

# FlexReport for WPF

2018.04.11 更新

グレースィティ株式会社

## 目次

<a href="#">FlexReport for WPF の概要</a>	3
<a href="#">主な特長</a>	4
<a href="#">FlexReportでC1Reportのサポート</a>	5
<a href="#">C1Reportレポート定義からC1FlexReportレポート定義への変換</a>	5-7
<a href="#">C1Report から FlexReport への重要な変更点</a>	7-8
<a href="#">FlexReport の詳細情報</a>	9
<a href="#">コンポーネントとコントロール</a>	9
<a href="#">オブジェクトモデルの概要</a>	9-11
<a href="#">FlexReport のセクション</a>	11
<a href="#">クイックスタート</a>	12-13
<a href="#">FlexReport の操作</a>	14
<a href="#">C1FlexReport と C1Document</a>	14
<a href="#">FlexReport のデータ連結</a>	14
<a href="#">データベースからのデータの取得</a>	14-15
<a href="#">ストアドプロシージャからのデータの取得</a>	15-16
<a href="#">データテーブルオブジェクトをデータソースとして使用</a>	16-17
<a href="#">カスタムデータソースオブジェクトの使用</a>	17-21
<a href="#">FlexReport のデータソース</a>	21-22
<a href="#">コードを使用した複数のデータソースへの接続</a>	22
<a href="#">複数データソースレポートでのパラメータへのデータの連結</a>	22-23
<a href="#">計算フィールドの定義</a>	23
<a href="#">デスクトップ用 FlexReport の開発</a>	23
<a href="#">実行時の FlexReport のロード</a>	23-26
<a href="#">パラメータの追加</a>	26-27
<a href="#">データのグループ化</a>	27-28
<a href="#">データのソート</a>	28-29
<a href="#">データのフィルタ処理</a>	29-30
<a href="#">レポートのエクスポート</a>	30-31
<a href="#">PdfFilterクラスを使用したFlexReportのレンダリング</a>	31
<a href="#">HTMLフィルタクラスを使用したFlexReportのレンダリング</a>	31-32
<a href="#">イメージ用さまざまなエクスポートフィルタを使用したFlexReportのレンダリング</a>	32

<a href="#">RtfFilterクラスを使用したFlexReportのエクスポート</a>	32-33
<a href="#">ExportFilterクラスを使用したレポートのエクスポート</a>	33-34
<a href="#">FlexReportの印刷</a>	34-35
<a href="#">VBScript の操作</a>	35-37
<a href="#">FlexReportセクションの分割</a>	37
<a href="#">フィールドの変更</a>	37-38
<a href="#">サブセクションの追加</a>	38-39
<a href="#">FlexReportDesigner について</a>	40
<a href="#">FlexViewer for WPF</a>	41
<a href="#">主な特長</a>	41-42
<a href="#">クイックスタート</a>	42-44
<a href="#">FlexViewerツールバー</a>	44-45
<a href="#">FlexReport と FlexViewer の連結</a>	45-46
<a href="#">FlexViewer の機能</a>	46-47
<a href="#">レポートプレビューのズーム</a>	47-48
<a href="#">レポートのビューの回転</a>	48-49
<a href="#">FlexViewerを使用したレポートのエクスポート</a>	49-50
<a href="#">FlexViewerを使用したレポートの印刷</a>	50-51
<a href="#">FlexViewerPane</a>	52
<a href="#">FlexReport と FlexViewerPane の連結</a>	52-53
<a href="#">レポートプレビューのズーム</a>	53-54
<a href="#">レポートのビューの回転</a>	54-55

## FlexReport for WPF の概要

**ComponentOne** には、**FlexReport for WPF** が導入されています。これは、複雑なレポートの作成からプレビュー、エクスポート、印刷まで、レポートソリューションを提供する軽量かつ包括的なレポートツールです。これを使用して、データを集約した形式で提示し、魅力的で機能豊富なレポートを柔軟に生成し、レポートを設計およびカスタマイズすることもできます。優れたオブジェクトモデル、プレビューペイン、高品質なレンダリング、さらに使いやすさを備えた FlexReport は、基礎レベルのレポート設計者だけでなく、上級レベルのレポート設計者にとっても必須のコントロールです。FlexReportコントロールを操作するには、最小限必要なサーバー構成はWindows 7 SP1またはWindows Server 2008 R2 SP1用プラットフォーム更新 (KB2670838) です。

## 主な特長

FlexReport for WPFの主な特長は次のとおりです。

- **軽量、高速**  
FlexReportは、最新の軽量かつ高速なレポートツールです。
- **レンダリングの向上**  
FlexReportは、DirectWrite/DirectXのような最新のレンダリングを使用することで、高パフォーマンスのレポートコンテンツを描画および生成します。これにより、テキスト、図形、境界線のレンダリングが向上し、精度と品質が高まります。
- **プレビューペイン**  
FlexReportは、FlexReportおよびSSRSレポートを読み込んでプレビューするためのプレビューコントロール FlexViewerPaneを提供します。
- **エクスポート機能**  
FlexReportを使用して、レポートやドキュメントをPDF、RTF、XLSX、TIFF、BMP、PNG、JPEG、およびGIF形式などのさまざまな形式にエクスポートできます。ExportFilterクラスを使用してレポートをエクスポートできます。
- **印刷**  
FlexReportでは、レポートをPrintメソッドを使用して直接印刷できます。

### FlexReport の制限

- **まだサポートされていない機能**  
FlexReport for WPFでは、チャートフィールド、カスタムフィールド (MapおよびSuperLabel) はサポートされません。

## FlexReportでC1Reportのサポート

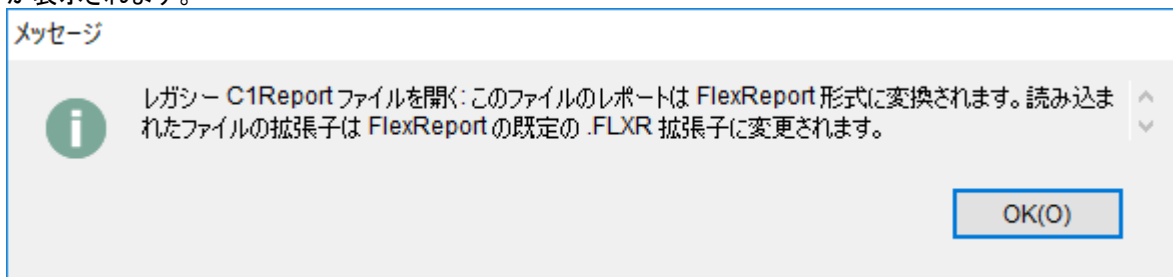
**FlexReport** では、C1Report をサポートするために、既存の C1Report レポート定義を **C1FlexReport** レポート定義に変換できます。FlexReportDesigner アプリケーションを使用するか、C1Report XML ファイルを WPF アプリケーションの C1FlexReport オブジェクトにロードして、C1Report レポート定義を **C1FlexReport** レポート定義に変換することができます。

## C1Reportレポート定義からC1FlexReportレポート定義への変換

C1Report を使用して作成されたレポートは、**C1FlexReport** と完全に互換性があります。以下に、既存の C1Report レポート定義(.xml)を C1FlexReport レポート定義(.flxr)に簡単にアップグレードまたは変換する 2 つの方法を示します。

### デザイナーで C1Report レポート定義を変換する

1. **C1FlexReportDesigner.4.exe**を実行します。
2. **[ファイル]**→**[開く]**に移動し、アップグレードする C1Report レポート定義(.xml)を選択します。次のダイアログボックスが表示されます。



3. **[OK]**をクリックします。
4. **[ファイル]**→**[保存]**に移動します。
5. **[レポート定義ファイルの保存]**ダイアログボックスで、**[ファイル名]**を指定し、**[保存]**をクリックします。

レポート定義がタイプ **.flxr** として保存されます。これで、レポート定義が C1Report の .xml から C1FlexReport の .flxr に変換されました。この例では、**FlexReportDesigner**アプリケーションで**TelephoneBillReport.xml**を開き、**TelephoneBillReport.flxr**として保存しました。

### Visual Studio で既存の C1Report WPF アプリケーションプロジェクトを FlexReport WPF アプリケーションプロジェクトにアップグレードする

1. FlexReportDesignerアプリケーションでC1Reportレポート定義(.xml)を開き、「**デザイナーで C1Report レポート定義を変換する**」の手順を使用して、.flxrとして保存します。
2. C1Report(.xml)ファイルを含む既存の WPF アプリケーションプロジェクトを開きます。
3. アプリケーションで、MainWindow.xamlを開きます。
4. 参照からC1.WPF.C1Report、C1.WPF.ReportViewerのエントリを削除します。
5. XAMLビューからC1DocumentViewerのエントリを削除します。この例では、C1DocumentViewerの次のエントリを削除しました。

XAML

```
<my:C1DocumentViewer Grid.Row="1" Name="c1DocumentViewer1" xmlns:my="clr-namespace:C1.WPF.C1Report;assembly=C1.WPF.C1Report" />
```

6. **C1FlexViewerPane**コントロールを<Grid> </Grid>タグ間にドラッグアンドドロップし、次のコードを**C1FlexViewerPane**のエントリに追加します。

XAML

```
<my:C1FlexViewerPane Grid.Row="1" Name="ViewerPane" xmlns:my="clr-namespace:C1.WPF.FlexViewer;assembly=C1.WPF.FlexViewer.4" />
```

7. アプリケーションに、C1.WPF.FlexReport.4.dllへの参照を追加します。

8. コードビューに切り替え、C1.C1Reportへの参照を削除します。
9. 次の名前空間を追加します。

Visual Basic

```
Imports C1.WPF.FlexReport
```

C#

```
using C1.WPF.FlexReport;
```

10. C1Reportコンポーネントの名前をC1FlexReportに変更します。

Visual Basic

```
Dim clr As New C1Report()
' 変更後
Dim rep As New C1FlexReport()
```

C#

```
C1Report clr = new C1Report();
// 変更後
C1FlexReport rep = new C1FlexReport();
```

11. 次のコードを置き換えます。

Visual Basic

```
clr.Load("../¥..¥TelephoneBillReport.xml", "TelephoneBillReport")
' 変更後
rep.Load("../¥..¥TelephoneBillReport.flxr", "TelephoneBillReport")
```

C#

```
clr.Load(@"../¥..¥TelephoneBillReport.xml", "TelephoneBillReport");
// 変更後
rep.Load(@"../¥..¥TelephoneBillReport.flxr", "TelephoneBillReport");
```

12. コード内の、C1ReportのすべてのインスタンスをC1FlexReportに置き換えます。
13. FlexViewerPaneでFlexReportをロードするには、次のコードを変更します。

Visual Basic

```
C1DocumentViewer1.Document = clr.C1Document.FixedDocumentSequence
' 変更後
ViewerPane.DocumentSource = rep
```

C#

```
C1DocumentViewer1.Document = clr.C1Document.FixedDocumentSequence;
// 変更後
ViewerPane.DocumentSource = rep;
```

14. アプリケーションを実行します。

## C1Report から FlexReport への重要な変更点

FlexReport のコードはゼロから作成されています。したがって、C1Report から FlexReport への API の移行には、次のように重要な変更点が含まれています。

- C1Report では、**DataSource.RecordSource** 内でしかパラメータを直接指定できません (SQL 文の直前のキーワード **PARAMETERS** の後)。

FlexReport では、レポートレベルの専用コレクション **C1FlexReport.Parameters** が別途用意されています。このコレクションで、レポートパラメータを指定できます。xml ファイルから C1Report レポート定義をインポートすると、**PARAMETERS** キーワードを使用して従来の方法で指定されたパラメータは、**C1FlexReport.Parameters** コレクションに自動的に追加されます。

- FlexReport では、データソースが開かれた後に **OnOpen** スクリプトが実行されます。したがって、そのスクリプトでメインデータソースに行われた変更は、レポートに影響しません。レポートを生成する前にデータソースの一部を変更するには、**GlobalScripts** を使用します。GlobalScripts には、関数定義、プロシージャ定義、およびこれらの定義内にはないコードを含めることができます。これらの定義とコードは、レポートのレンダリングが開始されてデータソースが開かれる前に、すべて実行されるようになりました。
- FlexReport で **C1Report.OutlineRootLevel** プロパティは削除されています。アウトライン構造を制御するには、**OutlineLabel** プロパティと **OutlineParent** プロパティを使用します。サブレポートによって生成されたアウトラインをオフにするには、**SubreportField.OutlinesVisible** プロパティを使用します。
- C1Report には、テキストを生成/レイアウトするためのメソッドが 2 つあり (デフォルトおよび「gdi+」)、それらは少し異なっています (**C1Report.UseGdiPlusTextRendering** を true に設定した場合は非デフォルト)。これらのメソッドは、多少異なるテキストレイアウトを行う場合があります (改行が異なる場所に配置されるなど)。FlexReport は常に、**UseGdiPlusTextRendering** が設定された C1Report と同様にテキストを生成/レイアウトします。ただし、それでも、**UseGdiPlusTextRendering** が設定された C1Report と FlexReport の間では改行に違いが発生する場合があります。
- **AddOutlineEntry** イベントは削除されています。フィールド/セクション/サブセクションによって生成されるアウトラインエントリのテキストを変更するには、**OutlineLabel** プロパティを使用します。
- **C1FlexReport.ReportError** イベントのために特殊な新しいイベントタイプ (**ReportErrorEventArgs** イベント引数を受け取る **ReportErrorHandler**) が追加されました。ReportEventArgs タイプが変更されました。イベント引数から Exception と Handled が削除されました。
- C1Report では、エラーが発生した場合、EndReport イベントが呼び出されません。FlexReport では、レンダリング中に致命的エラーが発生した場合でも、**EndReport** イベントが呼び出されます。
- 次の C1Report メソッド/プロパティは、C1FlexReport から削除されました。
  - C1Report.EmfType: C1FlexReport では、代わりに EMF+ を使用します。
- C1FlexReport では、コードビハインドで C1Report Render<X> メソッドにアクセスできません。これらのメソッドは内部的に使用されます。
- **C1FlexReport.Document** は System.Drawing.Printing.PrintDocument に変換できません。PrintDocument の C1Report.Document プロパティは存在しないため、FlexReport で PrintDocument を使用してはなりません。
- FlexReport では、**IC1FlexReportRecordset** に ApplyFilter() メソッドと ApplySort() メソッドがありません。代わりに、FlexReport の DataSource にあるフィルタ定義とソート定義を使用してください。IC1FlexReportRecordset を **DataSource.Recordset** に割り当てても、DataSource でフィルタ/ソートを定義できます。
- C1Report は、単一の「Field」クラスを使用して、レポート内のさまざまなコンテンツを表します。一方、C1FlexReport



は、TextField、BarCodeField、RtfField、SubreportField、RtfField、CheckBoxField、ImageField、ShapeField、ChartFieldなどの異なるクラスを使用します。C1Reportと同じFieldクラスも含まれますが、これが廃止され、下位互換性のためにのみ使用されます。

- FlexReport では、AddScriptObject イベントの代わりに、**GetScriptObject** イベントがあります。したがって、次のコードは

```
private void c1flxr_StartReport(object sender, System.EventArgs e)
{
    c1flxr.AddScriptObject("LookUp", new LookUpObject());
}
```

次のように変更します。

```
c1flxr.GetScriptObject += c1flxr_GetScriptObject;
...

void c1flxr_GetScriptObject
(object sender, C1.WPF.FlexReport.ReportGetScriptObjectEventArgs e)
{
    if (e.Name.ToLower() == "lookup")
        e.Object = new LookUpObject();
}
```

## FlexReport の詳細情報

**FlexReport** コントロールの使用や操作を開始する前に、FlexReport に付属しているコンポーネントとコントロール、FlexReport のオブジェクトモデル、FlexReport で使用できるセクションについて理解しておく必要があります。以下のセクションで、これらの詳細について説明します。

## コンポーネントとコントロール

**FlexReport for WPF** は、以下のアセンブリで構成されています。

### C1.WPF.FlexReport dll

これには、**C1FlexReport** クラスおよび関連するタイプが含まれています。また、すべてのレポートレンダリング機能とドキュメント生成機能も提供します。

### C1.WPF.FlexViewer dll

以下のコントロールまたはコンポーネントを通して、すべての表示機能を提供します。

- **C1FlexViewerPane**  
**C1FlexViewerPane** コントロールは、プレビュー中のドキュメントのページを表示します。

### 含まれるアプリケーション

レポート作成用のコントロールに加えて、FlexReportには以下のスタンドアロンアプリケーションも含まれます。

- **C1FlexReportDesigner**  
**C1FlexReportDesigner**は、C1FlexReport レポート定義ファイル(.FLXR)の作成および編集に使用するためのWinForms ベースデザイナーアプリケーションです。レガシーC1Report レポート定義(.XML)を新しい .FLXR 形式に変換することもできます。

**C1FlexReportDesigner** では、WPFバージョン、UWP バージョンと WinForms バージョンと互換性があるレポートを設計できます。**C1FlexReportDesigner.4.exe**は、「Any CPU」ターゲット用にビルドされ、x64システムでは64ビットモードで実行され、またx86ターゲット用に構築された**C1FlexReportDesigner32.4.exe**は、常に32ビットモードで実行されます。これらのアプリケーションは、次のパスにあります。

**C:\Program Files (x86)\ComponentOne\Apps\v4**

## オブジェクトモデルの概要

**FlexReport** は優れたオブジェクトモデルを備えています。これらのオブジェクト、コレクション、関連するプロパティやメソッドなどにより、FlexReport を柔軟かつ容易に生成することが可能です。次の表に、オブジェクトおよびそのプロパティとメソッドを一覧します。

