コンポーネントの再利用に必要な情報

えムナウ （児玉宏之）
http://mnow.jp/
http://mnow.wankuma.com/
http://blogs.wankuma.com/mnow/
http://www.ailight.jp/blog/mnow/

.NET UX Lab
.Net ユーザーエクスペリエンス研究所

えムナウのC#
プログラミングのページ

わんくま同盟

アジェンダ

- はじめに
- アトリビュート
- 型コンバータ
- まとめ

はじめに

- 汎用的で共通化したコンポーネントをライブラリとして蓄積すると工数の削減や後のプロジェクトでも再利用が可能です。
- 既存のコントロールをカスタマイズしたカスタムコントロールやユーザーコントロールを作るときにプロパティウィンドウを有効利用すると便利になります。
- プロパティウィンドウを有効利用する方法を見て行きましょう。

アトリビュート

- アトリビュートを設定するとプロパティウィンドウの見た目が変わったり便利な使い方が出来たりします。
- コンポーネント開発でよく使うアトリビュートを見て行きましょう。

アトリビュート

- クラスに設定するアトリビュート
  - DefaultPropertyAttribute
    - 既定のプロパティを指定
  - DefaultEventAttribute
    - 既定のイベントを指定
  - ToolboxBitmapAttribute
    - ツールボックスのアイコンのビットマップを指定

```
[
DefaultProperty("MyProperty")
DefaultEvent("MyEvent")
ToolboxBitmap(typeof(MyControl), "MyControlBitmap")
]
```

アトリビュート

- Webコントロールのクラスに必要なアトリビュート
  - AspNetHostingPermission
    - 保護された ASP.NET クラスにアクセスする
  - ToolboxDataAttribute
    - ツールボックスからドラッグされるとき コントロールに生成される既定のタグ

```
[
AspNetHostingPermission(SecurityAction.Demand,
        Level = AspNetHostingPermissionLevel.Minimal),
 AspNetHostingPermission(SecurityAction.InheritanceDemand,
        Level = AspNetHostingPermissionLevel.Minimal),
ToolboxData("<{0}:WebUserControl1 runat=¥"server¥"> </{0}:WebUserControl1>")
]
```

アトリビュート

- Propertyにつけるアトリビュート
  - DefaultValueAttribute
    - プロパティの既定値
  - BrowsableAttribute
    - プロパティウィンドウにプロパティやイベントを表示するかどうか
  - ReadOnlyAttribute
    - プロパティを読み取り専用にするかどうか

アトリビュート

- Propertyにつけるアトリビュート
  - CategoryAttribute
    - 項目別モードに設定されているときに分類するカテゴリの名前
  - DescriptionAttribute
    - プロパティまたはイベントの説明
  - BindableAttribute
    - バインディングに使用されるかどうか

アトリビュート

- Propertyにつけるアトリビュート
  - LocalizableAttribute
    - プロパティをローカライズする必要があるかどうか
  - DesignOnlyAttribute
    - プロパティを設定できるのがデザイン時だけかどうか
  - ParenthesizePropertyNameAttribute
    - プロパティの名前をかっこで囲んで プロパティウィンドウに表示するかどうか

アトリビュート

- Propertyにつけるアトリビュート
  - RefreshPropertiesAttribute
    - プロパティウィンドウを更新する必要があるどうか
  - NotifyParentPropertyAttribute
    - 親プロパティに通知するかどうか
  - DesignerSerializationVisibilityAttribute
    - Foo.Designer.cs にコレクションを初期化するコードを生成するかどうか

アトリビュート

- Propertyにつけるアトリビュート
  - TypeConverterAttribute
    - 型コンバータとして使用するクラスの型
  - EditorAttribute
    - プロパティを変更するために使用するエディタ

    (ダイアログ ボックスまたはドロップダウン ウィンドの為にUITypeEditorの派生クラスのみ指定できる)
  - DesignerAttribute
    - デザイン時サービスを実装するために使用するクラスを指定

