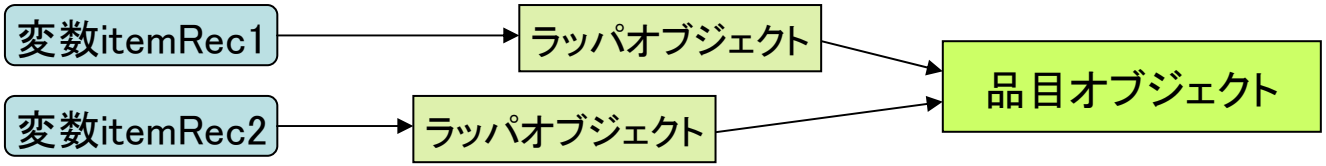
```
set resRec = resSet.ResouceRec( i )
```
でも同じ

レコードラッパとレコードの比較

- レコード同士を直接比較することはできない
  → レコードキー同士を比較する

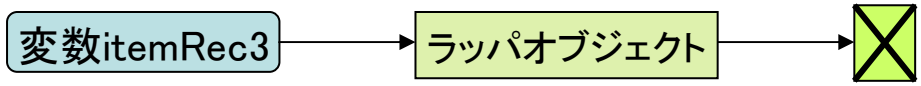
✕ if itemRec1 = itemRec2 then …

○ if itemRec1.RecordKey = itemRec2.RecordKey then …



- レコードがバインドされているか確認が必要

if itemRec3.IsBound then …



※自明ならばチェックを省いてもかまわない
※未バインドラッパのレコードキーは -1

コード

- レコードを一意に識別するための文字列
  例）itemRec.Code
- コードが無いデータセットもある
- コードによるレコード取得

```
set itemRec = itemSet.ItemRecByCode(“A”)
if itemRec.IsBound then
  …
end if
```

フラグとコメント

スクリプティングで多用

- レコードのフラグ

  itemRec.FreeFlag(“flag_name”)　　ブール値(True/False)

- レコードのコメント

  resRec.Comment(“comment_name”)　　文字列

そのほかにも
　仕様（仕様オブジェクト）
　数値仕様（実数値）
　カスタム変数（さまざまな型の値）
の利用機会も多い

作業からオーダーへのアクセス

- 作業には0個以上のオーダーがつながる
- ただしFLEXSCHE GPでは高々1個

（厳密にチェックする場合の例）

```
set operRec = taskRec.OperationRec
if operRec.CountOfOrderRecs > 0 then
  set orderRec = operRec.OrderRec(0)
  if orderRec.IsBound then
    MsgBox orderRec.Code
  end if
end if
```

オーダーが削除されている場合に備え、
チェックは必須（改善予定はあります）

データ更新結果の反映

変更内容に
応じて使い分け

GUIのビュー描画の更新

```
project.Views.UpdateViews
chart.UpdateView
```

プロジェクトパネルやEditorなどの表示の更新

```
project.FireEvent FLEXSCHE.FSEventAllSetsAreUpdated
```

OperationControllerの紹介（参考）

- 作業の割付けを制御するためのオブジェクト

```
set env = keyEntity.ParamObject( FLEXSCHE.ParamIDEnvironment )
set project = keyEntity.ParamObject( FLEXSCHE.ParamIDProject )
set sdSpace = project.DataSpace
set opCon = sdSpace.CreateOperationController
set opSet = sdSpace.OperationSet
set resSet = sdSpace.ResourceSet
set opRec = opSet.OperationRecByCode( "N0001:Proc1" )
opRec.Unassign
opCon.OperationRec = opRec
opCon.UpdateWithOperation
opCon.TimeDirection = SData.SDTDirectionForward
opCon.TestBasisTime = project.SchedulingStartTime
opCon.SetResourceRec opCon.FindTaskIndex( "#primary" ), _
  resSet.ResourceRecByCode( “Resource2" ), True
if opCon.AssignOperationTest then
  opCon.AssignOperation
end if
project.Views.UpdateViews
```