<b>C1FlexReport</b>
プロパティ: ReportName, Document, DoEvents, Page, MaxPages, Font, OnOpen, OnClose, OnNoData, OnPage, OnError メソッド: Load, GetReportList, Save, Clear, Render, Evaluate, Execute
<b>BarCodeField</b>
プロパティ: BarCode, BarCodeOptions, Font, Text
<b>CalculatedField</b>
プロパティ: DataSource, Expression, Type
<b>CheckBoxField</b>

<b>プロパティ:</b> CheckAlign, CheckMark, Text, ThreeState, Value
<b>DataField</b>
<b>プロパティ:</b> Calculated, Name, Type, Value
<b>DataSource</b>
<b>プロパティ:</b> CalculatedFields, ConnectionString, DataProvider, Filter, RecordSet, RecordSource, SortDefinitions <b>メソッド:</b> AssignFrom, Open, Close, ToString
<b>FieldBase</b>
<b>プロパティ:</b> Anchor, Height, ForcePageBreak, MarginBottom, MarginLeft, MarginRight, MarginTop, Section, SplitHorzBehavior, SplitVertBehavior <b>メソッド:</b> AssignFrom, Clone
<b>FieldCollection</b>
<b>メソッド:</b> Add, Remove, RemoveAt
<b>Group</b>
<b>プロパティ:</b> GroupBy, KeepTogether, ParentReport, SectionHeader, SectionFooter, Sort, SortExpression <b>メソッド:</b> AssignFrom
<b>GroupCollection</b>
<b>プロパティ:</b> Report <b>メソッド:</b> Add, Clear, RemoveAt
<b>ImageField</b>
<b>プロパティ:</b> Picture, PictureAlign, PictureScale
<b>ReportParameter</b>
<b>プロパティ:</b> AllowedValuesDefinition, DisplayText, ParentReport <b>メソッド:</b> AssignFrom, CreateSame
<b>ReportParameterCollection</b>
<b>プロパティ:</b> Item, Count
<b>Section</b>
<b>プロパティ:</b> Calculated, Fields, Height, SplitBehavior, SubSections <b>メソッド:</b> AssignFrom
<b>SectionCollection</b>
<b>プロパティ:</b> Detail, Footer, Header, PageFooter, PageHeader <b>メソッド:</b> FindSection
<b>ShapeField</b>
<b>プロパティ:</b> Line, Shape, ShapeBackColor, ShapeBackground, ShapeType <b>メソッド:</b> AssignFrom
<b>SortDefinition</b>
<b>プロパティ:</b> Direction, Expression <b>メソッド:</b> AssignFrom
<b>SubSection</b>

# FlexReport for WPF

プロパティ: AutoHeight, Calculated, Fields, ForcePageBreak, Height, ParentReport, ParentSection, SplitBehavior, Visible  
メソッド: AssignFrom

SubSectionCollection

メソッド: Add, Remove, RemoveAt

SubreportField

プロパティ: ParameterValues, Subreport, SubreportFilter

TextField

プロパティ: Format, Text, Value


## FlexReport のセクション

すべてのレポートは、次の 5 つの基本セクションで構成されます。

セクション	説明
詳細	詳細セクションには、ソースレコードセット内の各レコードごとに一度ずつレンダリングされるフィールドが含まれます。
ヘッダー	レポートのヘッダーセクションは、レポートの最初にレンダリングされます。
フッター	レポートのフッターセクションは、レポートの最後にレンダリングされます。
ページヘッダー	ページヘッダーセクションは、各ページの上部にレンダリングされます(オプションで、レポートヘッダーを含むページを除外することもできます)。
ページフッター	ページフッターセクションは、各ページの下部にレンダリングされます。

これら 5 つのセクションのほかに、1 つのグループを表す 2 つの追加セクションとして**グループヘッダー**と**グループフッター**も追加できます。たとえば、3 つのグループレベルを持つレポートには、11 のセクションがあることとなります。すなわち、レポートヘッダー、レポートフッター、ページヘッダー、ページフッター、詳細セクション、3 つのグループヘッダー、3 つのグループフッターとなります。

各セクションは、実際のレポートの内容が表示されるサブセクションで構成されます。セクションには常に少なくとも 1 つのサブセクションが含まれます。機能を強化するには、追加のサブセクションを追加することができます。たとえば、サブセクションの可視性は、条件によってはスクリプトで切り替えることができます。

 これらのセクションを非表示にすることはできますが、グループを追加または削除しない限り、セクションを追加または削除することはできません。

すべてのセクションの詳細については、「[FlexReport のセクション](#)」を参照してください。

## クイックスタート

このクイックスタートガイドでは、**FlexReport for WPF**のいくつかの機能について詳しく説明します。このセクションでは、レポート定義を作成し、レポートをFlexViewerコントロールロードし、レポートをレンダリングする方法を示します。

**FlexReportDesigner**アプリケーションまたはコードを使用してレポート定義を作成します。レポート定義を作成する最も簡単な方法は、FlexReport に付属しているスタンドアロンの**C1FlexReportDesigner**デスクトップアプリケーションを使用することです。

**C1FlexReportDesigner.4.exe** (64ビットシステムの場合、64ビットモード、また32ビットシステムの場合、32ビットモードで実行します)と**C1FlexReportDesigner32.4.exe** (常に32ビットモードで実行します)はお使いのコンピュータ上の「**C:\Program Files (x86)\ComponentOne\Apps\v4**」フォルダに置かれています。

**FlexReport for WPF**を使用して簡単なWPFアプリケーションを作成するには、次の手順に従います。

1. **手順 1: レポート定義の作成**
2. **手順 2: レポートのロード**
3. **手順 3: レポートのレンダリング**

### 手順 1: レポート定義の作成

FlexReport ウィザードを使用して、**FlexReportDesigner**アプリケーションで新しいレポート定義を作成します。レポート定義を作成するには、次の手順を実行します。

1. 32ビット版のレポートデザイナーアプリ**C1FlexReportDesigner32.4.exe**を実行します(ここでは、通常は32ビットのMS Jet OLE DBドライバーを使用できますので、32ビット版を実行しています)。ファイルメニューから新規を選択します。
2. デザイナーの左端にある「レポート」タブの「新規レポート」ドロップダウンをクリックし、[レポートウィザード]を選択します。
3. **データプロバイダ**ドロップダウンメニューから**OLEDBデータプロバイダ**を選択し、**接続文字列**のテキストボックスの横にある**省略符(...)**ボタンをクリックして、**プロバイダ**タブ内の**Microsoft Jet 4.0 OLE DB**プロバイダと**接続**タブ内の**C1Nwind.mdb**データベースを選択します。

 **C1Nwind.mdb**データベースは、**ドキュメント\ComponentOne Samples\Common**フォルダに見つけることができます。

4. 「**データソース**」タブからテーブルを選択し、「**次へ**」をクリックします。この例では、「**Products**」テーブルを選択しています。
5. レポートをデータソースに接続した後、レポートのフィールド、レイアウト、およびスタイルを選択します。レポートに適切なタイトルを付けて、「**完了**」をクリックします。

### 先頭に戻る

### 手順 2: レポートのロード

ファイルからレポート定義をロードするには、次の手順を実行します。

1. Visual Studioで新しい**WPFアプリケーション**を作成します。
2. **XAML** デザインに **Button** コントロールと **C1FlexViewer** コントロールを追加します。**C1FlexViewer**コントロールの名前を**Viewer**に設定します。
3. 「**手順 1: レポート定義の作成**」でデザイナーを使用してプロジェクトに作成したレポート定義ファイルを追加します。ここでは、というProducts Report という名前のレポートを使用しています。
4. アプリケーションに**C1.WPF.FlexReport.4.dll**への参照を追加します。
5. コードに次の名前空間を追加します。
  - C1.WPF.FlexReport
6. コードビューでは、**Button\_Click**イベント内に次のコードを追加して、レポートをロードします。
  - **Visual Basic**

```
Dim rep As New C1FlexReport()
'reポート定義をロードします
```

# FlexReport for WPF

```
rep.Load(@"..\..\Products Report.flxr", "Products Report")
    ○ C#
C1FlexReport rep = new C1FlexReport();
//レポート定義をロードします
rep.Load(@"..\..\Products Report.flxr", "Products Report");
```

[先頭に戻る](#)

## 手順 3: レポートのレンダリング

レポート定義を作成して**C1FlexReport** にロードしたら、レポートを**C1FlexViewer** コントロールにレンダリングできます。レポートをレンダリングするには、コードビューで**Button\_Click** イベントに次のコードを追加します。

- **Visual Basic**

'レポートをプレビューします

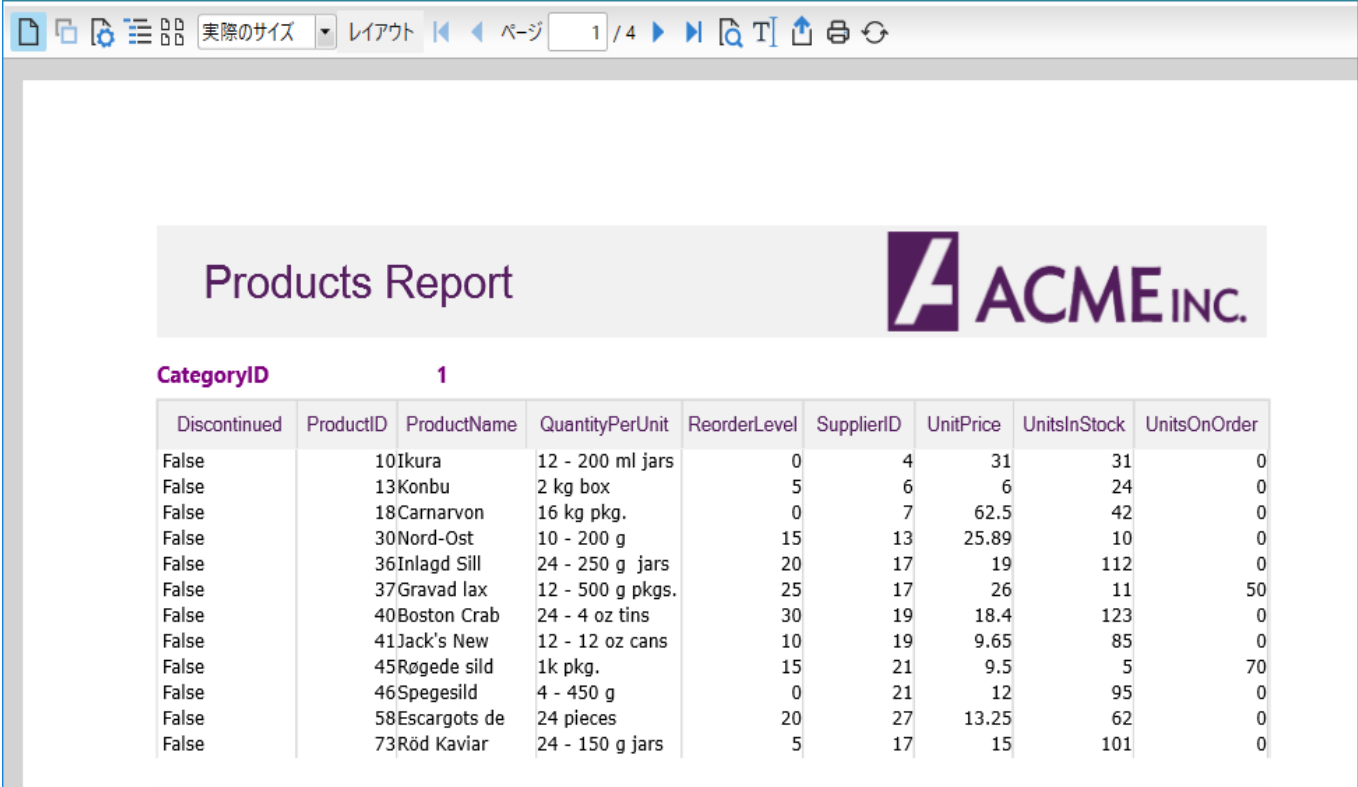
```
Viewer.DocumentSource = rep
```

- **C#**

//レポートをプレビューします

```
Viewer.DocumentSource = rep;
```

FlexViewer にレンダリングされて表示されたレポートを次に示します。



CategoryID	1							
Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	10	Ikura	12 - 200 ml jars	0	4	31	31	0
False	13	Konbu	2 kg box	5	6	6	24	0
False	18	Carnarvon	16 kg pkg.	0	7	62.5	42	0
False	30	Nord-Ost	10 - 200 g	15	13	25.89	10	0
False	36	Inlagd Sill	24 - 250 g jars	20	17	19	112	0
False	37	Gravad lax	12 - 500 g pkgs.	25	17	26	11	50
False	40	Boston Crab	24 - 4 oz tins	30	19	18.4	123	0
False	41	Jack's New	12 - 12 oz cans	10	19	9.65	85	0
False	45	Røgede sild	1k pkg.	15	21	9.5	5	70
False	46	Spegesild	4 - 450 g	0	21	12	95	0
False	58	Escargots de	24 pieces	20	27	13.25	62	0
False	73	Röd Kaviar	24 - 150 g jars	5	17	15	101	0

[先頭に戻る](#)

## FlexReport の操作

FlexReport は、さまざまなシナリオで使用できますが、ほとんどのシナリオに関連する主な手順は次のとおりです。

1. **レポート定義の作成:** **C1FlexReportDesigner** デスクトップアプリケーションを使用して実行できます。このデザイナーで .flxr レポート定義ファイルを作成したら、これをアプリケーションのランタイムから使用できるようにして、**C1FlexReport** にロードされるようにする必要があります。または、**C1FlexReport** のリッチオブジェクトモデルを使用して、レポート定義をコードで完全に実行時に作成する方法もあります。
2. **レポートへのデータの提供:** レポート定義は、WPF バージョンで使用できるデータソースを念頭に作成する必要があります。実行時にレポート定義を **C1FlexReport** コンポーネントにロードしたら、データをアクセス可能にして、レポートを生成できるようにする必要があります。レポートは完全にクライアントで生成されます。使用できるデータソースの詳細については、「[FlexReport のデータソース](#)」を参照してください。
3. **レポートのレンダリング:** レポートは、プリンタ、**FlexViewerPane** に直接レンダリングすることも、別の形式にエクスポートすることもできます。

## C1FlexReport と C1Document

**C1FlexReport** には、**C1.WPF.Document** アセンブリが必要です。これは、2 つの重要なクラス **C1Document** と **C1DocumentSource** を公開し、レポートのレンダリングとプレビューに必要なインフラストラクチャを提供します。特に、**C1DocumentSource** は **C1FlexReport** の派生元の抽象クラスで、**FlexViewerPane** コントロールで公開される **DocumentSource** プロパティのタイプです。これにより、このコントロールは、**C1DocumentSource** から派生された任意のドキュメントタイプを表示することができます。

**C1DocumentSource** から派生されたタイプとして **C1SSRSDocumentSource** もあります (同じく **C1.WPF.Document** アセンブリから提供)。これを使用して SSRS レポートを表示できます。

## FlexReport のデータ連結

FlexReport でレポートを作成するには、レポート定義のほかに実際のデータが必要です。通常は、データベースからデータを取得しますが、他の場所から取得することもできます。以下のトピックでは、SQLite や他のソースからデータを取得する方法を確認します。

## データベースからのデータの取得

FlexReport でレポートデータを取得またはロードするには、**C1FlexReport** の以下の **DataSource** プロパティを設定する必要があります。

- **ConnectionString**
- **RecordSource**

データソースを設定するには、次のコードを使用します。

Visual Basic

```
Dim rep As New C1FlexReport ()
' データソースを初期化します
Dim ds As DataSource = rep.DataSource
ds.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; DataSource=C:\...\ComponentOne Samples\Common\C1NWind.mdb;"
ds.RecordSource = "Employees"
```

C#

# FlexReport for WPF

```
C1FlexReport rep = new C1FlexReport();  
//データソースを初期化します  
DataSource ds = rep.DataSource;  
ds.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:  
¥...¥ComponentOne Samples¥Common¥C1NWind.mdb;";  
ds.RecordSource = "Employees";
```

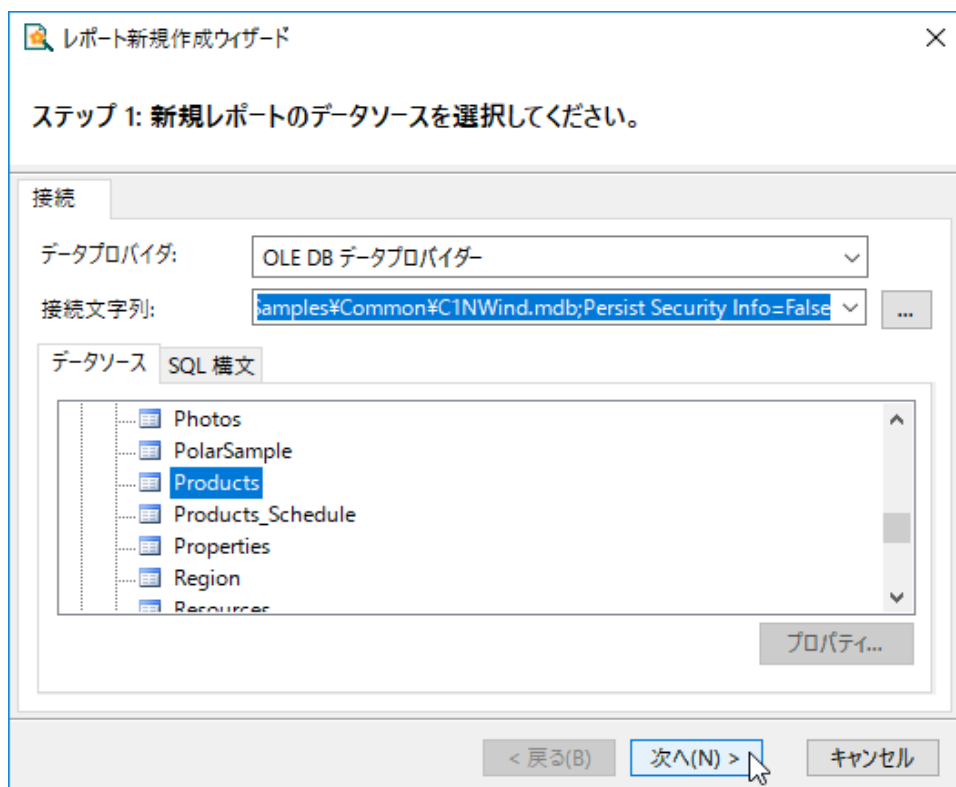
これらのプロパティを設定すると、C1FlexReport はデータソースを初期化し、これを使用してデータベースから自動的にデータをロードします。

