ユーザーズマニュアル

目次

第 1 章 ExcelCreator 5.0 for .NET について

1.　ExcelCreator 5.0 for .NET の概要

ExcelCreator 5.0 for .NET は、様々なアプリケーションの使用用途に対応するため、以下の Excel ファイル生成機能を提供します。

1. Excel ファイルを新規作成
2. 既存の Excel ファイルを読み込み専用でオープン
3. 既存の Excel ファイルを読み書きオープン
4. オーバーレイ機能について
5. オーバーレイオープン（埋め込みリソース）
6. Excel ファイルの切り替えオープン
7. データの設定 / 取得
8. 属性の設定
9. コピー / 貼り付け / クリア / 削除
10. PDF ファイルの出力

2. システム動作環境

OS	Windows 2000 (SP3 以上) / XP (SP1 以上) / Server 2003 ※.NET Framework 1.0 (SP3 以上) / 1.1 (SP1 以上) / 2.0 が必要です。 ※ASP.NET で使用する場合、IIS5.0 以上が必要です。
ハードディスク	20 MB 以上
メモリ	256 MB 以上
開発環境	Visual Studio .NET (SP1 以上) / Visual Studio .NET 2003 Visual Studio 2005
開発言語	Visual Basic.NET / Visual C# .NET (Windows アプリケーション / ASP.NET アプリケーション)
対応する Excel バージョン	Excel 97 / 2000 / 2002(XP) / 2003

3. ExcelCreator 5.0 for .NET が対応する Excel の機能

ExcelCreator 5.0 for .NET で Excel ファイルの作成と編集を行う時の対応する Excel の機能を表しています。【 ○:対応 ×:未対応 】

※対応可能の機能でも一部制限のあるプロパティ / メソッドがあります。各プロパティ / メソッドの注意点もあわせてご確認ください。
※表に記載されていない Excel の機能はサポート窓口までお問い合わせください。サポート窓口はサポートサービスについて (P.155) をご覧ください。

【項目の説明】
新規作成と編集　：　新規作成と編集 ExcelCreator 5.0 for .NET で Excel ファイルの作成と編集、PDF ファイルを作成を行う時の対応する Excel の機能を表します。
オーバレイ　：　ExcelCreator 5.0 for .NET でオーバーレイ機能（OpenBook , OpenBookEx , OpenBookEmbedEx メソッド）を使用して Excel ファイルや PDF ファイルを作成する時のオーバーレイ元ファイルに設定された Excel の機能をオーバレイ先の Excel ファイルに引き継ぐことができるかを表します。

Excelとの機能対応表

	新規作成と編集	オーバーレイ
Excel での設定…[ファイル] – [ページ設定] ページ		
印刷の向き（縦/横）	○ (Size / SizeFree メソッド)	○
拡大縮小印刷（拡大 / 縮小 10 ～ 400 %）	○ (Size メソッド)	○
拡大縮小印刷（横 1～32767 × 縦 1 ～ 32767 ページ）	×	×
用紙サイズ	○ (Size / SizeFree メソッド)	○
印刷品質	×	×
先頭ページ番号	×	×
Excel での設定…[ファイル] – [ページ設定] 余白		
上	○ (Margin メソッド)	○
下	○ (Margin メソッド)	○
左	○ (Margin メソッド)	○

	新規作成と編集	オーバーレイ
右	○ (Margin メソッド)	○
ヘッダー	○ (Margin メソッド)	○
フッター	○ (Margin メソッド)	○
ページ中央（水平）	×	×
ページ中央（垂直）	×	×
Excel での設定：[ファイル]-[ページ設定] の「ヘッダー / フッター」タブ / [表示]-[ヘッダーとフッター]		
ヘッダー	○ (Headerメソッド)	○
フッター	○ (Footerメソッド)	○
Excel での設定：[ファイル]-[ページ設定] の [シート] タブ		
印刷範囲	○ (PrintArea メソッド)	○
ページの方向（左から右へ / 上から下へ）	○ (PageOrder プロパティ)	○
Excel での設定：[編集]-[コピー]		
セルコピー	○ (Copyメソッド) (*3)	-
行コピー	○ (RowCopyメソッド) (*3)	-
列コピー	○ (ColumnCopyメソッド) (*3)	-
Excel での設定：[編集]-[貼り付け]		
セル貼り付け	○ (Pasteメソッド) (*3)	-
行貼り付け	○ (RowPasteメソッド) (*3)	-
列貼り付け	○ (ColumnPasteメソッド) (*3)	-
Excel での設定：[編集]-[クリア]-[すべて]		
セルクリア	○ (Clearメソッド)	-
行クリア	○ (RowClearメソッド)	-
列クリア	○ (ColumnClearメソッド)	-
Excel での設定：[編集]-[削除]		
行削除	○ (RowDeleteメソッド)	-
列削除	○ (ColumnDeleteメソッド)	-
Excel での設定：[編集]-[シートの削除] / [シートの移動またはコピー]		
ワークシートの削除	○ (DelSheetメソッド)	-
ワークシートの移動	×	-

	新規作成と編集	オーバーレイ
ワークシートのコピー	○ (CopySheetメソッド) (*4)	−
Excel での設定:[表示]-[ズーム]		
倍率	○ (Zoomプロパティ)	○
Excel での設定:[挿入]-[行] / [列] / [改ページ]		
行挿入	○ (RowInsertメソッド)	−
列挿入	○ (ColumnInsertメソッド)	−
ワークシート挿入	○ (AddSheetメソッド) (*2)	−
改ページ挿入	○ (Breakプロパティ)	−
Excel での設定:[書式]-[セル] の [表示形式] タブ		
[表示形式]-分類	○ (Formatプロパティ) (*1)	○
Excel での設定:[書式]-[セル] の [配置] タブ		
[配置] - 横位置- 標準	○ (PosHorzプロパティ)	○
[配置] - 横位置- 左詰め	○ (PosHorzプロパティ)	○
[配置] - 横位置- 中央揃え	○ (PosHorzプロパティ)	○
[配置] - 横位置- 右詰め	○ (PosHorzプロパティ)	○
[配置] - 横位置- 繰り返し	○ (PosHorzプロパティ)	○
[配置] - 横位置- 両端揃え	○ (PosHorzプロパティ)	○
[配置] - 横位置- 選択範囲内で中央	○ (PosHorzプロパティ)	○
[配置] - 横位置- 均等割付	○ (PosHorzプロパティ)	○
[配置] - 縦位置- 上詰め	○ (PosVertプロパティ)	○
[配置] - 縦位置- 中央揃え	○ (PosVertプロパティ)	○
[配置] - 縦位置- 下詰め	○ (PosVertプロパティ)	○
[配置] - 縦位置- 両端揃え	○ (PosVertプロパティ)	○
[配置] - 縦位置- 均等割り付け	○ (PosVertプロパティ)	○
[配置] - 折り返して全体を表示する	○ (OverReturnプロパティ)	○
[配置] - 縮小して全体を表示する	○ (Fitプロパティ)	○
[配置] - セルを結合する	○ (Jointプロパティ)	○
[配置] - 方向- 縦書き	○ (PosTurnプロパティ)	○

	新規作成と編集	オーバーレイ
[配置] - 方向- －90 ～ 90 度	○ (PosTurnプロパティ)	○
Excel での設定:[書式]-[セル] の [配置] タブ		
フォント名	○ (FontNameプロパティ)	○
スタイル（標準 / 斜体 / 太字 / 太字斜体）	○ (FontStyleプロパティ)	○
サイズ（1 ～ 409）	○ (FontPointプロパティ)	○
下線（なし / 下線 / 二重下線）	○ (FontULineプロパティ)	○
色	○ (FontColorプロパティ)	○
文字飾り（取り消し線）	○ (FontStyleプロパティ)	○
文字飾り（上付き）	○ (FontStyleプロパティ)	○
文字飾り（下付き）	○ (FontStyleプロパティ)	○
Excel での設定:[書式]-[セル] の [罫線] タブ		
プリセット（外枠）	○ (Boxメソッド)	○
罫線（上）	○ (LineTopプロパティ)	○
罫線（下）	○ (LineBottomプロパティ)	○
罫線（左）	○ (LineLeftプロパティ)	○
罫線（右）	○ (LineRightプロパティ)	○
罫線（左上がり）	○ (LineLeftUpプロパティ)	○
罫線（右上がり）	○ (LineRightUpプロパティ)	○
線（スタイル）	○	○
線（色）	○	○
Excel での設定:[書式]-[セル] の [パターン] タブ		
セルの網掛け（色）	○ (BackColorプロパティ)	○
セルの網掛け（パターン）	○ (Patternプロパティ)	○
Excel での設定:[書式]-[行] / [列]		
高さ（行）	○ (RowHeightプロパティ)	○
幅（列）	○ (ColWidthプロパティ)	○
表示しない / 再表示	×	○
Excel での設定:[書式]-[シート]		
名前の変更	○ (SheetNameメソッド)	-
表示しない / 再表示	×	○
Excel での設定:[ツール]-[オプション] の [表示] タブ		
ウィンドウオプション（枠線）	○ (ModeGridプロパティ)	○
Excel での設定:[ツール]-[オプション] の [全般] タブ		

	新規作成と編集	オーバーレイ
設定（新しい Excel ファイルのシート数）	○ (*1)	－
設定（標準フォント名）	○ (DefFontNameプロパティ)	○
設定（標準フォントサイズ）	○ (DefFontPointプロパティ)	○
*1：CreateBook / CreateBookEx メソッドで新規に Excel ファイルを作成する場合のみ、シート数を設定することができます。 *2：既存のシートの最後に新しいシートを追加します。 *3：同一シートのコピー / 貼り付けのみに対応しています。 *4：同一 Excel ファイルのシートコピーのみに対応しています。		

4. インストール方法

◆インストール時の注意事項

インストールを行うときは、他の起動しているアプリケーションを終了させた状態で行ってください。
特に、ウイルスチェックソフトなど、マシンの起動時からメモリ上に常駐しているソフトは、見落としやすいので注意が必要です。

◆インストール方法

① 製品パッケージに同梱された製品 CD-ROM を、CD-ROM ドライブにセットします。

② セットアップウィザードが起動し、インストール画面が表示されるので、画面に従いインストールを行ってください。
CD-ROM をセットしてもセットアップウィザードが起動しない場合は、マイコンピュータやエクスプローラから CD-ROM 内の Setup.exe をダブルクリックし、手動で起動させてください。

③ インストールの途中、製品のシリアル No を入力する画面が表示されます。製品 CD-ROM ケースの裏面に記載されている シリアル No を入力してください。

5. アンインストール方法

アンインストールは、コンポーネントパネルの「アプリケーションの追加と削除」から行なってください。
「アプリケーションの追加と削除」で「ExcelCreator 5.0 for .NET」を選択し、「追加と削除」ボタンをクリックすることで削除することができます。また、「スタート」－「プログラム」－「ExcelCreator 5.0 for .NET」－「アンインストール」から、削除することもできます。

6. アプリケーションのライセンス体系

ExcelCreator 5.0 for .NET コンポーネントを組み込んだアプリケーションの配布は、配布するアプリケーションの種類によってライセンス体系が異なります。

◆Windows アプリケーションの場合

ExcelCreator 5.0 for .NET ランタイムファイルの再配布は、ロイヤリティーフリーです。

◆Web アプリケーションの場合

ExcelCreator 5.0 for .NET ランタイムファイルの再配布は、配布するサーバー機の台数分のサーバーライセンスが必要です。

◇ サーバーライセンス
ExcelCreator 5.0 for .NET ランタイムファイルを IIS や CGI などのプロセス上で動作（不特定多数のクライアントからアクセス）する Web アプリケーションで使用する時に必要なライセンスです。

価　　格 ： 1 サーバーあたり ¥126,000-（税抜:¥120,000-）
製品構成 ： ライセンス数や契約内容を記載した契約書のみ
保守期間 ： 最初の 1 年間は無償。期間延長の場合は有償。

サーバーライセンスのお問い合わせは、下記の営業窓口をご利用ください。

電話番号　0776-21-9008
FAX 番号　0776-21-9022
E-Mail　info@adv.co.jp

第 2 章　コンポーネントをプロジェクトに組み込む方法

1. フォームにコンポーネントを配置する方法

Visual Studio .NET のツールボックスに追加

フォームにコンポーネントを配置する場合、Visual Studio .NET の［ツールボックス］に ExcelCreator 5.0 for .NET コンポーネントを追加します。

・Visual Studio 2005 の場合

メニュー［ツール］-［ツールボックス アイテムの選択］をクリックして［ツールボックス アイテムの選択］画面を開きます。

・Visual Studio .NET 2003 の場合

メニュー［ツール］-［ツールボックスのカスタマイズ］をクリックして［ツールボックスのカスタマイズ］画面を開きます。

・Visual Studio .NET 2002 の場合

メニュー［ツール］-［ツールボックス アイテムの追加と削除］をクリックし、［ツールボックスのカスタマイズ］画面を開きます。

［ツールボックスのカスタマイズ］画面、もしくは［ツールボックス アイテムの選択］画面にて、下記のコンポーネント名のチェックを入れて［OK］ボタンをクリックします。

名前	名前空間 / アセンブリ名
XlsCreator	ExcelCreator

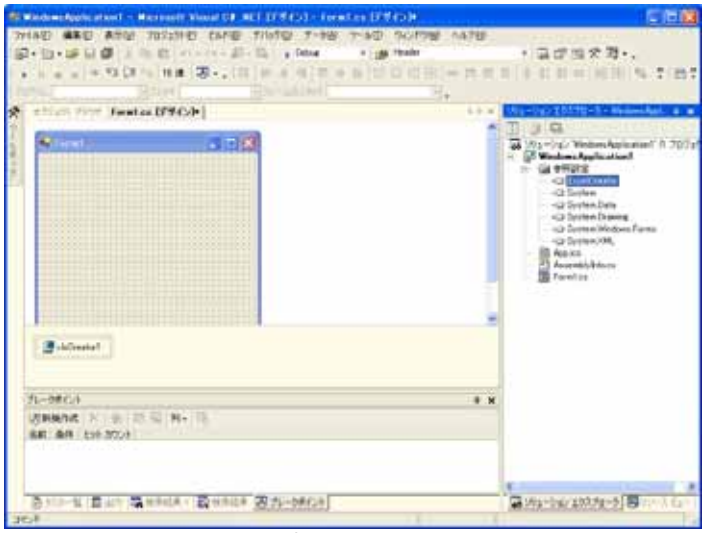
※Visual Studio 2005 の画面です。

Visual Studio .NET の［ツールボックス］に追加されていれば完了です。

フォームにコンポーネントを配置

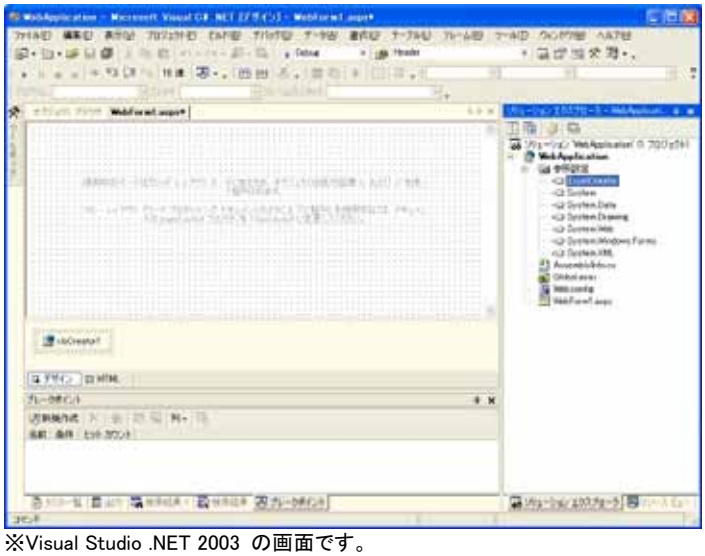
[ツールボックス]から指定した ExcelCreator 5.0 for .NET コンポーネント「xlsCreator」のアイコンを選択して、フォームに配置します。
プロジェクトのソリューションエクスプローラの参照設定に配置したコンポーネントの名前空間が追加されます。
(Visual Studio .NET の[ツールボックス]に ExcelCreator 5.0 for .NET コンポーネントを追加する操作は Visual Studio .NET のツールボックスに追加を参照してください。)

・Windows フォーム



※Visual Studio 2005 の画面です。

・Web アプリケーション



※Visual Studio .NET 2003 の画面です。

Visual Studio 2005 の場合、プロジェクトの参照設定に ExcelCreator 5.0 for .NET コンポーネントを追加し、プログラム中で ExcelCreator 5.0 for .NET コンポーネントのインスタンスを生成して使用してください。
(プロジェクトの参照設定に追加する操作は、参照設定に追加する方法を参照してください。)

2. 参照設定に追加する方法

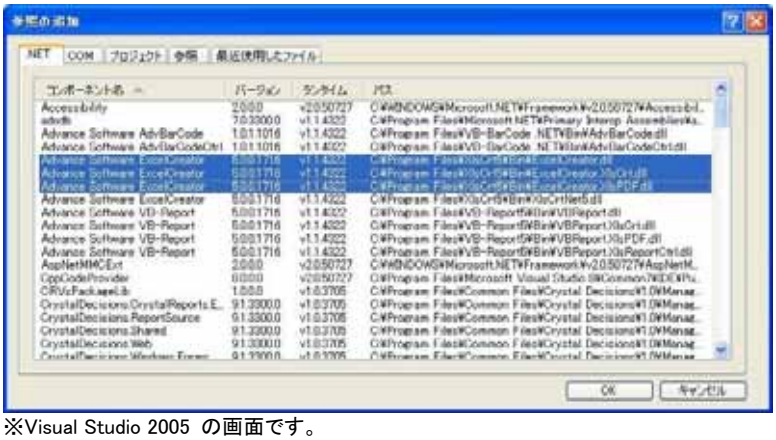
プロジェクトの参照設定に ExcelCreator 5.0 for .NET コンポーネントを追加し、プログラム中で ExcelCreator 5.0 for .NET コンポーネントのインスタンスを生成して使用する方法です。

1) Visual Studio .NET のメニュー［プロジェクト]-[参照の追加］で［参照の追加］画面を開きます。

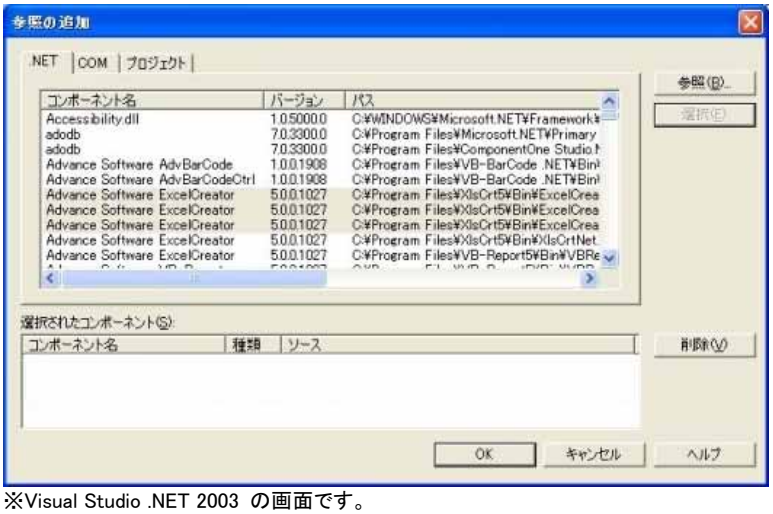
2) [参照の追加］画面の［.NET］タブを選択します。

3) 下記の ExcelCreator 5.0 for .NET コンポーネントを選択します。

コンポーネント名	パス
Advance Software ExcelCreator	¥<User Folder>¥XlsCrt5¥Bin¥ExcelCreator.dll
Advance Software ExcelCreator	¥<User Folder>¥XlsCrt5¥Bin¥ExcelCreator.XlsCrt.dll
Advance Software ExcelCreator	¥<User Folder>¥XlsCrt5¥Bin¥ExcelCreator.XlsPDF.dl

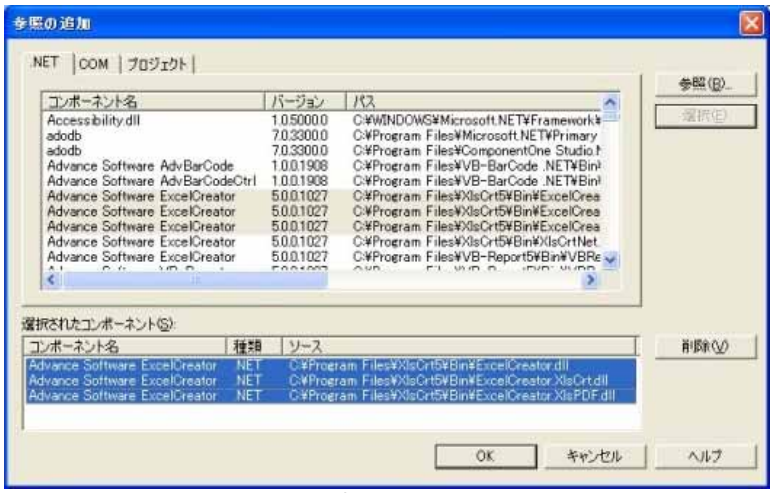


※Visual Studio 2005 の画面です。



※Visual Studio .NET 2003 の画面です。

4) Visual Studio .NET 2002 / Visual Studio .NET 2003 の場合は、上記のコンポーネントを選択した状態で［選択］ボタンをクリックし、［選択したコンポーネント］に追加します。

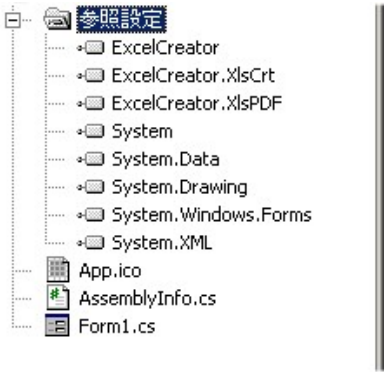


※Visual Studio .NET 2003 の画面です。

Visual Studio 2005 の場合は、選択した状態で次に進みます。

5) ［OK］ボタンをクリックし、［参照の追加］画面を閉じます。

6) プロジェクトのソリューションエクスプローラの参照設定に選択したコンポーネントが追加されます。



7) 参照設定で追加したコンポーネントのインスタンスをプログラム中で生成します。

◆コーディング例

[VB.NET]

```
Dim XlsCreator1 As New ExcelCreator.XlsCreator

XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile01.xls", 1,
ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A5").Long = 50000
XlsCreator1.CloseBook(True)

XlsCreator1.Dispose()
```

[C#]
```
private ExcelCreator.XlsCreator xlsCreator1 = new ExcelCreator.XlsCreator();

xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile01.xls", 1,
ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A5").Long = 50000;
xlsCreator1.CloseBook(true);

XlsCreator1.Dispose();
```

第 3 章　基本的な使用方法

1. Excel ファイルを新規作成

Excel ファイルを新規作成することができます。シート数や Excel ファイルバージョンも同時に指定できます。

シート数 3、Excel 2003 形式で、A1 セルに文字列を設定した Excel ファイルを新規作成します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StatupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Value = "アドバンスソフトウェア株式会社"
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StatupPath + @"¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1).Value = "アドバンスソフトウェア株式会社";
xlsCreator1.CloseBook(true);
```

2. 既存の Excel ファイルを読み込み専用でオープン

既存の Excel ファイルを読み込み専用でオープンし、セルのデータ取得を行うことができます。

既存の Excel ファイルを読み込み専用でオープンし、セルのデータ取得を行います。

[VB.NET]
```
XlsCreator1.ReadBook(Application.StatupPath & "¥ExcelFile.xls")
Dim strData As String = XlsCreator1.Cell("C2").Str
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.ReadBook(Application.StatupPath + @"¥ExcelFile.xls");
string strData = xlsCreator1.Cell("C2").Str;
xlsCreator1.CloseBook(true);
```

3. 既存の Excel ファイルを読み書きオープン

既存の Excel ファイルを読み書きオープンすることができます。
セルへデータ設定を行いながら、データ取得することができます。

既存の Excel ファイルを読み書きオープンし、セルのデータの設定や取得を行います。

[VB.NET]

```
XlsCreator1.OpenBook(Application.StatupPath & "¥ExcelFile.xls", "")
XlsCreator1.Cell("A1").Str = "アドバンスソフトウェア株式会社"
Dim strData As String = XlsCreator1.Cell("A1").Str
XlsCreator1.CloseBook(True)
```

[C#]