作業の割付けも
自在に制御

GPアンサンブルの利用

- IProject.EnsembleManagerプロパティから「アンサンブルマネージャオブジェクト」を取得

プロパティ

`CountOfSchedulingRules`	ルール数を取得
`CurrentSchedulingRuleIndex`	現在のルールのインデックスを取得
`SchedulingRuleName( Index )`	ルール名を取得
`SchedulingRuleIndexByName( Name )`	ルールの名前からインデックスを取得

メソッド

`ExecuteSchedulingRule Data, Index`	ルールを実行
`Import( Index )`	EDIFインポート。成功ならばTrue
`Export( Index )`	EDIFエクスポート 。成功ならばTrue

後ほど実演

株式会社フレクシェ

FLEXSCHE
スクリプティングフォームの紹介

FLEXSCHEスクリプティングフォーム

- GUI構築によりスクリプティングの適用範囲を広げる
- コーディング量を少なく抑えるシンプルな体系
- 「カーソル・行」モデルによるフォーム構築（座標指定も可）
- モーダルフォームとモードレスフォームを同一モデルで
- ハンドラによるイベント処理とシリアライズ

＊_Show	フォームを開くための関数（内部関数）	
＊_Initialize	フォームの構築	
＊_Load	project.xpsからの読み込み	必要に応じて
＊_Save	project.xpsへの書き出し	必要に応じて
＊_Finish	フォームを閉じるためのボタンが押されたときの処理	モーダルフォームのみ
＊_Event	マウスなどのイベントをハンドリング	

“＊”はフォーム名

FSFの操作

- ISDLFormインターフェースの4つの基本メソッドだけでFSFのすべての操作が可能

```
Setup( name, type, progID )
ShowForm( project )
Command( target, command, [param] )
AddControl( name, ctrlType, [option] )
```

- 利便性のために、Property、Value、NextRow、AddLabelなどの派生プロパティ・メソッドとISDLFormControlインターフェース

詳しくはリファレンスマニュアルをご覧ください

フォームの種類

- モーダルフォーム

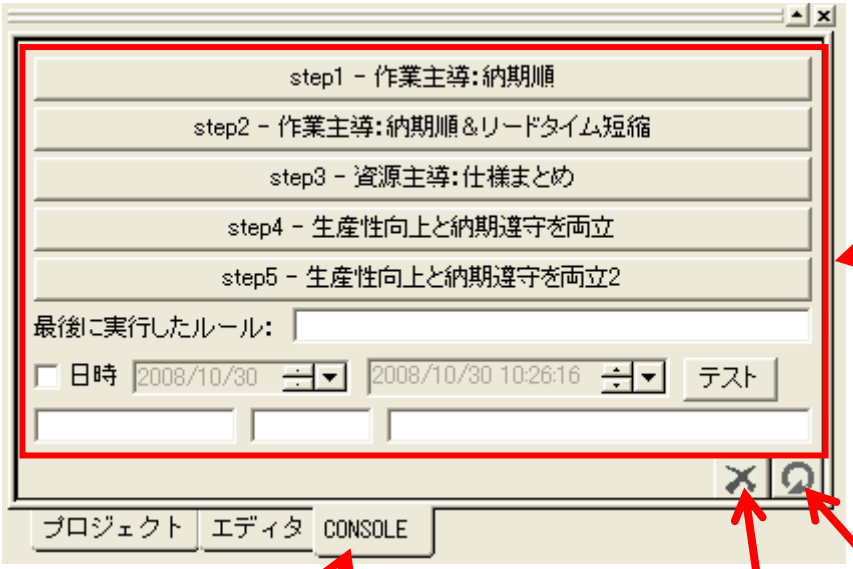
_ok	「OK」のみ
_ok_cancel	「OK」と「キャンセル」
_yes_no	「Yes」と「No」
_yes_no_cancel	「Yes」と「No」と「キャンセル」

- モードレスフォーム

view	カスタムビュー
panel	カスタムパネル
project_pane	プロジェクトペイン

フォームの構成

“project_pane”の場合



実装する部分

リビルドボタン:フォームを再構築
（モードレスフォームのみ）

キャプション

フォームを閉じる(project_paneのみ)

