ProgressBar for ASP.NET Web Forms

2018.04.11 更新

グレープシティ株式会社

ProgressBar for ASP.NET Web Forms

目次

製品の概要

ProgressBar for ASP.NET Web Forms を Web プロジェクトに追加することによって、エンドユーザーに通知し続けます。**ProgressBar for ASP.NET Web Forms** は、ゲージとインジケータの両方の役割を果たすグラフィカルユーザーインタフェース要素です。これによって、ユーザーは処理の進捗状況を確認でき、同時に、その処理がまだ実行中であることもわかります。このインタフェースを使用して、進行中であることを示す静的なシンボルを表示したり、現在実行されている処理の進捗状況を表示したりできます。**ProgressBar for ASP.NET Web Forms** では、5つの組み込み視覚スタイル、カスタマイズ可能なラベルとツールチップ、数十種類のアニメーションなどが利用できます。

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

C1ProgressBar の主な特長として、次の事項が挙げられます。

- **アニメーション**
  30 種類以上の組み込みアニメーションから選択して、プログレスバーの進行時の効果を魅力あるものにします。各アニメーションの実行速度と頻度は、選択することが可能です。
- **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。
- **カスタマイズ可能なラベル**
  コントロール内の右、下、左、上、または中央に表示されるように、ラベルをカスタマイズします。ラベルは、プログレスバーの進行に合わせてマーキーとして動作するように設定することもできます。ラベルは、現在の進捗量、現在の進捗割合、残りの進捗量、残りの進捗割合、最小値、および最大値という6つの進捗インジケータの1つを使用して設定できます。詳細については、「ラベル」を参照してください。
- **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 **ProgressBar for ASP.NET Web Forms** クイックスタートでは、1つの**C1ProgressBar** と2つの **Button** コントロールを備える ASP.NET Web アプリケーションプロジェクトを作成します。〈**スタート**〉ボタンをクリックすると、**RunTask** イベントが発生し、プログレスバーは **UpdateProgress** メソッドを介して 0.5 秒（500 ミリ秒）ごとに更新されるようになります。〈**ストップ**〉ボタンをクリックすると、プログレスバーは更新されなくなります。

手順 1：プロジェクトの作成とコントロールの追加

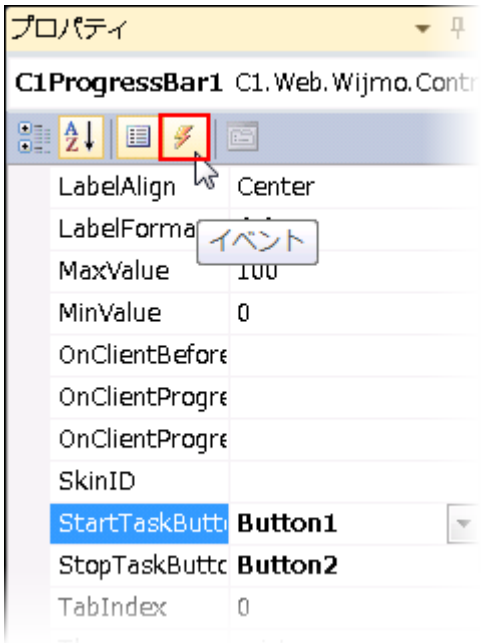
このクイックスタートの最初の手順では、ASP.NET Web アプリケーションプロジェクトを作成して、**C1ProgressBar** と2つの **Button** コントロールをページに追加します。

1. 新しい ASP.NET Web アプリケーションを作成します。
2. 〈**デザイン**〉ボタンをクリックして、デザインビューに入ります。
3. Visual Studio ツールボックスで**C1ProgressBar** アイコンをダブルクリックし、コントロールをページに追加します。
4. 2つの標準 **Button** コントロールを追加し、**Text** プロパティをそれぞれ〈**スタート**〉と〈**ストップ**〉に設定します。ページは次の例のようになります。

手順 2：各コントロールの設定

この手順では、プログレスバーの更新プロセスを開始および停止するボタンとしてボタンコントロールを指定する、**StartTaskButton** と **StopTaskButton** を指定します。

1. Visual Studio **プロパティ**ウィンドウで **C1ProgressBar** コントロールを選択し、**StartTaskButton** の横にあるドロップダウン矢印をクリックして **Button1** を選択します。
2. **StopTaskButton** プロパティを **Button2** に設定します。
3. プロパティウィンドウで、〈**イベント**〉ボタンをクリックし、**C1ProgressBar1** コントロールのイベントをすべてリストします。



RunTask イベントの横に、**ProgressBar1_RunTask** と入力します。これにより、コードビューにイベントが作成され、ここで以下のコードを入力できます。これにより、イベントは次のようになります。

