ScatterChart for ASP.NET Web Forms

2018.04.11 更新

グレープシティ株式会社

ScatterChart for ASP.NET Web Forms

目次

製品の概要

ScatterChart for ASP.NET Web Forms によって、ユーザーは2つの値の軸を表示でき、x 軸に1つの数値データセット、y 軸にもう1つの数値データセットを表示します。**C1ScatterChart** は、これらの値を組み合わせて1つのデータポイントにし、不規則な間隔やクラスタで表示します。

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

ScatterChart for ASP.NET Web Forms は、以下の独特な主要機能を備えています。

- **充実したクライアント側オブジェクトモデル**
  C1ScatterChart の充実したクライアント側オブジェクトモデルを使用することによって、for Web アプリケーションを効率化します。クライアント側で使用できるクライアント側プロパティ、メソッド、およびイベントが提供されます。
- **グラフ要素の高度な視覚効果**
  色、角度、階調度、輝度、スケール、形状を調整して効果を変更することにより、グラフ要素の外観を向上させます。
- **ツールチップのラベル**
  グラフ要素へのホバー時、C1ScatterChart は各ポイントと系列に関する情報を持つツールチップテキストを表示できます。ツールチップはユーザーがグラフ上でマウスを動かすと後を追います。
- **軸の複数の値タイプ**
  X 軸は string/numeric/datetime の型をサポートし、Y 軸は numeric/datetime の型をサポートしています。
  C1ScatterChart は、numeric 値と time 値に関して非常にインテリジェントです。各単位の間隔の値や目盛の配置場所が自動的に決定されます。
- **クロスブラウザ対応**
  IE、Firefox、Safari、Chrome、Opera などの主要なブラウザをサポートしています。

機能一覧

このトピックでは、ScatterChart for ASP.NET Web Formsの主な機能の一覧と、機能を理解するのに役立つページへのリンクを記載しています。

散布図の機能

機能	サンプル	ヘルプ	リファレンス	KB
マーカーの種類（円、三角形、逆三角形、四角形、ひし形、十字）			○	
ホバー時に拡大			○	
リロード時のアニメーション			○	

チャート全般の機能

機能	サンプル	ヘルプ	リファレンス	KB
チャート系列のスタイル			○	
マウスホバー時のスタイル			○	
チャートテキストのスタイル			○	
チャートラベルの表示		○	○	
チャートラベルの書式		○	○	
チャートラベルのスタイル		○	○	
テーマ			○	
影			○	
マージン			○1、○2、○3、○4	
カルチャ情報			○	

データ種別の機能

機能	サンプル	ヘルプ	リファレンス	KB
数値（Double）			○	
日時			○	
文字列（X軸のみ）			○	

データ連結の機能

機能	サンプル	ヘルプ	リファレンス	KB
データソース（ASP.NETデータソースコントロール）		○		
連結するデータメンバ		○	○	
X値のフィールド名			○	

X値のデータ型（数値、日時、文字列）			○	
Y値のフィールド名			○	
Y値のデータ型（数値、日時）			○	

軸の機能

機能	サンプル	ヘルプ	リファレンス	KB
最大値		○	○	
最大値を自動的に計算		○	○	
最小値		○	○	
最小値を自動的に計算		○	○	
原点の値			○	
軸の表示			○	
軸の位置（上、下、左、右）		○	○	
軸のスタイル		○	○	
軸タイトルの表示			○	
軸タイトルの値		○	○	
軸タイトルのスタイル			○	
軸タイトルの配置（左、中央、右）		○	○	
注釈の書式			○	
注釈の表示方法（値、ラベル）			○	
主目盛りの値を自動的に計算		○	○	
主目盛り記号の単位		○	○	
副目盛りの値を自動的に計算		○	○	
副目盛り記号の単位		○	○	

軸グリッド線の機能

機能	サンプル	ヘルプ	リファレンス	KB
スタイル			○	
表示			○	

軸目盛りの機能

機能	サンプル	ヘルプ	リファレンス	KB
長さ		○	○	
種類（内側、外側、両方、なし）		○	○	

軸ラベルの機能

機能	サンプル	ヘルプ	リファレンス	KB
スタイル			○	
テキストの配置（左、中央、右）			○	
幅			○	

値ラベルの機能

機能	サンプル	ヘルプ	リファレンス	KB
テキスト			○	
数値			○	
DateTime値			○	
グリッド線の表示			○	
グリッド線のスタイル			○	

ヘッダ／フッタの機能

機能	サンプル	ヘルプ	リファレンス	KB
表示			○	
位置（上、下、左、右）			○	
テキスト			○	
テキストのスタイル			○	
タイトルのスタイル			○	

凡例の機能

機能	サンプル	ヘルプ	リファレンス	KB
表示			○	
位置（上、下、左、右）			○	
方向（水平、垂直）			○	
アイコンのサイズ			○	
スタイル			○	
テキスト			○	
テキストの幅			○	
テキストの余白			○	
テキストのスタイル			○	

タイトルのスタイル			○	

ヒントの機能

機能	サンプル	ヘルプ	リファレンス	KB
表示			○	
位置（上、下、左、右）			○	
水平方向のオフセット			○	
垂直方向のオフセット			○	
スタイル			○	
コンテンツ			○	
コンテンツのスタイル			○	
タイトル			○	
タイトルのスタイル			○	
吹き出しの表示			○	
吹き出し			○	
吹き出しのスタイル			○	
アニメーション効果			○	
表示／非表示にする時間			○	
イージング			○	
表示のアニメーション効果			○	
表示の遅延時間			○	
表示にする時間			○	
表示のイージング			○	
非表示のアニメーション効果			○	
非表示の遅延時間			○	
非表示にする時間			○	
非表示のイージング			○	

アニメーションの機能

機能	サンプル	ヘルプ	リファレンス	KB
有効			○	
イージング（7種類）		○	○	
持続時間		○	○	

スタイルの機能

機能	サンプル	ヘルプ	リファレンス	KB
Clip-Rect			○	
マウスカーソル			○	
CXポイント値			○	
CYポイント値			○	
塗りつぶし色（単色、線形グラデーション、放射状グラデーション）			○	
塗りつぶしの不透明度			○	
フォント			○	
フォントサイズ			○	
フォントの太さ			○	
高さ			○	
ハイパーリンク			○	
不透明度			○	
パス			○	
Rの長さ			○	
回転			○	
RXポイント値			○	
RYポイント値			○	
スケール			○	
Src			○	
ストロークの色（単色、線形グラデーション、放射状グラデーション）			○	
ストロークのダッシュ配列			○	
ストロークのラインキャップ			○	
ストロークのLineJoin			○	
ストロークのMiterLimit			○	
ストロークの不透明度			○	
ストロークの幅			○	
ターゲット			○	
テキストアンカー			○	
タイトル			○	
変換			○	
幅			○	
X座標			○	
Y座標			○	

デザイン時の機能

機能	サンプル	ヘルプ	リファレンス	KB
スマートタグ			○	
コレクションエディター			○	

クイックスタート

このクイックスタートは、ASP.NET コントロールである **ScatterChart** の基本的な使用方法を説明します。このクイックスタートでは、1つの **C1ScatterChart** コントロールを含む ASP.NET アプリケーションを作成し、データを C1ScatterChart に追加して、C1ScatterChart の要素（ヘッダー、軸、およびグラフラベル）をカスタマイズします。

手順 1：アプリケーションの作成

このトピックでは、C1ScatterChart コントロールを追加します。

1. まず、ASP.NET Web アプリケーションの作成から始めます。この例では Visual Studio 2010 を使用します。
2. コントロールをツールボックスに追加します。
3. 「**デザイン**」タブを選択します。
4. デザインビューで、Visual Studio ツールボックスに移動し、**C1ScatterChart** アイコンをダブルクリックして **C1ScatterChart** をページのメインコンテンツに追加します。

手順 2：コントロールへのデータの追加

この手順では、プログラムによって ScatterChartSeries を作成して、系列のマーカとラベルをカスタマイズします。

1. フォームを右クリックし、[コードの表示]を選択してコードファイルを表示してから、次の指示文を追加して C1.Web.Wijmo.Controls.Chart 名前空間を宣言します。

