ReportViewer for WPF/Silverlight

2018.04.11 更新

グレープシティ株式会社

目次

製品の概要

WPF アプリケーションにレポート表示機能を追加します。**ReportViewer for WPF/Silverlight** は、Microsoft SQL Server Reporting Services や **C1Report** など、事実上あらゆるレポートサービスで生成された HTML および PDF ベースのレポートを表示することができます。この強力なビューアを使用して、ユーザーは、レポートを表示、検索、ズーム、選択、印刷、およびローカルファイルに保存することができます。

> **メモ**: 説明内に含まれるクラスおよびメンバーに対するリファレンスへのリンクは、原則としてWPF版のリファレンスページを参照します。Silverlight版については、目次から同名のメンバーを参照してください。

主な特長

WPF/Silverlight アプリケーションにレポート表示機能を追加します。**ReportViewer for WPF/Silverlight** は、Microsoft SQL Server Reporting Services や **C1Report** などのレポートサービスで生成された HTML および PDF ベースのレポートを表示することができます。この強力なビューアを使用して、ユーザーは、レポートを表示、検索、ズーム、選択、印刷、およびローカルファイルに保存することができます。

ReportViewer for WPF/Silverlight 使用すると、機能豊富でカスタマイズされたアプリケーションを作成できます。**ReportViewer for WPF/Silverlight** は、次の主な特長を備えています。

- **複数のソースから生成されたレポートの表示**
  C1ReportViewer コントロールは、最もよく使用されるドキュメント形式である HTML と PDF をサポートするため、エンジンに依存しません。**C1ReportViewer** を使用して、様々なレポートジェネレータで生成されたレポートを表示できます。これには、**C1Report**、Microsoft Reporting Services などの HTML 出力または PDF 出力を生成できるレポートプロバイダが含まれます。

- **ドキュメントのロードと保存**
  生成されたレポートの表示に加え、**C1ReportViewer** を使用して、PDF ドキュメントや HTML ドキュメントをロードできます。ロードしたファイルを保存することもできます。

- **印刷のサポート**
  C1ReportViewer を使用して、ユーザーが現在のドキュメント全体を印刷したり、選択したページを印刷することができます。また、**PrintDocument** メソッドを使用して、コードから直接印刷できます。

- **テキスト検索**
  ユーザーは、ドキュメント内でテキスト検索を実行できます。一致したテキストが見つかると、それらが表示され、ユーザーはそれらの検索結果の位置まですばやく直感的に移動することができます。

- **複数表示モード**
  C1ReportViewer は、複数の表示モードを備えており、ドキュメントを任意のスケールで表示できます。ユーザーは、ページがビューに収まるようにズームレベルを設定できます。1ページのみを表示したり、複数のページを並べて表示できます。

- **ページのカスタマイズ**
  ページサイズやマージンの幅などのページのプロパティを指定します。ページテンプレートを設計して、レポートの一部としては生成されないカスタムヘッダーやカスタムフッターを提供することもできます。

- **ツールバーのカスタマイズ**
  ReportViewer には、すばやく開発を実行できるようにデフォルトのツールバーが付属しています。デフォルトのツールバーの各ボタンは、コントロールのコマンドに対応しているため、**ReportViewer** のカスタムツールバーの作成はたいへん簡単です。

- **Silverlight Toolkit テーマ**
  ExpressionDark、ExpressionLight、WhistlerBlue、RainierOrange、ShinyBlue、BureauBlack など、最もよく使用されている Microsoft Silverlight Toolkit テーマが組み込みでサポートされており、それを使用して UI にスタイルを追加できます。

- **添付ファイルを含むPDFファイルのロードと表示**
  C1PDFViewerを使用して、添付ファイルを含むPDFファイルをロードと表示することができます。添付ファイルは、クリックして簡単に見ることができるクリップ形状アイコンの形で表示されます。

クイックスタート

このクイックスタートは、**ReportViewer for WPF/Silverlight** を初めて使用するユーザーのために用意されています。このクイックスタートでは、**C1ReportViewer** コントロールを使用して、簡単なプロジェクトを作成します。新しい WPF/Silverlight アプリケーションを作成し、**C1ReportViewer** コントロールをアプリケーションに追加します。**C1ReportViewer** コントロールに表示される PDF ファイルを追加し、**ReportViewer for WPF/Silverlight** に対して実行可能ないくつかの操作を確認してみます。

手順 1:アプリケーションの作成

この手順では、**ReportViewer for WPF/Silverlight** を使用して WPF/Silverlight アプリケーションを作成します。アプリケーションに **C1ReportViewer** コントロールを追加すると、PDF ファイルと HTML ファイルを内部に表示できる完全な機能を備えたドキュメントビューアインターフェイスになります。プロジェクトをセットアップし、**C1ReportViewer** コントロールをアプリケーションに追加するには、次の手順に従います。

1. Visual Studio の**[ファイル]**メニューから、**[新規作成]**を選択し、**[プロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、左側のメニューから言語と **WPF/Silverlight アプリケーション**を選択します。[**フレームワーク**]ドロップダウンリストで**[.NET Framework 4]**を選択し、プロジェクトの名前を入力します。必要に応じて、WPF/Silverlight のバージョン番号として WPF 4/Silverlight 5 を選択します。この例では、アプリケーションに "QuickStart" という名前を付けます。プロジェクトに別の名前を付けた場合は、後の手順で "QuickStart" を参照している箇所を実際のプロジェクトの名前に変更する必要があります。
3. ソリューションエクスプローラで、プロジェクト名を右クリックし、**[参照の追加]**を選択します。**[参照の追加]**ダイアログボックスで、**C1.WPF**と **C1.WPF.ReportViewer** /**C1.Silverlight** と **C1.Silverlight.ReportViewer** アセンブリを見つけて選択し、**[OK]**をクリックしてプロジェクトに参照を追加します。
4. MainWindow.xaml ファイルの XAML ビューを開きます。このクイックスタートでは、XAML マークアップを使用して C1ReportViewer コントロールを追加します。
5. 次のマークアップを使用して、XAML 名前空間を Window タグに追加します。
   xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
   名前空間は次のようになります。

WPF

```
<Window x:Class="MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
```

Silverlight

```
<UserControl x:Class="QuickStart.MainPage"
xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
xmlns:d=http://schemas.microsoft.com/expression/blend/2008
xmlns:mc=http://schemas.openxmlformats.org/markup-compatibility/2006
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml" mc:Ignorable="d"
d:DesignHeight="300" d:DesignWidth="400">
```

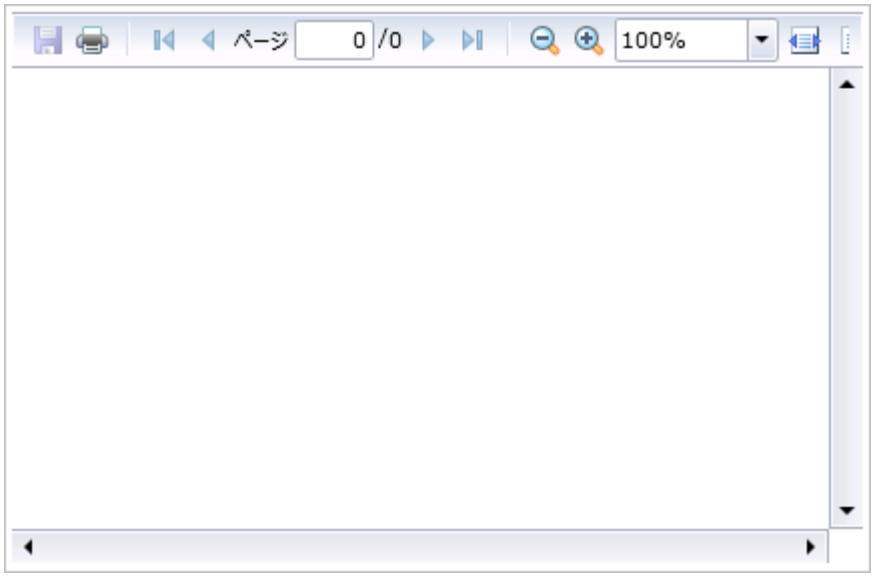
これは、複数の名前空間を追加しなくても、ほとんどの ComponentOne WPF/Silverlight コントロールを使用できるようにするための統合名前空間です。

