Controlデザイン時対応

.NET Framework

C# & VB

サンプルで学ぶ 独自コントロール 基礎から作り方まで

Visual Studio 2005 / 2008 対応

Visual Studio 2010 / 2012 コンバート対応

ソースコードは、全バージョンに対応

図解資料マスター（Excel 版）付き

丸山プログラミング塾

はじめに

コンポーネントとは何？

コントロールとは何？

「属性」とは何？

そもそもデザイン時対応とは何？

デザイン時対応はしなくても良いの？

デザイン時のデバッグ方法は？

プログラム実行時とデザイン時を考慮したカスタム コントロールのプログラミングとは？

これらの疑問について、基本を解説しながら実験コードとサンプル カスタム コントロールを用いて解きほぐしていく。

用語解説、サンプル カスタム コントロールの紹介、コントロールの基本構造、そして「デザイン時対応の概念－基礎講座」では、型に対応するエディタの種類をプログラムで検証しその結果を、ソースコードとイメージ一覧で紹介する。

サンプル カスタム コントロール プログラムの全体構造を図解、コントロール クラス本体を図解することで、ソースコードの確認作業が格段に向上し、デザイン時対応の UI デザインなどの実践的対応をソースコードから学びとれるようになっている。

「デザイン時対応」は、コントロールのユーザビリティ向上を目指し、よりリッチなデザイン環境を提供するために欠かせない機能である。

本書で「デザイン時対応」のメカニズム・機能・目的・結果・利用方法を明らかにする。

なお、ユーザーインターフェイスを備えた「コントロール」を主として

解説しているが、コントロール自体がコンポーネントから派生しているため「コンポーネント」の開発にも役立つ内容になっている。

付属

- 各種検証用プロジェクト サンプルコード
- サンプル カスタムコントロールプロジェクト

Visual Studio 2005/2008 の両環境、Visual Basic と.NET C#に対応。

- CButtun 解析資料（Excel2007）

本書は、 Visual Studio 2005/2008 Visual Basic または C#の経験者で、フォームベースのプログラミングの経験を前提としている。したがって、言語の解説および基礎的プログラミングの解説は含まれていない。

本書は「ノウハウ」をテキストにまとめた how to 書であり書籍ではない。提供は、PDF ファイルと付属一式になっている。

目次

用語解説

用語解説

コンポーネントとは

.NET では、Component クラスから派生されたクラス全般を指す。コントロールも Component クラスを継承しているためコンポーネントの一部と解釈できる。本書では、コントロールもコンポーネントとして扱っている。

平たくいえば DLL アセンブリ全般を指し、EXE アセンブリと連携して動作する再利用できるコードのユニット（部品）プログラムである。したがって、自力起動能力は持たない。

コントロールとは

.NET では、Component クラスを基底にした Control クラスを継承した DLL アセンブリのことをいう。つまり、ユーザーインターフェイスを持ったコンポーネントをコントロールと呼ぶ。

Visual Studio とのデザイン時対応が強化され、独自のインターフェイスを提供することができる。

デザイナ ホストとは

Visual Studio のフォームおよびコンポーネント（コントロール）に対するデザイン時機能全般を総称して「デザイナ ホスト」と呼ぶ。

デザイナ ホストは、ツール ボックス、フォーム デザイナ、プロパティ ウィンドウ、ドキュメント アウトラインなどの基本機能を提供する。

コントロールを追加すると、ツール ボックスに追加されたコントロールを表示する。コントロールをフォーム画面に貼り付けると、コントロールが持つプロパティリストを取得し、プロパティ ウィンドウに表示

する。プロパティウィンドウは、取得したプロパティリストから、カテゴリ名、プロパティ名、現在値、プロパティ解説、コマンドなどを表示・編集を管理する。ドキュメント アウトラインは、フォーム上に配置された全コントロールをツリー階層表示し編集を管理する。

また拡張機能として、カスタム コントロールに対して様々な機能を提供するための仕組み「デザイン時対応用のインターフェイス」が用意されている。本書では、この機能を中心に解説する。

デザイン時とは

デザイナ ホストを利用している状態を「デザイン時」と呼ぶ。

デザイン時対応とは

コンポーネントがデザイナ ホストとのインターフェイスを活用し、デザイン時における様々なサービスに対する機能拡張を行うことを「デザイン時対応」と呼ぶ。

再利用を考慮するためには

コンポーネント（コントロール）本体の汎用化が重要なポイントになる。特定のプロジェクトに限定したコンポーネントは汎用性を欠く場合があるため、他のプロジェクトで使用できない場合が多い。

単純なコンポーネントの場合やデザイン時のユーザビリティを意識する必要がない場合、全く何もしないか、プロパティの属性定義を追加する程度で済む場合が多い。

再利用を考慮するためには、固有のプロジェクトではなく、対象を一般化して、用途、機能、振る舞いなどをまとめ上げる必要がある。その際、実行時だけでなく、デザイン時対応も合わせて設計することが必要になる。

サンプル
カスタム コントロールの
機能とデザイン時対応

サンプル カスタム コントロールの機能とデザイン時対応

サンプルコントロール紹介

ここで紹介するサンプルは CodeProject サイトにある「Custom Button Control with Gradient Colors and Extra Image (VB.NET)」で、このサンプルは Control クラスからカスタムコントロールを構築している。