   C#コードの書き方

   C#
   ```
   Using C1.Web.Wijmo.Controls.Chart;
   ```

2. 作成するコードファイルに次の関数を追加し、データを C1ScatterChart に追加します。

   C#コードの書き方

   C#
   ```
   private void PrepareOptions()
             {
                    var valuesX = new List<double?>() { 161.2, 167.5, 159.5, 157,
   155.8, 170, 159.1, 166, 176.2, 160.2, 172.5, 170.9, 172.9, 153.4, 160, 147.2,
   168.2, 175, 157, 167.6, 159.5, 175, 166.8, 176.5, 170.2, 174, 173, 179.9, 170.5,
   160, 154.4, 162, 176.5, 160, 152, 162.1, 170, 160.2, 161.3, 166.4, 168.9, 163.8,
   167.6, 160, 161.3, 167.6, 165.1, 160, 170, 157.5, 167.6, 160.7, 163.2, 152.4,
   157.5, 168.3, 180.3, 165.5, 165, 164.5, 156, 160, 163, 165.7, 161, 162, 166,
   174, 172.7, 167.6, 151.1, 164.5, 163.5, 152, 169, 164, 161.2, 155, 170, 176.2,
   170, 162.5, 170.3, 164.1, 169.5, 163.2, 154.5, 159.8, 173.2, 170, 161.4, 169,
   166.2, 159.4, 162.5, 159, 162.8, 159, 179.8, 162.9, 161, 151.1, 168.2, 168.9,
   173.2, 171.8, 178, 164.3, 163, 168.5, 166.8, 172.7, 163.5, 169.4, 167.8, 159.5,
   167.6, 161.2, 160, 163.2, 162.2, 161.3, 149.5, 157.5, 163.2, 172.7, 155, 156.5,
   164, 160.9, 162.8, 167, 160, 160, 168.9, 158.2, 156, 160, 167.1, 158, 167.6,
   156, 162.1, 173.4, 159.8, 170.5, 159.2, 157.5, 161.3, 162.6, 160, 168.9, 165.1,
   162.6, 165.1, 166.4, 160, 152.4, 170.2, 162.6, 170.2, 158.8, 172.7, 167.6,
   162.6, 167.6, 156.2, 175.2, 172.1, 162.6, 160, 165.1, 182.9, 166.4, 165.1,
   177.8, 165.1, 175.3, 154.9, 158.8, 172.7, 168.9, 161.3, 167.6, 165.1, 175.3,
   ```

```
157.5, 163.8, 167.6, 165.1, 165.1, 168.9, 162.6, 164.5, 176.5, 168.9, 175.3,
159.4, 160, 170.2, 162.6, 167.6, 162.6, 160.7, 160, 157.5, 162.6, 152.4, 170.2,
165.1, 172.7, 165.1, 170.2, 170.2, 170.2, 161.3, 167.6, 167.6, 165.1, 162.6,
152.4, 168.9, 170.2, 175.2, 175.2, 160, 165.1, 174, 170.2, 160, 167.6, 167.6,
167.6, 154.9, 162.6, 175.3, 171.4, 157.5, 165.1, 160, 174, 162.6, 174, 162.6,
161.3, 156.2, 149.9, 169.5, 160, 175.3, 169.5, 160, 172.7, 162.6, 157.5, 176.5,
164.4, 160.7, 174, 163.8 };
                var valuesY = new List<double?>() { 51.6, 59, 49.2, 63, 53.6,
59, 47.6, 69.8, 66.8, 75.2, 55.2, 54.2, 62.5, 42, 50, 49.8, 49.2, 73.2, 47.8,
68.8, 50.6, 82.5, 57.2, 87.8, 72.8, 54.5, 59.8, 67.3, 67.8, 47, 46.2, 55, 83,
54.4, 45.8, 53.6, 73.2, 52.1, 67.9, 56.6, 62.3, 58.5, 54.5, 50.2, 60.3, 58.3,
56.2, 50.2, 72.9, 59.8, 61, 69.1, 55.9, 46.5, 54.3, 54.8, 60.7, 60, 62, 60.3,
52.7, 74.3, 62, 73.1, 80, 54.7, 53.2, 75.7, 61.1, 55.7, 48.7, 52.3, 50, 59.3,
62.5, 55.7, 54.8, 45.9, 70.6, 67.2, 69.4, 58.2, 64.8, 71.6, 52.8, 59.8, 49, 50,
69.2, 55.9, 63.4, 58.2, 58.6, 45.7, 52.2, 48.6, 57.8, 55.6, 66.8, 59.4, 53.6,
73.2, 53.4, 69, 58.4, 56.2, 70.6, 59.8, 72, 65.2, 56.6, 105.2, 51.8, 63.4, 59,
47.6, 63, 55.2, 45, 54, 50.2, 60.2, 44.8, 58.8, 56.4, 62, 49.2, 67.2, 53.8,
54.4, 58, 59.8, 54.8, 43.2, 60.5, 46.4, 64.4, 48.8, 62.2, 55.5, 57.8, 54.6,
59.2, 52.7, 53.2, 64.5, 51.8, 56, 63.6, 63.2, 59.5, 56.8, 64.1, 50, 72.3, 55,
55.9, 60.4, 69.1, 84.5, 55.9, 55.5, 69.5, 76.4, 61.4, 65.9, 58.6, 66.8, 56.6,
58.6, 55.9, 59.1, 81.8, 70.7, 56.8, 60, 58.2, 72.7, 54.1, 49.1, 75.9, 55, 57.3,
55, 65.5, 65.5, 48.6, 58.6, 63.6, 55.2, 62.7, 56.6, 53.9, 63.2, 73.6, 62, 63.6,
53.2, 53.4, 55, 70.5, 54.5, 54.5, 55.9, 59, 63.6, 54.5, 47.3, 67.7, 80.9, 70.5,
60.9, 63.6, 54.5, 59.1, 70.5, 52.7, 62.7, 86.3, 66.4, 67.3, 63, 73.6, 62.3,
57.7, 55.4, 104.1, 55.5, 77.3, 80.5, 64.5, 72.3, 61.4, 58.2, 81.8, 63.6, 53.4,
54.5, 53.6, 60, 73.6, 61.4, 55.5, 63.6, 60.9, 60, 46.8, 57.3, 64.1, 63.6, 67.3,
75.5, 68.2, 61.4, 76.8, 71.8, 55.5, 48.6, 66.4, 67.3 };

                //女性の系列一覧


                var series = new ScatterChartSeries();
                this.C1ScatterChart1.SeriesList.Add(series);
                series.MarkerType = MarkerType.Circle;
                series.Data.X.AddRange(valuesX.ToArray<double?>());
                series.Data.Y.AddRange(valuesY.ToArray<double?>());
                series.Label = "Female";
                series.LegendEntry = true;

                //男性の系列一覧


                valuesX = new List<double?>() { 174, 175.3, 193.5, 186.5,
187.2, 181.5, 184, 184.5, 175, 184, 180, 177.8, 192, 176, 174, 184, 192.7,
171.5, 173, 176, 176, 180.5, 172.7, 176, 173.5, 178, 180.3, 180.3, 164.5, 173,
183.5, 175.5, 188, 189.2, 172.8, 170, 182, 170, 177.8, 184.2, 186.7, 171.4,
172.7, 175.3, 180.3, 182.9, 188, 177.2, 172.1, 167, 169.5, 174, 172.7, 182.2,
164.1, 163, 171.5, 184.2, 174, 174, 177, 186, 167, 171.8, 182, 167, 177.8,
164.5, 192, 175.5, 171.2, 181.6, 167.4, 181.1, 177, 174.5, 177.5, 170.5, 182.4,
197.1, 180.1, 175.5, 180.6, 184.4, 175.5, 180.6, 177, 177.1, 181.6, 176.5, 175,
174, 165.1, 177, 192, 176.5, 169.4, 182.1, 179.8, 175.3, 184.9, 177.3, 167.4,
178.1, 168.9, 157.2, 180.3, 170.2, 177.8, 172.7, 165.1, 186.7, 165.1, 174,
175.3, 185.4, 177.8, 180.3, 180.3, 177.8, 177.8, 177.8, 177.8, 177.8, 163.8,
```

```
188, 198.1, 175.3, 166.4, 190.5, 166.4, 177.8, 179.7, 172.7, 190.5, 185.4,
168.9, 167.6, 175.3, 170.2, 190.5, 177.8, 190.5, 177.8, 184.2, 176.5, 177.8,
180.3, 171.4, 172.7, 172.7, 177.8, 177.8, 182.9, 170.2, 167.6, 175.3, 165.1,
185.4, 181.6, 172.7, 190.5, 179.1, 175.3, 170.2, 193, 171.4, 177.8, 177.8,
167.6, 167.6, 180.3, 182.9, 176.5, 186.7, 188, 188, 177.8, 174, 177.8, 171.4,
185.4, 185.4, 188, 188, 182.9, 176.5, 175.3, 175.3, 188, 188, 175.3, 170.5,
179.1, 177.8, 175.3, 182.9, 170.8, 188, 180.3, 177.8, 185.4, 168.9, 185.4,
180.3, 174, 167.6, 182.9, 160, 180.3, 167.6, 186.7, 175.3, 175.3, 175.9, 175.3,
179.1, 181.6, 177.8, 182.9, 177.8, 184.2, 179.1, 176.5, 188, 174, 167.6, 170.2,
167.6, 188, 174, 176.5, 180.3, 167.6, 188, 180.3, 167.6, 183, 183, 179.1, 170.2,
177.8, 179.1, 190.5, 177.8, 180.3, 180.3 };
                valuesY = new List<double?>() { 65.6, 71.8, 80.7, 72.6, 78.8,
74.8, 86.4, 78.4, 62, 81.6, 76.6, 83.6, 90, 74.6, 71, 79.6, 93.8, 70, 72.4,
85.9, 78.8, 77.8, 66.2, 86.4, 81.8, 89.6, 82.8, 76.4, 63.2, 60.9, 74.8, 70,
72.4, 84.1, 69.1, 59.5, 67.2, 61.3, 68.6, 80.1, 87.8, 84.7, 73.4, 72.1, 82.6,
88.7, 84.1, 94.1, 74.9, 59.1, 75.6, 86.2, 75.3, 87.1, 55.2, 57, 61.4, 76.8,
86.8, 72.2, 71.6, 84.8, 68.2, 66.1, 72, 64.6, 74.8, 70, 101.6, 63.2, 79.1, 78.9,
67.7, 66, 68.2, 63.9, 72, 56.8, 74.5, 90.9, 93, 80.9, 72.7, 68, 70.9, 72.5,
72.5, 83.4, 75.5, 73, 70.2, 73.4, 70.5, 68.9, 102.3, 68.4, 65.9, 75.7, 84.5,
87.7, 86.4, 73.2, 53.9, 72, 55.5, 58.4, 83.2, 72.7, 64.1, 72.3, 65, 86.4, 65,
88.6, 84.1, 66.8, 75.5, 93.2, 82.7, 58, 79.5, 78.6, 71.8, 116.4, 72.2, 83.6,
85.5, 90.9, 85.9, 89.1, 75, 77.7, 86.4, 90.9, 73.6, 76.4, 69.1, 84.5, 64.5,
69.1, 108.6, 86.4, 80.9, 87.7, 94.5, 80.2, 72, 71.4, 72.7, 84.1, 76.8, 63.6,
80.9, 80.9, 85.5, 68.6, 67.7, 66.4, 102.3, 70.5, 95.9, 84.1, 87.3, 71.8, 65.9,
95.9, 91.4, 81.8, 96.8, 69.1, 82.7, 75.5, 79.5, 73.6, 91.8, 84.1, 85.9, 81.8,
82.5, 80.5, 70, 81.8, 84.1, 90.5, 91.4, 89.1, 85, 69.1, 73.6, 80.5, 82.7, 86.4,
67.7, 92.7, 93.6, 70.9, 75, 93.2, 93.2, 77.7, 61.4, 94.1, 75, 83.6, 85.5, 73.9,
66.8, 87.3, 72.3, 88.6, 75.5, 101.4, 91.1, 67.3, 77.7, 81.8, 75.5, 84.5, 76.6,
85, 102.5, 77.3, 71.8, 87.9, 94.3, 70.9, 64.5, 77.3, 72.3, 87.3, 80, 82.3, 73.6,
74.1, 85.9, 73.2, 76.3, 65.9, 90.9, 89.1, 62.3, 82.7, 79.1, 98.2, 84.1, 83.2,
83.2 };

                series = new ScatterChartSeries();
                this.C1ScatterChart1.SeriesList.Add(series);
                series.MarkerType = MarkerType.Diamond;
                series.Data.X.AddRange(valuesX.ToArray<double?>());
                series.Data.Y.AddRange(valuesY.ToArray<double?>());
                series.Label = "Male";
                series.LegendEntry = true;
          }
```

3. コードページに以下のコードを追加し、ポストバックされないときに関数を呼び出します。

C#コードの書き方

C#

```
protected void Page_Load(object sender, EventArgs e)
{
   if (!IsPostBack)
   {
     PrepareOptions();
   }
}
```

手順 3:コントロールの各要素のカスタマイズ

このトピックでは、**Fill** プロパティを使用して系列の色、**Text** プロパティを使用してヘッダーと X 軸のテキスト、および **Stroke** プロパティ を使用して gridmajor、tickmajor、tickminor の各要素のストロークスタイルをカスタマイズします。

1. **デザイン**ビューで、C1ScatterChart を選択し、そのタスクメニューから**シリーズスタイル**を選択します。**ChartStyle コレクションエディター**が表示されます。
2. 〈**追加**〉をクリックします。新しい **ChartStyle** メンバが追加されます。
3. **Fill** ノードを拡張し、**Color** を **#AFE500**に設定します。
4. **Stroke** プロパティを**#AFE500** に設定します。
5. 〈**追加**〉をクリックします。2番目の**ChartStyle** メンバが追加されます。
6. **Fill** ノードを拡張し、**Color** を**#FF9900**に設定します。
7. **Stroke** プロパティを **#FF9900** に設定します。
8. **ソースビュー**で新しい ChartStyles のソースコードは次のようになるはずです。

