.NET Framework C# & VB

サンプルで学ぶ カスタムコントロール 入門

Visual Studio 2008 対応

Visual Studio 2010 / 2012 コンバート対応

丸山プログラミング塾

はじめに

もっと、

カスタム コントロール化して、プログラミングを楽にしよう！

です。

同じプログラミングするなら、楽に出来た方がいいですよね。

画面のコードはすっきりするし、
トラブル時やカストマイズ対応も早く出来るのですから、
やらない方がもったいないです。
そうしないのは、敢えて苦労を背負い込んでいるようなものですよ。

複数のコントロールが連動するプログラムを書くなら「ユーザーコントロール」でまとめてしまいましょう。

ちょっと、**込み入った独自のコントロール**が必要なら、既存の標準コントロールをベースにするか、独自のコントロールで作ります。

Form 画面に直接描画するのは『御法度』

なぜなら、画面のレイアウトができない、実行するまで何が表示されるか分からない・・・と弊害の可能性がふくらみ、Form 画面が複雑になることは避けられません。

はじめは良くても、後でクレームになったり、画面デザインが変更になったりしたとき、直接描画していると大変です。後で、そんな苦労をするのだったら、はじめからカスタム化しておいた方がいいと思いませんか？

画面に直接描画するということは、動的なレイアウトに対応しないということになります。

コントロール化するということは、動的レイアウトを意識しなければなりません。

カスタム化は面倒だ！？

もし、カスタム化は面倒だと思っているのでしたら、それは

- カスタム化の方法を知らないから
- カスタム化のメリットを知らないから
- 勉強の時間が必要（面倒）

  結局、知っている方法で対処するしかないとあきらめる？

でも、それは『誤解』です。

やってみてから、判断した方がいいと思いますがいかがですか？

例えば、Form 画面に様々なコントロールを貼り付けて、画面内で制御（振る舞いをコントロール）コードを書いていますよね。

それって、返ってコードの量は増えるし、複雑になっていると思いませんか？

Form 画面の機能を少し整理して、複数のコントロールをまとめてカスタム化すると「絶対に楽になります」「カスタム化が面白くなります」です。

ただし、その代償として

Control クラスや UserControl クラスの知識、継承、ポリモーフィズム、属性、グラフィック描画などの基礎知識と応用力が必要になってきます。本テキストで、ここまで解説することは出来ませんので、あらかじめご了承ください。

なお、サンプルコードを通して、その範囲で身に付けていただくことは出来ると思いますよ。

目次

用語解説

カスタム コントロールとは

標準コントロールのカストマイズ

標準コントロールを継承して、専用のコントロール（あるいは汎用的なコントロール）を作成します。

必要な部分だけ機能を拡張したり変更したりすればいいので、手軽にできる方法です。なお、Visual Studio のデザイン機能は使用できません。

複合標準コントロールをまとめる

個々のコントロールが連携する場合、独自の描画などのオリジナルコントロールを作成する場合、UserControl を継承して作成します。

Visual Studio のデザイン機能が利用できるため、Form 画面と同じ感覚で作成することが出来ます。

なお、プロパティ、イベントなどは、UserControl が公開されるため、内部のコントロールと直結する場合は、追加、変更する必要があります。

独自のコントロール

Control クラスから継承して独自のコントロールを作成する方法です。詳細は、『サンプルで学ぶ『カスタム コントロールの基礎と作り方』を参照してください。

共通事項

- コントロールは、Visual Studio でデザイン時対応を考慮するか、しないかを決定する必要があります。
  この機能については『サンプルで学ぶ『カスタム コントロールの基礎と作り方』』で詳しく述べていますのでここでは省略します。

- Form クラスのコードをすっきりとさせる

  Form 画面のコードには、Form（画面自身）と子コントロールの両方の『振る舞い』処理が記述されます。

  子コントロールの総数が増えるに従って、制御が複雑になるにしたがって、コードが読み辛くなってきます。また、バグの温床になりかねない重大な問題になる可能性もあります。
- 複数画面で同様の処理が必要な場合、カスタム化を検討
- カスタム コントロールの第三者提供

  ある程度使いやすくする必要がありため、デザイン時対応を検討した方がいいでしょう。

デザイン時対応とは

Visual Studio の『デザイン』タブで表示される画面は、すべてのコントロール（もちろん Form 自信も含む）を、Visual Studio が管理しています（Visual Studio はリフレクション機能を使って各コントロールを制御しています）。コントロールを貼り付けると、プロパティ設定の変更もリアルタイムに応答します。public なプロパティやイベントが Visual Studio のプロパティ ウィンドウに表示されない場合もあります。これは、コントロールの属性によって Visual Studio に対して定義しているからです。

これらを『デザイン時対応』と呼びます。

また、コントロール自身は、今、デザイン時モードなのか、実行時モードなのかを判断することが出来ます。それによって、振る舞いを変えることが出来ます。

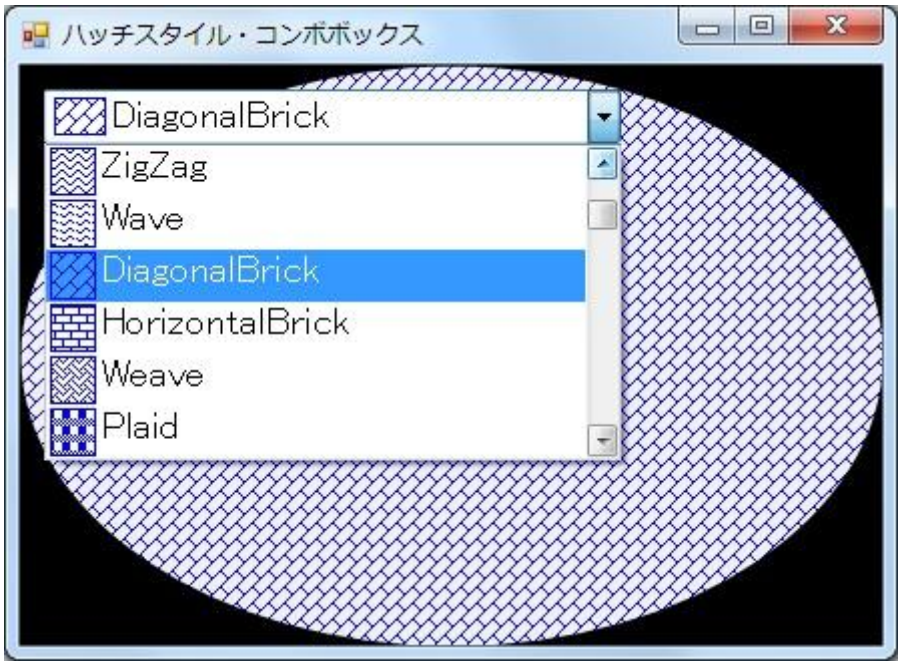
サンプル
カスタム コントロール化
入門

標準コントロールのカストマイズ

概要

標準の ComboBox コントロールを継承して、オーナー描画するコントロールを作成します。

ComboBox クラスを継承して、HatchStyleComboBox クラスを作成し、ハッチバックスタイルの一覧を表示するカスタム コントロールを作成します。このサンプルは、コンボボックスのアイテムとプルダウン時のリストボックスに対して、フォントサイズの高さと幅から項目の高さを求めて直接描画しています。



特徴

- コンボボックスのフォントサイズに連動