Dialog for ASP.NET Web Forms

2018.04.25 更新

グレープシティ株式会社

目次

製品の概要

Dialog for ASP.NET Web Forms を使用して、堅牢かつインタラクティブで、カスタマイズ可能なダイアログウィンドウを作成します。この特別な種類のダイアログウィンドウは、情報を表示したり、ユーザーからの入力を受信するためにクライアントまたはサーバー側で作成できます。

ComponentOne for ASP.NET Web Forms のヘルプ

ComponentOne for ASP.NET Web Forms の各コントロールで共通したトピック、アセンブリの追加、テーマの適用、クライアント側情報などについては「ASP.NET Web Forms ユーザーガイド」を参照してください。

主な特長

C1Dialog コントロールは、クライアントまたはサーバー側で作成できる、情報の表示やユーザーからの入力の受信用の特別な種類のダイアログウィンドウです。**C1Dialog** を使用すると、いずれかの Open() メソッドを呼び出して、モーダルまたはモードレスダイアログウィンドウを作成できます。

ダイアログウィンドウは次のように使用できます。

- ユーザー入力が必要な項目を使用するとき、追加情報の要求画面をユーザーに表示します。
- アプリケーションの入力を取得します。
- ユーザーが別のウィンドウで作業中にアプリケーションの関連情報またはオプションを表示します。

機能概要

C1Dialog は、充実したオブジェクトモデル、モーダル／モードレスダイアログウィンドウ、部分ページレンダリング、スタイルプロパティ、組み込みテーマ、テンプレートサポート、HTMLコンテンツ、外部コンテンツ、ウィンドウ配置、最小化オプション、移動可能なダイアログウィンドウ、およびサイズ変更可能なダイアログウィンドウなど、数々のユニークな機能を備えています。

- **モーダル／モードレスダイアログウィンドウ**
  Dialog for ASP.NET Web Forms は、モーダルおよびモードレスダイアログウィンドウという、2種類のダイアログウィンドウを提供します。モーダルダイアログウィンドウは、ユーザーが現在のアプリケーション上で作業を続行するときに閉じる必要がある子ウィンドウです。モードレスダイアログボックスでは、ユーザーはこのダイアログウィンドウを表示したままで、他のウィンドウをインタラクティブ操作できます。これらのダイアログウィンドウについての詳細は、「モーダル／モードレスダイアログウィンドウ」を参照してください。
- **部分ページレンダリング**
  アプリケーションの性能が向上し、ユーザーが部分ページレンダリング（PPR）技術を使用してアクションを実行するときに、より直接的なフィードバックが提供されます。
- **ドッキング**
  ウィンドウが最小化されたときにダイアログボックスをゾーンにドッキングできます。
- **スタイル**
  Dialog for ASP.NET Web Formsは、キャプションバーとコンテンツ要素に対し、固有プロパティのスタイルを提供します。
- **スピナー**
  C1Dialogコントロールは、コンテンツがロード中であることを示すスピナー要素を備えています。スタイルを使って、この画像をカスタマイズできます。
- **テンプレートのサポート**
  ダイアログウィンドウのコンテンツ領域にテンプレートを追加できます。動的テンプレートをダイアログウィンドウのコンテンツ領域に使用して、ダイアログウィンドウの充実した表示を達成できます。テンプレートの詳細については、「コンテンツテンプレート」を参照してください。
- **アニメーション**
  任意の組み込みアニメーション効果を使用して、ダイアログのインタラクティブ操作をカスタマイズします。ダイアログウィンドウの表示／非表示の方法、および拡張／縮小の方法を変更します。デフォルトでは、ダイアログウィンドウはアニメーション効果を使用しません。
- **テーマ**
  スマートタグをクリックするだけで、6種類のプレミアムテーマ（Arctic、Midnight、Aristo、Rocket、Cobalt、および Sterling）のいずれかを選択してプログレスバーの外観を変更します。オプションとして、jQuery UI から ThemeRoller を使用してカスタマイズしたテーマを作成します。
- **CSS のサポート**
  CSS（Cascading Style Sheet）のスタイルを使用して、カスタムスキンを定義します。

クイックスタート

このクイックスタートでは、**Dialog for ASP.NET Web Forms** を始める方法について説明します。クイックスタートでは、Visual Studio でプロジェクトを作成して、モードレス **C1Dialog** ウィンドウをページに追加し、ダイアログウィンドウの外観と動作を変更し、ダイアログウィンドウのランタイム動作を確認します。

手順 1:ページへの C1Dialog の追加

この手順では、新しいプロジェクトを作成して、**Dialog for ASP.NET Web Forms** ダイアログウィンドウをプロジェクトに追加します。クイックスタートを開始するには、以下の手順を実行します。

1. Visual Studio の[**ファイル**]メニューから、[**新規**]→[**プロジェクト**]を選択します。[**新規プロジェクト**]ダイアログボックスが表示されます。
2. [**新規プロジェクト**]ダイアログボックスで、左ペインにある言語を展開し、[**Web**]を選択します。右ペインで、[**ASP.NET 空の Web アプリケーション**]を選択し、アプリケーションの[**名前**]を入力して〈**OK**〉を選択します。新しいアプリケーションが作成されます。
3. ソリューションエクスプローラで、プロジェクトを右クリックしてコンテキストメニューを開き、[**追加**]→[**新しい項目**]を選択します。
4. [**新しい項目の追加**]ダイアログボックスで、テンプレートのリストから[**Web フォーム**]を選択し、項目に Default.aspx という名前を付けて、〈**追加**〉をクリックします。新しいページが開きます。
5. デザインビューで、Visual Studio ツールボックスに移動し、**Input (Button)** と **C1Dialog** アイコンをダブルクリックして、**Button** および **C1Dialog** コントロールをページに追加します。
6. **Button1** を選択し、**プロパティ**ウィンドウに移動して、コントロールの **Value** プロパティを「ダイアログを表示」に設定します。
7. ソースビューで、**onclick** エベントハンドラを **Button1** に追加します。コントロールのマークアップは次のようになります。

ソースビュー

```
<input id="Button1" type="button" value="ダイアログを表示"
    onclick="$('#<%=C1Dialog1.ClientID%>').c1dialog('open')"/>
```

ここで、**Button** をクリックすると **C1Dialog** ダイアログウィンドウが開きます。

新規プロジェクトを作成して、**C1Dialog** ダイアログウィンドウをプロジェクトに追加する操作が完了しました。次の手順では、ダイアログウィンドウの外観と動作をカスタマイズします。

手順 2:ダイアログウィンドウのカスタマイズ

この手順では、前の手順で作成したダイアログの外観と動作を変更します。次の手順を実行して、**Dialog for ASP.NET Web Forms** ダイアログウィンドウをカスタマイズします。

1. **C1Dialog** スマートタグを選択して、[**C1Dialog タスク**]メニューを開き、以下のように設定します。
   - [**オンロードの表示**]チェックボックスを OFF にして、当初はアプリケーションを実行したときにコントロールが表示されないようにします。
   - [**テーマ**]ドロップダウンボックスで **Rocket** を選択して、ダイアログウィンドウの外観を変更します。
2. **C1Dialog** コントロール上で1回クリックして選択し、**プロパティ**ウィンドウで次のプロパティを設定します。
   - **Title** プロパティを「C1Dialog」に設定して、キャプションバーのタイトルを設定します。
   - **Height** を **200** に設定して、ダイアログウィンドウのサイズを大きくします。
3. ダイアログウィンドウの[コンテンツ要素]で1回クリックして、「Hello World!」と入力してテキストコンテンツをダイアログウィンドウに追加します。マークアップが次のように表示されることに注意してください。

ソースビュー

```
<cc1:C1Dialog ID="C1Dialog1" runat="server" Height="200px" Title="C1Dialog">
<CaptionButtons>
<Pin IconClassOn="ui-icon-pin-w" IconClassOff="ui-icon-pin-s"></Pin>
<Refresh IconClassOn="ui-icon-refresh"></Refresh>
<Minimize IconClassOn="ui-icon-minus"></Minimize>
<Maximize IconClassOn="ui-icon-extlink"></Maximize>
<Close IconClassOn="ui-icon-close"></Close>
</CaptionButtons>
<Content>
Hello World!
</Content>
</cc1:C1Dialog>
```

Dialog for ASP.NET Web Forms クイックスタートの手順 2が完了しました。次の手順では、プロジェクトを実行します。

手順 3：アプリケーションの実行

これまでは、ダイアログウィンドウの外観と動作をカスタマイズしました。残された作業は、アプリケーションを実行して、**C1Dialog** ダイアログウィンドウのランタイム機能のいくつかを確認することです。

以下の手順を実行します。

1. 〈**ダイアログを表示**〉ボタンをクリックします。
   Web ページにダイアログウィンドウが表示されます。

2. ダイアログウィンドウのヘッダーをクリックし、ドラッグ &ドロップ操作を実行して、Web ページ上のこのダイアログウィンドウに移動します。
3. ダイアログウィンドウの右下端をクリックして、ドラッグ &ドロップ操作を実行し、ダイアログウィンドウのサイズを変更します。
4. ダイアログボックスの右上端にある〈閉じる〉ボタンをクリックしてダイアログウィンドウを閉じます。

おめでとうございます。**C1Dialog** ダイアログウィンドウが作成され、カスタマイズされ、**Dialog for ASP.NET Web Forms** クイックスタートが無事完了しました。

デザイン時のサポート

Dialog for ASP.NET Web Forms は、カスタマイズされたコンテキストメニュー、スマートタグ、およびデザイン時サポートを備え、オブジェクトモデルでの作業を単純化します。

以下のセクションでは、**Dialog for ASP.NET Web Forms** のデザイン時環境、特に **C1Dialog** のスマートタグおよび コンテキストメニューからアクセス可能な[**C1Dialog タスク**]メニューを使用して、**C1Dialog** コントロールを設定する方法について説明します。