ソースビュー

```
<SeriesStyles>
    <cc1:ChartStyle Stroke="#AFE500">
      <Fill Color="#AFE500">
      </Fill>
    </cc1:ChartStyle>
    <cc1:ChartStyle Stroke="#FF9900">
      <Fill Color="#FF9900">
      </Fill>
    </cc1:ChartStyle>
</SeriesStyles>
```

9. **デザインビュー**で、**Header** を拡張し、**Text** プロパティを **性別による72人の身長／体重のグラフ** に設定し、次に **TextStyle** を拡張し、FontSize を **18** に設定します。ヘッダーがグラフ領域の真上に表示されますが、**Compass** プロパティを使用して別の位置に移動できます。
10. **Axis->X** を拡張し、**Text** プロパティを **身長(センチ)** に設定します。
11. **Axis->X->-Labels->AxisLabelStyle** を拡張し、**FontSize** プロパティを **11pt** に設定します。これによって、X 軸ラベルのフォントサイズは 11ptに変わります。同様に、**Rotation** プロパティを-45 、および **Fill** 色を **#7F7F7F** に設定します。
12. **Axis->X->TickMajor** を拡張し、**Position** プロパティを **Outside** に設定します。これによって x 軸の位置が outside に変わります。
13. **Axis->Y** を拡張し、**Text** プロパティを **体重(キロ)** に設定します。
14. **Axis->Y>->Labels->AxisLabelStyle** を拡張し、**FontSize** プロパティを **11pt** に設定します。これによって、y 軸のラベルのフォントサイズは 11pt に変わります。同様に、**Visible** プロパティを False、**TextAlign** を **Center**、および **Fill** 色を **#7F7F7F** に設定します。
15. **Axis->Y>GridMajor->GridStyle** を拡張し、**Stroke** プロパティを**#353539** に設定します。
16. **Axis->Y->TickMajor** を拡張し、**Position** プロパティを **Outside** に設定します。これによって x 軸の位置が outside に変わります。
17. **Axis->Y>TickMajor->TickStyle** を拡張し、**Stroke** プロパティをto **#7F7F7F**に設定します。
18. **Axis->Y->TickMinor** を拡張し、**Position** プロパティを **Outside** に設定します。これによって x 軸の位置が outside に変わります。
19. **Axis->Y->TickMinor->TickStyle** を拡張し、**Stroke** プロパティを to **#7F7F7F** に設定します。

手順 4:グラフのツールチップの追加

このトピックでは、単純な JavaScript 関数をソースページに追加して、この JavaScript 関数の名前を **Hint** オブジェクトに適用します。

1. 以下の JavaScript 関数をソースページに追加します。

ソースビュー

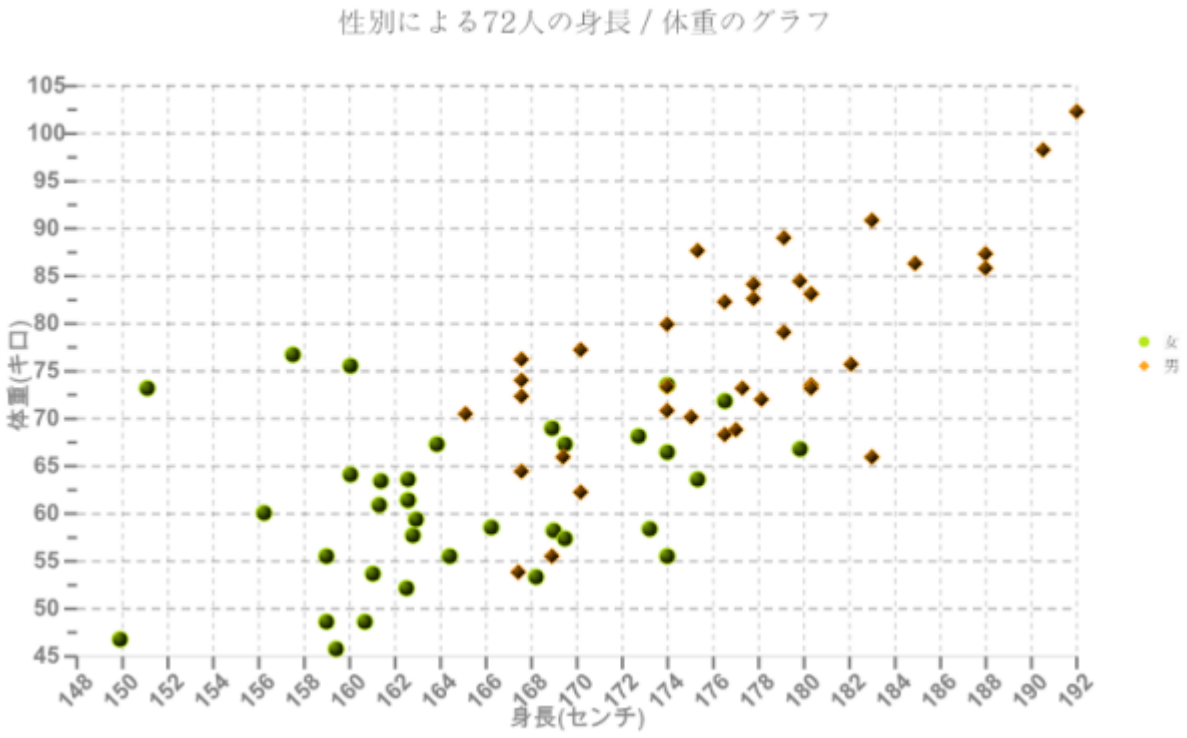
```
<script type="text/javascript">
 function hintContent() {
  return this.x + ' センチ, ' + this.y + ' キロ';
 }
</script>
```

2. デザインビューで **C1ScatterChart** を右クリックし、[**プロパティ**]を選択して **C1ScatterChart** のプロパティウィンドウを開きます。
3. **Hint->Content** を拡張し、**Function** プロパティを **hintContent** に設定します。これによって JavaScript 関数がグラフツールチップに適用されます。

手順 5：アプリケーションの実行

[**F5**]を押して、プロジェクトを実行します。次のような表示になります。

- scatter のデータがアニメーションによってどのような状態で左から右へ表示領域に入ってくるかに注意してください。
- 散布図を使用して、2つのデータセットである男と女が容易に比較されています。女性が男性よりも身長が低くて体重が軽いことが即座にわかります。
- 散布図の各要素がカスタマイズされて表示されています。
- データポイントの上にホバーするとツールチップが表示されます。各データポイントの x 値と y 値が説明されます。



デザイン時のサポート

C1ScatterChart 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、および充実したデザイン時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素化されています。

以下のセクションでは、C1ScatterChart のデザイン時環境を使用して **C1ScatterChart** コントロールを設定する方法について説明します。

C1ScatterChart スマートタグ

Visual Studio では、**C1ScatterChart** コントロールにスマートタグが用意されています。スマートタグは、**C1ScatterChart** 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。

C1ScatterChart コントロールでは、スマートタグによって、よく使用されるプロパティにすばやく簡単にアクセスできます。

[**C1ScatterChart タスク**]メニューにアクセスするには、**C1ScatterChart** 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、[**C1ScatterChart タスク**]メニューが開きます。

[**C1ScatterChart タスク**]メニューは次のように動作します。

- **データソースの選択**
  項目[データソースの選択]をクリックすると、既存のデータソースや連結する新しいデータソースを選択できるドロップダウンリストが開きます。
- **シリーズ一覧**
  シリーズ一覧 項目を選択した場合、**C1ScatterChart** コントロールに**ScatterChartSeries** メンバを追加したり、削除したりできる[**ScatterChartSeries コレクションエディター**]ダイアログボックスが開きます。**ScatterChartSeries** メンバを追加したら、そのプロパティを変更できます。
- **シリーズスタイル**
  シリーズスタイル項目をクリックすると、**C1ScatterChart** コントロールに**ChartStyle** メンバを追加したり、削除したりできる[**ChartStyle コレクションエディター**]ダイアログボックスが開きます。**ChartStyle** メンバを追加したら、そのプロパティを変更します。
- **シリーズホバースタイル**
  シリーズホバースタイル項目をクリックすると、**C1ScatterChart** コントロールに**ChartStyle** メンバを追加したり、削除したりできる[**ChartStyle コレクションエディター**]ダイアログボックスが開きます。**ChartStyle** メンバを追加したら、そのプロパティを変更します。
- **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。
- **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。