6. ページの Grid タグ内に <c1:C1ReportViewer x:Name="C1ReportViewer1" /> タグを追加して、アプリケーションに **C1ReportViewer** コントロールを追加します。
XAML は次のようになります。

XAML

```
<Grid x:Name="LayoutRoot" Background="White"> <c1:C1ReportViewer
x:Name="C1ReportViewer1" /> </Grid>
```

これで、"C1ReportViewer1" という名前の **C1ReportViewer** コントロールがアプリケーションに追加されます。アプリケーションを実行すると、次の画像のように表示されます。



これで、アプリケーションのユーザーインターフェイスが正しくセットアップされましたが、このアプリケーションを実行すると、**C1ReportViewer** コントロールにコンテンツがないことがわかります。次の手順では、**C1ReportViewer** コントロールにコンテンツを追加し、コントロールに対して実行可能ないくつかの操作を確認してみます。

手順 2:コンテンツの追加

前の手順では、WPF/Silverlight アプリケーションを作成し、**C1ReportViewer** コントロールをプロジェクトに追加しました。この手順では、**C1ReportViewer** コントロールに PDF コンテンツを追加します。この手順では、**ComponentOne** のサンプルに付属する PDF ファイルを追加します。必要に応じて、別の PDF ファイルを使用し、それに合わせて手順を変更することもできます。プロジェクトをカスタマイズしてアプリケーションの **C1ReportViewer** コントロールに PDF ファイルを追加するには、次の手順に従います。

1. ソリューションエクスプローラで、プロジェクト名を右クリックし、**[追加]**→**[既存の項目]**を選択します。
2. **[既存の項目の追加]**ダイアログボックスで、**ControlExplorer** サンプル内にある **見積書.pdf** ファイルを見つけます。必要に応じて、ファイルの種類ドロップダウンボックスで**[すべてのファイル]**を選択して、PDF ファイルを表示します。別の PDF ファイルを選択して使用してもかまいません。
3. ソリューションエクスプローラで、アプリケーションに追加したばかりの PDF ファイルをクリックします。**[プロパティ]**ウィンドウで、**ビルド アクション** プロパティを **Resource** に設定し、**[出力ディレクトリにコピー]**項目が**[コピーしない]**に設定されていることを確認します。

4. ページを右クリックしてコードビューに切り替え、**[コードの表示]**を選択します。次の手順では、XAML マークアップをアプリケーションに追加することで、ドロップダウンボックスにコンテンツを追加します。
5. ページの先頭に次の imports 文を追加します。

WPF

VisualBasic

```
Imports C1.WPF.ReportViewer
```

C#

```
using C1.WPF.ReportViewer;
```

Silverlight

VisualBasic

```
Imports C1.Silverlight.ReportViewer
```

C#

```
using C1.Silverlight.ReportViewer;
```

6. メインクラスに次のコードを追加します。

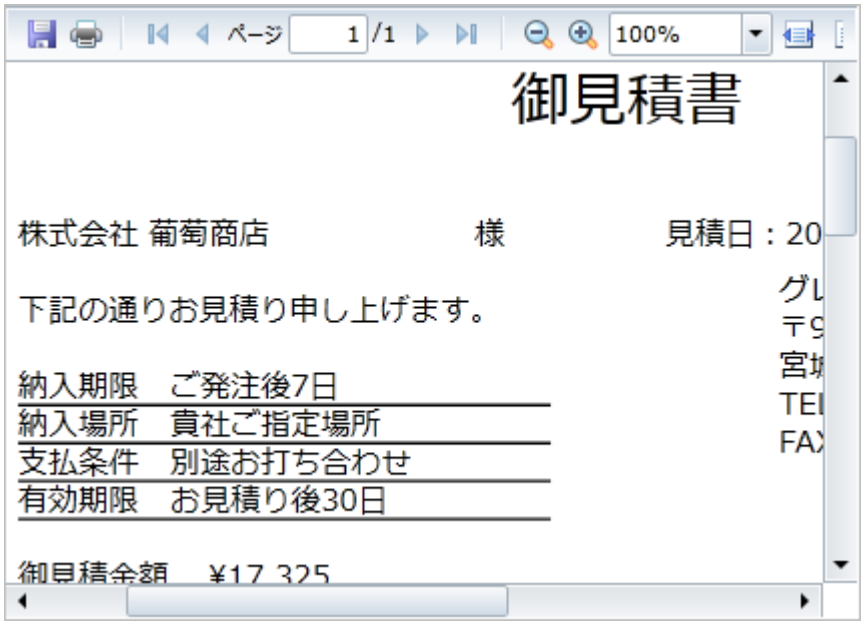
VisualBasic

```
Public Sub New()
    InitializeComponent()
    Dim resource = Application.GetResourceStream(New Uri
("QuickStart;component/C1XapOptimizer.pdf", UriKind.Relative))
Me.C1ReportViewer1.LoadDocument(resource.Stream)
End Sub
```

C#

```
public MainPage()
{
    InitializeComponent();
    var resource = Application.GetResourceStream(new Uri
("QuickStart;component/C1XapOptimizer.pdf", UriKind.Relative));
    this.C1ReportViewer1.LoadDocument(resource.Stream);
}
```

このコードは、ストリームを追加し、それを **C1ReportViewer** コントロールに読み込みます。アプリケーションに別の名前を付けた場合は、"QuickStart" をプロジェクトの名前に置き換える必要があります。別の PDF ファイルを追加した場合は、「見積書.pdf」を実際のファイル名に置き換えてください。ここでアプリケーションを実行すると、**C1ReportViewer** コントロール内のコンテンツウィンドウにアプリケーションが表示されます。

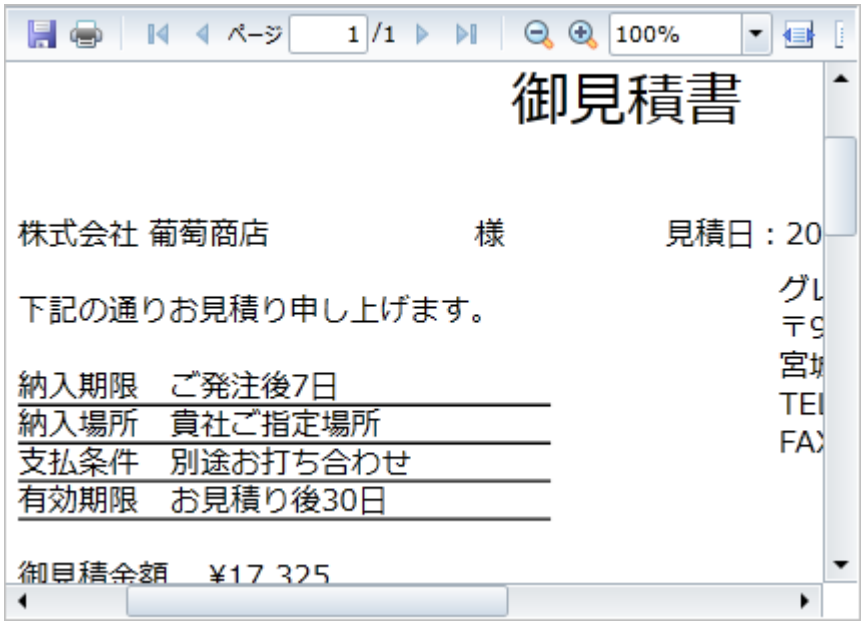


この手順では、**C1ReportViewer** コントロールにコンテンツを追加しました。次の手順では、このコントロールで可能な実行時の操作をいくつか示します。