Visual Basic コードの書き方

Visual Basic

```
Protected Sub ProgressBar1_RunTask(sender As Object, e As
C1.Web.Wijmo.Controls.C1ProgressBar.C1ProgressBarTaskEventArgs)
        For i As Integer = 0 To 99
                        System.Threading.Thread.Sleep(500)
                        e.UpdateProgress(i)
        Next
End Sub
```

C# コードの書き方

C#

```
protected void ProgressBar1_RunTask(object sender,
C1.Web.Wijmo.Controls.C1ProgressBar.C1ProgressBarTaskEventArgs e)
    {
        for (int i = 0; i < 100; i++)
        {
            System.Threading.Thread.Sleep(500);
            e.UpdateProgress(i);
        }
    }
```

〈**スタート**〉ボタンをクリックすると、**RunTask** イベントが発生し、**UpdateProgress** メソッドを呼び出して、500 ミリ秒ごとに1ずつプログレスバーを更新します。

手順 3:プロジェクトの実行

[**F5**]を押して、プロジェクトを実行し、〈**スタート**〉ボタンをクリックします。**C1ProgressBar** は 0.5 秒ごとに1%の割合で更新を開始します。更新を停止するには、〈**ストップ**〉ボタンをクリックします。

デザイン時のサポート

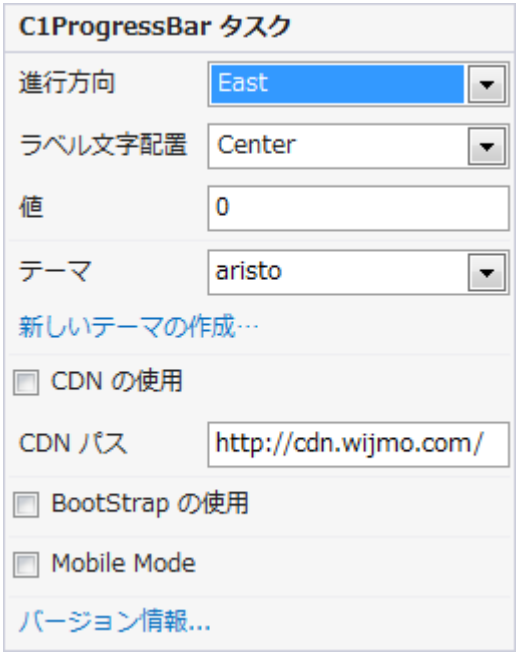
C1ProgressBar は、スマートタグ、および充実したデザイン時のサポートを提供するデザイナを備えており、オブジェクトモデルの操作が簡単になっています。

次のトピックでは、**C1ProgressBar** のデザイン時環境を使用して **C1ProgressBar** コントロールを設定する方法を説明します。

スマートタグとタスクメニュー

Visual Studio では、**C1ProgressBar** コントロールにスマートタグが含まれます。スマートタグは、**C1ProgressBar** で最もよく使用されるプロパティを提供するショートカットタスクメニューです。

[**C1ProgressBar タスク**]メニューにアクセスするには、**C1ProgressBar** コントロールの右上端にあるスマートタグをクリックします。これによって、[**C1ProgressBar タスク**]メニューが開きます。



[**C1ProgressBar タスク**]メニューは次のように動作します。

- **進行方向を取得**
  進捗インジケータによるプログレスバーの進行方向。
- **ラベル文字配置**
  プログレスバーのプログレスラベルの配置。
- **値**
  進捗インジケータの値。この値は **MinValue** プロパティと **MaxValue** プロパティの間のどこかに設定する必要があります。
- **テーマ**
  コントロールの Wijmo テーマ。6種類の組み込みテーマの1つを選択するか、このボックスにカスタムテーマの場所を入力できます。
- **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。

コンテキストメニュー

C1ProgressBar には、Visual Studio がすべての .NET コントロールや ASP.NET コントロールに提供しているコンテキストメニューで利用できる追加的なコマンドがあります。

C1ProgressBar コントロール上の任意の場所を右クリックし、コンテキストメニューを表示します。

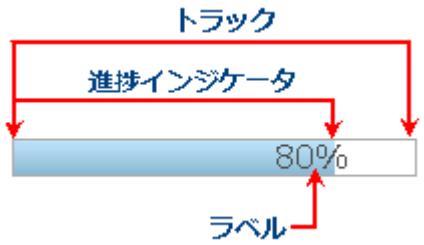
C1ProgressBar のコンテキストメニューは、次のように動作します。

- **スマート タグの表示**
  [**スマート タグの表示**]をクリックすると、[**C1ProgressBar タスク**]メニューが開きます。

C1ProgressBar の要素

C1ProgressBar コントロールは、ユーザーに処理の進捗状況を視覚的に示すグラフィカルユーザーインタフェース要素です。**C1ProgressBar** コントロールは、インタラクティブではないという点で、**ComponentOne for ASP.NET Web Forms** の他の大部分のコントロールとは異なります。このコントロールは、処理が実行中であることを示すだけです。