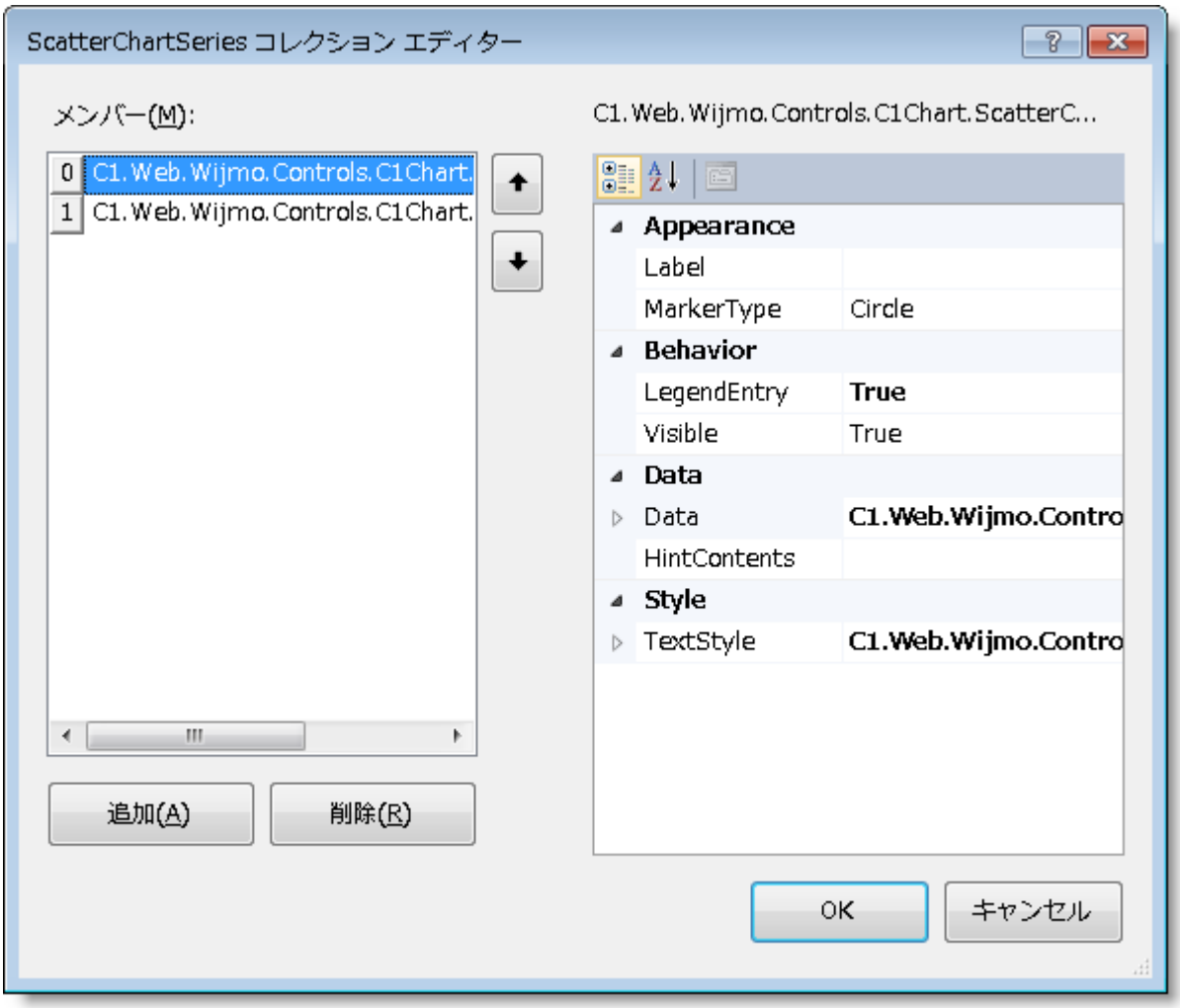
C1ScatterChart コレクションエディター

C1ScatterChart には、系列とグラフスタイルを追加／削除するために以下のコレクションエディターが用意されています。

- **ScatterChartSeries コレクションエディター**
- **ChartStyle コレクションエディター**

ScatterChartSeries コレクションエディター

The **ScatterChartSeries コレクションエディター**によって、ユーザーは選択した **ScatterChartSeries** メンバを**C1ScatterChart** コントロールに対して追加または削除できます。ScatterChartSeries メンバが追加されたら、そのプロパティを変更できます。



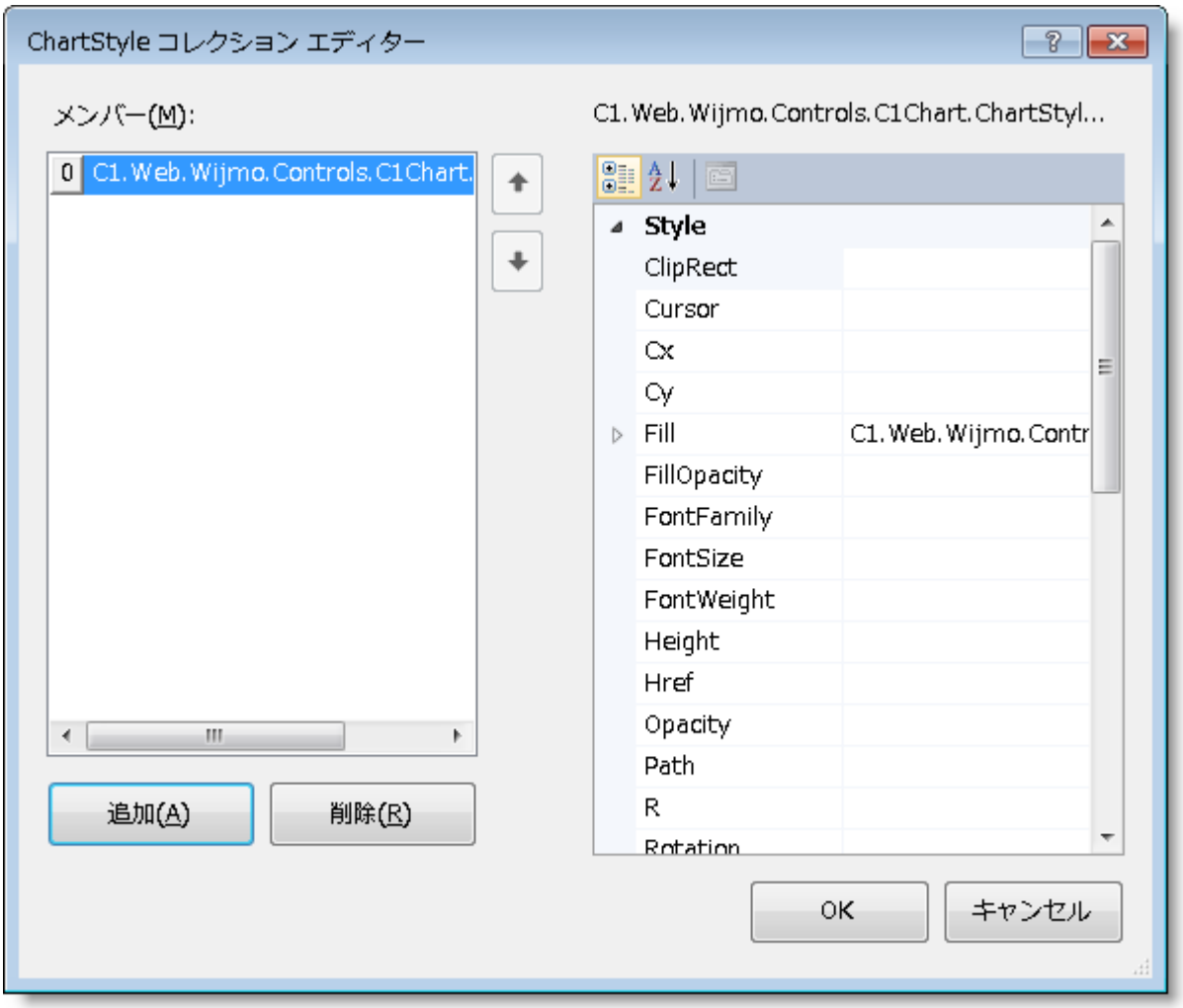
ScatterChartSeries コレクションエディターにアクセスするには、以下の手順を実行します。

1. **C1ScatterChart** コントロールを選択して、そのスマートタグをクリックします。
2. [**C1ScatterChart タスク**]メニューで、シリーズ一覧項目を選択します。これによって、[**ScatterChartSeries コレクションエディター**]ダイアログボックスが開きます。

ChartStyle コレクションエディター

The **ChartStyle コレクションエディター**によって、ユーザーは **ChartStyle** メンバを**C1ScatterChart** コントロールに対して追加または削除できます。**ChartStyle** メンバが追加されたら、そのプロパティを変更できます。同様に、ホバースタイルを

ChartStyle メンバに追加できます。



ChartStyle コレクションエディターにアクセスするには、以下の手順を実行します。

1. **C1ScatterChart**コントロールを選択して、そのスマートタグをクリックします。
2. [**C1ScatterChart タスク**]メニューで、シリーズスタイル項目を選択します。これによって、[**ChartStyle コレクションエディター**]ダイアログボックスが開きます。

C1ScatterChart の基礎

散布図は、クラスタ内かまたはグラフ領域全体に散らばった単純な数値データポイントを表示するグラフの1つです。散布図には変数間の非線形関係を表示する際に大きな視覚効果があります。統計、指数、および

折れ線グラフでは非数値データも表示できるのに対し、散布図の x 軸には数値データしか表示できません。

散布図の最大の特徴は変数間の非線形関係を示すことができる点です。

C1ScatterChart の要素

このセクションでは、C1ScatterChart コントロールを構成する各要素の概要を視覚的に説明します。各トピックは、C1ScatterChart コントロールのさまざまな特徴を表す各種の要素に分類されます。

軸

ChartAxes オブジェクトの X、Y の各プロパティは、グラフの軸の外観をカスタマイズできる **ChartAxis** オブジェクトを返します。各軸は、ChartAxes プロパティのサブプロパティである **X** と **Y** によって表されます。これらのプロパティは、それぞれ次のような主なプロパティを持つ ChartAxis オブジェクトを返します。

Axis プロパティ	説明
Alignment	軸テキストの配置を示す値。このプロパティのデフォルト値は ChartAxisAlignment.Center です。
AnnoFormatString	注釈の書式文字列を示す値。
AnnoMethod	注釈のメソッドを示す値。このプロパティのデフォルト値は ChartAxisAnnoMethod.Values です。
AutoMajor	主目盛記号の値を自動的に計算するかどうかを示す値。このプロパティのデフォルト値は true です。
AutoMax	軸の最大値を自動的に計算するかどうかを示す値。
AutoMin	軸の最小値を自動的に計算するかどうかを示す値。このプロパティのデフォルト値は True です。
AutoMinor	補助目盛記号の値を自動的に計算するかどうかを示す値。このプロパティのデフォルト値は true です。
Compass	軸の方位を示す値。このプロパティのデフォルト値は ChartCompass.South です。
GridMajor	主グリッド線の情報を提供する値。
GridMinor	副グリッド線の情報を提供する値。
Labels	ラベルの情報を提供する値。
Max	軸の最大値を示す値。このプロパティのデフォルト値は 0 です。
Min	軸の最小値を示す値。このプロパティのデフォルト値は 0 です。
Origin	軸の起点の値を示す値。
AxisStyle	軸のスタイルを示す値。
Text	軸のテキストを示す値。
TextStyle	軸のテキストのスタイルを示す値。
TextVisible	軸テキストの表示／非表示を示す値。このプロパティのデフォルト値は True です。
TickMajor	主目盛の情報を提供する値。
TickMinor	補助目盛の情報を提供する値。
UnitMajor	主目盛記号の単位を示す値。このプロパティのデフォルト値は 0 です。
UnitMinor	補助目盛記号の単位を示す値。このプロパティのデフォルト値は 0 です。
ValueLabelList	軸の valueLabels のコレクションを表示する値。
Visible	軸の表示／非表示を示す値。このプロパティのデフォルト値は True です。

軸の位置

通常、軸の注釈は軸のそばに表示されます。これは、起点が軸の最小値や最大値でないグラフで問題となる場合があります。グラフはそのグラフタイプに応じて、さまざまな状況における注釈の配置場所を自動的に決定できます。**Compass** プロパティも注釈の位置関係を指定できます。X 軸の **Compass** 値は North または South、Y 軸の値は East または West に設定できます。デフォルトでは、X 軸は South に設定され、Y 軸は West に設定されます。

軸の外観

Alignment プロパティは、**Center**、**Near**、または **Far** の3つの異なる設定に設定できます。配置を Center に設定すると、軸タイトルはグラフ領域に対して中央に配置されます。配置を Near に設定すると、軸タイトルはグラフ領域の左側に配置されます。配置を Far に設定すると、軸タイトルはグラフ領域の右側に配置されます。

デザイン時に X 軸のラベルを変更するには、**Axis**->**X**->**Labels**->**AxisLabelStyle** を拡張し、**FontSize** プロパティを設定します。

Fill.Color プロパティは軸線、目盛記号、ラベル、およびタイトルの色を変更します。たとえば、デザイン時に X 軸ラベルのフォント色を変更するには、**Axis**->**X**-> **Labels**->**AxisLabelStyle->Fill** を拡張した後、**Color** の横にある〈...〉ボタンをクリックして色を選択します。