## ストアプロシージャからのデータの取得

ストアプロシージャは、複数のアプリケーションにまたがるロジックの一貫した実装に役立つほか、パフォーマンスを向上させ、ユーザーがデータベース内のテーブルの詳細を知る必要をなくします。ストアプロシージャの大きな利点の 1 つは、データベースがレコードセットをフィルタ処理するためのパラメータを渡すことができることです。これにより、返されるデータセットが少量になるので、レポートをより速くより簡単に操作できます。

**C1FlexReport ウィザード**では、ストアプロシージャからレポートにデータを挿入できます。**C1FlexReport ウィザード**を開くには、**C1FlexReportDesigner**アプリケーションの[レポート]タブから[新しいレポート]ボタンをクリックします。

ストアプロシージャによるレポートへのデータの挿入は、SQL 文や直接テーブルを使用する方法と違いがありません。**C1FlexReport ウィザード**の最初の画面で、**省略符**ボタンをクリックしてデータソースを選択します。次に、有効なデータソースのリストから[ストアプロシージャ]を選択します。



[次へ]を選択し、ウィザードを続行します。

他の形式のデータのロードと同様に、次の 2 つのオプションがあります。

- DataSource の **ConnectionString** プロパティと **RecordSource** プロパティを使用してデータソースを選択できます。デザイナーで、(省略符[...]ボタンをクリックして)**DataSource** ダイアログボックスを使用して接続文字列を選択し、使用するテーブルまたは sproc をリストから選択します。次に例を示します。

```
connectionstring = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:¥Users
```



```
¥Windows 8.1¥Documents¥ComponentOne Samples¥Common¥NWind_ja.mdb;Persist
Security Info=False;" RecordSource = "[Products by Category]"
```

(この例では、ストアプロシージャ名にスペースが含まれているため、それが角かっこで囲まれています。)

- 任意の方法でデータソースを作成し、それを DataSource の **Recordset** プロパティに割り当てることができます。  
 使用する方法では、コードを記述する必要がありますが、どこかにデータがキャッシュされており、それを使用して複数のレポートを作成する場合に役立ちます。これは、上記の方法より優先されます  
 (**ConnectionString**、**RecordSource**、**Recordset** をすべて指定した場合、**C1FlexReport** は **Recordset** を使用します)。  
 使用する接続/アダプタのタイプ (OleDb、SQL、Oracle など) によって構文は異なります。簡単に正しい構文を使用するには、Visual Studio のサーバーエクスプローラーからフォームにテーブルまたは sproc をドラッグします。これで、すべての必要な要素が正しく追加されるので、コードを調べて重要な部分を確認できます。  
 ストアドプロシージャは、その名前を指定することでデータソースとして使用できます。sproc にパラメータがある場合は、名前をパラメータとして渡します。たとえば、MSSQL と ADVENTURE\_WORKS.mdf データベースに基づいて構築されたレポート定義において、**C1FlexReportDesigner** で指定された SQL 要求は次のとおりです (ADVENTUREWORKS\_DATA.MDF のパスは適宜調整が必要)。

```
PARAMETERS Employee Int 290;
DECLARE @RC int
DECLARE @EmployeeID int
set @EmployeeID = [Employee]
EXECUTE @RC = [C:¥ADVENTUREWORKS_DATA.MDF].[dbo].[uspGetEmployeeManagers]
@EmployeeID
```

## データテーブルオブジェクトをデータソースとして使用

**C1FlexReport** の外部にあるデータを操作し、それを DataTable オブジェクトにロードしなければならないアプリケーションはよくあります。このような場合は、DataTable オブジェクトをレポートデータソースとして使用して、レポートのレンダリング時にデータを再ロードする無駄を省くことができます。

この方法は、次のようなアプリケーションでも役立ちます。

- セキュリティ制限により、接続文字列を非公開にする必要があり、データだけ (ソースではなく) を公開できる場合。
- データベースが OleDb (C1FlexReport で内部的に使用されるプロバイダ) によってサポートされていない場合。
- データをデータベース以外から取得する場合。代わりに、DataTable を作成し、カスタムコードを使用してデータを挿入します。

DataTable オブジェクトを **C1FlexReport** のデータソースとして使用するには、レポート定義をロードし、DataTable を **C1FlexReport.DataSource.Recordset** プロパティに割り当てただけです。次に例を示します。

### Visual Basic

```
' キャッシュまたはセキュリティで保護されたカスタムプロバイダから DataTable をロードします
Dim dt As DataTable = GetMyDataTable()

' レポート定義をロードします(データソースを設定する前)
rep.Load("reportFile", "reportName")

' DataTable をデータソースとして使用します
rep.DataSource.Recordset = dt

Private Function GetMyDataTable() As DataTable
Dim result As New DataTable()
result.Columns.Add("ID", GetType(Integer))
```

# FlexReport for WPF

```
result.Columns.Add("Caption", GetType(String))
result.Rows.Add(1, "Red")
result.Rows.Add(2, "Blue")
result.Rows.Add(3, "Green")
Return result
End Function
```

C#

```
// キャッシュまたはセキュリティで保護されたカスタムプロバイダから DataTable をロードします
DataTable dt = GetMyDataTable();

// レポート定義をロードします(データソースを設定する前)
rep.Load(@"reportFile", "reportName");

// DataTable をデータソースとして使用します
rep.DataSource.Recordset = dt;

private DataTable GetMyDataTable()
{
    DataTable result = new DataTable();
    result.Columns.Add("ID", typeof(int));
    result.Columns.Add("Caption", typeof(string));
    result.Rows.Add(1, "Red");
    result.Rows.Add(2, "Blue");
    result.Rows.Add(3, "Green");
    return result;
}
```

## カスタムデータソースオブジェクトの使用

カスタムオブジェクトをデータソースとして使用できます。必要な条件は、カスタムオブジェクトが **IC1FlexReportRecordset** インタフェースを実装することだけです。

**IC1FlexReportRecordset** は、事実上どのようなデータの集合にも簡単に追加できる簡潔で実装しやすいインタフェースです。これは、**DataTable** オブジェクトを作成し、それにすべてのデータをコピーする方法より効率的であることが多くあります。たとえば、カスタムデータソースオブジェクトを使用して、ファイルシステムやカスタム .xml ファイルまたはカスタム .flxr ファイルをラップできます。

カスタムデータソースオブジェクトを使用するには、レポート定義をロードし、そのオブジェクトを **DataSource** クラスに **C1FlexReport** の **Recordset** プロパティに割り当てます。次に例を示します。

Visual Basic

```
Public Class MyRecordset
    Implements IC1FlexReportRecordset
    Private _recNo As Integer
    Private _recCount As Integer

    Public Sub New()
        _recCount = 60
    End Sub

    Public Property RecCount() As Integer
```

```

Get
Return _recCount
End Get
Set
_recCount = value
End Set
End Property

' IC1FlexReportRecordsetの実装
Public Function GetFieldNames() As String()
Return New String() {
    "ID",
    "Caption"
}
End Function

Public Function GetFieldTypes() As Type()
Return New Type() {
    GetType(Integer), GetType(String)
}
End Function

Public Function GetFieldValue(fieldIndex As Integer) As Object
Select Case fieldIndex
Case 0
Return _recNo + 1
Case 1
Return String.Format("Caption {0}", _recNo + 1)
Case Else
Return Nothing
End Select
End Function

Public Function BOF() As Boolean
Return _recNo < 0
End Function

Public Function EOF() As Boolean
Return _recNo >= _recCount
End Function

Public Sub MoveFirst()
_recNo = 0
End Sub

Public Sub MoveLast()
_recNo = _recCount - 1
End Sub

Public Sub MovePrevious()
_recNo -= 1
End Sub

```

```
Public Sub MoveNext ()
_recNo += 1
End Sub

Public Function GetBookmark() As Integer
Return _recNo
End Function

Public Sub SetBookmark(bkmk As Integer)
_recNo = bkmk
End Sub

Public ReadOnly Property Count() As Integer
Get
Return RecCount
End Get
End Property
End Class

Private Function GetMyCustomDataSource() As IC1FlexReportRecordset
Return New MyRecordset() With {
    Key.RecCount = 3
}
End Function

Private Sub btnGet_Click(sender As Object, e As EventArgs)
' カスタムデータソースオブジェクトを取得します
Dim rs As IC1FlexReportRecordset = GetMyCustomDataSource()

' レポート定義をロードします(データソースを設定する前)
rep.Load("..¥..¥Products Report.flxr", "Products Report")

' C1FlexReport コンポーネントでカスタムデータソースオブジェクトを使用します
rep.DataSource.Recordset = rs

Dim pf As New PdfFilter()
pf.Preview = True
pf.FileName = "..¥..¥Products Report.pdf"
rep.RenderToFilter(pf)
End Sub
```

C#

```
public class MyRecordset: IC1FlexReportRecordset {
    private int _recNo;
    private int _recCount;

    public MyRecordset() {
        _recCount = 60;
    }

    public int RecCount {
```

```
    get {
        return _recCount;
    }
    set {
        _recCount = value;
    }
}

// IC1FlexReportRecordsetの実装
public string[] GetFieldNames() {
    return new string[] {
        "ID",
        "Caption"
    };
}

public Type[] GetFieldTypes() {
    return new Type[] {
        typeof(int), typeof(string)
    };
}

public object GetFieldValue(int fieldIndex) {
    switch (fieldIndex) {
        case 0:
            return _recNo + 1;
        case 1:
            return string.Format("Caption {0}", _recNo + 1);
        default:
            return null;
    }
}

public bool BOF() {
    return _recNo < 0;
}

public bool EOF() {
    return _recNo >= _recCount;
}

public void MoveFirst() {
    _recNo = 0;
}

public void MoveLast() {
    _recNo = _recCount - 1;
}

public void MovePrevious() {
    --_recNo;
}
```

```
public void MoveNext() {
    ++_recNo;
}

public int GetBookmark() {
    return _recNo;
}

public void SetBookmark(int bkmk) {
    _recNo = bkmk;
}

public int Count {
    get {
        return RecCount;
    }
}

}

private IC1FlexReportRecordset GetMyCustomDataSource() {
    return new MyRecordset() {
        RecCount = 3
    };
}

private void btnGet_Click(object sender, EventArgs e)
{
    // カスタムデータソースオブジェクトを取得します
    IC1FlexReportRecordset rs = GetMyCustomDataSource();

    // レポート定義をロードします(データソースを設定する前)
    rep.Load(@"..\¥..\¥Products Report.flxr", "Products Report");

    // C1FlexReport コンポーネントでカスタムデータソースオブジェクトを使用します
    rep.DataSource.Recordset = rs;

    PdfFilter pf = new PdfFilter();
    pf.Preview = true;
    pf.FileName = @"..\¥..\¥Products Report.pdf";
    rep.RenderToFilter(pf);
}
```

## FlexReport のデータソース

FlexReport 定義には複数のデータソースを設定でき、これらのデータソースには **C1FlexReport.DataSources** コレクションからアクセスできます。このコレクション内のデータソースは、一意の名前で識別されます。これらのデータソースは次のように使用できます。

- **メインデータソース**:これはレポートのメインデータソースです。メインデータソースは、レポートの **C1FlexReport.DataSourceName** プロパティを使用して指定します。メインデータソースが指定されていない場合

(**DataSourceName** が空、または DataSources コレクションにない名前が指定されている)、C1FlexReport は非連結モードでレンダリングされ、詳細セクションが 1 つだけ含まれます。

- **パラメータ用のデータソース**:これは、レポートパラメータ(**C1FlexReport.Parameters** コレクション内の要素)として有効な値のソースです。パラメータ用のデータソースは、**ReportParameter.AllowedValuesDefinition.Binding.DataSourceName** プロパティを使用して指定します。

FlexReportでサポートされているデータソースタイプのリストは、次のとおりです。

- OLE DB
- ODBC
- XML
- Object in external assembly
- Microsoft SQL Server Compact Data Provider version 3.5 and 4.0

C1Report との下位互換性のため、**C1FlexReport** には DataSources[DataSourceName] を指す **DataSource** プロパティがあります。新しい C1FlexReport を作成すると、「Main」という名前の 1 つの要素が **C1FlexReport.DataSources** コレクションに追加され、「Main」が **C1FlexReport.DataSourceName** プロパティに割り当てられます。

C1Report では、メインデータソースがレポートの唯一のデータソースであることに注意してください。

## コードを使用した複数のデータソースへの接続

「FlexReport クイックスタート」では、メインデータソースに連結されるレポートの作成方法を学習しました。1 つのレポートが複数のデータソースを持つことができるため、チャートやパラメータを使用しながら、これらのデータソースへの接続方法について学習する必要があります。

以下のセクションでは、複数のデータソースを使用するレポートで、チャートやパラメータにデータを連結する方法について説明します。

## 複数データソースレポートでのパラメータへのデータの連結

パラメータにデータを連結することで、レポートパラメータ(**C1FlexReport.Parameters** コレクション内の要素)に対して有効な値が定義できます。**ReportParameter.AllowedValuesDefinition.Binding.DataSourceName** プロパティは、有効なパラメータ値のリストを作成するために使用されるデータソースを示します。次のコード例は、複数のデータソースを含むレポートのパラメータにデータを連結する方法を示します。

Visual Basic

・ データソースおよびこのデータソースを使用するパラメータを追加します

```
Dim mds As DataSource = rep.DataSource
Dim ds As New DataSource()
ds.Name = "CategoriesDS"
ds.ConnectionString = mds.ConnectionString
ds.RecordSource = "select * from categories"
ds.DataProvider = DataProvider.OLEDB
rep.DataSources.Add(ds)
mds.RecordSource = "select * from products where categoryid =
[CategoryParam]"
Dim rp As New ReportParameter()
rp.DataType = Doc.ParameterType.[Integer]
rp.Prompt = "Category"
rp.Name = "CategoryParam"
rp.AllowedValuesDefinition.Binding.DataSourceName = "CategoriesDS"
rp.AllowedValuesDefinition.Binding.ValueExpression = "CategoryID"
```

# FlexReport for WPF

```
rp.AllowedValuesDefinition.Binding.LabelExpression = "CategoryName"  
rep.Parameters.Add(rp)
```

C#

```
// データソースおよびこのデータソースを使用するパラメータを追加します  
DataSource mds = rep.DataSource;  
DataSource ds = new DataSource();  
ds.Name = "CategoriesDS";  
ds.ConnectionString = mds.ConnectionString;  
ds.RecordSource = "select * from categories";  
ds.DataProvider = DataProvider.OLEDB;  
rep.DataSources.Add(ds);  
mds.RecordSource = "select * from products where categoryid =  
[CategoryParam]";  
ReportParameter rp = new ReportParameter();  
rp.DataType = Doc.ParameterType.Integer;  
rp.Prompt = "Category";  
rp.Name = "CategoryParam";  
rp.AllowedValuesDefinition.Binding.DataSourceName = "CategoriesDS";  
rp.AllowedValuesDefinition.Binding.ValueExpression = "CategoryID";  
rp.AllowedValuesDefinition.Binding.LabelExpression = "CategoryName";  
rep.Parameters.Add(rp);
```

## 計算フィールドの定義

計算フィールドには、実行時に評価される式が設定されます。計算フィールドは、**DataSource.CalculatedFields.Add** メソッドを使用してデータソースに追加できます。


たとえば、「CategoryID \* 2」を計算する整数型の計算フィールド「Calc1」を追加するコードは、次のようになります。

Visual Basic

```
Dim ds As DataSource = C1FlexReport1.DataSources(0)  
ds.CalculatedFields.Add(New CalculatedField("Calc1", GetType(Integer),  
"CategoryID * 2"))
```

C#

```
DataSource ds = c1FlexReport1.DataSources[0];  
ds.CalculatedFields.Add(new CalculatedField("Calc1", typeof(int),  
"CategoryID * 2"));
```

 複数の計算フィールドがある場合、それらは一意的な名前を持つ必要があります。

## デスクトップ用 FlexReport の開発

一般的なデスクトップ環境では、レポートを生成および表示するコンピュータと同じコンピュータで **C1FlexReport** を実行します。その場合でも、レポートのデータ自体は、リモートサーバーから取得されることがあります。次のセクションでは、FlexReport が Visual Studio 環境でホストされているとします。



## 実行時の FlexReport のロード

実行時にレポートをロードするには、レポート定義ファイルとビューアが必要です。このようなアプリケーションの主な長所は、レポート形式を変更しても、アプリケーションを更新する必要がないことです。新しいレポート定義ファイルをユーザーに送るだけで済みます。

実行時にロードされるレポートを備えたアプリケーションを作成するには、次の手順に従います。

1. **C1FlexReportDesigner** アプリケーションで、必要なレポートをすべて作成します。詳細については、「[クイックスタート](#)」を参照してください。
2. 次のコントロールをアプリケーションに追加します。
  - **fv** という名前の **C1FlexViewerPanel** コントロール
  - **cmbReport** という名前の **ComboBox** コントロール
  - **status** という名前の **StatusBar** コントロール
  - **button1** という名前の **button** コントロール
3. コードビューで、次の `Import` 文をファイルの先頭に追加します。