手順 3：アプリケーションの実行

WPF/Silverlight アプリケーションを作成し、**C1ReportViewer** コントロールにコンテンツを追加しました。後は、アプリケーションを実行するだけです。アプリケーションを実行して **ReportViewer for WPF/Silverlight** の実行時の動作を確認するには、次の手順に従います。

1. **[デバッグ]**メニューから**[デバッグ開始]**を選択し、実行時にアプリケーションがどのように表示されるかを確認します。アプリケーションは次の図のように表示されます。



C1ReportViewer コントロールには、1つのツールバーとコンテンツ領域が表示されます。追加した PDF ファイルは、コントロールのコンテンツ領域に表示されます。

2. ツールバーで、**[次のページ]**矢印ボタンをクリックして、PDF ファイルの次のページに移動します。**[前のページ]**矢印ボタンをクリックして、前のページに戻ることができます。**[最初のページ]**ボタンと**[最後のページ]**ボタンを使用して、ドキュメントの最初のページまたは最後のページに移動することもできます。

3. **[ズームアウト]**ボタンをクリックして、PDF のより広い範囲をウィンドウに表示します。また、**[ズーム]**ドロップダウンボックスをクリックし、ズームレベルを選択することもできます。

4. **[幅に合わせる]**ボタンをクリックして、PDF ファイルの幅をビューアのコンテンツウィンドウのサイズに自動的に合わせます。他のオプションには、有効なスペースにページ全体を表示する**[単一ページ]**、有効なスペースにドキュメントの2ページを表示する**[見開きページ]**があります。

5. **[検索]**テキストボックスをクリックし、"金額" などの検索対象テキストを入力します。その単語の次の出現位置までドキュメントがスクロールされ、PDF ファイルでその単語が強調表示されます。ツールバーには、その語句の出現個数も表示されます。**[前を検索]**ボタンと**[次を検索]**ボタンをクリックして、その単語の前の出現位置または次の出現位置に移動できます。

6. **[保存]**ボタンをクリックします。**[名前を付けて保存]**ダイアログボックスで、ファイルの名前を入力し、**[保存]**ボタンをクリックして、ファイルを任意の場所に保存します。必要に応じて、ツールバーの**[印刷]**ボタンをクリックして、ファイルを印刷します。

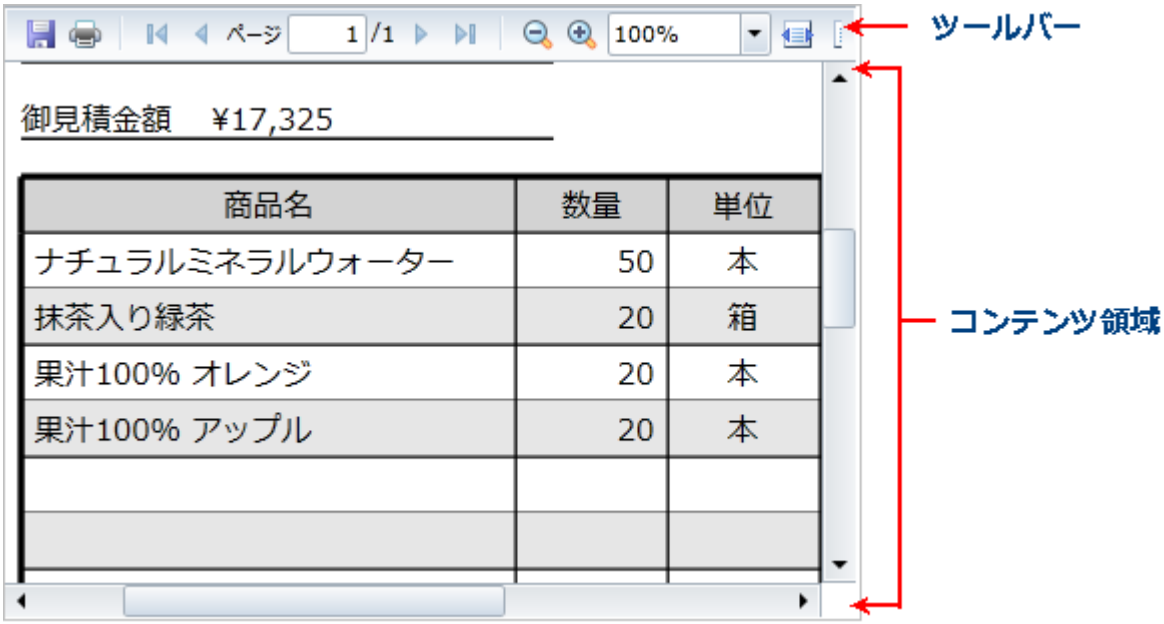
おめでとうございます。これで、**ReportViewer for WPF/Silverlight** クイックスタートは完了です。簡単な WPF/Silverlight アプリケーションを作成し、**ReportViewer for WPF/Silverlight** コントロールを追加してカスタマイズし、コントロールの実行時機能をいくつか確認することができました。

レイアウトおよび外観

以下のトピックでは、**C1ReportViewer** コントロールのレイアウトと外観をカスタマイズする方法について詳しく説明します。組み込みのレイアウトオプションを使用して、グリッドやキャンバスなどのコントロールをパネル内でレイアウトできます。テーマを使用することで、グリッドの外観をカスタマイズしたり、WPF/Silverlight の XAML ベースのスタイル設定を活用することができます。また、テンプレートを使用して、コントロールを書式設定およびレイアウトしたり、コントロールの操作をカスタマイズすることもできます。

ReportViewer の要素

C1ReportViewer コントロールは、ツールバーとコンテンツ領域の2つの部分で構成されます。次の図に、ツールバーとコンテンツ領域を示します。



C1ReportViewer コントロールにロードした HTML コンテンツまたは PDF は、コンテンツ領域に表示されます。ユーザーは、ツールバーを使用して、実行時にコンテンツを操作できます（コンテンツの印刷、ズームイン、ズームアウトなど）。コンテンツ領域とツールバーの詳細については、「ReportViewer のコンテンツ領域」と「ReportViewer ツールバー」のトピックを参照してください。

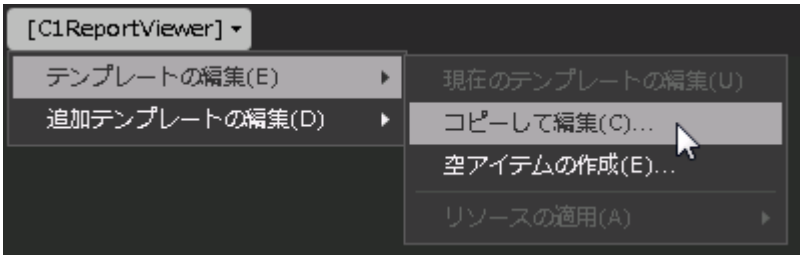
また、**ReportViewer for WPF/Silverlight** には、ツールバー要素だけで構成される **C1ReportViewerToolbar** コントロールも付属します。

テンプレート

WPF/Silverlight コントロールを使用する主な利点の1つは、これが自由にカスタマイズできるユーザーインターフェイスを持つ「外観のない」コントロールであることです。WPF/Silverlight アプリケーションのユーザーインターフェイスであるルックアンドフィールを独自に設計するのと同様に、**ReportViewer for WPF/Silverlight** で管理されるデータに関して独自の UI を提供できます。Extensible Application Markup Language（XAML。「ザムル」と発音する）は、コードを記述することなく独自の UI を設計するための簡単な方法を提供する XML ベースの宣言型言語です。

テンプレートへのアクセス

テンプレートにアクセスするには、Microsoft Expression Blend で、**C1ReportViewer** コントロールを選択し、メニューから**[テンプレートの編集]**を選択します。**[コピーして編集]**を選択して現在のテンプレートのコピーを作成して編集するか、**[空アイテムの作成]**を選択して新しい空のテンプレートを作成します。

