

SharpShooter Reports.WinRT (XAML)

ユーザーガイド

Last modified on: June 10, 2014

目次

はじめに.....	4
システムの必要条件.....	4
Sharp Shooter Reports.WinRT ビューアのユーザーインターフェイス	4
静的表示モード.....	4
フリップ表示モード.....	5
グリッド表示モード.....	6
ブックマークのツリー表示.....	8
「レポートなし」表示.....	9
エラー表示.....	10
SharpShooter Reports.WinRT ビューアの API	11
基本のプロパティやメソッド.....	11
最適化のメカニズム.....	13
ハイパーリンクのクリック処理.....	14
ドリル スルー.....	14
ナビゲーション履歴.....	15
レポートビューアのコマンド.....	15
パラメータ	17
一般情報.....	17
パラメータ入力 UI.....	18
コードによるパラメータの設定.....	19
パラメータ処理のカスタマイズ方法.....	20
エクスポート	26
標準のダイアログの使用.....	26
Export メソッドの使用.....	29
ExportToStream 非同期メソッドの使用.....	30
PDF や Open Xml Excel エクスポートのデフォルト設定の指定.....	32
エクスポート フィルタの一覧.....	32
外観のカスタマイズ	35
レポートビューアの構造.....	35
ViewerItemsControl のスタイル.....	36

コントロール、アイテム、プレゼンター	39
コントロール パネル	41
非表示のパネル	42
パラメータ パネル	42
サムネイル パネル	45
外観のカスタマイズ	46
アプリケーションのテーマ変更	48
レポートビューアのローカライズ	49

はじめに

SharpShooter Reports.WinRT は、Windows ストア アプリケーションにレポート機能を追加することができる XAML 対応コンポーネントです。WinRT のレポートビューアには、マウスやタッチ機能、検索、共有、設定チャームの統合をサポートする Windows 8 互換のインターフェイスがあります。

システムの必要条件

プロジェクトを開発する前に、以下をインストールしておく必要があります。

- Microsoft Visual Studio 2012 またはそれ以上
- サーバー側のプロジェクトには .NET Framework 3.5 またはそれ以上
クライアント側のプロジェクトには .NET 4.5 またはそれ以上
- ASP.NET 2.0 またはそれ以上
- SharpShooter Reports.WinRT 6.5 またはそれ以上
- Microsoft Windows 8

Sharp Shooter Reports.WinRT ビューアのユーザーインターフェイス

進行中の状態に応じて、ビューア ウィンドウはいずれかの表示モードで表示されます。各表示モードの説明とコントロールの機能は以下の通りです。

静的表示モード

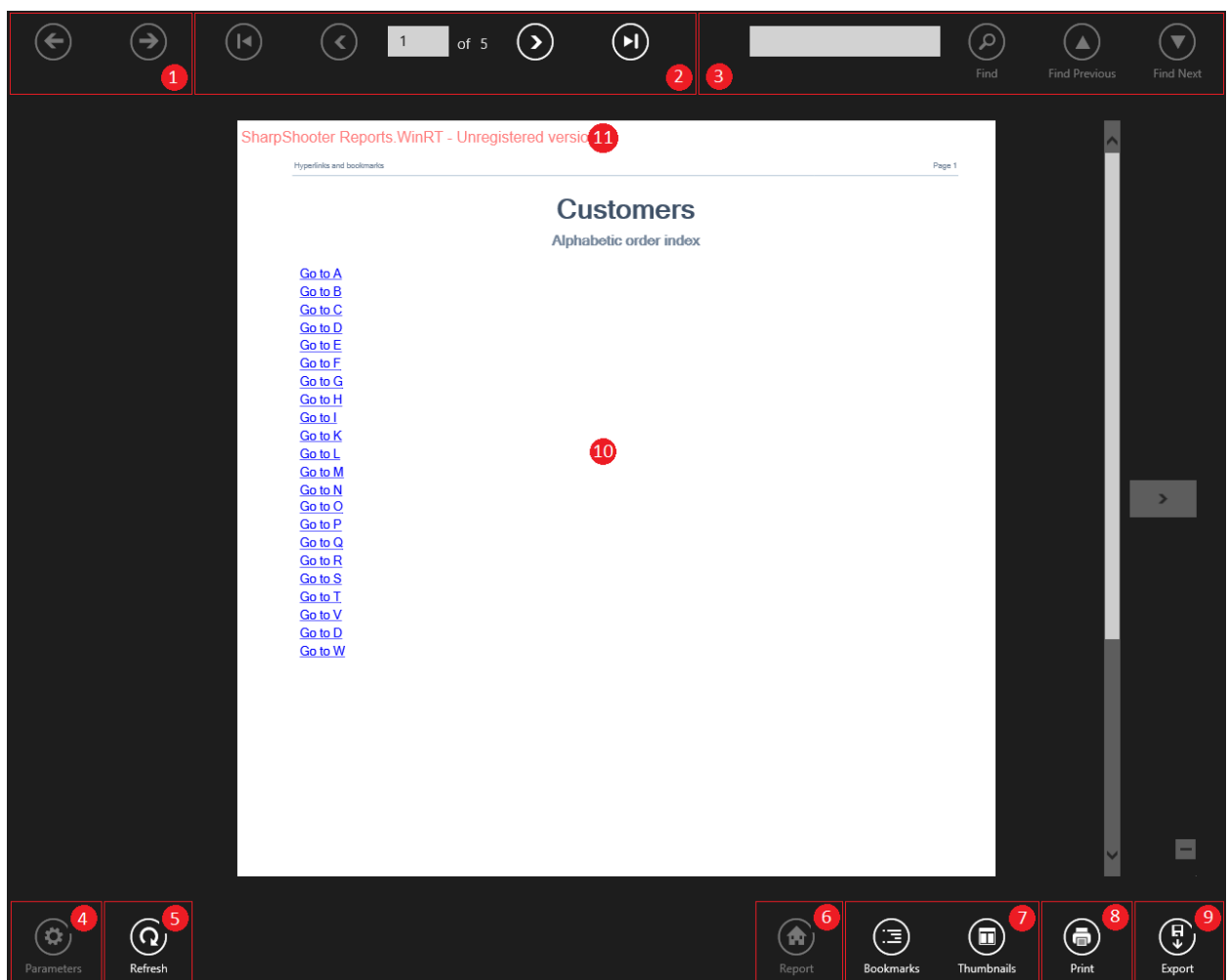


時間のかかる作業を処理する時にビューアにこのような画面が表示されます。この時、ビューアのコントロールボタンはどれも無効になります。

フリップ表示モード

ほとんどの場合、ユーザーはフリップ表示モードで作業しなければなりません。レポートはこのモードで表示され、主要なコントロールが含まれています。レポートの生成後、ビューアはこの表示モードに切り替えられて表示されます。また、グリッド表示モード（以下参照）の使用時にメイン表示に切り替えることもできます。ツールバーのアクションを使用するか、フリップ表示モードのレポートビューアを、グリッド表示モードに切り替わるまでズームアウトしてグリッド表示モードに切り替えることができます。

フリップ表示モードのメインウィンドウが表示されます。



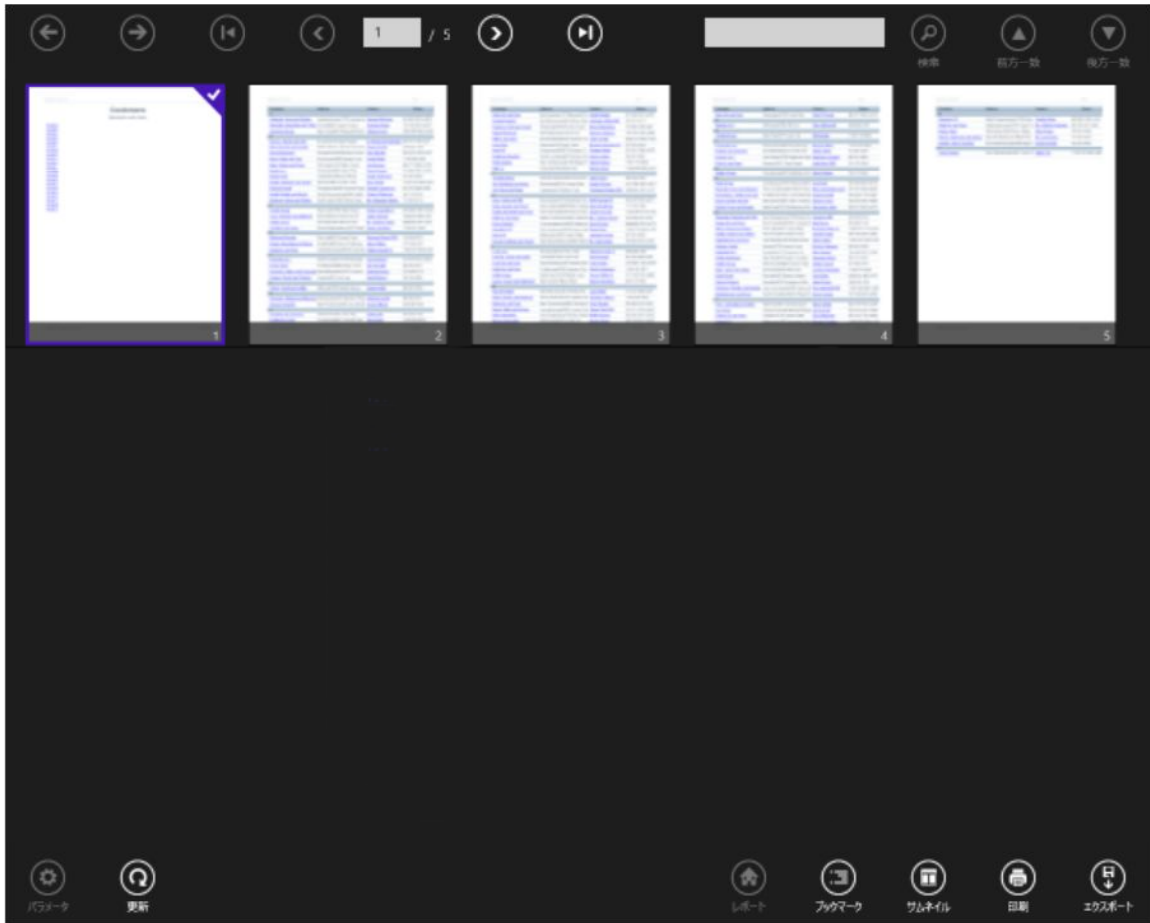
便宜上、コントロールは次のようにグループ化されています。

1. ナビゲーション履歴。相互参照付きの一連のレポートを使用して、次/前のレポートに移動できます。ナビゲーション履歴に関する詳細は「ナビゲーション履歴」を参照してください。

2. ページのナビゲーション。現在のページ、全ページを表示し、前/次/先頭/最終ページに移動できます。また、ご希望のページ番号を手で入力することもできます。
3. 検索。レポートのテキスト検索に使用します。別のセルに含まれているテキストの検索はサポートされておりませんのでご注意ください。
4. パラメータ。レポートに任意のパラメータがある場合に、入力パネルを表示します。レポートのパラメータ処理に関する詳細は、「パラメータ」を参照してください。
5. 更新。現在のレポートを更新します。
6. レポート表示。ブックマーク表示モードからレポート表示に切り替えることができます。
7. ブックマークとサムネイル。ブックマークボタンは、現在の表示からブックマーク表示モードに切り替えるために使用します（「ブックマークのツリー表示」を参照してください）。サムネイルボタンは、サムネイルパネルを開いて、レポートの特定ページに瞬時に移動できます（「グリッド表示モード」を参照してください）。
8. 印刷。印刷オプション（用紙サイズ、印刷枚数、ご使用のプリンタに応じたその他のオプション）を設定してレポートを印刷できます。
9. エクスポート。「エクスポート」を参照してください。
10. 前/次のレポートに移動するためのコントロールが付いたレポート表示領域。レポートをズームイン/ズームアウトするには、Windows の標準のナビゲーションのジェスチャーまたは Ctrl + マウスホイールのどちらかを使用できます。
11. 体験版をご利用の場合：レポート上部に「未登録バージョン」と表示されます。

グリッド表示モード

グリッド表示モードには、フリップ表示モードと同じコントロールがあります。基本的には、レポートページのサムネイルの構成が違います。ページのサムネイルは、レポートページの小型のプレビューで、瞬時に選択ページに移動できます。サムネイルをタップすると、フリップ表示モードに切り替わります。



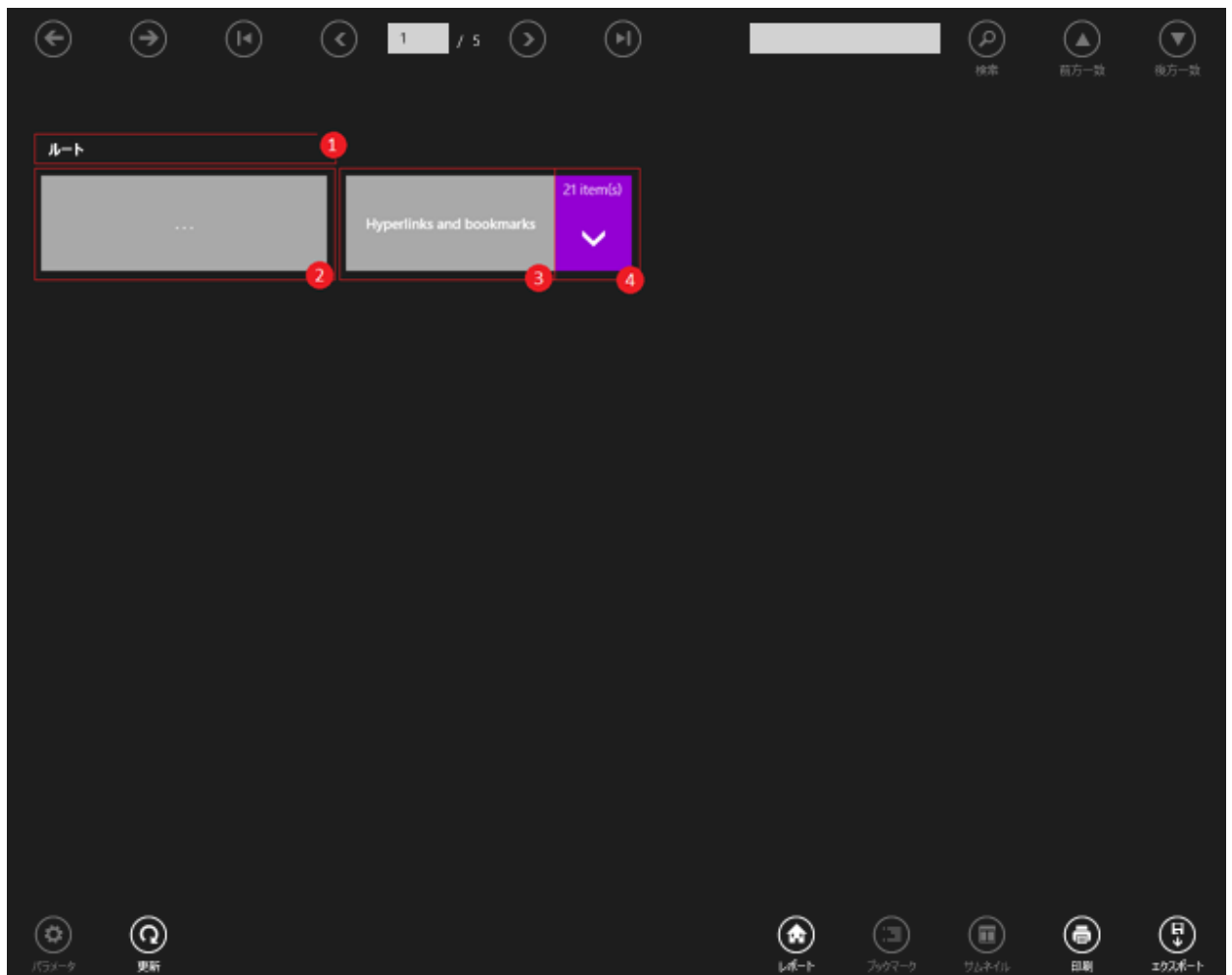
ブックマークのツリー表示

ツールバー下部にあるブックマークボタンを使用して、ブックマークのツリー表示に切り替えます。最初の表示モードに戻すには、ご希望の表示モードに切り替えるボタンを押すか、1つ上に移動するための専用領域（②を参照）を使用できます。