C1ProgressBar は、デフォルトで、トラック、進捗インジケータ、およびラベルという3つの異なる要素で構成されます。次の図に、これらの要素を示します。



- **トラック**:トラックはコントロールの全長にわたって実行され、ラベルコントロールを含んでいます。処理が開始されると、トラックには進捗インジケータが表示されます。
- **進捗インジケータ**:進捗インジケータは、進行中のタスクの完了状況を視覚的に表現したものです。
- **ラベル**:ラベルコントロールは、進捗状況を英数字で表現するために使用されます。ラベルは6種類の値を表示できます。また、オーバーライドして静的なテキストや静的な数値を表示することもできます。

次のトピックでは、**C1ProgressBar** の各要素について説明します。

トラック

C1ProgressBar コントロールのトラックは、コントロールの全長にわたります。デフォルトでは、トラックは単色(通常は灰色)で表示され、その状態では処理はまだ開始されていません。処理が開始されると、トラックは2つの対照的な色で表示されます。一方は完了した処理量を示し、もう一方は残っている処理量を示します。

C1ProgressBar のトラックのサイズは、**Width** プロパティと **Height** プロパティを使用して設定します。トラックは向きを垂直または水平に設定できます。

進捗インジケータ

進捗インジケータは、最小値と最大値の間で現在の進捗状況を視覚的に表現します。進捗インジケータは、**FillDirection** プロパティを使用して、右、左、上、または側面から進行するように設定できます。**FillDirection** プロパティは、次のように設定できます。

設定	説明
North	進捗インジケータは、コントロールの下から上へ進行します。これにより、コントロールの向きが垂直位置に切り替わります。
South	進捗インジケータは、コントロールの上から下へ進行します。これにより、コントロールの向きが垂直位置に切り替わります。
East	進捗インジケータは、コントロールの左から右へ進行します。
West	進捗インジケータは、コントロールの右から左へ進行します。

進捗インジケータは、**Value** プロパティの値と **MinValue** プロパティおよび **MaxValue** プロパティの値の比較に基づいたパーセンテージまでトラックを満たします。たとえば、**MinValue** プロパティが 0 に設定され、**MaxValue** が 100 に設定され、**Value** プロパティが 25 に設定されている場合、進捗インジケータはトラックの 25% を満たします。**MinValue** プロパティ

が 100 に設定され、**MaxValue** が 300 に設定され、**Value** プロパティが 200 に設定されている場合、進捗インジケータはトラックの 50% を満たします。

進捗インジケータは、処理が進行中であることを固定的に示したり、最新の進捗状況を表示したりできます。

ラベル

プログレスバーのラベルは、進捗状況を英数字で表現するために使用されます。ラベルには、以下の6つの書式のいずれかを表示できます。

トピック	説明
{0} または {ProgressValue}	現在の進捗量を表示します。
{1} または {PercentProgress}	現在の進捗の割合を表示します。
{2} または {RemainingProgress}	タスク完了までに必要な進捗量を表示します。
{3} または {PercentageRemaining}	残りの進捗の割合を表示します。
{4} または {Min}	MinValue プロパティの値を表示します。
{5} または {Max}	MaxValue プロパティの値を表示します。

ラベルをこれらの書式の1つに設定するには、**LabelFormatString** プロパティを設定します。中カッコの外側には任意のテキストを追加できます。たとえば、「{ProgressValue} % 完了」と入力すれば、タスク完了までに残されている処理の割合が示されます。また、ツールチップ全体を独自の文字列で上書きすることもできます。

レベルの配置

ラベル要素は、**LabelAlign** プロパティの設定によって、複数の異なる配置が可能です。以下の設定が使用可能です。

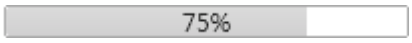
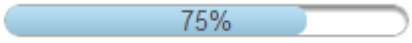
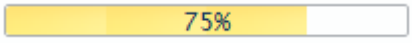
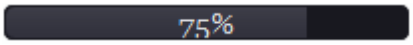
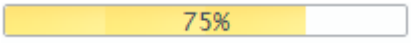
設定	説明
East	ラベルは、コントロールの右側に配置されます。
West	ラベルは、コントロールの左側に配置されます。
Center	ラベルは、コントロールの中央に配置されます。
North	ラベルは、コントロールの上部に配置されます。
South	ラベルは、コントロールの下部に配置されます。
Running	ラベルは、マーキーとして動作します。

C1ProgressBar の動作と外観

次のトピックでは、**C1ProgressBar** の外観と動作に関連する機能について説明します。これらの機能の中には、ロード時のコントロールの動作に影響を与えるものや、ユーザーによるコントロールのインタラクティブ操作に影響を与えるものがあります。