```
xlsCreator1.OpenBook(Application.StatupPath + @"¥ExcelFile.xls", "");
xlsCreator1.Cell("A1").Str = "アドバンスソフトウェア株式会社";
string strData = xlsCreator1.Cell("A1").Str;
xlsCreator1.CloseBook(true);
```

4. オーバーレイ機能

オーバーレイ機能について

Excel シート内にあらかじめ変数文字を記述し、プログラム内で値を変数に代入することにより、シートにデータの埋め込みを行うことができます。これにより、あらかじめ変数定義済みのセルを利用したグラフのシートを作成しておくとプログラム上で変数に値を設定するだけでグラフが作成することができます。毎月の業務アプリケーションなどで使用すると、月々のグラフ作成を定型業務化することもできます。

オーバーレイ機能の使用方法は、オーバーレイ機能の使用方法 (P.14)を参照してください。

◆オーバーレイ機能の変数名の制限事項

- ✓ 変数名にする
  変数名として使用する場合は変数名の先頭に、変数名の先頭キーワード文字列として半角のアスタリスク (*) を 2 個つけます。

- ✓ 変数名使用可能文字
  以下に示す文字以外、使用することができます。変数名として設定可能文字列は、変数名の先頭キーワード文字列 ** を含め 255 文字までです。
  ¥ ' " , ^ . [ ] (いずれも半角です)

- ✓ 変数の設定数
  変数の設定個数に制限はありません。

- ✓ 変数名の設定
  変数名を ExcelCreator 5.0 for .NET の文字列設定で記述する場合、オーバーレイ元ファイルを元に作成されるファイルについては変数名は無効となりますが、オーバーレイ元ファイルに直接上書き設定する場合には、変数名は設定可能です。

- ✓ 変数名の先頭キーワード文字列を設定
  KeyWord プロパティで設定します。デフォルトは ** です。

◆オーバーレイ機能設定可能情報

オーバーレイの元となる Excel シートに設定可能な情報は、ExcelCreator 5.0 for .NET が対応する Excel の機能 (P.2)をご参照ください。

オーバーレイ機能の使用方法

1. オーバーレイ用の変数を記述した Excel ファイルを作成します。
2. プログラムの記述を行います。
   ① OpenBook メソッド / OpenBookEx メソッド / OpenBookEmbedEx メソッドを使用し、作成するファイル名、元となるオーバーレイ元ファイル名を指定します。
   ② 設定対象シートを SheetNo プロパティで選択します。変数への値の設定は、現在の対象シートへの設定となります。 複数シートにわたって変数名を使用している場合、SheetNo プロパティで対象シートを切り替えて値を設定します。
   ③ Cell クラスの引数に直接オーバーレイ元ファイルに記述した変数名を文字列式で記述します。
   ④ オーバーレイ元ファイルに変数名を記述し、プログラム中で変数に値を設定しなかった場合、作成されるファイルの変数位置には、何も入りません。
   ⑤ CloseBook メソッド / CloseBookEx メソッドで、ファイルのクローズを行います。

✧ 設定例

ExcelTemplate.xls をオーバーレイ元ファイルとして、ExcelFile.xls を作成します。
オーバーレイ元ファイル ExcelTemplate.xls の内容は、次のとおりです。
（A1 セルに変数 **AAA 、B1 セルに変数 **BBB を設定しています。）

・ExcelTemplate.xls

	A	B	C	D
1	**AAA	**BBB		
2				
3				
4				
5				

このファイルを元に ExcelFile.xls を作成します。
（A1 セルに値 123 、C1 セルに文字列 **CCC を入れます。）

```
XlsCreator1.OpenBook(Applicaiotn.StartopPath + "¥ExcelFile.xls", _
                     Applicaiotn.StartopPath + "¥ExcelTemplate.xls")
XlsCreator1.Cell("**AAA").Long = 123      '**AAA と書かれたセルに値を設定します
XlsCreator1.Cell("C1").Str = "**CCC"      'C1 セルに変数名を指定します
XlsCreator1.CloseBook(True)
```

実行結果、下記の Excel ファイルが作成されます。
・ExcelFile.xls

	A	B	C	D
1	123			
2				
3				
4				
5				

変数 **AAA が設定されている A1 セルには、値 123 が入ります。
変数 **BBB が設定されている B1 セルには何も値を設定していないので、オーバーレイ元ファイルを元に作成された ExcelFile.xls には、何も入りません。
C1 セルに設定した文字列 **CCC は変数名なので、オーバーレイ元ファイルを元に作成された ExcelFile.xls には、何も入りません。

5. 埋め込みリソースのオーバーレイ

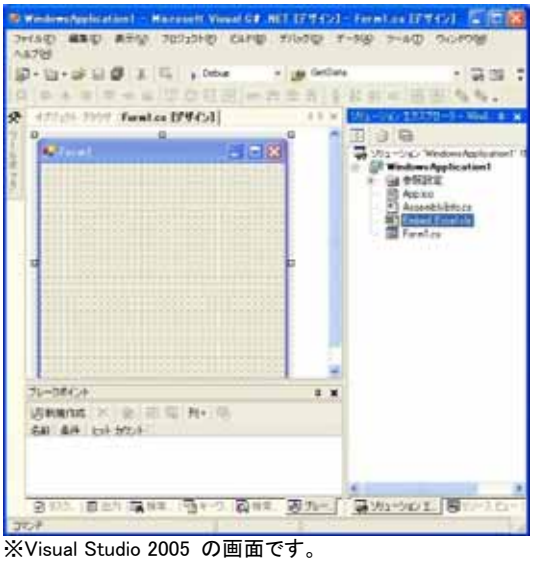
ExcelCreator 5.0 for .NET は、アプリケーションにオーバーレイ元ファイルを埋め込むことで、エンドユーザーからのデザインファイルの変更や、不用意な変更によるアプリケーションの意図しない動作を防ぐことができます。
下記のいずれかの方法でアプリケーションにデザインファイルを埋め込んで操作することができます。
※下記の手順は、Visual Studio .NET 2002 / 2003 共通です。

オーバーレイ元ファイル（Excel ファイル）を埋め込みリソースとしてプロジェクトに登録する

(1) 埋め込みリソースを登録するプロジェクトを Visual Studio .NET で開き、Visual Studio .NET のメニュー [プロジェクト]-[既存項目の追加] から、オーバーレイ元ファイル（Excel ファイル）を選択して追加します。 Visual Studio .NET のメニュー[プロジェクト]-[既存項目の追加]から、オーバーレイ元ファイルの追加を行います。
ここでは、オーバーレイ元ファイル（Embed_Excel.xls）が C:¥EmbedExcelFile フォルダにあると仮定します。
C:¥EmbedExcelFile フォルダのオーバーレイ元ファイル名（Embed_Excel.xls）を選択します。

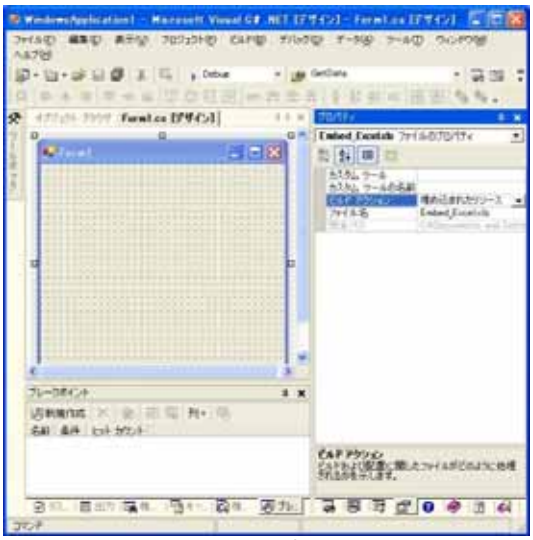
※Visual Studio 2005 の画面です。

プロジェクトに選択したオーバーレイ元ファイルが追加されます。



※Visual Studio 2005 の画面です。

(2) ソリューションエクスプローラーより、追加したオーバーレイ元ファイルを右クリックで開き、プロパティを表示します。
プロパティの［ビルドアクション］を「埋め込まれたリソース (Embedded Resource)」に変更します。



※Visual Studio 2005 の画面です。

(3) 埋め込みリソースの使用方法（コーディング例）

[VB.NET]

```
Public Class OpenBookEmbedSample
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
     Dim strOutFileName As String = Application.StartupPath & "¥OpenBookEmbed.Xls"
       ' リソース名="[プロジェクトの NameSpace].[埋め込みファイル名]"
      Dim strInFileName As String = "OpenBookEmbedSample.Embed_Excel.xls";

      ' 埋め込みリソースの読み込み
      System.Reflection.Assembly asmExcelFile = System.Reflection.Assembly.GetExecutingAssembly()
      System.IO.Stream srmExcelFile = asmExcelFile.GetManifestResourceStream(strInFileName)

      XlsCreator1.OpenBookEmbed(strOutFileName, srmExcelFile)
      XlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社"
      XlsCreator1.CloseBook(True)

      ' 読み込みリソースの解放
      srmExcelFile.Close()
   End Sub
End Class
```

[C#]

```
namespace OpenBookEmbedSample
{
   private void button1_Click(object sender, System.EventArgs e)
   {
      string strOutFileName = Application.StartupPath + @"¥OpenBookEmbed.Xls";
      // リソース名="[プロジェクトの NameSpace].[埋め込みファイル名]"
      string strInFileName = "OpenBookEmbedSample.Embed_Excel.xls";
```

```
        // 埋め込みリソースの読み込み
        System.Reflection.Assembly asmExcelFile =System.Reflection.Assembly.GetExecutingAssembly();
        System.IO.Stream srmExcelFile = asmExcelFile.GetManifestResourceStream(strInFileName);

        xlsCreator1.OpenBookEmbed(strOutFileName, srmExcelFile);
        xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社";
        xlsCreator1.CloseBook(true);

        // 読み込みリソースの解放
        srmExcelFile.Close();
    }
}
```

オーバーレイ元ファイル（Excel ファイル）を元に作成したリソースファイルを埋め込みリソースとしてプロジェクトに登録する

(1) オーバーレイ元ファイルから埋め込みリソースファイルの作成方法（コーディング例）

[VB.NET]

```
' オーバーレイ元ファイル（Excel ファイル）をバイト配列へ置き換えて読み込み
Dim fs As System.IO.FileStream = New System.IO.FileStream("C:¥EmbedExcel.xls",
System.IO.FileMode.Open)
Dim byt() As Byte = New Byte(fs.Length) {}
fs.Read(byt, 0, fs.Length)

' C:¥ResourceFile フォルダにリソースファイル（EmbedExcel.resources）作成
Dim rw As System.Resources.ResourceWriter = New
System.Resources.ResourceWriter("C:¥ResourceFile¥EmbedExcel.resources")
rw.AddResource("C:¥ResourceFile¥EmbedExcel.resources", byt)

' オーバーレイ元ファイル（Excel ファイル）とリソースファイルを閉じる
fs.Close()
rw.Generate()
rw.Close()
```

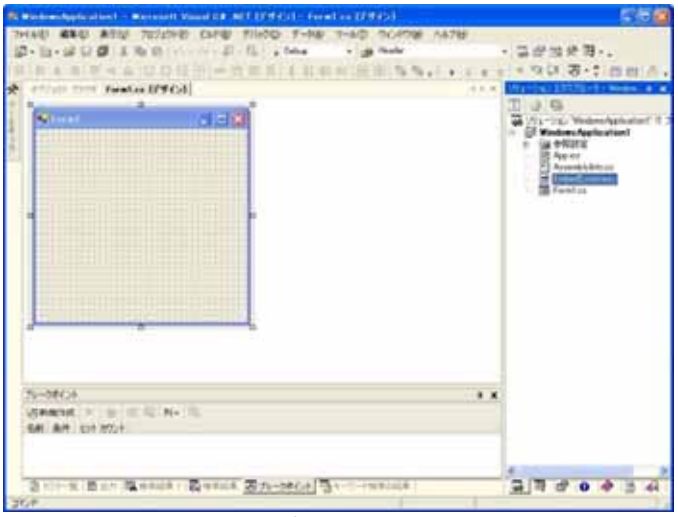
[C#]

```
// オーバーレイ元ファイル（Excel ファイル）をバイト配列へ置き換えて読み込み
System.IO.FileStream fs = new System.IO.FileStream(@"C:¥EmbedExcel.xls",
System.IO.FileMode.Open);
byte[] bytes = new byte[fs.Length];
fs.Read(bytes, 0, Convert.ToInt32(fs.Length));

// C:¥ResourceFile フォルダにリソースファイル（EmbedExcel.resources）作成
System.Resources.ResourceWriter rw = new
System.Resources.ResourceWriter(@"C:¥ResourceFile¥EmbedExcel.resources");
rw.AddResource(@"C:¥ResourceFile¥EmbedExcel.resources", bytes);

// オーバーレイ元ファイル（Excel ファイル）とリソースファイルを閉じる
fs.Close();
rw.Generate();
rw.Close();
```

(2)　埋め込みリソースを登録するプロジェクトを Visual Studio .NET で開き、Visual Studio .NET のメニュー [プロジェクト]-[既存項目の追加] から、作成したリソースファイルを選択して追加します。C:¥ResourceFile フォルダのリソースファイル（EmbedExcel.resources）を選択します。

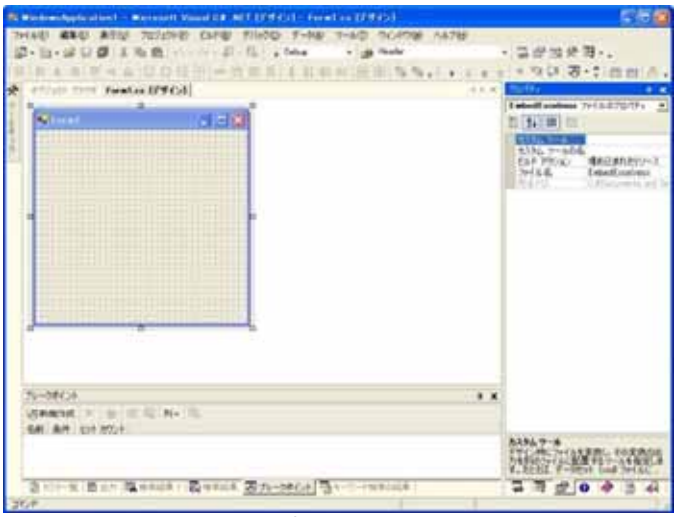
※Visual Studio 2005 の画面です。

プロジェクトに選択したリソースファイルが追加されます。



※Visual Studio 2005 の画面です。

(3)　ソリューションエクスプローラーより、追加したリソースファイルを右クリックで開き、プロパティを表示します。プロパティの［ビルドアクション］を「埋め込まれたリソース（Embedded Resource）」に変更します。



※Visual Studio 2005 の画面です。

(4)　埋め込みリソースの使用方法（コーディング例）

[VB.NET]

```
Public Class OpenBookEmbedSample
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
      Dim strOutFileName As String = Application.StartupPath & "¥OpenBookEmbed.Xls"

      Dim bytes() As Byte   ' リソース読み込みバッファ
      Dim res As System.Resources.ResourceManager

      ' 埋め込みリソースの読み込み
      ' リソース名="[プロジェクトの NameSpace].[埋め込みリソース名]"
      res = New System.Resources.ResourceManager("OpenBookEmbedSample.EmbedExcel",
Me.GetType().Assembly)
      ' リソースをバイト配列へ置き換え
      bytes = res.GetObject("ExcelFile")

      XlsCreator1.OpenBookEmbed(strOutFileName, m_bytes)
      XlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社"
      XlsCreator1.CloseBook(True)
   End Sub
End Class
```

[C#]

```
namespace OpenBookEmbedSample
{
   private void button1_Click(object sender, System.EventArgs e)
   {
      string strOutFileName = Application.StartupPath + @"¥OpenBookEmbed.Xls";
```

```
        byte[] bytes;   // リソースから読み込むバイナリバッファ
        System.Resources.ResourceManager res;

        // 埋め込みリソースの読み込み
        // リソース名="[プロジェクトの NameSpace].[埋め込みリソース名]"
        res = new System.Resources.ResourceManager("OpenBookEmbedSample.EmbedExcel",
    this.GetType().Assembly);
        // リソースをバイト配列へ置き換え
        bytes = (byte[])res.GetObject("ExcelFile");

        xlsCreator1.OpenBookEmbed(strOutFileName, bytes);
        xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社";
        xlsCreator1.CloseBook(true);
    }
}
```

6. Excel ファイルの切り替えオープン

CreateBookEx , ReadBookEx , OpenBookEx , OpenBookEmbed , OpenBookEmbedEx メソッドで開いた Excel ファイルの 識別 ID を元に、SelectBook メソッドで操作を行うファイルを選択することで、複数の Excel ファイルを扱うことができます。

[VB.NET]

```
Dim pID1 As UInt32
Dim pID2 As UInt32

' Excel ファイル新規作成オープン
' 1 つ目の Excel ファイルを新規作成オープン
XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile01.xls", 3, _
                                ExcelCreator.xlVersion.ver2003, ByRef pID1)
' 2 つ目の Excel ファイルを新規作成オープン
XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile02.xls", 3, _
                                ExcelCreator.xlVersion.ver2003, ByRef pID2)

' 1 つ目の Excel ファイルに対し編集
XlsCreator1.SelectBook(pID1)  ' 操作対象となる Excel ファイルを選択
XlsCreator1.Cell("A1").Value = "1 つ目の Excel ファイルです。"

' 2 つ目の Excel ファイルに対し編集
LOR=#FF0000>XlsCreator1.SelectBook(pID2)  ' 操作対象となる Excel ファイルを選択
XlsCreator1.Cell("A1").Value = "2 つ目の Excel ファイルです。"

' Excel ファイルクローズ
' 識別 ID を設定しクローズ
XlsCreator1.CloseBookEx(True, pID1)
XlsCreator1.CloseBookEx(True, pID2)
```

[C#]

```
UInt32 pID1 = 0;
UInt32 pID2 = 0;

// Excel ファイル新規作成オープン
// 1 つ目の Excel ファイルを新規作成オープン
xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile01.xls", 3,
                                ExcelCreator.xlVersion.ver2003, ref pID1);
```

```
// 2 つ目の Excel ファイルを新規作成オープン
xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile02.xls", 3,
                         ExcelCreator.xlVersion.ver2003, ref pID1);

// 1 つ目の Excel ファイルに対し編集
xlsCreator1.SelectBook(pID1);  // 操作対象となる Excel ファイルを選択
xlsCreator1.Cell("A1").Value = "1 つ目の Excel ファイルです。";

// 2 つ目の Excel ファイルに対し編集
xlsCreator1.SelectBook(pID2);  // 操作対象となる Excel ファイルを選択
xlsCreator1.Cell("A1").Value = "2 つ目の Excel ファイルです。";

// Excel ファイルクローズ
// 識別 ID を設定しクローズ
XlsCreator1.CloseBookEx(true, pID1);
XlsCreator1.CloseBookEx(true, pID2);
```

7. データの設定 / 取得

ExcelCreator 5.0 for .NET では、以下 4 つの方法で読み書きを行うセルをプログラム中から指定します。

1) A1 参照形式によるセルの指定
Cell クラスにより、A1 参照形式で読み書きを行うセルを指定することができます。

```
XlsCreator1.OpenBook(Applicaiotn.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.Cell("A1").Str = "A1 セルに文字列を設定"
XlsCreator1.CloseBook(True)
```

2) 座標形式によるセルの指定
Pos クラスにより、A1 セルを (0,0) として座標形式で読み書きを行うセルを指定することができます。

```
XlsCreator1.ReadBook(Applicaiotn.StartupPath & "¥ExcelFile.xls", "")
objValue = XlsCreator1.Pos(1,1).Value   'B2 セルの値を取得
XlsCreator1.CloseBook(True)
```

3) セル名によるセルの指定
Excel ファイルのセルにあらかじめ設定されたセル名を Cell クラスで指定することができます。

```
XlsCreator1.OpenBook(Applicaiotn.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.Cell("NAME").Value = "アドバンスソフトウェア"
XlsCreator1.CloseBook(True)
```

4) 変数名によるセルの指定
Excel ファイルのセルにあらかじめ記述した変数文字を Cell クラスで指定することができます。
ExcelCreator 5.0 for .NET は、先頭に半角の ＊(アスタリスク) が 2 つ付いた文字列を変数文字として認識します。
変数名をオープンする Excel ファイルのセルに設定する場合、変数名として設定可能な文字列は、先頭の ** を含め 255 文字までです。但し、¥ ' " , ^ . □ の半角文字は使用できません。

```
' オープンする Excel ファイルの A1 セルに「**Name」が設定されている場合
XlsCreator1.OpenBook(Applicaiotn.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.Cell("**Name").Value = "アドバンスソフトウェア"
```

```
XlsCreator1.CloseBook(True)
```

また、変数名の先頭のアスタリスクの部分を、別の文字列を使用したい場合、 KeyWord プロパティを使用することで、変数名のキーワードとなる先頭の ＊(アスタリスク) を任意の文字列に変更することができます。先頭キーワード文字列は Excel で文字として認識可能で、かつ、半角文字のみ設定することができます。但し、¥ ' " , ^ . [] の半角文字は使用できません。

```
' オープンする Excel ファイルの A1 セルに「!!ZipCode」が設定されている場合
XlsCreator1.Keyword = "!!"
XlsCreator1.OpenBook(Applicaiotn.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.Cell("!!ZipCode").Value = "〒918-8237"
XlsCreator1.CloseBook(True)
```

新規作成、読み書きオープンやオーバーレイでオープンした Excel ファイルのセルに、文字列、数値(整数、実数)、object 型、計算式と計算結果を設定できます。
読み書きや読み込み専用でオープンした Excel ファイルではセルのデータを文字列 / 数値 (整数 / 実数) / object 型を取得できます。計算結果は数値で取得できます。

・データの設定

新規作成した Excel ファイルにデータを設定します

[VB.NET]

```
xlsCreator1.CreateBook(Application.StatupPath & "¥ExcelFile.xls")

' 文字列を設定します
xlsCreator1.Cell("A1").Str = "アドバンスソフトウェア株式会社"

' 整数を設定します
xlsCreator1.Cell("A2").Long = 10000

' 実数を設定します
xlsCreator1.Cell("A3").Double = 156.79

' object 型を設定します
' セルには文字列で設定されます
xlsCreator1.Cell("A4").Value = "名前"
' セルには整数で設定されます
xlsCreator1.Cell("A5").Value = 25000

' 計算式と計算結果を設定します
xlsCreator1.Cell("A6").Func("=A2+A3", "10156.79")

xlsCreator1.CloseBook(True)
```

[C#]

```
xlsCreator1.CreateBook(Application.StatupPath + @"¥ExcelFile.xls");

// 文字列を設定します
xlsCreator1.Cell("A1").Str = "アドバンスソフトウェア株式会社";

// 整数を設定します
xlsCreator1.Cell("A2").Long = 10000;

// 実数を設定します
xlsCreator1.Cell("A3").Double = 156.79;
```

```
// object 型を設定します
// セルには文字列で設定されます
xlsCreator1.Cell("A4").Value = "名前";
// セルには整数で設定されます
xlsCreator1.Cell("A5").Value = 25000;

// 計算式と計算結果を設定します
xlsCreator1.Cell("A6").Func("=A2+A3", "10156.79");

xlsCreator1.CloseBook(true);
```

・データの取得

既存の Excel ファイルをオープンし、セルのデータを取得します。

[VB.NET]

```
xlsCreator1.ReadBook(Application.StatupPath & "¥ExcelFile.xls")

' 文字列を取得します
Dim strData As String = xlsCreator1.Cell("A1").Str

' 整数を取得します
Dim nNumber As Integer = xlsCreator1.Cell("A2").Long

' 実数を取得します
Dim dblNumber As Double = xlsCreator1.Cell("A3").Double

' object 型を取得します
Dim objData As Object
' 文字列が設定セルのデータを取得すると、objData に文字列で取得されます
objData = xlsCreator1.Cell("A4").Value
' 数値が設定セルのデータを取得すると、objData に数値で取得されます
objData = xlsCreator1.Cell("A5").Value

xlsCreator1.CloseBook(True)
```

[C#]

```
xlsCreator1.ReadBook(Application.StatupPath + @"¥ExcelFile.xls");

// 文字列を取得します
string strData = xlsCreator1.Cell("A1").Str;

// 整数を取得します
Int32 nNumber = xlsCreator1.Cell("A2").Long;

// 実数を取得します
double dblNumber = xlsCreator1.Cell("A3").Double;

// object 型を取得します
object objData;
// 文字列が設定セルのデータを取得すると、objData に文字列で取得されます
objData = xlsCreator1.Cell("A4").Value;
// 数値が設定セルのデータを取得すると、objData に数値で取得されます
objData = xlsCreator1.Cell("A5").Value;

xlsCreator1.CloseBook(true);
```

8. 属性の設定

新規作成、読み書きオープンやオーバーレイでオープンした Excel ファイルのセルに書式を設定することができます。
例えば、データの条件によってセルの書式をプログラム中から設定することができます。

・書式

新規作成した Excel ファイルのセルに書式を設定します。

[VB.NET]

```
xlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003)

xlsCreator1.Cell("A1").Str = "abcde"

' 指定セル内の文字列の水平位置を設定します
xlsCreator1.Cell("A1").Attr.PosHorz = ExcelCreator.xlPosHorz.phCenter
' セル内の文字列の垂直位置を設定します
xlsCreator1.Cell("A1").Attr.PosVert = ExcelCreator.xlPosVert.pvCenter
' セル内の文字列の回転角度を設定します
xlsCreator1.Cell("A1").Attr.PosTurn = ExcelCreator.xlPosTurn.ptTurn90

' セル内の列幅をこえる文字列を列幅で折り返して表示します
xlsCreator1.Cell("A2").Str = "abcdefghijklmnopqrstuvwxyz"
xlsCreator1.Cell("A2").Attr.OverReturn = True