ブックマークのツリー表示はフォルダベースです。ブックマークのツリー表示に切り替える場合、上位のフォルダが表示されます。

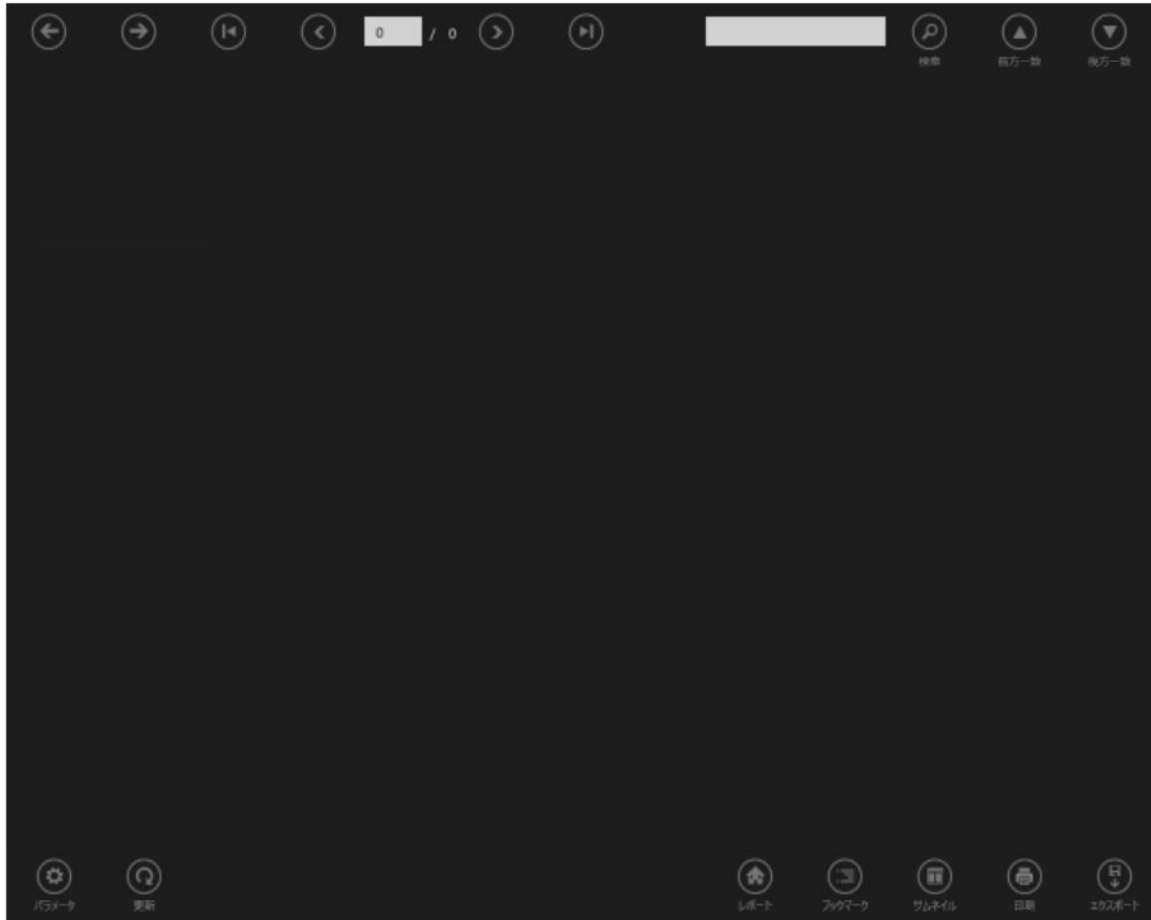
簡単にナビゲートするために、現在のブックマークフォルダのパスが上部に表示されます（①を参照）。要素をクリックしてレベルを変更できます。

ブックマークリストの左上端に、前に移動するボタンが常に表示されています。その他のブックマークのボタンはその隣にあります。ブックマークに進むには、ブックマークフィールドの左側（③を参照）を押してください。入れ子のブックマークリストに切り替えるには、ブックマークフィールドの右側に表示されている下向きの矢印（④を参照）を押してください。..



「レポートなし」表示

ビューアが起動してもレポート生成が始まらない場合、ビューアは下図のような表示になります。コンテンツ領域には何も表示されず、ボタンはすべて無効となります。



エラー表示

これは、レポート生成中に何らかのエラーが起きた場合に表示される特別な表示モードです。
DebugMode プロパティの値に応じて、ビューアにさまざまなエラーメッセージの詳細が表示されます。



SharpShooter Reports.WinRT ビューアの API

基本のプロパティやメソッド

プロパティ

- **ReportName** プロパティ：デフォルト値は空です。
- 生成するレポート名を設定します。WCF Report Service の ReportManager に指定した名前のレポートが登録されるはずですが、生成を成功させるにはこの値は必須です。
- **ServiceUrl** プロパティ：デフォルト値は空です。生成を成功させるにはこの値は必須です。WCF Report Service の完全パスを設定します。

構文は次の通りです。

<プロトコル>://<サーバー>:<ポート>/<パス>/<サービス名>.svc

URL 属性:

<プロトコル>-使用するプロトコル。(SSL 使用時は) **http** または **https**。値は必須です。

<サーバー>-配置されたレポートサービスを持つサーバー名。値は必須です。

<ポート>-レポートサービスで使用するポート。デフォルトのポート (**http** の場合は **80**、**https** の場合は **443**) を使用する場合はポートの指定を省略できます。

<パス>-レポートサービスのパス。レポートサービスがサイト/アプリケーションのルートにある場合は値を空にできます。

<サービス名>-レポートサービス名。値は必須です。

URL の正しい例 :

<http://localhost:5555/ReportService.svc>

<https://www.MyServer.com/Services/MyReportService.svc>

レポートサービスのサービス名や設定が正しく設定されていれば、サーバーアプリケーションを実行して指定したサービスの URL にアクセスするとウェブブラウザに **svc** サービスの **welcome** ページが表示されます。



- **DebugMode** プロパティ：デフォルト値は「Simple」です。エラーメッセージの詳細レベルを指定します。

値は以下の通りです。

値	説明
Simple	エラーが起きたことを伝えます。詳細は表示されません。
Message	エラーメッセージが表示されます。
Full	(取得可能な場合、エラーメッセージやスタックトレースを含む) 全エラー情報が表示されます。

- **Parameters** プロパティ：レポートパラメータを設定できます（「パラメータ」を参照）
- **ExportMode** プロパティ：デフォルト値は「SaveToFile」です。エクスポートモードを設定できます。値は以下の通りです。

値	説明
UserSelected	エクスポートする時に2つのオプション：共有メカニズムを使用する/ファイルに保存する、のどちらかを選択できます。
Share	エクスポート結果がエクスポート後の処理のために共有チャームに渡されます。エクスポート結果を取得するアプリケーションを選択する必要があります。
SaveToFile	エクスポート後にファイルに保存されます。保存するファイルの種類を選択する必要があります。

詳細は、「エクスポート」を参照してください。

- **HideDefaultToolbars** プロパティ：デフォルトのツールバーを非表示にできます。インターフェイスのカスタマイズに使用できます。カスタマイズに関する詳細は、「外観のカスタマイズ」を参照してください。
- **RestoreBookmarks** プロパティ：ブックマークのツリーを再表示する時に、ビューアに最後に表示したページを開くかを定義する値を設定します。設定した値が「false」の場合、ルートブックマークフォルダが使用されます。

メソッド

- **RenderDocument()** メソッド：レポートの生成を開始します。生成を成功させるには、**ReportName** プロパティと **ServiceUrl** プロパティの値の設定が不可欠です。
- **RenderReport(HistoryOptions historyOption)** メソッド：動作は **RenderDocument** メソッドと同じですが、ナビゲーション履歴を追加できます。
- **Print()** メソッド：現在のレポートを印刷します。
- **Export(string exportFormat, ExportMode mode)** メソッド：現在のレポートを指定した形式 (*exportFormat*) にエクスポートし、指定したモード (*mode* : このメソッドで **UserSelected** の使用は不可) に応じてエクスポート結果を処理します。

イベント

- **DocumentLoaded** イベント：ドキュメントの読込後に発生します。ドキュメントへのリンクはイベントの引数から取得できます。このイベントが発生した時には、まだドキュメントのページコンテンツはありません。コンテンツの読込が始まると、**PageLoaded** イベントが発生します。
- **PageLoaded** イベント：ページコンテンツの取得後に発生します。（コンテンツが既に読み込まれている）ドキュメントへのリンクと読み込まれたページ数はイベントの引数から取得できます。
- **HandlingError** イベント：エラーが生じると必ず発生します。この場合、レポートビューアはエラー表示に切り替わります。
- **ExportStarted** イベント：エクスポートの開始時に発生します。エクスポート中にトースト通知（トースト通知は **SaveToFile** エクスポート値が設定されている時にデフォルトで使用され、**Share** 値が選択された場合は表示されません）が必要な場合に、イベントの引数に「**ShowToastNotification = True**」を設定できます。また、**ExportEnded** イベントに渡される **State** 値を設定することもできます。
- **ExportEnded** イベント：エクスポート処理が終わると発生します。**ExportStarted** イベントに設定された **State** 値を引数に渡すことができます。
- **PrintStarted** イベント：印刷開始前に発生します。
- **PrintEnded** イベント：印刷終了後に発生します。
- **HyperlinkClick** イベント：ハイパーリンクを含んだ要素をクリックすると発生します。

最適化のメカニズム

これらのメカニズムは、事前読込機能を使用してページを切り替える時に、ユーザーには見えない隣り合わせのページを削除することで、大量のレポートを表示する時のメモリの使用量を減らし、ページの読込時間を短縮できます。

これらのメカニズムは、次のプロパティを使用して設定されます。

- **PreloadedPageCount** プロパティ：ページの切り替え時間短縮のために、現在ページの前後に事前に読み込むページ数を指定します。
- **UsePageBuffer** プロパティ：メモリに読み込むページ数を制限するか定義します。
- **PageBufferLengthAfter** プロパティ：メモリから解放されない、現在ページ以降のページ数を指定します。
- **PageBufferLengthBefore** プロパティ：メモリから解放されない、現在ページ以前のページ数を指定します。

ハイパーリンクのクリック処理

レポートビューアは、次の3つのハイパーリンクをサポートします。

- ブックマーク：あるレポート要素へのリンク。この種のリンクは、ビューアで自動処理されます。
- 標準の URL のハイパーリンク：リソースを開くためにシステムに渡されます。例えば、`http` リンクはウェブブラウザ、`mailto` リンクはメールクライアントアプリケーションで開かれます。
- 現在ビューアで正しく処理されないかもしれないその他の値。そのような場合、リンク処理のプロシージャはユーザーによって処理されます。

レポートビューアは特別な `HyperlinkClick` イベントを使用してカスタムなユーザーリンクまたは標準リンクのカスタム処理を行います。例えば、「戻る」ハイパーリンクを使用してレポートの先頭ページに移動しなければならない場合、次のコードで行えます。まず、イベントに記述する必要があります。

```
// イベントに記述
reportViewer.HyperlinkClick += reportViewer_HyperlinkClick;
```

次に、ビューアが先頭ページに切り替わるように、このイベントハンドラを使用して `GoToFirstPageCommand` を呼び出せます。

```
private void reportViewer_HyperlinkClick(object sender,
    PerpetuumSoft.WinRT.Viewer.ReportHyperlinkEventArgs e)
{
    if (e.Hyperlink.ToLower() == "back")
    {
        reportViewer.Commands.GoToFirstPageCommand.Execute(null);
        e.Handled = true;
    }
}
```

注意：イベントの引数を使用してリンクの値を取得します。通常、ビューアにそのようなリンクの処理をさせないようにするには、処理後に「`Handled = true`」の設定が必要です。

ドリル スルー

ドリルスルー機能を使用して、レポートの詳細情報を取得できます。ドリルスルーの利点をうまく利用するには、レポートをきちんと用意する必要があります。ユーザーは少なくとも2つのレポート（1つは省略型、もう一方は省略型レポートのある部分の詳細説明）を作成する必要があります。1つのレポートからもう一方のレポートへの移動は、ハイパーリンクを使用して行います。これを行うには、レポートにハイパーリンクを追加し、`HyperlinkClick` イベントに記述します。1つのレポートから別のレポートに移動するロジックは `HyperlinkClick` のイベントハンドラで実装されます。

```
reportViewer.HyperlinkClick += reportViewer_HyperlinkClick;

...

void reportViewer_HyperlinkClick(object sender,
PerpetuumSoft.WinRT.Viewer.ReportHyperlinkEventArgs e)
{
    if (e.Hyperlink == "GotoDetailedReport")
    {
        reportViewer.ReportName = "DetailedReport";
        reportViewer.RenderReport();
        e.Handled = true;
    }
}
```