Visual Basic

```
Imports C1.Win.FlexReport
Imports System.IO
```

C#

```
using C1.Win.FlexReport;
using System.IO;
```

これにより、完全名前空間を指定しなくても、**C1FlexReport** と **System.IO** のクラスとオブジェクトを参照できます。

4. レポート定義ファイルを読み取り、すべてのレポートのリストを構築し、ユーザーが選択したレポートをレンダリングために、次のコードをボタンクリックイベントに追加します。

Visual Basic

```
Private Sub btnRender_Click(sender As Object, e As EventArgs)
    ' アプリケーションパスを取得します
    Dim appPath As String
    appPath = System.IO.Path.GetDirectoryName
        (System.Reflection.Assembly.GetExecutingAssembly().Location).ToLower()

    Dim i As Integer = appPath.IndexOf(vbBack & "in")
    If(i < 0) Then
        i = appPath.IndexOf(vbBack & "in")
    End If
    If(i > 0) Then
        appPath = appPath.Remove(i, appPath.Length - i)
    End If
    ' レポート定義ファイルからレポートの名前を取得します
    Dim m_ReportDefinitionFile As String = appPath &
        Convert.ToString("¥Data¥Products Report.flxr")
    Dim reports As String() =
        C1FlexReport.GetReportList(m_ReportDefinitionFile)
    ' コンボボックスにデータを設定します
    cmbReport.Items.Clear()
```

```
For Each report As String In reports
cmbReport.Items.Add(report)
Next

Try
Cursor = System.Windows.Input.Cursors.Wait

    ' レポートをロードします
fv.StatusText = "Loading" + cmbReport.Text
rep.Load(m_ReportDefinitionFile, cmbReport.Text)

    ' 印刷プレビューコントロールにレンダリングします
fv.StatusText = "Rendering" + cmbReport.Text
fv.DocumentSource = rep

    ' 印刷プレビューコントロールにフォーカスを設定します
fv.Focus()
Finally
Cursor = InlineAssignHelper(Cursor,
System.Windows.Input.Cursors.Arrow)
End Try
End Sub
```

C#

```
private void btnRender_Click(object sender, EventArgs e)
{
    // アプリケーションパスを取得します
    string appPath;
    appPath = System.IO.Path.GetDirectoryName
(System.Reflection.Assembly.GetExecutingAssembly().Location).ToLower();

    int i = appPath.IndexOf("\bin");
    if ((i < 0)) {
        i = appPath.IndexOf("\bin");
    }
    if ((i > 0)) {
        appPath = appPath.Remove(i, appPath.Length - i);
    }
    // レポート定義ファイルからレポートの名前を取得します
    string m_ReportDefinitionFile = appPath + @"¥Data¥Products
Report.flxr";
    string[] reports =
C1FlexReport.GetReportList(m_ReportDefinitionFile);
    // コンボボックスにデータを設定します
    cmbReport.Items.Clear();

    foreach(string report in reports) {
        cmbReport.Items.Add(report);
    }
}
```

```

try {
    Cursor = System.Windows.Input.Cursors.Wait;

    // レポートをロードします
    fv.StatusText = "Loading" + cmbReport.Text;
    rep.Load(m_ReportDefinitionFile, cmbReport.Text);

    //印刷プレビューコントロールにレンダリングします
    fv.StatusText = "Rendering" + cmbReport.Text;
    fv.DocumentSource = rep;

    // 印刷プレビューコントロールにフォーカスを設定します
    fv.Focus();
} finally {
    Cursor = Cursor = System.Windows.Input.Cursors.Arrow;
}
}

```

このコードは、最初に、レポート定義が格納されているファイルの場所を取得します。それには、システム定義の **Path** クラスと **Application** クラスの静的メソッドを使用します。レポート定義ファイルの場所と名前に合わせてコードを調整してください。

次に、**GetReportList** メソッドを使用してレポート定義ファイル(手順 1 で作成)内のすべてのレポート名を含む配列を取得し、ユーザーがレポートを選択するためのコンボボックスにレポート名を挿入します。

5. プロジェクトを実行します。

## パラメータの追加

パラメータは、どのようなレポートでも重要な要素です。パラメータは、レポートに渡されたデータを操作することで、挿入されるデータに影響を及ぼします。パラメータを使用して、データのデフォルト値を変更したり、データにフィルタを適用することができます。また、複数値パラメータを使用して、複数の値を選択することもできます。

FlexReport のパラメータコレクション **C1FlexReport.Parameters** でパラメータを定義して、タイプ、キャプション、デフォルト値、有効な値などを指定できます。

**C1FlexReport.Parameters** コレクションでパラメータとして定義される各要素は、**ReportParameter** クラスのインスタンスです。これは次のプロパティを持ちます。

プロパティ	説明
<b>Nullable</b>	このパラメータの値を Null にできるかどうかを示す値を取得または設定します。複数値パラメータの場合は、true にできません。
<b>AllowBlank</b>	このパラメータの値を空の文字列にできるかどうかを示す値を取得または設定します。DataType が String でない場合は無視されます。
<b>MultiValue</b>	これが複数値パラメータ(値のセットを受け取ることができるパラメータ)であるかどうかを示す値を取得または設定します。
<b>Hidden</b>	このパラメータをユーザーに非表示にする(ただし、サブレポート、ドリルスルーレポートなどでプログラミングによって使用することは可能)かどうかを示す値を取得または設定します。
<b>Prompt</b>	パラメータ値の入力を求める際にエンドユーザーに表示されるプロンプトを取得または設定します。

<b>Value</b>	パラメータ値を取得または設定します。MultiValue が true の場合は、Value に配列を指定できます(この場合は、すべての項目が同じ項目タイプを持つ必要があります)。
<b>DataType</b>	パラメータのデータタイプを取得または設定します。
<b>AllowedValuesDefinition</b>	このパラメータの許容値のリストを定義する AllowedValuesDefinition を取得します。許容値は、AllowedValuesDefinition.Values プロパティを使用して静的リストとして指定するか、AllowedValuesDefinition.Binding プロパティを使用してレポートのデータソースの 1 つに連結された動的リストとして指定できます。

レポートパラメータは、FlexReportDesigner アプリケーションを使用して簡単に追加できます。詳細については、「[複数データソースレポートでのパラメータへのデータの連結](#)」を参照してください。

## データのグループ化

グループ化は、整理された方法でデータを表示するために最もよく使用される方法です。基本レイアウトの設計が完了したら、特定のフィールドなどの基準に基づいてレコードを区切ることで、読みやすいレポートを作成できます。データをグループ化することで、レコードをグループに分け、グループごとに概要やサマリーデータを表示することができます。グループの区切りは、グループ化式に基づいて行われます。この式は、1 つ以上のレコードセットフィールドに基づいて作成されることが普通ですが、必要に応じてさらに複雑な式にすることもできます。

FlexReport でグループ化を実行するには、**C1FlexReport.Groups** を使用します。

たとえば、特定の肩書きや役職に該当する従業員の名前をリストすることにします。この場合は、Title(肩書き)でリストをグループ化します。以下の手順は、従業員リストを Title でグループ化する方法を示します。この例では、「[FlexReport クイックスタート](#)」で作成したサンプルを使用します。

1. [クイックスタートプロジェクト](#)の XAML デザイン に、**C1CheckBox** を追加します。
2. C1CheckBox の **Name** を「groupC1CheckBox」に、また **Content** を「タイトルでレポートをグループ化する」に設定します。
3. **Checked** イベントを c1CheckBox1\_Checked という名前で作成します。
4. 次のコードを追加します。

```

    ◦ Visual Basic
Dim grp As Group

Private Sub c1CheckBox1_Checked(sender As Object, e As RoutedEventArgs)
    ' 従業員を役職でグループ化し、役職を昇順にソートします
    grp = rep.Groups.Add("GrpTitle", "Title", SortEnum.Ascending)
    ' 新しいグループのヘッダーセクションを書式設定します
    s = grp.SectionHeader
    s.Height = 1000
    s.Visible = True

    Dim f As New TextField()
    f.Name = "Title"
    f.Text.Expression = "Title"
    f.Left = 0
    f.Top = 0
    f.Width = rep.Layout.Width
    f.Height = 500
    f.Align = FieldAlignEnum.LeftMiddle
    f.Font.Bold = True
    f.Font.Size = 12
    f.Border = New C1.WPF.Document.Border _
        (2, Color.FromRgb(0, 0, 0), C1.WPF.Document.DashStyle.Solid)
    f.BackgroundColor = Color.FromRgb(150, 150, 220)
    f.MarginLeft = 100
    s.Fields.Add(f)
    rep.Render()
End Sub

    ◦ C#
Group grp;

```

```
private void c1CheckBox1_Checked(object sender, RoutedEventArgs e)
{
    // 従業員を役職でグループ化し、役職を昇順にソートします
    grp = rep.Groups.Add("GrpTitle", "Title", SortEnum.Ascending);
    // 新しいグループのヘッダーセクションを書式設定します
    s = grp.SectionHeader;
    s.Height = 1000;
    s.Visible = true;

    TextField f = new TextField();
    f.Name = "Title";
    f.Text.Expression = "Title";
    f.Left = 0;
    f.Top = 0;
    f.Width = rep.Layout.Width;
    f.Height = 500;
    f.Align = FieldAlignEnum.LeftMiddle;
    f.Font.Bold = true;
    f.Font.Size = 12;
    f.Border = new Cl.WPF.Document.Border
    (2, Color.FromRgb(0, 0, 0), Cl.WPF.Document.DashStyle.Solid);
    f.BackgroundColor = Color.FromRgb(150, 150, 220);
    f.MarginLeft = 100;
    s.Fields.Add(f);
    rep.Render();
}
}
```

- プロジェクトを実行します。[Employees]ボタンをクリックして、レポートをレンダリングします。
- [タイトルでレポートをグループ化する]チェックボックスをクリックして、レポートにグループを表示します。役職のリストがタイトルでグループ化され、タイトルが昇順にソートされていることを確認します。

The screenshot shows a WPF application window titled "Employeesレポート". The report content is as follows:

ID	名	姓	肩書き	メモ
<b>札幌支店営業課</b>				
4	秀樹	田中	札幌支店営業課	私は、東京生まれの東京育ち。趣味と言えほどのものはありませんが、強いて言うならばが好きなので、たまにドライブへ出かけます。新緑や紅葉の季節は特に山がきれいです。
<b>仙台支店営業部</b>				
7	哲也	古田	仙台支店営業部	私の出身は、ロサンゼルスです。肉ごころにいた子供時代は、よく両親にディスニーランドへ連れて行ってもらいました。日本にも東京ディスニーランドが出来、やはり、子供に連れて行って一緒に出かけます。
<b>大阪支社営業部</b>				
6	誠一	東海	大阪支社営業部	私は、スポーツが好きで、特にサッカーは見るのもやるのも夢中になります。並行して、家が以前、エレーターの敷置を聞いていたので、装飾もよく演替したりしていました。
<b>長崎支店営業課</b>				
9	真紀	高橋	長崎支店営業課	私は学生時代は演劇部に所属していたので、今もよく、休みの日は芝居を見に行きます。一晩は、舞台の仕立に参観した事もあり、舞台関係の季節は少し構っていただけました。

## データのソート

ソートは、データを昇順または降順に整理する方法です。

FlexReport でソートを実行するには、**DataSource.SortDefinitions** を使用します。

# FlexReport for WPF

たとえば、従業員の名前のリストを昇順に表示することになります。この場合は、リストを「名 (First Name)」でソートする必要があります。以下の手順は、従業員リストの名前をアルファベット順にソートする方法を示します。この例では、「FlexReport」で作成したサンプルを使用します。

1. FlexReport クイックスタートプロジェクトのフォームに、**C1Button** を追加します。
2. C1Button の **Name** を「sortButton」に設定し、**Text** を「従業員の名前でレポートをソートする」に設定します。
3. **Click** イベントを sortC1Button\_Click という名前で作成します。
4. 従業員の名前を昇順でソートするには、次のコードを追加します。

○ **Visual Basic**

```
Dim asc As Boolean = True
```

```
Private Sub sortC1Button_Click(sender As Object, e As RoutedEventArgs)
    If asc Then
        Dim sd As New SortDefinition("[FirstName]", SortDirection.Ascending)
        rep.DataSource.SortDefinitions.Add(sd)
        asc = False
    End If
    rep.Render()
End Sub
```

○ **C#**

```
bool asc = true;
private void sortC1Button_Click(object sender, RoutedEventArgs e)
{
    if (asc)
    {
        SortDefinition sd = new SortDefinition("[FirstName]", SortDirection.Ascending);
        rep.DataSource.SortDefinitions.Add(sd);
        asc = false;
    }
    rep.Render();
}
```

5. レポートをプレビュー表示します。[Employees] ボタンをクリックして、レポートをレンダリングします。
6. [Sort Report by Employee First Name] ボタンをクリックして、レポートのソート結果を表示します。



ID	名	姓	肩書き	メモ
1	恵子	正門	本社第一営業部	私は、幼稚園の頃から水泳を始めて、学生時代もずっと水泳部でした。海にも毎年行きます。イルカが好きで、夏に行ける機会はありません。食べ物ではイクラが好きです。
2	優久馬	森上	本社第二営業部	私は中学生までの間、合宿をやっていました。声が高いので、バートはテニスでした。流行のサッカーよりも野球の方が好きで、ジャイアンツのファンです。
3	和明	木下	立川営業所	私の家は水産なので、よく手伝われ、地産に穴をたくさん掘った経験があります。そのおかげか、岸はとて大変です。休日は、ときどきですが、所属しているボウリングクラブの練習をしています。
4	秀樹	田中	札幌支店営業課	私は、東京生まれの東京育ち。極端と言えほどのものではありませんが、強いて言うなら車が好きなもので、たまにドライブに出かけます。新緑や紅葉の季節は特に山がきれいです。
5	典央	明山	名古屋支店営業部	私はご褒めとあり、声が高いので、学生のときの経済活動は、バスケットボールでした。こんなに聲が伸びた原因は、やはり家が魚屋だからだろうと思います。ものごころの付いた頃には、既に毎日、魚店には魚がよっていました。
6	誠一	東海	大阪支社営業部	私は、スポーツが好きで、特にサッカーは見るのも夢中になります。並行して、家が以前、エレベーターの設置を聞いていたので、歯磨きもよく練習したりしていました。
7	哲也	古田	仙台支店営業部	私の出身は、ロサンゼルスです。向こうにいた子供時代は、よく両親にデイスニランドへ連れて行ってもらいました。日本にも東京デイスニランドが出来、やはり、子供に連れて行って一層に出かけます。
8	彩子	松沢	福岡支店営業部	私は、両親共にアメリカ人ですが、生まれも育ちも日本です。日本の文化はもともと気に入っていますが、何と言っても、技術の進歩の速さに興味があります。個人的には、パソコンが趣味です。
9	真紀	高橋	長崎支店営業課	私は学生時代は演劇部に所属していたので、今もよく、休みの日は芝居を見に行きます。一時は、舞台の仕事に就こうと思ったこともあり、舞台関係の学校に少し通っていました。

## データのフィルタ処理

特定の条件に基づいてデータの一部だけを表示する場合は、データのフィルタ処理が重要となります。FlexReport では、**DataSource.Filter** を使用してデータがフィルタ処理されます。

たとえば、「FlexReport クイックスタート」で作成したレポートで、1つの Employee ID に対応する従業員詳細を表示することになります。Detail セクションで EmployeeID フィールドが追加されている場所に、次のコードを追加して、'EmployeeID = 2' に対応する従業員詳細をフィルタ処理します。

- Visual Basic

```
'EmployeeID = 2に対応する従業員の詳細をフィルタリングします
rep.DataSource.Filter = "EmployeeID = 2"
```

- C#

```
//EmployeeID = 2に対応する従業員の詳細をフィルタリングします
rep.DataSource.Filter="EmployeeID = 2";
```

プロジェクトを実行します。出力に EmployeeID = 2 の従業員の名前 (First name)、姓 (Last name)、役職 (Title)、およびメモ (Notes) が表示されます。

The screenshot shows a report titled "Employeesレポート" with a table containing one row of data for Employee ID 2. The table has four columns: ID, Name, Title, and Memo.