実装されているコントロール

※2008年11月時点

label	ラベル
list	リストボックス
combo_list	コンボボックス
combo_edit	文字列を編集可能なコンボボックス
edit	テキストボックス
button	ボタン
radio	ラジオボタン
check	チェックボックス
check_group	チェックボックスをグループ化する不可視オブジェクト
date	日付入力
datetime	日時入力
null	不可視な水平方向スペーサー

そのほかのコントロールには随時対応予定

フォームを開く

- WSCGenでアドインエントリーを1つとフォームを1つ作成
- メソッド “Show_<フォーム名>” を呼ぶ
- 必要なパラメタを付加する

```
sub SelfRegistration( addIns )
  set addin = addIns.AddAddIn( "Test.ShowMyForm", "Test.Manager", _
    "ShowMyForm", FLEXSCHE.AddInKeyEventOnResourceOnResourceGanttChart )
  addin.SubKeyID = FLEXSCHE.AddInSubKeyEvent_DoubleClick
end sub

function ShowMyForm( keyEntity )
  set timeChart = keyEntity.ParamObject( FLEXSCHE.ParamIDTimeChart )
  set resRec = keyEntity.ParamObject( FLEXSCHE.ParamIDResourceRec )
  Show_MyForm timeChart.Project, resRec
  ShowMyForm = True
end function

sub Show_MyForm( project, resRec )
  set form = CreateObject( "SDLib.Form" )
  form.Setup "MyForm", "_ok_cancel","Test.Manager"
  form.FormData( "Resource" ) = resRec
  if form.ShowForm( project ) = "ok" then
    ' ...
  end if
end sub
```

太字が入力する部分
それ以外は雛形のまま

フォームのハンドラ内で取得可能

実際に生成してみましょう

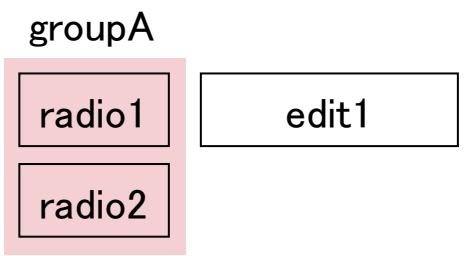
フォームへのデータ渡し

- form.Project　※取得のみ
  - ShowFormメソッドの引数として渡したもの
- form.FormData(“キー文字列”)　※取得・設定
  - プロジェクト以外の任意のデータ

ハンドラの実装例

```
Sub MyForm_Initialize( form )
  form.FormCaption = "サンプル"
  form.AddControl "groupA", "radio_group"
  form.AddControl "radio1", "radio:Choice1"
  form.AddControl "radio2", "radio:Choice2", "down"
  form.AddControl "edit1", "edit::50"
  form.Value("groupA") = "radio1"
End Sub

Function MyForm_Event( form, ctrl, evnt, param )
  MyForm_Event = True
  if ctrl = "groupA" then
    if evnt = "sel_change" then
      form.Value( "edit1" ) = form.Value( "groupA" )
    end if
  end if
End Function
```



実際にコメントをはずして試してみましょう

プロパティ値の取得と設定

- 以下のコードはいずれも同じ意味

取得

```
w = form.Property( “edit1”, “width” )

set ctrl = form.Control( “edit1” )
s = ctrl.Property( “width” )
```

設定

```
form.Property( “edit1”, “width” ) = 80

set ctrl = form.Control( “edit1” )
ctrl.Property( “width” ) = 80
```

プロパティ名

コントロールの名前

プロパティ名についてはリファレンスマニュアルをご覧ください

コントロール値の取得と設定

- 以下のコードはいずれも同じ意味

取得

```
s = form.Value( “edit1” )

s = form.Property( “edit1”, “value” )

set ctrl = form.Control( “edit1” )
s = ctrl.Value
```

コントロールの名前

設定

```
form.Value( “edit1” ) = “abc”

form.Property( “edit1”, “value” ) = “abc”

set ctrl = form.Control( “edit1” )
ctrl.Value = “abc”
```

各コントロールの値(Value)の意味についてはリファレンスマニュアルをご覧ください

初期レイアウト

- “＊_Initialize”ハンドラ内で構築
- AddControlメソッドでコントロールを追加
- 拡張オプションや生成オプションで細かく制御

```
form.AddControl "btnDelete", "button:削除(&D)", "x:50/y:+30/update_cursor"
```