ナビゲーション履歴

ナビゲーション履歴でレポート間を移動できます。ビューアでレポートをプレビューしながら、ドリルスルー機能を使用するか、コードに他のレポート名を設定して **RenderReport** メソッドを呼び出してレポート間を移動できます。**RenderReport** メソッドに関しては、ビューアのボタンをクリックするか、



次のビューアコマンドのうちのどれかを実行することで、現在表示されているレポートがナビゲーション履歴に追加されます。

```
reportViewer.Commands.ForwardCommand.Execute(null);
```

```
reportViewer.Commands.BackwardCommand.Execute(null);
```

ナビゲーション履歴に前のレポートを追加せずに次のレポートに移動しなければならない場合、次のパラメータが付いた **RenderReport** メソッドを呼び出します。

```
reportViewer.RenderReport(PerpetuumSoft.WinRT.Viewer.HistoryOptions.NotSave)
```

レポートビューアのコマンド

レポートビューアは、さまざまなコマンド (Commands プロパティ) を提供します。

各コマンドは、 **ICommand** から継承された **IExtendedCommand** インターフェイスを実装します。このインターフェイスには付加的な **GetIsSupported()** メソッドがあり、コマンドがサポートされているか調べることができます。

コマンド	機能	適用先
<i>レポート生成コマンド</i>		
RenderReportCommand (パラメータとして RenderOptions が必要です)	レポートを生成します	RenderReport() メソッドを呼び出します
RefreshReportCommand	現在のレポートを再生成	RefreshReportButton 、

	します	ReportParametersMenu
<u>ナビゲーションコマンド</u>		
BackwardCommand	ナビゲーション履歴の前のレポートに移動します	BackwardButton、 ReportNavigationPanel
ForwardCommand	ナビゲーション履歴の次のレポートに移動します	ForwardButton、 ReportNavigationPanel
GoToFirstPageCommand	現在のレポートの先頭ページに移動します	GoToFirstPageButton、 ReportNavigationPanel
GoToLastPageCommand	現在のレポートの最終ページに移動します	GoToLastPageButton、 ReportNavigationPanel
GoToNextPageCommand	現在のレポートの次のページに移動します	GoToNextPageButton、 ReportNavigationPanel
GoToPreviousPageCommand	現在のレポートの前のページに移動します	GoToPreviousPageButton、 ReportNavigationPanel
<u>検索コマンド</u>		
FindTextCommand	検索フィールドに指定した、最初に見つかるテキスト/単語を検索します	FindButton、 ReportSearchPanel
FindNextTextCommand	検索フィールドに指定した、次に見つかるテキスト/単語を検索します。 (ループ検索。ドキュメントの最後まで行くと、初めから検索されます。)	FindNextButton、 ReportSearchPanel
FindPreviousTextCommand	検索フィールドに指定した、前に見つかるテキスト/単語を検索します。 (ループ検索。コメントの先頭まで行くと、最後から検索されます。)	FindPrevButton、 ReportSearchPanel
<u>レポート表示コマンド</u>		
SwitchToSemanticZoomViewCommand	セマンティックズーム表示に切り替えます	SemanticZoomViewButton、 ReportViewModePanel
SwitchToGridViewCommand	グリッド表示に切り替えます	GridViewButton、 ReportViewModePanel
SwitchToFlipViewCommand (パラメータとしてページインデックスが必要です)	フリップ表示に切り替えます	FlipViewButton、 ReportViewModePanel
ShowThumbnailsPanelCommand	サムネイルパネルを表示します	ThumbnailsButton、 ReportViewModePanel
ShowBookmarksTreeCommand	(可能な場合) ブックマークのツリーを表示します	BookmarksButton、 ReportViewModePanel
GoToPageContentCommand (パラメ	サムネイルからグリッド	グリッド表示の

ータとしてページインデックスが必要ですが)	表示に切り替えます	ViewerGridItem とサムネイルパネル
<i>その他のコマンド</i>		
PrintCommand	ドキュメントを印刷に送ります (印刷するページを用意します)	PrintButton、ReportPrintPanel
ShowParametersPanelCommand	(可能な場合) レポートパラメータが付いたパネルを表示します	ParametersButton、ReportParametersMenu

パラメータ

一般情報

パラメータは、クライアントからレポートに任意の情報を送るための主要なツールです。通常、こうした情報にはレポートの見たい目やレポートの生成処理に作用する特有の設定が含まれています。例えば、ビューアのクライアントとサーバーの性質を考えれば、レポートで使用されている時間パラメータをクライアントから送って、レポートの生成中に使用されたサーバーの時間を指定することができます。あるいは、例えば、データを選択するための期間をパラメータに渡すことができます。

パラメータ入力 UI

レポートビューアには、一般的な種類のパラメータ入力 UI を自動表示する機能があります。主なパラメータ型は次の通りです。

- Integer
- Fractional number
- Boolean
- Date
- String

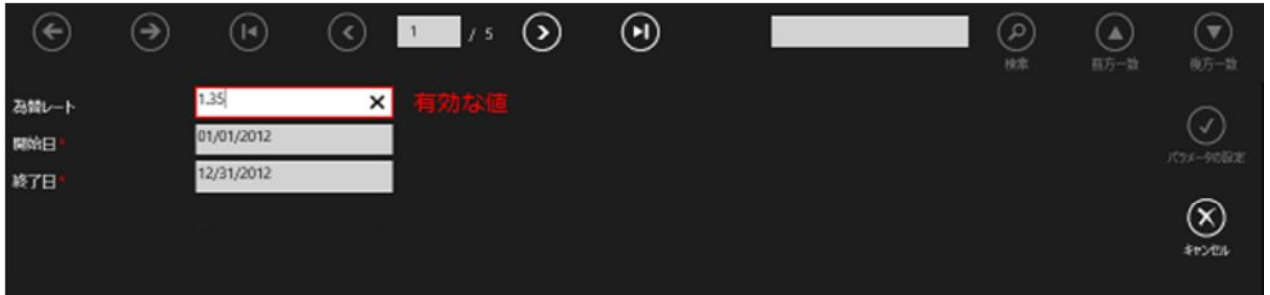
更に、上記の型を入力するカスタムフィールドを追加して、入力フィールドなどの代わりに使用する場合があります。従って、例えば、レポートに **FirstName** と **LastName** フィールドを含んだ **Person** という複雑なオブジェクトを渡して、弊社のインターフェイスに **PersonFirstName** と **PersonLastName** という二つの入力フィールドだけ表示します。カスタム化に関する詳細は、「外觀のカスタマイズ」を参照してください。

注意：パラメータ入力 UI が表示されるのは、1つ/複数のレポートパラメータにデフォルト値がなく、レポート生成の呼び出しが発生する前のコードに設定されなかった場合のみです。

いずれの場合でも、パラメータボタンを押して表示される入力パネルを使って、パラメータをいつでも変更できます。



ビューア上部に入力パネルが表示されます。見た目は以下の通りです。



整数の入力値の検証が行われ、「パラメータの設定」ボタンは全フィールドに正しいデータが入力されるまで無効になっているので注意してください。また、デフォルト値のないフィールドにはアスタリスク (*) が付き、そのフィールドは必須であるという点も重要です。

デフォルトで数値フィールドの検証が行われますが、このユーザーガイドのカスタマイズに書かれているメカニズムを使用して、サーバー側とクライアント側の両方にカスタムな検証を追加できます。エラーメッセージは埋め込み式の検証と同じように表示されます。

必須のパラメータを全部入力するまではレポートの生成は行えませんので注意してください。キャンセルボタンをクリックすると、パラメータはサーバーに送られず、レポート生成は行われません。

コードによるパラメータの設定

例えば、クライアント側の時間、現在のカルチャなどの情報をクライアントアプリケーションのコードからレポートに転送しなければならない場合があります。もちろん、クライアント側で現在時刻を手で入力するよう促すことはできますが、当然アプリケーションからそういった要求が来れば驚かれると思います。更に、そういったパラメータは非表示にした方が妥当です。パラメータのカスタマイズに関する詳細は、「パラメータ処理をカスタマイズするメカニズム」を参照してください。

他にこれが役立つようなのはデフォルト値の設定です。例えば、先月のレポートをデフォルトで表示したい場合です。

コードでパラメータを設定するメカニズムはいたってシンプルです。レポートビューアの `Parameters` プロパティのインデクサにパラメータ名を渡す必要があります。例えば、`DateFrom` パラメータには今月初めの値を、`DateTo` パラメータには今月末の値を設定します。

```
var today = DateTime.Today;
reportViewer.Parameters["DateFrom"] = today.AddDays(1 - today.Day);
reportViewer.Parameters["DateTo"] = new DateTime(today.Year, today.Month,
    DateTime.DaysInMonth(today.Year, today.Month));
```

レポート生成の開始前にパラメータを設定した場合のみ有効になるので注意してください。

パラメータ処理のカスタマイズ方法

ビューアで提供されている機能では物足らず、機能を拡張したい場合があります。

パラメータ処理の拡張メカニズムの主なポイント：

- サーバー上で **GetReportParametersInfo** サービスメソッドをオーバーライドできます。クライアントで取得したパラメーター一覧をサーバーから変更できます。カスタマイズの一般的な方法は、指定したレポートのパラメーター一覧を形成するための基本実装から取得しており、**ReportParametersInfo** パラメータの値を変更することでユーザーに見せられるパラメータを変更、追加、削除しています。
- サーバー上で **ValidateParameters** サービスメソッドをオーバーライドできます。このメソッドは全パラメータに値が提供されたか、この値とパラメータ型が一致するかを確認します。検証をもっと厳しくする必要がある場合は、このメソッドで実装する必要があります。
- サーバー上で **PreprocessParameters** サービスメソッドをオーバーライドできます。このメソッドはすべてのパラメータの検証後に実行されます。ユーザーから取得したパラメータをレポートで要求されるパラメータに変換する必要があります。例えば、ユーザーに **Name** (ID リスト) を表示して、このメソッドで **ID** の値をデータベースオブジェクトに変換できます。
- **ParametersListReceived** ビューア イベントに記述できます。このイベントは、サーバーからパラメーター一覧を取得した後に発生します。パラメータを非表示にして、このイベントハンドラに新しいパラメータを追加できます (クライアントでこのイベントに追加されたパラメータはサーバーには渡されないの注意してください)。
- **ParametersFilled** ビューア イベントに記述できます。このイベントは、レポートパラメータを入力し「パラメータの設定」ボタンを押した後に発生します。このイベントは正しい正当なパラメータに発生します。

イベント処理をカスタマイズする一般的な方法について説明します。

1. クライアント側でのパラメータ検証

いたってシンプルで、イベントに記述してください。

```
reportViewer.ParametersFilled +=reportViewer_ParametersFilled;
```

このイベントハンドラで、イベントの引数から渡された値コレクションからパラメータ値を抽出してください。エラーが検出された場合、パラメータ名とエラーメッセージのテキストを **ValidationErrors** 辞書のキーとして追加する必要があります。

```
private void reportViewer_ParametersFilled(object sender,
    PerpetuumSoft.WinRT.Viewer.Managing.ParametersFilledEventArgs e)
{
    var parameter = e.Values.Where(x => x.Name == "param").SingleOrDefault();
    if (parameter != null)
    {
        var value = int.Parse(parameter.Value);
        if (value < 10)
        {
            e.ValidationErrors.Add("param", "Value cannot be less than 10!");
        }
    }
}
```

クライアント側の検証はサーバーに要求を送る必要がないので、この種の検証を使うことを強く推奨します。

2. サーバー側でのパラメータ検証

これもいたってシンプルですが、クライアント側のやり方とは違います。

`ValidateParameters` サービスメソッドをオーバーライドしてその中でパラメータを検証する必要があります。このメソッドはパラメータの説明とパラメータ値を含んだ `ParameterData` リストを取得します。パラメータ値が有効でない場合、`ParameterData.ParameterInfo.IsValid` フラグを削除し、`ParameterData.ParameterInfo.ValidationErrorMessage` プロパティにエラーメッセージを設定する必要があります。

```
protected override void ValidateParameters(List<ParameterData> parametersData)
{
    base.ValidateParameters(parametersData);
    var parameter = parametersData
        .Where(x => x.ParameterInfo.Name == "parameter")
        .SingleOrDefault();
    if (parameter != null && parameter.Value != null)
    {
        var value = Convert.ToDouble(parameter.Value);
        if (value < 10)
        {
            parameter.ParameterInfo.IsValid = false;
            parameter.ParameterInfo.ValidationErrorMessage =
                "Parameter cannot be less than 10";
        }
        if (value > 20)
        {
            parameter.ParameterInfo.IsValid = false;
            parameter.ParameterInfo.ValidationErrorMessage =
                "Parameter cannot be more than 20";
        }
    }
}
```

Convert クラスのメソッドを使用することが推奨されているという点に注目してください。問題は、Double 型のパラメータ値が int 型として渡される可能性があり、明白なキャストイングの場合、エラーが生じることになります。

3. クライアント側でパラメータを非表示にする

パラメータを非表示にするには、ParametersListReceived イベントハンドラでプロパティを PromptUser=false に設定する必要があります。

```
private void reportViewer_ParametersListReceived(object sender,
    PerpetuumSoft.WinRT.Viewer.Managing.ParametersListReceivedEventArgs e)
{
    var parameter = e.Parameters
        .Where(x => x.Name == "parameter")
        .SingleOrDefault();
    if (parameter != null)
    {
        parameter.PromptUser = false;
    }
}
```

4. サーバー側でパラメータを非表示にする

サーバー側でパラメータを非表示にするには、GetReportParametersInfo オーバーライドメソッドで ShowOnPanel フラグをリセットする必要があります。

```
public override ReportParametersInfo GetReportParametersInfo(
    string reportName,
    string parameters,
    string parametersType,
    out ExceptionDetailBase reportError)
{
    var result = base
        .GetReportParametersInfo(reportName,
            parameters, parametersType, out reportError);

    if (reportError == null)
    {
        var parameter = result.Parameters
            .Where(x => x.Name == "parameter")
            .SingleOrDefault();
        if (parameter != null)
        {
            parameter.ShowOnPanel = false;
        }
    }
}
```