    (IDesignerをインターフェイスを実装したクラス)

型コンバータ

- 型コンバータはクラスやプロパティに付加できます。
- クラスの値と文字列の相互変換や.Net Framework がインスタンスを作成するために必要な情報を持ったクラスに変換します。
- 型コンバータを作成すると Enum ではないプロパティをコンボボックスから選択できたりします。

型コンバータ

- コンバータの型に変換するの為のメソッド
  - CanConvertFrom
    - 特定の型のオブジェクトをコンバータの型に変換できるかどうか
  - ConvertFrom
    - コンテキストとカルチャ情報を使用して、オブジェクトをコンバータの型に変換

型コンバータ

```
public override bool CanConvertFrom(ITypeDescriptorContext context, Type sourceType)
{
    if (sourceType == typeof(string)) return true;
    return base.CanConvertFrom(context, sourceType);
}
public override object ConvertFrom(ITypeDescriptorContext context,
        System.Globalization.CultureInfo culture, object value)
{
    if (value.GetType() == typeof(string))   {
        Complex newval = new Complex(0, 0);
        string[] input = ((string)value).Split(new Char[] { ',' });
        if (input.GetLength(0) == 2)    {
            newval.r = double.Parse(input[0]);
            newval.i = double.Parse(input[1]);
        }
        return newval;
    }
    return base.ConvertFrom(context, culture, value);
}
```

型コンバータ

- コンバータの型から変換するの為のメソッド
  - CanConvertTo
    - コンバータの型から特定の型のオブジェクトに変換できるかどうか
  - ConvertTo
    - コンテキストとカルチャ情報を使用して、コンバータの型をオブジェクトに変換
  - InstanceDescriptor に変換することも必要
  - InstanceDescriptor はFrameworkがインスタンスを作るのに利用する

型コンバータ

```
public override bool CanConvertTo(ITypeDescriptorContext context,
Type destinationType)
{
    if (destinationType == typeof(string)) return true;
    if (destinationType == typeof(InstanceDescriptor)) return true;
    return base.CanConvertTo(context, destinationType);
}
```

InstanceDescriptorをサポートする

型コンバータ

```
public override object ConvertTo(ITypeDescriptorContext context,
        System.Globalization.CultureInfo culture, object value, Type destType)
{
    if (destType == typeof(string) && value is Complex)
    {
        Complex ic = (Complex)value;
        return ic.r.ToString() + "," + ic.i.ToString();
    }
    if (destType == typeof(InstanceDescriptor) && value is Complex)
    {
        Complex ic = (Complex)value;
        System.Reflection.ConstructorInfo ctor = typeof(Complex).GetConstructor(
                new Type[] { typeof(double), typeof(double) });
        if (ctor != null)
        {
            return new InstanceDescriptor(ctor, new object[] { ic.r, ic.i });
        }
    }
    return base.ConvertTo(context, culture, value, destType);
}
```

型コンバータ

- コンボボックスから選択するには Enum の殻を利用するのが簡単
- コンボボックスから選択するの為のメソッド
  - GetStandardValuesSupported
    - リストから選択できる標準値セットをオブジェクトがサポートするかどうか
  - GetStandardValues
    - リストから選択できる標準値のコレクションを作成

型コンバータ

```
public override bool GetStandardValuesSupported
        (ITypeDescriptorContext context)
{
    return true;
}

public override TypeConverter.StandardValuesCollection
        GetStandardValues(ITypeDescriptorContext context)
{
    string[] values = { "日本", "カナダ" };
    StandardValuesCollection svc = new StandardValuesCollection(values);
    return svc;
}
```

配列を作って変換する

まとめ

- プロパティウィンドウを有効利用する方法を見てきました。
- 簡単なコンポーネントライブラリを作るには十分な内容です。
- しかし、プロパティウィンドウやデザイナを拡張するまだ入り口のところです。
- 「デザイン時サポートの拡張」をヘルプで検索して調べると色々な方法が見えてきます。