テーマ

C1ProgressBar コントロールには、6種類の組み込みテーマが含まれています。これらのテーマの1つを選択すると、ページ上の他のすべての **ComponentOne for ASP.NET Web Forms** コントロールはそれに応じてスキンされます。テーマは **C1ProgressBar** コントロールに次のように表示されます。

arctic	
aristo	
cobalt	
midnight	
rocket	
sterling	

C1ProgressBar コントロールのテーマを設定するには、単に **Theme** プロパティをいずれかの組み込みテーマに設定します。

ツールチップ

ToolTip プロパティを使用して、使いやすいインタフェースを作成できます。ツールチップは、UI 要素に関してユーザーに情報提供できるグラフィカルユーザーインタフェース要素です。ユーザーが要素上にマウスポインタを置くと、追加的な情報が記載されたボックスが表示されます。

C1ProgressBar コントロールのツールチップ は、実行中の処理についてユーザーに情報提供するために使用するのが普通です。それによって、ユーザーはその処理がどれくらい続くのかを知ることができます。

C1ProgressBar コントロールのツールチップには、以下の6つの書式のいずれかを表示できます。

トピック	説明
{0} または {ProgressValue}	現在の進捗量を表示します。
{1} または {PercentProgress}	現在の進捗の割合を表示します。
{2} または {RemainingProgress}	タスク完了までに必要な進捗量を表示します。
{3} または {PercentageRemaining}	残りの進捗の割合を表示します。
{4} または {Min}	MinValue プロパティの値を表示します。
{5} または {Max}	MaxValue プロパティの値を表示します。

イージング効果の説明

C1ProgressBar には、移動時のプログレスバーの反応を変更する、30 種類以上のアニメーション効果が組み込まれています。デフォルトのイージング効果は **Swing** ですが、**Easing** プロパティを使用して別の効果に設定できます。**Animation.Disabled** プロパティを **True** に設定することで、アニメーションをすべて OFF にすることもできます。

下の表で、各アニメーション効果について説明します。

名前	説明
Linear	直線的なイージング。加速も減速もなく、滑らかに移動します。
Swing（デフォルト）	これはデフォルトのアニメーション効果です。
EaseInQuad	2次型のイージングイン。開始は遅く、それから加速します。
EaseOutQuad	2次型のイージングアウト。開始は速く、それから減速します。
EaseInOutQuad	2次型のイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInCubic	3次型のイージングイン。開始は遅く、それから加速します。
EaseOutCubic	3次型のイージングアウト。開始は速く、それから減速します。
EaseInOutCubic	3次型のイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInQuart	4次型のイージングイン。開始は遅く、それから加速します。
EaseOutQuart	4次型のイージングアウト。開始は速く、それから減速します。
EaseInOutQuart	4次型のイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInQuint	5次型のイージングイン。開始は遅く、それから加速します。
EaseOutQuint	5次型のイージングアウト。開始は速く、それから減速します。
EaseInOutQuint	5次型のイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInSine	正弦型のイージングイン。開始は遅く、それから加速します。
EaseOutSine	正弦型のイージングアウト。開始は速く、それから減速します。
EaseInOutSine	正弦型のイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInExpo	級数的なイージングイン。開始は遅く、それから加速します。
EaseOutExpo	級数的なイージングアウト。開始は速く、それから減速します。
EaseInOutExpo	級数的なイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInCirc	円形のイージングイン。開始は遅く、それから加速します。
EaseOutCirc	円形のイージングアウト。開始は速く、それから減速します。
EaseInOutCirc	円形のイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInElastic	しなやかなイージングイン。開始は遅く、それから加速します。
EaseOutElastic	しなやかなイージングアウト。開始は速く、それから減速します。
EaseInOutElastic	しなやかなイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInBack	バックのイージングイン。開始は遅く、それから加速します。
EaseOutBack	バックのイージングアウト。開始は速く、それから減速します。
EaseInOutBack	バックのイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。
EaseInBounce	バウンドしながらのイージングイン。開始は遅く、それから加速します。
EaseOutBounce	バウンドしながらのイージングアウト。開始は速く、それから減速します。
EaseInOutBounce	バウンドしながらのイージングインとイージングアウト。開始は遅く、途中で加速し、それから減速します。

アニメーションの持続時間

C1ProgressBar のアニメーション効果の長さは、**Duration** プロパティを使用して設定できます。アニメーション効果の持続時間の指定に使用される時間の単位はミリ秒であり、**Duration** プロパティのデフォルト設定値は **500** ミリ秒（0.5 秒）です。アニメーション効果を長くするにはこの値を増加させ、短くするにはこの値を減少させます。

タスク別ヘルプ

タスク別ヘルプのセクションは、Visual Studio ASP.NET 環境でのプログラミングに精通し、**C1ProgressBar** 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、**C1ProgressBar** 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。各トピックで説明されている手順に従うことによって、さまざまな **C1ProgressBar** 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。