5. クライアント側でパラメータ一覧を提供する

パラメータ値を手入力するのではなく、提供されたパラメータ一覧から選択しなければならない場合があります。これは、クライアント側で ParametersListReceived イベントハンドラの ValidValues コレクションを初期設定して行えます。

```
private void reportViewer_ParametersListReceived(object sender,
    PerpetuumSoft.WinRT.Viewer.Managing.ParametersListReceivedEventArgs e)
{
    var parameter = e.Parameters
        .Where(x => x.Name == "gender")
        .SingleOrDefault();
    if (parameter != null)
    {
        parameter.ValidValues
            = new ObservableCollection<ValidValue>()
            {
                new ValidValue(){ Label="Male", Value="MALE" },
                new ValidValue(){ Label="Female", Value="FEMALE" },
            };
    }
}
```

6. サーバー側でパラメータ一覧を提供する

サーバー側のパラメータ一覧は、GetReportParametersInfo オーバーライドイベントに記述されます。パラメータの値コレクションはこのように記述されます。

```
public override ReportParametersInfo GetReportParametersInfo(
    string reportName,
    string parameters,
    string parametersType,
    out ExceptionDetailBase reportError)
{
    var result = base
        .GetReportParametersInfo(reportName,
            parameters, parametersType, out reportError);

    if (result != null)
    {
        var parameter = result.Parameters
            .Where(x => x.Name == "string")
            .SingleOrDefault();
        if (parameter != null)
        {
            parameter.Values.Add(new LabeledValue()
            {
                Label = "Male",
                Value = "MALE"
            });
            parameter.Values.Add(new LabeledValue()
            {
                Label = "Female",
                Value = "FEMALE"
            });
        }
    }
}
```

7. クライアントでのパラメータの追加

ParametersListReceived イベントハンドラにクライアント側のパラメータを追加できます。クライアント側のこのイベントに作成された付加的なパラメータはサーバー側には渡されません。サーバー側で処理するパラメータを作成しなければならない場合は、それに応じてサーバー側でパラメータを作成してください。ParametersFilled イベントはクライアント側で作成されたパラメータの処理に使用できます。

```
private void reportViewer_ParametersListReceived(object sender,
    PerpetuumSoft.WinRT.Viewer.Managing.ParametersListReceivedEventArgs e)
{
    e.Parameters.Add(new PerpetuumSoft.WinRT.Viewer.Model.ReportParameter()
        {
            Name = "ClientSideParameter",
            Prompt = "My Parameter",
            PromptUser = true,
            Type = PerpetuumSoft.WinRT.Viewer.Model.ParameterTypeEnum.String,
            Visibility = true,
        });
}
```

8. サーバーでのパラメータの追加

サーバー側にパラメータを追加するには、GetReportParametersInfo メソッドをオーバーライドし、メソッドの基本実装で作成されたコレクションに新しいパラメータを追加する必要があります。

```
public override ReportParametersInfo GetReportParametersInfo(
    string reportName,
    string parameters,
    string parametersType,
    out ExceptionDetailBase reportError)
{
    var result = base
        .GetReportParametersInfo(reportName,
            parameters, parametersType, out reportError);

    if (result != null)
    {
        result.Parameters.Add(new ParameterInfo()
            {
                DisplayName = "My Parameter",
                Name = "ServerSideParameter",
                OriginalType = "System.String",
                ShowOnPanel = true,
                Type = ParameterType.String,
            });
    }
}
```

9. 処理できないパラメータ処理

前述した通り、限られた数のパラメータ型の入力フィールドをビューアに自動表示できます。サポートされている型と一致しないパラメータを処理しなければならない場合に

備えて、そのようなパラメータ一覧が `OnUnprocessableParametersDetected` メソッドに渡されます。このように、このメソッドをオーバーライドするとそのようなパラメータを処理できるようになります。次の例では、2つのパラメータを使用して `User` パラメータの `first name` と `last name` を入力し、入力された氏名の値を使用して `PreprocessParameters` メソッドで `User` オブジェクトを作成します。

```
protected override void OnUnprocessableParametersDetected(
    List<PerpetuumSoft.Reporting.DOM.Parameter> problemParameters,
    List<ParameterInfo> resultList)
{
    var user = problemParameters.Where(x => x.Name == "User")
        .SingleOrDefault();
    if (user != null)
    {
        resultList.Add(new ParameterInfo()
            {
                DisplayName = "User First Name",
                Name = "UserFirstName",
                OriginalType = "System.String",
                ShowOnPanel = true,
                Type = ParameterType.String
            });
        resultList.Add(new ParameterInfo()
            {
                DisplayName = "User Last Name",
                Name = "UserLastName",
                OriginalType = "System.String",
                ShowOnPanel = true,
                Type = ParameterType.String
            });
    }
}

protected override void PreprocessParameters(
    IDictionary<string, object> parameters, ReportParametersInfo parameterInfo)
{
    base.PreprocessParameters(parameters, parameterInfo);

    if (parameters.ContainsKey("UserFirstName")
        && parameters.ContainsKey("UserLastName"))
    {
        parameters.Add("User",
            new User(
                Convert.ToString(parameters["UserFirstName"]),
                Convert.ToString(parameters["UserLastName"])));
    }
}
```

エクスポート

ReportViewer オブジェクトを使用してコードからドキュメントをエクスポートする場合があります。

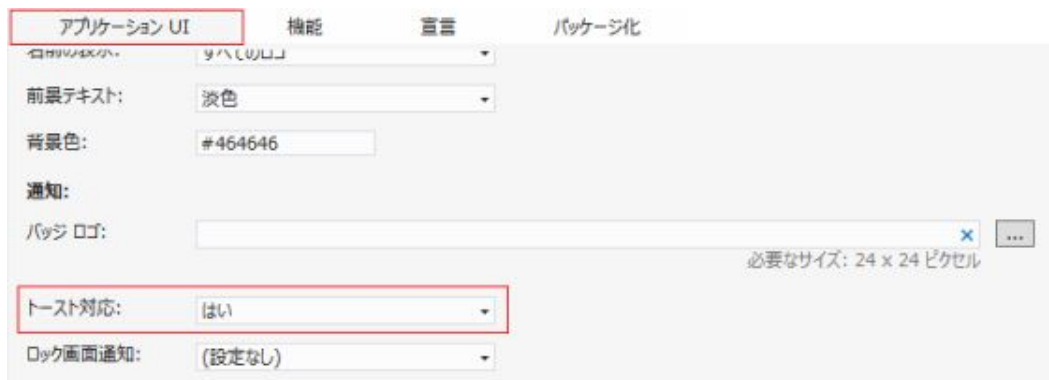
標準のダイアログの使用

ファイルへの保存を取得しオプションを共有するには、アプリケーションのマニフェストを開いて [トースト対応] を「はい」に指定しなければなりません。

Package.appxmanifest マニフェストを開き、次のように設定してください。

エクスポートは、トースト通知を使用してエクスポートの開始と終了を通知します。従って、アプリケーションのトーストを有効にする必要があります。

(アプリケーション UI タブで) [トースト対応] を「はい」に設定してください。



エクスポートを開始するには、「エクスポート」ボタンをタップする必要があります。



標準のエクスポートフィルタの一覧が表示されます。



1つ選ぶと、オプションメニューが表示されます。

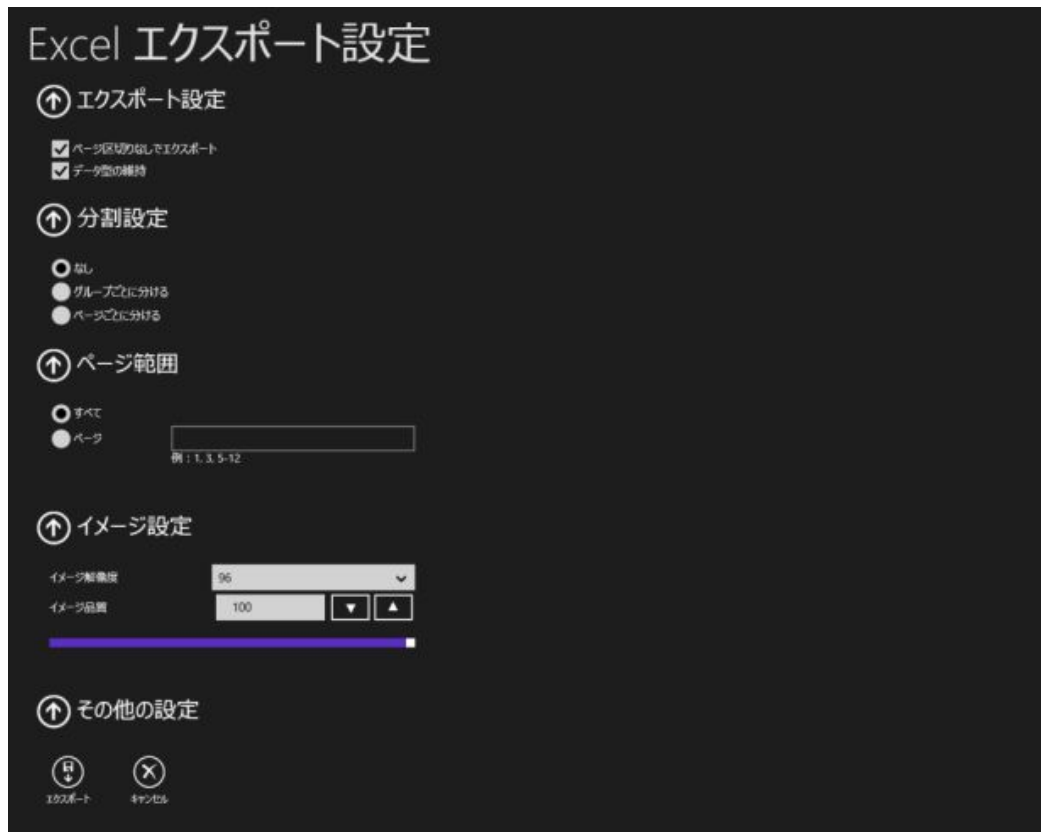
(このオプションメニューを表示するには、MainPage.xaml.cs に次の黄色い部分のコードを設定する必要があります。)

```
private void MainPage_Loaded(object sender, RoutedEventArgs e)
{
    reportViewer.RenderReport();
    reportViewer.ExportMode = PerpetuumSoft.WinRT.Viewer.ExportMode.UserSelected;
}
```



製品にデフォルトで同梱されているダイアログは2つです。

1. OpenXmlExcel エクスポート設定



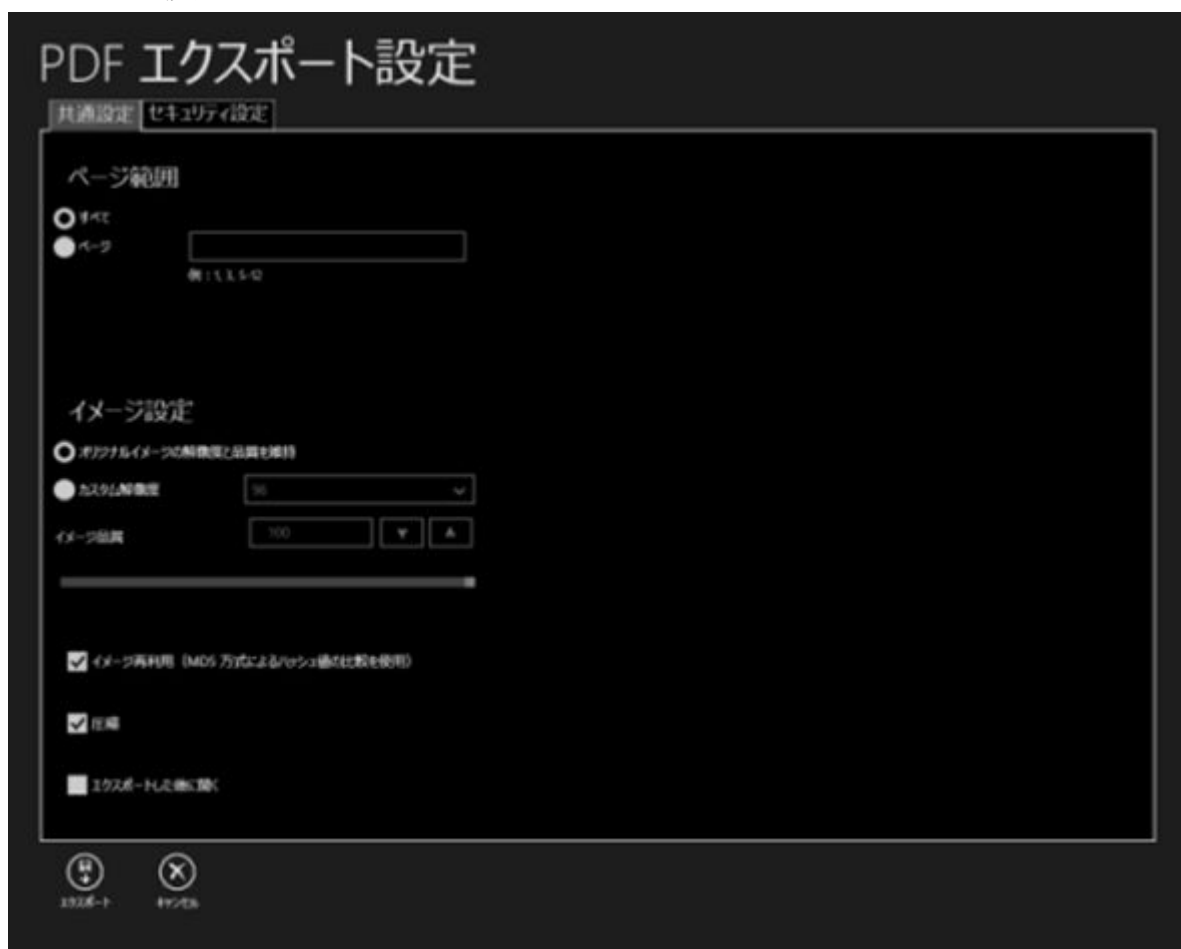
Export without page delimiters

ページ間のページフッターとページヘッダーを省略するか指定します

Export to one sheet	全ページを1シートにエクスポートするか、各ページを各シートにエクスポートするか指定します
PageRange	エクスポートするページを指定します
ImageResolution	エクスポートするイメージの解像度を指定します
ImageQuality	Jpeg イメージの品質ロス。100 - ロスなし
Open After Export	エクスポート後にドキュメントを開くか定義します。