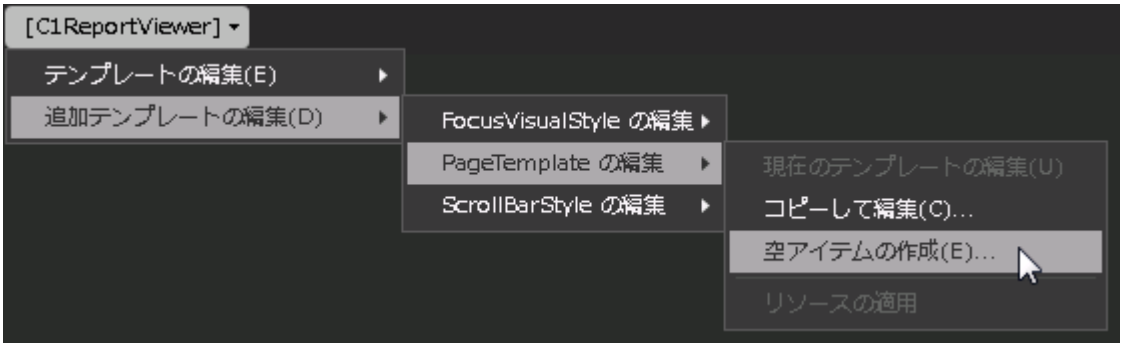


新しく作成されたテンプレートは、**[オブジェクトとタイムライン]**ウィンドウに表示されます。Template プロパティを使用してテンプレートをカスタマイズできます。

> メニューを使用して新しいテンプレートを作成する場合、テンプレートはそのテンプレートのプロパティに自動的にリンクされます。手作業でテンプレートの XAML を作成する場合は、作成したテンプレートに適切な Template プロパティをリンクする必要があります。

追加のテンプレート

デフォルトテンプレートのほかに、**C1ReportViewer** コントロールには追加のテンプレートがいくつかあります。これらの追加テンプレートには、Microsoft Expression Blend からもアクセスできます。Blend で **C1ReportViewer** コントロールを選択し、メニューから**[追加テンプレートの編集]**を選択します。テンプレートを選択し、**[空アイテムの作成]**を選択します。



スタイル

ReportViewer for WPF/Silverlight の **C1ReportViewer** コントロールは、コントロールの外観を変更するために使用できるスタイルのプロパティを提供します。これらのスタイルの一部について、次の表で説明します。

スタイル	説明
FocusVisualStyle	この要素がキーボードフォーカスを受け取ったときに適用される外観、効果などのスタイル特性をカスタマイズするためのプロパティを取得または設定します。これは依存プロパティです。
FontStyle	フォントスタイルを取得または設定します。これは依存プロパティです。
PageTemplate	ページの表示に使用される DataTemplate を取得または設定します。
Style	この要素のレンダリング時に使用されるスタイルを取得または設定します。これは依存プロパティです。
ToolbarStyle	この **C1ReportViewer** コントロールのツールバーに適用されるスタイルを取得または設定します。

表示状態

Microsoft Expression Blend で、カスタム状態や状態グループを追加して、ユーザーコントロールの状態ごとに異なる外観を定義できます。たとえば、マウスが置かれたときのコントロールの表示状態を変更できます。新しいテンプレートを作成すると、表示状態を確認、編集できます。これで、そのパーツで利用可能な表示状態が**[状態]**ウィンドウに表示されます。

よく使用される状態としては、項目の通常の外観を示す **Normal**、マウスが置かれている項目を示す **MouseOver**、有効でない項目を示す **Disabled** などがあります。フォーカスの状態には、項目にフォーカスがないときの **Unfocused**、項目にフォーカスがあるときの **Focused** などがあります。

実行時の操作

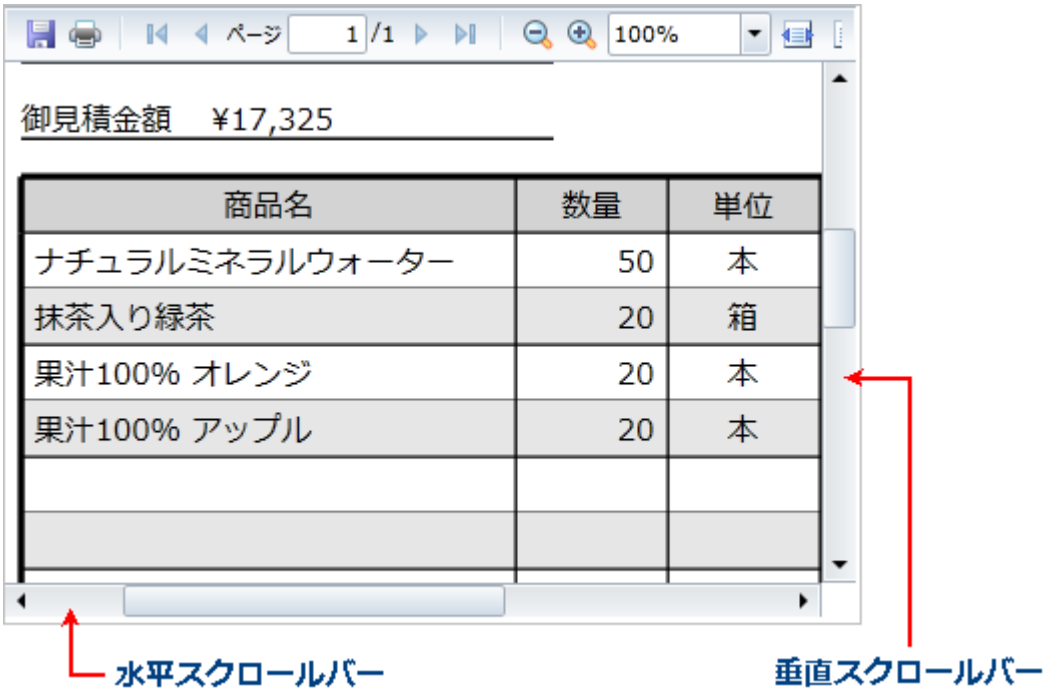
実行時に、ユーザーは、**C1ReportViewer** コントロールのツールバーとコンテンツ領域にある項目を操作することができます。コンテンツ領域のコンテンツを移動およびドラッグしたり、ツールバーを使用して、コンテンツ領域に表示されたドキュメントを操作することができます。

ReportViewer のコンテンツ領域

実行時に、ユーザーは、**C1ReportViewer** コントロールに含まれるコンテンツをスクロール、選択、およびコピーして、コンテンツ領域のコンテンツを操作することができます。

コンテンツのスクロール

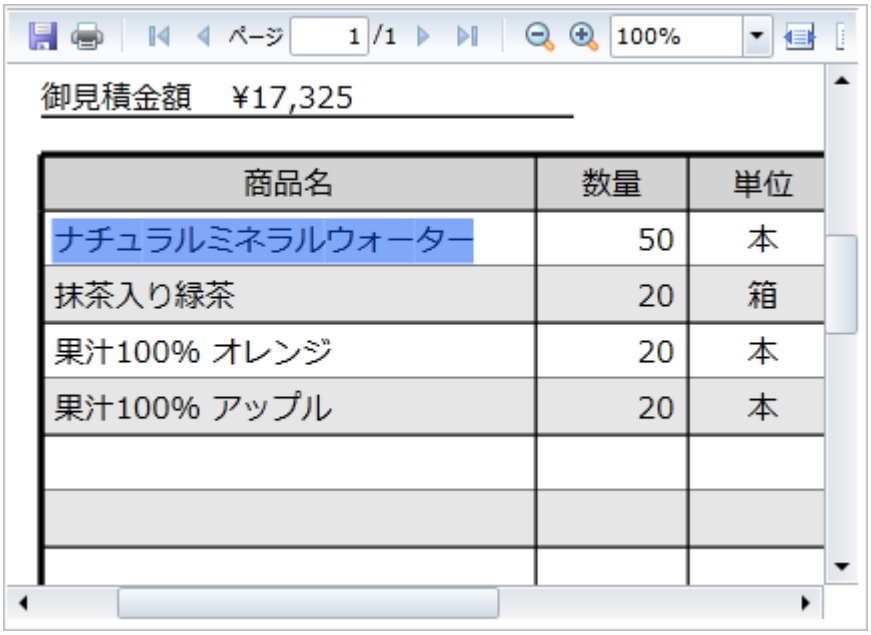
コントロールのコンテンツの幅または高さがコントロールのコンテンツ領域の表示領域より大きい場合は、ドキュメントのさまざまな場所を表示できるようにスクロールバーが表示されます。