C1Dialog スマートタグ

C1Dialog コントロールは、スマートタグを備えています。スマートタグは、各コンポーネント／コマンドで最もよく使われるプロパティを提供するショートカットのタスクメニューを表します。

[**C1Dialog タスク**]メニューにアクセスするには、**C1Dialog** コントロールの右上端にあるスマートタグ矢印をクリックします。[**C1Dialogタスク**]メニューが開きます。

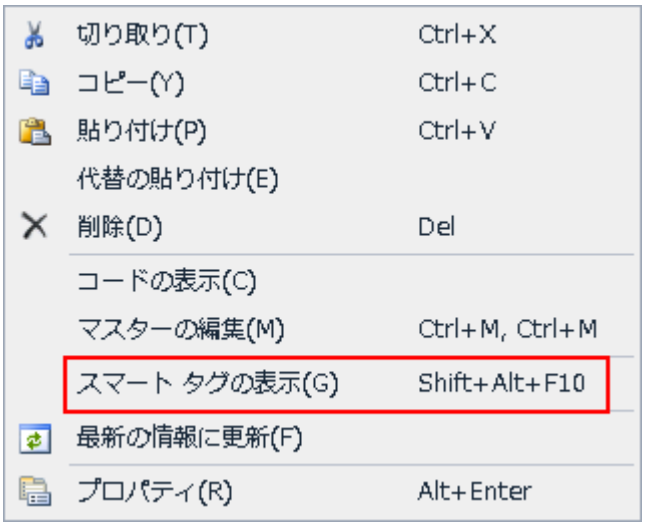
[**C1Dialog タスク**]メニューは次のように動作します。

- **オンロードの表示**
  [**オンロードの表示**]チェックボックスが ON のとき、**ShowOnLoad** プロパティが **True**に設定され、ロード時にダイアログウィンドウがページに表示されます。デフォルトでは、[**オンロードの表示**]チェックボックスが ON、**ShowOnLoad**プロパティが **True** に設定されます。
- **テーマ**
  [**テーマ**][**テーマ**]ドロップダウンをクリックして提供される6つの組み込み **ComponentOne for ASP.NET Web Forms** テーマから選択してコントロールに適用することができます。
- **新しいテーマの作成**
  [**新しいテーマの作成**]オプションをクリックすると、**ThemeRoller for Visual Studio** が開きます。したがって、開発環境内でテーマをカスタマイズすることができます。アプリケーションで **ThemeRoller for Visual Studio** を使用する方法については、「ThemeRoller for Visual Studio」を参照してください。
- **CDN の使用**
  [**CDN の使用**]チェックボックスを ON にすると、CDN からクライアントリソースがロードされます。これはデフォルトで OFF です。
- **CDN パス**
  CDN の URL パスを表示します。
- **Bootstrap の使用**
  [**Bootstrap の使用**]オプションを選択すると、コントロールに Bootstrap テーマを適用することができます。アプリケーションで Bootstrap テーマを使用する方法については、「Bootstrap for ASP.NET Web Forms クイックスタート」を参照してください。
- **バージョン情報**

［**バージョン情報**］をクリックすると、製品のバージョン情報を確認できるダイアログボックスが表示されます。

C1Dialog コンテキストメニュー

C1Dialog コントロールは、Visual Studio がすべての .NET コントロールに提供するコンテキストメニューの追加コマンドを備えています。**C1Dialog** コンテキストメニューにアクセスするには、**C1Dialog** コントロール上のどこかを右クリックします。**C1Dialog** コンテキストメニューが開きます。



C1Dialog コンテキストメニューには、**C1Dialog** で追加される以下のカスタムコマンドが含まれます。

- **スマート タグの表示**
  C1Dialog コントロールのスマートタグを表示します。スマートタグの使用方法や利用可能な機能についての詳細は、「C1Dialog スマートタグ」を参照してください。

モーダル／モードレスダイアログウィンドウ

ダイアログボックスは、ユーザーからの入力を取り込むアプリケーションとして一般に使用されます。一部のアプリケーションでは、ダイアログボックスは、ユーザーに入力を要求するために使用され、アプリケーションが入力を取り込むと、ダイアログボックスは自動的に閉じるか、無効になります。

他方、一部のアプリケーションでは、ダイアログボックスは、ユーザーが別のウィンドウで作業しているときに情報を表示するために使用されます。たとえば、Microsoft Word でスペルチェックするとき、ユーザーがドキュメントのテキストの検索や編集ができるよう、1つのダイアログボックスを開いたままにして、スペルチェッカは次のスペルミスの単語を検索します。アプリケーションでのダイアログボックスのさまざまな使用方法をサポートするために、**C1Dialog**は、モーダル および モードレス ダイアログウィンドウという、2種類のダイアログウィンドウを提供します。

モーダルダイアログウィンドウ

モーダルダイアログウィンドウは、ユーザーが現在のアプリケーション上で作業を続行するときに閉じる必要がある子ウィンドウです。通常、モーダルダイアログウィンドウは、閉じられるまでは、これらのウィンドウを表示させているシステム全体またはアプリケーションを制御します。たとえば、アプリケーションでユーザーが作業を続行できるようにするために、先に、モーダルダイアログウィンドウを使用して、ユーザーからログイン情報を取得できます。モーダルダイアログウィンドウを表示するには、**Modal** プロパティを **True** に設定し、JavaScript で **Open()** メソッドを使用します。

モーダルウィンドウは、重要情報を提示する場合やユーザー操作が必要な場合に役立ちます。ダイアログがアプリケーションに対してどこに表示されるかを指定するには、AppendToプロパティを使用することができます。このプロパティを使用すると、ページ上の特定の要素にC1Dialogコントロールを追加することができます。

モードレスダイアログウィンドウ

モードレスダイアログボックスでは、ユーザーはこのダイアログウィンドウを表示したままで、他のウィンドウをインタラクティブ操作できます。要求された情報が操作の続行を必要としない場合、このタイプのダイアログウィンドウを使用します。モードレスダイアログウィンドウは、入力フォーカスを維持しないため、2つのアプリケーションで同時に作業できます。**Modal** プロパティを **False** に設定し、JavaScript で **Open()** メソッドを使用して、モードレスダイアログウィンドウを表示します。

モードレスダイアログウィンドウは、ダイアログウィンドウとアプリケーションウィンドウを同時に使用できる、メニューやヘルプシステムで一般に使用されます。たとえば、ツールバーはモードレスダイアログウィンドウです。その理由は、ツールバーをアプリケーションから独立させることができるためで、ユーザーはツールバーの項目を選択して、さまざまな機能を独立または分離したたアプリケーションに適用できます。AppendToプロパティを使用して、C1Dialogコントロールは、ページ上のどの要素に追加されるかを指定することができます。

C1Dialog の要素

このセクションでは、**C1Dialog** コントロールを構成する各要素の視覚的および記述的な概要を提供します。各トピックは、ダイアログウィンドウコントロールの各側面を表す、コンテンツ要素、キャプションバー要素、ステータスバー要素の3種類の要素に分類されます。

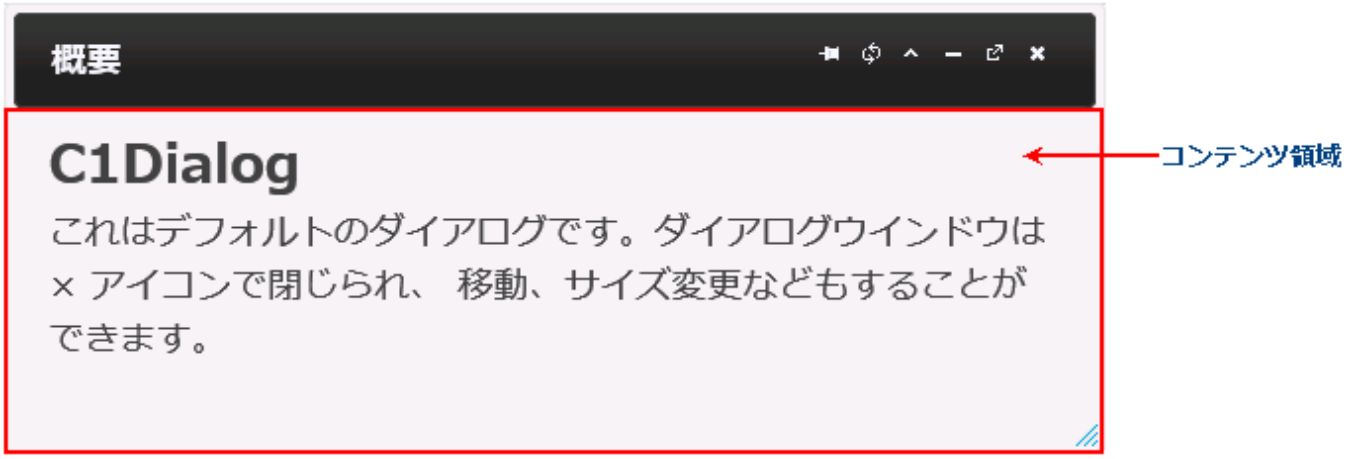
コンテンツ要素

C1Dialog コントロールの主要部分はコンテンツ領域です。コンテンツ領域では、カスタム HTML コンテンツからはリッチテキスト、**ContentUrl** プロパティからは URL リンク、コンテンツテンプレートからは任意のコントロールを追加できます。単純なドラッグ &ドロップ操作で、コントロール上でコントロールのコンテンツ領域の各要素の追加と移動ができます。

C1Dialog には、テキスト、画像、任意のコントロール、およびコンテンツ領域へのリンクなどの各種の項目を追加および簡単にカスタマイズできる以下の各プロパティが含まれます。

- **Content**
- **ContentUrl**

以下の画面は、**C1Dialog** コントロールのコンテンツ領域を示します。



スタイルシートを使用して、独自のスタイルを**C1Dialog** コントロールのコンテンツに適用できます。

デザイン時にダイアログウィンドウのコンテンツ領域にテキストを入力できます。テキストをコンテンツ領域に入力すると、次のように、**C1Dialog** は、`<cc1:C1Dialog>` タグ内に `<ContentTemplate>` タグを追加します。