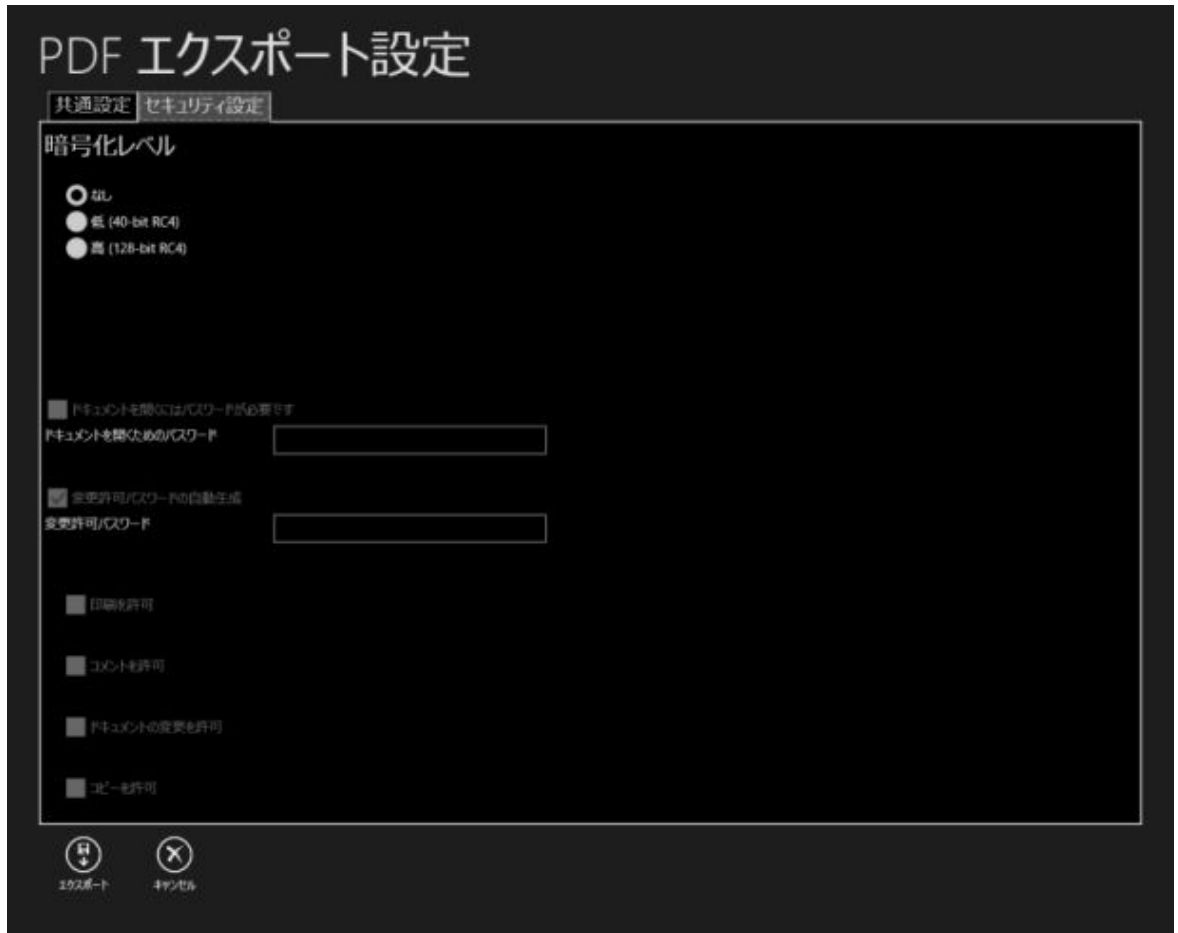
2. PDF エクスポート設定

a. 共通設定



PageRange	エクスポートするページを指定します
KeepOriginalImageResolutionAndQuality	イメージをドキュメントと同じように PDF にエクスポートするか、指定したイメージ設定を適用するかを指定します
Custom Resolution	エクスポートするイメージの解像度を指定します
ImageQuality	Jpeg イメージの品質ロス 100 - ロスなし
Reuse Images	重複イメージに、イメージの再利用アルゴリズムを使用するか指定します
Compress	PDF を圧縮するか指定します
Open After Export	エクスポート後にドキュメントを開くか定義します

b. セキュリティ設定



Encription Level	PDF のセキュリティレベル (None、Low40RC4、または High128RC4) を指定します
Document opening password	PDF ドキュメントのパスワードを指定します
Change permissions password	PDF ドキュメントの変更を許可するパスワードを指定します
AllowPrinting	エクスポートした PDF の印刷を許可するか指定します
AllowCommenting	エクスポートしたドキュメントにコメント許可するか指定します
AllowChangingDocument	エクスポートしたドキュメントの変更を許可するか指定します
AllowCopyingContent	エクスポートしたドキュメントのコピーを許可するか指定します

Export メソッドの使用

ダイアログの他に、コードからエクスポートを呼び出すこともできます。

```
public void Export(string exportFormat, ExportMode mode)
```

例 :

```
this.reportViewer.Export("pdf", PerpetuumSoft.WinRT.Viewer.ExportMode.SaveToFile);
```

標準のエクスポート形式がいくつかあります。

```
"csv"
"excel"
"excelxml"
"html"
"xlsx"
"pdf"
"rtf"
"xps"
```

SaveToFile を使用し、コードから *ExportMode* を共有できます。

ExportToStream 非同期メソッドの使用

このメソッドのオーバーロードは2つあります。

```
async Task<ExportResult> ExportToStream(string exportFormat)
```

```
async Task<ExportResult> ExportToStream(string exportFormat, ExportOptions exportOptions)
```

これらのメソッドを使用するには、ドキュメントが読み込まれるまで待って、このメソッドに *async* 変更子を付けます。

```
async void reportViewer_DocumentLoaded(object sender,
PerpetuumSoft.WinRT.Viewer.Managing.DocumentLoadedEventArgs e)
{
    var result = await this.reportViewer.ExportToStream("pdf");
}
```

ExportResult に次の情報が含まれます。

IsSucceed	エクスポートが無事終了したかを定義する値を取得します
Error	エラー記述またはエクスポートが無事終了した場合は <i>null</i>
Data	エクスポートされたドキュメントのバイト
FileName	ドキュメント名+.拡張子
MimeType	ファイルの <i>mimetype</i>

ExportOptions クラスはさまざまなエクスポート設定 (ページ範囲、エクスポート後に開く、など) を表します。 *PDFExportOptions* と *ExcelExportOptions* という2つの継承クラスがあります。

デフォルトのエクスポート形式は一覧の通りです。オプションについて説明します。

ExportOptions クラス : 共通のエクスポートフィルタ設定を定義します。

OpenAfter	エクスポート後にドキュメントを開くか定義します
PageRange	エクスポートするページを指定します

ExcelExportOptions クラス : xlsx エクスポート設定を定義します。

ExportWithoutPageDelimiters	ページ間のページフッターとページヘッダーを省略するか指定します
ExportToOneSheet	全ページを1シートにエクスポートするか、各ページを各シートにエクスポートするか指定します
ImageResolution	エクスポートするイメージの解像度を指定します
ImageQuality	Jpeg イメージの品質ロス。100 - ロスなし

PDFExportOptions クラス : PDF エクスポート設定を定義します。

EmbedFonts	エクスポート時に PDF ドキュメントにフォントを埋め込みます
Compress	PDF ドキュメントを圧縮するか指定します
KeepOriginalImageResolutionAndQuality	イメージをドキュメントと同じように PDF にエクスポートするか、指定したイメージ設定を適用するか指定します
ImageResolution	エクスポートするイメージの解像度を指定します
EmbedPrivateCharacters	PDF ドキュメントに外字を埋め込むか指定します
ImageQuality	Jpeg イメージの品質ロス。100 - ロスなし
SecurityLevel	PDF のセキュリティレベル (None、Low40RC4、または High128RC4) を指定します
AllowPrinting	エクスポートした PDF ドキュメントの印刷を許可するか指定します
AllowCommenting	エクスポートしたドキュメントのコメントを許可するか指定します
AllowChangingDocument	エクスポートしたドキュメントの変更を許可するか指定します
AllowCopyingContent	エクスポートしたドキュメントのコピーを許可するか指定します
UserPassword	PDF ドキュメントのパスワードを指定します
ChangePermissionsPassword	PDF ドキュメントの変更を許可するパスワードを指定します
ReuseImage	重複イメージにイメージ再利用アルゴリズムを使用するか指定します
IsUserPasswordUsed	パスワードを使用するか指定します
IsAutomaticGenerationUsed	パスワードを自動生成するか指定します

PDF や Open Xml Excel エクスポートのデフォルト設定の指定

コードでエクスポート設定を指定する場合は、ProcessExportParameters イベントが必要です。

```
this.reportViewer.ProcessExportParameters += reportViewer_ProcessExportParameters;
```

以下に示してあるように、イベントハンドラからデフォルト設定を指定できます。

```
void reportViewer_ProcessExportParameters(object sender,
PerpetuumSoft.WinRT.Viewer.ProcessExportParametersEventArgs e)
{
    if (e.FormatName == "pdf")
    {
        e.ExportOptions = new PDFExportOptions()
        {
            OpenAfter = true,
            SecurityLevel = SecurityLevel.High128RC4,
            AllowPrinting = true,
            AllowCopyingContent = true,
            AllowCommenting = true,
            AllowChangingDocument = true
        };
        // e.Handled = true;
    }

    if (e.FormatName == "xlsx")
    {
        e.ExportOptions = new ExcelExportOptions()
        {
            OpenAfter = true,
            ExportToOneSheet = false,
            ExportWithoutPageDelimiters = false,
            ImageQuality = 100,
            ImageResolution = 600,
        };
    }
}
```

上記コードではコメントにしてありますが、e.Handled プロパティを true に指定すればエクスポート設定ウィンドウをスキップすることもできます。

エクスポート フィルタの一覧

レポートビューアにはあらかじめ設定されているエクスポートフィルタがいくつかあります。1つ/複数のエクスポートフィルタを削除できます。サービスで GetExportFilters メソッドをオーバーライドし、基本実装からエクスポートフィルタの一覧を取得して、不要なものを削除する必要があります。

```
public override List< ExportFilterDefinition>
    GetExportFilters(out ExceptionDetailBase ReportError)
{
    var filters = base.GetExportFilters(out ReportError);
    return filters
        .Where(filter => filter.FilterName != "Csv")
        .ToList();
}
```


エクスポートフィルタを追加するには、サービスデザイナーにご希望のエクスポートフィルタ コンポーネントをドラッグ&ドロップする必要があります。追加したエクスポートオプションのアイコンを追加するには、レポートビューアの **ExportLogos** 辞書にエクスポートフィルタ名と一致するキーを持つ新しい **ButtonTemplateSet** 要素を追加する必要があります。3つの状態のテンプレート (Normal、Hover、Pressed) も追加してください。

例えば、XPS エクスポートオプションのアイコンを取得しました。



XAML コードの設定は次の通りです。

```
<Canvas Width="48" Height="48">
  <Ellipse Width="30" Height="30" Canvas.Top="4"
    Canvas.Left="4" Stroke="White" StrokeThickness="2"/>
  <TextBlock Canvas.Left="10" Canvas.Top="16" FontSize="20"
    Text="XPS" Foreground="Black" FontWeight="Bold"/>
  <TextBlock Canvas.Left="11" Canvas.Top="17" FontSize="20"
    Text="XPS" Foreground="White" FontWeight="Bold"/>
</Canvas>
```

このアイコンを追加するか、これに変える必要があります。製品のテーマは白黒ベースなので、ボタンが押されたことを示すために、少なくとも2つのイメージ (黒地に白、白地に黒) を提供する必要があります。確かにコードから必要なオブジェクトを作成できましたが、**Page.Resources** に設定した方が便利です。

```
<Page.Resources>
  <customControls:ButtonTemplateSet x:Key="xpsLogo">
    <customControls:ButtonTemplateSet.NormalTemplate>
      <ControlTemplate>
        <Canvas Width="48" Height="48">
          <Ellipse Width="30" Height="30" Canvas.Top="4"
            Canvas.Left="4" Stroke="White" StrokeThickness="2"/>
          <TextBlock Canvas.Left="10" Canvas.Top="16" FontSize="20"
            Text="XPS" Foreground="Black" FontWeight="Bold"/>
          <TextBlock Canvas.Left="11" Canvas.Top="17" FontSize="20"
            Text="XPS" Foreground="White" FontWeight="Bold"/>
        </Canvas>
      </ControlTemplate>
    </customControls:ButtonTemplateSet.NormalTemplate>
    <customControls:ButtonTemplateSet.PressedTemplate>
      <ControlTemplate>
        <Canvas Width="48" Height="48">
          <Ellipse Width="30" Height="30" Canvas.Top="4"
            Canvas.Left="4" Stroke="Black" StrokeThickness="2"/>
          <TextBlock Canvas.Left="10" Canvas.Top="16" FontSize="20"
            Text="XPS" Foreground="White" FontWeight="Bold"/>
          <TextBlock Canvas.Left="11" Canvas.Top="17" FontSize="20"
            Text="XPS" Foreground="Black" FontWeight="Bold"/>
        </Canvas>
      </ControlTemplate>
    </customControls:ButtonTemplateSet.PressedTemplate>
  </customControls:ButtonTemplateSet>
</Page.Resources>
```

```
</customControls:ButtonTemplateSet.PressedTemplate>  
</customControls:ButtonTemplateSet>  
</Page.Resources>
```

以下のコードを使用してアイコンを設定：

```
var item = this.Resources["xpsLogo"] as  
    PerpetuumSoft.WinRT.Viewer.Controls.ButtonTemplateSet;  
reportViewer.ExportLogos.Remove("xps");  
reportViewer.ExportLogos.Add("xps", item);
```

更に、ビューアのスタイルをオーバーライドすることにより、標準のエクスポート オプションのイメージを変更（してカスタムなイメージを追加）できます。`wirt:ReportViewer` コントロールのスタイルをオーバーライドして、`ExportLogos` プロパティのセッターに設定されたパラメータを変更する必要があります。

外観のカスタマイズ

レポートビューアの構造

レポートビューアの構造は以下のように複雑です。

The screenshot shows a report viewer interface for 'Orders by Employee'. The interface is divided into several sections:

- 1 - ReportNavigationPanel:** Located at the top left, it contains navigation icons for back, forward, first, and last, along with a page indicator showing '1 of 30'.
- 2 - ReportSearchPanel:** Located at the top right, it includes a search input field and buttons for 'Find', 'Find Previous', and 'Find Next'.
- 3 - ReportParametersMenu:** Located at the bottom left, it contains icons for 'Parameters' and 'Refresh'.
- 4 - ReportViewModePanel:** Located at the bottom right, it contains icons for 'Report', 'Bookmarks', and 'Thumbnails'.
- 5 - ReportPrintPanel:** Located at the bottom right, it contains an icon for 'Print'.
- 6 - ReportExportPanel:** Located at the bottom right, it contains an icon for 'Export'.
- 7 - ViewerItemsControl:** The main content area in the center, displaying a table of orders grouped by employee.