スクロールバーのサムボタンを移動したり、矢印ボタン、キーボードの方向キー、またはマウスのスクロールホイールを使用して、コンテンツ領域をスクロールすることができます。

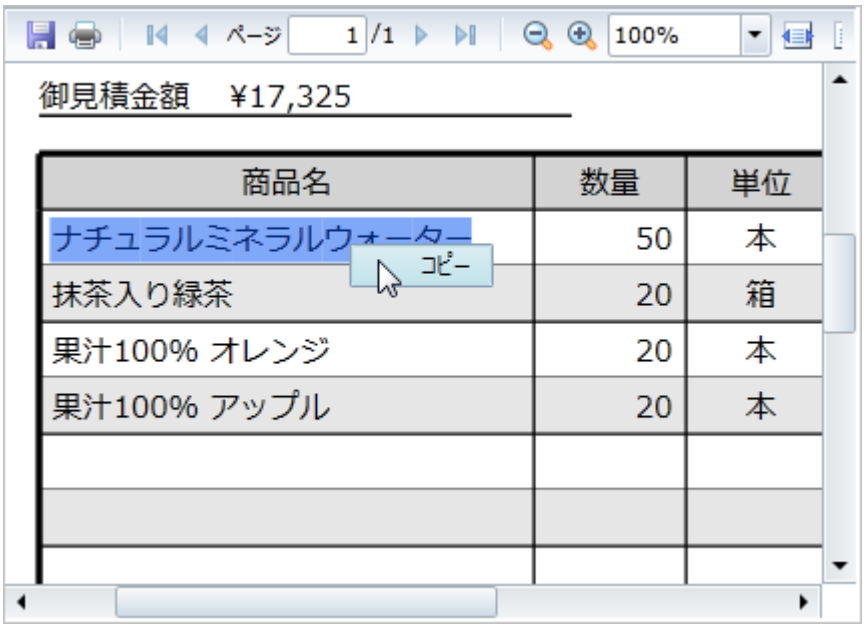
コンテンツの選択

選択対象のコンテンツ内をクリックしてマウスポインタをドラッグすることで、コンテンツを選択することができます。選択されたコンテンツは強調表示されます。たとえば、下の画像では語句「ナチュラルミネラルウォーター」が選択されています。



コンテンツのコピー

C1ReportViewer コントロールは、コンテンツをコピーするためのコンテキストメニューを備えています。まず、コピーするコンテンツを選択し、ドキュメントを右クリックします。コンテキストメニューが表示されます。コンテキストメニューの**[コピー]**を選択して、コンテンツをコピーできます。

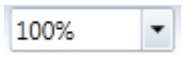


キーボードの **Ctrl** + **C** キーで、選択したコンテンツをコピーすることもできます。

ReportViewer ツールバー

実行時に、ユーザーは、ツールバーを使用して、コンテンツ領域に表示されるドキュメントを操作することができます。デフォルトでは、次の画像のようなツールバーが表示されます。

デフォルトでは、ツールバーの一部の項目は非アクティブまたは非表示になります。たとえば、ドキュメントの最初のページでは、**[前のページ]**ボタンがアクティブになりません。ツールバーには次のオプションがあります。

画像	名前	説明
	保存	現在のドキュメントをローカルファイルシステムに保存します。
	印刷	現在のドキュメントを印刷します。
	最初のページ	ドキュメントの最初のページに移動します。
	前のページ	ドキュメントの前のページに移動します。
	ページ	テキストボックスに入力されたページに移動します。
	次のページ	ドキュメントの次のページに移動します。
	最後のページ	ドキュメントの最後のページに移動します。
	ズームアウト	ドキュメントからズームアウトします。
	ズームイン	ドキュメントにズームインします。
	ズーム	選択された値までズームします。
	幅を合わせる	ドキュメントの幅をビューポートのサイズに合わせます。
	1ページ	ページ全体が表示されるように、ドキュメントのサイズをビューポートのサイズに合わせます。
	2ページ	2ページを並べて表示します。
	検索	ボックスに入力されたテキストをドキュメント内で検索します。
	前を検索	検索テキストの前の出現位置に移動します。
	次を検索	検索テキストの次の出現位置に移動します。

ツールバーの各操作をプログラムから実行することで、組み込みのツールバーを独自のカスタムツールバーに簡単に置き換えることもできます。カスタムツールバーを作成する場合は、**ToolbarVisibility** プロパティを使用して、組み込みのツールバーを非表示にすることができます。

タスク別ヘルプ

次のタスク別ヘルプトピックは、ユーザーの皆様が Visual Studio および Expression Blend に精通しており、**C1ReportViewer** コントロールの一般的な使用方法を理解していることを前提としています。**ReportViewer for WPF/Silverlight** 製品に精通していない場合は、まず「クイックスタート」を参照してください。

このセクションの各トピックは、**ReportViewer for WPF/Silverlight** 製品を使用して特定のタスクを実行するためのソリューションを提供します。また、タスク別ヘルプトピックのほとんどは、新しい WPF/Silverlight アプリケーションが作成され、**C1ReportViewer** コントロールがプロジェクトに追加されていることを前提としています（コントロールの作成については、「C1ReportViewer を追加する」を参照）。

C1ReportViewer を追加する

C1ReportViewer コントロールをアプリケーションに追加するには、次の手順に従います。

1. Visual Studio の**[ファイル]**メニューから、**[新規作成]**を選択し、**[プロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、左側のメニューから言語を選択し、**[フレームワーク]**ドロップダウンリストで**[.NET Framework 4]**を選択し、プロジェクトの名前を入力します。
3. ソリューションエクスプローラで、プロジェクト名を右クリックし、**[参照の追加]**を選択します。**[参照の追加]**ダイアログボックスで、以下のアセンブリを見つけて選択し、**[OK]**をクリックしてプロジェクトに参照を追加します。

WPF

C1.WPF

C1.WPF.ReportViewer

C1.WPF.RichTextBox

C1.WPF.Zip

Silverlight

C1.Silverlight

C1.Silverlight.ReportViewer

C1.Silverlight.RichTextBox

C1.Silverlight.Zip

4. MainWindow.xaml ファイルの XAML ビューを開き、マークアップ xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml" を使用して、Window タグに XAML 名前空間を追加します。
名前空間は次のようになります。

WPF

```
<Window x:Class="MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
```

```
    Title="MainWindow" Height="350" Width="525">
```

Silverlight

```
<UserControl x:Class="QuickStart.MainPage"
xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
xmlns:d=http://schemas.microsoft.com/expression/blend/2008
xmlns:mc=http://schemas.openxmlformats.org/markup-compatibility/2006
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml" mc:Ignorable="d"
d:DesignHeight="300" d:DesignWidth="400">
```