Button Control の基本機能は、標準の Button と同じだが、標準には無い拡張された様々な表現力を備えている。同時に、拡張機能に対するデザイン時への対応もカストマイズされているため、カスタム コントロールの学習には最適な素材になっている。また、グラフィック GDI+ を使ったグラデーション描画なども含まれている。

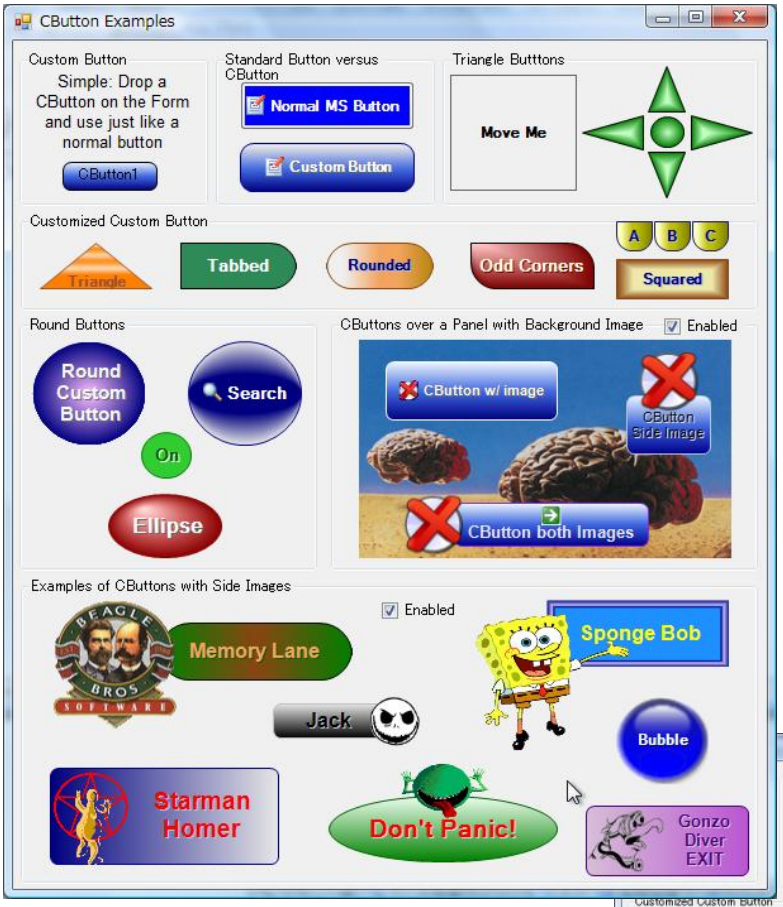
なお、上記サイトでの公開版は Visual Basic のみとなっているため、本書作成のため C#版への移植を行った（ファイル構成は異なっている）。

本サンプルのライセンスは、The Code Project Open License (CPOL) に記載されている。ソフトウェアとしての保証は無いが、商用、配布、派生はロイヤリティ不要。ライセンスに従い、付属の VS 2005 Visual Basic 版は、サイトに登録されているものを添付している。

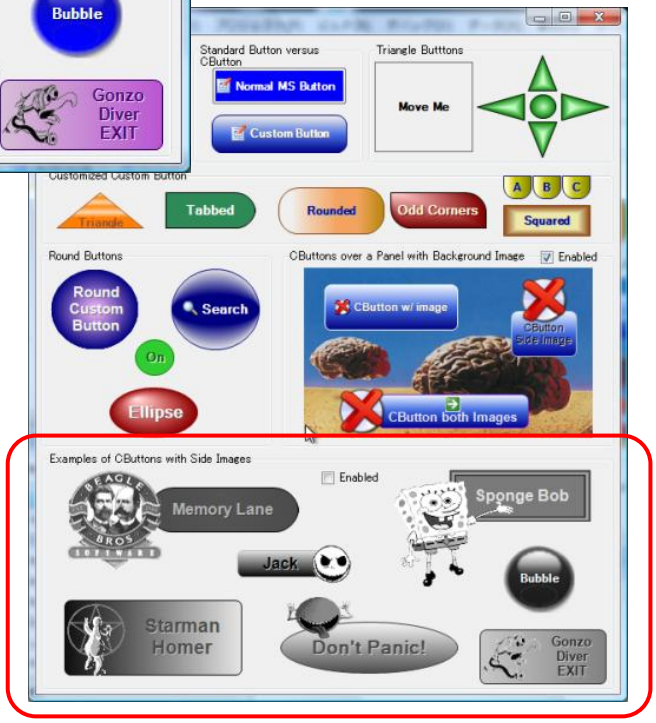
特徴

- 形状－矩形ボタン、角丸ボタン、丸・楕円ボタン、三角形ボタン
- 描画－グラデーション描画による、立体的表現、クリスタル表現など
- イメージ－2 つのイメージに対応
  - ボタン内に配置する Image
  - ボタン領域内に配置する Image

- テキストーテキストの影付き
- 透過対応ー画像上に貼り付けても非表示部分は透過される
- その他ーグラデーションパスの場合、デザイナ上から中心点と焦点位置を操作できる（プロパティでも可能）



1つのコントロールでこれだけの表現を行っている。



プロパティの Enable を false にするとモノクロ グラデーション表示に切り替わる