# DEMO1 まずはやってみよう



#### 🖉 WorkflowConsoleApplication1 - Microsoft Visual Studio ファイル(E) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) データ(A) テスト(S) ツール(T) ウィンドウ(W) コミュニティ(C) ヘルプ(H) 🋅 • 🛃 🎒 👗 🐚 🛝 🔊 - 🔍 - 🕨 🍺 Debug 🔹 🔹 💀 🎦 • 🖕 2 🖑 🏹 🔄 🗉 🗉 🗄 🖬 🜌 🖕 Workflow1.cs [デザイン] スタート ページ 👻 🗙 🛛 ソリューション エクスプロー 🕒 🕞 🖬 🗵 🗔 🖧 ~ WorkflowConsoleApplication1 シーケンシャル ワークフロー 🗄 📴 Properties ■ 🔄 参照設定 Ð Program.cs 🛓 📲 Workflow1.cs アクティビティをドロッ りしてシーケンシャル ワークフロー 多作成 • 4 × プロパティ 0 Workflow1.cs ファイルのプロパティ 8∎ 2↓ I 🖻 日 その他 ✓ ファイル名 Workflow1.cs 2 その他 📑 🗶 📦 **# (** [ エラー一覧] プロジェクト 'WorkflowConsoleApplication1' の作成に成功しました。

# CodeActivity をぽとぺ



c	deActivity1 System	m.Workflow.Activ	
-		_	
2↓ 🔲 🖋 🖾			
Ξ	アクティビティ		
	(Name)	codeActivity1	
	Description		
	Enabled	True	
Ξ	ハンドラ		
	ExecuteCode 🔰 😣 🕫		

アクティビティをダブルクリック
namespace WorkflowConsoleApplication1
{
 public sealed partial class Workflow1: SequentialWorkflowActivity
 {
 public Workflow1()
 {
 InitializeComponent();
 }
 private void codeActivity1\_ExecuteCode(object sender, EventArss
 {
 }
 }
 }
}

C  $\lambda \alpha = - F \delta \hat{a} \hat{n}$ string str = Console. ReadLine();
int  $\exists - F = int. Parse(str);$ this. UserData["Key"] = $\exists - F;$ ;
this. UserData["Key"] = $(\exists - F\% 2) == 0;$ 

If-Else アクティビティをぽとぺ

作成・プロジェクト

### $C\# \Longrightarrow Workflow$

新しいプロジェクト			? 🛛		
プロジェクトの種類(P):		テンプレートロン			
ビジネス インテリジ	ジェンス プロジェクト	Visual Studio にインストールされたテンプレート			
<ul> <li>Watal Batic</li> <li>Watal Batic</li> <li>Wardowe</li> <li>Office</li> <li>Office</li> <li>Office</li> <li>Arc+ FJV(ス</li> <li>- プージャースス</li> <li>- パンタート キント(ス)</li> <li>- プージャースス</li> <li>- パンタート キント(ス)</li> <li>Watal Ja</li> <li>Tay Jay - Yay</li> <li>- 分野シスカム ソリージョン</li> <li>- 会野シスカム ソリージョン</li> <li>- マリンクション</li> </ul>		マシーケンシャル ワーカフロー コンソール アガリケ、マシーケンシャル ワークフロー ライブラリ マリーケンマロー アジオ・ビディ タイブラリ マステート マシンのワークフロー シーノン・ル アガン マステート マシンのワークフロー ライブラリ マイ マンパレート	L.		
		コンテレートの検索.     コンテレートの検索.     コン・シンプレートの検索.     コン・シンプレートの人     コン・シン・シン・シン・シン・シンプレートの人     コン・シン・シン・シン・シン・シン・シン・シン・シン・シン・シン・シン・シン・シン			
シーケンシャル ワークフ	ロー コンソール アプリケーシ	ョンを作成するためのプロジェクトです。			
ブロジェクト名(U): WorkflowConsoleAppli 場所(L): C*Documents and Se		cation1			
		tings¥localnaka¥My Documents¥Visual Studio 2005¥projects 🛛 🗸 🎯	展( <u>B</u> )		
ソリューション名(M):	WorkflowConsoleApplication1 マンリューションのディレクトリを作成(D)				
		OK ¥X	ンセル		

シーケンシャルと、ステートマシン、それぞれのコ ンソールアプリ、あとライブラリがある。

今回はシーケンシャルのコンソール



左から順に評価されていきます。 左のブランチの条件を設定します。 宣言型

	· · ·	
□ 条件		
Condition 🛛 🛛	はい	<b>~</b>
	はい	
Condition	宣言型のルール条件	
このアクティビティを有効に	コードの条件	
VECU0		

ConditionName に適当に設定

Expression ...