' セル内の文字を縮小して全体を表示します
xlsCreator1.Cell("A3").Str = "ABCDEFGHIJKLMNO"
xlsCreator1.Cell("A3").Attr.Fit = True

' セルの結合を行います
xlsCreator1.Cell("A4:B4").Attr.Joint = True

xlsCreator1.CloseBook(True)
```

[C#]

```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003);

xlsCreator1.Cell("A1").Str = "abcde"

// 指定セル内の文字列の水平位置を設定します
xlsCreator1.Cell("A1").Attr.PosHorz = ExcelCreator.xlPosHorz.phCenter;
// セル内の文字列の垂直位置を設定します
xlsCreator1.Cell("A1").Attr.PosVert = ExcelCreator.xlPosVert.pvCenter;
// セル内の文字列の回転角度を設定します
xlsCreator1.Cell("A1").Attr.PosTurn = ExcelCreator.xlPosTurn.ptTurn90;

// セル内の列幅をこえる文字列を列幅で折り返して表示します
xlsCreator1.Cell("A2").Str = "abcdefghijklmnopqrstuvwxyz";
xlsCreator1.Cell("A2").Attr.OverReturn = True;

// セル内の文字を縮小して全体を表示します
xlsCreator1.Cell("A3").Str = "ABCDEFGHIJKLMNO";
```

```
xlsCreator1.Cell("A3").Attr.Fit = true;

// セルの結合を行います
xlsCreator1.Cell("A4:B4").Attr.Joint = true;

xlsCreator1.CloseBook(true);
```

・書式（フォント）

新規作成した Excel ファイルのセルにフォント書式を設定します。

[VB.NET]

```
xlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003)

xlsCreator1.Cell("A1").Str = "abcde"

' セルのフォントを設定します
xlsCreator1.Cell("A1").Attr.FontName = "MS Pゴシック"
' セルのフォントサイズを設定します
xlsCreator1.Cell("A1").Attr.FontPoint = 20
' セルのフォントの色を設定します
xlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed
' セルのフォントスタイルを設定します
xlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold
' セルのフォントの下線スタイルを設定します
xlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble

xlsCreator1.CloseBook(True)
```

[C#]

```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003);

xlsCreator1.Cell("A1").Str = "abcde"

' セルのフォントを設定します
xlsCreator1.Cell("A1").Attr.FontName = "MS Pゴシック";
' セルのフォントサイズを設定します
xlsCreator1.Cell("A1").Attr.FontPoint = 20;
' セルのフォントの色を設定します
xlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed;
' セルのフォントスタイルを設定します
xlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold;
' セルのフォントの下線スタイルを設定します
xlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble;

xlsCreator1.CloseBook(true);
```

9. コピー / 貼り付け / クリア / 削除

新規作成、読み書きオープンやオーバーレイでオープンした Excel ファイルのセルに対し、コピーや貼り付け、クリアの操作を行うことができます。
また、行や列にはコピー、貼り付け、削除、挿入の操作を行うことができます。

・セルのコピーや貼り付け、クリア

新規作成した Excel ファイルにセルのコピー、貼り付け、クリアを行います。

[VB.NET]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003)

' セルコピー元を設定
XlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn03
XlsCreator1.Cell("A1:B1").Str = "セルコピー元"

' セル範囲コピー (A1:B1 範囲セルを A2 セルにコピー)
XlsCreator1.Cell("A1:B1").Copy("A2")
' 単一セルコピー (A1 セルを C2 セルにコピー)
XlsCreator1.Cell("A1").Copy("C2")

' A1 セルをメモリ上にコピーし A3 セルに貼り付け
XlsCreator1.Cell("A1").Copy("")
XlsCreator1.Cell("A3").Paste()

' A1:B1 セルをクリアします
XlsCreator1.Cell("A1:B1").Clear()

XlsCreator1.CloseBook(True)
```

[C#]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003);

// セルコピー元を設定
XlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn03;
XlsCreator1.Cell("A1:B1").Str = "セルコピー元";

// セル範囲コピー (A1:B1 範囲セルを A2 セルにコピー)
XlsCreator1.Cell("A1:B1").Copy("A2");
// 単一セルコピー (A1 セルを C2 セルにコピー)
XlsCreator1.Cell("A1").Copy("C2");

// A1 セルをメモリ上にコピーし A3 セルに貼り付け
XlsCreator1.Cell("A1").Copy("");
XlsCreator1.Cell("A3").Paste();

// A1:B1 セルをクリアします
XlsCreator1.Cell("A1:B1").Clear();

XlsCreator1.CloseBook(true);
```

・行や列のコピー、貼り付け、削除、挿入

新規作成の Excel ファイルに行のコピー、貼り付け、削除、挿入を行います。

[VB.NET]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003)
```

```
' コピー元を設定
XlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn03
XlsCreator1.Cell("A1:B1").Str = "コピー元"

' 1 行目を 20 行目に行コピー
XlsCreator1.RowCopy(0, 19)
' 1 行目をメモリ上に行コピーし、21 行目に貼り付け
XlsCreator1.RowCopy(0, -1)
XlsCreator1.RowPaste(20)

' 単一行の挿入 (2 行目から 1 行)
XlsCreator1.RowInsert(1, 0)
' 複数行の挿入 (2 行目から 2 行分)
XlsCreator1.RowInsert(1, 2)

' 単一行の削除 (2 行目から 1 行)
XlsCreator1.RowDelete(1, 0)
' 複数行の削除 (2 行目から 2 行分)
XlsCreator1.RowDelete(1, 2)

' 単一行のクリア (2 行目から 1 行)
XlsCreator1.RowClear(1, 0)
' 複数行のクリア (2 行目から 2 行分)
XlsCreator1.RowClear(1, 2)

XlsCreator1.CloseBook(True)
```

[C#]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003);

// コピー元を設定
XlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn03;
XlsCreator1.Cell("A1:B1").Str = "コピー元";

// 1 行目を 20 行目に行コピー
XlsCreator1.RowCopy(0, 19);
// 1 行目をメモリ上に行コピーし、21 行目に貼り付け
XlsCreator1.RowCopy(0, -1);
XlsCreator1.RowPaste(20);

// 単一行の挿入 (2 行目から 1 行のみ挿入)
XlsCreator1.RowInsert(1, 0);
// 複数行の挿入 (2 行目から 2 行分挿入)
XlsCreator1.RowInsert(1, 2);

// 単一行の削除 (2 行目から 1 行のみ削除)
XlsCreator1.RowDelete(1, 0);
// 複数行の削除 (2 行目から 2 行分削除)
XlsCreator1.RowDelete(1, 2);

// 単一行のクリア (2 行目から 1 行)
XlsCreator1.RowClear(1, 0);
// 複数行のクリア (2 行目から 2 行分)
XlsCreator1.RowClear(1, 2);

XlsCreator1.CloseBook(true);
```

新規作成の Excel ファイルに列のコピー、貼り付け、削除、挿入を行います

[VB.NET]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003)

' コピー元を設定
XlsCreator1.Cell("A1:G1").Attr.Pattern = ExcelCreator.xlPattern.pn03
XlsCreator1.Cell("A1:G1").Str = "コピー元"

' 1 列目を 20 列目に列コピー
XlsCreator1.ColumnCopy(0, 19)
' 1 列目をメモリ上に行コピーし、21 列目に貼り付け
XlsCreator1.ColumnCopy(0, -1)
XlsCreator1.ColumnPaste(20)

' 単一列の挿入 (B 列から 1 列のみ)
XlsCreator1.ColumnInsert(1, 1)
' 複数列の挿入 (B 列から 2 列分)
XlsCreator1.ColumnInsert(1, 2)

' 単一列の削除 (E 列のみ)
XlsCreator1.ColumnDelete(4, -1)
'複数列の削除 (E 列から 2 列分)
XlsCreator1.ColumnDelete(4, 2)

'単一列のクリア (E 列のみ)
XlsCreator1.ColumnClear(4, -1)
'複数列のクリア (E 列から 2 列分)
XlsCreator1.ColumnClear(4, 2)

XlsCreator1.CloseBook(True)
```

[C#]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);

' コピー元を設定
XlsCreator1.Cell("A1:G1").Attr.Pattern = ExcelCreator.xlPattern.pn03;
XlsCreator1.Cell("A1:G1").Str = "コピー元";

' 1 列目を 20 列目に列コピー
XlsCreator1.ColumnCopy(0, 19);
' 1 列目をメモリ上に行コピーし、21 列目に貼り付け
XlsCreator1.ColumnCopy(0, -1);
XlsCreator1.ColumnPaste(20)

' 単一列の挿入 (B 列から 1 列のみ)
XlsCreator1.ColumnInsert(1, 1);
' 複数列の挿入 (B 列から 2 列分)
XlsCreator1.ColumnInsert(1, 2);

' 単一列の削除 (E 列のみ)
XlsCreator1.ColumnDelete(4, -1);
'複数列の削除 (E 列から 2 列分)
XlsCreator1.ColumnDelete(4, 2);
```

```
'単一列のクリア (E 列のみ)
XlsCreator1.ColumnClear(4, -1);
'複数列のクリア (E 列から 2 列分)
XlsCreator1.ColumnClear(4, 2);

XlsCreator1.CloseBook(true);
```

10. PDF ファイルの出力

CloseBook、CloseBookEx メソッド実行時に作成や編集を行った Excel ファイルを元に PDF ファイルへ出力できます。
PDF ファイルの出力先は、ファイル名 [String クラス] と [System.IO.Stream クラス] で行います。

1) ファイル名 [String クラス]
CloseBook、CloseBookEx メソッド実行時に PDF ファイル名を文字列 [String クラス] で指定し、PDF ファイル出力を行います。
指定したディスクドライブやフォルダ、ファイル名で PDF ファイルを出力することができます。

新規作成した Excel ファイルを PDF ファイルに出力します
新規作成した Excel ファイルはクローズ時に削除します

[VB.NET]

```
xlsCreator1.CreateBook(Application.StatupPath & "¥ExcelFile.xls")
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社"
xlsCreator1.CloseBook(True, Application.StatupPath & "¥PDFFile.pdf", True)
```

[C#]

```
xlsCreator1.CreateBook(Application.StatupPath + @"¥ExcelFile.xls");
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社";
xlsCreator1.CloseBook(true, Application.StatupPath + @"¥PDFFile.pdf", true);
```

ExcelTemplate.xls をオーバーレイ元ファイルとし、作成した Excel ファイルを PDF ファイルに出力します
作成した Excel ファイルは削除しません

[VB.NET]

```
xlsCreator1.OpenBook(Application.StatupPath & "¥ExcelFile.xls", _
                     Application.StatupPath & "¥ExcelTemplate.xls")
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社"
xlsCreator1.CloseBook(True, Application.StatupPath & "¥PDFFile.pdf", False)
```

[C#]

```
xlsCreator1.OpenBook(Application.StatupPath + @"¥ExcelFile.xls",
                     Application.StatupPath + @"¥ExcelTemplate.xls");
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社";
xlsCreator1.CloseBook(true, Application.StatupPath + @"¥PDFFile.pdf", false);
```

2) System.IO.Stream クラス

CloseBook、CloseBookEx メソッド実行時に PDF ファイル名を [System.IO.Stream クラス] で指定し、PDF ファイル出力を行います。

[System.IO.Stream クラス] で指定することにより、アプリケーション上で定義した
System.IO.Stream クラス] の変数に PDF ファイルを出力することができます。
ここでは、[System.IO.FileStream クラス] と [System.IO.MemoryStream クラス] の変数に PDF ファイルを出力するコーディング例を掲載しています。

新規作成した Excel ファイルを [System.IO.FileStream クラス] の変数に PDF ファイルに出力します
新規作成した Excel ファイルはクローズ時に削除します

[VB.NET]

```
Dim fileStm As System.IO.FileStream
fileStm = New System.IO.FileStream(Application.StartupPath & "¥PdfFile01.xls",
IO.FileMode.OpenOrCreate)

xlsCreator1.CreateBook(Application.StatupPath & "¥ExcelFile01.xls")
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社"
xlsCreator1.CloseBook(True, fileStm, True)

fileStrm.Close()
```

[C#]

```
System.IO.FileStream fileStrm = new FileStream(Application.StartupPath + @"¥PdfFile01.xls",
FileMode.OpenOrCreate);

xlsCreator1.CreateBook(Application.StatupPath + @"¥ExcelFile01.xls");
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社";
xlsCreator1.CloseBook(true, fileStrm, true);

fileStrm.Close();
```

ExcelTemplate.xls をオーバーレイ元ファイルとし、作成した Excel ファイルを
[System.IO.MemoryStream クラス] の変数に PDF ファイルに出力します
作成した Excel ファイルは削除しません

[VB.NET]

```
Dim memStm As System.IO.MemoryStream
memStm = New System.IO.MemoryStream

xlsCreator1.OpenBook(Application.StatupPath & "¥ExcelFile02.xls", _
                     Application.StatupPath & "¥ExcelTemplate.xls")
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社"
xlsCreator1.CloseBook(True, memStm, False)

memStrm.Close()
```

[C#]

```
System.IO.MemoryStream memStrm = new MemoryStream();

xlsCreator1.OpenBook(Application.StatupPath + @"¥ExcelFile02.xls",
                     Application.StatupPath + @"¥ExcelTemplate.xls");
xlsCreator1.Cell("C2").Value = "アドバンスソフトウェア株式会社";
xlsCreator1.CloseBook(true, memStrm, false);

memStrm.Close()
```

第 4 章 ExcelCreator 5.0 for .NET コンポーネント機能一覧

新規作成オープン

名前	内容	マニュアルページ
CreateBook	Excel ファイルの新規作成を行います。	P35
CreateBookEx	Excel ファイルを新規作成し、その識別 ID を取得します。(SelectBook メソッド専用)	P36

読み書きオープン / オーバーレイオープン

名前	内容	マニュアルページ
OpenBook	既存 Excel ファイルをオープンします。既存の Excel ファイルを元に別の Excel ファイルをオープンすることもできます。	P38
OpenBookEx	既存 Excel ファイルをオープンし、その識別 ID を取得します。既存の Excel ファイルを元に別の Excel ファイルをオープンすることもできます。(SelectBook メソッド専用)	P40

埋め込み Excel ファイルオーバーレイオープン

名前	内容	マニュアルページ
OpenBookEmbed	リソースに埋め込まれたバイナリデータを既存 Excel ファイルとしてオープンます。既存の Excel ファイルを元に別の Excel ファイルをオープンすることもできます。	P42
OpenBookEmbedEx	リソースに埋め込まれたバイナリデータを既存 Excel ファイルとしてオープンし、その識別 ID を取得します。既存の Excel ファイルを元に別の Excel ファイルをオープンすることもできます。(SelectBook メソッド専用)	P43

読み込みオープン

名前	内容	マニュアルページ
ReadBook	既存 Excel ファイルを読み込み専用でオープンします。	P44
ReadBookEx	既存 Excel ファイルを読み込み専用でオープンし、その識別 ID を取得します。(SelectBook メソッド専用)	P45

Excel ファイルのクローズ / PDF ファイル出力

名前	内容	マニュアルページ
CloseBook	Excel ファイルのクローズ、または破棄を行います。PDF ファイル出力も行うことができます。	P47
CloseBookEx	CreateBookEx / ReadBookEx / OpenBookEx / OpenBookEmbedEx で Excel ファイルを操作した場合に、Excel ファイルのクローズ、または破棄を行います。PDF ファイル出力も行うことができます。(SelectBook メソッド専用)	P48

Excel ファイル選択

名前	内容	マニュアルページ
SelectBook	CreateBookEx / ReadBookEx / OpenBookEx / OpenBookEmbedEx で Excel ファイルを操作する場合に、操作対象となる Excel ファイルの選択を行います。	P50

エラー取得

名前	内容	マニュアルページ
ErrorNo	エラー番号を取得します。	P52
ErrorMessage	エラーメッセージを取得します。	P54

Excel ファイル全体

名前	内容	マニュアルページ
DefFontName	Excel ファイル全シートのデフォルトフォント名を設定します。	P55
DefFontPoint	Excel ファイル全シートのデフォルトフォントサイズを設定します。	P56
ModeGrid	グリッド線の表示 / 非表示の設定します。	P57
Zoom	シートの表示倍率を設定します。	P58
KeyWord	変数名の先頭キーワード文字列を設定します。	P59
VarCheck	オーバーレイ元ファイルにある変数名をチェックします。	P60
VarInsertMode	オーバーレイ元ファイルのセル変数名への差し込みモードを設定します。	P61

シート

名前	内容	マニュアルページ
AddSheet	Excel ファイルに新規シートを追加します。	P63
DelSheet	シートを削除します。	P64
CopySheet	シートを設定した場所にコピーします。	P65
ActiveSheet	アクティブ シートをシート番号を取得、または、設定します。	P66
SheetCount	Excel ファイルのシート総数を取得します。	P67
RefSheet	参照するシート名を設定します。	P68
SheetNo	操作対象となるシートの選択を行います。	P69
SheetNo2	シート名からシート番号を取得します。	P70
SheetName	シート名の取得、または、設定します。	P71
SheetName2	シート番号からシート名を取得します。	P72

行・列コピー

名前	内容	マニュアルページ
RowCopy	行をコピーします。	P73
RowPaste	RowCopy メソッドでコピーした行を貼り付けます。	P74
RowDelete	行を削除します。	P75
RowClear	行をクリア（初期化）します。	P76
RowInsert	行を挿入します。	P77
ColumnCopy	セル列をコピーします。	P78
ColumnPaste	ColumnCopy メソッドでコピーした列を貼り付けます。	P79
ColumnDelete	セル列を削除します。	P80
ColumnClear	セル列をクリア（初期化）します。	P81
ColumnInsert	列を挿入します。	P82

印刷設定

名前	内容	マニュアルページ
Size	ページ属性の設定を行います。	P83
SizeFree	ユーザー定義の用紙サイズの設定を行います。	P84
Margin	シート単位にページ余白の設定を行います。	P86
Center	ページ中央を設定します。	P87
PageOrder	ページの印刷方向を設定します。	P88
PrintArea	シートの印刷範囲を設定します。	P89
Header	シートのヘッダーを設定します。	P90
Footer	シートのフッタを設定します。	P92

範囲取得

名前	内容	マニュアルページ
MaxData	データ（文字列 / 値 / 計算式）が設定されているセルの最終座標を取得します。	P94
MaxArea	データ（文字列 / 値 / 計算式）の他、罫線やセルの塗りつぶし等のセル書式が設定されている、最終セルの座標を取得します。	P95

イベント

名前	内容	マニュアルページ
Error	ExcelCreator 5.0 for .NET の実行時に不正な処理を行った場合に発生します。	P97
Progress	PDF ファイル出力時の処理状況を通知します。	P99

セルの範囲設定

名前	内容	マニュアルページ
Cell クラス	出力対象となるセルの範囲をセル位置（A1 参照形式）/ 変数名 / セル名で設定します。	P101
Pos クラス	出力対象となるセルの範囲を座標形式で設定します。	P103

値の取得・設定

名前	内容	マニュアルページ
Value	数値や文字列、日付などの値を設定します。値の取得も可能です。	P104
Str	文字列の取得、または設定します。	P105
Long	整数の取得、または、設定します。	P106
Double	実数の取得、または、設定します。	P107
Func	関数の書き込みをします。結果を示す値も同時に設定できます。	P108
AttrNo	セルの属性番号を取得します。	P109
Value2	数値や文字列、日付などの値と属性番号を設定します。	P110
Str2	文字列と属性番号を設定します。	P111
Long2	整数と属性番号を設定します。	P112
Double2	実数と属性番号を設定します。	P113

セルコピー・貼り付け、初期化

名前	内容	マニュアルページ
Copy	セルをコピーします。	P114
Paste	Copy メソッドでコピーしたセルを貼り付けます。	P115
Clear	セルをクリア（初期化）します。	P116

改ページの設定、セル幅・高さの取得・設定

名前	内容	マニュアルページ
Break	改ページの設定 / 解除を設定します。	P117
ColWidth	セルの幅を設定します。	P118
RowHeight	セルの高さを設定します。	P119
ColWidth2	セルの幅を設定 / 取得します。	P120
RowHeight2	セルの高さを設定/ 取得します。	P121

セルの属性

名前	内容	マニュアルページ
Cell/Pos.Attr クラス	セルの属性を設定します。	P **エラー! ブックマークが定義されていません。**

表示形式

名前	内容	マニュアルページ
Format	表示書式を設定します。	P120

表示形式（配置）

名前	内容	マニュアルページ
PosHorz	文字の配置（横位置）を設定します。	P124
PosVert	文字の配置（縦位置）を設定します。	P125
PosTurn	文字方向（角度）を設定します。	P126
OverReturn	「折り返して全体を表示」を設定します。	P127
Fit	「縮小して全体を表示する」を設定します。	P128
Joint	指定範囲のセルに「セルを結合する」を設定します。	P129

表示形式（フォント）

名前	内容	マニュアルページ
FontName	フォント名を設定します。	P130
FontPoint	フォントサイズを設定します。	P131
FontColor	フォント色を設定します。	P132
FontStyle	フォントスタイルを設定します。	P133
FontULine	フォントの下線スタイルを設定します。	P134

表示形式（罫線）

名前	内容	マニュアルページ
LineTop	上罫線を設定します。	P135
LineBottom	下罫線を設定します。	P135
LineLeft	左罫線を設定します。	P135
LineRight	右罫線を設定します。	P135
LineLeftUp	左上罫線を設定します。	P135
LineRightUp	右上罫線を設定します。	P135
Box	設定したセル領域を囲む Box 罫線を設定します。	P135

表示形式（パターン）

名前	内容	マニュアルページ
Pattern	パターンを設定します。	P139
BackColor	背景色を設定します。	P140

PDF

名前	内容	マニュアルページ
PDF クラス	出力する PDF ファイルの各種設定を行います。	P141
Title	文書プロパティの概要［タイトル］を設定します。	P142
Subject	文書プロパティの概要［サブタイトル］を設定します。	P143
Producer	文書プロパティの概要［PDF 変換］を設定します。	P144
Creator	文書プロパティの概要［アプリケーション］を設定します。	P145
Author	文書プロパティの概要［作成者］を設定します。	P146
EmbedFonts	埋め込むフォントを設定します。	P147
ImageType	イメージの種類を設定します。	P148
ProgressDialog	処理経過を表示するプログレスダイアログの表示 / 非表示を設定します。	P149

CreateBook メソッド

Excel ファイルの新規作成を行います。

書式

[VB.NET]
Public Function CreateBook(ByVal strFileName As String, ByVal sheetmax As Integer, ByVal version As ExcelCreator.xlVersion) As Integer

[C#]
public System.Int32 CreateBook(System.String strFileName , System.Int32 sheetmax , ExcelCreator.xlVersion version)

設定値

strFileName
　作成するファイル名を設定します。
sheetmax
　作成するシート数を設定します。
　省略する場合、-1 を設定してください。省略時には、デフォルト値 3 で作成されます。
version
　Excel ファイルのバージョンを xlVersion 列挙体のメンバで設定します。

定数	値	内容
ver97	97	Excel 97 の形式
ver2000	2000	Excel 2000 の形式
ver2002	2002	Excel 2002 の形式
ver2003	2003	Excel 2003 の形式

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。
引数 strFileName に既存の Excel ファイル名を設定した場合、既存ファイルを削除した後にファイルを新規作成します。

参照

CreatBookEx メソッド、CloseBook メソッド

コーディング例

3 シートある Excel2003 形式の Excel ファイルを新規作成します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("D1").Str = "abcde"
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("D1").Str = "abcde";
xlsCreator1.CloseBook(true);
```

CreateBookEx メソッド

Excel ファイルを新規作成し、その識別 ID を取得します。

書式

[VB.NET]
Public Function CreateBookEx(string strFileName, int sheetmax, xlVersion version, ref System.UInt32 pID) As Integer

[C#]
public System.Int32 CreateBookEx(System.String strFileName , System.Int32 sheetmax , ExcelCreator.xlVersion version , System.UInt32 pID)

設定値

strFileName
　作成するファイル名を設定します。
sheetmax
　作成するシート数を設定します。
　省略する場合、-1 を設定してください。省略時には、デフォルト値 3 で作成されます。
pID
　作成する Excel ファイルの識別 ID を取得するの変数を設定します。
version
　Excel ファイルのバージョンを xlVersion 列挙体のメンバで設定します。

定数	値	内容
ver97	97	Excel 97 の形式
ver2000	2000	Excel 2000 の形式
ver2002	2002	Excel 2002 の形式
ver2003	2003	Excel 2003 の形式

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。
一つの ExcelCreator 5.0 for .NET コンポーネントで複数の Excel ファイルを作成する場合、ファイルの識別 ID を使用して操作の対象となるファイルを SelectBook メソッドで切り替えて入出力処理を行います。
CreateBookEx でファイルを作成した場合は、必ず CloseBookEx でクローズしてください。
引数 strFileName に既存のファイル名を設定した場合、既存ファイルを削除した後にファイルを新規作成します。

参照

CloseBookEx メソッド、ReadBookEx メソッド、OpenBookEx メソッド、OpenBookEmbedEx メソッド、SelectBook メソッド