ID	名	姓	肩書き	メモ
2	徳久馬	森上	本社第二営業部	私は中学生までの間、合宿をやっていました。声が高いので、パートはテニスでした。流行のサッカーよりも野球の方が好きで、ジャイアンツのファンです。

## レポートのエクスポート

レポートをファイルにエクスポートし、それをクライアントや同僚に電子メールで配布する場合があります。これは、FlexReport コントロールを使用して行うことができます。FlexReport は、次のエクスポート形式をサポートしています。

形式	説明
PDF (.pdf)	Adobe の Acrobat ビューアまたはブラウザプラグインを備えたコンピュータ上で表示できる PDF ファイルを作成します。
HTML(.html)	レポートのHTMLファイルを作成します。
RTF (.rtf)	一般的なワードプロセッサ (Microsoft Word や WordPad) で開くことができる RTF ファイルを作成します。これは、ページ付き XML ドキュメントまたは Open XML ドキュメントとして保存できます。
Microsoft Excel Open XML (.xlsx)	Microsoft Excel 2007以上で開くことができる.xlsxファイルを作成します。
Open XML Word (*.docx)	Microsoft Word 2007以上で開くことができるDOCXファイルを作成します。

形式	説明
TIFF (.tiff), BMP (.bmp), PNG (.png), JPEG (.jpg), GIF (.gif) images	TIFF (Tag Image File Format)、BMP (Bitmap Images)、PNG (Portable Network Graphic)、JPEG (Joint Photographic Experts Group)、または GIF (Graphic Interchange Format) 型の画像ファイルを作成します。
Compressed Metafile (*.zip)	EmfOnly、EmfPlusOnly、およびEmfPlusDualの圧縮メタファイルを作成します。

## PdfFilterクラスを使用したFlexReportのレンダリング

**FlexReport** では、**PdfFilter** クラスを使用して、レポートを PDF 形式にレンダリングすることができます。**PdfFilter** クラスは、レポートを PDF ストリームまたはファイルにレンダリングします。

FlexReport を PDF 形式でレンダリングするには、次のコードを使用します。この例では、「[クイックスタート](#)」で作成したサンプルを使用します。

- Visual Basic

```
Dim f As New PdfFilter()  
f.FileName = "..\..\ProductsReport.pdf"  
rep.RenderToFilter(f)  
System.Diagnostics.Process.Start(f.OutputFiles(0))
```

- C#

```
PdfFilter f = new PdfFilter();  
f.FileName = @"..\..\ProductsReport.pdf";  
rep.RenderToFilter(f);  
System.Diagnostics.Process.Start(f.OutputFiles[0]);
```

**PdfFilter** クラスの次のプロパティを使用することもできます。

プロパティ	説明
EmbedFonts	PDF ドキュメントへのフォント情報の埋め込みに使用されます。
PdfACompatible	PDF/A 互換ドキュメントの生成に使用されます。
PdfSecurityOptions	PDF ドキュメントを誰が使用し、どのアクションを使用できるかを指定するために使用されます。
UseCompression	PDF ドキュメントの圧縮に使用されます。
UseOutlines	アウトラインツリーを追加するために使用されます。

## HTMLフィルタクラスを使用したFlexReportのレンダリング

**FlexReport** では、**HtmlFilter** クラスを使用して、レポートを HTML 形式にレンダリングすることができます。**HtmlFilter** クラスは、レポートを HTML ストリームまたはファイルにレンダリングします。

FlexReport を HTML 形式でレンダリングするには、次のコードを使用します。この例では、「[FlexReport クイックスタート](#)」で作成したサンプルを使用します。

- Visual Basic

```
Dim f As New HtmlFilter()  
f.FileName = "..\..\ProductsReport.html"  
rep.RenderToFilter(f)  
System.Diagnostics.Process.Start(f.OutputFiles(0))
```



- C#

```
HtmlFilter f = new HtmlFilter();
f.FileName = @"..\..\ProductsReport.html";
rep.RenderToFilter(f);
System.Diagnostics.Process.Start(f.OutputFiles[0]);
```

## さまざまな画像ファイル形式でレポートのレンダリング

**FlexReport** では、PNG、JPG、BMP、GIF、TIFF などのさまざまな画像ファイル形式でレポートをレンダリングすることができます。これらの画像ファイル形式はそれぞれ、特定の形式の専用エクスポートフィルタを備えています。**PngFilter** クラスは、レポートを PNG 形式でレンダリングするために使用できます。同様に、**JpegFilter** クラスはレポートを JPEG 形式でレンダリングするために使用され、**GifFilter** クラスはレポートを GIF 形式でレンダリングするために使用され、**BmpFilter** クラスはレポートを BMP 形式でレンダリングするために使用され、**TiffFilter** クラスはレポートを TIFF 形式でレンダリングするために使用されます。

FlexReport を PNG 形式でレンダリングするには、次のコードを使用します。この例では、「FlexReport クイックスタート」で作成したサンプルを使用します。

- Visual Basic

```
Dim f As New PngFilter()
f.FileName = "..\..\ProductsReport.png"
rep.RenderToFilter(f)
System.Diagnostics.Process.Start(f.OutputFiles(0))
```

- C#

```
PngFilter f = new PngFilter();
f.FileName = @"..\..\ProductsReport.png";
rep.RenderToFilter(f);
System.Diagnostics.Process.Start(f.OutputFiles[0]);
```

## RtfFilterクラスを使用したFlexReportのエクスポート

**FlexReport** では、**RtfFilter** クラスを使用して、レポートを RTF 形式にエクスポートすることができます。**RtfFilter** クラスは、レポートを RTF ストリームまたはファイルにエクスポートします。

FlexReport を RTF 形式にエクスポートするには、次のコードを使用します。この例では、「FlexReport クイックスタート」で作成したサンプルを使用します。

- Visual Basic

```
Dim f As New RtfFilter()
f.FileName = "..\..\ProductsReport.rtf"
rep.RenderToFilter(f)
System.Diagnostics.Process.Start(f.OutputFiles(0))
```

- C#

```
RtfFilter f = new RtfFilter();
f.FileName = @"..\..\ProductsReport.rtf";
rep.RenderToFilter(f);
System.Diagnostics.Process.Start(f.OutputFiles[0]);
```

**RtfFilter** クラスの次のプロパティを使用することもできます。

プロパティ	説明
OpenXML	OpenXML 形式でのファイルのエクスポートに使用されます。

Paged	元のレポートのページレイアウトを保持するために使用されます。
ShapesWord2007Compatible	DOCX 形式で保存するレポートで Word 2007 図形形式互換を示すために使用されます。

## ExportFilterクラスを使用したレポートのエクスポート

特定の形式フィルタ(PdfFilter など)を使用して特定の形式(PDF 形式など)にレポートをエクスポートする代わりに、**FlexReport** では、**ExportFilter** クラスを使用して、レポートをさまざまな形式にエクスポートすることができます。**ExportFilter** クラスは、すべてのエクスポートクラスの抽象基本クラスです。**ExportFilter** クラスのオブジェクトは、レポートをさまざまな形式にエクスポートするために使用されません。**ExportFilter** クラスは、サポートされるエクスポート形式を記述する **ExportProvider** 抽象クラスにアクセスします。

次のコードでは、**ExportProvider** クラスにアクセスする **ExportFilter** クラスを使用して、この **ExportProvider** クラスに記述されたすべてのエクスポート形式を **ComboBox** に一覧表示しています。そのために、有効なエクスポート形式のリストを表示する **ComboBox** コントロールをデザインビューに追加し、その名前を **cbxExportFormat** に設定します。

1. C1.WPF.4.dll、C1.WPF.Document.4.dll、および C1.WPF.FlexReport.4.dll への参照を追加します。
2. コードで、次の名前空間を追加します。

Visual Basic

```
Imports C1.WPF.Document.Export
Imports C1.WPF.FlexReport
Imports System.IO
```

C#

```
using C1.WPF.Document.Export;
using C1.WPF.FlexReport;
using System.IO;
```

3. コードビューに次のコードを追加して、**C1FlexReport** のインスタンスを作成します。

Visual Basic

```
Private _report As New C1FlexReport()
```

C#

```
private C1FlexReport _report = new C1FlexReport();
```

4. XAML の相互作用ロジックの InitializeComponent() メソッドの下に次のコードを追加します。

- Visual Basic

- ・ サポートされているエクスポートフィルタのリストを作成します

```
For Each e In _report.SupportedExportProviders
    cbxExportFormat.Items.Add(e.FormatName)
Next
cbxExportFormat.SelectedIndex = 0
```

- C#

- // サポートされているエクスポートフィルタのリストを作成します

```
foreach (var e in _report.SupportedExportProviders)
{
    cbxExportFormat.Items.Add(e.FormatName);
}
cbxExportFormat.SelectedIndex = 0;
```

5. **ExportFilter** と **ExportProvider** クラスを使用して FlexReport をさまざまな形式にエクスポートするために、MainWindow に 1 つの Button コントロールを追加します。次のコードを使用します。

- Visual Basic

```
Dim reports As String() = C1FlexReport.GetReportList("../..\FlexCommonTasks_WPF.flxr")
Dim ep As ExportProvider = _report.SupportedExportProviders(cbxExportFormat.SelectedIndex)
```

```

Dim ef As ExportFilter = TryCast(ep.NewExporter(), ExportFilter)
If Not Directory.Exists("../..\ExportResults") Then
    Directory.CreateDirectory("../..\ExportResults")
End If

For Each reportName As String In reports
    Using rep As New C1FlexReport()
        rep.Load("../..\FlexCommonTasks_WPF.flxr", reportName)
        ef.FileName = String.Format("../..\ExportResults\{0}.{1}",
            reportName, ep.DefaultExtension)
        Try
            rep.RenderToFilter(ef)
        Catch ex As Exception
            MessageBox.Show(String.Format("Exception while export [{0}] report:" _
                & vbCrLf & vbLf & "{1}", reportName, ex.Message), "Error",
                MessageBoxButton.OK, MessageBoxImage.Error)
        End Try
    End Using
Next

o C#
string[] reports = C1FlexReport.GetReportList(@"..\..\FlexCommonTasks_WPF.flxr");
ExportProvider ep = _report.SupportedExportProviders[cbxExportFormat.SelectedIndex];
ExportFilter ef = ep.NewExporter() as ExportFilter;
if (!Directory.Exists(@"..\..\ExportResults"))
    Directory.CreateDirectory(@"..\..\ExportResults");

foreach (string reportName in reports)
{
    using (C1FlexReport rep = new C1FlexReport())
    {
        rep.Load(@"..\..\FlexCommonTasks_WPF.flxr", reportName);
        ef.FileName = string.Format(@"..\..\ExportResults\{0}.{1}",
            reportName, ep.DefaultExtension);
        try
        {
            rep.RenderToFilter(ef);
        }
        catch (Exception ex)
        {
            MessageBox.Show(string.Format
                ("Exception while export [{0}] report:\r\n{1}", reportName, ex.Message),
                "Error", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}

```

**ExportFilter** クラスの次のプロパティを使用することもできます。

プロパティ	説明
FileName	出力ファイルに名前を付けるために使用されます。
MultiFile	エクスポート時に複数のファイルが生成されたかどうかを示すために使用されます。
OutputFiles	Export(string) の呼び出しによって生成されたファイルのリストを表示するために使用されます。
PageSettings	ページに適用する設定を指定するために使用されます。
Preview	ドキュメントをディスクファイルにエクスポートした後に、そのドキュメントを表示するかどうかを示すために使用されます。
Range	エクスポートするページの範囲を表すために使用されます。
ShowOptions	ドキュメントのエクスポートの前にオプションダイアログを表示するかどうかを示すために使用されます。
Stream	出力ストリームを指定するために使用されます。
UseZipForMultipleFiles	エクスポート時に複数のファイルが生成される場合に、出力(ストリームまたはファイル)を、生成されたファイルを含む 1 つの zip 形式アーカイブにするかどうかを示すために使用されます。

## FlexReportの印刷

レポートを作成したら、それを印刷することもできます。それには、**C1DocumentSource** クラスの **Print** メソッドを使用します。このメソッドを使用して、生成したレポートを印刷することができます。

レポートを印刷するには、次のコードを使用します。

- **Visual Basic**

```
Dim rep As New C1FlexReport()  
rep.Load("../..\Products Report.flxr", "Products Report")  
rep.Render()  
rep.Print()
```

- **C#**

```
C1FlexReport rep = new C1FlexReport();  
rep.Load(@"..\..\Products Report.flxr", "Products Report");  
rep.Render();  
rep.Print();
```

## VBScript の操作

**VBScript 式**は、レポートのコンテンツを取得、計算、表示、グループ化、ソート、フィルタ、パラメータ化、および書式設定するためにレポート定義内で広く使用されます。いくつかの式は自動的に作成されます。たとえば、ツールボックスからフィールドをレポートのセクションにドラッグすると、テキストボックスにそのフィールドの値を取得する式が表示されます。ただし、ほとんどの場合、レポートに機能を追加するには独自に式を作成する必要があります。

VBScript の式と文には次の違いがあります。

- **式**は値を返します。これを **Field.Text** プロパティなどに割り当てることができます。  
`Field1.Text.Expression = "iif( 1=1, 1+2, 1+3 )"`
- **文**は値を返しません。**OnFormat** などのイベントプロパティに割り当てることができます。次に例を示します。  
`c1FlexReport.OnOpen = "if 1=1 then msgbox("OK!!!") else msgbox("oops")"`

**C1FlexReport** は VBScript を使用して、計算フィールドの式を評価したり、レポートイベントを処理します。

VBScript は完全な機能を備えた言語で、C1FlexReport の式を記述する際は、VBScript のすべてのメソッドと関数にアクセスできます。VBScript 言語の組み込みの機能については、[Microsoft Developer's Network \(MSDN\)](#) を参照してください。

新しい VBScript エディタでグローバルスクリプトを記述できます。このエディタを使用して、レポート全体からアクセス可能な VBScript の関数やサブルーチンを定義できます。VBScript エディタに直接アクセスするには、**F7** を押します。エディタを閉じて変更を保存するには、ショートカット **Ctrl+W** を使用します。エディタ内で、スクリプトを切り替えたり、フォントや色などのオプションを変更することができます。また、構文チェック、定義済み VBScript 関数、再編成されたスクリプト関数などの高度な機能を使用して、直感的なスクリプティングを簡単に行うことができます。

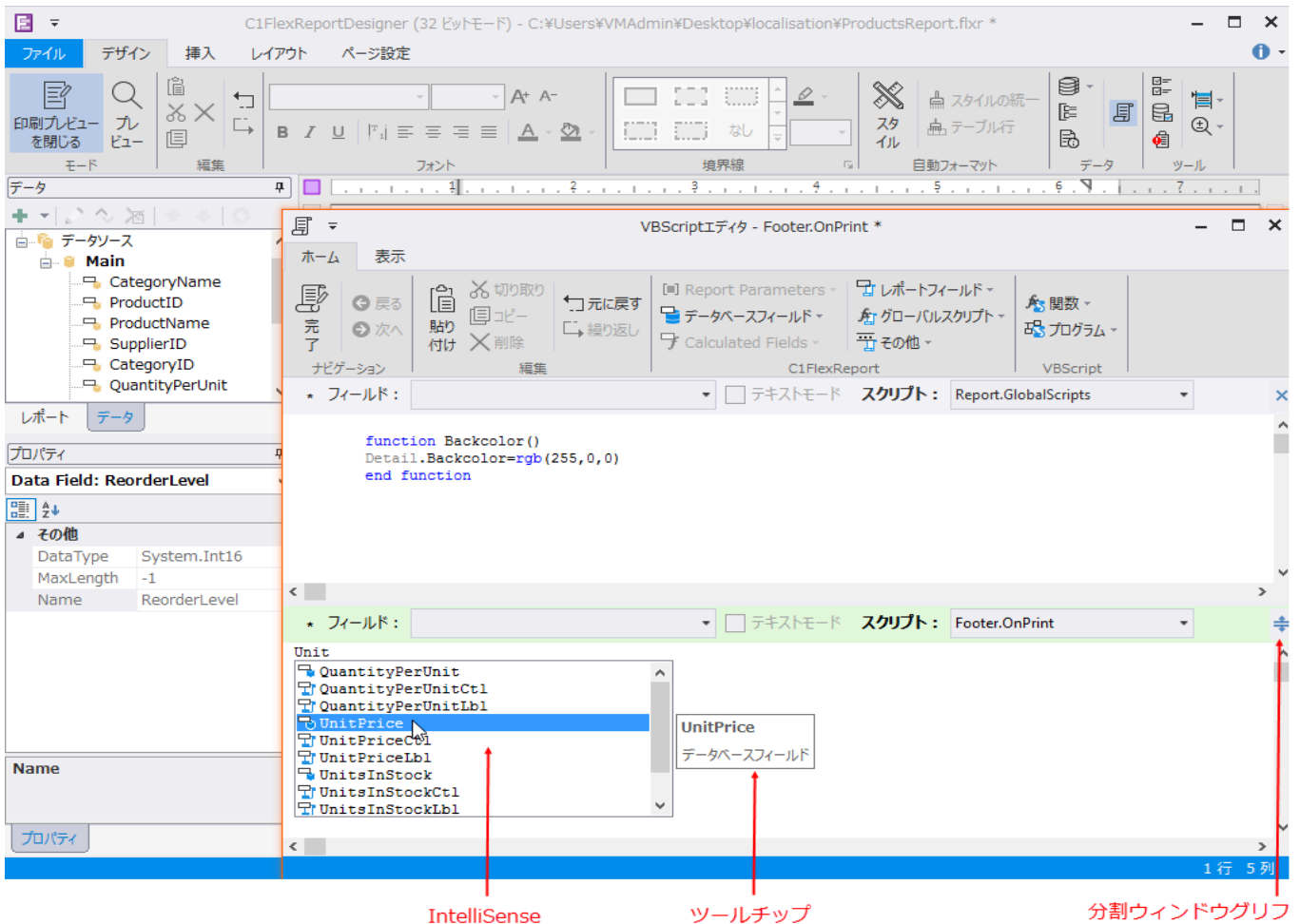
**[VBScript エディタ]** オプションを使用してグローバルスクリプトを記述するには

1. **C1FlexReportDesigner** の **[ホーム]** タブに移動します。
2. **[VBScript エディタ]** をクリックし、次の例のようなグローバルスクリプトを記述します。

```
function Backcolor()  
Detail.Backcolor=rgb(255,0,0)  
end function
```

次のように、**C1FlexReportDesigner** の **GlobalScripts** プロパティを使用してグローバルスクリプトを記述することもできます。

1. グローバルスクリプトを記述するレポートを選択します。
2. レポートの GlobalScripts プロパティに移動し、省略符をクリックします。これで、[VBScript エディタ]ダイアログボックスが開きます。
3. 上のように、VBScript エディタでグローバルスクリプトを記述します。