	5611			
Ξ	Condition		宣言型のルール条件	-
	ConditionName	0		
	Expression		条件式	



(bool)this.UserData["偶数フラグ"] == true という条件をかく

# これ Workflow1.rules に記述される。

andmarteformetion.id/10 distance of the second sec

そして左右のブランチに CodeActivity を配置する



private void codeActivity2\_ExecuteCode(object
sender, EventArgs e)
{

Console.WriteLine(this.UserData["Key"].ToString() + "だよ~~ん"); }

private void codeActivity3\_ExecuteCode(object
sender, EventArgs e)

{

```
Console.WriteLine(this.UserData["Key"].ToString() + "ですよ!!");
```

} 一瞬で消えちゃうので、下に CodeActivity で、 ReadLine をつける。 実行

奇数

偶数

試しましょう。

# DEMO2

# プロジェクト作成

新しいプロジェクト				? 🛛
<b>ガビレップロジェクト</b> <b>プロジェクトの時間(9)</b> ・どうネスインテリジェンスプロジェクト ・どの語る Biolo ・ Viceal Biolo ・ Viceal Till Formmervit 30 ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・		テンプレート(I) Visual Studio にインストールされたテンプレート Windows Application (WPF) WOF Service Library Coustom Control Library (WPF) マイ テンプレート 同力シテイン テンプレートの検索		
■ その他のフロシェク ■ テスト プロジェクト A project for creating	トの理論 an application with a .1 Windows Opelia stime1	IET Framework 3.0 user interface		
クロジェクト名(型): Windows Application 1 場所(型): C*Documents and Se		Alexandria Martin Deserve and 10 Gener	I Studie 200EV-minute	4482 (P)
		ungs#iucainaka#imy Documents#Visua		ANNE /
ソリューション名(団):	WindowsApplication1		✓ソリューションのディレクトリを作成(D)	
			ОК	キャンセル

追加でWFが選べないことを確認

いったんスライドに戻る

# いったん終了 Csproj を開く

<ProjectTypeGuids>{14822709-B5A1-4724-98CA-57A101D1B079};

<Reference

Include="System.Workflow.Activities" />

<Reference

Include="System.Workflow.ComponentModel" />

<Reference

Include="System.Workflow.Runtime" />

<Import

Project="\$(MSBuildExtensionsPath)¥Microsoft¥ Windows Workflow Foundation¥v3.0¥Workflow.Targets" />

シーケンシャルワークフローを追加します。 ここでかならず(コード)を選ぶ。XOML だとまとも に動かない。なぜかは今のところわかりません。



ワークフローの追加ができました。 コードアクティビティを追加します。



ダブルクリック デバッグ用なので、ブレークポイントを置く



# こんな XAML を追加する。

```
FontSize="50">
```

```
<Grid>
```

<Grid.RowDefinitions>

<rowdefinition height="*"></rowdefinition>	ジックち泊
<rowdefinition height="*"></rowdefinition>	ンツクを迫
<rowdefinition height="*"></rowdefinition>	public vo
	RoutedEventArgs ar
<button <="" content="ワークフロー開始" td=""><td>{</td></button>	{
Click="開始" Grid.Row="1"/>	Workfl
	thiswr.CreateWor
	wi.Sta
	}
Cs にイベントを追加する。	
public void 開始(object sender,	テスト
RoutedEventArgs args)	
{	ブレーク OK
}	

Using は5個くらい using System.Workflow; using System.Workflow.Activities; WorkflowRuntime(); コンストラクタで、ランタイム開始 this.\_wr.StartRuntime(); 開始にワークフローを開始するロ 加 <mark>pid 開始(object</mark> sender, rgs) lowInstance wi = rkflow(typeof(Workflow1)); art();

using System.Workflow.ComponentModel;

using System. Workflow. Runtime. Hosting;

ワークフローランタイムをメンバに追加

WorkflowRuntime \_wr = new

using System. Workflow. Runtime;