拡張オプション　生成オプション（省略可能）

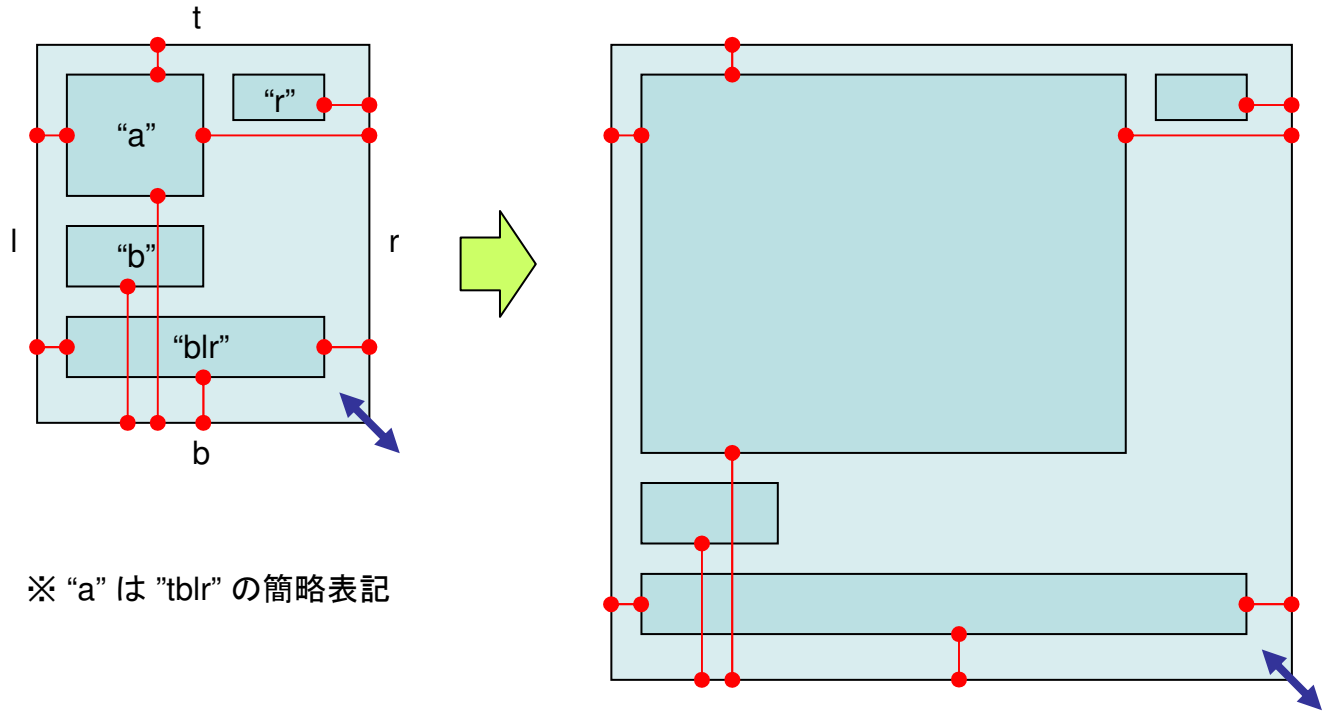
- コントロールの初期値を設定

初期レイアウトの方法

- 行・カーソルモデル
  - カーソルの右方向への暗黙の移動
  - NextRowメソッドによる改行
  - downオプションにより一時的に下方向へ移動
  - タブ指定による列揃え
  - align_rightオプションによる右揃え
- 座標指定
  - フォーム左上を基点とする絶対座標指定
  - 直前のコントロールを基点とする相対座標指定

動的レイアウト

- コントロールのアンカリング
  - フォームの各辺への距離を固定する仕組み



※ “a” は ”tblr” の簡略表記

```
form.AddControl “btn1”, “button:ボタン1”, “anchor:lr”
form.AddControl “btn2”, “button:ボタン2”
form.Property( “btn2”, “anchor” ) = “lr”
```