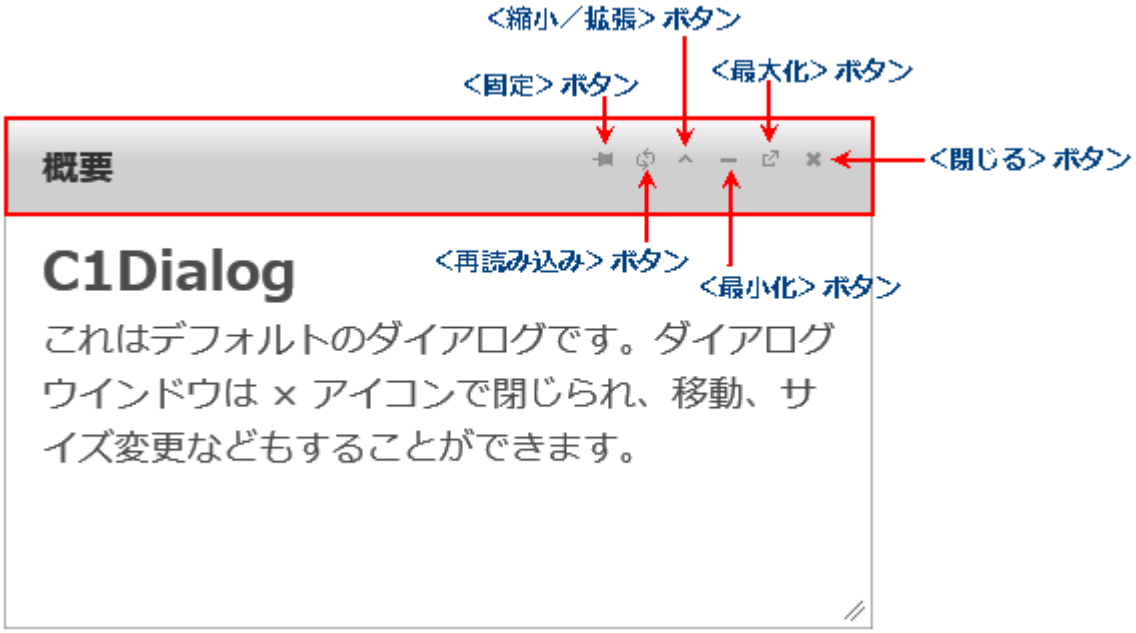
ソースビュー

```
<cc1:C1Dialog ID="C1Dialog1" runat="server" Height="200px" Title="概要">
<CaptionButtons>
<Pin IconClassOn="ui-icon-pin-w" IconClassOff="ui-icon-pin-s"></Pin>
<Refresh IconClassOn="ui-icon-refresh"></Refresh>
<Minimize IconClassOn="ui-icon-minus"></Minimize>
<Maximize IconClassOn="ui-icon-extlink"></Maximize>
<Close IconClassOn="ui-icon-close"></Close>
</CaptionButtons>
<Content>
<h2>
C1Dialog</h2>
<p>これはデフォルトのダイアログです。ダイアログウインドウは × アイコンで閉じられ、移動、サイズ変更などもすることができます。</p>
</Content>
```

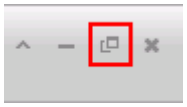
キャプションバー要素

キャプションバーは、**C1Dialog** ダイアログウィンドウの上部に表示され、表示は、従来のダイアログボックスに類似していますが、いくつかの違いがあります。左から右への順番で、キャプションバーには、アイコン、タイトルテキスト、および複数のボタンが格納されます。オプションで、固定、再読み込み、縮小／拡張、最小化、最大化、閉じるの各ボタンがあり、これらは、**Pin**、**Refresh**、**Minimize**、**Maximize**、**Close** の各ボタンを有効にしたときに、キャプションバーの右側に表示されます。**Title** プロパティの文字列を設定して、キャプションバー上のタイトルのテキストを指定できます。

下図に、キャプションバーに表示されるプロパティを示します。



下の表で、キャプションバー上の各要素について説明します。

要素	説明
Title	**C1Dialog** ダイアログウィンドウのキャプションバーに表示するテキストを取得または設定します。デフォルトでは、**Title** は空です。
Pin	**C1Dialog** ダイアログウィンドウを固定または固定解除するボタン。デフォルトでは、**Pin** は **Visible**ではありません。
Refresh	**ContentUrl** プロパティによって設定された **C1Dialog** ダイアログウィンドウのコンテンツを再ロードするボタン。デフォルトでは、**Visible**ではありません。
Minimize	**C1Dialog** ダイアログウィンドウを最小化するボタン。デフォルトでは、**Minimize** は **Visible**ではありません。
Maximize	ブラウザウィンドウ内で使用可能な全スペースに合わせて **C1Dialog** ダイアログウィンドウをサイズ変更するボタン。デフォルトでは、**Maximize** は **Visible**ではありません。
Close	**C1Dialog** ダイアログウィンドウを閉じるボタン。デフォルトでは、**Close** は **Visible**ではありません。
元のサイズに戻す 	ウィンドウを最小化または最大化するときに **C1Dialog** ダイアログの **Minimize** ボタンまたは **Maximize** ボタンの代わりに表示するボタン。このボタンをクリックすると、ウィンドウが前のサイズに復元されます。

C1Dialog の外観

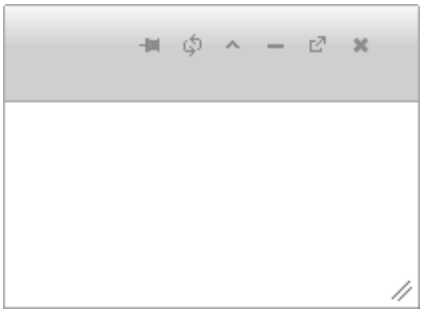
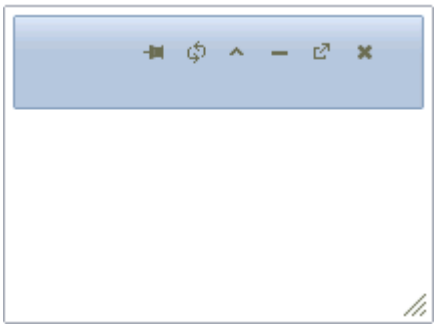
C1Dialog は、簡単にカスタマイズできるように設計されています。**C1Dialog** ダイアログウィンドウの外観の変更については、可能性は無限です。**C1Dialog** は、キャプションバー、コンテンツ領域、ステータスバーの各要素用のさまざまなスタイルや、組み込みテーマを備えています。

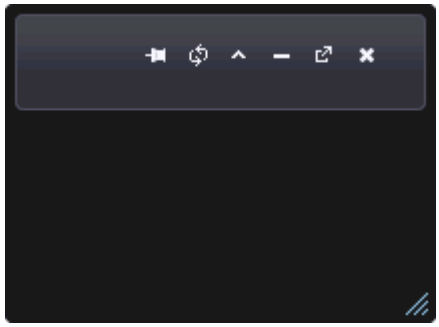
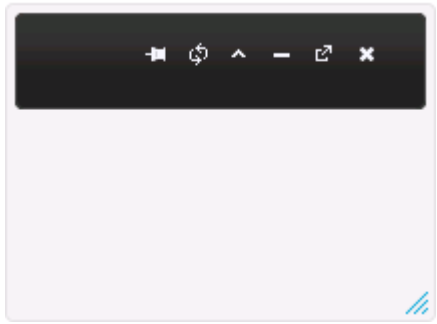
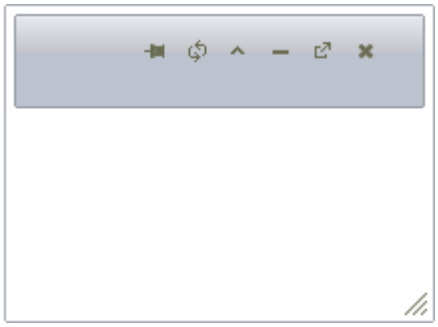
各ダイアログウィンドウ要素のプロパティスタイルに加え、**C1Dialog** は、CSS スタイルの完全サポートも備えているため、CSS スタイルを通じて各ダイアログウィンドウ要素をさらにカスタマイズできます。

テーマ

C1Dialog には、コントロールの外観を簡単に変更できるテーマが含まれています。コントロールには、コントロールの外観をアプリケーション用にカスタマイズできる複数の組み込みテーマが用意されています。[**C1Dialog タスク**]メニュー、**プロパティ**ウィンドウ、およびコードを使用して、簡単にテーマを変更できます。

Dialog for ASP.NET Web Forms には、以下のテーマが含まれています。

テーマ	プレビュー
Arctic	
Aristo	
Cobalt	

Midnight	
Rocket	
Sterling	

コンテンツテンプレート

ダイアログウィンドウのコンテンツ領域のコンテンツは、テンプレートを使用して制御できます。**C1Dialog** は、これらを使用して、テンプレートを **C1Dialog** コントロールのコンテンツ領域とステータスバー領域に適用できる特別なプロパティ、**Content** を備えています。

ダイアログウィンドウのコンテンツテンプレートは、ダイアログウィンドウをカスタマイズして、その概観をアプリケーションに統合したり、コンテンツをダイアログウィンドウのコンテンツ領域に追加したりする場合に便利です。