コーディング例

次のプログラムではファイルの新規作成を行います。

```
[VB.NET]
Dim pID1 As UInt32
Dim pID2 As UInt32

' Excel ファイル新規作成時に、ID を設定します
XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile01.xls", 1, ExcelCreator.xlVersion.ver2003, pID1)
XlsCreator1.Cell("A5").Long = 50000

XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile02.xls", 1, ExcelCreator.xlVersion.ver2003, pID2)
XlsCreator1.Cell("C5").Long = 75000

' ファイルクローズ時に、オープンした ID を選択し閉じます
```

```
XlsCreator1.CloseBookEx(True, pID2)
XlsCreator1.CloseBookEx(True, pID1)


[C#]
uint pID1 = 0;
uint pID2 = 0;

// Excel ファイル新規作成時に、ID を設定します
xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile01.xls", 1, ExcelCreator.xlVersion.ver2003,
ref pID1);
xlsCreator1.Cell("A5").Long = 50000;

xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile02.xls", 1, ExcelCreator.xlVersion.ver2003,
ref pID2);
xlsCreator1.Cell("C5").Long = 75000;

// ファイルクローズ時に、オープンした ID を選択し閉じます
xlsCreator1.CloseBookEx(true, pID2);
xlsCreator1.CloseBookEx(true, pID1);
```

OpenBook メソッド

既存 Excel ファイルを読み書きオープンします。

書式

[VB.NET]
```
Public Function OpenBook(ByVal strFileName As String, ByVal strOverlay As String) As Integer
```

[C#]
```
public System.Int32 OpenBook(System.String strFileName , System.String strOverlay)
```

設定値

strFileName
　オープンする Excel ファイル名を設定します。
strOverlay
　オーバーレイ元ファイル名を設定します。省略時には "" を設定します。

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。
引数 strFileName のファイルは、引数 strOverlay のファイル全体の情報をコピーした新たなファイルとしてオープンします。
引数 strOverlay を省略した場合、引数 strFileName のファイルをオープンします。
引数 strFileName のファイルは新規で作成されるため、既存のファイル名を設定した場合は既存ファイルは上書きされます。
オーバーレイ元ファイルにあらかじめ、変数文字やセルにセル名を設定している場合は、変数文字やセル名を利用したセル指定で、文字列や値、属性の設定を行うこともできます。

参照

CloseBook メソッド、OpenBookEx メソッド、OpenBookEmbed メソッド、OpenBookEmbedEx メソッド

コーディング例

既存ファイルを開き、値を設定します。

[VB.NET]
```
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.Cell("D1").Long = 100     'D1 に 100 を設定します
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
xlsCreator1.Cell("D1").Long = 100;     //D1 に 100 を設定します
xlsCreator1.CloseBook(true);
```

オーバーレイ用としてファイルを開き、新たなファイルを作成します
オーバーレイ用として ExcelTemplate.xls を開き、オーバーレイ用ファイル内の D1 セルに、100 と代入された新たな ExcelFile.xls を作成します

[VB.NET]
```
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", _
                     Application.StartupPath & "¥ExcelTemplate.xls")
XlsCreator1.Cell("D1").Long = 100
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls",
                     Application.StartupPath + @"¥ExcelTemplate.xls");
```

```
xlsCreator1.Cell("D1").Long = 100;
xlsCreator1.CloseBook(true);
```

注意事項

・保護された Excel ファイルをオープンすることはできません。

OpenBookEx メソッド

既存 Excel ファイルを読み書きオープンし、その識別 ID を取得します。

書式

[VB.NET]
```
Public Function OpenBookEx(ByVal strFileName As String, ByVal strOverlay As String, ByRef pID As System.UInt32) As Integer
```

[C#]
```
public System.Int32 OpenBookEx(System.String strFileName , System.String strOverlay , System.UInt32 pID)
```

設定値

strFileName
　オープンする Excel ファイル名を設定します。
strOverlay
　オーバーレイ元ファイル名を設定します。省略時には "" を設定します。
pID
　オープンするファイルの識別 ID を取得する変数を設定します。

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。
一つの ExcelCreator 5.0 for .NET コンポーネントで複数の Excel ファイルを作成する場合、ファイルの識別 ID を使用して操作の対象となる Excel ファイルを SelectBook メソッドで切り替えて入出力処理を行います。
OpenBookEx で Excel ファイルを作成した場合は、必ず CloseBookEx でクローズしてください。

引数 strFileName のファイルは、引数 strOverlay のファイルをコピーした新たなファイルとしてオープンします。
引数 strOverlay を省略した場合、引数 strFileName のファイルをオープンします。
引数 strFileName のファイルは新規で作成されるため、既存のファイル名を設定した場合、既存ファイルは上書きされます。
オーバーレイ元ファイルにあらかじめ、変数文字やセルにセル名を設定している場合は、変数文字やセル名を利用したセル指定で、文字列や値、属性の設定を行うこともできます。

参照

CloseBookEx メソッド、OpenBookEmbed メソッド、OpenBookEmbedEx メソッド、SelectBook メソッド

コーディング例

オーバーレイ用としてファイルを開き、新たなファイルを作成します。
オーバーレイ用として ExcelTemplate.xls を開き、セルに値を設定した新たな ExcelFile01.xls / ExcelFile02.xls を作成します。

[VB.NET]
```
Dim pID1 As UInt32
Dim pID2 As UInt32

' Excel ファイルのオーバーレイオープン時に、ID を設定します
XlsCreator1.OpenBookEx(Application.StartupPath & "¥ExcelFile01.xls", _
                       Application.StartupPath & "¥ExcelTemplate.xls", pID1)
XlsCreator1.Cell("A5").Long = 50000

XlsCreator1.OpenBookEx(Application.StartupPath & "¥ExcelFile02.xls", _
                       Application.StartupPath & "¥ExcelTemplate.xls", pID2)
XlsCreator1.Cell("C5").Long = 75000

' ファイルクローズ時に、オープンした ID を選択し閉じます
XlsCreator1.CloseBookEx(True, pID2)
XlsCreator1.CloseBookEx(True, pID1)
```

```
[C#]
uint pID1 = 0;
uint pID2 = 0;

// Excel ファイルのオーバーレイオープン時に、ID を設定します
xlsCreator1.OpenBookEx(Application.StartupPath + @"¥ExcelFile01.xls",
                       Application.StartupPath + @"¥ExcelTemplate.xls", ref pID1);
xlsCreator1.Cell("A5").Long = 50000;

xlsCreator1.OpenBookEx(Application.StartupPath + @"¥ExcelFile02.xls",
                       Application.StartupPath + @"¥ExcelTemplate.xls", ref pID2);
xlsCreator1.Cell("C5").Long = 75000;

// ファイルクローズ時に、オープンした ID を選択し閉じます
xlsCreator1.CloseBookEx(true, pID2);
xlsCreator1.CloseBookEx(true, pID1);
```

注意事項

・保護された Excel ファイルをオープンすることはできません。

OpenBookEmbed メソッド

リソースに埋め込まれたバイナリデータを既存 Excel ファイルとしてオープンします。

書式

[VB.NET]
Public Function OpenBookEmbed(ByVal strFileName As String, ByVal stream As System.IO.Stream) As Integer
Public Function OpenBookEmbed(ByVal strFileName As String, ByVal bytes() As Byte) As Integer

[C#]
public System.Int32 OpenBookEmbed(System.String strFileName , System.IO.Stream stream)
public System.Int32 OpenBookEmbed(System.String strFileName , byte[] bytes)

設定値

strFileName
　オープンするファイル名を設定します。
stream , bytes
　オーバーレイ元となるリソースに埋め込まれたバイナリデータを設定します。

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。
OpenBookEmbed でファイルを作成した場合は、必ず CloseBook でクローズしてください。

引数 strFileName のファイルは、引数 stream や bytes の情報をコピーした新たなファイルとしてオープンします。
オーバーレイ元ファイルにあらかじめ、変数文字やセルにセル名を設定している場合は、変数文字やセル名を利用したセル指定で、文字列や値、属性の設定を行うこともできます。

使用方法やコーディング例は、 埋め込みリソースのオーバーレイ / オーバーレイ元ファイル（Excel ファイル）を埋め込みリソースとしてプロジェクトに登録する / オーバーレイ元ファイル（Excel ファイル）を元に作成したリソースファイルを埋め込みリソースとしてプロジェクトに登録する を参照してください。

参照

CloseBook メソッド

注意事項

・保護された Excel ファイルをオープンすることはできません。

OpenBookEmbedEx メソッド

リソースに埋め込まれたバイナリデータを既存 Excel ファイルとして扱い、読み書きオープンし、その識別 ID を取得します。

書式

[VB.NET]
Public Function OpenBookEmbedEx(ByVal strFileName As String, ByVal stream As System.IO.Stream, ByRef pID As System.UInt32) As Integer
Public Function OpenBookEmbedEx(ByVal strFileName As String, ByVal bytes() As Byte, ByRef pID As System.UInt32) As Integer

[C#]
public System.Int32 OpenBookEmbedEx(System.String strFileName , System.IO.Stream stream , System.UInt32 pID)
public System.Int32 OpenBookEmbedEx(System.String strFileName , byte[] bytes , System.UInt32 pID)

設定値

stream , bytes
　オーバーレイ元となるリソースに埋め込まれたバイナリデータを設定します。
pID
　作成するファイルの識別 ID を取得する変数を設定します。

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。
一つの ExcelCreator 5.0 for .NET コンポーネントで複数の Excel ファイルを作成する場合、ファイルの識別 ID を使用して操作の対象となるファイルを SelectBook メソッドで切りて入出力処理を行います。
OpenBookEmbedEx でファイルを作成した場合は、必ず CloseBookEx でクローズしてください。

引数 strFileName のファイルは、引数 stream や bytes の情報をコピーした新たなファイルとしてオープンします。
オーバーレイ元ファイルにあらかじめ、変数文字やセルにセル名を設定している場合は、変数文字やセル名を利用したセル指定で、文字列や値、属性の設定を行うこともできます。

使用方法やコーディング例は、 埋め込みリソースのオーバーレイ / オーバーレイ元ファイル（Excel ファイル）を埋め込みリソースとしてプロジェクトに登録する / オーバーレイ元ファイル（Excel ファイル）を元に作成したリソースファイルを埋め込みリソースとしてプロジェクトに登録する を参照してください。

参照

CloseBookEx メソッド、SelectBook メソッド

注意事項

・保護された Excel ファイルをオープンすることはできません。

ReadBook メソッド

既存 Excel ファイルを読み込み専用でオープンします。

書式

[VB.NET]
Public Function ReadBook(ByVal strFileName As String) As Integer

[C#]
public System.Int32 ReadBook(System.String strFileName)

設定値

strFileName
　読み込み専用でオープンするファイル名を設定します。

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。

参照

CloseBook メソッド、ReadBookExメソッド

コーディング例

読み込み用としてファイルを開き、セルの値を読み込みます。

[VB.NET]
```
Dim nLong As Long
XlsCreator1.ReadBook(Application.StartupPath & "¥ExcelFile.xls")
nLong = XlsCreator1.Cell("D1").Long
XlsCreator1.CloseBook(True)
```

[C#]
```
long  nValue;
xlsCreator1.ReadBook(Application.StartupPath + @"¥ExcelFile.xls");
nValue = xlsCreator1.Cell("D1").Long;
xlsCreator1.CloseBook(true);
```

注意事項

・保護された Excel ファイルをオープンすることはできません。
・ReadBook メソッドは読み込み専用でファイルをオープンするため、使用できるメソッドやプロパティに制限があります。

ReadBookEx メソッド

既存 Excel ファイルを読み込み専用オープンし、その識別 ID を取得します。

書式

[VB.NET]
Public Function ReadBookEx(ByVal strFileName As String, ByRef pID As System.UInt32) As Integer

[C#]
public System.Int32 ReadBookEx(System.String strFileName , System.UInt32 pID)

設定値

strFileName
　読み込み専用でオープンする Excel ファイル名を設定します。
pID
　作成するファイルの識別 ID を取得する変数を設定します。

解説

戻り値はメソッドが正常終了した場合には 0 以上の値を、それ以外の場合には 0 より小さい値を返します。
一つの ExcelCreator 5.0 for .NET コンポーネントで複数の Excel ファイルを作成する場合、ファイルの識別 ID を使用して操作の対象となるファイルを SelectBook メソッドで切り替えて入出力処理を行います。
ReadBookEx でファイルを作成した場合は、必ず CloseBookEx でクローズしてください。

参照

CloseBookEx メソッド、SelectBook メソッド

コーディング例

読み込み用としてファイルを開きます。

```
[VB.NET]
Dim pID1 As UInt32
Dim pID2 As UInt32
Dim lValue As Long

' Excel ファイル読み込み専用オープン時に、ID を設定します
XlsCreator1.ReadBookEx(Application.StartupPath & "¥ExcelFile01.xls", pID1)
lValue = XlsCreator1.Cell("A1").Long

XlsCreator1.ReadBookEx(Application.StartupPath & "¥ExcelFile02.xls", pID2)
lValue = XlsCreator1.Cell("C1").Long

' ファイルクローズ時に、オープンした ID を選択し閉じます
XlsCreator1.CloseBookEx(True, pID2)
XlsCreator1.CloseBookEx(True, pID1)


[C#]
uint pID1 = 0;
uint pID2 = 0;
long lValue;

// Excel ファイル読み込み専用オープン時に、ID を設定します
xlsCreator1.ReadBookEx(Application.StartupPath + @"¥ExcelFile01.xls", ref pID1);
lValue = xlsCreator1.Cell("A1").Long;

xlsCreator1.ReadBookEx(Application.StartupPath + @"¥ExcelFile02.xls", ref pID2);
lValue = xlsCreator1.Cell("C1").Long;

// ファイルクローズ時に、オープンした ID を選択し閉じます
```

```
xlsCreator1.CloseBookEx(true, pID2);
xlsCreator1.CloseBookEx(true, pID1);
```

注意事項

・保護された Excel ファイルをオープンすることはできません。
・ReadBook メソッドは読み込み専用で Excel ファイルをオープンするため、使用できるメソッドやプロパティに制限があります。

CloseBook メソッド

Excel ファイルのクローズ、または破棄を行います。PDF ファイル出力も行うことができます。

書式

[VB.NET]
Public Sub CloseBook(ByVal mode As Boolean)
Public Sub CloseBook(ByVal mode As Boolean, ByVal strPDFName As String, ByVal bDelete As Boolean)
Public Sub CloseBook(ByVal mode As Boolean, ByVal strm As System.IO.Stream, ByVal bDelete As Boolean)

[C#]
public void CloseBook(System.Boolean mode)
public void CloseBook(System.Boolean mode , System.String strPDFName , System.Boolean bDelete)
public void CloseBook(System.Boolean mode , System.IO.Stream strm , System.Boolean bDelete)

設定値

strPDFName , strm
出力する PDF ファイルを設定します。
mode

定数	内容
True	Excel ファイルをクローズします。
False	変更内容を破棄し、オープン時点の内容に戻します。

bDelete

定数	内容
True	PDF ファイル保存時に Excel ファイルを削除します。
False	PDF ファイル保存時に Excel ファイルは削除しません。

解説

CreateBook、OpenBook、ReadBook、OpenBookEmbed メソッドでオープンした後は、必ず、Excel ファイルのクローズ、または、破棄を行ってください。
PDF ファイルを作成した時に Excel ファイルが不要な場合は、引数 bDelete に True を設定します。

参照

CreateBook メソッド、ReadBook メソッド、OpenBook メソッド、OpenBookEmbed メソッド

コーディング例

開いているファイルを閉じます。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("D1").Str = "abcde"
XlsCreator1.CloseBook(True)  '開いているファイルを閉じます


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("D1").Str = "abcde";
xlsCreator1.CloseBook(true);  //開いているファイルを閉じます
```

CloseBookEx メソッド

CreateBookEx / ReadBookEx / OpenBookEx / OpenBookEmbedEx で Excel ファイルを操作した場合に、クローズ、または破棄を行います。PDF ファイル出力も行うことができます。

書式

[VB.NET]
Public Sub CloseBookEx(ByVal mode As Boolean, ByVal uiID As System.UInt32)
Public Sub CloseBookEx(ByVal mode As Boolean, ByVal uiID As System.UInt32, ByVal strPDFName As String, ByVal bDelete As Boolean)
Public Sub CloseBookEx(ByVal mode As Boolean, ByVal uiID As System.UInt32, ByVal strm As System.IO.Stream, ByVal bDelete As Boolean)

[C#]
public void CloseBookEx(System.Boolean mode , System.UInt32 uiID)
public void CloseBookEx(System.Boolean mode , System.UInt32 uiID , System.IO.Stream strm , System.Boolean bDelete)
public void CloseBookEx(System.Boolean mode , System.UInt32 uiID , System.String strPDFName , System.Boolean bDelete)

設定値

strm , strPDFName
　出力する PDF ファイルを設定します。
pID
　Excel ファイルの識別 ID を取得する変数を設定します。
mode

定数	内容
True	Excel ファイルをクローズします。
False	変更内容を破棄し、オープン時点の内容に戻します。

bDelete

定数	内容
True	PDF ファイル保存時に Excel ファイルを削除します。
False	PDF ファイル保存時に Excel ファイルは削除しません。

解説

CreateBookEx / ReadBookEx / OpenBookEx / OpenBookEmbedEx メソッドでオープンした後は、必ずファイルのクローズ、または、破棄を行ってください。
PDF ファイルを作成した時に Excel ファイルが不要な場合は、引数 bDelete に True を設定します。

参照

CreateBookEx メソッド、ReadBookEx メソッド、OpenBookEx メソッド、OpenBookEmbedEx メソッド

コーディング例

開いているファイルを閉じます。

```
[VB.NET]
Dim pID1 As UInt32
Dim pID2 As UInt32

XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile01.xls", _
                         1, ExcelCreator.xlVersion.ver2003, pID1)
XlsCreator1.Cell("A1").Value = 10000

XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile02.xls", _
                         1, ExcelCreator.xlVersion.ver2003, pID2)
```

```
XlsCreator1.Cell("A5").Value = 18000

' ファイルクローズ時に、オープンした ID を選択します
XlsCreator1.CloseBookEx(True, pID2)
XlsCreator1.CloseBookEx(True, pID1)


[C#]
uint pID1 = 0;
uint pID2 = 0;

xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile01.xls",
                         1, ExcelCreator.xlVersion.ver2003, ref pID1);
xlsCreator1.Cell("A1").Value = 10000;

xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile02.xls",
                         1, ExcelCreator.xlVersion.ver2003, ref pID2);
xlsCreator1.Cell("A5").Value = 18000;

// ファイルクローズ時に、オープンした ID を選択し閉じます
xlsCreator1.CloseBookEx(true, pID2);
xlsCreator1.CloseBookEx(true, pID1);
```

複数の開いているファイルを閉じます。

```
 [VB.NET]
Dim pID1 As UInt32
Dim pID2 As UInt32

XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile01.xls", _
                         1, ExcelCreator.xlVersion.ver2003, pID1)
XlsCreator1.PDF.Title = "2005 年 1 月売り上げ一覧"

XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile02.xls", _
                         1, ExcelCreator.xlVersion.ver2003, pID2)
XlsCreator1.PDF.Title = "2005 年 2 月売り上げ一覧"

' ファイルクローズ時に、オープンした ID を選択し PDF ファイルへ出力します
XlsCreator1.CloseBookEx(True, pID2, Application.StartupPath & "¥PdfFile02.pdf", True)
XlsCreator1.CloseBookEx(True, pID1, Application.StartupPath & "¥PdfFile01.pdf", True)


[C#]
uint pID1 = 0;
uint pID2 = 0;

xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile01.xls",
                         1, ExcelCreator.xlVersion.ver2003, ref pID1);
xlsCreator1.PDF.Title = "2005 年 1 月売り上げ一覧";

xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile02.xls",
                         1, ExcelCreator.xlVersion.ver2003, ref pID2);
xlsCreator1.PDF.Title = "2005 年 2 月売り上げ一覧";

// ファイルクローズ時に、オープンした ID を選択し PDF ファイルへ出力します
xlsCreator1.CloseBookEx(true, pID2, Application.StartupPath + @"¥PdfFile02.pdf", true);
xlsCreator1.CloseBookEx(true, pID1, Application.StartupPath + @"¥PdfFile01.pdf", true);
```

SelectBook メソッド

CreateBookEx / ReadBookEx / OpenBookEx / OpenBookEmbedEx で Excel ファイルを操作する場合に、操作対象となる Excel ファイルの選択を行います。

書式

[VB.NET]
Public Sub SelectBook(ByVal uiID As System.UInt32)
Public Sub CloseBookEx(ByVal mode As Boolean, ByVal uiID As System.UInt32, ByVal strPDFName As String, ByVal bDelete As Boolean)
Public Sub CloseBookEx(ByVal mode As Boolean, ByVal uiID As System.UInt32, ByVal strm As System.IO.Stream, ByVal bDelete As Boolean)

[C#]
public void SelectBook(System.UInt32 uiID)

設定値

uiID
操作対象とする Excel ファイルの識別 ID を設定します。

参照

CreateBookEx メソッド、ReadBookEx メソッド、OpenBookEx メソッド、OpenBookEmbedEx メソッド、CloseBookEx メソッド

コーディング例

選択対象の Excel ファイルを変更しながら値設定を行います。

[VB.NET]
```
Dim pID1 As UInt32
Dim pID2 As UInt32

' Excel ファイル新規作成時に、ID を設定します
XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile01.xls", _
                         1, ExcelCreator.xlVersion.ver2003, pID1)
XlsCreator1.CreateBookEx(Application.StartupPath & "¥ExcelFile02.xls", _
                         1, ExcelCreator.xlVersion.ver2003, pID2)

' 操作対象の ID を選択します
XlsCreator1.SelectBook(pID1)
XlsCreator1.Cell("A1").Value = 10000
XlsCreator1.SelectBook(pID2)
XlsCreator1.Cell("A5").Value = 18000

XlsCreator1.CloseBookEx(True, pID2)
XlsCreator1.CloseBookEx(True, pID1)
```

[C#]
```
uint pID1 = 0;
uint pID2 = 0;

// Excel ファイル新規作成時に、ID を設定します
xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile01.xls",
                         1, ExcelCreator.xlVersion.ver2003, ref pID1);
xlsCreator1.CreateBookEx(Application.StartupPath + @"¥ExcelFile02.xls",
                         1, ExcelCreator.xlVersion.ver2003, ref pID2);

// 操作対象の ID を選択します
xlsCreator1.SelectBook(pID1);
```

```
        xlsCreator1.Cell("A1").Value = 10000;
        xlsCreator1.SelectBook(pID2);
        xlsCreator1.Cell("A5").Value = 18000;

        xlsCreator1.CloseBookEx(true, pID2);
        xlsCreator1.CloseBookEx(true, pID1);
```

ErrorNo プロパティ

エラー番号を取得します。実行時に値の取得のみ可能です。

書式

[VB.NET]
Public ReadOnly Property ErrorNo As ExcelCreator.xlErrorNo

[C#]
Public ExcelCreator.xlErrorNo ErrorNo [get]

戻り値

xlErrorNo
実行時のエラー番号が渡されます。
エラー番号については、次の値を参照してください。

定数	値	内容
errDontOpen	1	ファイルがオープンできない
errFileHandle	2	無効なファイルハンドル
errNotOpen	3	未オープンエラー
errSheetNo	4	シート番号エラー
errWrite	5	書き込み時のエラー
errAction	6	動作モードエラー
errDataCat	7	データ種別エラー
errValue	8	エラー値
errClear	20	クリアエラー
errCopy	21	コピーエラー
errPaste	22	貼り付けエラー
errInsert	23	挿入エラー
errDelete	24	削除エラー
errLength	31	長さエラー
errLocate	32	座標エラー
errAttr	33	属性番号エラー
errParam	34	パラメータエラー
errNoData	35	データが無い
errEndOfData	36	データの終わり
errVarPoint	37	指定した変数存在しない
errBreakCount	38	改ページ数が制限超えた
errMemory1	40	メモリー不足エラー1
errMemory2	41	メモリー不足エラー2
errOther	50	内部エラー
errFunction	100	計算式形式エラー
errCreate	300	ファイル作成エラー
errTempCreate	301	Temp ファイル作成エラー
errTempOpen	302	Temp ファイルオープンエラー
errStream	303	ストリームエラー
errProtect	304	保護されたファイルです
errDoubleOpen	305	二重オープンです
rMutex	400	ミューテックスエラー
errMaxFileName	401	ファイル名が長すぎます
errInvalidFileName	402	無効なファイル名
errEmbedFile	403	埋込みバイナリエラーです

解説

エラーが発生した場合にエラー番号を取得します。
正常時の戻り値は 0 を返します。

参照

ErrorMessage プロパティ、Error イベント

コーディング例

ErrorNo プロパティを使用してエラー処理を行います。
ファイル名が正しくないため、エラー時にメッセージを表示して処理を抜けます。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xl", _
                       3, ExcelCreator.xlVersion.ver2003)
If XlsCreator1.ErrorNo <> 0 Then
    MsgBox("エラーです")
    Exit Sub
End If
XlsCreator1.Cell("A2").Str = "abcd"
XlsCreator1.CloseBook(True)

[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xl",
                       3, ExcelCreator.xlVersion.ver2003);
if (xlsCreator1.ErrorNo != 0)
{
    MessageBox.Show("エラーです");
    return;
}
xlsCreator1.Cell("A2").Str = "abcd";
xlsCreator1.CloseBook(true);
```