ラベルのカスタマイズ

次のトピックでは、**C1ProgressBar** コントロールのラベル要素をカスタマイズする方法を説明します。

ラベルの配置

デフォルトでは、**C1ProgressBar** コントロールのラベルはコントロールの中央に配置されます。このトピックでは、デザインビュー、ソースビュー、およびコードでラベルの配置を変更する方法を学びます。

デザインビューでのラベルの配置

以下の手順を実行します。

1. 〈**デザイン**〉ボタンをクリックして、デザインビューに入ります。
2. **C1ProgressBar** コントロールを右クリックしてそのコンテキストメニューを開き、[**プロパティ**]を選択します。
   プロパティウィンドウが開いて、**C1ProgressBar** コントロールのプロパティがフォーカス状態になります。
3. **LabelAlign** プロパティを指定し、そのドロップダウン矢印をクリックして、リストから **East** を選択します。
4. [F5]を押してプロジェクトを実行し、ラベルがコントロールの右側に表示されることを確認します。

ソースビューでのラベルの配置

以下の手順を実行します。

1. 〈**ソース**〉ボタンをクリックしてソースビューに入ります。
2. `LabelAlign ="East"` を `<cc1:C1ProgressBar>` タグに追加して、マークアップを次のように記述します。

ソースビュー

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server" LabelAlign="East" />
```

3. [**F5**]を押してプロジェクトを実行し、ラベルがコントロールの右側に表示されることを確認します。

コードでのラベルの配置

以下の手順を実行します。

1. Visual Studio ツールバーで、[**表示**]→[**コード**]をクリックしてコードビューに入ります。
2. 以下の名前空間をプロジェクトにインポートします。

Visual Basic コードの書き方

Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1ProgressBar
```

C# コードの書き方

C#

```
using C1.Web.Wijmo.Controls.C1ProgressBar;
```

3. 以下のコードを Page_Load イベントに追加してラベルを配置します。

Visual Basic コードの書き方

```
Visual Basic
C1ProgressBar1.LabelAlign = LabelAlign.East
```

C# コードの書き方

```
C#
C1ProgressBar1.LabelAlign = LabelAlign.East;
```

4. [**F5**]を押してプロジェクトを実行し、ラベルがコントロールの右側に表示されることを確認します。

このトピックの作業結果

このトピックでは、**C1ProgressBar** コントロールのラベルを右側に配置しました。このトピックの結果は、次のようになります。

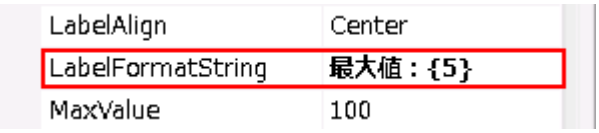
ラベルの書式設定

デフォルトでは、**C1ProgressBar** コントロールのラベルには、現在の進捗の割合が表示されます。このトピックでは、修正された文字列とコントロールの最大値設定が表示されるようにラベルをカスタマイズします。このトピックは、**C1ProgressBar** コントロールを含む ASP.NET プロジェクトを作成済みであることが前提となります。

デザインビューでのラベルの書式設定

以下の手順を実行します。

1. 〈**デザイン**〉ボタンをクリックして、デザインビューに入ります。
2. **C1ProgressBar** コントロールを右クリックしてそのコンテキストメニューを開き、リストから[**プロパティ**]を選択します。プロパティウィンドウが開いて、**C1ProgressBar** コントロールのプロパティがフォーカス状態になります。
3. **LabelFormatString** プロパティを「最大値:{5}」に設定します。



4. [**F5**]を押してプロジェクトを実行し、マウスポインタを**C1ProgressBar** コントロール上に置きます。ラベルにその文字列として「最大値:100」が表示されることを確認してください。

ソースビューでのラベルの書式設定

以下の手順を実行します。

1. 〈**ソース**〉ボタンをクリックしてソースビューに入ります。
2. `LabelFormatString ="最大値:{5}"` を `<cc1:C1ProgressBar>` タグに追加して、マークアップを次のように記述します。

```
ソースビュー
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server" Height="22px"
LabelFormatString="最大値:{5}" Width="288px>
```

3. [F5]を押してプロジェクトを実行し、マウスポインタを C1ProgressBar コントロール上に置きます。ラベルにその文字列として「最大値:100」が表示されることを確認してください。

コードでのラベルの書式設定

以下の手順を実行します。

1. Visual Studio ツールバーで、[表示]→[コード]をクリックしてコードビューに入ります。
2. 以下の名前空間をプロジェクトにインポートします。

Visual Basic コードの書き方

```
Visual Basic
Imports C1.Web.Wijmo.Controls.C1ProgressBar
```

C# コードの書き方

```
C#
using C1.Web.Wijmo.Controls.C1ProgressBar;
```

3. 以下のコードを **Page_Load** イベントに追加してラベルをカスタマイズします。

Visual Basic コードの書き方

```
Visual Basic
C1ProgressBar1.LabelFormatString = "最大値:{5}"
```