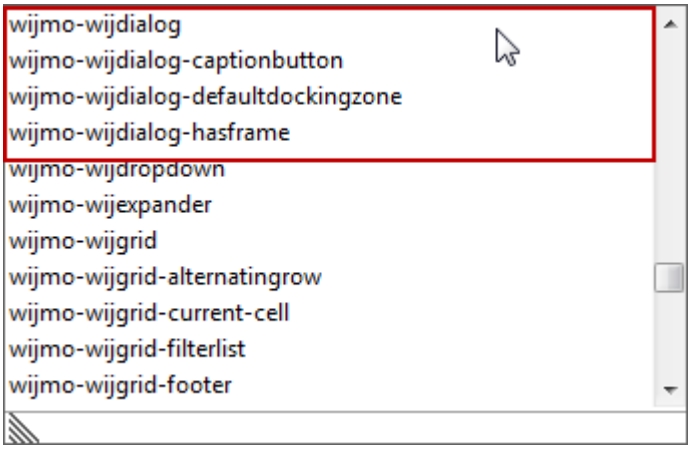
C1Dialog CSS セレクタ

CSS を使用して多くの **C1Dialog** 要素にスタイルを適用し、それらの外観を独特のものにすることができます。カスタマイズを簡素化するために、ComponentOne には、その6種類の組み込みテーマごとに CSS セレクタが組み込まれています。

枠、背景、テキスト、フォント、マージン、埋め込み、リスト、輪郭、表などの一般的な CSS プロパティを該当する CSS セレクタに適用できます。

一般に使用される個々の CSS セレクタとグループ化された CSS セレクタのリストについては、プロジェクトの **C1Dialog** コントロールを選択し、Visual Studio **プロパティ**ウィンドウで **CssClass** プロパティの横にあるドロップダウンリストを表示します。

C1Dialog CSSセレクタは以下の wijmo-wijdialog で開始します。



個々の CSS セレクタをグループとして組み合わせ、CSS セレクタをより具体的かつ強力なものにすることができます。

クライアント側の機能

Dialog for ASP.NET Web Forms コントロールには、非常に充実したクライアント側オブジェクトモデルがあります。そのメンバは、ほとんどがサーバー側コントロールのメンバと同じです。

C1Dialog コントロールが表示されると、クライアント側コントロールのインスタンスが自動的に生成されます。これは、サーバーにポストバックしなくても、**C1Dialog** コントロールのプロパティやメソッドにアクセスできるということです。

クライアント側コードを使用すれば、時間をかけて Web サーバーに情報を送信しなくても、Web ページに多くの機能を実装できます。クライアント側オブジェクトモデルを使用すると、Web サイトの効率を向上させることができます。

クライアント側イベント

Dialog for ASP.NET Web Formsには、複数のクライアント側イベントが含まれています。それらを利用すれば、ダイアログウィンドウのサイズ変更などの処理が行われたときに、**C1Dialog**コントロールを操作できます。

クライアント側イベントの表にリストされたサーバー側プロパティを使用して、特定のクライアント側イベントに反応する JavaScript 関数の名前を指定できます。たとえば、「Resize」という JavaScript 関数を割り当てて、ダイアログウィンドウがサイズ変更されたときに応答させるには、**OnClientResize** プロパティを resize に設定します。

下の表に、クライアントスクリプトで使用できるイベントを示します。これらのプロパティはサーバー側で定義されていますが、実際のイベントや各 JavaScript 関数用に宣言する名前はクライアント側で定義されます。

イベントのサーバー側プロパティ名	イベント名	**説明**
OnClientBeforeClose	beforeClose	コントロールが閉じる前に発生します。
OnClientClose	close	コントロールが閉じられたときに発生します。
OnClientDrag	drag	コントロールがドラッグされたときに発生します。
OnClientDragStart	dragStart	コントロールのドラッグが開始されたときに発生します。
OnClientDragStop	dragStop	コントロールのドラッグが停止されたときに発生します。
OnClientFocus	focus	コントロールがフォーカスされたときに発生します。
OnClientOpen	open	コントロールが開かれたときに発生します。
OnClientResize	resize	コントロールがサイズ変更されたときに発生します。
OnClientResizeStart	resizeStart	サイズ変更が開始されたときに発生します。
OnClientResizeStop	resizeStop	サイズ変更が停止したときに発生します。

C1Dialog クライアント側イベントの説明と構文の例については、「クライアント側リファレンス」でも確認できます。

タスク別ヘルプ

タスク別ヘルプセクションでは、Visual Studio ASP.NET 環境でのプログラミングに精通し、**Dialog for ASP.NET Web Forms** コントロールを全般的に理解しているユーザーを対象としています。

各トピックでは、**C1Dialog** コントロールを使用した特定のタスクのソリューションを示します。各トピックで概説されている手順に従うことによって、さまざまな **C1Dialog** 機能を使用したプロジェクトを作成できます。

タスク別ヘルプの各トピックでは、新しい ASP.NET プロジェクトを既に作成していることを前提としています。

コードによる C1Dialog コントロールの作成

コードによる **C1Dialog** の作成は簡単な作業です。このトピックでは、ページに **PlaceHolder**コントロールを追加し、重要なステートメントを追加し、**C1Dialog** コントロールをカスタマイズして、コントロールを **PlaceHolder** に追加します。

以下の手順を実行します。

1. デザインビューで、Visual Studio ツールボックスに移動し、ページに **PlaceHolder** コントロールを追加します。
2. ページをダブルクリックして **Page_Load** イベントを追加し、コードビューに切り替えます。
3. コードエディタで冒頭に次のステートメントを追加し、必要な名前空間をインポートします。

Visual Basic コードの書き方

```
Visual Basic
Imports C1.Web.Wijmo.Controls.C1Dialog
```

C# コードの書き方

```
C#
using C1.Web.Wijmo.Controls.C1Dialog;
```

4. 次のコードを **Page_Load** イベントに追加してコントロールを作成し、**C1Dialog** の高さと幅を設定して、コントロールを **PlaceHolder** に追加します。

Visual Basic コードの書き方

```
Visual Basic
' 新しいダイアログを作成
Dim C1D As New C1Dialog()
' コントロールのサイズを設定
C1D.Height = 200
C1D.Width = 200
' コントロールを PlaceHolder に追加
PlaceHolder1.Controls.Add(C1D)
```

C# コードの書き方

```
C#
//' 新しいダイアログを作成
C1Dialog C1D = new C1Dialog();
// コントロールのサイズを設定
C1D.Height = 200;
C1D.Width = 200;
```

```
// C1Dialog を PlaceHolder コントロールに追加
PlaceHolder1.Controls.Add(C1D);
```

クライアント側で C1Dialog コントロールの作成

コードを使用して **C1Dialog** コントロールを作成し、表示できます。このヘルプでは、標準 Button コントロールと標準 PlaceHolder コントロールをページ上に配置し、コードを使用して **C1Dialog** コントロールを作成し、button_click イベントを使用してそれを表示します。

以下の手順を実行します。

1. 標準 **PlaceHolder** コントロールと標準 **Button** コントロールを ASP.NET プロジェクトに追加します。
2. ソースビューに切り替えて、最初の タグに次のスクリプトを追加します。

ソースビュー

```
<!--jQuery References-->
  <script src="http://code.jquery.com/jquery-1.8.2.min.js" type="text/javascript"></script>
  <script src="http://code.jquery.com/ui/1.9.1/jquery-ui.min.js" type="text/javascript"></script>
    <asp:PlaceHolder runat="server">
  <script type="text/javascript">
    $(document).ready(function () {
      $("#<%=btnShowDialog.ClientID %>").click(function () {
        $("#<%=C1Dialog1.ClientID %>").c1dialog("open");
        return false;
      });
    });
  </script>
  </asp:PlaceHolder>
```

3. そのままソースビューで、Button コントロールと PlaceHolder コントロールのマークアップを見つけて編集します。その結果、次のようになります。

ソースビュー

```
<asp:Button ID="btnShowDialog" runat="server" Text="実行時に作成されたダイアログを表示" />
  <asp:PlaceHolder ID="c1Dialog1" runat="server"></asp:PlaceHolder>
```

4. ソースビューページを右クリックし、メニューから[**コードの表示**]を選択します。
5. コードビューで、ページの一番上に以下のステートメントを追加します。

C# コードの書き方

C#

```
using C1.Web.Wijmo.Controls.C1Dialog;
```

Visual Basic コードの書き方

Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1Dialog
```

6. 次のコードを追加して、button_click イベントを作成し、**C1Dialog** コントロールを作成します。

C# コードの書き方

C#

```
protected void btnShow_Click(object sender, EventArgs e)
```

```
    {
        C1Dialog dlg = new C1Dialog();
        dlg.Title = "実行時に作成されたダイアログ";
        dlg.Modal = true;
        dlg.Content = new MyTemplateClass();
        C1Dialog1.Controls.Add(dlg);
    }
```

Visual Basic コードの書き方

```
Visual Basic
Protected Sub btnShow_Click(sender As Object, e As EventArgs) Handles btnShow.Click
    Dim dlg As C1Dialog = New C1Dialog()
    dlg.Title = "実行時に作成されたダイアログ"
    dlg.Modal = True
    dlg.Content = New MyTemplateClass()
    C1Dialog1.Controls.Add(dlg)
  End Sub
```

7. 次のコードで新しいクラスを開始します。これは **C1Dialog** コントロールのテンプレートです。

C# コードの書き方

```
C#
public class MyTemplateClass : ITemplate
    {
        public void InstantiateIn(Control container)
        {
            Label label = new Label();
            label.ID = "lblmyLabel";
            label.Text = "これは C1Dialog です。";
            Button btnOK = new Button();
            btnOK.Text = "OK";
            container.Controls.Add(label);
            container.Controls.Add(btnOK);
        }
```

Visual Basic コードの書き方