イベントハンドラの実装

- コントロールで発生したイベントへの対応を記述
- コントロールの識別とイベントの種類の確認

```
Function MyForm_Event( form, ctrl, evnt, param )
  MyForm_Event = True
  if ctrl = "btn1" then
    if evnt = "clicked" then
      ...
    end if
  elseif ctrl = "name" then
    if evnt = "change" then
      ...
    end if
  end if
End Function
```

イベントの種類

コントロールの名前

イベントの種類についてはリファレンスマニュアルをご覧ください

モードレスフォームの挙動

- singletonプロパティ
  - Trueの場合（既定値）
    - フォーム生成時に既存ウィンドウがあればリサイクル
  - Falseの場合
    - いくつでもウィンドウを作れる

```
form.Property( “form”, “singleton” ) = False
```

- リビルドボタン
  - 現在のInitializeハンドラでフォームを構築しなおす 

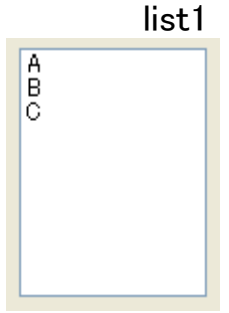
モードレスフォームの内容のシリアライズ

- プロジェクトを開いたときと保存したときに
  フォームの内容をproject.xpsへ読み書きする

例）listコントロールの内容を保存する

```
Function MyForm_Load( form, xmlElem )
  list = form.Control( “list1” )
  codeAttrs = xmlElem.selectNodes( “items/item/@name” )
  c = codeAttrs.length;
  for i = 0 to c - 1
    list.Command “add_string”, codeAttrs.item( i ).value
  next
End Function

Function MyForm_Save( form, xmlElem )
  doc = xmlElem.ownerDocument
  list = form.Control( “list1” )
  c = list.Property( “count” )
  if c > 0 then
    itemsElem = doc.createElement( “items” )
    xmlElem.appendChild itemsElem
    for i = 0 to c - 1
      itemElem = doc.createElement( “item” )
      itemElem.setAttribute “name”, list.Property( “string:” & i )
      itemsElem.appendChild itemElem
    next
  end if
End Function
```



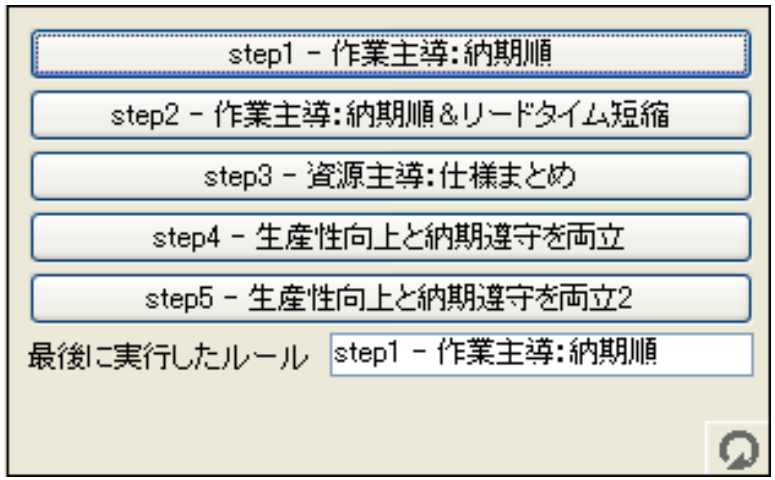

```
…
   <items>
     <item name=“A” />
     <item name=“B” />
     <item name=“C” />
   </item>
…
```

project.xps

詳しくはMSXMLのマニュアル等をご覧ください

フォーム作成の実演

- ルール実行ボタンを配置したパネルを作成
- GPアンサンブル利用の実演も兼ねて



FLEXSCHE GP
外部メソッド

FLEXSCHE GP 外部メソッド

- スケジューリングメソッド「外部メソッド呼び出し」で、外部モジュールを呼び出して実行

- スケジューリングの途中に外部プログラムを実行できる
  cf.アドイン

外部メソッドのインターフェース

- スケジューリングメソッド「外部メソッド呼び出し」
  - 外部モジュールの クラス名 と エントリー名 を指定

- 外部メソッドのシグネチャ

  Function XXXXX(gpManager, settings)