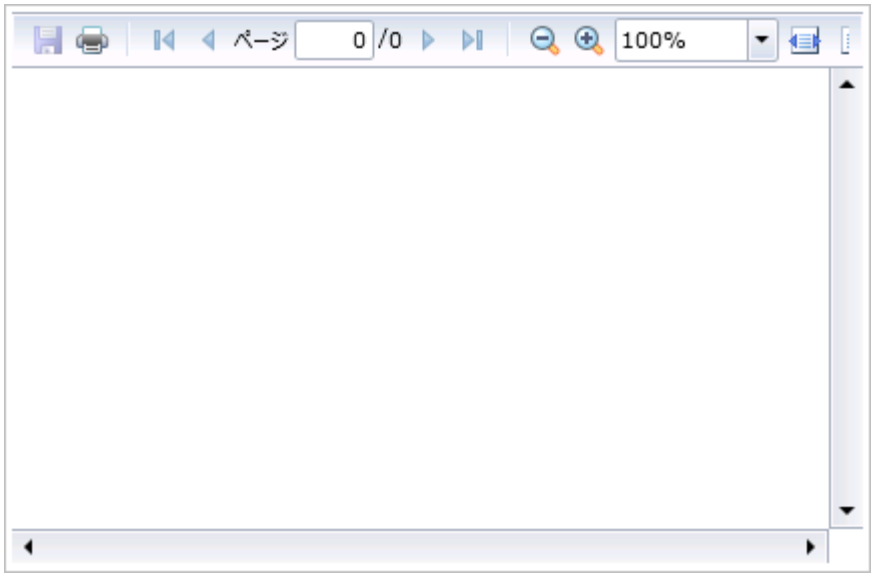
これは、複数の名前空間を追加しなくても、ほとんどの ComponentOne WPF/Silverlight コントロールを使用できるようにするための統合名前空間です。

5. ページの Grid タグ内に <c1:C1ReportViewer x:Name="C1ReportViewer1" /> タグを追加して、アプリケーションに **C1ReportViewer** コントロールを追加します。
XAML は次のようになります。

XAML

```
<Grid x:Name="LayoutRoot" Background="White">
    <c1:C1ReportViewer x:Name="C1ReportViewer1" />
</Grid>
```

これで、"C1ReportViewer1" という名前の **C1ReportViewer** コントロールがアプリケーションに追加されます。アプリケーションを実行すると、次の画像のように表示されます。



これで、アプリケーションのユーザーインターフェイスが正しくセットアップされましたが、このアプリケーションを実行すると、**C1ReportViewer** コントロールにコンテンツがないことがわかります。コンテンツの読み込みオプションについては、「ドキュメントを読み込む」のトピックを参照してください。

> **C1ReportViewer** コントロールが Visual Studio のツールボックスにインストールされている場合は、ページにコントロールをドラッグするだけで、上のすべての手順が自動的に実行されます。

C1ReportViewer へのドキュメントの読み込み

アプリケーションに **C1ReportViewer** コントロールを追加してからアプリケーションを実行すると、ページに空の **C1ReportViewer** が開きます。次の手順は、**LoadDocument** メソッドを呼び出して、コントロールにコンテンツを追加することです。**LoadDocument** メソッドでは、**Stream** オブジェクト（通常は PDF、HTML、または MHTML ドキュメントを含む）または **string**（PDF、HTML、または MHTML ドキュメントのファイル名）からコンテンツを読み込むことができます。

アプリケーションリソースからドキュメントを簡単に読み込むことができます。たとえば、次の手順に従います。

1. ソリューションエクスプローラで、プロジェクト名を右クリックし、[追加]→[既存の項目]を選択します。
2. [既存の項目の追加]ダイアログボックスで、PDF ファイルを見つけます。必要に応じて、ファイルの種類ドロップダウンボックスで[すべてのファイル]を選択して、PDF ファイルを表示します。別の PDF ファイルを選択して使用してもかまいません。
3. ソリューションエクスプローラで、アプリケーションに追加した PDF ファイルをクリックします。この例では、このファイルの名前を resource.pdf とします。[プロパティ]ウィンドウで、BuildAction プロパティを Resource に設定し、[出力ディレクトリにコピー]項目が[コピーしない]に設定されていることを確認します。
4. ページを右クリックしてコードビューに切り替え、[コードの表示]を選択します。次の手順では、XAML マークアップをアプリケーションに追加することで、ドロップダウンボックスにコンテンツを追加します。
5. ページの先頭に次の imports 文を追加します。

WPF

VisualBasic

```
Imports C1.WPF.ReportViewer
```

C#

```
using C1.WPF.ReportViewer;
```

Silverlight

VisualBasic

```
Imports C1.Silverlight.ReportViewer
```

C#

```
using C1.Silverlight.ReportViewer;
```

6. メインクラスに次のコードを追加します。

VisualBasic

```
Public Sub New()
    InitializeComponent()
    Dim resource = Application.GetResourceStream(New
Uri("AppName;component/resource.pdf", UriKind.Relative))
Me.C1ReportViewer1.LoadDocument(resource.Stream)
End Sub
```

C#

```
public MainPage()
{
    InitializeComponent();
    var uri = new Uri("/AppName;component/resource.pdf", UriKind.Relative);
    var resource = Application.GetResourceStream(uri);
    this.C1ReportViewer1.LoadDocument(resource.Stream);
}
```

このコードは、ストリームを追加し、それを **C1ReportViewer** コントロールに読み込みます。アプリケーションに別の名前を付けた場合は、"AppName" をプロジェクトの名前に置き換える必要があります。別の PDF ファイルを追加した場合は、"resource.pdf" を実際のファイル名に置き換えてください。

アプリケーションリソースからドキュメントを読み込む

アプリケーションリソースからドキュメントを簡単に読み込むことができます。たとえば、次の手順に従います。

1. ソリューションエクスプローラで、プロジェクト名を右クリックし、**[追加]**→**[既存の項目]**を選択します。
2. **[既存の項目の追加]**ダイアログボックスで、PDF ファイルを見つけます。必要に応じて、ファイルの種類ドロップダウンボックスで**[すべてのファイル]**を選択して、PDF ファイルを表示します。別の PDF ファイルを選択して使用してもかまいません。
3. ソリューションエクスプローラで、アプリケーションに追加した PDF ファイルをクリックします。この例では、このファイルの名前を **resource.pdf** とします。[プロパティ]ウィンドウで、**[ビルド アクション]** プロパティを **Resource** に設定し、**[出力ディレクトリにコピー]**項目が**[コピーしない]**に設定されていることを確認します。
4. ページを右クリックしてコードビューに切り替え、**[コードの表示]**を選択します。次の手順では、XAML マークアップをアプリケーションに追加することで、ドロップダウンボックスにコンテンツを追加します。
5. ページの先頭に、次の import 文を追加します。

VisualBasic

```
Imports C1.Silverlight.ReportViewer
```

C#

```
using C1.Silverlight.ReportViewer;
```

6. メインクラスに次のコードを追加します。

VisualBasic

```
Public Sub New()
    InitializeComponent()
    Dim resource = Application.GetResourceStream(New Ur
   ("AppName;component/resource.pdf", UriKind.Relative))
    Me.C1ReportViewer1.LoadDocument(resource.Stream)
End Sub
```

C#

```
public MainPage()
{
    InitializeComponent();
    var uri = new Uri("/AppName;component/resource.pdf", UriKind.Relative);
    var resource = Application.GetResourceStream(uri);
    this.C1ReportViewer1.LoadDocument(resource.Stream);
}
```

このコードは、ストリームを追加し、それを **C1ReportViewer** コントロールに読み込みます。アプリケーションに別の名前を付けた場合は、"AppName" をプロジェクトの名前に置き換える必要があります。別の PDF ファイルを追加した場合は、"resource.pdf" を実際のファイル名に置き換えてください。

メモ:このトピックの内容は、ComponentOne for Silverlight にのみ適用されます。

クライアントマシン内のファイルからドキュメントを読み込む

この例では、ユーザーがローカルファイルシステムからファイルを読み込むことができるようにアプリケーションを設定します。アプリケーションにボタンを追加し、実行時にファイルを選択して開くためのコードを追加します。この例は **OpenFileDialog** コントロールを使用するため、このコードを **Button_Click** イベントで実行する必要があります。このトピックは、"C1ReportViewer1" という名前の **C1ReportViewer** コントロールがアプリケーションに追加されていることを前提としています。

次の手順に従います。

1. アプリケーションの MainPage.xaml ファイルを開き、XAML ビューを開きます。
2. 次のマークアップを追加して、ボタンコントロールをアプリケーションに追加します。

XAML

```
<Button Content="ファイルを読み込む" Height="23" Name="Button1"
Click="Button1_Click" />
```