The table 'Orders by Employee' lists various employees and their orders, including details like order ID, dates, and amounts. The total 'Orders sum' for each employee is also provided.

1 – ReportNavigationPanel

2 – ReportSearchPanel

3 – ReportParametersMenu

4 – ReportViewModePanel

5 – ReportPrintPanel

6 – ReportExportPanel

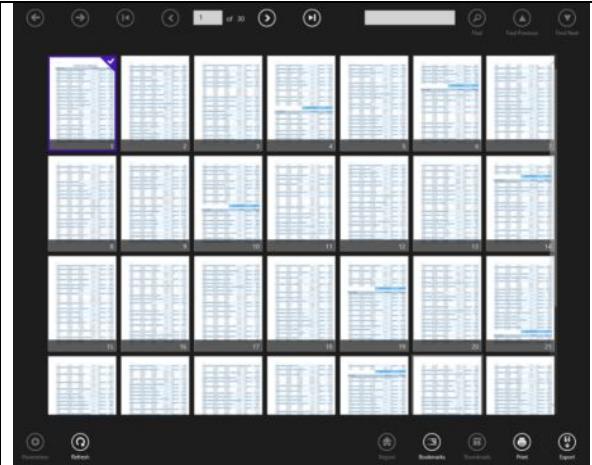
7 - ViewerItemsControl

ViewerItemsControl のスタイル

レポートビューアの外觀変更はすべて ViewerItemsControl のスタイルの差し換えがベースです。

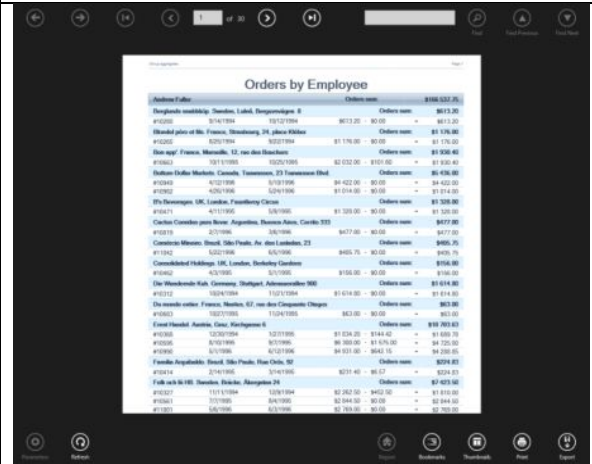
GridStyle

レポートのプレビュー：
サムネイルにページ番号が付いた、グリッド表示
(この表示に切り替えるコマンドはデフォルトで無効になっています)



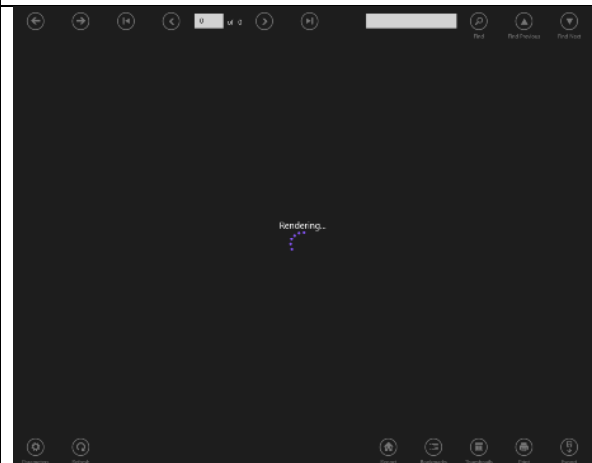
FlipStyle

レポートのページ表示
(この表示に切り替えるコマンドは無効になっています)



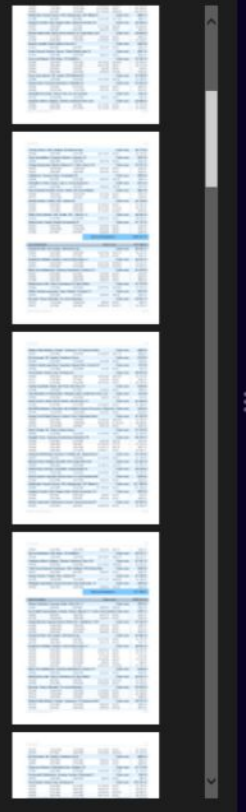
StaticStyle

中間状態（初期化、生成、読込に時間がかかる操作）の様子



ListStyle

アプリケーションがスナップ状態の時にサムネイルを一覧で表示した様子

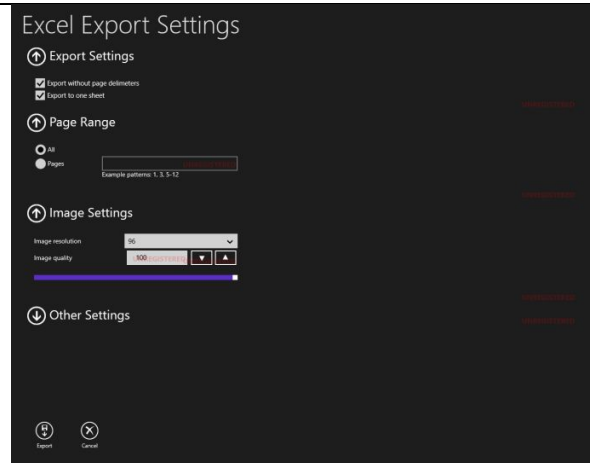


SemanticZoomStyle

ズームイン表示では FlipStyle、ズームアウト表示では GridStyle のように見える SemanticZoom コントロール。

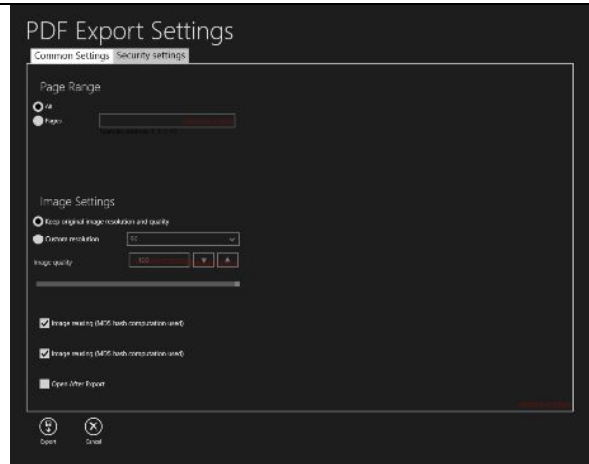
xlsxOptionsStyle

エクスポートオプションダイアログの様子



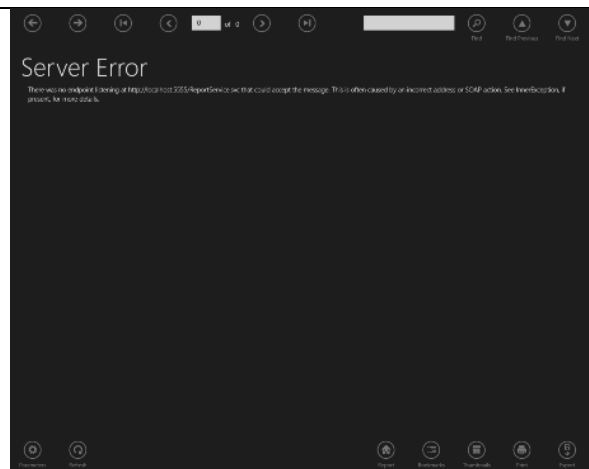
pdfOptionsStyle

エクスポートオプションダイアログの様子



ErrorStyle

エラーの様子

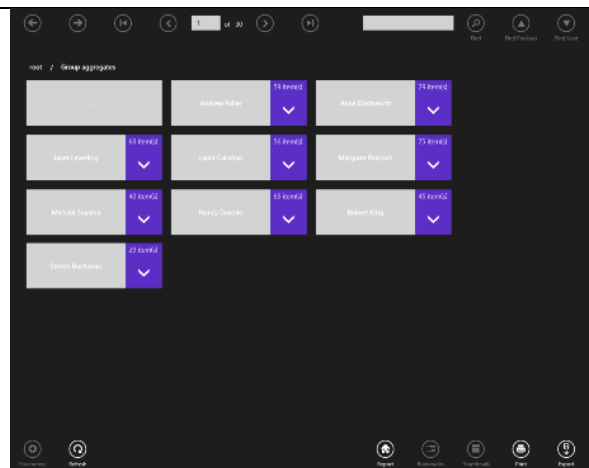


WaitingStyle

待機

BookmarkStyle

ブックマークのツリー表示の様子



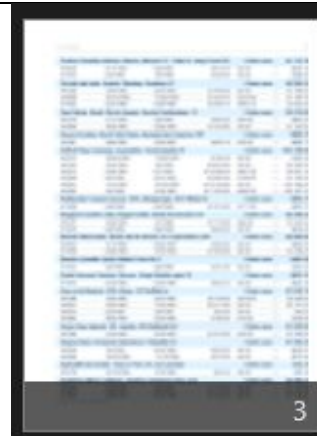
コントロール、アイテム、プレゼンター

色々なスタイルで `ViewerItemsControl` を作成するために、特別なコントロールが使用されます。使用されている名前空間の一覧は以下の通りです。

```
xmlns:customItems="using:PerpetuumSoft.WinRT.Viewer.Controls.CustomItems"
xmlns:local="using:PerpetuumSoft.WinRT.Viewer.Extenders"
xmlns:customControls="using:PerpetuumSoft.WinRT.Viewer.Controls"
xmlns:win8controls="using:PerpetuumSoft.Controls"
xmlns:parameterControls="using:PerpetuumSoft.WinRT.Viewer.Controls.ReportParameterControls"
xmlns:winrt="using:PerpetuumSoft.WinRT.Viewer"
xmlns:localization="using:PerpetuumSoft.WinRT.Viewer.Controls.Localization"
xmlns:utils="using:PerpetuumSoft.WinRT.Viewer.Utills"
```

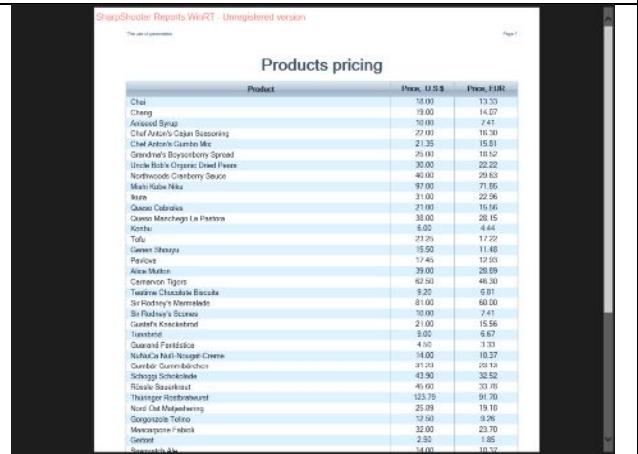
`customItems:ViewerGridItem`

`GridStyle` の `GridViewItem` の `DataTemplate` として使用されます

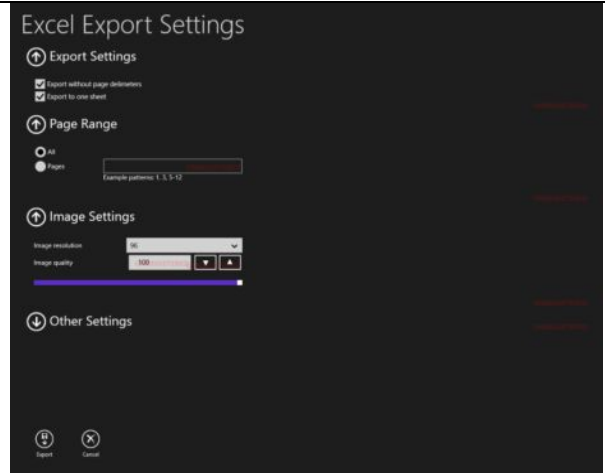


`customItems:ViewerFlipItem`

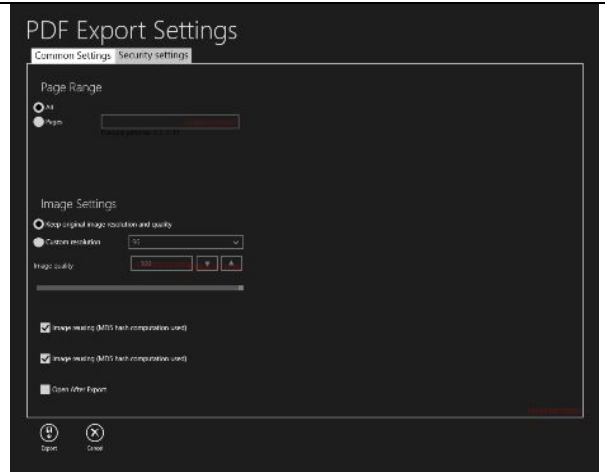
`FlipStyle` の `FlipViewItem` の `DataTemplate`



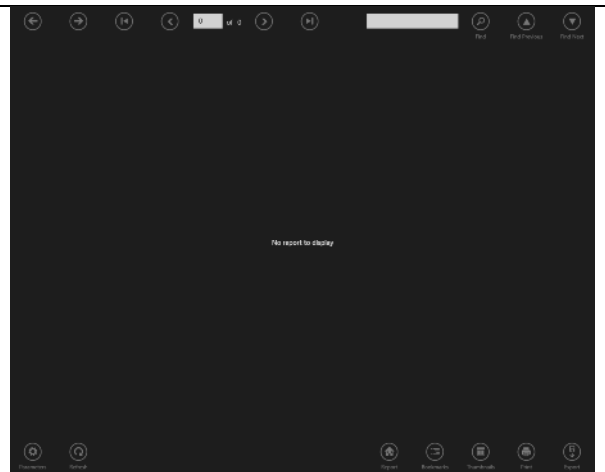
`customItems:ExcelOptionsPresenter`
`xlsxOptionsStyle` のベース。DataContext として
`ExcelOptionsViewModel` が必要です