これで、レポート全体から使用できるグローバル関数 'Backcolor()' が定義されます。

VBScript エディタには、次の追加機能があります。

- **IntelliSense**: VBScript エディタがサポートするスクリプトの自動コード補完プロンプトを提供します。VBScript エディタの IntelliSense には、次の機能があります。
  - IntelliSense ウィンドウには、使用可能な語の状況依存リストと共に、VBScript 関数およびキーワードに関する詳細なヘルプが小さなツールチップまたはヘルプウィンドウに表示されます。詳細ヘルプ内の斜体フォントは、基本的に、現在の項目が属するカテゴリを示します (VBScript 関数、C1FlexReport 集計スクリプト関数、.NET オブジェクトプロパティなど)。
  - DataSource.Filter の編集時、エディタは「**式エディタ - DataSource.Filter**」として開かれ、IntelliSense には、エディタで使用できるキーワードまたは関数と、対応するヘルプが表示されます。
  - IntelliSense エントリに関連付けられているアイコンは、エントリのタイプを示します。アイコンのカラーパレットは、VBScript、レポート組み込み要素、DataSource.Filter によって異なります。
  - ユーザーがキー入力し、Intellisense ウィンドウが開くと、入力された文字に基づいてリストがフィルタ処理されます。たとえば、「t」と入力すると、文字「t」を含む語だけが表示され、「te」と入力すると、「te」を含む語にさらに絞り込まれます。
  - IntelliSense ウィンドウでバックスペースキーを押すと、最後のフィルタを元に戻します。
  - 角かっこ (I) キーを押すと、使用可能なデータベースフィールドのリストが表示されます。
  - レポート、フィールド、セクションなどのオブジェクトの名前の後にピリオド (.) キーを押すと、そのオブジェクトで使用できる .NET プロパティが表示されます。

# FlexReport for WPF

- [Ctrl]+[J]キー、[Ctrl]+[Space]キー、または文字以外のキーの後に文字キーを押すと、使用可能な VBScript 関数、キーワードなどのリストが状況に応じて表示されます。
- **分割ウィンドウ**: 同じ VBScriptEditor 内で 2 つの同じまたは異なるスクリプトを表示または記述できます。デフォルトでは、VBScript エディタは単一ウィンドウで開きます。


## 分割ウィンドウに切り替えるには

分割ウィンドウモードに切り替えるには、分割ウィンドウグリフをクリックして下へドラッグすると、上部にもう 1 つエディタが開きます。ウィンドウのサイズは、ウィンドウ間の分割線をドラッグして変更できます。

## 単一ウィンドウに戻るには

ウィンドウの右上隅にある [x] グリフをクリックすると、上側のウィンドウが閉じられ、分割モードがオフになり、下側のウィンドウがズームアウトします。リボンスターの有効/無効の状態は、現在のウィンドウによって決まります。現在のウィンドウは、薄緑のキャプションバーで示されます。分割ウィンドウモードには、次の追加機能があります。

- [F6] キーを押すと、2 つのウィンドウを切り替えることができます。
- 分割ウィンドウグリフまたは分割線を上側のウィンドウの上部までドラッグすると、分割ウィンドウモードの上側のウィンドウを非表示にできます。

 **VBScript エディタの[グローバルスクリプト]ドロップダウン**は、以前にレポートでグローバルスクリプトを定義した場合にのみ有効になります。

**C1FlexReport** は、追加のオブジェクト、変数、および関数を公開することで、VBScript を拡張します。

## FlexReport セクションの分割

FlexReports では、セクションやサブセクションをページ間で分割するかどうかを設定できます。それには、**SplitBehavior** プロパティを **SplitIfNeeded** または **KeepTogether** に設定します。同様に、フィールドや境界線の分割を **SplitHorzBehavior** プロパティと **SplitVertBehavior** プロパティで制御します。

次のコードは、セクションとサブセクションの **SplitBehavior** を設定します。

Visual Basic

◦ 必要に応じてセクションを分割できるようにします

```
rep.Sections.Header.SplitBehavior = SplitBehavior.SplitIfNeeded
```

◦ 必要に応じてサブセクションを分割できるようにします

```
rep.Sections.Header.SubSections(0).SplitBehavior =  
SplitBehavior.SplitIfNeeded
```

C#

// 必要に応じてセクションを分割できるようにします

```
rep.Sections.Header.SplitBehavior = SplitBehavior.SplitIfNeeded;
```

// 必要に応じてサブセクションを分割できるようにします

```
rep.Sections.Header.SubSections[0].SplitBehavior =  
SplitBehavior.SplitIfNeeded;
```

## フィールドの変更

VBScript は、計算フィールドで式を評価する場合にのみ使用できるわけではありません。レポートのレンダリング時に実行されるスクリプトを指定することもできます。このようなスクリプトを使用して、レポートの書式設定を変更できます。これらのスク

リポートは、イベントプロパティに置かれます。イベントプロパティは Visual Basic のイベントハンドラに似ていますが、レポートを表示するアプリケーションのスコープではなく、レポートのスコープ内でスクリプトが実行される点が異なります。

たとえば、イベントプロパティを使用すると、フィールドの **Font** プロパティや **ForeColor** プロパティをフィールドの値に基づいて設定できます。この動作は、レポート自体に組み込まれ、レンダリングに使用されるアプリケーションに関係なく保持されます。

もちろん、従来のイベントも使用できます。従来のイベントは、レポートではなくアプリケーションに影響を及ぼす動作を実装するために使用します。たとえば、**StartPage** イベントのハンドラを記述して、レンダリングされているレポートに関係なく、アプリケーションでページ数を更新できます。

次の表に、使用できるイベントプロパティと、その一般的な使用方法を示します。

オブジェクト	プロパティ	説明
C1FlexReport	<b>OnOpen</b>	レポートがレンダリングを開始すると実行されます。ConnectionString プロパティや RecordSource プロパティを変更したり、VBScript の変数を初期化するために使用できます。
	<b>OnClose</b>	レポートがレンダリングを終了すると実行されます。クリーンアップ作業を実行するために使用できます。
	<b>OnNoData</b>	レポートのレンダリングが開始されたが、ソースレコードセットが空の場合に実行されます。 <b>Cancel</b> プロパティを <b>True</b> に設定して、レポートの生成を中止できます。ダイアログボックスを表示して、レポートが表示されない理由をユーザーに知らせることもできます。
	<b>OnPage</b>	新しいページが開始されると実行されます。さまざまなフィールドのセクションの <b>Visible</b> プロパティを状況に応じて設定するために使用できます。コントロールは、新しいページが開始されるたびに自動的に 1 ずつ増える Page 変数を保持しています。
	<b>OnError</b>	エラーが発生すると実行されます。
	<b>GlobalScripts</b>	グローバル VBScript モジュールを指定します。ここで(標準の VBScript 構文を使用して)定義される関数とサブルーチンは、標準の VBScript 関数と同様に、現在のレポートの他のスクリプトで使用できます。
Section	<b>OnFormat</b>	セクション内のフィールドが評価される前に実行されます。この時点で、ソースレコードセット内のフィールドには、これからレンダリングされる値が反映されていますが、レポートフィールドには反映されていません。
	<b>OnPrint</b>	セクション内のフィールドが印刷される前に実行されます。この時点で、フィールドの評価は終了しており、条件付き書式設定を実行できます。

## サブセクションの追加

サブセクションは、レポートの任意のセクションに挿入できる追加セクションです。「[FlexReport のセクション](#)」で説明するように、通常、FlexReport には、詳細、ヘッダー、フッター、ページヘッダー、ページフッター、グループヘッダー、およびグループフッターセクションが含まれます。

これらのセクションには、それぞれ少なくとも 1 つのサブセクションが含まれますが、1 つのセクションに必要な数だけサブセクションを追加できます。

レポートのヘッダーセクションに 1 つのサブセクションを追加するには、次のコードを使用します。

Visual Basic

'ヘッダーセクションに 1 つのサブセクションを作成します

```
Dim ss As SubSection = rep.Sections.Header.SubSections.Add()
```

## FlexReport for WPF

```
' 高さを 10 mm に設定します  
ss.Height = 10 * 1440 / 25.4
```

C#

```
//ヘッダーセクションに 1 つのサブセクションを作成します  
SubSection ss = rep.Sections.Header.SubSections.Add();  
//高さを 10 mm に設定します  
ss.Height = 10 * 1440 / 25.4;
```




## FlexReportDesigner について

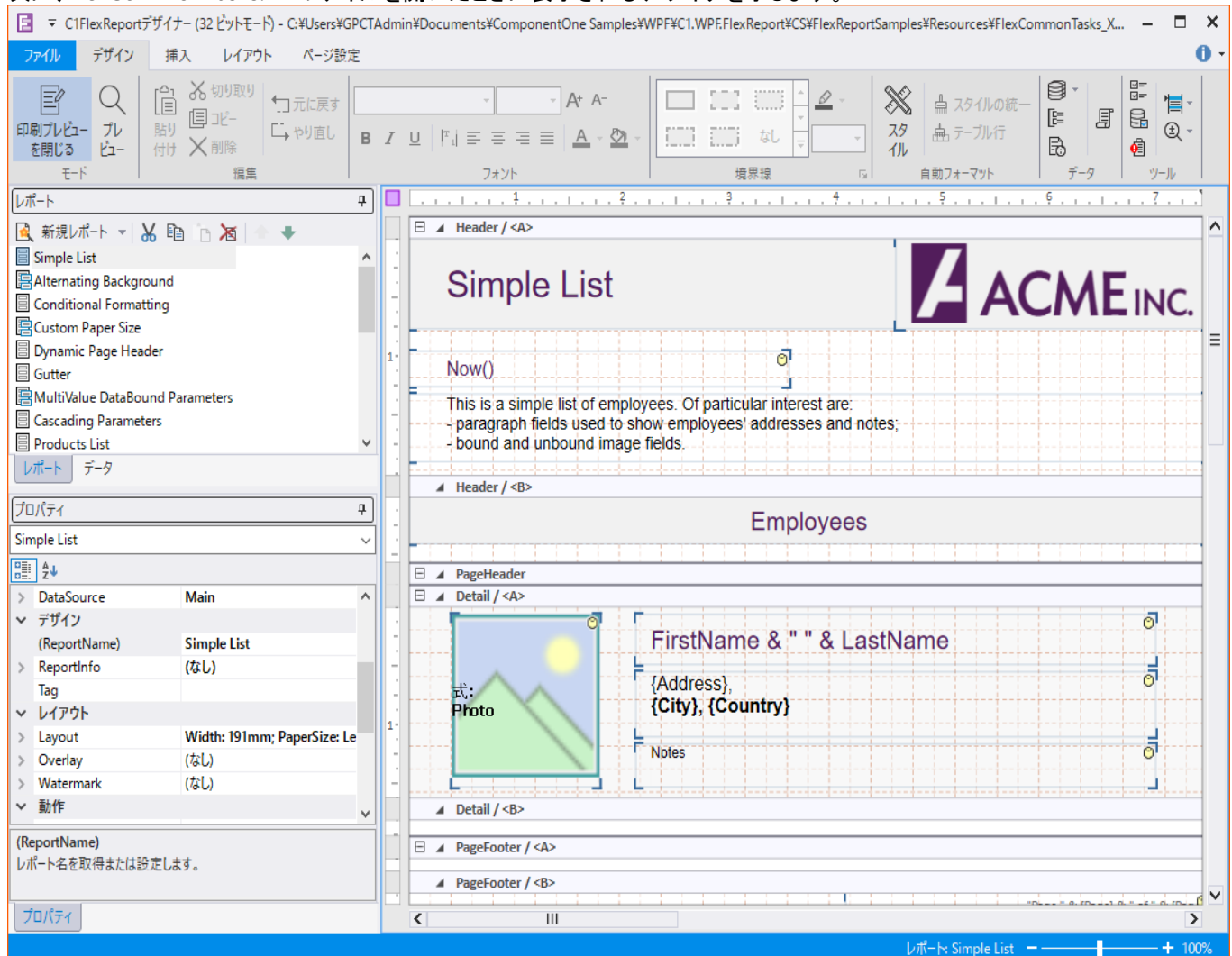
**FlexReportDesigner** アプリケーションは、**C1FlexReport** レポート定義ファイルを作成および編集するためのツールです。このデザイナーでは、.flxr 拡張子が付いたファイルの作成、編集、ロード、および保存を行うことができます。Microsoft Access ファイル(.mdb)や Crystal Reports(.rpt)から C1Report(.xml)やレポート定義をインポートすることもできます。

デザイナーを実行するには、**C1FlexReportDesigner.exe** (64ビットプラットフォーム用)または**C1FlexReportDesigner32.4.exe** (32ビットプラットフォーム用)ファイルをダブルクリックします。このファイルは、.NET 4.0 の場合、デフォルトで次のパスにあります。

**C:\Program Files (x86)\ComponentOne\Apps\v4**

 この場所は、デフォルトのインストールパスを反映します。

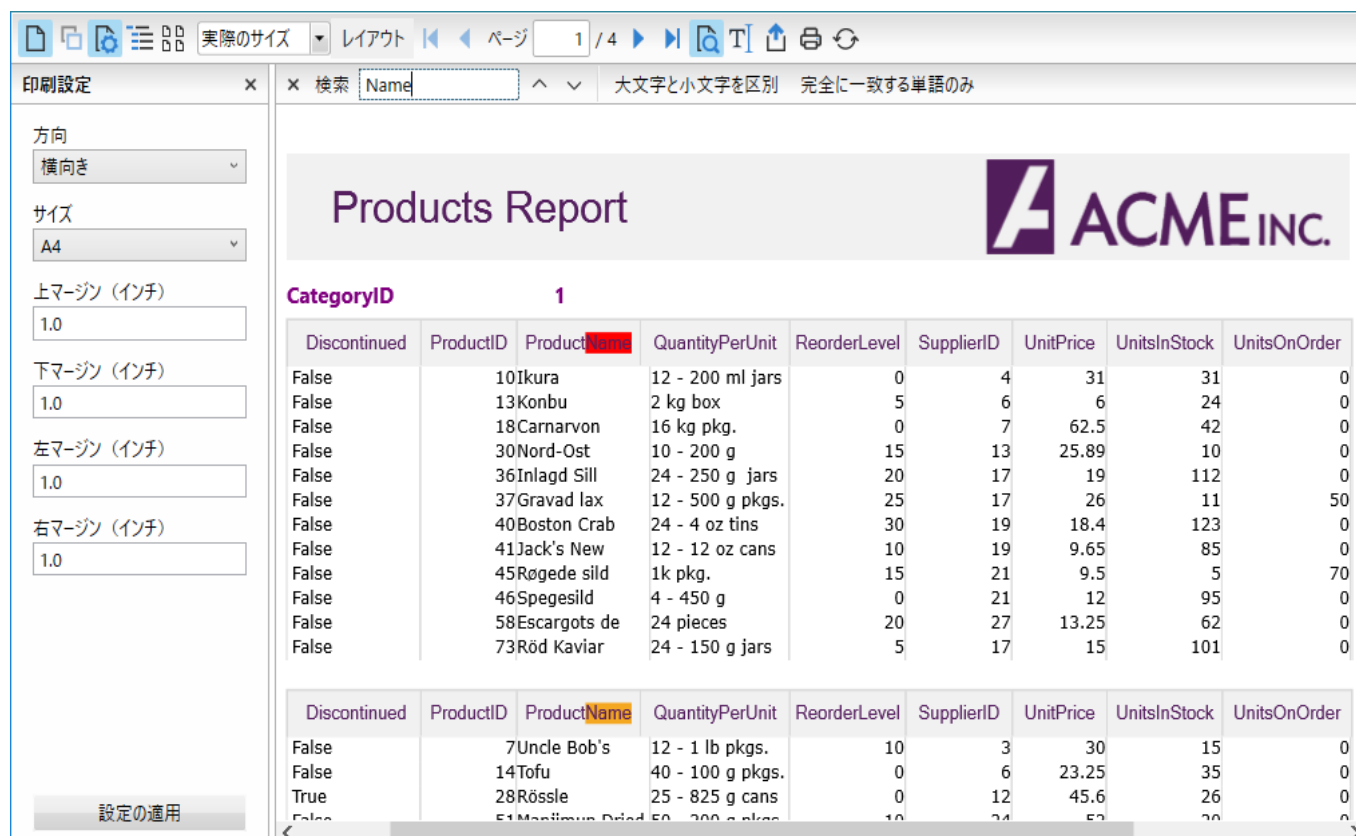
次に、FlexCommonTasks.flxr ファイルを開いたときに表示されるデザイナーを示します。



デザイナーウィンドウのコンポーネントの詳細については、「[WinForms 用の FlexReportDesigner アプリケーション](#)」を参照してください。

## FlexViewer for WPF

FlexViewer for WPFは、FlexReport レポートと SSRS レポートを表示するためのプレビューコントロールです。FlexViewer コントロールは、対話式の使いやすいユーザーインターフェース (UI) を備え、レポートページの簡単なナビゲーション、サムネイルの表示、パラメータの設定、ページ設定の変更に便利な使いやすいインターフェースを提供します。



The screenshot displays the FlexViewer interface for a 'Products Report'. The main area shows a table with columns: Discontinued, ProductID, ProductName, QuantityPerUnit, ReorderLevel, SupplierID, UnitPrice, UnitsInStock, and UnitsOnOrder. The report is for CategoryID 1. The sidebar on the left contains print settings such as direction (horizontal), size (A4), and margins (top, bottom, left, right, all set to 1.0). A search bar at the top allows filtering by product name.