以下の例は、設定された後の**Color** プロパティのソースビューを示しています。

ソースビュー

```
<Axis>
      <X Max="2010" AutoMin ="false" Min="2005"  Text="年">
         <TextStyle FontSize="16">
         </TextStyle>
         <Labels>
            <Style FontSize="12">
               <Fill Color="#0033CC">
               </Fill>
            </Style>
         </Labels>
         <TickMajor Position="Outside">
         </TickMajor>
</Axis>
      </X>
```

軸のタイトルと回転

軸にタイトルを追加すると、その軸で何が表示されているかが明確になります。軸に沿ったタイトルまたは注釈も回転できます。

軸タイトルの追加

軸の **Text** プロパティを使用して、タイトルを軸に追加します。タイトルを削除するには、Text プロパティからテキストを削除します。

軸タイトルの回転

Rotation プロパティを使用して、軸タイトルを 90 度、180 度、または 270 度回転します。90 度と 270 度の回転は垂直軸に対して非常に効果的です。デザイン時に X 軸ラベルの回転を変更するには、**Axis**->**X**-> **Labels**->**AxisLabelStyle** を拡張して、**Rotation**プロパティを設定します。

軸の目盛記号

目盛記号は、グラフの測定単位を表示するために軸に直交して表示される線です。**ChartAxisTick.Positon** を Cross または Outside に設定した場合、バブルグラフ上に主目盛記号と補助目盛記号を表示できます。目盛の間隔や属性のカスタマイズは、プロパティセットを操作するだけで容易です。

TickMajor and **TickMinor**の各プロパティは、軸の目盛記号の状態を設定します。このプロパティは、任意の **ChartAxisTickPosition** 値に設定できます。

目盛記号の位置

これらの値は目盛記号の表示位置と表示／非表示を設定します。

値	説明
ChartAxisTickPosition.None	軸の目盛記号なし。
ChartAxisTickPosition.Cross	目盛記号は軸上を横断。
ChartAxisTickPosition.Outside	目盛記号を軸上のグラフ領域の外側に配置。
ChartAxisTickPosition.Inside	目盛記号を軸上のグラフ領域の内側に配置。

目盛記号の間隔

AutoMajor プロパティと**AutoMinor** プロパティは、目盛記号がグラフによって自動的に設定されるかどうかを設定します。これらのプロパティの両方を **True** に設定した場合、グラフは現在のデータを使用して主目盛記号と補助目盛記号を論理的に配置します。**AutoMajor** プロパティが true の場合、軸の注釈の重なりを有効にする必要はありません。

UnitMajorプロパティと**UnitMinor** プロパティは、目盛を配置する際の単位を設定します。**UnitMajor** プロパティを設定すると、**UnitMinor** プロパティはグラフによって **UnitMajor** 値の半分の値に自動的に設定されます。グラフが**UnitMinor** プロパティを自動的に設定しますが、手動で異なる値に変更することもできます。

目盛記号の長さ

Factor プロパティを使用し、主目盛記号と補助目盛記号の長さを伸ばすことができます。**Factor** プロパティを使用する前に、**Position** を **Outside** または **Cross** に設定します。目盛記号は、軸線の太さと目盛係数に基づいてサイズ調整されます。目盛係数を2倍にすると、軸の目盛記号の長さは2倍になります。X 軸の目盛記号に負の値を使用する場合、目盛記号は X 軸ラベルの上側に表示されます。Y 軸の目盛記号に負の値を使用する場合、目盛記号は Y 軸ラベルの左側に表示されます。

> **注意**:**ChartAxisTick.Factor** プロパティの値を増加した場合、適切な **MarginBottom**、**MarginLeft**、**MarginRight**、または **MarginTop** の各プロパティも目盛記号の長さの伸長に合わせて適切なスペースを増加させる必要があります。

軸のグリッド線

グリッド線は、主／副単位間隔ごとに主目盛記号および補助目盛記号と直交して表示されます。主間隔の軸に直交して表示される線は **GridMajor** プロパティによって制御され、副間隔の軸に直交して表示される線は **GridMinor**プロパティによって

制御されます。グリッド線を設定すれば、正確な値を確認する際のグラフの読みやすさが向上します。

軸の範囲

通常、グラフでは、含まれているデータがすべて表示されます。しかし、軸の範囲を調整することによって、グラフの特定の部分を表示することもできます。

グラフでは、最低値と最高値、および数値の増分を考慮することによって、各軸の範囲が決まります。**Min**、**Max**、**AutoMin**、および **AutoMax** の各プロパティを設定すれば、このプロセスをカスタマイズできます。

軸の最小値と最大値

特定の軸の値でグラフの枠を設定するには、**Min** プロパティと**Max** プロパティを使用します。グラフの X 軸の値が 0 ～ 100 の場合、**Min**を 0、**Max** を 10 に設定すると、値は 10 までしか表示されません。

グラフでは、**Min** と **Max**の値を自動的に計算することもできます。**AutoMax** プロパティと**AutoMin**プロパティを **True** に設定した場合、グラフでは、現在のデータセットに合わせて軸の数値が自動設定されます。

軸の注釈

各軸の注釈は、どのようなグラフでも重要な部分です。グラフは、データが変化しても可能なかぎり自然な注釈を自動生成します。

以下のプロパティは、C1ScatterChart の軸の注釈の書式とレイアウトを表します。

Axis プロパティ	**説明**
AnnoFormatString	注釈の書式文字列を示す値。
AnnoMethod	注釈のメソッドを示す値。
ValueLabels	軸の valueLabels のコレクションを表示する値。

値の注釈

値の注釈は、グラフがデータ自体に基づいて数値型の注釈を自動的に生成する実装です。値の注釈は任意の軸に対し、任意のグラフタイプと任意のデータレイアウトで使用できます。値の注釈は軸の以下のプロパティによって制御されます。

プロパティ	**説明**
AnnoFormatString	注釈の書式文字列を示す値。

グラフのラベル

バブルグラフのグラフラベルは、データポイントの各セットの内側またはすぐ外側に C1ScatterChart の x 値と y 値を表示するラベルを表します。

グラフラベルは、重要データポイントを強調表示する際に役立ちますが、一般に情報をデータ上やグラフ上に提供する際にも使用できます。

x、y 値を指定すると、グラフラベルはデータポイントの内側に自動表示されます。グラフを移動する場合、ChartLabels は**ShowChartLabels** プロパティを False に設定して非表示にできます。

グラフラベルの書式設定

グラフラベルは、**ChartLabelFormatString** プロパティを使用して書式設定できます。

グラフラベルの外観

ChartLabelStyle プロパティを使用し、グラフラベルの外観をカスタマイズできます。

ヘッダーとフッター

ヘッダー要素とフッター要素は、グラフに関する説明情報を表示するために使用されます。これらは **Header** プロパティと **Footer** プロパティによって制御されます。

グラフのヘッダーとフッターの各プロパティは、以下の主なプロパティを含む**ChartTitle** オブジェクトを返します。

プロパティ	説明
Compass	タイトルの方位を示す値。これは、タイトルの位置を North（グラフの上側）、South（グラフの下側）、East（グラフの右側）、および West（グラフの左側）から決定します。
Style	タイトルのフォント、向き、色、枠を設定するプロパティを含みます。
Text	タイトルの位置を決定します。
TextStyle	タイトルテキストのスタイルを示す値。
Visible	タイトルを表示するかどうかを決定します。

C1Chart は、タイトルのサイズと位置をそのコンテンツと **Compass** の設定値に基づいて自動的に設定します。

ヘッダー要素とフッター要素のカスタマイズ

ヘッダー要素とフッター要素のテキストと配置、位置、枠、色、フォントは、**ChartTitle**のプロパティを使用してカスタマイズできます。

ヘッダー要素は、**X** プロパティを使用して左側（負の値）または右側（正の値）に移動でき、**Y** プロパティを使用して上側（正の値）または下側（負の値）に移動できます。

凡例

凡例要素は、グラフの各データ系列に関する情報を表示します。グラフの凡例は、物理的な色とデータ系列の間のマッピングを表示します。

C1ScatterChart は、グラフにデータが存在して、**LegendEntry** プロパティが有効の場合は常に凡例を自動的に生成します。**Label** プロパティを指定すると、各系列の名前が凡例に示されます。**Label** プロパティに指定された値が存在しない場合は、系列名は未定義として凡例に表示されます。

凡例は、**Legend** プロパティによって制御され、以下の主なプロパティを持つ **ChartLegend** オブジェクトを返します。

プロパティ	説明
Text	凡例タイトルに表示するテキストを含みます。
Style	凡例のフォント、向き、色、枠を設定するプロパティを含みます。
Compass	凡例の位置を決定します。
Visible	凡例を表示するかどうかを決定します。

Orientation	凡例の項目を水平または垂直方向に表示するかどうかを決定します。

C1Chart は、凡例のサイズと位置をそのコンテンツと **Compass** プロパティおよび **Orientation** プロパティに基づいて自動生成します。

凡例の向きは、**Orientation** プロパティから水平または垂直に設定でき、その位置は **Compass** プロパティから north、south、east、または west に設定できます。

系列

ScatterChartSeries オブジェクトは、散布図にプロットされるデータを表します。1つ以上の ScatterChartSeries を C1ScatterChart 上に持つことができます。各 ScatterChartSeries は異なる色で表されます。ScatterChartSeries は、デザインビューで **ScatterChartSeries コレクションエディター**から追加でき、ソースビューでは ScatterChartSeries 要素から追加できます。また、プログラムによって **ScatterChartSeries** オブジェクトから追加できます。

コードビューでの ScatterChartSeries の追加

1. フォームを右クリックし、[**コードの表示**]を選択してコードファイルを表示してから、次の指示文を追加して C1.Web.Wijmo.Controls.Chart 名前空間を宣言します。

   C#コードの書き方

   C#
   ```
   Using C1.Web.Wijmo.Controls.Chart;
   ```

2. 作成するコードファイルに次の関数を追加し、データを **C1ScatterChart** に追加します。

   C#コードの書き方

   C#
   ```
   private void PrepareOptions()
            {
                    var valuesX = new List<double?>() { 161.2, 167.5, 159.5, 157,
   155.8, 170, 159.1, 166, 176.2, 160.2 };
                    var valuesY = new List<double?>() { 51.6, 59, 49.2, 63, 53.6,
   59, 47.6, 69.8, 66.8, 75.2};

                    //女性の系列一覧


                    var series = new ScatterChartSeries();
                    this.C1ScatterChart1.SeriesList.Add(series);
                    series.MarkerType = MarkerType.Circle;
                    series.Data.X.AddRange(valuesX.ToArray<double?>());
                    series.Data.Y.AddRange(valuesY.ToArray<double?>());
                    series.Label = "Female";
                    series.LegendEntry = true;

                    //男性の系列一覧


                    valuesX = new List<double?>() { 174, 175.3, 193.5, 186.5,
   187.2, 181.5, 184, 184.5, 175, 184};
   ```

```
                valuesY = new List<double?>() { 65.6, 71.8, 80.7, 72.6, 78.8,
74.8, 86.4, 78.4, 62, 81.6};

                series = new ScatterChartSeries();
                this.C1ScatterChart1.SeriesList.Add(series);
                series.MarkerType = MarkerType.Diamond;
                series.Data.X.AddRange(valuesX.ToArray<double?>());
                series.Data.Y.AddRange(valuesY.ToArray<double?>());
                series.Label = "Male";
                series.LegendEntry = true;
            }
```