  引数

  1. gpManager
     - GPEns.IManager オブジェクト
     - DataSpace を取得するには gpManager.DataSpace
  2. Settings
     - MSXML2.IXMLDOMElementオブジェクト
     - 「外部メソッド呼び出し」メソッドの設定ダイアログで指定した <settings>要素以下

外部メソッド　例題

- 各オーダーに対して
  コメント（キー”ItemQty”）に
  [オーダー品目] x [オーダー数量] をセット

- ポイント

```
gpManager
  └ DataSpace
    └ OrderSet
      └ OrderRec
        ├ ItemRec
        ├ Qty
        └ Comment
```

外部メソッド　コード例(VBScript)

```
'外部メソッド
'クラス名：ExtMethodTest.Manager
'エントリー名：SetOrderItemQty
Function SetOrderItemQty( gpManager, settings )
  ' ここにコードを書いてください
  set sdSpace = gpManager.DataSpace
  set orderSet = sdSpace.OrderSet
  cOrders = orderSet.CountOfRecords
  commentKey = "ItemQty"
  for iOrder = 0 to cOrders - 1
    set orderRec = orderSet.OrderRec( iOrder )
    if orderRec.IsBound then
      orderRec.Comment( commentKey ) = orderRec.ItemRec.Code & " x " & orderRec.Qty
    end if
  next
  SetOrderItemQty = True
end Function
```

FLEXSCHE GP
外部ファンクション

FLEXSCHE GP 外部ファンクション

- takt式（計算式）の中で
  外部のモジュールを呼び出して値を取得
- 外部ファンクション呼び出し関数の例
  - Bool.External
  - Long.External
  - Double.External
  - Time.External

「（戻り値の型）.External」

外部ファンクションのインターフェース

- 呼び出し側（takt式側）のシグネチャ

  xxx.External(progid, entryName, arg1, …)
  （xxxは返す値のタイプ）

  引数
  - progid　　外部モジュールのProgID（文字列）
  - entryName　外部エントリー関数名（文字列）
  - arg1,…　　自由に渡せる引数（最大5つ）

- 外部モジュールのシグネチャ

  Function YYYYY(args())

  引数
  - args()　　xxx.External に渡されたarg1,…の配列

外部ファンクション　例題

- 資源ガントチャートのタスクデータチップ文字列に、各作業の入力作業の数を表示する
- ポイント
  - 計算式 Long.External で外部ファンクションを呼び出す
  - takt式「object」: コンテクストのターゲットオブジェクト
  - 作業の入力作業？
    - CountOfLinks
    - LinkType(index)
    - LinkOperationRec(index)
    - IsBound

外部ファンクション　コード例（VBScript）

```
'ProgID：ExtFuncTest.Manager
'エントリー名：CountOfInputOperations
function CountOfInputOperations( args )
  ' 参照する引数の分だけコメントをはずしてください
  'arg1 = args(0)
  set operRec = args(0)
  'arg2 = args(1)
  'arg3 = args(2)
  'arg4 = args(3)
  'arg5 = args(4)
  ' ここにコードを書いてください
  clnOpers = 0
  cLinks = operRec.CountOfLinks
  for iLink = 0 to cLinks - 1
    if operRec.LinkType( iLink ) = SData.SDLTypeIn then
      set linkOperRec = operRec.LinkOperationRec( iLink )
      if linkOperRec.IsBound then
        clnOpers = clnOpers + 1
      end if
    end if
  next
  CountOfInputOperations = clnOpers ' 必要とする型の値を返してください
end function
```

外部ファンクション　呼び出し例

- 計算式(takt式)での指定　(コンテクスト「作業」)

```
Long.External( "ExtFuncTest.Manager", "CountOfInputOperations", object )
```