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	10	Ikura	12 - 200 ml jars	0	4	31	31	0
False	13	Konbu	2 kg box	5	6	6	24	0
False	18	Carnarvon	16 kg pkg.	0	7	62.5	42	0
False	30	Nord-Ost	10 - 200 g	15	13	25.89	10	0
False	36	Inlagd Sill	24 - 250 g jars	20	17	19	112	0
False	37	Gravad lax	12 - 500 g pkgs.	25	17	26	11	50
False	40	Boston Crab	24 - 4 oz tins	30	19	18.4	123	0
False	41	Jack's New	12 - 12 oz cans	10	19	9.65	85	0
False	45	Rogede sild	1k pkg.	15	21	9.5	5	70
False	46	Spegesild	4 - 450 g	0	21	12	95	0
False	58	Escargots de	24 pieces	20	27	13.25	62	0
False	73	Röd Kaviar	24 - 150 g jars	5	17	15	101	0

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	7	Uncle Bob's	12 - 1 lb pkgs.	10	3	30	15	0
False	14	Tofu	40 - 100 g pkgs.	0	6	23.25	35	0
True	28	Rössle	25 - 825 g cans	0	12	45.6	26	0
False	51	Manjimun Dried	50 - 200 g pkgs	10	24	52	20	0

FlexViewer コントロール内からレポートを複数の形式に簡単にエクスポートできます。FlexViewer では、標準的な印刷設定を提供する[印刷]アイコンを使用して、レポートを印刷することもできます。

## 主な特長

FlexViewer の主な特長は次のとおりです。

### ● レポートのプレビュー

FlexViewer は、レポートプレビュー機能との最新のインターフェースを備えており、FlexReport レポートおよび SSRS レポートをロードして表示することができます。コントロールを対話式に使いやすくするツールバーも組み込まれています。

### ● ページナビゲーション

FlexViewer には、レポートページ間を移動しやすくするページナビゲーション機能が用意されています。1 回のクリックでレポートの前、次、最初、または最後のページに移動したり、目的のページ番号を入力しただけでそのページに直接ジャンプすることができます。

### ● ズームオプション

FlexViewer は、レポートのさまざまなズームオプションをサポートします。

- 実際のサイズ - 実際のサイズでページを表示します。
- ページ幅 - ページをプレビューウィンドウの幅に合わせます。
- ページ全体 - プレビューウィンドウにページ全体を合わせます。

## ● レイアウトオプション

FlexViewer は、レポートのさまざまなレイアウトオプションをサポートします。

- 時計回りに回転 - ビューを時計回りに回転します。
- 反時計回りに回転 - ビューを反時計回りに回転します。
- 1 ページ - レポートページを単一ページビューに 1 ページずつ表示します。
- 見開きページ - レポートページを並べて表示します。
- 2 ページ - 2 ページビューを表示します。
- 4 ページ - ページを 4x4 モードで表示します。

## ● パラメータの使用/リセット

FlexViewer コントロールは、レポートに表示する必要があるデータパラメータを入力することができ、対話性が向上しています。String、Boolean、Date、Integer、Float 型のパラメータをサポートします。

## ● サムネイルと階層の表示

FlexViewer は、レポートページのサムネイルビューを表示できます。レポートに見出しマップが含まれる場合は、FlexViewer にドキュメントアウトラインボタンが表示されます。ここから、ジャンプ先の選択に便利なアウトラインパネルを表示できます。

## ● ページ設定

FlexViewer コントロールを使用すると、レポートを印刷する前に、要件に応じてページ設定を変更できます。左パネルの[ページ設定]アイコンをクリックして、方向、サイズ、およびマージンを設定できます。

## ● 印刷

FlexViewer では、標準的な印刷設定を提供する[印刷]アイコンを使用して、レポートを印刷することもできます。

## ● エクスポート

FlexViewer を使用すると、HTML、PDF、RTF、GIF、JPEG、PNG、BMP、TIFF、Open XML Exce、Open XML Word などの形式にレポートやドキュメントをエクスポートできます。エクスポート後に、エクスポートしたドキュメントを自動的に開くように選択することもできます。

## ● RightToLeft

FlexViewer では、FlowDirection プロパティを使用して、FlexViewer ツールパネルの方向を右から左または左から右に設定できます。

## ● テキスト検索

FlexViewer では、ツールバーのテキスト検索ボタンを使用して、FlexReport 内のテキストを検索できます。テキスト検索ボタンは検索ボックスを開きます。

## ● テキスト選択

FlexViewer では、ツールバーのテキスト選択ツールボタンを使用して、コピーするテキストを選択できます。

## クイックスタート

クイックスタートガイドでは、FlexViewer コントロールでのレポートのレンダリングについて詳しく説明します。このセクションでは、FlexReport を **FlexViewer** コントロールにロードしてレンダリングする方法について説明します。

**FlexViewer** コントロールを使用して簡単な WPF アプリケーションを作成するには、次の手順を実行します。

1. レポートのロード
2. レポートのレンダリング

### 手順 1: レポートのロード

ファイルからレポート定義をロードするには、次の手順を実行します。

# FlexReport for WPF

1. Visual Studio で新しい **WPF アプリケーション** を作成します。
2. **XAML** デザインに **Button** コントロールと **C1FlexViewer** コントロールを追加します。**C1FlexViewer** コントロールの名前を **Viewer** に設定し、その高さと幅を調整します。
3. プロジェクトに FlexReport を追加します。ここでは、コンピュータにインストールされている FlexReport サンプルにある TelephoneBillReport.flxr という名前のレポートを使用しました。
4. **C1.WPF.FlexReport.4.dll** への参照をアプリケーションに追加します。
5. コードで、次の名前空間を追加します。
  - C1.WPF.FlexReport
6. コードビューで **Button\_Click** イベント内に次のコードを追加します。

- **Visual Basic**

```
Dim rep As New C1FlexReport()  
'レポート定義をロードします  
rep.Load("../..\TelephoneBillReport.flxr", "TelephoneBill")
```

- **C#**

```
//レポート定義をロードします  
C1FlexReport rep = new C1FlexReport();  
rep.Load(@"..\..\TelephoneBillReport.flxr", "TelephoneBill");
```

## 先頭に戻る

### 手順 2: レポートのレンダリング

レポートをレンダリングするには、最初にレポートをロードする必要があります。レポート定義の作成、データソースの定義、レポート定義のロードが終わったら、レポートを **FlexViewer** コントロールでレンダリングできます。

コードビューで **Button\_Click** イベントに次のコードを追加して、FlexViewer コントロールで FlexReport をレンダリングします。

- **Visual Basic**

```
'レポートをプレビューします  
Viewer.DocumentSource = rep
```

- **C#**

```
//レポートをプレビューします  
Viewer.DocumentSource = rep;
```

FlexViewer にレンダリングされて表示されたレポートを次に示します。

ACME INC.

Tim Richard  
120 Hanover Sq., Victoria Lane, Near Xamine Store  
Pittsburg, Pennsylvania

Page: 1 of 1  
Date: Jun 22, 2016  
Bill Cycle Date: 12/27/2010 - 01/26/2011  
Account: 125345871322147014

### Monthly Statement

Bill-At-A-Glance	
Previous Balance	\$228.33
Payment - Thank You	\$189.00
Adjustments	\$39.00
Balance	\$0.00
New Charges	\$332.07
<b>Total Amount Due</b>	<b>\$332.00</b>
Amount Due in Full by	10/2/2011

Payments & Adjustments			
S.No	Description		
1.	Payment posted	5/15/2016	\$189.00
2.	Service Charges		\$332.07
3.	Adjustments		\$39.00
<b>Total Payments &amp; Adjustments</b>			<b>\$332.00</b>

**Service Summary**

**Wireless**

Group 1 Usage Summary -  
FamilyTalk Nation 850 w/Rollover Minutes - \$9.99  
Each Additional Line 850 Shared Anytime Minutes  
with Rollover. Nationwide Long Distance &

先頭に戻る

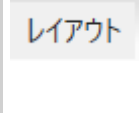
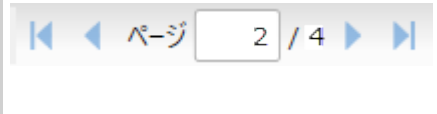





## FlexViewerツールバー

ツールバーは、FlexViewer コントロールの上部に表示されます。



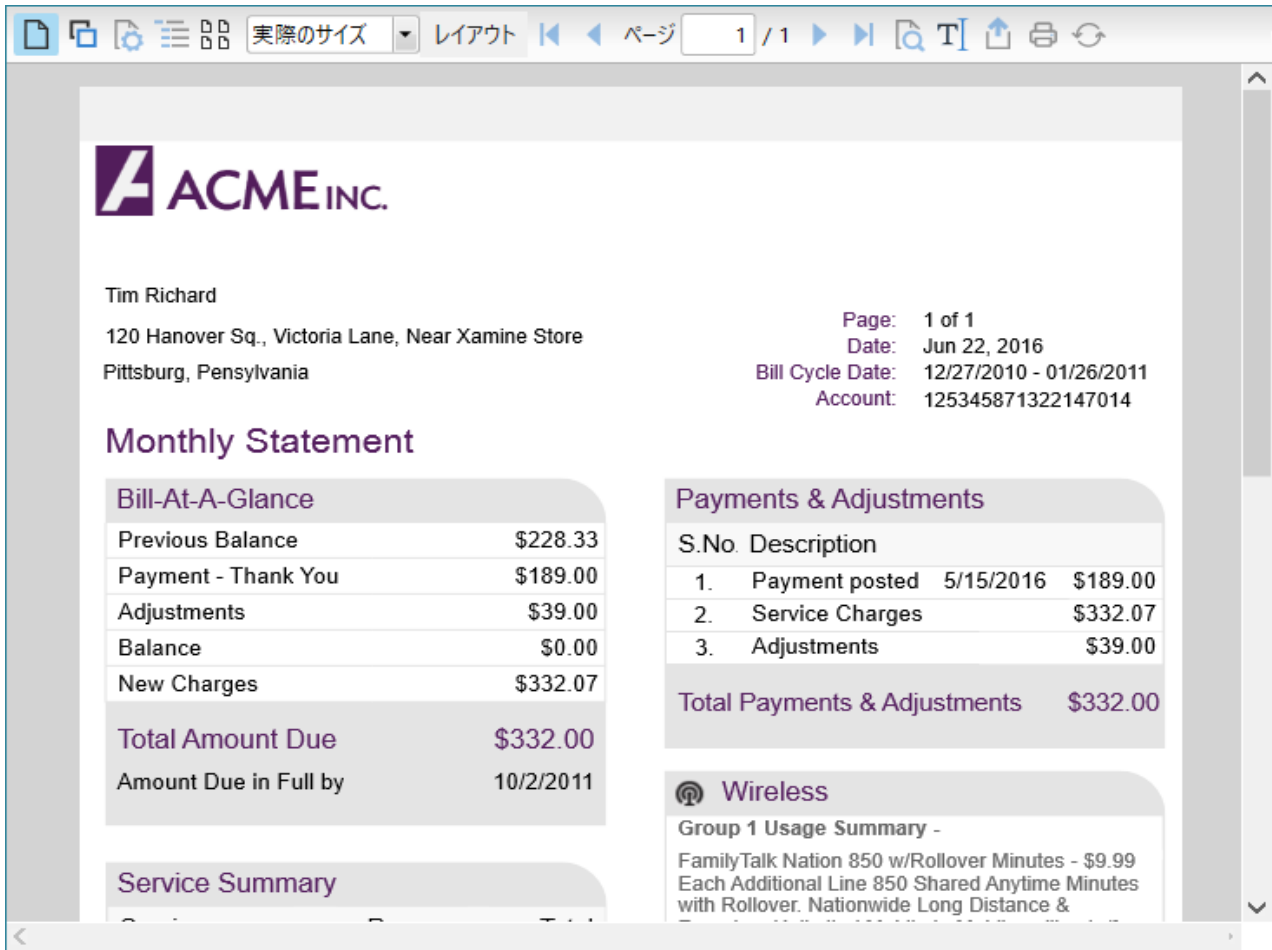
ツールバーには次のコマンドボタンがあります。

コマンドボタン	コマンドボタン名	説明
	印刷レイアウト	印刷ページに表示されるとおりにレポートページレイアウトを表示します。
	レポートのパラメータ	レポートページのパラメータセットを表示します。
	印刷設定	レポートページの方向、サイズ、マージンを設定できます。
	ドキュメントアウトライン	レポートページのアウトラインを表示します。
	ページサムネイル	表示されたレポートにあるすべてのページのサムネイルを表示します。
	ズームオプション	レポートページをズームするためのさまざまなオプションを表示します。

	レイアウトオプション	レポートページのさまざまなレイアウトを選択することができます。
	ページナビゲーション	レポートの最初、最後、前、次のページに移動できます。ページ番号テキストボックスに目的のページ番号を入力することで、特定のページに直接移動できます。
	テキスト検索	レポート内のテキストを検索できます。
	テキスト選択ツール	コピーするテキストを選択します。
	ファイルにエクスポート	さまざまな形式にレポートをエクスポートできます。
	印刷	レポートを印刷できます。
	更新	レポートページをリフレッシュします。

## FlexReport と FlexViewer の連結

FlexReport を FlexViewer と連結する目的は、FlexViewer コントロール内で FlexReport をプレビューできることです。FlexReport を FlexViewer コントロールと連結するには、まずレポート定義(.flxr)を作成し、それを C1FlexReport オブジェクトにロードする必要があります。レポート定義を **C1FlexReport** にロードしたら、それを FlexViewer コントロール内で表示することができます。**C1FlexViewer** では、**C1FlexViewer** クラスの **DocumentSource** プロパティを使用して、FlexReport を FlexViewer コントロールと連結することができます。**DocumentSource** プロパティは、**C1FlexReport** オブジェクトからレポート定義が入った値を受け取ります。



FlexViewer コントロールでレポートをプレビューするには、次のコードを使用します。

#### Visual Basic

```
'レポート定義をロードします
FlexReport1.Load("..¥..¥Products Report.flxr", "Products Report")
'レポートをプレビューします
Viewer.DocumentSource = FlexReport1
```

#### C#

```
//レポート定義をロードします
FlexReport1.Load(@"..¥..¥Products Report.flxr", "Products Report");
//レポートをプレビューします
Viewer.DocumentSource = FlexReport1;
```

## FlexViewer の機能

FlexViewer の機能セクションでは、FlexViewer コントロールが備えるすべての機能について説明します。

### レポートプレビューのズーム

XAML、プログラム、および実行時にズーム機能を実装する方法について説明します。

### レポートのビューの回転

XAML、プログラム、および実行時にビューの回転機能を実装する方法について説明します。

### FlexViewer を使用したレポートのエクスポート

実行時に FlexViewer を使用してレポートをエクスポートする方法について説明します。

# FlexReport for WPF

## FlexViewer を使用したレポートの印刷

実行時に FlexViewer を使用してレポートを印刷する方法について説明します。

## レポートプレビューのズーム

FlexViewer では、ツールバーにズームオプションドロップダウンメニューが用意されています。ここで、実際のサイズ、ページ幅、ページ全体、ズームパーセンテージレベルなどのさまざまなズームオプションを設定することができます。

次の図に、ズームモードを WholePage に設定したレポートを示します。

The screenshot shows a software interface for viewing reports. At the top, there is a toolbar with icons for file operations, a dropdown menu for page width, a layout button, and navigation controls. The main content area displays a report titled "Products Report" for "ACME INC.". The report includes a table with the following data:

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	10	Ikura	12 - 200 ml jars	0	4	31	31	0
False	13	Konbu	2 kg box	5	6	6	24	0
False	18	Camarvon	16 kg pkg.	0	7	62.5	42	0
False	30	Nord-Ost	10 - 200 g	15	13	25.89	10	0
False	36	Inlagd Sill	24 - 250 g jars	20	17	19	112	0
False	37	Gravad lax	12 - 500 g pkgs.	25	17	26	11	50
False	40	Boston Crab	24 - 4 oz tins	30	19	18.4	123	0
False	41	Jack's New	12 - 12 oz cans	10	19	9.65	85	0
False	45	Røgede sild	1k pkg.	15	21	9.5	5	70
False	46	Spegesild	4 - 450 g	0	21	12	95	0
False	58	Escargots de	24 pieces	20	27	13.25	62	0
False	73	Röd Kaviar	24 - 150 g jars	5	17	15	101	0

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	7	Uncle Bob's	12 - 1 lb pkgs.	10	3	30	15	0
False	14	Tofu	40 - 100 g pkgs.	0	6	23.25	35	0
True	28	Rössle	25 - 825 g cans	0	12	45.6	26	0
False	51	Manjimup Dried	50 - 300 g pkgs.	10	24	53	20	0
False	74	Longlife Tofu	5 kg pkg.	5	4	10	4	20

FlexViewer コントロールに表示されるレポートのズームレベルを設定するには、C1FlexViewer の **ZoomFactor** プロパティを使用します。**ZoomFactor** プロパティは、レポートのズームレベルを設定する浮動小数点値を受け取ります。FlexViewer では、**MinZoomFactor** プロパティと **MaxZoomFactor** プロパティを使用して、表示されるレポートのズームレベルを制限することもできます。

さらに、FlexViewer では、プレビューされたレポートのズーム動作を管理することができます。プレビューズームモードを指定するには、**ZoomMode** プロパティを使用します。**ZoomMode** プロパティは、FlexViewerZoomMode 列挙の値を受け取ります。

### XAML の場合

**ZoomMode** プロパティの値は、XAML とコードで設定することができます。ズームモードを設定するには、XAML で次のコードを使用します。

XAML

```
<c1:C1FlexViewer x:Name="Viewer" Grid.Row="1" ZoomMode="PageWidth"/>
```



## コードの場合

コードでプレビューズームモードを設定するには、**ZoomMode** プロパティと **FlexViewerZoomMode** 列挙を使用します。次のコードは、**ZoomMode** プロパティと **FlexViewerZoomMode** 列挙の使用方法を示します。この例では、「クイックスタート」で作成したサンプルを使用します。

Visual Basic

```
Viewer.ZoomMode = FlexViewerZoomMode.PageWidth
```

- C#

//ズームモードを設定します

```
Viewer.ZoomMode = FlexViewerZoomMode.PageWidth;
```

## レポートのビューの回転

**FlexViewer** では、要件に合わせて柔軟に、さまざまな角度にレポートビューを回転させることができます。さまざまな回転角度にレポートのビューを回転させるには、**C1FlexViewer** クラスの **RotateView** プロパティを設定します。**RotateView** プロパティは、ビューの回転角度を記述する **FlexViewerRotateView** 列挙の値を受け取ります。