ErrorMessage プロパティ

エラーメッセージを取得します。実行時に値の取得のみ可能です。

書式

[VB.NET]
Public ReadOnly Property ErrorMessage As String

[C#]
Public string ErrorMessage [get]

戻り値

エラーメッセージが文字列で渡されます。
エラーメッセージについては、ErrorNo プロパティを参照してください。

解説

エラーが発生した場合、エラーメッセージを文字列型で取得します。

参照

ErrorNo プロパティ、Error イベント

コーディング例

同一ファイルを開いているため、「ファイルがオープンできない」というメッセージが表示されます。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
If XlsCreator1.ErrorNo <> ExcelCreator.xlErrorNo.errNoError Then
    MsgBox(XlsCreator1.ErrorMessage)
    Exit Sub
End If
XlsCreator1.CloseBook(True)

[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
if (xlsCreator1.ErrorNo != ExcelCreator.xlErrorNo.errNoError)
{
    MessageBox.Show(xlsCreator1.ErrorMessage);
    return;
}

xlsCreator1.CloseBook(true);
```

DefFontName プロパティ

Excel ファイルの全シートのデフォルトフォント名を設定します。

書式

[VB.NET]
Public WriteOnly Property DefFontName As String

[C#]
Public string DefFontName [set]

設定値

フォント名を設定します。

解説

Excel ファイルの全シートで、フォントの書式設定を行っていないセルが設定対象になります。

参照

DefFontPoint プロパティ

コーディング例

Excel ファイルの全シートのフォント名、フォントサイズを設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.DefFontName = "ＭＳ ゴシック"
XlsCreator1.DefFontPoint = 9
XlsCreator1.Cell("A1").Str = "ABCDE"
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.DefFontName = "ＭＳ ゴシック";
xlsCreator1.DefFontPoint = 9;
xlsCreator1.Cell("A1").Str = "ABCDE";
xlsCreator1.CloseBook(true);
```

DefFontPoint プロパティ

Excel ファイルの全シートのデフォルトフォントサイズを設定します。

書式

[VB.NET]
Public WriteOnly Property DefFontPoint As Double

[C#]
Public double DefFontPoint [set]

設定値

フォントのサイズ（1 ～ 409 ポイント）を設定します。

解説

Excel ファイルの全シートで、フォントの書式設定を行っていないセルが設定対象になります。

参照

DefFontName プロパティ

コーディング例

Excel ファイルの全シートのフォント名、フォントサイズを設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.DefFontName = "ＭＳ ゴシック"
XlsCreator1.Cell("A1").Str = "ABCDE"
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.DefFontName = "ＭＳ ゴシック";
xlsCreator1.DefFontPoint = 9;
xlsCreator1.Cell("A1").Str = "ABCDE";
xlsCreator1.CloseBook(true);
```

ModeGrid プロパティ

グリッド線の表示 / 非表示を設定します。

書式

[VB.NET]
```
Public WriteOnly Property ModeGrid As Boolean
```

[C#]
```
Public bool ModeGrid [set]
```

設定値

定数	内容
True	グリッド線を表示する。 （デフォルト値）
False	グリッド線を非表示にする。

コーディング例

シート内のグリッド線を非表示にします。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.ModeGrid = False
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.ModeGrid = false;
xlsCreator1.CloseBook(true);
```

Zoom プロパティ

シートの表示倍率を設定します。

書式

[VB.NET]
Public WriteOnly Property Zoom As Integer

[C#]
Public System.Int32 Zoom [set]

設定値

シートに対する表示倍率（10 ～ 400）を設定します。

コーディング例

シートの表示倍率を設定します。

```
[VB.NET]
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.SheetNo = 0
XlsCreator1.Zoom = 90
XlsCreator1.CloseBook(True)

[C#]
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
xlsCreator1.SheetNo = 0;
xlsCreator1.Zoom = 90;
xlsCreator1.CloseBook(true);
```

KeyWord プロパティ

変数名の先頭キーワード文字列を設定します。

書式

[VB.NET]
Public WriteOnly Property KeyWord As String

[C#]
Public string KeyWord [set]

設定値

変数文字列の先頭キーワード文字列を設定します。(最大 64 文字)

解説

変数名の先頭キーワード文字列を設定します。
デフォルトの変数名の先頭キーワード文字列 ** です。デフォルト以外を使用したい場合に KeyWord プロパティで先頭キーワード文字列を設定できます。
先頭キーワード文字列は Excel で文字として認識可能で、かつ、半角文字のみ設定することができます。但し、¥ ' " , ^ . [ ] の半角文字は使用できません。
なお、先頭キーワード文字列は ExcelCreator 5.0 for .NET コンポーネントの 1 つのオブジェクトで共通となりますので、処理の途中において先頭キーワード文字列を変更した場合は、その後にオープンする Excel ファイルに影響を受けます。

コーディング例

変数名の先頭キーワード文字列を -- に設定します。

[VB.NET]
```
XlsCreator1.KeyWord = "--"
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Str = "--NAME"
XlsCreator1.CloseBook(True)

XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.Cell("--NAME").Long = 2000
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.KeyWord = "--";
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Str = "--NAME";
xlsCreator1.CloseBook(true);

xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
xlsCreator1.Cell("--NAME").Long = 2000;
xlsCreator1.CloseBook(true);
```

VarCheck プロパティ

オーバーレイ元ファイルにある変数名をチェックします。

書式

[VB.NET]
Public Function VarCheck(ByVal strVarName As String) As Boolean

[C#]
Public System.Boolean VarCheck(System.String strVarName)

設定値

strVarName
チェックする変数名を設定します。

解説

存在する場合は True、存在しない場合は False が返ります。

存在しない変数名に書式や値を設定するとエラーが発生するため、あらかじめ、変数名が存在するか確認を行うことができます。

参照

KeyWord プロパティ

コーディング例

変数名をチェックして、データを差し込みます。

```
[VB.NET]
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", _
                     Application.StartupPath & "¥ExcelTemplate.xls")

' セル変数がある時にデータを設定します
If XlsCreator1.VarCheck("**POST") = True Then
    XlsCreator1.Cell("**POST").Value = "918-8237"
End If
If XlsCreator1.VarCheck("**ADDRESS") = True Then
    XlsCreator1.Cell("**ADDRESS").Value = "福井県福井市和田東 1 丁目 222 番地"
End If

XlsCreator1.CloseBook(True)

[C#]
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", _
                     Application.StartupPath + @"¥ExcelTemplate.xls");

// セル変数がある時にデータを設定します
if (xlsCreator1.VarCheck("**POST"))
    xlsCreator1.Cell("**POST").Value = "918-8237";
if (xlsCreator1.VarCheck("**ADDRESS"))
    xlsCreator1.Cell("**ADDRESS").Value = "福井県福井市和田東 1 丁目 222 番地";

xlsCreator1.CloseBook(true);
```

VarInsertMode プロパティ

オーバーレイ元ファイルのセル変数名への差し込みモードを設定します。

書式

[VB.NET]
Public WriteOnly Property VarInsertMode() As ExcelCreator.xlVarInsertMode

[C#]
public ExcelCreator.xlVarInsertMode VarInsertMode [set]

設定値

xlVarInsertMode 列挙体のメンバで設定します。

定数	値	内容
vmValidAll	0	全ての変数名を対象（デフォルト値）
vmInvalidAnswer	1	計算結果の変数名は対象外
vmInvalidTextBox	2	TextBox の変数名は対象外
vmInvalidCellText	4	セルに直接設定した変数名は対象外

解説

計算式の結果として参照するセルに変数文字列が設定されている場合、計算式が設定されたセルの値も変数文字列となりますが、その変数文字列に差し込みを行うと、計算式が差し込んだデータに置き換わってしまいます。
例えば、A1 セルに "=A2" が設定されていて、A2 セルの内容に "**Name" などの変数文字列が設定されている場合、A1 セル、A2 セルの内容はともに "**Name" となりますが、プログラム中で "**Name" にデータ（例 : 100）を差し込むと、A1 セル、A2 セルの値はともに 100 になり、A1 セルに設定されていた計算式は上書きされます。
このようなケースで計算式の内容が置き換わらないようにするには、このプロパティで vmInvalidAnswer を設定することで、計算結果としての変数文字列には差し込みを行わないようにすることができます。

オーバーレイ元ファイルのオープン前に設定してください。オープン後に変更した場合は、次のオープン以降で有効になります。

・「全ての変数名を対象」の場合
オーバーレイ元のシートに定義している全ての変数名（関数結果とテキストボックスの変数名も含む）に値を差し込みます。

・「関数結果の変数名は対象外」の場合
例えば A1 セルに **Name 、A2 セルに =A1 と設定すると A2 セルは関数結果として変数名が表示されます。「計算結果の変数名は対象外」の場合、前述の A2 セルに値は差し込まれません。

・「テキストボックスの変数名は対象外」の場合
オーバーレイ元ファイルに定義しているテキストボックスの変数名に値は差し込まれません。

・「セルに直接設定した変数名は対象外」の場合
オーバーレイ元ファイルのセルに直接変数名文字列を定義しているセルに値は差し込まれません。関数結果の変数名やテキストボックスの変数名には値が差し込まれます。

参照

KeyWord プロパティ

コーディング例

セル変数名への差し込みモードを「セルに直接設定した変数名は対象外」に設定します。
この場合、**ZIPCODE にはデータは差し込まれません。

```
[VB.NET]
XlsCreator1.VarInsertMode = ExcelCreator.xlVarInsertMode.vmInvalidCellText

XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", _
                     Application.StartupPath & "¥ExcelTemplate.xls")

XlsCreator1.Cell("**TEXTBOX").Value = "918-8237"

XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.VarInsertMode = ExcelCreator.xlVarInsertMode.vmInvalidCellText;

xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls",
                     Application.StartupPath + @"¥ExcelTemplate.xls");

xlsCreator1.Cell("**TEXTBOX").Value = "918-8237";

xlsCreator1.CloseBook(true);
```

注意事項

・VarInsertMode プロパティの設定は、ブック作成／オープンの前に行います。

AddSheet メソッド

Excel ファイルに新規シートを追加します。

書式

[VB.NET]
```
Public Sub AddSheet(ByVal count As Integer)
```

[C#]
```
public void AddSheet(System.Int32 count)
```

設定値

count
追加するシート数（1 ～）を設定します。

解説

新規シートは、Excel ファイルの最後に追加します。
追加したシート名「Sheet<FONT COLOR=#FF0000>N</FONT>」（N は現在の最大シート数 + 1）が設定されます。

参照

DelSheet メソッド

コーディング例

Excel ファイルに 5 シート追加を行います。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.AddSheet(5)
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.AddSheet(5);
xlsCreator1.CloseBook(true);
```

DelSheet メソッド

シートを削除します。

書式

[VB.NET]
Public Sub DelSheet(ByVal sheetno As Integer, ByVal count As Integer) Public Sub DelSheet(ByVal sheetno As Integer, ByVal count As Integer)

[C#]
public void DelSheet(System.Int32 sheetno , System.Int32 count)

設定値

sheetno
　削除するシート番号（0 ～ ）を設定します。
count
　削除するシート数（1 ～ ）を設定します。

解説

引数 count に全シート数を超える値を設定すると、存在する全てのシートを削除します。

参照

AddSheet メソッド

コーディング例

Excel ファイル内の 4 シート目から 2 シート分の削除を行います。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.DelSheet(3, 2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.DelSheet(3, 2);
xlsCreator1.CloseBook(true);
```

CopySheet メソッド

シートを設定した場所にコピーします。

書式

[VB.NET]
Public Sub CopySheet(ByVal sheetno As Integer, ByVal insertpos As Integer, ByVal sheetname As String)

[C#]
Public void CopySheet(System.Int32 sheetno , System.Int32 insertpos , System.String sheetname)

設定値

sheetno
　コピー元のシート番号値（0 ～）を設定します。
insertpos
　コピー元のシートを挿入するシート番号値（0 ～）を設定します。
sheetname
　シート名を示す文字列を設定します。

コーディング例

シートのコピーを行います。
0 番目のシートを 1 番目の位置にコピーし、シート名を「コピーしたシート」に設定します。。
3 番目のシートを 5 番目の位置にコピーします。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 10, ExcelCreator.xlVersion.ver2003)
XlsCreator1.CopySheet(0, 1, "コピーしたシート")
XlsCreator1.CopySheet(3, 5, "")
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 10, ExcelCreator.xlVersion.ver2003);
xlsCreator1.CopySheet(0, 1, "コピーしたシート");
xlsCreator1.CopySheet(3, 5, "");
xlsCreator1.CloseBook(true);
```

注意事項

・コピー元シートにグラフが含まれる場合、グラフはコピーされますが、グラフ内のシート参照については変更されません。そのためグラフ内でセルのデータを参照している場合、そのシート参照が「#REF」となります。
・Excel ファイルのブック間、シート間のコピー・貼り付けには対応しておりません。

ActiveSheet プロパティ

アクティブ シートをシート番号を取得、または、設定します。

書式

[VB.NET]
Public Property ActiveSheet As Integer

[C#]
Public System.Int32 ActiveSheet [get, set]

設定値

アクティブ シートのシート番号を取得する変数を設定します。
アクティブにするシート番号（0 ～）を設定します。

解説

アクティブ シートは、Excel ファイルを Excel で開いた時に最初に表示されるシートです。
セルの書式設定や値を設定・取得を行うシートの設定には、SheetNo プロパティを使用します。

参照

SheetNo プロパティ

コーディング例

アクティブ シートの番号の取得を行います。

```
[VB.NET]
Dim nActiveSheetNo As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nActiveSheetNo = XlsCreator1.ActiveSheet
XlsCreator1.CloseBook(True)

[C#]
Int32 nActiveSheetNo;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nActiveSheetNo = xlsCreator1.ActiveSheet;
xlsCreator1.CloseBook(true);
```

2 シート目をアクティブ シートに設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.ActiveSheet = 1
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.ActiveSheet = 1;
xlsCreator1.CloseBook(true);
```

SheetCount プロパティ

Excel ファイルのシート総数を取得します。

書式

[VB.NET]
```
Public ReadOnly Property SheetCount As Integer
```

[C#]
```
Public System.Int32 SheetCount [get]
```

コーディング例

Excel ファイル内のシート総数の取得を行ないます。

[VB.NET]
```
Dim nSheetCount As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nSheetCount = XlsCreator1.SheetCount
XlsCreator1.CloseBook(True)
```

[C#]
```
Int32 nSheetCount;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nSheetCount = xlsCreator1.SheetCount;
xlsCreator1.CloseBook(true);
```

RefSheet プロパティ

参照するシート名を設定します。

書式

[VB.NET]
Public WriteOnly Property RefSheet As String

[C#]
Public string RefSheet [set]

設定値

参照するシート名を設定します。
複数のシートを参照する場合には "/" スラッシュで区切って設定します。

解説

Func メソッドの指定セルに参照シートを含む場合には、必ず Func メソッドを設定する前に参照されているシート名を設定してください。

参照

Func メソッド

コーディング例

シートに対し各種設定を行います。
4 番目のシート名を 4 月に設定し、セル A1 に数値 400 を入れます。
次に 1 番目のシート(Sheet1)セル A1 に数値 100 を入れ、セル A1 に数値 100 を入れます。1 番目のシート(Sheet1)に 他のシートのセル値を参照する関数を書き込みます。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 4, ExcelCreator.xlVersion.ver2003)

XlsCreator1.SheetNo = 3
XlsCreator1.SheetName = "４月"
XlsCreator1.Cell("A1").Long = 400

XlsCreator1.SheetNo = 0
XlsCreator1.Cell("A1").Long = 100
XlsCreator1.RefSheet = "Sheet1/４月"
XlsCreator1.Cell("A2").Func("=SUM(Sheet1:４月!A1)", "")

XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 4, ExcelCreator.xlVersion.ver2003);

xlsCreator1.SheetNo = 3;
xlsCreator1.SheetName = "４月";
xlsCreator1.Cell("A1").Long = 400;

xlsCreator1.SheetNo = 0;
xlsCreator1.Cell("A1").Long = 100;
xlsCreator1.RefSheet = "Sheet1/４月";
xlsCreator1.Cell("A2").Func("=SUM(Sheet1:４月!A1)", null);

xlsCreator1.CloseBook(true);
```

SheetNo プロパティ

操作対象となるシートの選択を行います。

書式

[VB.NET]
Public WriteOnly Property SheetNo As Integer

[C#]
Public System.Int32 SheetNo [set]

設定値

操作対象とするシートを値で設定します。
左端シートを 0 として、全シート数 － 1 まで設定することができます。

解説

既存 Excel ファイルをオープンした場合には 0 が初期状態です。

コーディング例

シートに対し各種設定を行います。
4 番目のシート名を 4 月に設定し、セル A1 に数値 400 を入れます。
シート名を取得し、メッセージボックスに表示します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 4, ExcelCreator.xlVersion.ver2003)
XlsCreator1.SheetNo = 3
XlsCreator1.SheetName = "４月"
XlsCreator1.Cell("A1").Long = 400
XlsCreator1.CloseBook(True)

XlsCreator1.ReadBook(Application.StartupPath & "¥ExcelFile.xls")
XlsCreator1.SheetNo = 0
MsgBox(XlsCreator1.SheetName)
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 4, ExcelCreator.xlVersion.ver2003);
xlsCreator1.SheetNo = 3;
xlsCreator1.SheetName = "４月";
xlsCreator1.Cell("A1").Long = 400;
xlsCreator1.CloseBook(true);

xlsCreator1.ReadBook(Application.StartupPath + @"¥ExcelFile.xls");
xlsCreator1.SheetNo = 0;
MessageBox.Show(xlsCreator1.SheetName);
xlsCreator1.CloseBook(true);
```

SheetNo2 プロパティ

シート名からシート番号を取得します。

書式

[VB.NET]
Public ReadOnly Property SheetNo2(ByVal sheetname As String) As Integer

[C#]
Public System.Int32 SheetNo2(System.String sheetname)

設定値

sheetname
　シート番号を取得する元のシート名を示す文字列を設定します。

解説

シート番号は左端から 0 から数えた番号が取得されます。
設定したシート名が Excel ファイルに存在しない場合、戻り値は -1 となります。

コーディング例

「Sheet3」のシート番号を取得します。

```
[VB.NET]
Dim nSheetNo As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nSheetNo = XlsCreator1.SheetNo2("Sheet3")
XlsCreator1.CloseBook(True)


[C#]
Int32 nSheetNo;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nSheetNo = xlsCreator1.SheetNo2("Sheet3");
xlsCreator1.CloseBook(true);
```

SheetName プロパティ

シート名の取得、または、設定します。

書式

[VB.NET]
Public Property SheetName As String

[C#]
Public string SheetName [get, set]

設定値

半角文字数 1 ～ 31 文字で設定します。

コーディング例

シート名を 4 月に設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 4, ExcelCreator.xlVersion.ver2003)
XlsCreator1.SheetName = "４月"
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 4, ExcelCreator.xlVersion.ver2003);
xlsCreator1.SheetName = "４月";
xlsCreator1.CloseBook(true);
```

シート名を取得し、メッセージボックスに表示します。

```
[VB.NET]
XlsCreator1.ReadBook(Application.StartupPath & "¥ExcelFile.xls")
MsgBox(XlsCreator1.SheetName)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.ReadBook(Application.StartupPath + @"¥ExcelFile.xls");
MessageBox.Show(xlsCreator1.SheetName);
xlsCreator1.CloseBook(true);
```

SheetName2 プロパティ

シート番号からシート名を取得します。

書式

[VB.NET]
Public Property sheetname SheetName2(Int32 sheetno)

[C#]
Public property String* SheetName2(Int32 sheetno) [get]

設定値

sheetno
　シート番号を示す数値を設定します。

解説

シート番号には Excel ファイル内の左端シートを 0 として、全シート数 － 1 まで設定することができます。

コーディング例

シート番号からシート名の取得を行います。

```
[VB.NET]
Dim strSheetName As String
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
strSheetName = XlsCreator1.SheetName2(2)
XlsCreator1.CloseBook(True)


[C#]
string strSheetName;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
strSheetName = xlsCreator1.SheetName2(2);
xlsCreator1.CloseBook(true);
```

RowCopy メソッド

行をコピーします。

書式

[VB.NET]
Public Sub RowCopy(ByVal sy As Integer, ByVal dy As Integer)

[C#]
Public void RowCopy(System.Int32 sy , System.Int32 dy)

設定値

sy
　コピー元の行番号値（0 ～）を設定します。
dy
　コピー先の行番号値（0 ～）を設定します。

解説

引数 sy の設定行を、引数 dy の行へコピーします。
引数 dy に -1 を設定した時は、引数 sy の設定行がメモリ上にコピーされます。コピーした行は、RowPaste メソッドで設定した行に貼り付けることができます。

参照

RowPaste メソッド

コーディング例

1 行目を 20 行目にコピー、1 行目をメモリ上にコピーし 21 行目に貼り付けます。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Func("=sum(B1:C1)", "")  'コピー元を設定
XlsCreator1.RowCopy(0, 19)
XlsCreator1.RowCopy(0, -1)
XlsCreator1.RowPaste(20)
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Func("=sum(B1:C1)", null);  //コピー元を設定
xlsCreator1.RowCopy(0, 19);
xlsCreator1.RowCopy(0, -1);
xlsCreator1.RowPaste(20);
xlsCreator1.CloseBook(true);
```

注意事項

・コピー元セルのセル属性と値のみコピーします。
・コピー元セルに変数名が設定されている場合、変数名はコピーされません。
・Excel ファイルのブック間、シート間のコピー・貼り付けには対応しておりません。

RowPaste メソッド

RowCopy メソッドでコピーした行を貼り付けます。

書式

[VB.NET]
Public Sub RowPaste(ByVal dy As Integer)

[C#]
Public void RowPaste(System.Int32 dy)

設定値

dy
貼り付けを行う行番号値 (0 ～) を設定します。

解説

RowCopy メソッドにより、メモリ上にコピーされた行を 引数 dy に貼り付けます。元の行は上書きされます。

参照

RowCopy メソッド

コーディング例

1 行目を 20 行目にコピー、1 行目をメモリ上にコピーし 21 行目に貼り付けます。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Func("=sum(B1:C1)", "")  'コピー元を設定
XlsCreator1.RowCopy(0, 19)
XlsCreator1.RowCopy(0, -1)
XlsCreator1.RowPaste(20)
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Func("=sum(B1:C1)", null);  //コピー元を設定
xlsCreator1.RowCopy(0, 19);
xlsCreator1.RowCopy(0, -1);
xlsCreator1.RowPaste(20);
xlsCreator1.CloseBook(true);
```

注意事項

・コピー元セルのセル属性と値のみコピーします。
・コピー元セルに変数名が設定されている場合、変数名はコピーされません。
・Excel ファイルのブック間、シート間のコピー・貼り付けには対応しておりません。

RowDelete メソッド

行を削除します。

書式

[VB.NET]
Public Sub RowDelete(ByVal sy As Integer, ByVal count As Integer)

[C#]
Public void RowDelete(System.Int32 sy , System.Int32 count)

設定値

sy
　削除する開始行番号値（0 ～）を設定します。
count
　削除する行数値（1 ～）を設定します。

解説

引数 sy の設定行から、引数 count の行数分を削除します。

コーディング例

2 行目から 1 行のみ削除、2 行目から 2 行分削除します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.RowDelete(1, 0)
XlsCreator1.RowDelete(1, 2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.RowDelete(1, 0);
xlsCreator1.RowDelete(1, 2);
xlsCreator1.CloseBook(true);
```

注意事項

・行削除を行っても、Excel シート上に設定されているオートシェイプや画像の位置は移動されません。

RowClear メソッド

行をクリア（初期化）します。

書式

[VB.NET]
Public Sub RowClear(ByVal sy As Integer, ByVal count As Integer)

[C#]
Public void RowClear(System.Int32 sy , System.Int32 count)

設定値

sy
　クリアする開始行番号値（0 ～）を設定します。
count
　クリアする行数値（1 ～）を設定します。

解説

引数 sy の設定行から、引数 count の行数分をクリア（初期化）します。

コーディング例

2 行目から 1 行のみクリア、2 行目から 2 行分クリアします。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.RowClear(1, 0)
XlsCreator1.RowClear(1, 2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.RowClear(1, 0);
xlsCreator1.RowClear(1, 2);
xlsCreator1.CloseBook(true);
```

RowInsert メソッド

行を挿入します。

書式

[VB.NET]
Public Sub RowInsert(ByVal sy As Integer, ByVal count As Integer)

[C#]
Public void RowInsert(System.Int32 sy , System.Int32 count)

設定値

sy
　挿入する開始行番号値（0 ～）を設定します。
count
　挿入する行数値（1 ～）を設定します。

解説

引数 sy の設定行から、引数 count の行数分を挿入します。

コーディング例

2 行目から 1 行のみ挿入、2 行目から 2 行分挿入します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.RowInsert(1, 0)
XlsCreator1.RowInsert(1, 2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.RowInsert(1, 0);
xlsCreator1.RowInsert(1, 2);
xlsCreator1.CloseBook(true);
```