```
Visual Basic
Public Class MyTemplateClass
  Implements ITemplate
  Public Sub InstantiateIn(container As Control) Implements ITemplate.InstantiateIn
    Dim label As New Label()
    label.ID = "lblmyLabel"
    label.Text = "これは C1Dialog です。"
    Dim btnOK As New Button()
    btnOK.Text = "OK"
    container.Controls.Add(label)
    container.Controls.Add(btnOK)
  End Sub
End Class
```

8. アプリケーションを実行すると、次のような表示になります。

9. ボタンをクリックすると、**C1Dialog** は次の図のように表示されます。

コンテンツ領域でのコンテンツの設定

以下の各手順では、コンテンツテンプレートのコンテンツ、外部 Web サイトコンテンツ、カスタム HTML コンテンツ、および部分レンダリングコンテンツなど、さまざまな種類のコンテンツを **C1Dialog** に追加する方法について示します。

デザイン時のコンテンツの設定

デザインビューまたはソースビューのいずれかで、デザイン時にコンテンツ領域でコンテンツを簡単に設定できます。

コンテンツ領域でコンテンツを表示するには、以下の手順を実行します。

デザインビューの場合

以下の手順を実行します。

1. プロジェクトのダイアログコントロールを選択します。
2. ダイアログコンテンツ領域にカーソルを置きます。
3. コンテンツ領域に表示するコンテンツを入力するか、または貼り付けます。
4. [F5]を押して、プロジェクトを実行します。

ソースビューの場合

以下の手順を実行します。

1. プロジェクトのソースビューに移動します。
2. <CaptionButtons/> の終了タグの後に、次のタグセットを挿入します。

ソースビュー
<Content> </Content>

3. `<Content>` タグの間にテキストを挿入します。
4. [F5]を押してプロジェクトを実行し、ダイアログウィンドウでテキストを確認します。

コンテンツ領域での外部コンテンツの表示

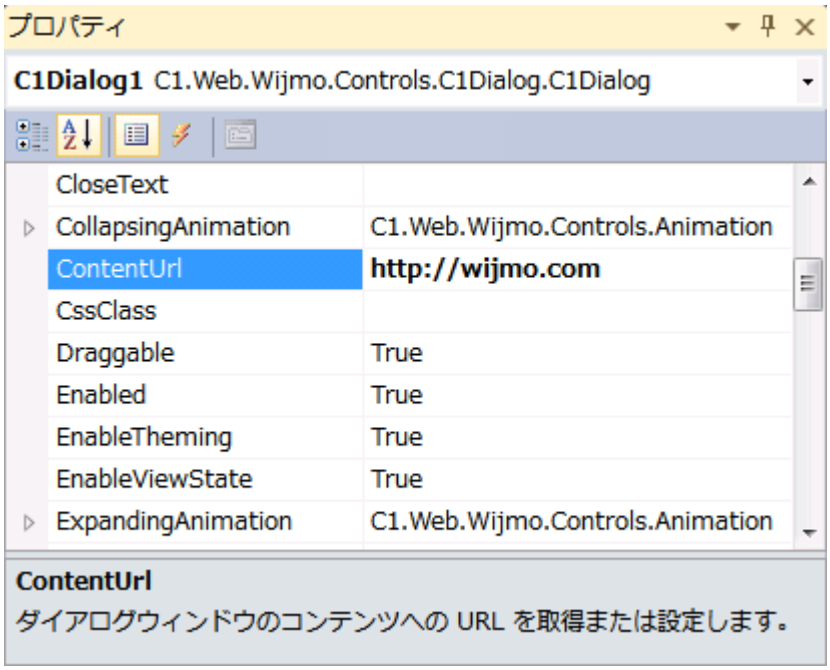
C1Dialog では、ダイアログコンテンツ領域に外部 URL コンテンツを表示できます。プロジェクト内部の他の Web ページからのコンテンツ、またはプロジェクト外部の URL からのコンテンツをダイアログコンテンツ領域に表示することができます。

コンテンツ領域で URL コンテンツを表示するには、以下の手順を実行します。

デザインビューの場合

以下の手順を実行します。

1. Visual Studio のメニューから、[**表示**]→[**プロパティ**]を選択します。プロパティウィンドウの上部にあるドロップダウンリストから、**C1Dialog** を選択します。
2. **ContentURL** プロパティまで下方へスクロールします。
3. **ContentURL** プロパティを、プロジェクト内に表示する URL に設定します。



4. [F5]を押して、プログラムを実行します。選択した Web ページがダイアリグウィンドウに表示されます。

ソースビューの場合

ソースビューで ContentURL を設定するには、以下の手順を実行します。

1. ContentUrl="https://www.grapecity.co.jp/developer/wijmo"
   `<asp:Content>` タグ内のマークアップは次のようになります。

ソースビュー
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent"> <cc1:C1Dialog ID="C1Dialog1" runat="server" Width="760px" Height="460px" Stack="True" CloseText="True" MaintainVisibilityOnPostback="False" CloseOnEscape="False" ContentURL="https://www.grapecity.co.jp/developer/wijmo">

2. [F5]または〈デバッグ開始〉を押して、プログラムを実行します。Web ページがダイアログウィンドウに表示されます。

コードの場合

次のコードを Page_Loadイベントに追加し、**C1Dialog** コントロールで外部コンテンツが表示されるように **ContentUrl**プロパティを設定します。

Visual Basic コードの書き方

Visual Basic
Me.C1Dialog1.ContentUrl = "https://www.grapecity.co.jp/developer/wijmo"

C# コードの書き方

C#

```
this.C1Dialog1.ContentUrl = "https://www.grapecity.co.jp/developer/wijmo";
```

カスタム HTML コンテンツの作成

カスタム HTML コンテンツを設定してコンテンツ領域で表示できます。これはプロジェクトのソースビューで実行されます。このタスク別ヘルプでは、ユーザーがテキストボックスにテキストを入力し、それを書式設定された、ポップアップダイアログボックスに表示できる HTML マークアップを挿入します。

以下の手順を実行します。

1. <cc1:C1Dialog> タグ内に、次のタグセットを挿入します。

   ソースビュー

   ```
   <Content></Content>
   ```

2. <Content> タグの間に次のマークアップを挿入します。

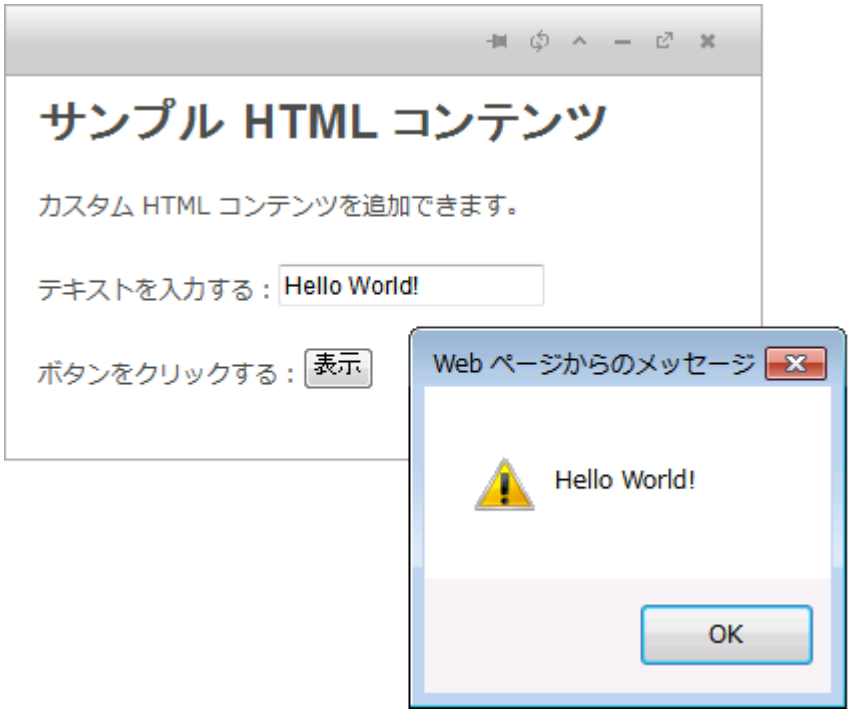
   ソースビュー

   ```
   <h2>サンプル HTML コンテンツ</h2>
   <p>カスタム HTML コンテンツを追加できます。
   <p>テキストを入力する: <input type=text id="input1" />
   <p>ボタンをクリックする: <input type="button" value="表示"
   onclick="alert(document.getElementById('input1').value)" />
   ```

3. [F5]を押して、プロジェクトを実行します。**C1Dialog** は、次のような図になります。

   ユーザーがテキストを入力して〈**表示**〉ボタンをクリックすると、次のダイアログボックスが表示されます。



部分レンダリングの使用

C1Dialog は、Partial Page Rendering（PPR）技術をサポートしており、カスタムの JavaScript を使用せずに、部分ページの更新が可能です。このトピックでは、PPR を使用するプロジェクトを作成する手順について説明します。

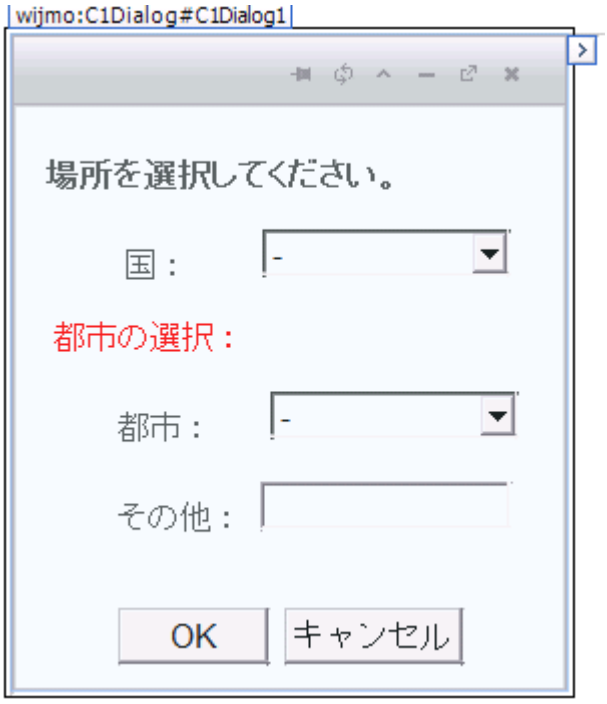
1. Visual Studio のメニューから[**表示**]→[**プロパティ**]を選択し、ウィンドウの上部にあるドロップダウンリストから C1Dialog を選択します。
2. 以下の各プロパティを設定します。
   - **Height** プロパティを **325px** に設定します。
   - **Width** プロパティを **275px** に設定します。
3. ソースビューに切り替え、次のマークアップを <cc1:C1Dialog> タグに追加します。

ソースビュー