次の図に、時計回りに 180 度回転させたレポートを示します。

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
True	9	Mishi Kobe Niku	18 - 500 g pkgs.	0	4	97	29	0
False	74	Longlife Tofu	5 kg pkg.	5	4	10	4	20
False	51	Manjimup Dried	50 - 300 g pkgs.	10	24	53	20	0
True	28	Rossie	25 - 825 g cans	0	12	45.6	26	0
False	14	Tofu	40 - 100 g pkgs.	0	6	23.25	35	0
False	7	Uncle Bob's	12 - 1 lb pkgs.	10	3	30	15	0
False	73	Rod Kaval	24 - 150 g jars	5	17	15	101	0
False	58	Escargots de	24 pieces	20	27	13.25	62	0
False	46	Spegestd	4 - 450 g	0	21	12	95	0
False	45	Rogede sild	1k pkg.	15	21	9.5	5	70
False	41	Jack's New	12 - 12 oz cans	10	19	9.65	85	0
False	40	Boston Crab	24 - 4 oz tins	30	19	18.4	123	0
False	37	Gravd lax	12 - 500 g pkgs.	25	17	26	11	50
False	36	Inlagd Sill	24 - 250 g jars	20	17	19	112	0
False	30	Nord-Ost	10 - 200 g	15	13	25.89	10	0
False	18	Cararvon	16 kg pkg.	0	7	62.5	42	0
False	13	Konbu	2 kg box	5	6	24	24	0
False	10	Kura	12 - 200 ml jars	0	4	31	31	0

# FlexReport for WPF

## XAML の場合

XAML ビューでレポートのビューを回転させるには、次のコードに示すように **RotateView** プロパティを使用します。ここでは、FlexReport の RotateView を 180 度に設定しています。

XAML

```
<c1:C1FlexViewer x:Name="Viewer" HorizontalAlignment="Left" VerticalAlignment="Top" Height="580" Width="782" Margin="0,30,0,0" RotateView="Rotation180"/>
```

## コードの場合

次のコードは、**FlexViewerRotateView** 列挙を使用して、レポートのビューを時計回りに 90 度回転します。この例では、「FlexReport クイックスタート」で作成したサンプルを使用します。

Visual Basic

'ビューの回転を設定します

```
Viewer.RotateView = FlexViewerRotateView.Rotation180
```

- C#

//ビューの回転を設定します

```
Viewer.RotateView = FlexViewerRotateView.Rotation180;
```

同様に、レポートのビューを反時計回りに 90 度および 180 度回転することができます。

## FlexViewerを使用したレポートのエクスポート

FlexViewer にはエクスポート機能が用意されており、レポートを印刷する代わりにデジタルファイルとして同僚やクライアントに送信することができます。FlexViewer コントロールでは、ツールバーの[ファイルに保存]コマンドボタンを使用してレポートをエクスポートできます。[ファイルに保存]コマンドボタンをクリックすると、[エクスポート]ダイアログが表示されます。ここから、PDF、HTML、RTF、Microsoft Excel Open XML、Open XML Word、ZIP、TIFF、BMP、PNG、JPEG、GIF などのさまざまな形式にレポートをエクスポートすることができます。この機能は、**C1FlexViewer** クラスの **ExportCommand** プロパティを使用して実現されます。

次の図は、[ファイルに保存]コマンドボタンをクリックして FlexReport をエクスポートしたときに表示されるレポートです。

Products Report

ACME INC.

CategoryID 1

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	10	Ikura	12 - 200 ml jars	0	4	31	31	0
False	13	Konbu	2 kg box	5	6	6	24	0
False	18	Carnarvon	16 kg pkg.	0	7	62.5	42	0
False	30	Nord-Ost	10 - 200 g	15	13	25.89	10	0
False	36	Inlagd Sill	24 - 250 g jars	20	17	19	112	0
False	37	Gravad lax	12 - 500 g pkgs.	25	17	26	11	50
False	40	Boston Crab	24 - 4 oz tins	30	19	18.4	123	0
False	41	Jack's New	12 - 12 oz cans	10	19	9.65	85	0
False	45	Røgedede sild	1k pkg.	15	21	9.5	5	70
False	46	Spegesild	4 - 450 g	0	21	12	95	0
False	58	Escargots de	24 pieces	20	27	13.25	62	0
False	73	Rød Kaviar	24 - 150 g jars	5	17	15	101	0

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	7	Uncle Bob's	12 - 1 lb pkgs.	10	3	30	15	0
False	14	Tofu	40 - 100 g pkgs.	0	6	23.25	35	0

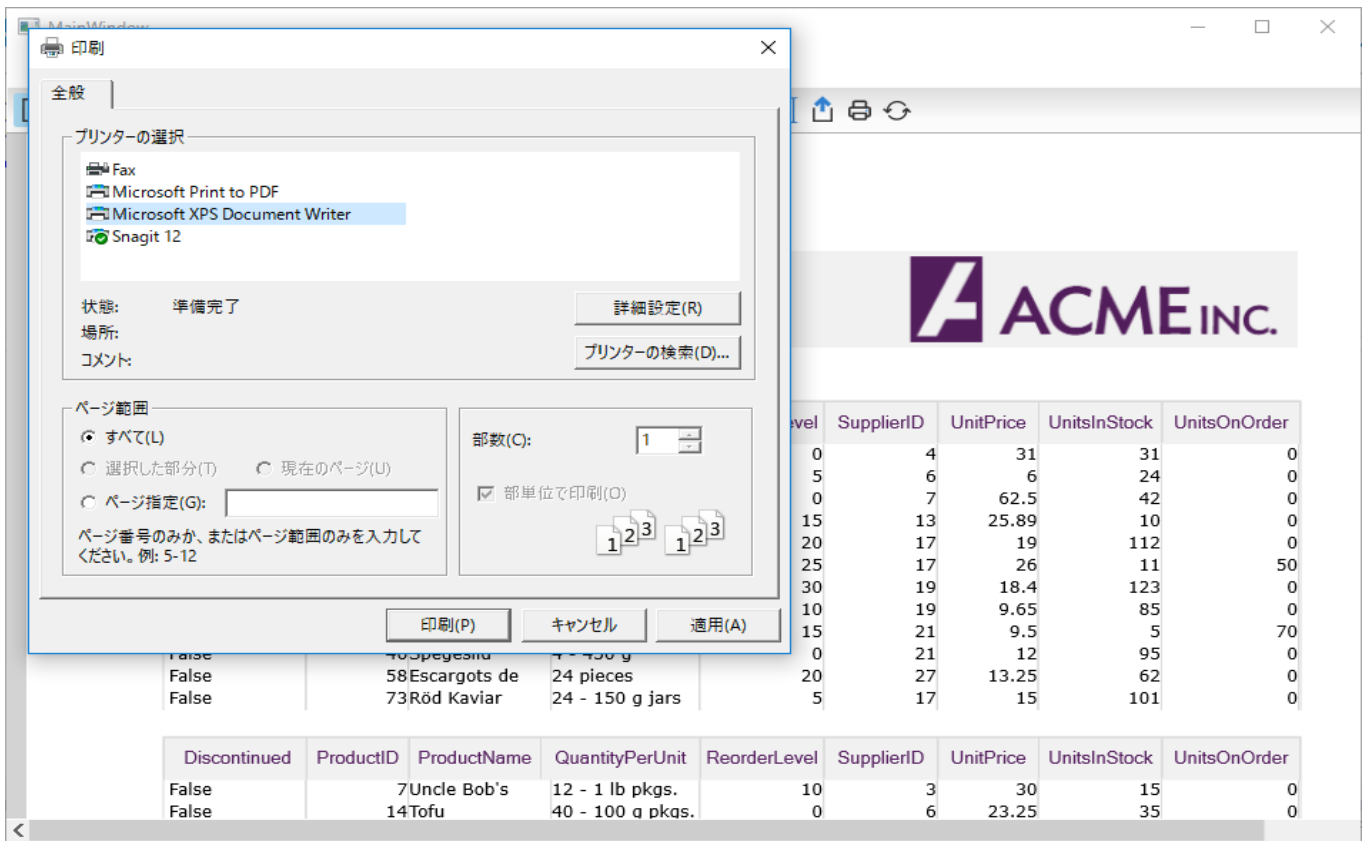
サポートされる形式と、コードによってレポートをエクスポートする方法の詳細については、「[レポートのエクスポート](#)」を参照してください。

## FlexViewerを使用したレポートの印刷

FlexViewer では、FlexViewer ツールバーの[印刷]コマンドボタンを使用して、FlexReport をプレビューまたは印刷することができます。[印刷]コマンドボタンをクリックすると、[印刷]ダイアログが表示されます。ここで、プリンタの設定を指定することができます。この機能は、**C1FlexViewer** クラスの **PrintCommand** プロパティを使用して実現されます。

次の図に、[印刷]コマンドボタンをクリックしたときに表示される[印刷]ダイアログを示します。

# FlexReport for WPF



コードによる FlexReport の印刷の詳細については、「[FlexReport の印刷](#)」を参照してください。

## FlexViewerPane

FlexReport for WPF には、レポートプレビュー機能を備えた FlexViewerPane コントロールがあります。FlexViewerPane コントロールは、FlexReport と SSRS レポートの表示に使用できるプレビューコントロールです。さらに、FlexViewerPane では、実際のサイズ、ページ幅などのズーム設定、サムネイルの表示、レポートビューの回転角度の指定、見開きページを並べて表示などを行うことができます。FlexViewerPane コントロールは、印刷もサポートします。

Products report

ACME INC.

CategoryID 1

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	10	Ikura	12 - 200 ml jars	0	4	31	31	0
False	13	Konbu	2 kg box	5	6	6	24	0
False	18	Carnarvon	16 kg pkg.	0	7	62.5	42	0
False	30	Nord-Ost	10 - 200 g	15	13	25.89	10	0
False	36	Inlagd Sill	24 - 250 g jars	20	17	19	112	0
False	37	Gravad lax	12 - 500 g pkgs.	25	17	26	11	50
False	40	Boston Crab	24 - 4 oz tins	30	19	18.4	123	0
False	41	Jack's New	12 - 12 oz cans	10	19	9.65	85	0
False	45	Røgede sild	1k pkg.	15	21	9.5	5	70
False	46	Spegesild	4 - 450 g	0	21	12	95	0
False	58	Escargots de	24 pieces	20	27	13.25	62	0
False	73	Röd Kaviar	24 - 150 g jars	5	17	15	101	0

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	7	Uncle Bob's	12 - 1 lb pkgs.	10	3	30	15	0
False	14	Tofu	40 - 100 g pkgs.	0	6	23.25	35	0
True	28	Rössle	25 - 825 g cans	0	12	45.6	26	0

## FlexReport と FlexViewerPane の連結

FlexReport を FlexViewerPane と連結するには、まずレポートをロードする必要があります。レポート定義を作成して **C1FlexReport** にロードしたら、**C1FlexViewer** コントロールを XAML デザイナに追加し、**C1FlexViewerPane** クラスの **DocumentSource** プロパティを使用してレポートをコントロールに連結できます。

FlexReport を FlexViewerPane と連結するには、次のコードを使用して、**C1FlexViewerPane** コントロールの名前を **ViewerPane** に設定します。

Visual Basic

```
Dim rep As New C1FlexReport()
'レポート定義をロードします
rep.Load("../¥..¥Products Report.flxr", "Products Report")
'レポートをプレビューします
ViewerPane.DocumentSource = rep
```

C#

```
C1FlexReport rep = new C1FlexReport();  
//レポート定義をロードします  
rep.Load(@"..\¥..¥Products Report.flxr", "Products Report");  
//レポートをプレビューします  
ViewerPane.DocumentSource=rep;
```

## レポートプレビューのズーム

**C1FlexViewerPane** には、**FlexViewerPane** コントロールに表示されるレポートのズームレベルを設定するための **ZoomFactor** プロパティが用意されています。**ZoomFactor** プロパティに浮動小数点値を設定して、レポートのズームレベルを設定することができます。**MinZoomFactor** と **MaxZoomFactor** を設定して、表示されるレポートのズームレベルを制限することもできます。

さらに、**C1FlexViewerPane** では、**FlexViewerPane** コントロール内でプレビューされたレポートのズーム動作を管理することができます。プレビューズームモードを指定するには、**ZoomMode** プロパティを使用します。**ZoomMode** プロパティは、**FlexViewerZoomMode** 列挙の次の値を受け取ります。この列挙は、**FlexViewerPane** コントロールでサポートされるズームモードを記述しています。

- **実際のサイズ**: 実際のサイズでページを表示します。
- **カスタム**: カスタムズームモードを設定します。
- **ページ幅**: ページをプレビューウィンドウの幅に合わせます。
- **ページ全体**: プレビューウィンドウにページ全体を合わせます。

**ZoomMode** プロパティの値は、XAML とコードで設定することができます。ズームモードを設定するには、XAML の `<c1:C1FlexViewerPane>` `</c1:C1FlexViewerPane>` タグ内に次のコードを追加します。

XAML

```
<c1:C1FlexViewerPane x:Name="ViewerPane" Grid.Row="1" ZoomMode="WholePage"/>
```

コードでプレビューズームモードを設定するには、**ZoomMode** プロパティと **FlexViewerZoomMode** 列挙を使用します。次のコードは、**ZoomMode** プロパティと **FlexViewerZoomMode** 列挙の使用方法を示します。この例では、「[FlexReport クイックスタート](#)」で作成したサンプルを使用します。

- **Visual Basic**

```
ViewerPane.ZoomMode = FlexViewerZoomMode.WholePage
```

- **C#**

```
ViewerPane.ZoomMode = FlexViewerZoomMode.WholePage;
```

Products report		ACME INC.						
CategoryID		1						
Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
False	10	Kura	12 - 200 ml jars	0	4	31	31	0
False	13	Kanbu	2 kg box	5	6	6	24	0
False	18	Carnarvon	16 kg pkg.	0	7	62.5	42	0
False	30	Nord-Ost	10 - 200 g	15	13	25.89	10	0
False	36	Inlagd Sill	24 - 250 g jars	20	17	19	112	0
False	37	Gravad lax	12 - 500 g pkgs.	25	17	26	11	50
False	40	Boston Crab	24 - 4 oz tins	30	19	18.4	123	0
False	41	Jack's New	12 - 12 oz cans	10	19	9.65	85	0
False	45	Rogede sild	1k pkg.	15	21	9.5	5	70
False	46	Spegesild	4 - 450 g	0	21	12	95	0
False	58	Escargots de	24 pieces	20	27	13.25	62	0
False	73	Röd Kaviar	24 - 150 g jars	5	17	15	101	0
False	7	Uncle Bob's	12 - 1 lb pkgs.	10	3	30	15	0
False	14	Tofu	40 - 100 g pkgs.	0	6	23.25	35	0
True	28	Rössle	25 - 825 g cans	0	12	45.6	26	0
False	51	Manjimup Dried	50 - 300 g pkgs.	10	24	53	20	0
False	74	Longlife Tofu	5 kg pkg.	5	4	10	4	20
True	9	Mishi Kobe Niku	18 - 500 g pkgs.	0	4	97	29	0

2017/03/29 18:54:42 Page 1 of 4

## レポートのビューの回転

**FlexViewerPane** では、要件に合わせて柔軟に、さまざまな角度にレポートビューを回転させることができます。レポートビューをさまざまな角度に回転させるには、**C1FlexViewerPane** クラスの **RotateView** プロパティを設定します。**RotateView** プロパティは、ビューの回転角度を記述する **FlexViewerRotateView** 列挙に含まれる次の値を受け取ります。

- **NoRotation:** ビューに回転が適用されません。
- **Rotation180:** ビューを 180 度回転します。
- **Rotation90Clockwise:** ビューを時計回りに 90 度回転します。
- **Rotation90CounterClockwise:** ビューを反時計回りに 90 度回転します。

次のコードは、**FlexViewerRotateView** 列挙を使用して、レポートのビューを時計回りに 90 度回転します。この例では、「FlexReport クイックスタート」で作成したサンプルを使用します。

- **Visual Basic**

```
ViewerPane.RotateView = FlexViewerRotateView.Rotation90Clockwise
```

- **C#**

```
ViewerPane.RotateView = FlexViewerRotateView.Rotation90Clockwise;
```

## Products report

CategoryID 1

Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	Sup
False	10	Ikura	12 - 200 ml jars	0	
False	13	Konbu	2 kg box	5	
False	18	Cararvon	16 kg pkg.	0	
False	30	Nord-Ost	10 - 200 g	15	
False	36	Inlagd Sill	24 - 250 g jars	20	
False	37	Gravad lax	12 - 500 g pkgs.	25	
False	40	Boston Crab	24 - 4 oz tins	30	
False	41	Jack's New	12 - 12 oz cans	10	
False	45	Ragede sild	1k pkg.	15	
False	46	Spegesild	4 - 450 g	0	
False	58	Escargots de	24 pieces	20	
False	73	Röd Kavlar	24 - 150 g jars	5	
<b>Discontinued</b>					
Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	Sup
False	7	Uncle Bob's	12 - 1 lb pkgs.	10	
False	14	Tofu	40 - 100 g pkgs.	0	
True	28	Rössle	25 - 825 g cans	0	
False	51	Mantimup Dried	50 - 300 g pkgs.	10	
False	74	Longlife Tofu	5 kg pkg.	5	
<b>Discontinued</b>					
Discontinued	ProductID	ProductName	QuantityPerUnit	ReorderLevel	Sup
True	9	Mishi Kobe Niku	18 - 500 g pkgs.	0	

2017/03/29 18:57:03

同様に、レポートのビューを反時計回りに 90 度および 180 度回転することができます。