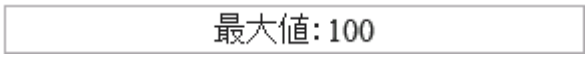
C# コードの書き方

```
C#
C1ProgressBar1.LabelFormatString = "最大値:{5}";
```

4. [**F5**]を押してプロジェクトを実行し、マウスポインタを **C1ProgressBar** コントロール上に置きます。ラベルにその文字列として「最大値:100」が表示されることを確認してください。

このトピックの作業結果

このトピックでは、**C1ProgressBar** コントロールのカスタムラベルを作成しました。次の図は、**LabelFormatString** プロパティを「最大値:{5}」に設定した場合のプログレスバーを示しています。



テーマの利用

このセクションのトピックは、組み込みテーマとカスタムテーマを利用する方法を示します。

組み込みテーマの使用

C1ProgressBar コントロールには、ほんの数クリックで適用できる6種類の組み込みテーマが用意されています。このトピックでは、デザインビュー、ソースビュー、およびコードでテーマを変更する方法を説明します。テーマについての詳細は、「テーマ」を参照してください。

デザインビューでのテーマの変更

以下の手順を実行します。

1. **C1ProgressBar** のスマートタグ()をクリックして、[**C1ProgressBar タスク**]メニューを開きます。
2. [**テーマ**]ドロップダウン矢印をクリックして、リストからテーマを選択します。この例では、**rocket** を選択します。**rocket** テーマが **C1ProgressBar** コントロールに適用されます。

ソースビュー

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server" Theme="rocket"/>
```

ソースビューでのテーマの変更

ソースビューで C1ProgressBar のテーマを変更するには、`Theme="rocket"` を `<cc1:C1ProgressBar>` タグに追加します。その結果、次のようになります。

ソースビュー

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server" Theme="rocket"/>
```

コードでのテーマの変更

以下の手順を実行します。

1. 以下の名前空間をプロジェクトにインポートします。

   Visual Basic コードの書き方

   Visual Basic

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

   C# コードの書き方

   C#

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

2. **Theme** プロパティを設定する次のコードを、**Page_Load** イベントに追加します。

   Visual Basic コードの書き方

   Visual Basic

   ```
   C1ProgressBar1.Theme = "rocket"
   ```

   C# コードの書き方

   C#

   ```
   C1ProgressBar1.Theme = "rocket";
   ```

3. プログラムを実行します。

このトピックの作業結果

次の図は、**rocket** テーマが設定された**C1ProgressBar** コントロールを示しています。

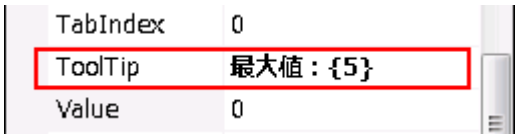
ツールチップの書式設定

実行時にユーザーがマウスポインタを**C1ProgressBar** コントロール上に置いた場合、このコントロールはツールチップを表示します。デフォルトでは、このツールチップには現在の進捗の割合が表示されます。このトピックでは、修正された文字列とコントロールの最大値設定が表示されるようにツールチップをカスタマイズします。このトピックは、**C1ProgressBar** コントロールを含む ASP.NET Web サイトを作成済みであることが前提となります。

デザインビューでのツールチップの書式設定

以下の手順を実行します。

1. 〈**デザイン**〉ボタンをクリックして、デザインビューに入ります。
2. **C1ProgressBar** コントロールを右クリックしてそのコンテキストメニューを開き、リストから[**プロパティ**]を選択します。**プロパティ**ウィンドウが開いて、**C1ProgressBar** コントロールのプロパティがフォーカス状態になります。
3. ToolTip プロパティを「最大値:{5}」に設定します。



4. [F5]を押してプロジェクトを実行し、マウスポインタを C1ProgressBar コントロール上に置きます。ツールチップにその文字列として「最大値:100」が表示されることを確認してください。

ソースビューでのツールチップの書式設定

以下の手順を実行します。

1. 〈**ソース**〉ボタンをクリックしてソースビューに入ります。
2. `ToolTip ="最大値:{5}"` を `<cc1:C1ProgressBar>` タグに追加して、マークアップを次のように記述します。

ソースビュー

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server"
Width="288px" Height="22px" Theme="Arctic" ToolTip="最大値:{5}" />
```

3. [**F5**]を押してプロジェクトを実行し、マウスポインタを **C1ProgressBar** コントロール上に置きます。ツールチップにその文字列として「最大値:100」が表示されることを確認してください。

コードでのツールチップの書式設定

以下の手順を実行します。

1. Visual Studio ツールバーで、[**表示**]→[**コード**]をクリックしてコードビューに入ります。
2. 以下の名前空間をプロジェクトにインポートします。

Visual Basic コードの書き方

Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1ProgressBar
```