3. コードページに以下のコードを追加し、ポストバックされないときに関数を呼び出します。

C#コードの書き方

C#

```
protected void Page_Load(object sender, EventArgs e)
{
   if (!IsPostBack)
   {
     PrepareOptions();
   }
}
```

デザインビューでの ScatterChartSeries の追加

1. **C1ScatterChart** コントロールを選択して、そのスマートタグをクリックします。
2. [**C1ScatterChart タスク**]メニューで、シリーズ一覧項目を選択します。これによって、[**ScatterChartSeries コレクションエディター**]ダイアログボックスが開きます。
3. 〈**追加**〉をクリックし、**ScatterChartSeries** メンバを **シリーズ一覧コレクション**に追加します。

ソースビューでの ScatterChartSeries の追加

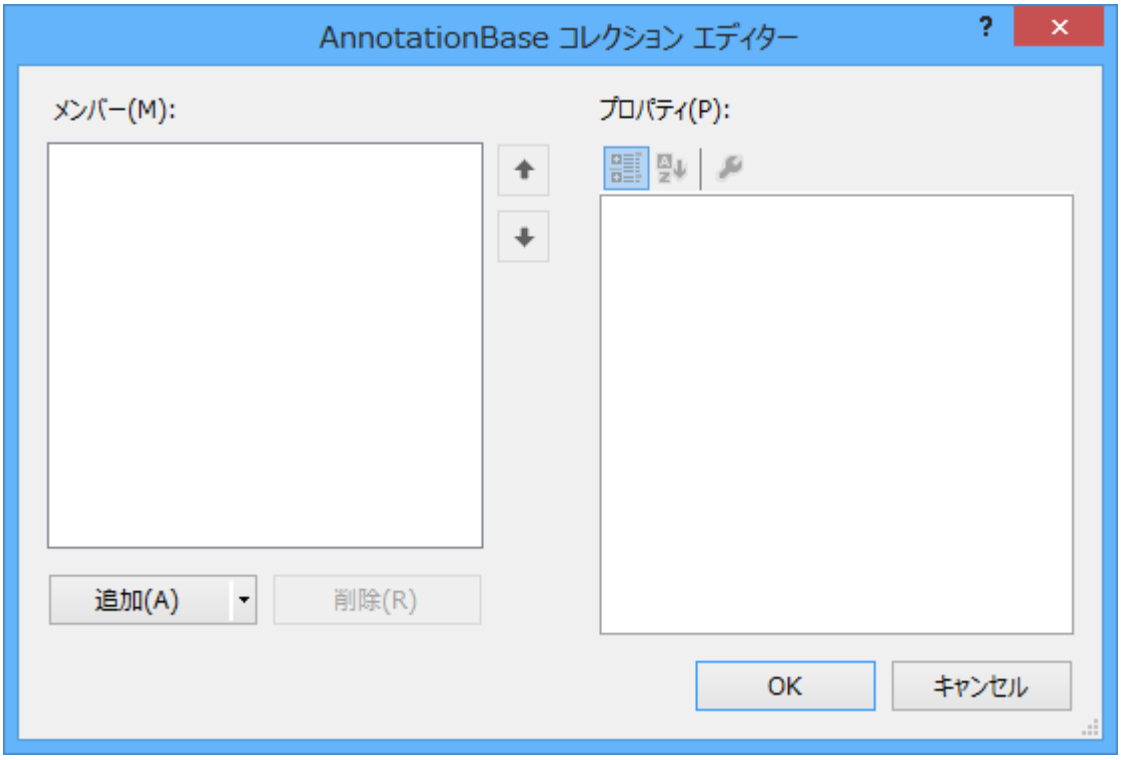
ソースビュー

```
<SeriesList>
 <cc1:ScatterChartSeries Label="女" LegendEntry="True">
   <Data>
     <X>
       <Values>
         <cc1:ChartXData DoubleValue="161.4" />
         <cc1:ChartXData DoubleValue="169.0" />
         <cc1:ChartXData DoubleValue="166.2" />
         <cc1:ChartXData DoubleValue="159.4" />
         <cc1:ChartXData DoubleValue="162.5" />
         <cc1:ChartXData DoubleValue="159.0" />
         <cc1:ChartXData DoubleValue="162.8" />
         <cc1:ChartXData DoubleValue="159.0" />
         <cc1:ChartXData DoubleValue="179.8" />
         <cc1:ChartXData DoubleValue="162.9" />
       </Values>
     </X>
```

```
    <Y>
      <Values>
          <cc1:ChartYData DoubleValue="63.4" />
          <cc1:ChartYData DoubleValue="58.2" />
          <cc1:ChartYData DoubleValue="58.6" />
          <cc1:ChartYData DoubleValue="45.7" />
          <cc1:ChartYData DoubleValue="52.2" />
          <cc1:ChartYData DoubleValue="48.6" />
          <cc1:ChartYData DoubleValue="57.8" />
          <cc1:ChartYData DoubleValue="55.6" />
          <cc1:ChartYData DoubleValue="66.8" />
          <cc1:ChartYData DoubleValue="59.4" />
      </Values>
    </Y>
  </Data>
 </cc1:ScatterChartSeries>
<cc1:ScatterChartSeries Label="男" LegendEntry="True">
  <Data>
    <X>
      <Values>
        <cc1:ChartXData DoubleValue="175.0" />
        <cc1:ChartXData DoubleValue="174.0" />
        <cc1:ChartXData DoubleValue="165.1" />
        <cc1:ChartXData DoubleValue="177.0" />
        <cc1:ChartXData DoubleValue="192.0" />
        <cc1:ChartXData DoubleValue="176.5" />
        <cc1:ChartXData DoubleValue="169.4" />
        <cc1:ChartXData DoubleValue="182.1" />
        <cc1:ChartXData DoubleValue="179.8" />
        <cc1:ChartXData DoubleValue="175.8" />
      </Values>
    </X>
    <Y>
     <Values>
        <cc1:ChartYData DoubleValue="70.2" />
         <cc1:ChartYData DoubleValue="73.4" />
         <cc1:ChartYData DoubleValue="70.5" />
         <cc1:ChartYData DoubleValue="68.9" />
         <cc1:ChartYData DoubleValue="102.3" />
         <cc1:ChartYData DoubleValue="68.4" />
         <cc1:ChartYData DoubleValue="65.9" />
         <cc1:ChartYData DoubleValue="75.7" />
         <cc1:ChartYData DoubleValue="84.5" />
         <cc1:ChartYData DoubleValue="87.7" />
    </Values>
     </Y>
  </Data>
 </cc1:ScatterChartSeries>
</SeriesList>
```

注釈

C1BarChartでは、**Annotations** プロパティを設定して注釈を追加できます。注釈を追加またはカスタマイズするために AnnotationBaseコレクションエディタを使用することができます。

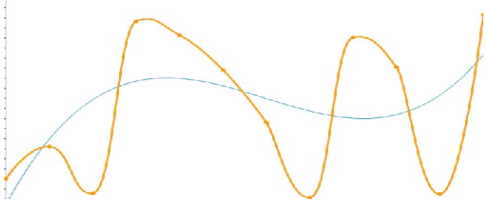
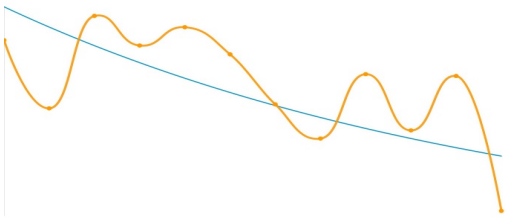
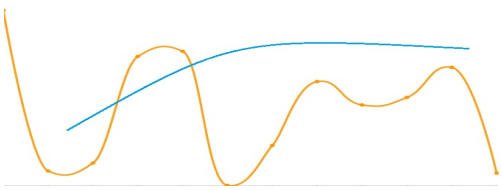
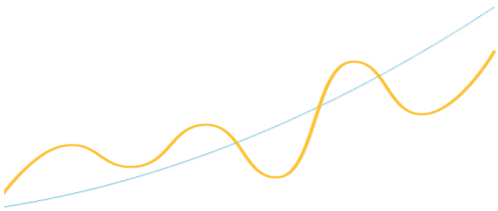


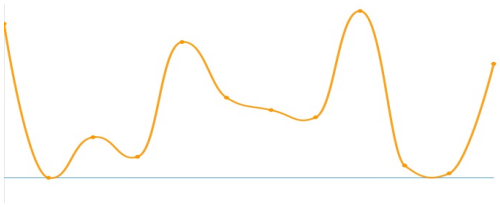
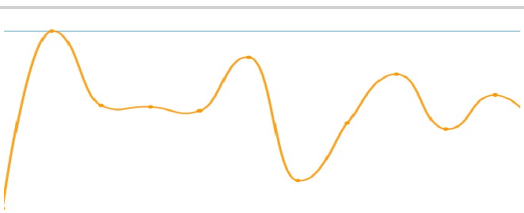
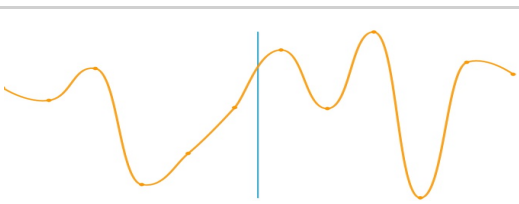
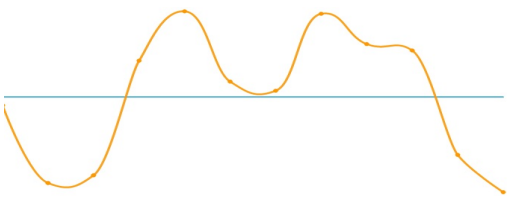
近似曲線

近似曲線は、データの傾向を示し、予測問題の検討に利用されるツールです。価格チャートや金融チャートで使用されるのが多いですが、株価のテクニカル分析で使用される取引指標、MACD(Moving Average Convergence Divergence、移動平均収束拡散法)や金融市場の解析に使用されるテクニカル指標、RSI(Relative Strength Index、相対力指数)のようなテクニカル分析グラフでも使用されます。

近似曲線の種類

以下の一覧で対応するすべてのFitTypesが説明されています。近似曲線の各種類がそれぞれの計算式に基づいて作成されます。

FitType	説明	例画像
Polynom	データが変動する場合に使用される曲線です。大量のデータで利益や損失を分析するのに採用されます。	
Exponent	データ値の増減率が高くなっていく場合に適します。0 または負の値が含まれているデータの場合、指数近似曲線を作成できません。	
Logarithmic	データの可視化に最適する曲線です。データの変化率が急速に増加または減少してから平になる場合に適します。この曲線では負の値と正の値を使用できます。	
Power	特定な比率で増加する測定を比較するデータセットの場合に適します。たとえば、超特急列車の加速を 1 秒間隔で計測した値など。	
Fourier	フーリエ級数式で作成され、波動関数を簡単な正弦波の組み合わせとして表示する方法です。	
Min X	グラフからXの最小値を取得して近似曲線を作成します。	