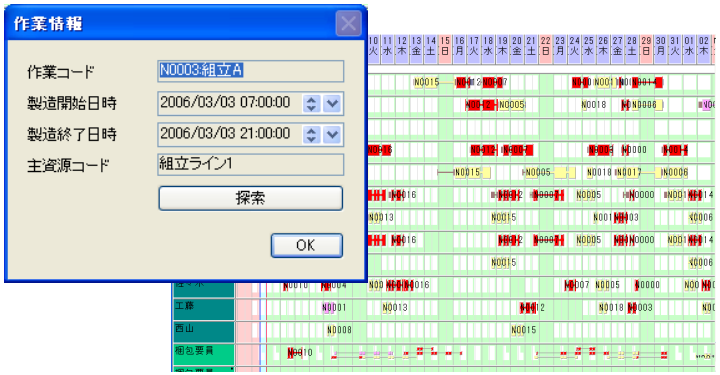
外部ファンクションの注意点

- takt式のxxx.External()で
  引数(arg1,...)を指定しなければ
  外部ファンクションには何も渡されない
- したがって
  値を返すのに必要な「手がかり」は
  適宜引数として渡す必要がある

演習

演習の課題

1. 資源ガントチャートのタスクポップアップメニューから作業の情報（任意）を表示するフォームを開く
2. 資源ガントチャートで、指定した資源を主資源として利用するオーダーを目立つ色で表示する
3. スケジューリングの最後に、各作業の滞留時間を作業の数値仕様にセットする



※ 言語はいずれでも可
※ Data1を使ってください

演習１のヒント

- 単にメッセージボックスに作業コードを表示するだけでもよいが、余力があればモーダルフォームやモードレスフォームでも

アドインキー	AddInKeyEventOnTaskOnResourceGanttChart
サブキー	AddInSubKeyEvent_ContextMenu

ISDLForm.FormData(key)	フォームへのデータ受け渡し
ISDTaskRec.OperationRec	タスクの作業
ISD***Rec.Code	コード
ISDOperationRec.PrimaryTaskRec	主資源タスク
ISDTaskRec.AssignedResourceRec	タスクの資源
ISDOperationRec.ManufactureStartTime	製造開始日時
ISD***Rec.SpecCollection	仕様コレクション
ISDSpecCollection.SpecRec(key)	仕様
ISD***Rec.Comment(key)	コメント
ISDOperationRec.CountOfOrderRecs	作業のオーダー数
ISDOperationRec.OrderRec(index)	作業のオーダー

モーダルフォームの場合、アドインのパラメタとして渡されたタスク（またはその作業）をFormDataプロパティを介してフォームに渡す

演習2のヒント

- 表示設定だけで無理やり実現できなくはないが、
  オーバーヘッドが大きく、対象資源の切り替えも面倒

  表示色指定の条件式　`.Order.Operations.CheckExisting([.PrimaryResource="組立ライン1"])`

- 資源ガント左部ポップアップメニューから切り替え

アドインキー	`AddInKeyEventOnResourceOnResourceGanttChart`
サブキー	`AddInSubKeyEvent_ContextMenu`

`ITimeChart.Project`	プロジェクト
`IProject.DataSpace`	データスペース
`ISDSpace.***Set`	データセット
`ISD***Set.CountOfRecords`	レコード数
`ISD***Rec.FreeFlag( key )`	データセット内のレコード
`ISD***Set.***Rec( index )`	フラグ
`ISDOperationRec.PrimaryResourceRec`	作業の主資源
`ISD***Rec.RecordKey`	レコードキー(比較用)
`ISDOperationRec.CountOfOrderRecs.`	作業のオーダー数
`ISDOperationRec.OrderRec( index )`	作業のオーダー
`IProject.Views`	すべてのビューの集合
`IViews.UpdateViews()`	すべてのビューを再描画

すべての作業を反復して、主資源がマウスで指した資源と一致するものがあれば、その作業のオーダーのフラグを立てる。
オーダーのフラグは最初にすべてOffにしておく。
フラグが立っている作業が目立つ色で表示されるように表示設定しておく。

表示色指定の条件式　`.Order.Flag("use_res")`

演習3のヒント

- 各作業を取得するには？
  gpManager
    DataSpace
      OperationSet
        OperationRec

- 未割り付け作業を除外すること
  OperationRec.IsAssigned
  (OperationRec.IsSplitBase)

- 作業の滞留時間？
  OperationRec.CalcPEST
  OperationRec.ManufactureStartTime

- 作業の数値仕様に値をセットするには？
  OperationRec.NumSpec(“key”) = 123.45
  あるいは
  OperationRec.SpecCollection.NumSpec(“key”) = 123.45