C# コードの書き方

C#

```
using C1.Web.Wijmo.Controls.C1ProgressBar;
```

3. 以下のコードを **Page_Load** イベントに追加してツールチップをカスタマイズします。

Visual Basic コードの書き方

Visual Basic

```
C1ProgressBar1.ToolTip ="最大値:{5}"
```

C# コードの書き方

C#

```
C1ProgressBar1.ToolTip ="最大値:{5}";
```

4. [**F5**]を押してプロジェクトを実行し、マウスポインタを **C1ProgressBar** コントロール上に置きます。ツールチップにその文字列として「最大値:100」が表示されることを確認してください。

このトピックの作業結果

このトピックでは、**C1ProgressBar** コントロールのカスタムツールチップを作成しました。次の図は、**ToolTip** プロパティを「最大値:{5}」に設定した場合のプログレスバーを示しています。

進捗インジケータの進行方向の変更

デフォルトでは、進捗インジケータはコントロールの左側からトラックを進行します。進行方向は、**FillDirection** という1つのプロパティを設定することで、右、下、または上から進行するように変更できます。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 **FillDirection** プロパティを設定します。

デザインビューでの進行方向の変更

以下の手順を実行します。

1. 〈**デザイン**〉ボタンをクリックして、デザインビューに入ります。
2. **C1ProgressBar** コントロールを右クリックしてそのコンテキストメニューを開き、[**プロパティ**]を選択します。**プロパティ**ウィンドウが開いて、**C1ProgressBar** コントロールのプロパティがフォーカス状態になります。
3. **プロパティ**ウィンドウで、**FillDirection** プロパティを設定します。この例では、**West** に設定します。

ソースビューでの進行方向の変更

以下の手順を実行します。

1. 〈**ソース**〉ボタンをクリックしてソースビューに入ります。
2. `FillDirection="West"` を `<cc1:C1ProgressBar>` のタグに追加します。マークアップは次のような表示になります。

Title Text

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server"
FillDirection = "West" Value="100" AnimationDuration="6000"/>
```

コードでの進行方向の変更

以下の手順を実行します。

1. Visual Studio ツールバーで、[**表示**]→[**コード**]をクリックしてコードビューに入ります。
2. 以下の名前空間をプロジェクトにインポートします。

Visual Basic コードの書き方

```
Visual Basic
Imports C1.Web.Wijmo.Controls.C1ProgressBar
```

C# コードの書き方

```
C#
using C1.Web.Wijmo.Controls.C1ProgressBar;
```

3. 次のコードを **Page_Load** イベントに追加します。

Visual Basic コードの書き方

```
Visual Basic
' FillDirection プロパティを FromRightOrBottom に設定
C1ProgressBar1.FillDirection = FillDirection.West
```

C# コードの書き方

```
C#
// FillDirection プロパティを FromRightOrBottom に設定
C1ProgressBar1.FillDirection = FillDirection.West;
```

アニメーションの設定

C1ProgressBar コントロールは、31 種類の組み込みアニメーションを備えています。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、アニメーション、アニメーションの持続時間、およびアニメーションの遅延時間を設定します。

デザインビューでのアニメーションの設定

以下の手順を実行します。

1. 〈**デザイン**〉ボタンをクリックして、デザインビューに入ります。
2. **C1ProgressBar** コントロールを右クリックしてそのコンテキストメニューを開き、[**プロパティ**]を選択します。**プロパティ**ウィンドウが開いて、**C1ProgressBar** コントロールのプロパティがフォーカス状態になります。
3. **プロパティ**ウィンドウで、以下のプロパティを設定します。
   - **AnimationOptions.Easing** プロパティを **EaseOutBounce** に設定します。これによって、アニメーション効果が設定されます。
   - **AnimationOptions.Duration** プロパティを「6000」に設定します。これによって、アニメーションの長さが設定されます。
   - **AnimationDelay** プロパティを「500」に設定します。これにより、アニメーションが始まるまでの経過時間が設定されます。
   - **Value** プロパティを「100」に設定します。これによって、実行時に進捗インジケータがトラック上を進行するようになります。
4. [**F5**]を押してプロジェクトを実行します。アニメーションの開始までに約 0.5 秒かかり、完了には6秒かかることを確認してください。アニメーション **EaseOutBounce** が設定されると、進捗インジケータはトラックの終端に当たってバウンドしてから安定した状態に落ち着きます。

ソースビューでのアニメーションの設定

以下の手順を実行します。

1. 〈**ソース**〉ボタンをクリックしてソースビューに入ります。
2. `AnimationDelay="500"` を `<cc1:C1ProgressBar>` タグに追加して、マークアップを次のように記述します。

ソースビュー

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server" AnimationDelay="500">
</cc1:C1ProgressBar>
```