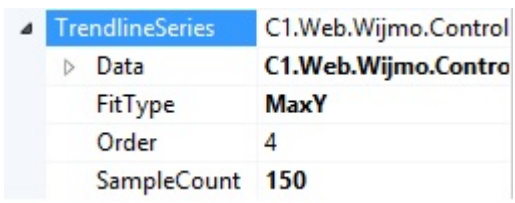
Min Y	>グラフからYの最小値を取得して近似曲線を作成します。	
Max X	グラフからXの最大値を取得して近似曲線を作成します	
Max Y	グラフからYの最大値を取得して近似曲線を作成します。	
Average X	グラフからXの平均値を取得して近似曲線を作成します。	
Average Y	グラフからYの平均値を取得して近似曲線を作成します。	

近似曲線をデザイナまたは、ソースビューやコードでも追加できます。C1BarChartに近似曲線を追加するには、以下の手順に従います。

デザインビューの場合

以下の手順では、チャートにデータが追加されていることを前提します。詳細については、C1BarChartのクイックスタートを参照してください。

1. **ScatterChart** スマートタグをクリックし、[**C1ScatterChart タスク**]メニューから **シリーズ一覧** を選択します。**ScatterChartSeries コレクションエディター**が表示されます。
2. **ScatterChartSeriesコレクションエディター**で、〈**追加**〉ボタンをクリックして、新しい **ScatterChartSeries** をBarChartに追加します。
3. 新規に追加された系列の**isTrendline**プロパティをTrueに設定します。
4. **Trendline Series**のプロパティグループを展開して以下のプロパティを設定できます。



- **FitType** -近似曲線の種類を指定します。
- **SampleCount**-近似曲線の関数計算用にサンプルカウントを指定します。FitTypeがpolynom、power、exponent、logarithmicやfourierの場合のみ機能します。

- **Order** - 多項式の次数を指定します。FitTypeがpolynom、power、exponent、logarithmicやfourierの場合のみ機能します。

5. TrendlineSeries.Dataのプロパティグループにあ るTrendlineSeries.Data.X.Valuesプロパティの横にある〈...〉ボタンをクリックして**ChartXData コレクションエディター**を開きます。
6. X軸に表示される値を入力して、**OK**をクリックします。
7. TrendlineSeries.Dataのプロパティグループにあ るTrendlineSeries.Data.Y.Valuesプロパティの横にある〈...〉ボタンをクリックして**ChartYData コレクションエディター**を開きます。
8. Y軸に表示される値を入力して、**OK**をクリックします。
9. **FitType**プロパティ、**Order**プロパティと**SampleCount**プロパティを設定します。
10. 〈**OK**〉をクリックして保存し、**ScatterChartSeriesコレクションエディター**を閉じます。

ソースビューの場合

<SeriesList> </SeriesList>タグ内に以下のマークアップを追加してグラフに近似曲線を追加します。

SourceView

```
<cc1:ScatterChartSeries  LegendEntry="True" IsTrendline="true">

 <TrendlineSeries FitType="Polynom">
         <Data>
             <X>
                 <Values>
                       <cc1:ChartXData StringValue="QTR1" />
                       <cc1:ChartXData StringValue="QTR2" />
                       <cc1:ChartXData StringValue="QTR3" />
                       <cc1:ChartXData StringValue="QTR4" />
                 </Values>
             </X>
              <Y>
                 <Values>
                     <cc1:ChartYData DoubleValue="13" />
                     <cc1:ChartYData DoubleValue="4"  />
                     <cc1:ChartYData DoubleValue="18" />
                     <cc1:ChartYData DoubleValue="8"  />
                 </Values>
             </Y>
         </Data>
</TrendlineSeries>

</cc1:ScatterChartSeries>
```

コードの場合

Page_Loadイベントに以下のコードを追加することでチャートに近似曲線を追加できます。

C#コードの書き方

C#

```
// 新規系列を作成します。
varseriesTrendline = new ScatterChartSeries();
```

```
seriesTrendline.IsTrendline = true;
seriesTrendline.Label = "Trendline";
seriesTrendline.TrendlineSeries.FitType = TrendlineFitType.Polynom;
seriesTrendline.TrendlineSeries.Order = 4;
seriesTrendline.TrendlineSeries.SampleCount = 100;

// チャートに系列を追加します。
this.C1BarChart1.SeriesList.Add(seriesTrendline);

// X軸のデータを追加します。
seriesTrendline.TrendlineSeries.Data.X.Add("QTR1");
seriesTrendline.TrendlineSeries.Data.X.Add("QTR2");
seriesTrendline.TrendlineSeries.Data.X.Add("QTR3");
seriesTrendline.TrendlineSeries.Data.X.Add("QTR4");

// Y軸のデータを追加します。
seriesTrendline.TrendlineSeries.Data.Y.Add(13);
seriesTrendline.TrendlineSeries.Data.Y.Add(4);
seriesTrendline.TrendlineSeries.Data.Y.Add(18);
seriesTrendline.TrendlineSeries.Data.Y.Add(8);
```

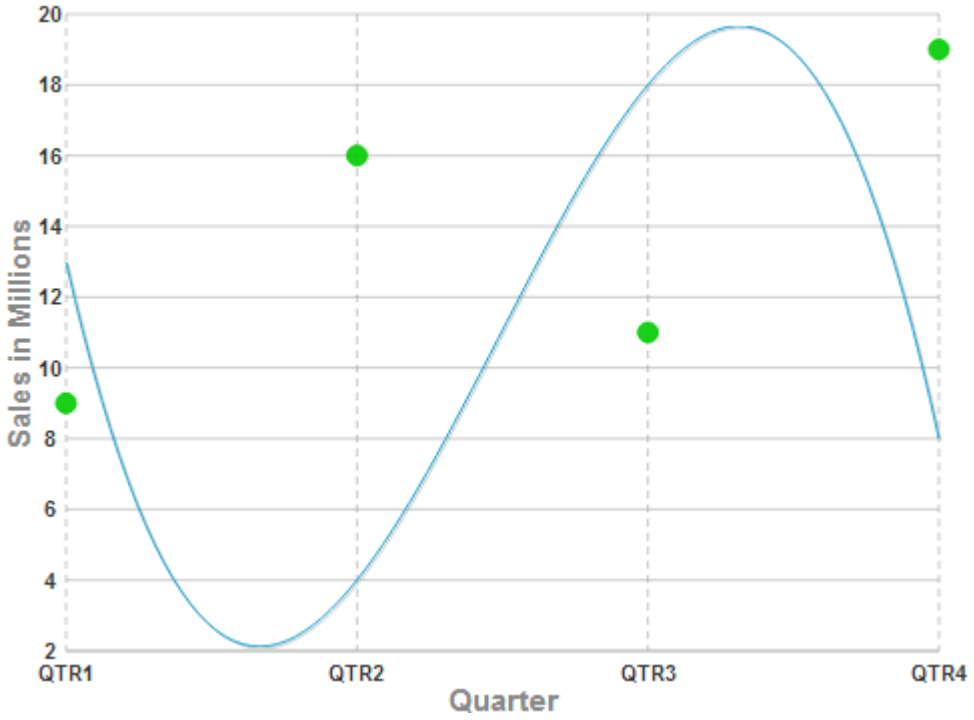
Visual Basicコードの書き方

VB

```
'新規系列を作成します。
Dim seriesTrendline = New ScatterChartSeries()
seriesTrendline.IsTrendline = True
seriesTrendline.Label = "Trendline"
seriesTrendline.TrendlineSeries.FitType = TrendlineFitType.Polynom
seriesTrendline.TrendlineSeries.Order = 4
seriesTrendline.TrendlineSeries.SampleCount = 100

' チャートに系列を追加します。
Me.C1BarChart1.SeriesList.Add(seriesTrendline)

' X軸のデータを追加します。
seriesTrendline.TrendlineSeries.Data.X.Add("QTR1")
seriesTrendline.TrendlineSeries.Data.X.Add("QTR2")
seriesTrendline.TrendlineSeries.Data.X.Add("QTR3")
seriesTrendline.TrendlineSeries.Data.X.Add("QTR4")

' Y軸のデータを追加します。
seriesTrendline.TrendlineSeries.Data.Y.Add(13)
seriesTrendline.TrendlineSeries.Data.Y.Add(4)
seriesTrendline.TrendlineSeries.Data.Y.Add(18)
seriesTrendline.TrendlineSeries.Data.Y.Add(8)
```

このトピックの作業結果

プロジェクトを実行すると、チャートに緑色の近似曲線が表示されます。



データ連結

C1ScatterChart は、サーバー上の外部データソースからのデータ連結をサポートしています。DataBinding は、DataSourceID または DataSource と、DataBindings を設定すると使用可能になります。以下のプロパティは、X 値と Y 値を指定されたデータフィールドに連結するために使用されます。

- DataSourceID
- DataBindings
- C1ChartBinding.XField
- C1ChartBinding.XFieldType
- C1ChartBinding.YField
- C1ChartBinding.YFieldType

DataBindings は、C1ScatterChartBindings のインスタンスを含むコレクションです。C1ScatterChartBinding は以下のプロパティを含みます。

- DataMember - このプロパティは、データソースが複数のリストを含む場合にデータのリストの名前を指定するために使用されます。
- HintField - このプロパティは Hint コンテンツを指定されたフィールド名に連結するために使用されます。HintField を設定した場合、マウスが系列に移動すると、その系列と同じインデックスを持つヒント値を表示します。

エクスポートサービス

C1 ASP.NET Export Service を使用すると、複雑なエクスポートアプリケーションを作成しなくても、チャートを画像または PDF としてにエクスポートできます。このサービスはアプリケーションサーバー上に置かれます。

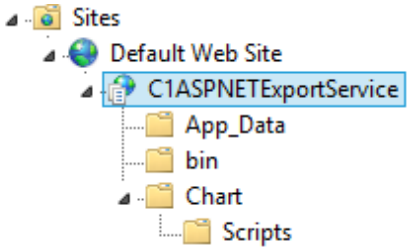
C1 ASP.NET Export Service を使用する利点は次のとおりです。

- 書式設定を維持したままチャートをエクスポートできます。
- エクスポートされたファイルの設定を必要に応じて変更できます。

トピックの内容

インストール

Export Service は、インターネットインフォメーションサービス(IIS)上に配布される Web アプリケーションです。C:\Program Files\ComponentOne\ASP.NET Web Forms フォルダにある **C1ASPNETExportService** インストーラを実行してください。次のファイルが IIS にインストールされます。



これらのファイルは、次の場所にも格納されています。

C:\ProgramData\ComponentOne\C1ASPNET\C1APNETExportService

システム要件

サービスホストのシステム要件は次のとおりです。

- Microsoft Windows 7 以上。
- ASP.NET 4.0 以上(.NET Framework 4.0)を含む IIS 7.0 以上。
- サービスホスト上に IE9 以上。