注意事項

・行挿入を行っても、Excel シート上に設定されているオートシェイプや画像の位置は移動されません。

ColumnCopy メソッド

セル列をコピーします。

書式

[VB.NET]
Public Sub ColumnCopy(ByVal sx As Integer, ByVal dx As Integer)

[C#]
Public void ColumnCopy(System.Int32 sx , System.Int32 dx)

設定値

sx
　コピー元の列番号値（0 ～）を設定します。
dx
　コピー先の列番号値（0 ～）を設定します。

解説

引数 sx の設定列を、引数 dx の列へコピーします。
引数 dx に -1 を設定した時は、引数 sx の設定列がメモリ上にコピーされます。
コピーした列は、ColumnPaste メソッドで設定した列に貼り付けることができます。

参照

ColumnPaste メソッド

コーディング例

1 列目を 20 列目にコピーします。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Func("=sum(B1:C1)", "")  'コピー元を設定
XlsCreator1.ColumnCopy(0, 19)
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Func("=sum(B1:C1)", null);  //コピー元を設定
xlsCreator1.ColumnCopy(0, 19);
xlsCreator1.CloseBook(true);
```

注意事項

・コピー元セルのセル属性と値のみコピーします。
・コピー元セルに変数名が設定されている場合、変数名はコピーされません。
・Excel ファイルのブック間、シート間のコピー・貼り付けには対応しておりません。

ColumnPaste メソッド

ColumnCopy メソッドでコピーした列を貼り付けます。

書式

[VB.NET]
Public Sub ColumnPaste(ByVal dx As Integer)

[C#]
Public void ColumnPaste(System.Int32 dx)

設定値

dx
貼り付けを行う列番号値（0 ～）を設定します。

解説

ColumnCopy メソッドによりメモリ上にコピーされた列を、引数 dx に貼り付けます。このとき、元の列は上書きされます。

参照

ColumnPaste メソッド

コーディング例

1 列目を 20 列目にコピーし、1 列目をメモリ上にコピーし 21 列目に貼り付けます。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Func("=sum(B1:C1)", "")  'コピー元を設定
XlsCreator1.ColumnCopy(0, 19)
XlsCreator1.ColumnCopy(0, -1)
XlsCreator1.ColumnPaste(20)
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Func("=sum(B1:C1)", null);  //コピー元を設定
xlsCreator1.ColumnCopy(0, 19);
xlsCreator1.ColumnCopy(0, -1);
xlsCreator1.ColumnPaste(20);
xlsCreator1.CloseBook(true);
```

注意事項

・コピー元セルのセル属性と値のみコピーします。
・コピー元セルに変数名が設定されている場合、変数名はコピーされません。
・Excel ファイルのブック間、シート間のコピー・貼り付けには対応しておりません。

ColumnDelete メソッド

セル列を削除します。

書式

[VB.NET]
Public Sub ColumnDelete(ByVal sx As Integer, ByVal count As Integer)

[C#]
Public void ColumnDelete(System.Int32 sx , System.Int32 count)

設定値

sx
　削除する開始列番号値（0 ～）を設定します。
count
　削除する列数値（1 ～）を設定します。

解説

引数 sx の設定列から、引数 count の列数分を削除します。

コーディング例

E 列を削除し、E 列から 2 列分削除します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.ColumnDelete(4, -1)
XlsCreator1.ColumnDelete(4, 2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.ColumnDelete(4, -1);
xlsCreator1.ColumnDelete(4, 2);
xlsCreator1.CloseBook(true);
```

注意事項

・列削除を行っても、Excel シート上に設定されているオートシェイプや画像の位置は移動されません。

ColumnClear メソッド

セル列をクリア（初期化）します。

書式

[VB.NET]
Public Sub ColumnClear(ByVal sx As Integer, ByVal count As Integer)

[C#]
Public void ColumnClear(System.Int32 sx , System.Int32 count)

設定値

sx
　クリアする開始列番号値（0 ～）を設定します。
count
　クリアする列数値（1 ～）を設定します。

解説

引数 sx の設定列から、引数 count の列数分をクリア（初期化）します。

コーディング例

E 列をクリア、E 列から 2 列分クリアします。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.ColumnClear(4, -1)
XlsCreator1.ColumnClear(4, 2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.ColumnClear(4, -1);
xlsCreator1.ColumnClear(4, 2);
xlsCreator1.CloseBook(true);
```

注意事項

・列挿入を行っても、Excel シート上に設定されているオートシェイプや画像の位置は移動されません。

ColumnInsert メソッド

列を挿入します。

書式

[VB.NET]
Public Sub ColumnInsert(ByVal sx As Integer, ByVal count As Integer)

[C#]
Public void ColumnInsert(System.Int32 sx , System.Int32 count)

設定値

sx
　挿入する開始列番号値（0 ～）を設定します。
count
　挿入する列数値（1 ～）を設定します。

解説

引数 sx の設定列から、引数 count の列数分を挿入します。

コーディング例

B 列から 1 列のみ挿入、B 列から 2 列分挿入します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.ColumnInsert(1, 1)
XlsCreator1.ColumnInsert(1, 2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.ColumnInsert(1, 1);
xlsCreator1.ColumnInsert(1, 2);
xlsCreator1.CloseBook(true);
```

Size メソッド

用紙情報の設定を行います。

書式

[VB.NET]
Public Sub Size(ByVal nOrientation As Integer, ByVal nScale As Integer, ByVal nPaperSize As System.Drawing.Printing.PaperKind nPaperSize)

[C#]
public void Size(System.Int32 nOrientation, System.Int32 nScale, System.Drawing.Printing.PaperKind nPaperSize)

設定値

nScale
　印刷時の拡大 / 縮小率（10 ～ 400%）を設定します。
nPaperSize
　用紙サイズを System.Drawing.Printing.PaperKind のメンバで設定します。
nOrientation
　印刷の向きを設定します。

定数	値	内容
orLandscape	0	横
orPortrait	2	縦

解説

ユーザー定義用紙サイズは、SizeFree メソッドで設定します。

参照

SizeFree メソッド、Margin メソッド

コーディング例

印刷用紙を B4 横、倍率 70% に設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Size(ExcelCreator.xlOrientation.orLandscape, 70, System.Drawing.Printing.PaperKind.B4)
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Size(ExcelCreator.xlOrientation.orLandscape, 70, System.Drawing.Printing.PaperKind.B4);
xlsCreator1.CloseBook(true);
```

SizeFree メソッド

ユーザー定義用紙の設定を行います。

書式

[VB.NET]
Public Sub SizeFree(ByVal Width As Double, ByVal Height As Double, ByVal strPaperName As String)

[C#]
public void SizeFree(System.Double Width, System.Double Height, System.String strPaperName)

設定値

Width
　ユーザー定義用紙の幅を設定します。設定単位はミリ（mm）です。
Height
　ユーザー定義用紙の長さを設定します。設定単位はミリ（mm）です。
strPaperName
　ユーザー定義用紙名を設定します。

解説

・WindowsNT/2000/XP の場合
プリントサーバーのプロパティで登録した用紙名を引数 strPaperName に、用紙サイズ（高さ・幅）を引数 Width と Height に設定してください。
・Windows98/Me の場合
引数 Width と Height には、ユーザー定義の用紙サイズ（高さ・幅）を設定してください。なお、引数 strPaperName には、"" を設定してください。

規定用紙の場合は、Size メソッドにて設定します。

参照

Size メソッド、Margin メソッド

コーディング例

297.0(mm)× 210.0(mm) のユーザー定義の用紙サイズを設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.SizeFree(297, 210, "ユーザー定義用紙")
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.SizeFree(297, 210, "ユーザー定義用紙");
xlsCreator1.CloseBook(true);
```

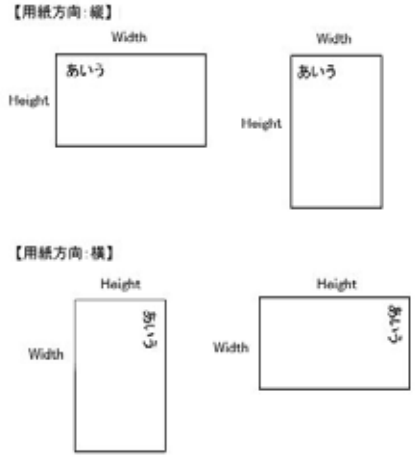
Windows98/Me にて、297.0(mm)× 210.0(mm) のユーザー定義の用紙サイズを設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.SizeFree(297, 210, "")
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.SizeFree(297, 210, "");
```

```
xlsCreator1.CloseBook(true);
```

注意事項

設定する幅・高さは、用紙方向の向きに関係します。
例えば幅 100mm、長さ 200mm のユーザー定義用紙の場合、用紙方向が縦ならば引数 Width に 100 、引数 Height に 200 を設定します。用紙方向が横ならば引数 Width に 200 、引数 Height に 100 を設定します。



Margin メソッド

シート単位でページ余白の設定を行います。

書式

[VB.NET]
Public Sub Margin(ByVal left As Integer, ByVal right As Integer, ByVal top As Integer, ByVal bottom As Integer, ByVal header As Integer, ByVal footer As Integer)

[C#]
public void Margin(System.Int32 left , System.Int32 right , System.Int32 top , System.Int32 bottom , System.Int32 header , System.Int32 footer)

設定値

left
　左余白を示す値を mm 単位で設定します。
right
　右余白を示す値を mm 単位で設定します。
top
　上余白を示す値を mm 単位で設定します。
bottom
　下余白を示す値を mm 単位で設定します。
header
　ヘッダー余白を示す値を mm 単位で設定します。
footer
　フッター余白を示す値を mm 単位で設定します。

省略時には -1 を設定します。デフォルト値が設定されます。

解説

設定を省略するとページ余白は、デフォルト値の上・下 は 25mm、左・右 は 20mm、ヘッダー・フッターは 13mm になります。

参照

Size メソッド、SizeFree メソッド

コーディング例

ページとヘッダー・フッター余白の設定を行います。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Margin(15, 10, 10, 10, 2, 3)
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Margin(15, 10, 10, 10, 2, 3);
xlsCreator1.CloseBook(true);
```

Center プロパティ

ページ中央を設定します。

書式

[VB.NET]
Public WriteOnly Property Center As ExcelCreator.xlPageCenter

[C#]
public ExcelCreator.xlPageCenter Center [set]

設定値

水平、垂直方向の印刷位置を、xlPageCenter 列挙体のメンバで設定します。

定数	値	内容
pcNone	0	指定なし。(デフォルト値)
pcHorz	1	水平方向に中央寄せを行います。
pcVert	2	垂直方向に中央寄せを行います。

解説

印刷の中央位置を設定します。
pcHorz と pcVert と両方設定する場合は「 + 」演算子で結合して設定します。

コーディング例

印刷位置を水平、垂直方向に中央に設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Page.Attr.Center = ExcelCreator.xlPageCenter.pcHorz + ExcelCreator.xlPageCenter.pcVert
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
XlsCreator1.Page.Attr.Center = (ExcelCreator.xlPageCenter)(ExcelCreator.xlPageCenter.pcHorz +
(Int32)ExcelCreator.xlPageCenter.pcVert);
xlsCreator1.CloseBook(true);
```

PageOrder プロパティ

ページの印刷方向を設定します。

書式

[VB.NET]
Public WriteOnly Property PageOrder As ExcelCreator.xlPageOrder

[C#]
public ExcelCreator.xlPageOrder PageOrder [set]

設定値

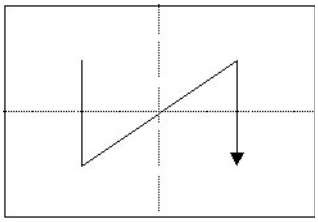
ページの方向を設定する値を、xlPageOrder 列挙体のメンバで設定します。

定数	値	内容
poRightFromLeft	0	左から右へ印刷します。
poBottomFromTop	1	上から下へ印刷します。

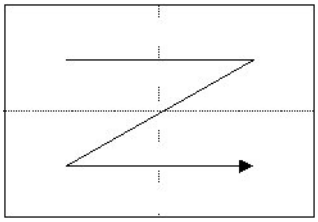
解説

省略した場合は、デザインファイル内の設定が適用されます。

左から右へ印刷



上から下へ印刷



参照

Size メソッド、SizeFree メソッド、Margin メソッド

コーディング例

次のプログラムでは、ページの方向を「上から下」に設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PageOrder = ExcelCreator.xlPageOrder.poBottomFromTop
XlsCreator1.CloseBook(True)

[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PageOrder = ExcelCreator.xlPageOrder.poBottomFromTop;
xlsCreator1.CloseBook(true);
```

PrintArea メソッド

シートの印刷範囲を設定します。

書式

[VB.NET]
Public Sub PrintArea(ByVal sx As Integer, ByVal sy As Integer, ByVal ex As Integer, ByVal ey As Integer)

[C#]
Public void PrintArea(System.Int32 sx , System.Int32 sy , System.Int32 ex , System.Int32 ey)

設定値

sx
　印刷範囲の開始列位置を設定します。
sy
　印刷範囲の開始行位置を設定します。
ex
　印刷範囲の終了列位置を設定します。
ey
　印刷範囲の終了行位置を設定します。

解説

設定するシートは SheetNo プロパティのシート番号が対象です。

参照

SheetNo プロパティ

コーディング例

1 番目のシートと 2 番目のシートの印刷範囲の設定を行います。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)

XlsCreator1.SheetNo = 0
XlsCreator1.PrintArea(0, 0, 9, 19)

XlsCreator1.SheetNo = 1
XlsCreator1.PrintArea(0, 0, 29, 19)

XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);

XlsCreator1.SheetNo = 0;
xlsCreator1.PrintArea(0, 0, 9, 19);

XlsCreator1.SheetNo = 1;
xlsCreator1.PrintArea(0, 0, 29, 19);

xlsCreator1.CloseBook(true);
```

Header メソッド

シートのヘッダーを設定します。

書式

[VB.NET]
Public Sub Header(ByVal left As String, ByVal center As String, ByVal right As String)

[C#]
Public void Header(System.String left , System.String center , System.String right)

設定値

left
　左ヘッダーを示す文字列を設定します。
center
　中央ヘッダーを示す文字列を設定します。
right
　右ヘッダーを示す文字列を設定します。

各文字列は、下記の特殊な文字列に対応しています。

書式コード	内容
&E	二重下線
&X	上付き文字
&Y	下付き文字
&B	ボールド
&I	イタリック
&U	下線
&S	取消線
&P	ページ番号
&P+99	ページ番号に値を加えて表示 **例**:先頭ページ番号を 3 とする場合 … &P + 3)
&P-99	ページ番号に値を引いて表示
&D	現在の日付
&T	現在の時刻
&A	シート見出し
&&	&(アンパサンド) を表示
&"フォント"	フォントを指定
&99	ポイント数を指定
&N	全体のページ数

解説

文字数の制限は書式コードを含めて 255 文字までです。
なお、引数 left、center、right に "" を設定した場合は元の設定が適用されます。

参照

Footer メソッド

コーディング例

シートのヘッダーを設定します。

[VB.NET]
```
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", Application.StartupPath & "¥ExcelFile.xls")
XlsCreator1.SheetNo = 0
XlsCreator1.Header("&""ＭＳ Ｐ明朝""&16ExcelCreator 新機能一覧ヘッダー", "&A", "&D")
XlsCreator1.CloseBook(True)
```

```
[C#]
XlsCreator1.OpenBook(Application.StartUpPath + @"¥ExcelFile.xls",
                     Application.StartUpPath + @"¥ExcelTemplate.xls");
XlsCreator1.SheetNo = 0;
XlsCreator1.Header("&¥"ＭＳ Ｐ明朝¥"&16ExcelCreator 新機能一覧ヘッダー", "&A", "&D");
XlsCreator1.CloseBook(true);
```

Footer メソッド

シートのフッタを設定します。

書式

[VB.NET]
Public Sub Footer(ByVal left As String, ByVal center As String, ByVal right As String)

[C#]
Public void Footer(System.String left , System.String center , System.String right)

設定値

left
　左フッタを示す文字列を設定します。
center
　中央フッタを示す文字列を設定します。
right
　右フッタを示す文字列を設定します。

各文字列は、下記の特殊な文字列に対応しています。

書式コード	内容
&E	二重下線
&X	上付き文字
&Y	下付き文字
&B	ボールド
&I	イタリック
&U	下線
&S	取消線
&P	ページ番号
&P+99	ページ番号に値を加えて表示 **例:**先頭ページ番号を 3 とする場合 … &P + 3)
&P-99	ページ番号に値を引いて表示
&D	現在の日付
&T	現在の時刻
&A	シート見出し
&&	&(アンパサンド)を表示
&"フォント"	フォントを指定
&99	ポイント数を指定
&N	全体のページ数

解説

文字数の制限は書式コードを含めて 255 文字までです。
なお、引数 left、center、right に "" を設定した場合は元の設定が適用されます。

参照

Header メソッド

コーディング例

シートのフッタを設定します。

```
[VB.NET]
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
XlsCreator1.SheetNo = 0
XlsCreator1.Footer("&""ＭＳ Ｐ明朝""&16ExcelCreator 新機能一覧フッター", "&A", "&D")
XlsCreator1.CloseBook(True)
```

```
[C#]
XlsCreator1.OpenBook(Application.StartUpPath + @"¥ExcelFile.xls",
                     Application.StartUpPath + @"¥ExcelTemplate.xls");
XlsCreator1.SheetNo = 0;
XlsCreator1.Footer("&¥"ＭＳ Ｐ明朝¥"&16ExcelCreator 新機能一覧フッタ", "&A", "&D");
XlsCreator1.CloseBook(true);
```

MaxData メソッド

データ（文字列 / 値 / 計算式）が設定されているセルの最終座標を取得します。

書式

[VB.NET]
Public ReadOnly Property MaxData(ByVal pt As ExcelCreator.xlPoint) As System.Drawing.Size

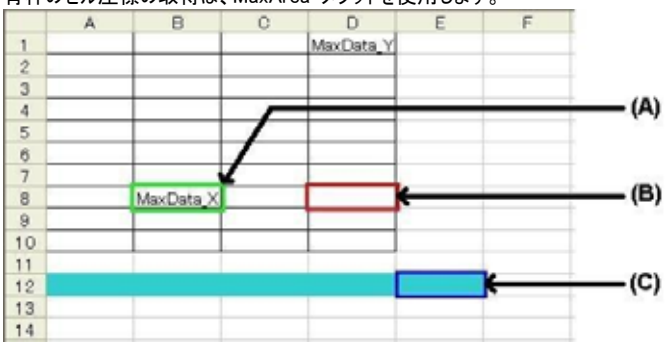
[C#]
Public System.Drawing.Size MaxData(ExcelCreator.xlPoint pt)

設定値

定数	内容
xlMaxPoint	データが設定されているセルのうち、最下部のセルと最右部のセルが交わる地点にあるセル座標を取得します。
xlEndPoint	データが設定されているセルのうち、最も下にあり、かつ、最も右側にあるセル座標を取得します。

解説

x 座標は Size.Width プロパティに、y 座標は Size.Height プロパティにそれぞれ設定されます。
xlEndPoint 設定時、データが設定されているセルのうち最も下にあり、かつ、最も右側にあるセル座標を取得します（黄緑枠のセル座標）。
また、xlMaxPoint 設定時、データが設定されているセルのうち、最下部のセルと最右部のセルが交わる地点にあるセル座標を取得します（赤枠のセル座標）。
青枠のセル座標の取得は、MaxArea メソッドを使用します。



参照

MaxArea メソッド

コーディング例

データが設定されているセルのマックスポイントを取得します。

[VB.NET]
```
Dim sz As System.Drawing.Size
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
sz = XlsCreator1.MaxData(ExcelCreator.xlPoint.ptMaxPoint)
MsgBox("x 座標:" + sz.Width.ToString() + " / y 座標:" + sz.Height.ToString())
XlsCreator1.CloseBook(True)
```

[C#]
```
System.Drawing.Size sz;
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
sz = xlsCreator1.MaxData(ExcelCreator.xlPoint.ptMaxPoint);
MessageBox.Show("x 座標:" + sz.Width.ToString() + " / y 座標:" + sz.Height.ToString());
xlsCreator1.CloseBook(true);
```

MaxArea メソッド

データ（文字列 / 値 / 計算式）の他、罫線やセルの塗りつぶし等のセル書式が設定されている、最終セルの座標を取得します。

書式

[VB.NET]
Public ReadOnly Property MaxArea(ByVal pt As ExcelCreator.xlPoint) As System.Drawing.Size

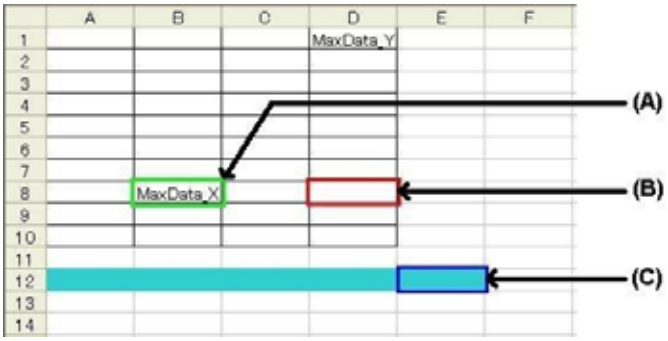
[C#]
Public System.Drawing.Size MaxArea(ExcelCreator.xlPoint pt)

設定値

定数	内容
xlMaxPoint	データやセル書式が設定されているセルのうち、最下部のセルと、最右部のセルが交わる地点にあるセル座標を取得します。
xlEndPoint	データやセル書式が設定されているセルのうち、最も下にあり、かつ、最も右側にあるセル座標を取得します。

解説

x 座標は Size.Width プロパティに、y 座標は Size.Height プロパティにそれぞれ設定されます。
xlEndPoint 設定時、データが設定されているセルのうち最も下にあり、かつ、最も右側にあるセル座標を取得します（青枠のセル座標）。
また、xlMaxPoint 設定時、データが設定されているセルのうち、最下部のセルと最右部のセルが交わる地点にあるセル座標を取得します（青枠のセル座標）。
赤枠や黄緑枠のセル座標の取得は、MaxData メソッドを使用します。



参照

MaxData メソッド

コーディング例

データやセル書式が設定されているセルのエンドポイントを取得します。

```
[VB.NET]
Dim sz As System.Drawing.Size
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
sz = XlsCreator1.MaxArea(ExcelCreator.xlPoint.ptEndPoint)
MsgBox("x 座標:" + sz.Width.ToString() + " / y 座標:" + sz.Height.ToString())
XlsCreator1.CloseBook(True)


[C#]
System.Drawing.Size sz;
```

```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
sz = xlsCreator1.MaxArea(ExcelCreator.xlPoint.ptEndPoint);
MessageBox.Show("x 座標:" + sz.Width.ToString() + " / y 座標:" + sz.Height.ToString());
xlsCreator1.CloseBook(true);
```

Error イベント

ExcelCreator 5.0 for .NET の実行時に、不正な処理を行った場合に発生します。実行時にエラー番号値の取得も可能です。

書式

[VB.NET]
```
Protected object_Error(ByVal i1 As Object, ByVal err As XlsEventArgs)
```

[C#]
```
protected void object_Error(object i1 , XlsEventArgs err)
```

設定値

エラー番号は、次の通りです。

定数	値	内容
errDontOpen	1	ファイルがオープンできない
errFileHandle	2	無効なファイルハンドル
errNotOpen	3	未オープンエラー
errSheetNo	4	シート番号エラー
errWrite	5	書き込み時のエラー
errAction	6	動作モードエラー
errDataCat	7	データ種別エラー
errValue	8	エラー値
errClear	20	クリアエラー
errCopy	21	コピーエラー
errPaste	22	貼り付けエラー
errInsert	23	挿入エラー
errDelete	24	削除エラー
errLength	31	長さエラー
errLocate	32	座標エラー
errAttr	33	属性番号エラー
errParam	34	パラメータエラー
errNoData	35	データが無い
errEndOfData	36	データの終わり
errVarPoint	37	指定した変数存在しない
errParame	34	パラメータエラー
errBreakCount	38	改ページ数が制限超えた
errMemory1	40	メモリー不足エラー1
errMemory2	41	メモリー不足エラー2
errOther	50	内部エラー
errFunction	100	計算式形式エラー
errCreate	300	ファイル作成エラー
errTempCreate	301	Temp ファイル作成エラー
errTempOpen	302	Temp ファイルオープンエラー
errStream	303	ストリームエラー
errProtect	304	保護されたファイルです
errDoubleOpen	305	二重オープンです
errMutex	400	ミューテックスエラー
errMaxFileName	401	ファイル名が長すぎます
errInvalidFileName	402	無効なファイル名
errEmbedFile	403	埋込みバイナリエラーです