```
<Content> <br /> <strong>場所を選択してください。</strong>
<br /><br />
        国:
        
<asp:asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True" Width="123px">
<asp:ListItem Selected="True">-</asp:ListItem>
<asp:ListItem>UK</asp:ListItem>
<asp:ListItem>USA</asp:ListItem>
<asp:ListItem>Russia</asp:ListItem>
<asp:ListItem>Canada</asp:ListItem>
</asp:DropDownList><br /><br />
 <asp:asp:Label ID="Label1" runat="server" ForeColor="Red">都市の選択:</asp:Label><br /><br />
        都市:    
<asp:DropDownList ID="DropDownList2" runat="server" Width="123px">
<asp:ListItem Selected="True">-</asp:ListItem>
</asp:DropDownList><br />         その他:
<asp:TextBox ID="TextBox1" runat="server" EnableViewState="False" Width="117px""></asp:TextBox><br
/>
<br /><br />
```

```
<input id="Button1" style="width:75px" type="button"
value="OK" onclick="__doPostBack('','');" />
<input id="Button2" style="width:89px" type="button" value="キャンセル" onclick="hide()" /><br />
</Content>
```

4. デザインビューに切り替えます。フォームは次のような表示になります。



5. Web ページをダブルクリックして、Load イベントのイベントハンドラを作成します。**Page_Load** イベントに次のコードを入力して、コントロールを初期化します。

Visual Basic コードの書き方

Visual Basic

```
[String].Format("javascript:openWindow({0});", C1Window1.ClientID)
If Me.IsPostBack Then
Dim dl As DropDownList = DirectCast(C1Window1.FindControl("DropDownList1"),
DropDownList)
Dim dlc As DropDownList = DirectCast(C1Window1.FindControl("DropDownList2"),
DropDownList)
Dim tb As TextBox = DirectCast(C1Window1.FindControl("TextBox1"), TextBox)
If dl.Text <> "-" AndAlso (dlc.Text <> "-" OrElse tb.Text <> "") Then
Dim text As String = ""
If dlc.Text <> "-" Then
text += dlc.Text
Else
text += tb.Text
End If
text += ", " + dl.Text
Label2.Text = text
End If
Else
Label2.Text = ""
```

```
End If
```

C# コードの書き方

C#

```
protected void Page_Load(object sender, EventArgs e)
String.Format("javascript:openWindow({0});", C1Dialog1.ClientID);
if (this.IsPostBack)
            {
DropDownList dl = (DropDownList)C1Dialog1.FindControl("DropDownList1");
DropDownList dlc = (DropDownList)C1Dialog1.FindControl("DropDownList2");
TextBox tb = (TextBox)C1Dialog1.FindControl("TextBox1");
if (dl.Text != "-" && (dlc.Text != "-" || tb.Text != ""))
                {
string text = "";
if (dlc.Text != "-")
                    {
text += dlc.Text;
                    }
else
                    {
text += tb.Text;
                    }
text += ", " + dl.Text;
Label1.Text = text;
                }
            }
else
            {
Label1.Text = "";
            }
     }
```

6. 以下の **SelectedIndexChanged** イベントをコードに追加します。

Visual Basic コードの書き方

Visual Basic

```
Protected Sub DropDownList1_SelectedIndexChanged(ByVal sender As Object, ByVal e
As EventArgs)
Dim l As Label = DirectCast(C1Cialog1.FindControl("Label1"), Label)
Dim dl As DropDownList = DirectCast(C1Dialog1.FindControl("DropDownList1"),
DropDownList)
Dim dlc As DropDownList = DirectCast(C1Dialog1.FindControl("DropDownList2"),
DropDownList)
dlc.Items.Clear()
dlc.Items.Add(New ListItem("-"))
If dl.Text <> "-" Then
l.Text = "都市の選択:" + dl.Text
Select Case dl.Text
Case "UK"
dlc.Items.Add(New ListItem("London"))
dlc.Items.Add(New ListItem("Birmingham"))
```

```
dlc.Items.Add(New ListItem("Leeds"))
dlc.Items.Add(New ListItem("Glasgow"))
dlc.Items.Add(New ListItem("Glasgow"))
dlc.Items.Add(New ListItem("Sheffield"))
dlc.Items.Add(New ListItem("Bradford"))
dlc.Items.Add(New ListItem("Edinburgh"))
dlc.Items.Add(New ListItem("Liverpool"))
Exit Select
Case "USA"
dlc.Items.Add(New ListItem("New York, New York"))
dlc.Items.Add(New ListItem("Los Angeles, California"))
dlc.Items.Add(New ListItem("Chicago, Illinois"))
dlc.Items.Add(New ListItem("Houston, Texas"))
dlc.Items.Add(New ListItem("Philadelphia, Pennsylvania"))
dlc.Items.Add(New ListItem("Phoenix, Arizona"))
dlc.Items.Add(New ListItem("San Diego, California"))
dlc.Items.Add(New ListItem("Dallas, Texas"))
dlc.Items.Add(New ListItem("Detroit, Michigan"))
Exit Select
Case "Russia"
dlc.Items.Add(New ListItem("Moscow"))
dlc.Items.Add(New ListItem("Chelyabinsk"))
dlc.Items.Add(New ListItem("Ekaterinburg"))
dlc.Items.Add(New ListItem("Irkutsk"))
dlc.Items.Add(New ListItem("St. Petersburg"))
dlc.Items.Add(New ListItem("Volgograd"))
dlc.Items.Add(New ListItem("Petrozavodsk"))
dlc.Items.Add(New ListItem("Nizhni Novgorod"))
dlc.Items.Add(New ListItem("Novosibirsk"))
Exit Select
Case "Canada"
dlc.Items.Add(New ListItem("Toronto, Ontario"))
dlc.Items.Add(New ListItem("Montreal, Quebec"))
dlc.Items.Add(New ListItem("Vancouver, British Columbia"))
dlc.Items.Add(New ListItem("Calgary, Alberta"))
dlc.Items.Add(New ListItem("Edmonton, Alberta"))
dlc.Items.Add(New ListItem("Ottawa - Gatineau, Ontario/Quebec"))
Exit Select
Case Else
Exit Select
End Select
End If
End Sub
```

C# コードの書き方

C#

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
Label l = (Label)C1Dialog1.FindControl("Label1");
DropDownList dl = (DropDownList)C1Dialog1.FindControl("DropDownList1");
DropDownList dlc = (DropDownList)C1Dialog1.FindControl("DropDownList2");
```

```
dlc.Items.Clear();
dlc.Items.Add(new ListItem("-"));
if (dl.Text != "-")
   {
l.Text = "都市の選択:" + dl.Text;
switch (dl.Text) {
case "UK":
dlc.Items.Add(new ListItem("London"));
dlc.Items.Add(new ListItem("Birmingham"));
dlc.Items.Add(new ListItem("Leeds"));
dlc.Items.Add(new ListItem("Glasgow"));
dlc.Items.Add(new ListItem("Glasgow"));
dlc.Items.Add(new ListItem("Sheffield"));
dlc.Items.Add(new ListItem("Bradford"));
dlc.Items.Add(new ListItem("Edinburgh"));
dlc.Items.Add(new ListItem("Liverpool"));
break;
case "USA":
dlc.Items.Add(new ListItem("New York, New York"));
dlc.Items.Add(new ListItem("Los Angeles, California"));
dlc.Items.Add(new ListItem("Chicago, Illinois"));
dlc.Items.Add(new ListItem("Houston, Texas"));
dlc.Items.Add(new ListItem("Philadelphia, Pennsylvania"));
dlc.Items.Add(new ListItem("Phoenix, Arizona"));
dlc.Items.Add(new ListItem("San Diego, California"));
dlc.Items.Add(new ListItem("Dallas, Texas"));
dlc.Items.Add(new ListItem("Detroit, Michigan"));
break;
case "Russia":
dlc.Items.Add(new ListItem("Moscow"));
dlc.Items.Add(new ListItem("Chelyabinsk"));
dlc.Items.Add(new ListItem("Ekaterinburg"));
dlc.Items.Add(new ListItem("Irkutsk"));
dlc.Items.Add(new ListItem("St. Petersburg"));
dlc.Items.Add(new ListItem("Volgograd"));
dlc.Items.Add(new ListItem("Petrozavodsk"));
dlc.Items.Add(new ListItem("Nizhni Novgorod"));
dlc.Items.Add(new ListItem("Novosibirsk"));
break;
case "Canada":
dlc.Items.Add(new ListItem("Toronto, Ontario"));
dlc.Items.Add(new ListItem("Montreal, Quebec"));
dlc.Items.Add(new ListItem("Vancouver, British Columbia"));
dlc.Items.Add(new ListItem("Calgary, Alberta"));
dlc.Items.Add(new ListItem("Edmonton, Alberta"));
dlc.Items.Add(new ListItem("Ottawa - Gatineau, Ontario/Quebec"));
break;
default:
break;
          }
   }
}
```

このトピックは、次のことを示します。

アプリケーションを実行して、以下の手順を実行します。

1. 最初のドロップダウンリストから国を選択します。
2. 次の選択可能な都市のドロップダウンリストはサーバーから更新されることに注意してください。
3. ドロップダウンリストから都市を選択して、〈OK〉をクリックします。

Web ページ上に都市の名前が反映されます。

コントロールのカスタマイズ

数多くの **C1Dialog** 要素をカスタマイズできます。このセクションでは、コントロールのコンテンツ領域、キャプションバー、および幅と高さをカスタマイズする方法を学びます。