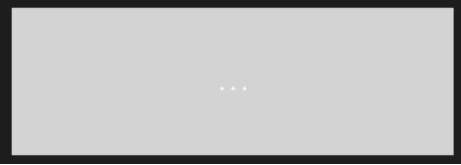
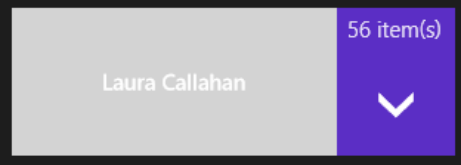
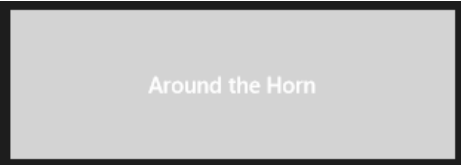
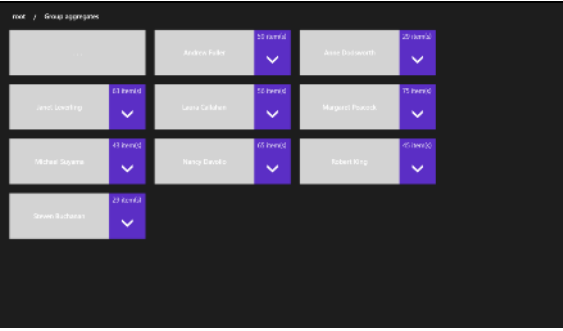
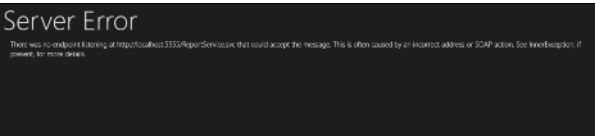


`customItems:PDFOptionsPresenter`
`pdfOptionsStyle` のベース。DataContext として
`PdfOptionsViewModel` が必要です。



`customItems:NoReportPresenter`
`NoReportStyle` のベース。



<p><code>customItems:BookmarkItemPresenter</code></p>	<p>上位に戻る :</p>  <p>入れ子のブックマークあり</p>  <p>入れ子のブックマークなし。</p> 
<p><code>customControls:BookmarkTreePresenter</code> ブックマークのツリーの現在位置を表示します</p>	
<p><code>customItems:ErrorPresenter</code> エラーメッセージを表示します</p>	

コントロール パネル

レポートビューアには複数のパネルがあり、それぞれ決まった機能があります。Generic.xaml は全パネルのデフォルトスタイルを表します。全パネルを AppBar に入れることができます。

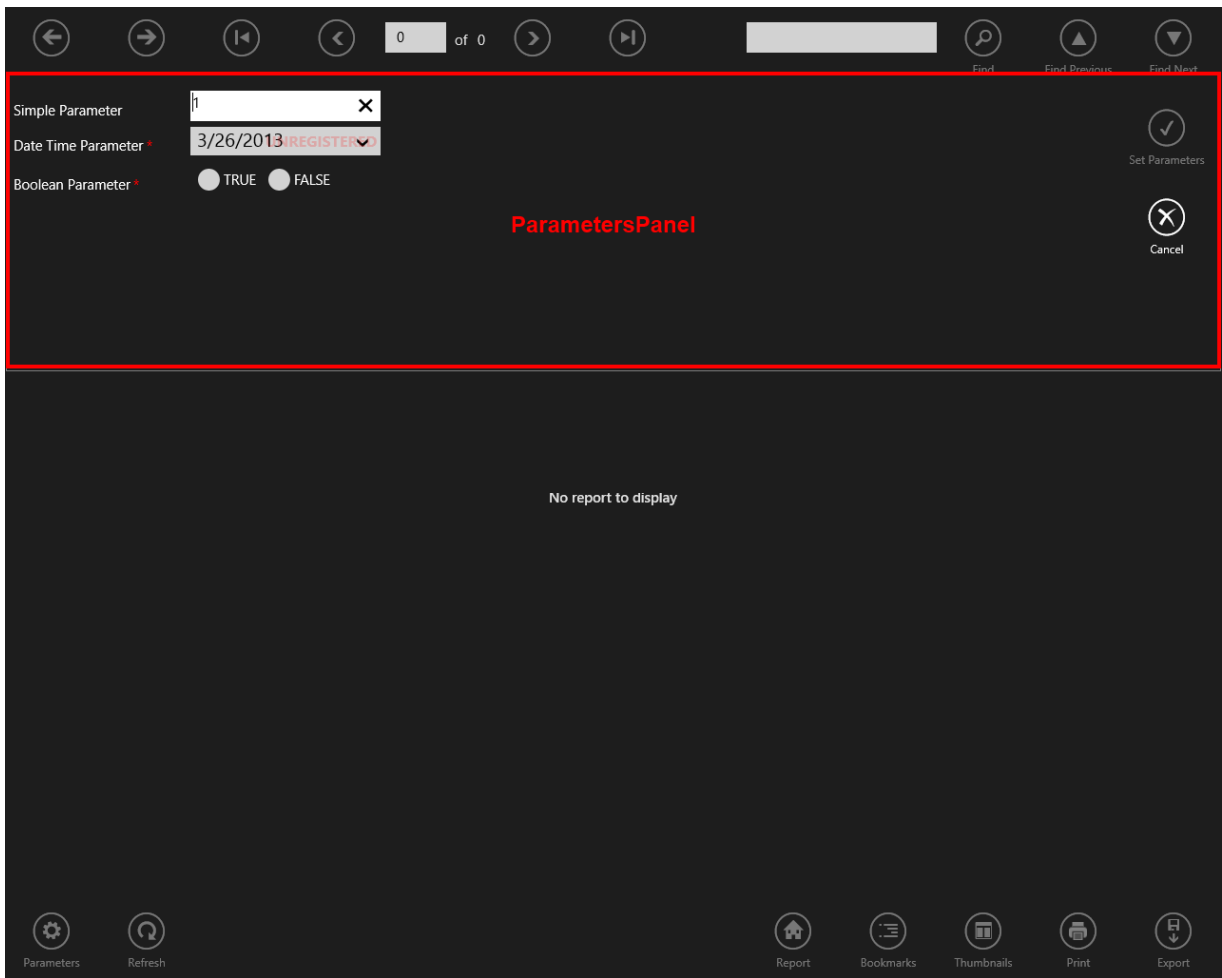
<p><code>customControls:ReportNavigationPanel</code> レポート内の移動や（レポートにハイパーリンクがある場合は）レポート間の移動</p> <ul style="list-style-type: none"> 1-戻る 2-進む 3-先頭ページ 4-前のページ 5-次のページ 	
--	--

<p>6 - 最終ページ</p> <p>customControls:ReportSearchPanel レポートのテキスト検索</p> <p>1 - 検索 2 - 前方一致 3 - 後方一致</p>	
<p>customControls:ReportParametersMenu レポートパラメータの設定、現在のレポートの更新</p> <p>1 - パラメータ 2 - 更新</p>	
<p>customControls:ReportViewModePanel さまざまな表示への切り替え、サムネイルパネル</p> <p>1 - セマンティックズーム 2 - ブックマーク 3 - サムネイル</p>	
<p>customControls:ReportPrintPanel レポート印刷</p>	
<p>customControls:ReportExportPanel レポートを特定形式にエクスポート。 ItemsControlStyle プロパティは形式一覧を持つフライアウトパネルのスタイルを表します。</p>	

非表示のパネル

パラメータ パネル

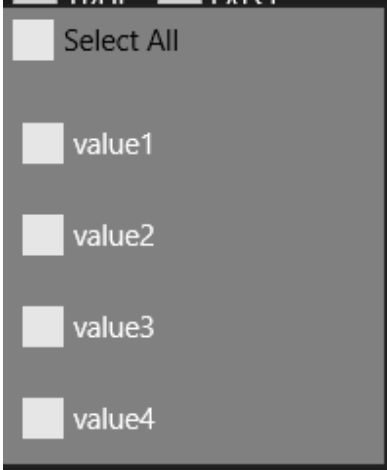
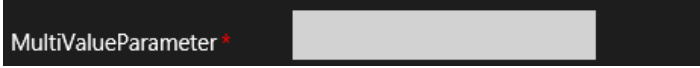


パラメータパネル (**customControls:ParametersPanel**) は、上からスワイプするとレポートパラメータを作成できます。



パラメータパネルは、デフォルトスタイルの `ParametersPanelStyle` を使用して表示されます。

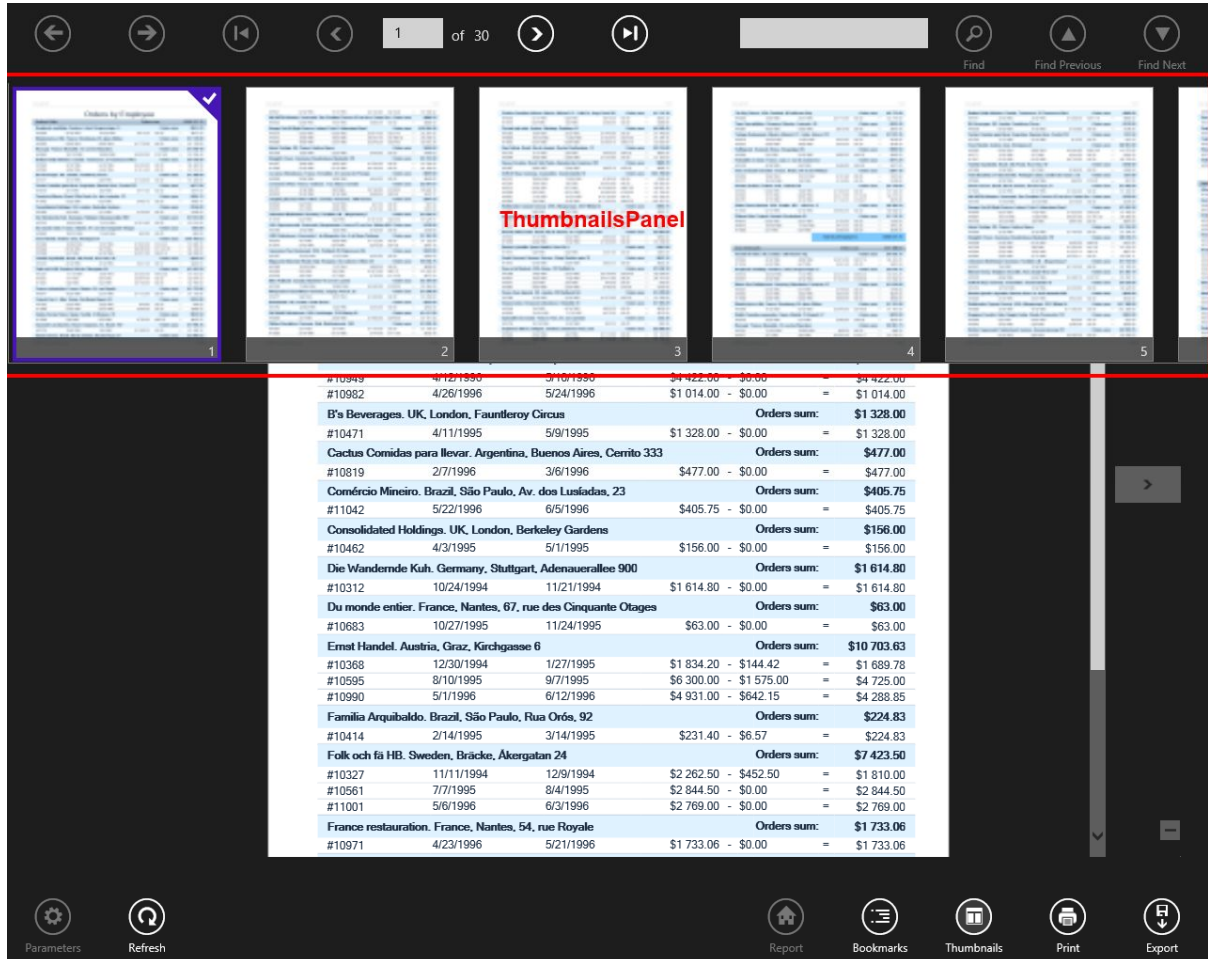
デフォルトスタイルを持つ各コントロールは、さまざまなパラメータ型に使用されます。

<code>parameterControls</code> <code>:BooleanParameterControl</code>	BooleanParameter * <input type="radio"/> TRUE <input type="radio"/> FALSE
<code>parameterControls</code> <code>:DateTimeParameterControl</code>	DateTimeParameter * 3/26/2013 REGISTERED <input type="checkbox"/> Null
<code>parameterControls</code> <code>:MultiValidValueParameterControl</code>	MultiValidValueParameter * <input type="text"/> ▼

<p><code>customControls:ComboBoxSelectAll</code> (MultiValidValueParameterControl に使用 されます。「すべて選択」チェックボ ックスは指定した <code>SelectAllControlStyle</code> スタイルを使用 します。</p>	
<p><code>parameterControls</code> <code>:MultiValueParameterControl</code></p>	
<p><code>parameterControls</code> <code>:SingleValidValueParameterControl</code></p>	
<p><code>parameterControls</code> <code>:StringParameterControl</code></p>	

サムネイル パネル

サムネイルパネル (`customControls:ThumbnailsPanel`) は、現在ページがズームイン表示になっている状態で上からスワイプすると、そのレポートを表示したままプレビューできます。



このパネルはデフォルトスタイルの `ThumbnailsPanelStyle` で表示されます。ViewerGridItem はアイテムの DataTemplate として使用されます。

外観のカスタマイズ

一例として、ViewerGridItem の見た目をカスタマイズしていきます。サムネイルの枠とページ番号を変更してみます。

ViewerGridItem のデフォルトスタイルは下記の通りです。

```
<Style TargetType="customItems:ViewerGridItem">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="customItems:ViewerGridItem">
        <Border BorderBrush="{StaticResource TooltipBorderThemeBrush}"
          BorderThickness="1">
          <Grid Background="White">
            <Image
              Width="{TemplateBinding ImageWidth}"
              Height="{TemplateBinding ImageHeight}"
              Margin="5"
              Source="{Binding _Source}"
              Stretch="UniformToFill"
              HorizontalAlignment="Center"
            />
            <Border Background="#A5000000"
              Height="25"
              VerticalAlignment="Bottom">
              <TextBlock
                HorizontalAlignment="Right"
                Text="{Binding PageNumber}"
                FontFamily="Segoe UI"
                FontSize="14"
                Padding="5"
                Foreground="#CCFFFFFF"/>
            </Border>
          </Grid>
        </Border>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```



Generic.xaml のコードを少し変えてみます。