解説

エラー発生時にエラー番号を取得する場合、XlsEventArgs 列挙体の ErrorNo プロパティの値を取得します。

参照

ErrorNo プロパティ、ErrorMessage プロパティ

コーディング例

```
[VB.NET]
Friend WithEvents XlsCreator1 As ExcelCreator.XlsCreator
Private Sub XlsCreator1_Error(ByVal sender As Object, ByVal err As XlsEventArgs) Handles XlsCreator1.Error
    MessageBox.Show(err.ErrorNo.ToString() + " : " + XlsCreator1.ErrorMessage, "Error イベント")
End Sub

[C#]
private ExcelCreator.XlsCreator xlsCreator1 = new ExcelCreator.XlsCreator();
xlsCreator1.Error += new XlsEventHandler (XlsCreator1_Error);
protected void XlsCreator1_Error(System.Object i1 , XlsEventArgs err)
{
    MessageBox.Show(err.ErrorNo.ToString() + " : " + xlsCreator1.ErrorMessage, "Error イベント");
}
```

Progress イベント

PDF ファイル出力時の処理状況を通知します。

書式

[VB.NET]
Protected Overridable Sub OnProgress(ByVal e As ExcelCreator.XlsCreatorProgressArgs)

[C#]
protected virtual new void OnProgress (ExcelCreator.XlsCreatorProgressArgs e)

設定値

XlsCreatorProgressArgs の各メンバは、次の通りです。

Type	処理を識別する、xlProgressType 列挙体のメンバです。 pgFile = 2 … ファイルに出力中 pgData = 3 … データ出力中
Count	現在の進捗状況です。
Total	総ページ数や総シート数を表します。
Stop	処理を制御します。True を設定した場合、処理を中断します。

解説

Progress イベントは現在の処理の進行状況を通知します。処理の中断も可能です。
ProgressDialog プロパティで「表示しない」を設定した場合、このイベントが発生します。ユーザー独自の処理進行状況の表示を行う場合に、このイベントで制御を行います。

参照

ProgressDialog プロパティ

コーディング例

出力中に「中止」ボタンを押した時点で印刷を中断します
「中止」が押されていない場合、出力ページ数を表示します

[VB.NET]

```
Dim WithEvents ExcelCreator1 As New ExcelCreator.ExcelCreator()

Public btnBreak As Boolean
'「中止」ボタン
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    btnBreak = True
End Sub

Public Event Progress(ByVal sender As Object, ByVal e As ExcelCreator.XlsCreatorProgressArgs)

Private Sub ExcelCreator1_Progress(ByVal sender As Object, ByVal e As ExcelCreator.XlsCreatorProgressArgs)
Handles ExcelCreator1.Progress
    If (prg.Type = ExcelCreator.xlProgressType.pgData) Then
        'btnBreak は「中止」ボタン押下時に True が入ります
        If (btnBreak = True) Then
            '処理を中止します
            prg.Stop = True
        Else
            '出力ページ数の表示を行います
            Label1.Text = prg.Count.ToString() + " / " + prg.Total.ToString()
        End If
    End If
End Sub
```

```
[C#]
ExcelCreator1.Progress += new ExcelCreator.XlsCreatorProgressHandler(ExcelCreator1_Progress);

static bool btnBreak;
//「中止」ボタン
private void button1_Click(object sender, System.EventArgs e)
{
    btnBreak = true;
}

protected void ExcelCreator1_Progress(System.Object i1 , ExcelCreator.ExcelCreatorProgressArgs prg)
{
    if (prg.Type == ExcelCreator.xlProgressType.pgData)
    {
        //btnBreak は「中止」ボタン押下時に True が入ります
        if (btnBreak)
        {
            //処理を中止します
            prg.Stop = true;
        }
        else
        {
            //出力ページ数の表示を行います
            Label1.Text = prg.Count.ToString() + " / " + prg.Total.ToString();
        }
    }
}
```

Cell クラス

出力対象となるセルの範囲をセル位置（A1 参照形式）/ 変数名 / セル名で設定します。

書式

[VB.NET]
Public ReadOnly Property Cell(ByVal cell As String)
Public ReadOnly Property Cell(ByVal cell As String, ByVal cx As Integer, ByVal cy As Integer)

[C#]
Public Cell(String cell)
Public Cell(String cell , Int32 cx , Int32 cy)

設定値

cell
対象セルを文字列で設定します。
セル位置（A1 参照形式）、変数名、セル名で設定します。
cx
cell で設定した対象セルからの範囲を左、または右への移動量を設定します。
cy
cell で設定した対象セルからの範囲を上、または下への移動量を設定します。

解説

cell の範囲について
・変数名による指定方法
Excel ファイル内に変数が記述されている場合、変数名を設定することができます。変数は、先頭に ＊（アスタリスク）が２つ付加された文字列になります。
【例】Excel ファイルの A1 セルに **No と記述した場合
.Cell("**No")

また、変数名の先頭のアスタリスクの部分に別の文字列を使用したい場合、 KeyWord プロパティを使用することで、変数名のキーワードとなる先頭の ＊（アスタリスク）を任意の文字列に変更することができます。先頭キーワード文字列は、Excel で文字として認識可能で、かつ、半角文字のみ設定することができます。但し、¥ ' " , ^ . [] の半角文字は使用できません。
詳しくは KeyWord プロパティを参照してください。

・セル名による指定方法
Excel ファイル内に特定のセルに名前を設定している場合、名前による指定ができます。
【例】Excel ファイルの A1 セルに TEL と名前をつけた場合
.Cell("TEL")

・セル位置(A1 参照形式)による指定方法
範囲を表す演算子([:]コロン)、複数の範囲を表す演算子（[,]カンマ）を含めることができますが、他のブックやシートを直接参照することはできません。
【例】.Cell("A1")、.Cell("A1:B1")、.Cell("A1:C3")

複数の範囲を設定する場合、カンマで区切って設定することができます。
【例】.Cell("**No, TEL, A1:B1")

範囲指定を行った場合、属性については設定した範囲に対して設定されます。

cx, cy について
指定されたセル範囲に対してオフセットを設定することができます。オフセットを使用することによって、プログラム中にてセルを移動する際、 A1,A2,A3...と指定せずに、cx, cy にて指定されたセル範囲に対して + / −1 ～ n 移動したセルに設定することができます。 Excel の制限を無視した設定を行うとエラーが発生します。
【例】Excel ファイルの A1 セルに **No と記述している場合に F5 セルに設定

.Cell("**No", 5, 4)

このクラスの全てのメンバの一覧については、以下の Cell メンバを参照してください。

パブリックプロパティ

Break	改ページの設定 / 解除を設定します。	P117
ColWidth	セルの幅を設定します。	P118
RowHeight	セルの高さを設定します。	P119
ColWidth2	セルの幅を設定 / 取得します。	P120
RowHeight2	セルの高さを設定/ 取得します。	P121
Value	数値や文字列、日付などの値を設定します。値の取得も可能です。	P104
Str	文字列の取得、または設定します。	P105
Long	整数の取得、または、設定します。	P106
Double	実数の取得、または、設定します。	P107
AttrNo	セルの属性番号を取得します。	P109

パブリックメソッド

Copy	セルをコピーします。	P114
Paste	Copy メソッドでコピーしたセルを貼り付けます。	P115
Clear	セルをクリア（初期化）します。	P116
Value2	数値や文字列、日付などの値と属性番号を設定します。	P110
Str2	文字列と属性番号を設定します。	P111
Long2	整数と属性番号を設定します。	P112
Double2	実数と属性番号を設定します。	P113
Func	関数の書き込みをします。結果を示す値も同時に設定できます。	P108

Pos クラス

出力対象となるセルの範囲を座標形式で設定します。

書式

[VB.NET]
Public ReadOnly Property Pos(ByVal sx As Integer, ByVal sy As Integer)
Public ReadOnly Property Pos(ByVal sx As Integer, ByVal sy As Integer, ByVal ex As Integer, ByVal ey As Integer)

[C#]
Public ExcelCreator.XlsCell Pos(Int32 sx , Int32 sy)
Public ExcelCreator.XlsCell Pos(Int32 sx , Int32 sy , Int32 ex , Int32 ey)

設定値

sx , sy
　対象セルの座標を設定します。
ex , ey
　対象セルの範囲設定を行う場合、終了の座標を設定します。

解説

sx, ex には列を示す（0 ～ 255）の値を設定します。
sy, ey には行を示す（0 ～ 65535）の値を設定します。
Cell(″A1:B2″) = Pos(0,0,1,1) いずれも、対象セルは同じです。

Pos クラスで範囲指定を行った場合、属性は設定した範囲に対して設定されます。 値の設定や取得については、設定したセル範囲内で先頭座標（0,0）に最も近い座標が対象となります。

このクラスの全てのメンバの一覧については、以下の Pos メンバを参照してください。

パブリックプロパティ

Break	改ページの設定 / 解除を設定します。	P117
ColWidth	セルの幅を設定します。	P118
RowHeight	セルの高さを設定します。	P119
ColWidth2	セルの幅を設定 / 取得します。	P120
RowHeight2	セルの高さを設定/ 取得します。	P121
Value	数値や文字列、日付などの値を設定します。値の取得も可能です。	P104
Str	文字列の取得、または設定します。	P105
Long	整数の取得、または、設定します。	P106
Double	実数の取得、または、設定します。	P107
AttrNo	セルの属性番号を取得します。	P109

パブリックメソッド

Copy	セルをコピーします。	P114
Paste	Copy メソッドでコピーしたセルを貼り付けます。	P115
Clear	セルをクリア（初期化）します。	P116
Value2	数値や文字列、日付などの値と属性番号を設定します。	P110
Str2	文字列と属性番号を設定します。	P111
Long2	整数と属性番号を設定します。	P112
Double2	実数と属性番号を設定します。	P113
Func	関数の書き込みをします。結果を示す値も同時に設定できます。	P108

Value プロパティ

数値や文字列、日付などの値を設定 / 取得します。

書式

[VB.NET]
Public Property value = Value As object

[C#]
Public object Value [get, set]

設定値

数値や文字列を設定します。

解説

セルに設定されている属性に関係なく、値を設定することができます。文字列を設定した場合は文字列として値が設定され、数値で設定した場合は数値として設定されます。
また、値を取得することもできます。
計算式と共に計算結果をセルに設定する場合は、Func メソッドを使用してください。

コーディング例

各セルに実数・整数・文字列・関数を設定し作成した Excel ファイルのセルの値を取得します

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("B1").Value = 12.345     'セル B1 には 12.345 が入ります
XlsCreator1.Cell("B2").Value = 1000       'セル B2 には 1000 が入ります
XlsCreator1.Cell("B3").Value = "abcde"    'セル B3 には abcde が入ります
XlsCreator1.Cell("B4").Value = "=A2+999"  'セル B4 には =A2+999 が入ります"
XlsCreator1.CloseBook(True)

Dim objData As Object
XlsCreator1.ReadBook(Application.StartupPath & "¥ExcelFile.xls")
objData = XlsCreator1.Cell("B1").Value    'objData には 12.345000 が入ります
objData = XlsCreator1.Cell("B2").Value    'objData には 1000.000000 が入ります
objData = XlsCreator1.Cell("B3").Value    'objData には abcde が入ります
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("B1").Value = 12.345;    //セル B1 には 12.345 が入ります
xlsCreator1.Cell("B2").Value = 1000;      //セル B2 には 1000 が入ります
xlsCreator1.Cell("B3").Value = "abcde";   //セル B3 には abcde が入ります
xlsCreator1.Cell("B4").Value = "=A2+999"; //セル B4 には =A2+999 が入ります"
xlsCreator1.CloseBook(true);

object objData;
xlsCreator1.ReadBook(Application.StartupPath + @"¥ExcelFile.xls");
objData = xlsCreator1.Cell("B1").Value;   //objData には 12.345000 が入ります
objData = xlsCreator1.Cell("B2").Value;   //objData には 1000.000000 が入ります
objData = xlsCreator1.Cell("B3").Value;   //objData には abcde が入ります
xlsCreator1.CloseBook(true);
```

注意事項

・Cell クラスに変数名を設定した場合、セル値の取得には対応しておりません。

Str プロパティ

文字列を取得 / 設定します。

書式

[VB.NET]
Public Property Str As String

[C#]
Public string Str [get, set]

設定値

文字列を示す文字列を設定します。

解説

セルに計算式や関数式が設定されている場合は、その結果値を文字列で取得することができます。

コーディング例

A3 セルに値を設定し、作成したファイルのセルの値を取得します

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A3").Str = "abcde"     'セル A3 には abcde が入ります
XlsCreator1.CloseBook(True)

Dim str1 As String
XlsCreator1.ReadBook(Application.StartupPath & "¥ExcelFile.xls")
str1 = XlsCreator1.Cell("A3").Str        'str1 には abcde が入ります
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A3").Str = "abcde";     //セル A3 には abcde が入ります
xlsCreator1.CloseBook(true);

String str1;
xlsCreator1.ReadBook(Application.StartupPath + @"¥ExcelFile.xls");
str1 = xlsCreator1.Cell("A3").Str;        //str1 には abcde が入ります
xlsCreator1.CloseBook(true);
```

注意事項

・Cell クラスに変数名を設定した場合、セル値の取得には対応しておりません。

Long プロパティ

整数を取得 / 設定します。

書式

[VB.NET]
Public Property Long As int

[C#]
Public int Long [get, set]

設定値

整数を設定します。

解説

セルに計算式や関数式が設定されている場合は、その結果値を整数で取得することができます。

コーディング例

A1 セルに値を設定し、作成したファイルのセルの値を取得します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A2").Long = 1000      'セル A2 には 1000 が入ります
XlsCreator1.CloseBook(True)

Dim nLong As Long
XlsCreator1.ReadBook(Application.StartupPath & "¥ExcelFile.xls")
nLong = XlsCreator1.Cell("A2").Long     'nLong には 1000 が入ります
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A2").Long = 1000;      //セル A2 には 1000 が入ります
xlsCreator1.CloseBook(true);

int nValue;
xlsCreator1.ReadBook(Application.StartupPath + @"¥ExcelFile.xls");
nValue = xlsCreator1.Cell("A2").Long;    //nLong には 1000 が入ります
xlsCreator1.CloseBook(true);
```

注意事項

・Cell クラスに変数名を設定した場合、セル値の取得には対応しておりません。
・Excel ファイルが保持する整数値は 32bit 値 (Int32 型) であるため、32bit 値の範囲を超える値を設定する場合は、Double プロパティを使用してください。

Double プロパティ

実数を取得 /設定します。

書式

[VB.NET]
Public Property Double As Double

[C#]
Public Double double [get, set]

設定値

実数を設定します。

解説

設定したセルに計算式や関数式が設定されている場合は、その結果値を実数で取得する事ができます。

コーディング例

A1 セルに値を設定し、作成したファイルのセルの値を取得します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Double = 12.345     'セル A1 には 12.345 が入ります
XlsCreator1.CloseBook(True)

Dim nDouble As Double
XlsCreator1.ReadBook(Application.StartupPath & "¥ExcelFile.xls")
nDouble = XlsCreator1.Cell("A1").Double     'nDouble には 12.345 が入ります
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Double = 12.345;     //セル A1 には 12.345 が入ります
xlsCreator1.CloseBook(true);

Double nDouble;
xlsCreator1.ReadBook(Application.StartupPath + @"¥ExcelFile.xls");
nDouble = xlsCreator1.Cell("A1").Double;     //nDouble には 12.345 が入ります
xlsCreator1.CloseBook(true);
```

注意事項

・Cell クラスに変数名を設定した場合、セル値の取得には対応しておりません。

Func メソッド

関数の書き込みをします。結果を示す値も同時に設定できます。

書式

[VB.NET]
Public Sub Func(ByVal func As String, ByVal answer As object)

[C#]
public void Func(System.String func , System.object answer)

設定値

strFunc
　計算式や関数式を設定します。
objAns
　strFunc に対する結果値を設定します。

解説

引数 objAns の設定値は、計算式を示す引数 strFun の結果とは関係なく設定されます。
Func メソッドで計算結果をあらかじめ設定することで、ExcelCreator で作成した Excel ファイルを Excel で開き、修正を行わずに終了した時に表示される保存の確認メッセージを非表示にできます。

コーディング例

A2 セルに 1000 、A4 セルに計算式 (=A2+999)、C1 セルに計算式と計算結果 (=A2+999,1999) を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A2").Long = 1000                'A2 セルには 1000 が入ります
XlsCreator1.Cell("A4").Func("=A2+999", "")        'A4 セルには =A2+999 が入ります
XlsCreator1.Cell("C1").Func("=A2+999", "1999")    'C1 セルには =A2+999 が入ります
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A2").Long = 1000;               //A2 セルには 1000 が入ります
xlsCreator1.Cell("A4").Func("=A2+999", null);     //A4 セルには =A2+999 が入ります
xlsCreator1.Cell("C1").Func("=A2+999", "1999");   //C1 セルには =A2+999 が入ります
xlsCreator1.CloseBook(true);
```

AttrNo プロパティ

セルの属性番号を取得します。

書式

[VB.NET]
Public ReadOnly Property AttrNo As Integer

[C#]
Public System.Int32 AttrNo [get]

設定値

セルの属性番号を格納する変数を設定します。

解説

AttrNo プロパティを使用して、各セルで保持する罫線や表示形式等の書式情報を属性番号として取得し、取得した属性番号は Long2 / Str2 / Double2 / Value2 の各メソッドを使用して、データの差し込みと同時に取得した属性番号と同じ書式情報を設定することができます。

参照

Long2 メソッド、Str2 メソッド、Double2 メソッド、Value2 メソッド

コーディング例

セルの属性番号を取得し、Long2 / Str2 / Double2 / Value2 の各メソッドで別のセルに属性番号と値を設定します。

```
[VB.NET]
Dim nAttrNo As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nAttrNo = XlsCreator1.Cell("A1").AttrNo
XlsCreator1.Cell("B1").Long2(200, nAttrNo)
XlsCreator1.Cell("C1").Double2(200.1, nAttrNo)
XlsCreator1.Cell("D1").Str2("文字列", nAttrNo)
XlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo)
XlsCreator1.CloseBook(True)


[C#]
Int32 nAttrNo;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nAttrNo = xlsCreator1.Cell("A1").AttrNo;
xlsCreator1.Cell("B1").Long2(200, nAttrNo);
xlsCreator1.Cell("C1").Double2(200.1, nAttrNo);
xlsCreator1.Cell("D1").Str2("文字列", nAttrNo);
xlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo);
xlsCreator1.CloseBook(true);
```

注意事項

・Cell クラスに変数名を設定した場合は対応しておりません。

Value2 メソッド

数値や文字列、日付などの値と属性番号を設定します。

書式

[VB.NET]
Public Sub Value2(ByVal value As object, ByVal attrno As Integer)

[C#]
Public void Value2(System.object value , System.Int32 attrno)

設定値

objValue
　属性番号の値と共に設定する値を設定します。
nAttrNo
　AttrNo プロパティで取得した属性番号を設定します。

解説

AttrNo プロパティを使用して、各セルで保持する罫線や表示形式等の書式情報を属性番号として取得し、取得した属性番号は Value2 メソッドを使用して、データの差し込みと同時に取得した属性番号と同じ書式情報を設定することができます。

参照

AttrNo プロパティ

コーディング例

セルの属性番号を取得し、Long2 / Str2 / Double2 / Value2 の各 メソッドで属性番号と値を設定します。

```
[VB.NET]
Dim nAttrNo As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nAttrNo = XlsCreator1.Cell("A1").AttrNo
XlsCreator1.Cell("B1").Long2(200, nAttrNo)
XlsCreator1.Cell("C1").Double2(200.1, nAttrNo)
XlsCreator1.Cell("D1").Str2("文字列", nAttrNo)
XlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo)
XlsCreator1.CloseBook(True)


[C#]
Int32 nAttrNo;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nAttrNo = xlsCreator1.Cell("A1").AttrNo;
xlsCreator1.Cell("B1").Long2(200, nAttrNo);
xlsCreator1.Cell("C1").Double2(200.1, nAttrNo);
xlsCreator1.Cell("D1").Str2("文字列", nAttrNo);
xlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo);
xlsCreator1.CloseBook(true);
```

Str2 メソッド

文字列と属性番号を設定します。

書式

[VB.NET]
Public Sub Str2(ByVal value As String, ByVal attrno As Integer)

[C#]
Public void Str2(System.String value , System.Int32 attrno)

設定値

value
　属性番号の値と共に設定する文字列を設定します。
attrno
　AttrNo プロパティで取得した属性番号の値を設定します。

解説

AttrNo プロパティを使用して、各セルで保持する罫線や表示形式等の書式情報を属性番号として取得し、取得した属性番号は Str2 メソッドを使用して、データの差し込みと同時に取得した属性番号と同じ書式情報を設定することができます。

参照

AttrNo プロパティ

コーディング例

設定したセルの属性番号を取得し、Long2 / Str2 / Double2 / Value2 の各 メソッドで、その属性番号と値を設定します。

```
[VB.NET]
Dim nAttrNo As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nAttrNo = XlsCreator1.Cell("A1").AttrNo
XlsCreator1.Cell("B1").Long2(200, nAttrNo)
XlsCreator1.Cell("C1").Double2(200.1, nAttrNo)
XlsCreator1.Cell("D1").Str2("文字列", nAttrNo)
XlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo)
XlsCreator1.CloseBook(True)


[C#]
Int32 nAttrNo;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nAttrNo = xlsCreator1.Cell("A1").AttrNo;
xlsCreator1.Cell("B1").Long2(200, nAttrNo);
xlsCreator1.Cell("C1").Double2(200.1, nAttrNo);
xlsCreator1.Cell("D1").Str2("文字列", nAttrNo);
xlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo);
xlsCreator1.CloseBook(true);
```

Long2 メソッド

整数と属性番号を設定します。

書式

[VB.NET]
Public Sub Long2(ByVal value As , ByVal attrno As Integer)

[C#]
Public void Long2(System.Int32 value , System.Int32 attrno)

設定値

value
　属性番号の値と共に設定する整数値を設定します。
attrno
　AttrNo プロパティで取得した属性番号の値を設定します。

解説

AttrNo プロパティを使用して、各セルで保持する罫線や表示形式等の書式情報を属性番号として取得し、取得した属性番号は Long2 メソッドを使用して、データの差し込みと同時に取得した属性番号と同じ書式情報を設定することができます。

参照

AttrNo プロパティ

コーディング例

設定したセルの属性番号を取得し、Long2 / Str2 / Double2 / Value2 の各 メソッドで、その属性番号と値を設定します。

```
[VB.NET]
Dim nAttrNo As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nAttrNo = XlsCreator1.Cell("A1").AttrNo
XlsCreator1.Cell("B1").Long2(200, nAttrNo)
XlsCreator1.Cell("C1").Double2(200.1, nAttrNo)
XlsCreator1.Cell("D1").Str2("文字列", nAttrNo)
XlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo)
XlsCreator1.CloseBook(True)


[C#]
Int32 nAttrNo;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nAttrNo = xlsCreator1.Cell("A1").AttrNo;
xlsCreator1.Cell("B1").Long2(200, nAttrNo);
xlsCreator1.Cell("C1").Double2(200.1, nAttrNo);
xlsCreator1.Cell("D1").Str2("文字列", nAttrNo);
xlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo);
xlsCreator1.CloseBook(true);
```

注意事項

・Excel ファイルが保持する整数値は 32bit 値（Int32 型）であるため、32bit 値の範囲を超える値を設定する場合は、Double2 メソッドを使用してください。

Double2 メソッド

実数と属性番号を設定します。

書式

[VB.NET]
Public Sub Double2(ByVal value As Double, ByVal attrno As Integer)

[C#]
Public void Double2(System.Double value , System.Int32 attrno)

設定値

value
　属性番号の値と共に設定する実数値を設定します。
attrno
　AttrNo プロパティで取得した属性番号の値を設定します。

解説

AttrNo プロパティを使用して、各セルで保持する罫線や表示形式等の書式情報を属性番号として取得し、取得した属性番号は Double2 メソッドを使用して、データの差し込みと同時に取得した属性番号と同じ書式情報を設定することができます。

参照

AttrNo プロパティ

コーディング例

設定したセルの属性番号を取得し、Long2 / Str2 / Double2 / Value2 の各 メソッドで、その属性番号と値を設定します。

```
[VB.NET]
Dim nAttrNo As Integer
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", "")
nAttrNo = XlsCreator1.Cell("A1").AttrNo
XlsCreator1.Cell("B1").Long2(200, nAttrNo)
XlsCreator1.Cell("C1").Double2(200.1, nAttrNo)
XlsCreator1.Cell("D1").Str2("文字列", nAttrNo)
XlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo)
XlsCreator1.CloseBook(True)


[C#]
Int32 nAttrNo;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", "");
nAttrNo = xlsCreator1.Cell("A1").AttrNo;
xlsCreator1.Cell("B1").Long2(200, nAttrNo);
xlsCreator1.Cell("C1").Double2(200.1, nAttrNo);
xlsCreator1.Cell("D1").Str2("文字列", nAttrNo);
xlsCreator1.Cell("E1").Value2("バリアント型", nAttrNo);
xlsCreator1.CloseBook(true);
```

Copy メソッド

セルをコピーします。

書式

[VB.NET]
Public Sub Copy()
Public Sub Copy(ByVal cell As String)

[C#]
public void Copy()
Public void Copy(System.String cell)

設定値

strCell
コピー先のセルを設定します。

解説

コピーするセルは、単一セル、セル範囲指定に対応しています。
引数を省略した場合、セルがメモリ上にコピーされ、Paste メソッドで設定したセルに貼り付けることができます。
Pos クラスでセル範囲を設定する場合、終了座標≧開始座標と設定してください。

参照

Paste メソッド

コーディング例

A1:B1 範囲セルを A2 セルにコピーし、A1 セルを C2 セルにコピーします。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn03
XlsCreator1.Cell("A1:B1").Str = "セルコピー元"
XlsCreator1.Cell("A1:B1").Copy("A2")
XlsCreator1.Cell("A1").Copy("C2")
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn03;
xlsCreator1.Cell("A1:B1").Str = "セルコピー元";
xlsCreator1.Cell("A1:B1").Copy("A2");
xlsCreator1.Cell("A1").Copy("C2");
xlsCreator1.CloseBook(true);
```