前景色と背景色のカスタマイズ

このトピックでは、デザインビューとソースビューの両方でコンテンツ領域の前景色と背景色を設定する方法について説明します。

デザインビューの場合

以下の手順を実行します。

1. **C1Dialog** コントロールを ASP.NET プロジェクトに追加します。
2. **C1Dialog** コントロールを選択します。
3. Visual Studio の[タスク]ツールバーで、〈**前景色**〉ボタンと〈**背景色**〉ボタンを探します。
4. 〈**前景色**〉ボタンをクリックしてテキストの色を選択します。
5. 〈**背景色**〉ボタンをクリックして背景色を選択します。
6. **C1Dialog** コントロールにテキストを入力します。
7. [F5]または〈**デバッグ開始**〉を押して、プロジェクトを実行します。**C1Dialog** ウィンドウは、次の図のようになるはずです。

ソースビューの場合

以下の手順を実行します。

1. <cc1:C1Dialog> タグの間に次の .html マークアップを挿入します。

ソースビュー

```
style="color:#99CCFF; background-color: #003366"
<Content>
<h2>
C1Dialog<</h2>
<p>これはデフォルトのダイアログです。ダイアログウインドウは × アイコンで閉じられ、 移動、サイズ変更などもすることができます。</p>
</Content>
```

2. プログラムを実行して、書式設定された**C1Dialog** とテキストを確認します。

フォントスタイルのカスタマイズ

このトピックでは、デザインビューとソースビューでダイアログウィンドウのフォントスタイルをカスタマイズする手順について説明します。

デザインビューの場合

以下の手順を実行します。

1. **C1Dialog** コントロールを選択します。
2. Visual Studio ツールバーの[フォント]ドロップダウンリストに移動します。
3. リストからフォントを選択します。このトピックでは、**Blackadder ITC** を選択します。
4. **C1Dialog** コントロールのコンテンツ領域にテキストを入力するか、または貼り付けます。
5. [F5]または〈**デバッグ開始**〉を押して、プログラムを実行します。ダイアログウィンドウのテキストが選択したフォントで表示されます。

ソースビューの場合

以下の手順を実行します。

1. プロジェクトのソースビューに移動します。
2. <cc1:C1Dialog> タグを探します。次のマークアップと <Content> タグを <cc1:C1Dialog> タグに追加します。

ソースビュー

```
Style="font-family:Blackadder ITC"
```

```
<Content>  </Content>
```

3. <Content> タグの間にコンテンツを入力するか、または貼り付けます。
4. [F5]または〈**デバッグ開始**〉を押して、プログラムを実行します。**C1Dialog** ウィンドウのテキストが選択したフォントで表示されます。

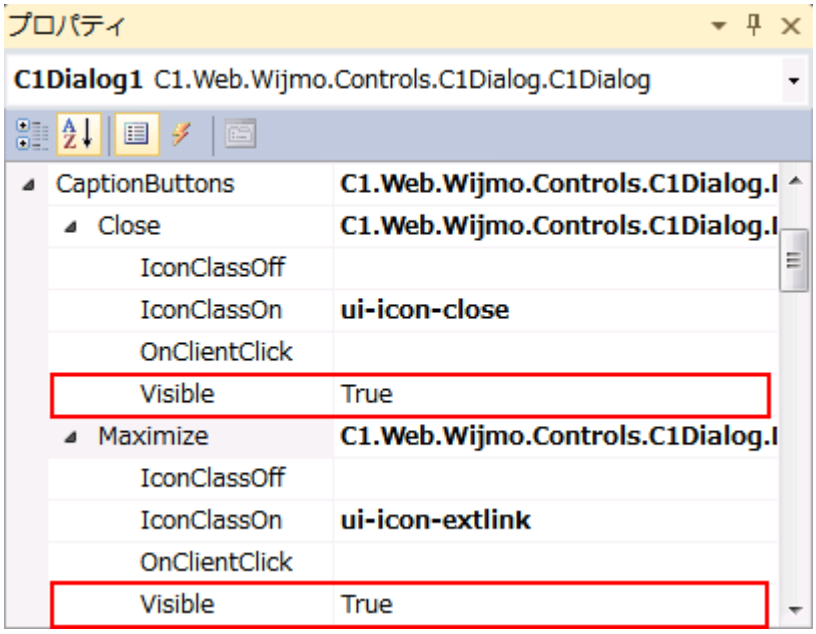
キャプションバーのカスタマイズ

このトピックでは、デザインビューとソースビューの両方でキャプションバーに表示されるボタンをカスタマイズする手順について説明します。

デザインビューの場合

以下の手順を実行します。

1. **C1Dialog** プロパティウィンドウに移動して、**CaptionButtons** プロパティを探します。
2. 矢印を使用してプロパティノードを展開します。
3. 各**DialogCaptionButtons** ノードを展開してそれぞれのプロパティを確認します。
4. 各**DialogCaptionButton** の **Visible** プロパティを探して、それを **True** または **False** に設定します。**プロパティ**ウィンドウは、次の図のようになるはずです。



5. [F5]または〈**デバッグ開始**〉を押して、プログラムを実行します。
False に設定したキャプションバーボタン項目は表示されなくなることに注意してください。

ソースビューの場合

以下の手順を実行します。

1. <cc1:C1Dialog> タグの間に次のマークアップを挿入します。

ソースビュー

```
<CaptionButtons>
<Pin IconClassOn="ui-icon-pin-w" IconClassOff="ui-icon-pin-s"></Pin>
<Refresh IconClassOn="ui-icon-refresh" Visible="False"></Refresh>
<Minimize IconClassOn="ui-icon-minus"></Minimize>
```

```
<Maximize IconClassOn="ui-icon-extlink"></Maximize>
<Close IconClassOn="ui-icon-close" Visible="False"></Close>
</CaptionButtons>
```

2. [F5]または〈**デバッグ開始**〉を押して、プログラムを実行します。**Visible** プロパティを False に設定したキャプションバーボタンは表示されないことを確認してください。

C1Dialog の幅と高さのカスタマイズ

このトピックでは、**C1Dialog** の幅と高さをカスタマイズする手順について説明します。デザインビュー、ソースビュー、およびコードで、または実行時にカスタマイズできます。

デザインビューの場合

以下の手順を実行します。

1. **C1Dialog プロパティ**ウィンドウに移動して、**Height** プロパティを探します。
2. コントロールの高さをピクセル単位で設定します（このトピックでは 400px に設定）。
3. リストで **Width** プロパティを探します。コントロールの幅をピクセル単位で設定します（このトピックでは 400px に設定）。
4. [F5]を押してプログラムを実行し、加えた変更を確認します。

ソースビューの場合

以下の手順を実行します。

1. プロジェクトのソースビューに移動します。
2. **Height** と **Width** プロパティは、両方とも <cc1:C1Dialog> タグ内にあります。

ソースビュー

```
<cc1:C1Dialog ID="C1Dialog1" runat="server" CloseOnEscape="False"
Height="400px" Width="400px">
```

3. **Height** と **Width** プロパティを 400px に設定します。
4. [F5]を押して、プログラムを実行します。

コードの場合

次のコードを Page_Load イベントに追加し、**C1Dialog** コントロールの Height と Width プロパティを設定します。

Visual Basic コードの書き方

Visual Basic

```
Me.C1Dialog1.Height = 150
Me.C1Dialog1.Width = 150
```

C# コードの書き方

C#

```
this.C1Dialog1.Height = 150;
this.C1Dialog1.Width = 150;
```

実行時の場合

以下の手順を実行します。

実行時にも、ダイアログウィンドウの高さと幅を変更できます。

1. [F5]を押して、プログラムを実行します。
2. **C1Dialog** コントロールの端部にマウスポインタを置きます。
3. 矢印を使用して **C1Dialog** コントロールのサイズを変更します。

C1Dialog CSS セレクタでの作業

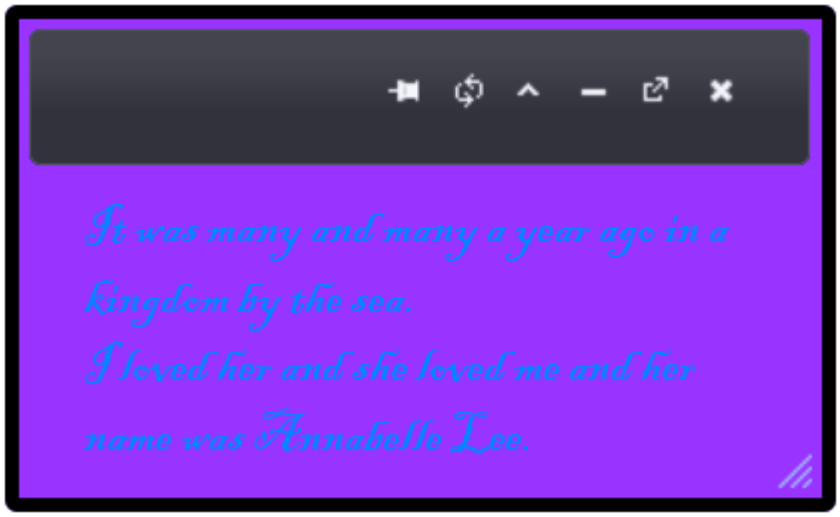
C1Dialog は、コントロールの外観を完全にカスタマイズできる CSS スタイル設定をサポートします。このトピックでは、**C1Dialog** コントロールに CSS スタイル設定を適用する手順について説明します。

1. デザインビューで、Visual Studio のメニューから、[**表示**]→[**プロパティ**]を選択します。プロパティウィンドウの上部にあるドロップダウンリストから **C1Dialog** を選択して、**CssClass** プロパティを探します。
2. ドロップダウンリストを使用して適切な CSS セレクタ(このトピックでは **wijmo-wijdialog** セレクタ)を選択します。
3. ソースビューに切り替えて、`<asp:Content>` タグの最初のセットを探します。
4. `<asp:Content>` タグの間に`<style type="text/css"></style>` タグを挿入します。このタグのセットを使って、コントロールに CSS スタイル設定を追加できます。
5. `<style>` タグの間に以下のスクリプトを挿入して、コントロールに CSS スタイル設定を追加します。