```

<Style TargetType="customItems:ViewerGridItem">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="customItems:ViewerGridItem">
        <Border BorderBrush="{StaticResource TooltipBorderThemeBrush}"
          BorderThickness="3">
          <Grid Background="White">
            <Image
              Width="{TemplateBinding ImageWidth}"
              Height="{TemplateBinding ImageHeight}"
              Margin="5"
              Source="{Binding _Source}"
              Stretch="UniformToFill"
              HorizontalAlignment="Center"
            />
            <Border Background="#A57F3300"
              Height="40"
              VerticalAlignment="Bottom">
              <TextBlock
                HorizontalAlignment="Center"
                Text="{Binding PageNumber}"
                FontFamily="Segoe UI"
                FontSize="18"
                Padding="5"
                Foreground="#CCFFFFFF"/>
            </Border>
          </Grid>
        </Border>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>

```

変更後、コントロールの見た目が変わったのがわかります。



アプリケーションのテーマ変更

レポートビューアはアプリケーションが使用しているテーマから塗りつぶしの色を取得します。

App.xaml とコードの両方からアプリケーションのテーマを変更できます。

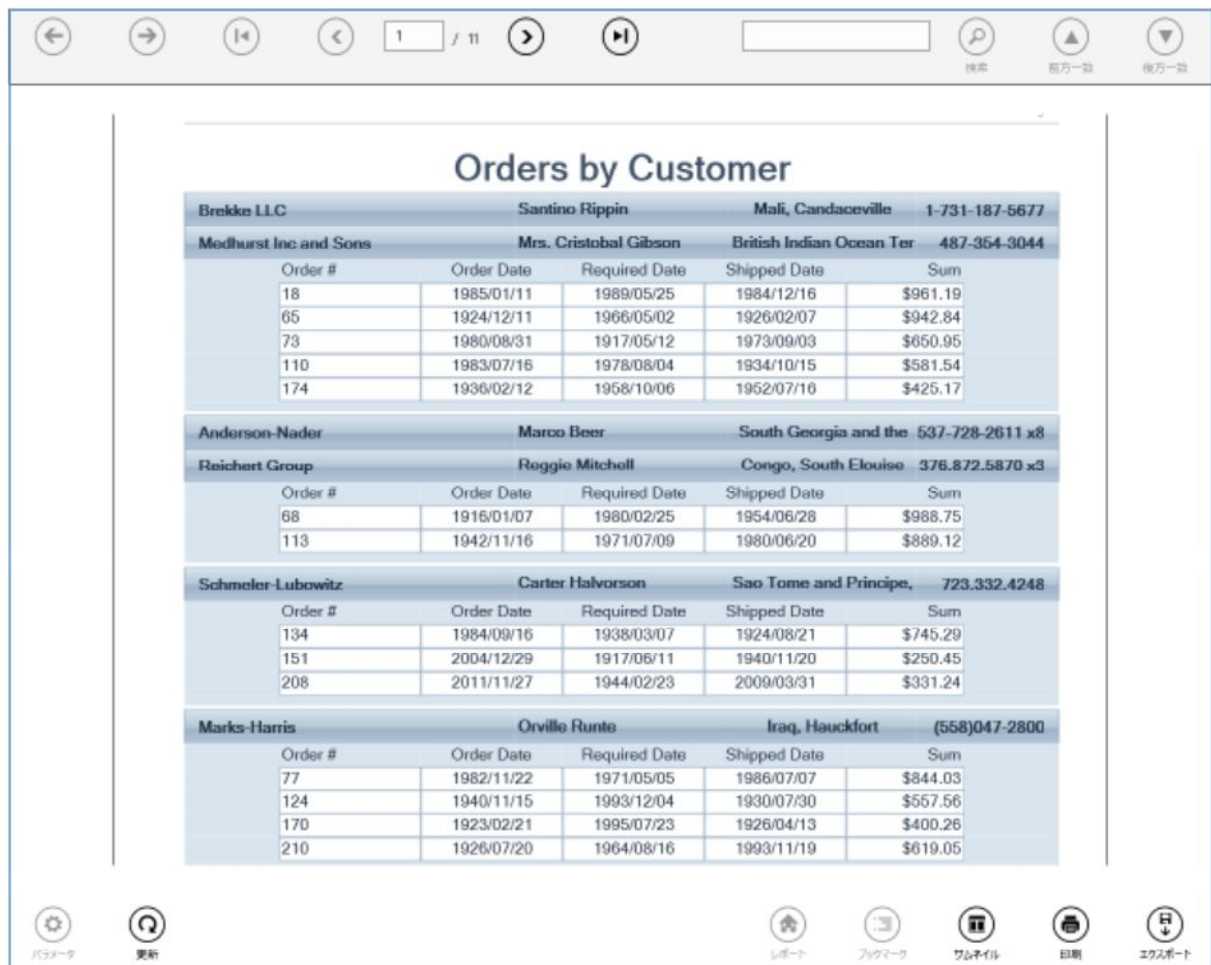
XAML の場合 :

```
<Application
...
    RequestedTheme="Light">
```

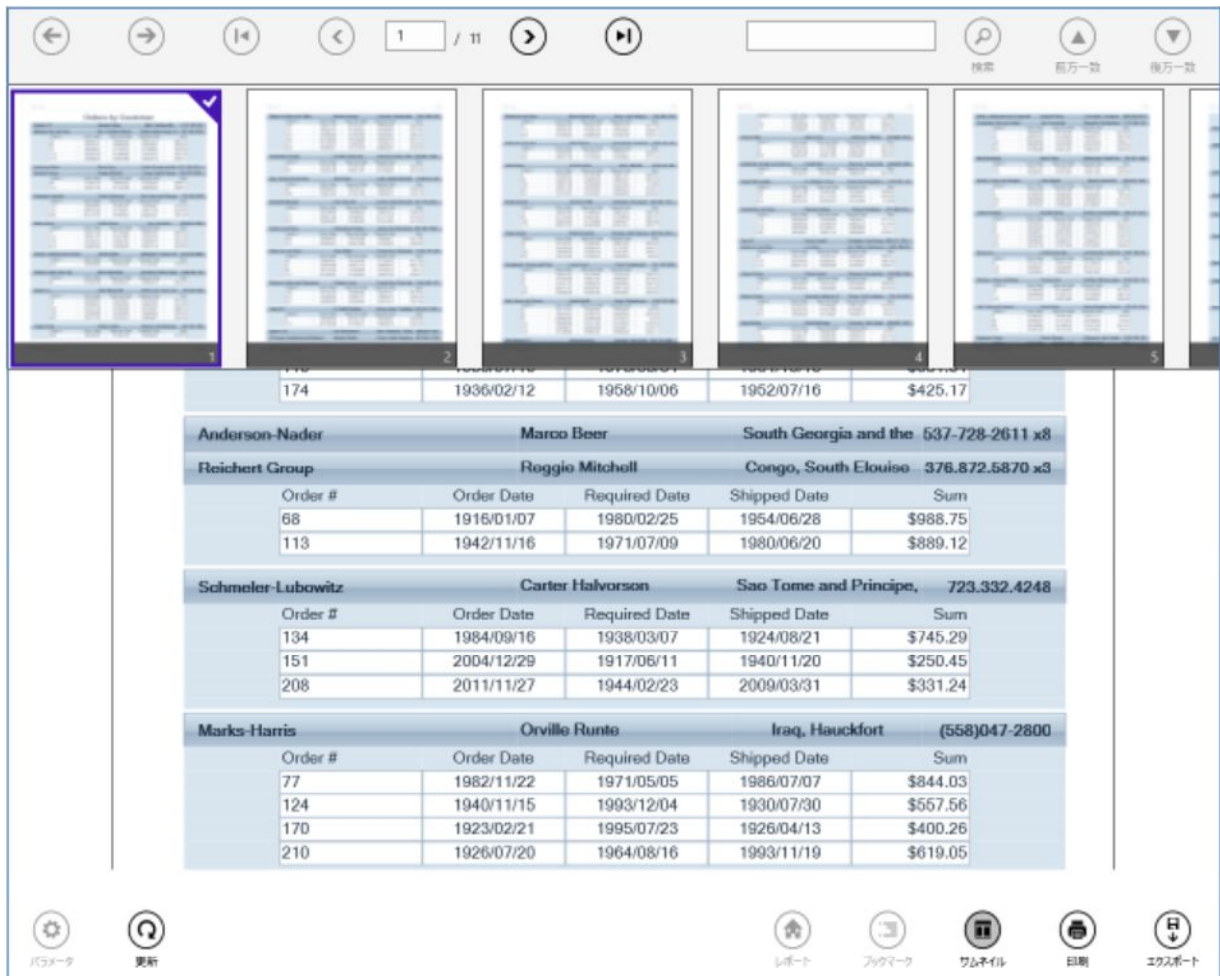
コードの場合 :

```
public App()
{
    this.InitializeComponent();
    App.Current.RequestedTheme = ApplicationTheme.Light;
}
```

レポートビューアの LightTheme の見た目は次の通りです。



Order #	Order Date	Required Date	Shipped Date	Sum
Brekke LLC Santino Rippin Mali, Candaceville 1-731-187-5677				
Medhurst Inc and Sons Mrs. Cristobal Gibson British Indian Ocean Ter 487-354-3044				
18	1985/01/11	1989/05/25	1984/12/16	\$961.19
65	1924/12/11	1966/05/02	1926/02/07	\$942.84
73	1980/08/31	1917/05/12	1973/09/03	\$650.95
110	1983/07/16	1978/08/04	1934/10/15	\$581.54
174	1936/02/12	1958/10/06	1952/07/16	\$425.17
Anderson-Nader Marco Beer South Georgia and the 537-728-2611 x8				
Reichert Group Reggie Mitchell Congo, South Elouise 378.872.5870 x3				
68	1916/01/07	1980/02/25	1954/06/28	\$988.75
113	1942/11/16	1971/07/09	1980/06/20	\$889.12
Schmeler-Lubowitz Carter Halvorson Sao Tome and Principe, 723.332.4248				
134	1984/09/16	1938/03/07	1924/08/21	\$745.29
151	2004/12/29	1917/06/11	1940/11/20	\$250.45
208	2011/11/27	1944/02/23	2009/03/31	\$331.24
Marks-Harris Orville Runte Iraq, Hauckfort (558)047-2800				
77	1982/11/22	1971/05/05	1986/07/07	\$844.03
124	1940/11/15	1993/12/04	1930/07/30	\$557.56
170	1923/02/21	1995/07/23	1926/04/13	\$400.26
210	1926/07/20	1964/08/16	1993/11/19	\$619.05



レポートビューアのローカライズ

レポートビューアのインターフェイスをローカライズするには、次の手順を行う必要があります。

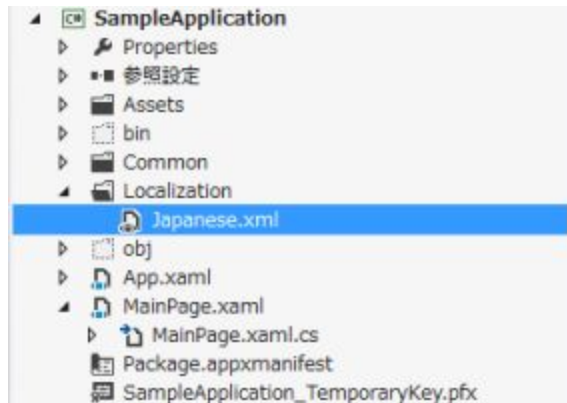
1. (多分、既存のファイルを翻訳して) すべてのローカライズ文字列を含んだ xml ファイルを作成します。ファイル構造は以下の通りです。

```
<?xml version="1.0" encoding="utf-8" ?>
<dictionary>
  <entry key="Progress.PageLoading">Page is loading...</entry>
  ...
</dictionary>
```

ローカライゼーションファイルは、Localization\WinRT フォルダにあります。

一部の文字列だけローカライズする必要がある場合は、xml ファイルの必要なキーだけ指定できます。残りの文字列はデフォルトの値を使用します。

- 次に、レポートビューアを使用するアプリケーションにローカライゼーションファイルを追加します。



- レポートビューアがローカライゼーションファイルを使用するよう、MainPage.xaml.cs ファイルに次のディレクティブ:

```
using System.Xml;
using PerpetuumSoft.WinRT.Viewer.Localization;
```

とコード（黄色い部分）を追加します。

```
public MainPage()
{
    this.InitializeComponent();
    XmlReader reader = XmlReader.Create(@"Localization/Japanese.xml");
    LocalizationManager.RegisterLocalizationProvider(new
        XMLLocalizationProvider(reader));
    Loaded += reportViewer_Loaded;
    SubscribeSharing();
}
```

すべての手順を行うと、レポートビューアは新しいローカライゼーション文字列を使用します。

Navigation: 1 / 11

Orders by Customer

Lowe-Volkman		Elsie Welch		Botswana, Douglasside 133.645.3533 x3	
Order #	Order Date	Required Date	Shipped Date	Sum	
97	1991/12/16	1929/05/05	1936/05/24	\$290.17	
210	1946/01/04	1942/08/25	2009/04/17	\$771.04	

Koch, Fritsch and Mayerl		Jeromy Lubowitz		Uganda, New Kobelon 003.503.5541 x7	
Order #	Order Date	Required Date	Shipped Date	Sum	
10	1972/10/17	1962/05/24	1976/03/20	\$64.75	
61	1993/12/15	1961/07/22	1979/09/26	\$846.43	
125	1925/11/18	1993/09/02	1975/05/27	\$955.70	
213	1914/09/26	1974/11/07	1991/11/03	\$277.70	

Haloy-Kohler		Daron Oberbrunner		Saint Helena, Mohrberg 851.377.5101 x8	
Order #	Order Date	Required Date	Shipped Date	Sum	
94	1916/06/27	2004/01/07	1974/08/07	\$947.10	
152	1984/05/31	1996/04/16	1983/10/26	\$916.18	
180	1934/04/13	1993/01/25	1938/01/08	\$636.05	

Bartell, Rice and Erdman		Enola Schamberger		Iraq, West Diamond (263)426-5202	
Order #	Order Date	Required Date	Shipped Date	Sum	
75	1937/09/10	1979/04/15	1979/11/22	\$324.20	
138	1954/10/09	1988/08/24	1930/07/11	\$384.38	
243	1940/02/10	1975/03/31	1942/07/24	\$715.86	

Hermann, Schuster and Hermann		Pietro Bergstrom		Grenada, West Randalls (338)132-4075	
Order #	Order Date	Required Date	Shipped Date	Sum	
162	1958/01/07	2007/08/14	2011/09/08	\$687.97	

Navigation: 戻る, 前ページ, 後ページ, 検索

Bottom Bar: 印刷, 更新, レポート, フラグマーク, サムネイル, 印刷, エクスポート

PDF エクスポート設定

共通設定 | セキュリティ設定

ページ範囲

すべて

ページ 頁 1-11 / 11 頁

イメージ設定

オリジナルイメージの解像度と品質を保持

カスタム解像度 %

イメージ品質 %

イメージ再利用 (MDS 方式によるバージョンごとの比較を使用)

圧縮

エクスポートした後に開く

Buttons: エクスポート, キャンセル