#### DEMO3

画面の xaml をテキストボックスと、テキストブロッ プロパティで src. dest を追加する クを追加した形に変更 private int src; <TextBox Text="{Binding Path=src}"/> <Button Content="ワークフロー開始" Click="開始" public int src Grid. Row="1"/> get { return \_src; } <TextBlock Text="{Binding Path=dest}" Grid. Row="2"/> set { src = value; やってはいけないけど、データコンテキストに自身を指定 this.FirePropertyChanged("src"); } this. DataContext = this; private int dest; public int dest Using する using System. ComponentModel; get { return \_dest; } インターフェイス実装する set { dest = value; . InotifyPropertyChanged this.FirePropertyChanged("dest"); } ファイアを追加する public event PropertyChangedEventHandler パラメータ渡しするように変更 public void 開始(object sender, RoutedEventArgs PropertyChanged; private void FirePropertyChanged(string args) propertyname) Dictionary<string, object> dic = new if (this. PropertyChanged != null) Dictionary<string, object>(); dic. Add ("src", this. src); { this. PropertyChanged(this, new PropertyChangedEventArgs(propertyname)); } WorkflowInstance wi = this. wr.CreateWorkflow(typeof(Workflow1), dic); wi.Start();

}

# Workflow側にもsrcを追加

private int \_src;

public int src
{
 get { return \_src; }

set { \_src = value; }

ここまででテスト



### 実装2倍にするだけ

private void codeActivity1\_ExecuteCode(object
sender, EventArgs e)
{
 this.dest = this.src \* 2;
}

### 受け渡しインターフェイスを作成

```
[ExternalDataExchange]
public interface I受け渡し
{
void 受け渡し(int value);
}
```

CallExternalMethod をぽとぺ



### さっきのメソッドを指定する

		Enabled	True
		InterfaceType	WindowsApplication3.I受け渡し
		MethodName	受け渡し
	Ξ	パラメータ	
-11	Ŧ	value s	Activity=Workflow1, Path=dest 🛛 🛄
21	Ξ	ハンドラ	

受け渡し実装クラスの作成

```
public class 受け渡しクラス : 【受け渡し
```

```
#region【受け渡しメンバ
```

public void 受け渡し(int value)

```
if (this.受け渡しデリゲート!= null)
{
this.受け渡しデリゲート(value);
}
```

public Action<int> 受け渡しデリゲート;

# #endregion

}

受け渡し実装クラスの登録

コンストラクタに追加

```
ExternalDataExchangeService exservice = new
```

ExternalDataExchangeService();

this.\_wr.AddService(exservice); 受け渡しクラス uke = new 受け渡しクラス(); uke.受け渡しデリゲート= delegate(int

value)

this.dest = value;

};

exservice.AddService(uke);

#### DEMO4

I受け渡しを以下のように変更する。 [ExternalDataExchange] public interface I受け渡し void 受け渡し(Guid guid, int value); **event** EventHandler<ExternalDataEventArgs> 許 可; event EventHandler<ExternalDataEventArgs> 不許 可; Guid を持たせるのは、対話するためのインスタンス番号を 戻さなくてはいけないため。 ワークフロー側にインスタンス GUID を取得するプロパテ ィを追加する。 public Guid guid get

```
return
```

WorkflowEnvironment.WorkflowInstanceId;

}



public class 受け渡しクラス : 【受け渡し **#region** 【受け渡しメンバ public void 受け渡し(Guid guid, int value) if (this.受け渡しデリゲート!= null) this.受け渡しデリゲート(guid, value); public delegate void 受け渡しdelegate (Guid guid, int value); public 受け渡しdelegate 受け渡しデリゲート; public event EventHandler<ExternalDataEventArgs> 許可; public void Fire許可(Guid guid) if (this.許可!= null) this.許可(null, new ExternalDataEventArgs(guid));

#### public event

```
EventHandler<ExternalDataEventArgs> 不許可;
      public void Fire不許可(Guid guid)
          if (this.不許可!= null)
             this.不許可(null, new
ExternalDataEventArgs(guid));
      #endregion
   }
画面を編集する。
4行目を追加して、許可不許可とする。
                <Grid Grid.Row="3">
                        <Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="0.5*"/>
```

<ColumnDefinition Width="0.5\*"/>

</Grid.ColumnDefinitions>

<Button Grid.Column="0"

```
Content="許可" Click="許可"/>
```

<Button Grid.Column="1"

Content="不許可" Click="不許可"/>

 $\langle Grid \rangle$ 

```
画面のイベントを追加する。

public void 許可 (object sender, RoutedEventArgs
args)

{

uke.Fire許可(this.guid);

}

public void 不許可(object sender,

RoutedEventArgs args)

{

uke.Fire不許可(this.guid);
```