ソースビュー

```
.wijmo-wijdialog
{
color:#0088FF;
background:#9933FF;
border-color: #000000;
border-width:thick;
font-family:Blackadder ITC;
font-size:large;
}
```

6. プログラムを実行します。**C1Dialog** コントロールは次の図のように表示されます。



モーダルダイアログオプションの使用

C1Dialog をモーダルダイアログボックスにすることができます。この子ウィンドウは、閉じられるまで現在のアプリケーションを制御します。モーダルオプションでは、**C1Dialog** を使用して重要な情報を伝えたり、ユーザー操作を要求したりできます。このセクションでは、モーダルダイアログボックス、アラートダイアログボックス、および確認ダイアログボックスを作成する方法を学びます。

モーダルダイアログボックスの作成

このトピックでは、デザインビュー、ソースビュー、およびコードで、**C1Dialog** をモーダルダイアログボックスとして設定する手順について説明します。

デザインビューの場合

以下の手順を実行します。

1. **C1Dialog** プロパティウィンドウで、**Modal** プロパティまでスクロールダウンします。
2. ドロップダウンメニューを使用して、プロパティを **True** に設定します。
3. [F5]または〈**デバッグ開始**〉を押して、プログラムを実行します。モーダルウィンドウを閉じるまでは、その他のページを制御できないことに注意してください。

ソースビューの場合

以下の手順を実行します。

1. `<cc1:C1Dialog>` タグ内に次のマークアップを追加します。

ソースビュー
Modal="True"

2. プログラムを実行して、モーダルウィンドウがその他の Web ページに及ぼす効果を確認します。

コードの場合

次のコードを **Page_Load** にイベントに追加し、**C1Dialog** の**Modal** プロパティを設定します。

Visual Basic コードの書き方

Visual Basic
Me.C1Dialog1.Modal = True

C# コードの書き方

C#
this.C1Dialog1.Modal = true;

アラートダイアログボックスの作成

C1Dialog を使用して、アップロードが完了したときなど、ユーザーが認識する必要がある情報をユーザーに通知できます。このトピックでは、アップロードが完了したことをユーザーに知らせるモーダルアラートウィンドウを作成します。

以下の手順を実行します。

1. プロジェクトのソースビューに移動します。

2. <asp:Content> タグの最初のペアを探します。それらのタグペアの間に次のスクリプトを挿入します。

ソースビュー

```
<script type="text/javascript">
function btnClick() {
$(this).c1dialog("close");
    }
</script>
```

3. <cc1:C1Dialog> タグを探して、runat="server"プロパティの後に次のマークアップを挿入します。

ソースビュー

```
Width="450px" Height="285px" Modal="true"
Stack="True" CloseText="Close" Title="ファイルのアップロード">
<Content>
<p>
<span class="ui-icon ui-icon-circle-check"></span>ファイルは正常にアップロードされました！
</p>
</Content>
<ExpandingAnimation Duration="400">
</ExpandingAnimation>
<Buttons>
<cc1:DialogButton OnClientClick="btnClick" Text="OK" />
</Buttons>
<CaptionButtons>
<Pin Visible="False" />
<Refresh Visible="False" />
<Toggle Visible="False" />
<Minimize Visible="False" />
<Maximize Visible="False" />
</CaptionButtons>
<CollapsingAnimation Duration="300" />
```

4. プログラムを実行します。アラートウィンドウは、次の図のようになるはずです。

5. 〈閉じる〉ボタン以外のキャプションバーボタンはすべて無効になることに注意してください。

確認ダイアログボックスの作成

C1Dialog は、ごみ箱を空にするなどのユーザーの操作を確認するためにも使用できます。このトピックでは確認ダイアログウィンドウを作成します。

以下の手順を実行します。

1. プロジェクトのソースビューに移動します。
2. <cc1:C1Dialog> タグの最初のペアを探して、runat="server"プロパティの後に次のスクリプトを挿入します。

ソースビュー

```
<script type="text/javascript">
function delClick() {
$(this).c1dialog("close");
      }
function cancelClick() {
$(this).c1dialog("close");
      }
</script>
</asp:Content>
```

3. <cc1:C1Dialog> タグを探します。runat="server"の後ろに次のマークアップを挿入します。

ソースビュー

```
<cc1:C1Dialog ID="C1Dialog1" runat="server"Title="アップロード完了"
Modal="True" CloseText="Close">
<Content>
<p>
<span class="ui-icon ui-icon-circle-check"></span>ファイルは正常にアップロードされました！
</p>
</Content>
<ExpandingAnimation Duration="400" />
<CollapsingAnimation Duration="300" />
<Buttons>
<cc1:DialogButton onclientclick="delClick" text="削除" />
<cc1:DialogButton onclientclick="cancelClick" text="キャンセル" />
</Buttons>
<CaptionButtons>
<Pin Visible="false" />
<Refresh Visible="False" />
<Toggle Visible="False" />
<Minimize Visible="False" />
<Maximize Visible="False" />
</CaptionButtons>
```

4. プログラムを実行します。確認ダイアログウィンドウは、次の図のようになるはずです。

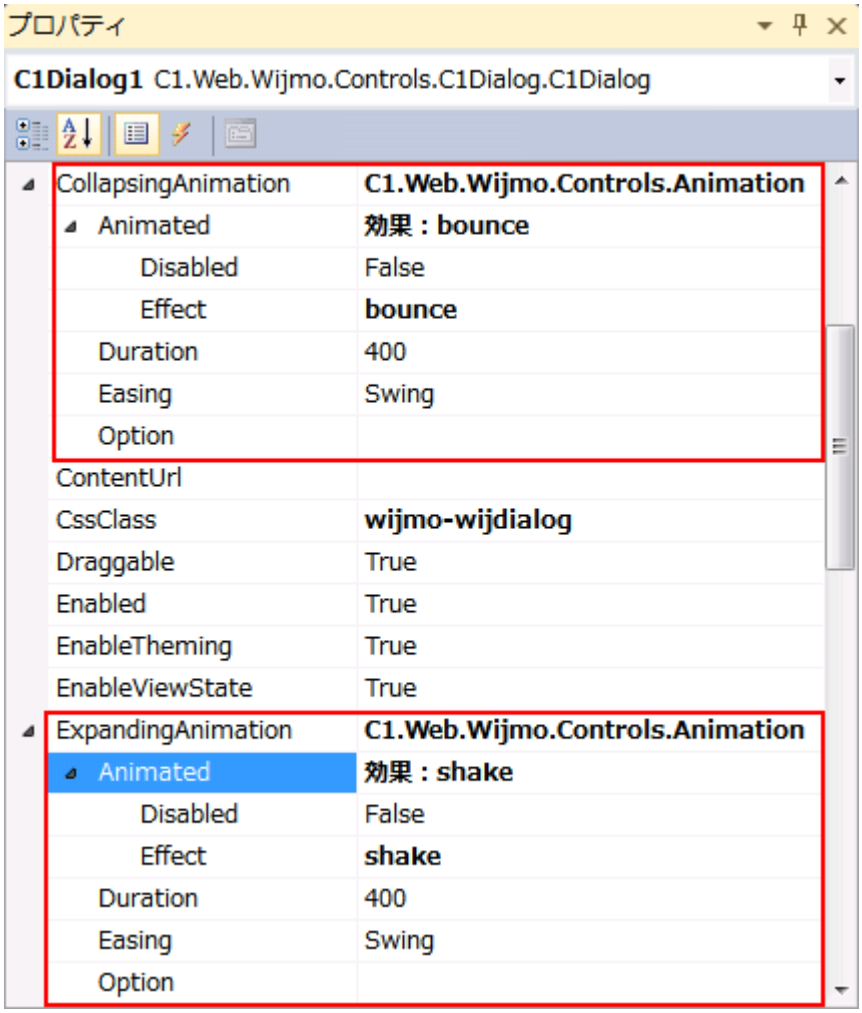
アニメーション化

デザインビューまたはソースビューのいずれかで**C1Dialog** プロパティにアニメーション効果を追加できます。**C1Dialog** のアニメーション効果を変更できる **ClosingAnimation**、**ExpandingAnimation**、**Hide**、**Show** という4つのプロパティが用意されています。このトピックでは、デザインビューとソースビューでアニメーション効果を適用する手順について説明します。

デザインビューの場合

以下の手順を実行します。

1. Visual Studio のメニューから[**表示**]→[**プロパティ**]を選択し、ウィンドウの上部にあるドロップダウンリストから **C1Dialog** を選択します。
2. リストで **ClosingAnimation** プロパティを探します。
3. 矢印を使用してプロパティノードを展開します。
4. **ClosingAnimation:Animated:Effect** に移動して bounce を入力します。
5. **ExpandingAnimation** プロパティを探し、矢印を使用してプロパティノードを展開します。
6. **ExpandingAnimation:Animation:Effect** に移動して **shake** を入力します。**プロパティ**ウィンドウは、次の図のようになるはずです。



7. [F5]または〈**デバッグ開始**〉を押して、プログラムを実行します。

ソースビューの場合

1. `<cc1:C1Dialog>` タグ内に次のマークアップを追加して、Show および Hide アニメーション効果を設定します。

ソースビュー

```
Show="pulsate" Hide="explode"
```

2. <Content> タグの前に次のマークアップを追加して、Expanding および Collapsing アニメーション効果を設定します。

ソースビュー

```
<CollapsingAnimation>
<Animated Effect="bounce" />
</CollapsingAnimation>
<ExpandingAnimation>
<Animated Effect="shake" />
</ExpandingAnimation>
```

3. プログラムを実行して、プロジェクトに適用したアニメーション効果を確認します。