3. <cc1:C1ProgressBar> タグの間に次のマークアップを置きます。

ソースビュー

```
<AnimationOptions Duration="6000" Easing="EaseOutBounce" />
```

4. [**F5**]を押してプロジェクトを実行します。アニメーションの開始までに約 0.5 秒かかり、完了には6秒かかることを確認してください。アニメーション **EaseOutBounce** が設定されると、進捗インジケータはトラックの終端に当たってバウンドしてから安定した状態に落ち着きます。

コードでのアニメーションの設定

以下の手順を実行します。

1. Visual Studio ツールバーで、[**表示**]→[**コード**]をクリックしてコードビューに入ります。
2. 次のコードを **Page_Load** イベントに追加します。

Visual Basic コードの書き方

Visual Basic

```
' アニメーションのプロパティを設定
C1ProgressBar1.AnimationDelay = 500;
C1ProgressBar1.AnimationOptions.Duration = 6000;
C1ProgressBar1.AnimationOptions.Easing =
C1.Web.Wijmo.Controls.Easing.EaseOutBounce;
' 進捗インジケータが実行時に自動的に進行するように Value プロパティを設定
C1ProgressBar1.Value = 100
```

C# コードの書き方

C#

```
// アニメーションのプロパティを設定
C1ProgressBar1.AnimationDelay = 500;
C1ProgressBar1.AnimationOptions.Duration = 6000;
C1ProgressBar1.AnimationOptions.Easing =
C1.Web.Wijmo.Controls.Easing.EaseOutBounce;
// 進捗インジケータが実行時に自動的に進行するように Value プロパティを設定
C1ProgressBar1.Value = 100;
```

3. [**F5**]を押してプロジェクトを実行します。アニメーションの開始までに約 0.5 秒かかり、完了には6秒かかることを確認してください。アニメーション **EaseOutBounce** が設定されると、進捗インジケータはトラックの終端に当たってバウンドしてから安定した状態に落ち着きます。

値の設定

C1ProgressBar コントロールには、最小値、最大値、および値があります。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、**MinValue**、**MaxValue**、**Value** の各プロパティの値をデフォルト値（それぞれ 0、100、0）から変更します。

デザインビューでの値の設定

以下の手順を実行します。

1. 〈**デザイン**〉ボタンをクリックして、デザインビューに入ります。
2. **C1ProgressBar** コントロールを右クリックしてそのコンテキストメニューを開き、[**プロパティ**]を選択します。
   プロパティウィンドウが開いて、**C1ProgressBar** コントロールのプロパティがフォーカス状態になります。
3. **プロパティ**ウィンドウで、以下のプロパティを設定します。
   - **MaxValue** プロパティを「300」に設定します。
   - **MinValue** プロパティを「100」に設定します。
   - **Value** プロパティを「150」に設定します。
4. [**F5**]を押してプロジェクトを実行し、進捗インジケータがプログレスバーの4分の1だけ進行していることを確認します。
   これは、コントロールの値（150）が、最小値（100）と最大値（300）の間隔の4分の1であるためです。

ソースビューでの値の設定

以下の手順を実行します。

1. 〈**ソース**〉ボタンをクリックしてソースビューに入ります。
2. `MinimumValue="100"`、MaximumValue="300"、および Value="150" を `<cc1:C1ProgressBar>` タグに追加して、マークアップを次のように記述します。

ソースビュー

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server"
ToolTip="最大値:{5}" LabelFormatString="{0}" MaximumValue="300"
MinimumValue="100" Value="150" />
```

3. [**F5**]を押してプロジェクトを実行し、進捗インジケータがプログレスバーの4分の1だけ進行していることを確認します。
   これは、コントロールの値（150）が、最小値（100）と最大値（300）の間隔の4分の1であるためです。

コードでの値の設定

以下の手順を実行します。

1. アニメーションを設定するには、以下の手順を実行します。
2. Visual Studio ツールバーで、[**表示**]→[**コード**]をクリックしてコードビューに入ります。
3. **MaxValue** プロパティを設定するには、次のコードを **Page_Load** イベントに追加します。

Visual Basic コードの書き方

Visual Basic

```
C1ProgressBar1.MaxValue = 300
```

C# コードの書き方

C#

```
C1ProgressBar1.MaxValue = 300;
```

4. **MinValue** プロパティを設定するには、次のコードを **Page_Load** イベントに追加します。

Visual Basic コードの書き方

Visual Basic

```
C1ProgressBar1.MinValue = 100
```

C# コードの書き方

C#

```
C1ProgressBar1.MinValue = 100;
```

5. **Value** プロパティを設定するには、次のコードを **Page_Load** イベントに追加します。

   Visual Basic コードの書き方

   ```
   Visual Basic
   C1ProgressBar1.Value = 150
   ```

   C# コードの書き方

   ```
   C#
   C1ProgressBar1.Value = 150;
   ```

6. [**F5**]を押してプロジェクトを実行し、進捗インジケータがプログレスバーの4分の1だけ進行していることを確認します。これは、コントロールの値(150)が、最小値(100)と最大値(300)の間隔の4分の1であるためです。