> **Microsoft Windows 7** または **Microsoft Windows Server 2008 R2** では .Net Framework 4.0 をアップデートできます。詳細については、「http://support.microsoft.com/kb/2468871」を参照してください。

チャートのエクスポート設定

チャートをエクスポートするための設定は次のとおりです。

画像としてエクスポート

- **ファイル形式**: *.jpg*、*.bmp*、*.gif*、*.png*、または *.tiff* 画像としてエクスポートします。
- **作成者**: データの作成に対して責任を負う人または組織の名前を指定します。
- **サーバー URL**: サーバーの URL を設定します。"<サーバーの URL>**/exportapi/chart**" を入力します。
- **ファイル名**: エクスポートされる画像に使用するファイル名を設定します。

PDF にエクスポート

- **幅の自動調整**: 自動調整を有効にします。
- **横**: 横長モードを有効にします。
- **ファイルの内容**:
  - 画質: 画質を Low(低)、Medium(中)、または High(高)に設定します。
  - 圧縮: 圧縮レベルを Default(デフォルト)、None(なし)、Best Speed(速度優先)、または Best Compression(圧縮率優先)に設定します。

- フォントタイプ：フォントタイプを True Type または Embedded（埋め込み）に設定します。

- **ドキュメント情報**：
  - 作成者：ドキュメントを作成した人または組織の名前を設定します。
  - 作成元アプリケーション：元のドキュメントを作成したアプリケーションの名前を設定します。
  - サブタイトル：ドキュメントのサブタイトルを設定します。
  - タイトル：タイトルバーに表示されるドキュメントのタイトルを設定します。
  - PDF 作成アプリケーション：PDF ドキュメントを作成したアプリケーションの名前を設定します。
  - キーワード：PDF ドキュメントに関連付けられるキーワードを設定します。これらのキーワードは、ドキュメントの検索に使用できます。
- **ドキュメントのセキュリティ**：
  - 暗号化の種類：暗号化の種類を NotPermit（許可しない）、Standard40、Standard128、または Aes128 に設定します。
  - 所有者パスワード：ドキュメントの権限を編集するために必要なパスワードを設定します。
  - ユーザーパスワード：ドキュメントを開くために必要なパスワードを設定します。
  - コンテンツのコピーを許可：コンテンツのコピーを許可または禁止します。
  - 注釈の編集を許可：ユーザーが注釈を編集することを許可または禁止します。
  - コンテンツの編集を許可：ユーザーがドキュメントの内容を編集することを許可または禁止します。
  - 印刷を許可：ドキュメントの印刷を許可または禁止します。
- **構成設定**：
  - サーバー URL：サーバーの URL を設定します。"<サーバーの URL>**/exportapi/chart**" を入力します。
  - ファイル名：エクスポートされる PDF に使用するファイル名を設定します。

用途

チャートを画像または PDF としてエクスポートするには、`exportChart` メソッドを呼び出します。チャートを画像にエクスポートするには、`<head>` タグと `</head>` タグの間に次のコードを追加します。

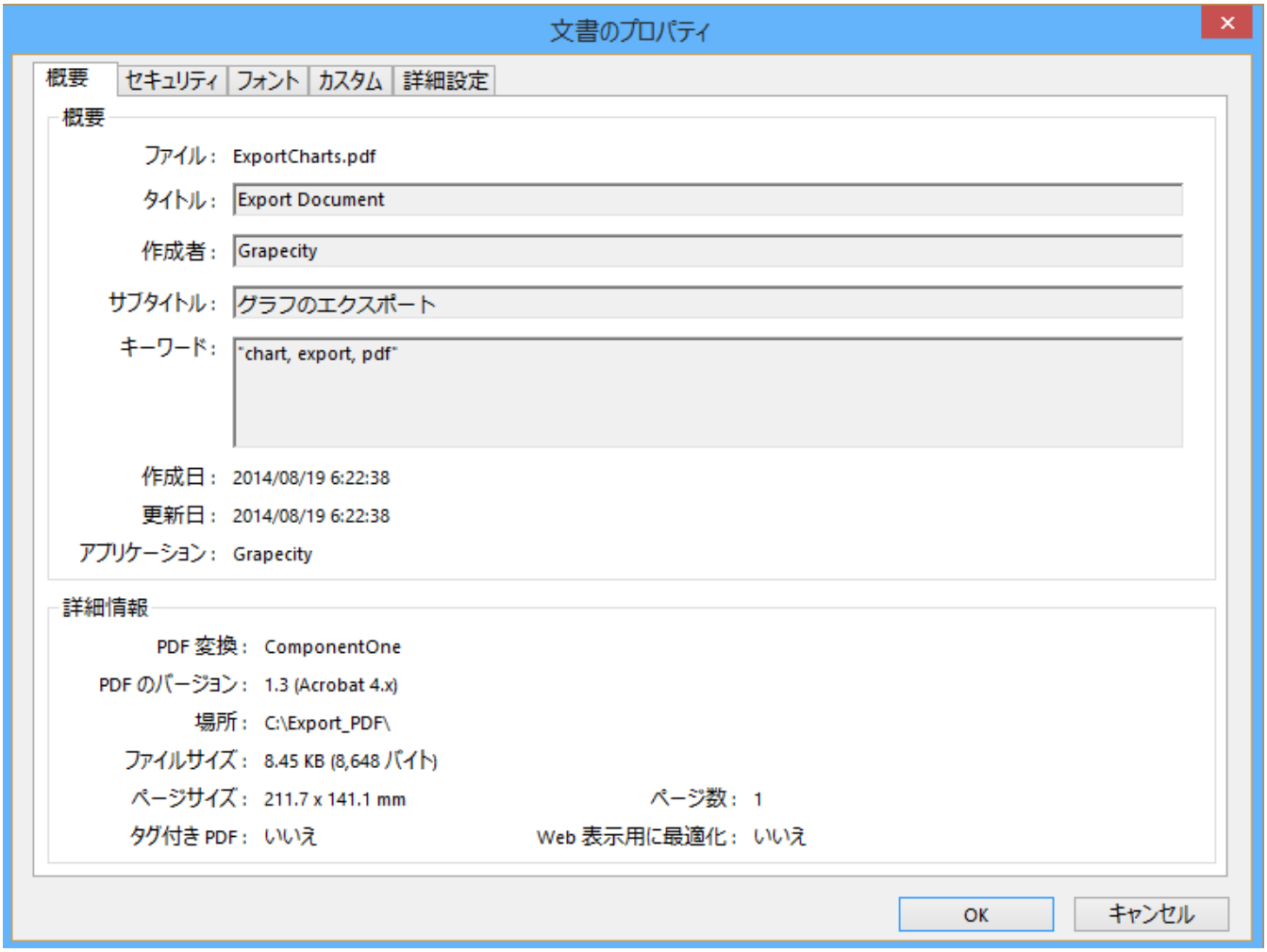
```
    <script src="http://code.jquery.com/jquery-1.9.1.min.js" type="text/javascript">

</script>
<asp:PlaceHolder runat="server">
    <!--Export メソッド-->
   <script type="text/javascript">
       $(function () {
           $("#Button1").click(exportImage);
       });
       function getChart() { return $("#<%=C1ScatterChart1.ClientID%>"); }

       // Export 関数
       function exportImage() {
   var fileName ="ExportImage";
   var type = "Png";
   var url = "http://demos.componentone.com/ASPNET/ExportService" + "/exportapi/chart";
   var chart = getChart();
$("#<%=C1BarChart1.ClientID%>").C1ScatterChart("exportChart", fileName, type, url);
    }
</script>
    </asp:PlaceHolder>
```

チャートを PDF にエクスポートするには、`<head>` タグと `</head>` タグの間に次のコードを追加します。

```
<script src="http://code.jquery.com/jquery-1.9.1.min.js" type="text/javascript">

</script>
    <asp:PlaceHolder runat="server">
<script type="text/javascript">
    $(function () {
        $("#Button1").click(exportPdf);
    });
```

```
    // Export 関数
    function exportPdf() {
    var fileName = "ExportCharts";
    var url = "http://demos.componentone.com/ASPNET/ExportService" + "/exportapi/chart";
    var pdfSetting = {
        imageQuality: 'Low',
        compression: 'BestCompression',
        fontType: 'TrueType',
        author: 'Grapecity',
        creator: 'Grapecity',
        subject: 'グラフのエクスポート',
        keywords: 'chart, export, pdf',
        allowCopyContent: true,
        allowEditAnnotations: true,
        allowEditContent: true,
        allowPrint: true
       }

$("#<%=C1ScatterChart1.ClientID%>").C1ScatterChart("exportChart", fileName, "pdf",

       pdfSetting, url);
       }
</script>
        </asp:PlaceHolder>
```

生成される PDF のプロパティを次の図に示します。



ファイルが Internet Explorer でダウンロードされない場合は、Internet Explorer の保護モードをオフにしてファイルをエクスポー

トするか、Internet Explorer を管理者として実行します。保護モードをオフにするには、次のようにします。

- Internet Explorer の**[設定]**を開き、**[インターネットオプション]**を選択します。
- **[セキュリティ]**タブで、**[インターネット]**を選択し、**[保護モードを有効にする]**のチェックを外します。

C1ScatterChart のアニメーション

C1ScatterChart の系列は、**Duration** プロパティと **Easing** プロパティを使用してアニメーション化できます。

遷移効果

Enabled プロパティが true のとき、アニメーション効果をバブルグラフの系列に適用できます。アニメーション化されたスライド状態／フェード状態の間に遷移効果を追加すれば、それらの状態間にシームレスな流れが生まれ、バブルグラフの魅力を高めることができます。ロード時に左から右へスムーズに移動するバブルグラフ系列の代わりに、バブルグラフを系列のスライドイン時にバウンドインさせ、系列のスライドアウト時にバウンドアウトさせることができます。デフォルトでは、**Easing** プロパティは **EaseLinear** に設定され、棒グラフをリロードすると、各系列はスムーズで直線的な遷移効果でリロードされます。

以下の遷移効果は、状態間の遷移をアニメーション化するために使用できます。これにより、棒グラフ系列をロードする際にユーザーにとって動きがスムーズに見えます。

遷移の名前	遷移の説明
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。
EaseInCubic	3次型のイージングイン。開始は速度ゼロで、それから加速します。
EaseInOutCubic	3次型のイージングインとイージングアウト。開始は速度ゼロで、途中まで加速し、それから再び速度ゼロまで減速します。
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。
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。バウンドの回数は持続時間に関係します。持続時間が延びれば、バウンドの回数は多くなります。
EaseOutCubic	3次型のイージングインとイージングアウト。開始は全速で、それからゼロまで減速します。
EaseOutElastic	5次型のイージングアウト。開始は全速で、それからゼロまで減速します。これはデフォルト設定です。

アニメーション効果の持続時間

C1ScatterChart のアニメーション効果の長さは、**ChartAnimation** 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。