3. ページを右クリックし、**[コードの表示]**を選択します。コードビューで、前の手順で追加したボタンを初期化するためのコードを追加します。
4. ページの先頭に、次の import 文を追加します。

VisualBasic

```
Imports C1.Silverlight.ReportViewer
```

C#

```
using C1.Silverlight.ReportViewer;
```

5. 次の **Button_Click** イベントハンドラコードを追加します。

VisualBasic

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Button1.Click
    Dim dialog = New OpenFileDialog()
    dialog.Filter = "PDF ファイル|*.pdf|HTML ファイル|*.htm;*.html"
    If dialog.ShowDialog() = True Then
        Using fileStream = dialog.File.OpenRead()
            Try
                C1ReportViewer1.LoadDocument(fileStream)
            Catch ex As Exception
                MessageBox.Show("ドキュメントの読み込みに失敗しました。")
            End Try
        End Using
    End If
End Sub
```

C#

```
private void Button1_Click(System.Object sender, System.Windows.RoutedEventArgs
e)
{
   var dialog = new OpenFileDialog();
   dialog.Filter = "PDF ファイル|*.pdf|HTML ファイル|*.htm;*.html";
   if (dialog.ShowDialog() == true) {
         using (var fileStream = dialog.File.OpenRead()) {
               try {
                     C1ReportViewer1.LoadDocument(fileStream);
               } catch (Exception ex) {
                     MessageBox.Show("ドキュメントの読み込みに失敗しました。");
               }
         }
   }
}
```

このコードは、ボタンがクリックされたときに表示されるダイアログボックスを初期化します。このダイアログボックスで、ユーザーは C1ReportViewer コントロールで開くファイルを選択できます。上のコードでは、**LoadDocument** メソッドで PDF コンテンツと HTML コンテンツをどのように読み込むことができるかに注目してください。

6. アプリケーションを実行します。
7. 実行中のアプリケーションで、**[ファイルの読み込み]**ボタンをクリックします。ダイアログボックスが表示され、任意の PDF ファイルまたは HTML ファイルを選択できます。
8. ローカルマシンにある目的の PDF ファイルを検索して選択し、**[開く]**ボタンをクリックします。ダイアログボックスが閉じ、選択したファイルが C1ReportViewer コントロールに読み込まれます。

メモ:このトピックの内容は、ComponentOne for Silverlight にのみ適用されます。

サーバー内のファイルからドキュメントを読み込む

C1ReportViewer コントロールを使用するシナリオとしてよくあるのは、スケジュールに基づいてレポートサーバー（**C1Report** や Microsoft SQL Server Reporting Services など）でレポートを生成し、生成したレポートをサーバーのファイルシステムに配置する場合です。次に、Silverlight アプリケーションで、サーバーからこれらのファイルを取得し、ほとんどオーバーヘッドなくユーザーに表示できます。

C1ReportViewer をアプリケーションに追加した後で、Silverlight 対応 WCF サービスをサーバープロジェクトに追加する必要があります。このサービスは、存在するレポートのリストと、各レポートの実際のドキュメントストリームを Silverlight クライアントに提供します。

たとえば、次のコードは、レポートプロバイダ Web サービスの典型的な実装です。

VisualBasic

```
<ServiceContract([Namespace] := "")> _
<AspNetCompatibilityRequirements(RequirementsMode :=
AspNetCompatibilityRequirementsMode.Allowed)> _
Public Class ReportingService
      <OperationContract> _
      Public Function GetReportList() As String()
      Dim path__1 = Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory,
"Resources")
      Return Directory.GetFiles(path__1, "*.pdf")
      End Function
      <OperationContract>
Public Function GetReportStream(reportName As String) As Byte()
     ' ファイル名を取得します
      Dim path__1 = Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory,
"Resources")
      reportName = Path.Combine(path__1, reportName)
     ' ファイルをストリームに読み込みます
      Dim ms = New MemoryStream()
      Dim buff = New Byte(63999) {}
      Using sr = New FileStream(reportName, FileMode.Open)
            While True
                  Dim read As Integer = sr.Read(buff, 0, buff.Length)
                  ms.Write(buff, 0, read)
                  If read = 0 Then
                        Exit While
                  End If
            End While
      End Using
End Function
' バイトストリームを返します
Return ms.ToArray()
End Class
```

C#

```
[ServiceContract(Namespace = "")]
[AspNetCompatibilityRequirements(RequirementsMode =
        AspNetCompatibilityRequirementsMode.Allowed)]
```

```
public class ReportingService
{
  [OperationContract]
  public string[] GetReportList()
  {
    var path = Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory,
"Resources");
    return Directory.GetFiles(path, "*.pdf");
  }
  [OperationContract]
  public byte[] GetReportStream(string reportName)
  {
    // ファイル名を取得します
      var path = Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory,
"Resources");
      reportName = Path.Combine(path, reportName);
      // ファイルをストリームに読み込みます
      var ms = new MemoryStream();
      var buff =new byte[64000];
      using (var sr = new FileStream(reportName, FileMode.Open))
      {
        for (; ; )
        {
          int read = sr.Read(buff, 0, buff.Length);
          ms.Write(buff, 0, read);
          if (read == 0) break;
        }
      }
    }
    // バイトストリームを返します
    return ms.ToArray();
  }
}
```

このように、コードはたいへん標準的です。最初のメソッドはサーバーにあるレポートをリストし、Silverlight アプリケーションはユーザーにレポートのリストを表示します。2番目のメソッドは、選択されたレポートを表すバイトストリームを返します。

アプリケーションのクライアント部分は、次のようにサービスを使用します。

VisualBasic

```
Public Sub New()
      InitializeComponent()
     ' 存在するレポートのリストを取得します
      Dim svc = New ReportingServiceReference.ReportingServiceClient()
      AddHandler svc.GetReportListCompleted, AddressOf svc_GetReportListCompleted
      svc.GetReportListAsync()
End Sub
' サーバーにあるレポートのリストを ComboBox に設定します
Private Sub svc_GetReportListCompleted(sender As Object, e As
ReportingServiceReference.GetReportListCompletedEventArgs)
      _cmbReport.Items.Clear()
```

```
      For Each file As String In e.Result
            _cmbReport.Items.Add(Path.GetFileNameWithoutExtension(file))
      Next
      _cmbReport.IsEnabled = True
      _cmbReport.SelectedIndex = 0
End Sub
' 選択されたレポートを表示します
Private Sub ReportType_Click(sender As Object, e As EventArgs)
     ' レポート名を作成します
      Dim reportName As String = DirectCast(_cmbReport.SelectedItem, String)
      reportName += If(_btnPDF.IsChecked.Value, ".pdf", ".mhtml")
     ' ストリームを取得します
      Dim svc = New ReportingServiceReference.ReportingServiceClient()
      AddHandler svc.GetReportStreamCompleted, AddressOf svc_GetReportStreamCompleted
      svc.GetReportStreamAsync(reportName)
End Sub
' レポートを表示します
Private Sub svc_GetReportStreamCompleted(sender As Object, e As
ReportingServiceReference.GetReportStreamCompletedEventArgs)
      Dim ms = New MemoryStream(e.Result)
      _reportViewer.LoadDocument(ms)
End Sub
```

C#

```
public MainPage()
{
  InitializeComponent();
  // 存在するレポートのリストを取得します
  var svc = new ReportingServiceReference.ReportingServiceClient();
  svc.GetReportListCompleted += svc_GetReportListCompleted;
  svc.GetReportListAsync();
}
// サーバーにあるレポートのリストを ComboBox に設定します
void svc_GetReportListCompleted(object sender,
  ReportingServiceReference.GetReportListCompletedEventArgs e)
{
  _cmbReport.Items.Clear();
  foreach (string file in e.Result)
  {
    _cmbReport.Items.Add(Path.GetFileNameWithoutExtension(file));
  }
  _cmbReport.IsEnabled = true;
  _cmbReport.SelectedIndex = 0;
}
// 選択されたレポートを表示します
void ReportType_Click(object sender, EventArgs e)
{
  // レポート名を作成します
  string reportName = (string)_cmbReport.SelectedItem;
    reportName += _btnPDF.IsChecked.Value ? ".pdf" : ".mhtml";
```

```
    // ストリームを取得します
    var svc = new ReportingServiceReference.ReportingServiceClient();
    svc.GetReportStreamCompleted += svc_GetReportStreamCompleted;
    svc.GetReportStreamAsync(reportName);
  }
  // レポートを表示します
  void svc_GetReportStreamCompleted(object sender,
    ReportingServiceReference.GetReportStreamCompletedEventArgs e)
  {
    var ms = new MemoryStream(e.Result);
    _reportViewer.LoadDocument(ms);
  }
```

メモ:このトピックの内容は、ComponentOne for Silverlight にのみ適用されます。

レポートを動的に作成して読み込む

動的なレポートの作成もかなり一般的なシナリオです。この場合は、レポートサーバーをアプリケーションサーバーからアクセスできるように構成し、「サーバー内のファイルからドキュメントを読み込む」トピックで説明した Web サービスを使用して、(ファイルから読み込むのではなく)レポートサーバーからレポートストリームを直接取得します。

具体的な手順は、使用しているレポートサーバーによって異なります。たとえば、Microsoft SQL Server Reporting Services サーバーから PDF レポートを動的に取得するには、「サーバー内のファイルからドキュメントを読み込む」トピックに示されている Web サービスを変更します。次に例を示します。

VisualBasic

```
<ServiceContract([Namespace] := "")> _
<AspNetCompatibilityRequirements(RequirementsMode :=
AspNetCompatibilityRequirementsMode.Allowed)> _
Public Class ReportsService
      <OperationContract> _
      Public Function GetReportStream(reportName As String) As Byte()
            Dim reportServer As String = "YOUR REPORT SERVER NAME"
            Dim url = String.Format("http://{0}/ReportServer?/{1}&rs:Format=PDF",
reportServer, reportName.Replace(" "C, "+"C))
            Dim wc = New System.Net.WebClient()
            wc.UseDefaultCredentials = True
            Dim stream = wc.OpenRead(url)
            Dim ms = New MemoryStream()
            Dim buf = New Byte(64 * 1024 - 1) {}
            While True
                  Dim read As Integer = stream.Read(buf, 0, buf.Length)
                  If read = 0 Then
                        Exit While
                  End If
                  ms.Write(buf, 0, read)
            End While
            ms.Flush()
            Return ms.ToArray()
      End Function
End Class
```

C#

```
[ServiceContract(Namespace = "")]
[AspNetCompatibilityRequirements(RequirementsMode =
     AspNetCompatibilityRequirementsMode.Allowed)]
public class ReportsService
{
  [OperationContract]
  public byte[] GetReportStream(string reportName)
  {
    string reportServer = "YOUR REPORT SERVER NAME";
    var url = string.Format("http://{0}/ReportServer?/{1}&rs:Format=PDF",
      reportServer,
      reportName.Replace(' ', '+'));
    var wc = new System.Net.WebClient();
    wc.UseDefaultCredentials = true;
    var stream = wc.OpenRead(url);
    var ms = new MemoryStream();
    var buf = new byte[64 * 1024];
    for (; ; )
    {
      int read = stream.Read(buf, 0, buf.Length);
      if (read == 0) break;
      ms.Write(buf, 0, read);
    }
    ms.Flush();
    return ms.ToArray();
  }
}
```

このように、違いはわずかです。この方法では、レポートサーバーでレポートキャッシュポリシーを指定することもできるため、パフォーマンスやスケーラビリティが損なわれることがありません。

メモ:このトピックの内容は、ComponentOne for Silverlight にのみ適用されます。

ツールバーを非表示にする

カスタマイズしたツールバーを作成する場合は(「ツールバーをカスタマイズする」トピックなどを参照)、**C1ReportViewer** コントロールのデフォルトのツールバーを非表示にしなければならないことがあります。組み込みのツールバーを非表示にするには、**ToolbarVisibility** プロパティを使用します。次に例を示します。

XAML

```
<c1:C1ReportViewer x:Name="C1ReportViewer1" ToolbarVisibility="Collapsed"/>
```

VisualBasic

```
Me.C1ReportViewer.ToolbarVisibility = Visibility.Collapsed.
```

C#

```
this.C1ReportViewer.ToolbarVisibility = Visibility.Collapsed;
```

ツールバーをカスタマイズする

C1ReportViewer のカスタムツールバーの作成はたいへん簡単です。デフォルトツールバーの各ボタンはコントロールのコマンドに対応しているため、XAML でのみカスタムツールバーを作成できます。**C1Toolbar** を使用して **C1ReportViewer** ツールバーを作成するサンプルコードを次に示します。

XAML

```
<Grid x:Name="LayoutRoot">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition />
    </Grid.RowDefinitions>
    <c1:C1ToolbarStrip>
        <c1:C1ToolbarButton
            Content="最初"
            Command="{Binding FirstPageCommand,ElementName=reportViewer}" />
        <c1:C1ToolbarButton
            Content="前へ"
            Command="{Binding PreviousPageCommand,ElementName=reportViewer}" />
        <ContentPresenter
            Content="{Binding PageNumber,ElementName=reportViewer}" />
        <TextBlock Text="/"/>
        <ContentPresenter
            Content="{Binding PageCount,ElementName=reportViewer}" />
        <c1:C1ToolbarButton
            Content="次へ"
            Command="{Binding NextPageCommand,ElementName=reportViewer}" />
        <c1:C1ToolbarButton
            Content="最終"
            Command="{Binding LastPageCommand,ElementName=reportViewer}" />
        <ComboBox
            SelectedItem="{Binding Zoom,ElementName=reportViewer,Mode=TwoWay}">
            <sys:Double>0.5</sys:Double>
            <sys:Double>1</sys:Double>
            <sys:Double>1.5</sys:Double>
        </ComboBox>
    </c1:C1ToolbarStrip>
    <c1:C1ReportViewer
        x:Name="reportViewer"
        Grid.Row="1"
        ToolbarVisibility="Collapsed"/>
</Grid>
```

各ボタンで **Command** プロパティをどのように **C1ReportViewer** のコマンドに連結しているかに注目してください。また、**PageNumber** プロパティと **PageCount** プロパティに連結して、簡単に現在のページと総ページ数を表示できます。最後

に、**ComboBox** を **Zoom** プロパティに連結すると、ユーザーがズーム倍率を制御できます。

さまざまなコマンドを使用して、その他のボタンをカスタマイズできます。以下は、コマンドのリストです。

コマンド	説明
SaveCommand	ドキュメントを保存します。
PrintCommand	ドキュメントを印刷します。
FirstPageCommand	ドキュメントの最初のページに移動します。
PreviousPageCommand	ドキュメントの前のページに移動します。
NextPageCommand	ドキュメントの次のページに移動します。
LastPageCommand	ドキュメントの最後のページに移動します。
DecreaseZoomCommand	ドキュメントからズームアウトします。
IncreaseZoomCommand	ドキュメントにズームインします。
FindPreviousCommand	検索テキストの前の出現位置を検索します。
FindNextCommand	検索テキストの次の出現位置を検索します。

C1ReportViewerToolbar のテンプレートでは、次の名前の **ToggleButton** も使用されます。

オプション	説明
FitWidth	ドキュメントの幅をコントロールのサイズに合わせます。
OnePage	1ページを表示します。
TwoPages	2ページを並べて表示します。

カスタムツールバーのシナリオで、これらのトグルボタンの1つ、たとえば TwoPages を使用するには、**C1ReportViewer** テンプレートに TwoPages トグルボタンを配置する必要があります。コントロールの外部に独自のツールバーを作成する場合は、1つのボタンを追加し、クリックハンドラで **C1ReportViewer.ViewMode = ViewMode.TwoPages** を設定します。