注意事項

・コピー元セルのセル属性と値のみコピーします。
・Copy メソッドの Cell クラスに変数名を設定した場合、セルはコピーされません。
・Excel ファイルのブック間、シート間のコピー・貼り付けには対応しておりません。

Paste メソッド

Copy メソッドでコピーしたセルを貼り付けます。

書式

[VB.NET]
Public Sub Paste()

[C#]
Public void Paste()

参照

Copy メソッド

コーディング例

A1 セルのコピーを A3 セルに貼り付けます。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Attr.Pattern = ExcelCreator.xlPattern.pn03
XlsCreator1.Cell("A1").Str = "セルコピー元"
XlsCreator1.Cell("A1").Copy("")
XlsCreator1.Cell("A3").Paste()
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Attr.Pattern = ExcelCreator.xlPattern.pn03;
xlsCreator1.Cell("A1").Str = "セルコピー元";
xlsCreator1.Cell("A1").Copy("");
xlsCreator1.Cell("A3").Paste();
xlsCreator1.CloseBook(true);
```

注意事項

・コピー元セルのセル属性と値のみコピーします。
・Paste メソッドの Cell クラスに変数名を設定した場合、Copy メソッドでコピーしたセルは貼り付けられません。
・Excel ファイルのブック間、シート間のコピー・貼り付けには対応しておりません。

Clear メソッド

セルをクリア（初期化）します。

書式

[VB.NET]
Public Sub Clear()

[C#]
Public void Clear()

コーディング例

A1:B1 セル、A3 セルをクリアします。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn02
XlsCreator1.Cell("A1:B1").Clear()
XlsCreator1.Cell("A3:B3").Attr.Pattern = ExcelCreator.xlPattern.pn03
XlsCreator1.Cell("A3").Clear()
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1:B1").Attr.Pattern = ExcelCreator.xlPattern.pn02;
xlsCreator1.Cell("A1:B1").Clear();
xlsCreator1.Cell("A3:B3").Attr.Pattern = ExcelCreator.xlPattern.pn03;
xlsCreator1.Cell("A3").Clear();
xlsCreator1.CloseBook(true);
```

注意事項

・Clear メソッドは Cell クラスに変数名を設定した場合は対応しておりません。

Break プロパティ

改ページの設定 / 解除を設定します。

書式

[VB.NET]
Public WriteOnly Property Break As Boolean

[C#]
Public bool Break [set]

設定値

定数	内容
True	改ページを設定します。
False	改ページを解除します。

解説

行に対する改ページ、列に対する改ページ、行列に対する改ページを設定できます。
行方向に 1026 個まで、列方向に 256 個まで設定できます。

コーディング例

A35 セルに改ページを設定設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A35").Break = True
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A35").Break = true;
xlsCreator1.CloseBook(true);
```

ColWidth プロパティ

セルの列幅を設定します。

書式

[VB.NET]
Public WriteOnly Property ColWidth As Integer

[C#]
Public System.Int32 ColWidth [set]

解説

列幅を半角文字数（0 ～ 255 文字）で設定します。
0 を設定した場合は非表示列となります。

参照

RowHeight プロパティ

コーディング例

A ～ G 列の列幅を 20 文字、10 行目の行高を 20 ポイントに設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1:G1").ColWidth = 20
XlsCreator1.Cell("G10").RowHeight = 20
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1:G1").ColWidth = 20;
xlsCreator1.Cell("G10").RowHeight = 20;
xlsCreator1.CloseBook(true);
```

解説

・ColWidth2 プロパティで取得した値を ColWidth プロパティに設定しないでください。

RowHeight プロパティ

セルの高さを設定します。

書式

[VB.NET]
Public WriteOnly Property RowHeight As Double

[C#]
Public double RowHeight [set]

解説

高さをポイント（0 ～ 409）で設定します。
0 を設定した場合は非表示行となります。

参照

ColWidth プロパティ

コーディング例

A ～ G 列の列幅を 20 文字、10 行目の行高を 20 ポイントに設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1:G1").ColWidth = 20
XlsCreator1.Cell("G10").RowHeight = 20
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1:G1").ColWidth = 20;
xlsCreator1.Cell("G10").RowHeight = 20;
xlsCreator1.CloseBook(true);
```

解説

・RowHeight2 プロパティで取得した値を RowHeight プロパティに設定しないでください。

ColWidth2 プロパティ

セルの幅を取得 / 設定します。

書式

[VB.NET]
Public Property ColWidth2() As Integer

[C#]
public int ColWidth2 [get, set]

解説

ColWidth2 プロパティは、特定のセルの幅を別のセル幅に合わせる場合に使用します。
Excel 上では、セル幅は [文字数 × 256 + 余白(1/256 文字単位)] で管理され、ColWidth2 プロパティは、この値をマイナスの値として取得します。ColWidth2 プロパティで取得した値を他のセルに設定することにより、セル幅を合わせることができます。

参照

RowHeight2 プロパティ

コーディング例

C 列の列幅を B 列の列幅に合わせます。

[VB.NET]
```
Dim nCellColWidth2 As Inetger
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
nCellColWidth2 = XlsCreator1.Cell("B").ColWidth2
XlsCreator1.Cell("C").ColWidth2 = nCellColWidth2
XlsCreator1.CloseBook(True)
```

[C#]
```
int nCellColWidth2;
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
nCellColWidth2 = xlsCreator1.Cell("B").ColWidth2;
xlsCreator1.Cell("C").ColWidth2 = nCellColWidth2;
xlsCreator1.CloseBook(true);
```

解説

・下記の様に、ColWidth2 プロパティの設定値に直接 ColWidtht2 プロパティを設定することはできません。ColWidth2 プロパティの取得値を変数に格納し、その変数を設定してください。

誤)
```
XlsCreator1.Cell("A").ColWidth2 = XlsCreator1.Cell("B").ColWidth2
```

正)
```
Dim nWidth As Inetger
nWidth = XlsCreator1.Cell("B").ColWidth2
XlsCreator1.Cell("A").ColWidth2 = nWidth
```

・ColWidth2 プロパティで取得した値を ColWidth プロパティに設定しないでください。ColWidth プロパティは文字数で幅を設定しますが、ColWidth2 プロパティは Excel が保持する論理値を直接扱います。
・結合したセルの列幅を取得した場合、セル全体の列幅ではなく、結合したセルの先頭セルの列幅を取得します。

RowHeight2 プロパティ

セルの高さを取得 / 設定します。

書式

[VB.NET]
Public Property RowHeight2() As Double

[C#]
public double RowHeight2 [get, set]

解説

高さをポイント（0 ～ 409）で設定します。
0 を設定した場合は非表示行となります。
取得した値を他の行に設定することで、セルの高さを合わせることができます。

参照

ColWidth2 プロパティ

コーディング例

3 行目の行高を 2 行目の行高に合わせます。

```
[VB.NET]
Dim dblCellRowHeight2 As Double
XlsCreator1.OpenBook(Application.StartupPath & "¥ExcelFile.xls", Application.StartupPath &
"¥ExcelTemplate.xls")
dblCellRowHeight2 = XlsCreator1.Cell("A2").RowHeight2
XlsCreator1.Cell("A3").RowHeight2 = dblCellRowHeight2
XlsCreator1.CloseBook(True)


[C#]
double dblCellRowHeight2;
xlsCreator1.OpenBook(Application.StartupPath + @"¥ExcelFile.xls", Application.StartupPath +
@"¥ExcelTemplate.xls");
dblCellRowHeight2 = xlsCreator1.Cell("A2").RowHeight2;
xlsCreator1.Cell("A3").RowHeight2 = dblCellRowHeight2;
xlsCreator1.CloseBook(true);
```

解説

・下記の様に、RowHeight2 プロパティの設定値に直接 RowHeight2 プロパティを設定することはできません。
RowHeight2 プロパティの取得値を変数に格納し、その変数を設定してください。

```
誤）
XlsCreator1.Cell("A3").RowHeight2 = XlsCreator1.Cell("A2").RowHeight2

正）
Dim dblHeight As Double
dblHeight = XlsCreator1.Cell("A2").RowHeight2
XlsCreator1.Cell("A3").RowHeight2 = dblHeight
```

・RowHeight2 プロパティに設定する値は、RowHeight2 プロパティで取得した値を設定してください。

Cell/Pos.Attr クラス

セルの属性を設定します。

書式

[VB.NET]
Public Cell.Attr
Public Pos.Attr

[C#]
public Cell.Attr
public Pos.Attr

解説

このクラスの全てのメンバの一覧については、以下の Cell/Pos.Attr メンバを参照してください。

パブリックプロパティ

Format	表示書式を設定します。	P120
PosHorz	文字の配置（横位置）を設定します。	P124
FontName	フォント名を設定します。	P130
FontPoint	フォントサイズを設定します。	P131
FontColor	フォント色を設定します。	P132
FontStyle	フォントスタイルを設定します。	P133
FontULine	フォントの下線スタイルを設定します。	P134
LineTop	上罫線を設定します。	P135
LineBottom	下罫線を設定します。	P135
LineLeft	左罫線を設定します。	P135
LineRight	右罫線を設定します。	P135
LineLeftUp	左上罫線を設定します。	P135
LineRightUp	右上罫線を設定します。	P135
Pattern	パターンを設定します。	P139
BackColor	背景色を設定します。	P140
PosVert	文字の配置（縦位置）を設定します。	P125
PosTurn	文字方向（角度）を設定します。	P126
OverReturn	「折り返して全体を表示」を設定します。	P127
Fit	「縮小して全体を表示する」を設定します。	P128
Joint	指定範囲のセルに「セルを結合する」を設定します。	P129

パブリックメソッド

Box	設定したセル領域を囲む Box 罫線を設定します。	P135

Format プロパティ

表示形式を設定します。

書式

[VB.NET]
Public WriteOnly Property Format As String

[C#]
Public string Format [set]

設定値

表示書式を設定します。表示形式は、Microsoft 提供の「Excel ヘルプ」を参照してください。

コーディング例

セルに様々な表示形式を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)

XlsCreator1.Cell("A1").Attr.Format = "[色 5]#.#"          '色を付けて表示
XlsCreator1.Cell("A1").Long = 100
XlsCreator1.Cell("A2").Attr.Format = "###,###,###,###"  'カンマ区切り
XlsCreator1.Cell("A2").Long = 1000000000
XlsCreator1.Cell("A3").Attr.Format = "¥""¥¥¥""000.00"   '¥マークを付ける
XlsCreator1.Cell("A3").Long = 1000000
XlsCreator1.Cell("A4").Attr.Format = ";[赤]-0.0"         '値がマイナスの時に赤色にする
XlsCreator1.Cell("A4").Long = -123456
XlsCreator1.Cell("A5").Attr.Format = ";;""ZERO"""        '値が 0 の時に、ZERO と表示する
XlsCreator1.Cell("A5").Long = 0

XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);

//各指定セルの表示書式を設定します
xlsCreator1.Cell("A1").Attr.Format = "[色 5]#.#";          //色を付けて表示
xlsCreator1.Cell("A1").Long = 100;
xlsCreator1.Cell("A2").Attr.Format = "###,###,###,###";  //カンマ区切り
xlsCreator1.Cell("A2").Long = 1000000000;
xlsCreator1.Cell("A3").Attr.Format = "¥¥000.00";          //¥¥マークを付ける
xlsCreator1.Cell("A3").Long = 1000000;
xlsCreator1.Cell("A4").Attr.Format = ";[赤]-0.0";         //値がマイナスの時に赤色にする
xlsCreator1.Cell("A4").Long = -123456;
xlsCreator1.Cell("A5").Attr.Format = ";;¥¥ZERO¥¥";       //値が 0 の時に ZERO と表示する
xlsCreator1.Cell("A5").Long = 0;

xlsCreator1.CloseBook(true);
```

PosHorz プロパティ

文字の配置（横位置）を設定します。

書式

[VB.NET]
Public WriteOnly Property PosHorz As ExcelCreator.xlPosHorz

[C#]
Public ExcelCreator.xlPosHorz PosHorz [set]

設定値

xlPosHorz
文字の水平位置を示す xlPosHorz 列挙体のメンバで設定します。

定数	値	内容
	1～15	左詰めインデント値
phNormal	20	標準（デフォルト値）
phLeft	21	左詰め
phCenter	22	中央揃え
phRight	23	右詰め
phLoop	24	繰り返し
phBothEven	25	両端揃え
phSelCenter	26	選択範囲内で中央
phEven	27	均等割付

解説

C# で左詰めインデントとして直接数値を設定するには、(ExcelCreator.xlPosTurn. xlPosHorz)10 の様にキャストして設定します。

コーディング例

D7 ～ E8 セルの配置（横位置）を「中央揃え」を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("D7").Str = "abcde"
XlsCreator1.Cell("D7:E8").Attr.PosHorz = ExcelCreator.xlPosHorz.phCenter
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("D7").Str = "abcde";
xlsCreator1.Cell("D7:E8").Attr.PosHorz = ExcelCreator.xlPosHorz.phCenter;
xlsCreator1.CloseBook(true);
```

PosVert プロパティ

文字の配置（縦位置）を設定します。

書式

[VB.NET]
Public WriteOnly Property PosVert As ExcelCreator.xlPosVert

[C#]
Public ExcelCreator.xlPosVert PosVert [set]

設定値

xlPosVert
文字の垂直位置を示す xlPosVert 列挙体のメンバで設定します。

定数	値	内容
pvUp	0	上詰め
pvCenter	1	中央揃え
pvDown	2	下詰め（デフォルト値）
pvBothEven	3	両端揃え
pvEven	4	均等割付

コーディング例

D7 セルの配置（縦位置）を「中央揃え」に設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("D7").Str = "abcde"
XlsCreator1.Cell("D7").Attr.PosVert = ExcelCreator.xlPosVert.pvCenter
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("D7").Str = "abcde";
xlsCreator1.Cell("D7").Attr.PosVert = ExcelCreator.xlPosVert.pvCenter;
xlsCreator1.CloseBook(true);
```

PosTurn プロパティ

文字方向（角度）を設定します。

書式

[VB.NET]
Public WriteOnly Property PosTurn As ExcelCreator.xlPosTurn

[C#]
Public ExcelCreator.xlPosTurn PosTurn [set]

設定値

xlPosTurn
文字列の回転角度を示す xlPosTurn 列挙体のメンバで設定します。

定数	値	内容
	-90～90	設定した角度
ptHorz	100	横（デフォルト値）
ptVert	101	縦
ptRvTurn90	102	半時計回りに 90 度
ptTurn90	103	時計回りに 90 度

解説

C# で設定した角度として直接数値を設定するには、(ExcelCreator.xlPosTurn.xlPosTurn)90 や (ExcelCreator.xlPosTurn.xlPosTurn)(-90) の様にキャストして設定します。

コーディング例

D7 セルの文字方向（角度）を「時計回りに 90 度回転」に設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("D7").Str = "abcde"
XlsCreator1.Cell("D7").Attr.PosTurn = ExcelCreator.xlPosTurn.ptTurn90
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("D7").Str = "abcde";
xlsCreator1.Cell("D7").Attr.PosTurn = ExcelCreator.xlPosTurn.ptTurn90;
xlsCreator1.CloseBook(true);
```

OverReturn プロパティ

「折り返して全体を表示する」を設定します。

書式

[VB.NET]
Public WriteOnly Property OverReturn As Boolean

[C#]
Public bool OverReturn [set]

設定値

値	内容
True	「折り返して全体を表示」を設定します。
False	「折り返して全体を表示」を設定しない。（デフォルト値）

コーディング例

D7 セルに「折り返して全体を表示」を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("D7").Str = "abcde"
XlsCreator1.Cell("D7").Attr.OverReturn = True
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("D7").Str = "abcde";
xlsCreator1.Cell("D7").Attr.OverReturn = true;
xlsCreator1.CloseBook(true);
```

Fit プロパティ

「縮小して全体を表示する」を設定します。

書式

[VB.NET]
Public WriteOnly Property Fit As Boolean

[C#]
Public bool Fit [set]

設定値

値	内容
True	縮小して全体を表示する。
False	縮小して全体を表示しない。（デフォルト値）

解説

「縮小して全体を表示する」を設定した場合、自動的にフォントサイズは変更されますが、FontPoint プロパティで設定したフォントサイズは変更されません。

コーディング例

D7 セルに「縮小して全体を表示する」を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("D7").Str = "abcde"
XlsCreator1.Cell("D7").Attr.Fit = True
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("D7").Str = "abcde";
xlsCreator1.Cell("D7").Attr.Fit = true;
xlsCreator1.CloseBook(true);
```

Joint プロパティ

指定範囲のセルに「セルを結合する」を設定します。

書式

[VB.NET]
```
Public WriteOnly Property Joint As Boolean
```

[C#]
```
Public bool Joint [set]
```

設定値

値	内容
True	セルを結合する。
False	セルを結合しない。（デフォルト値）

解説

結合したセルに書式の属性やデータの設定を行うには、結合セル全体を設定してください。
例:
```
Cell("A1:C2").Attr.Joint = True
Cell("A1:C2").Attr.FontName = "MS 明朝"
```

コーディング例

A10 ～ D10 セルを結合します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A10:D10").Str = "abcde"
XlsCreator1.Cell("A10:D10").Attr.Joint = True
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A10:D10").Str = "abcde";
xlsCreator1.Cell("A10:D10").Attr.Joint = true;
xlsCreator1.CloseBook(true);
```

注意事項

オーバーレイ元ファイルで結合されているセルや、プログラム中で結合したセルを解除することはできません。

FontName プロパティ

フォント名を設定します。

書式

[VB.NET]
Public WriteOnly Property FontName As String

[C#]
Public string FontName [set]

コーディング例

A1 セルに「MS P ゴシック」、フォントサイズを「13 ポイント」、フォント色を「赤色」、「太字」「二重下線」を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック"
XlsCreator1.Cell("A1").Attr.FontPoint = 12
XlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed
XlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold
XlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック";
xlsCreator1.Cell("A1").Attr.FontPoint = 12;
xlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed;
xlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold;
xlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble;
xlsCreator1.CloseBook(true);
```

FontPoint プロパティ

フォントサイズを設定します。

書式

[VB.NET]
Public WriteOnly Property FontPoint As Double

[C#]
Public double FontPoint [set]

設定値

フォントサイズを示すポイント（1 ～ 409 ポイント）を設定します。

コーディング例

A1 セルに「MS P ゴシック」、フォントサイズを「13 ポイント」、フォント色を「赤色」、「太字」「二重下線」を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック"
XlsCreator1.Cell("A1").Attr.FontPoint = 12
XlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed
XlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold
XlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック";
xlsCreator1.Cell("A1").Attr.FontPoint = 12;
xlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed;
xlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold;
xlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble;
xlsCreator1.CloseBook(true);
```

FontColor プロパティ

フォント色を設定します。

書式

[VB.NET]
Public WriteOnly Property FontColor As ExcelCreator.xlColor

[C#]
Public ExcelCreator.xlColor FontColor [set]

設定値

xlColor
フォントの色を示す xlColor 列挙体のメンバ、または、色を示す定数で設定します。

定数	値	内容
xcDefault	0	自動 （デフォルト値）
xcBlack	8	黒
xcWhite	9	白
xcRed	10	赤
xcGreen	11	明るい緑
xcBlue	12	青
xcYellow	13	黄色
xcPink	14	ピンク
xcCyan	15	水色

色を示す定数は ExcelCreator 5.0 for .NET ヘルプ [補足]-[色を示す定数] を参照してください。

コーディング例

A1 セルに「MS P ゴシック」、フォントサイズを「13 ポイント」、フォント色を「赤色」、「太字」「二重下線」を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック"
XlsCreator1.Cell("A1").Attr.FontPoint = 12
XlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed
XlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold
XlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble
XlsCreator1.CloseBook(True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック";
xlsCreator1.Cell("A1").Attr.FontPoint = 12;
xlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed;
xlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold;
xlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble;
xlsCreator1.CloseBook(true);
```

FontStyle プロパティ

フォントスタイルを設定します。

書式

[VB.NET]
Public WriteOnly Property FontStyle As ExcelCreator.xlFontStyle

[C#]
Public ExcelCreator.xlFontStyle FontStyle [set]

設定値

xlFontStyle
xlFontStyle 列挙体のメンバで設定します。

スタイル

定数	値	内容
xsNormal	&H00	標準 （デフォルト値）
xsBold	&H01	太字
xsItalic	&H02	斜体
xsBold + xsItalic	&H03	太字 + 斜体

文字飾り1

定数	値	内容
xsStrike	&H04	取消し線

文字飾り2

定数	値	内容
xsUp	&H08	上付き
xsDown	&H10	下付き

コーディング例

A1 セルに「MS P ゴシック」、フォントサイズを「13 ポイント」、フォント色を「赤色」、「太字」「二重下線」を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック"
XlsCreator1.Cell("A1").Attr.FontPoint = 12
XlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed
XlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold
XlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック";
xlsCreator1.Cell("A1").Attr.FontPoint = 12;
xlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed;
xlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold;
xlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble;
xlsCreator1.CloseBook(true);
```

FontULine プロパティ

セルのフォント下線を設定します。

書式

[VB.NET]
Public WriteOnly Property FontULine As ExcelCreator.xlFontULine

[C#]
Public ExcelCreator.xlFontULine FontULine [set]

設定値

xlFontULine
フォントの下線スタイルを示す xlFontULine 列挙体のメンバで設定します。

スタイル

定数	値	内容
fuNone	0	下線なし(デフォルト値)
fuNormal	1	一重下線
fuDouble	2	二重下線

コーディング例

A1 セルに「MS P ゴシック」、フォントサイズを「13 ポイント」、フォント色を「赤色」、「太字」「二重下線」を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック"
XlsCreator1.Cell("A1").Attr.FontPoint = 12
XlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed
XlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold
XlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A1").Attr.FontName = "ＭＳ Ｐゴシック";
xlsCreator1.Cell("A1").Attr.FontPoint = 12;
xlsCreator1.Cell("A1").Attr.FontColor = ExcelCreator.xlColor.xcRed;
xlsCreator1.Cell("A1").Attr.FontStyle = ExcelCreator.xlFontStyle.xsBold;
xlsCreator1.Cell("A1").Attr.FontULine = ExcelCreator.xlFontULine.fuDouble;
xlsCreator1.CloseBook(true);
```

Box メソッド

設定したセル領域を囲む Box 罫線を設定します。

書式

[VB.NET]
Public Sub Box(ByVal nType As ExcelCreator.xlBoxType, ByVal nStyle As ExcelCreator.xlLineStyle)

[C#]
public void Box(ExcelCreator.xlBoxType nType , ExcelCreator.xlLineStyle nStyle)

設定値

nType
xlBoxType 列挙体のメンバで設定します。

定数	内容
btBox	箱線（デフォルト値）
btLtc	格子線
btOver	上横線
btUnder	下横線
btLeft	左縦線
btRight	右縦線

nStyle
xlLineStyle 列挙体のメンバで設定します。

定数	内容
lsNone	罫線なし(デフォルト値)
lsNormal	実線
lsThick	太線
lsBroken	破線
lsDot	点線
lsThick2	極太線
lsDouble	二重線
lsSlender	細実線
lsMidBroken	中破線
lsSlnChain1	細一点鎖線
lsMidChain1	中一点鎖線
lsSlnChain2	細二点鎖線
lsMidChain2	中二点鎖線
lsMidChains	中一点斜鎖線

参照

LineTop プロパティ、LineBottom プロパティ、LineLeft プロパティ、LineRight プロパティ、LineLeftUp プロパティ、LineRightUp プロパティ

コーディング例

セル A10 から C20 に Box 罫線（極太線）の書き込みをします。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("A20:C25").Attr.Box(ExcelCreator.xlBoxType.btBox, ExcelCreator.xlLineStyle.lsThick2)
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("A20:C25").Attr.Box(ExcelCreator.xlBoxType.btBox, ExcelCreator.xlLineStyle.lsThick2);
xlsCreator1.CloseBook(true);
```

LineTop 、LineBottom 、LineRight 、LineLeft 、LineRightUp 、LineLeftUp プロパティ

上、下、左、右、左上がり斜線、右上がり斜線の各罫線を設定します。

書式

[VB.NET]
```
Public WriteOnly Property LineTop As ExcelCreator.xlLineStyle
Public WriteOnly Property LineBottom As ExcelCreator.xlLineStyle
Public WriteOnly Property LineRight As ExcelCreator.xlLineStyle
Public WriteOnly Property LineLeft As ExcelCreator.xlLineStyle
Public WriteOnly Property LineRightUp As ExcelCreator.xlLineStyle
Public WriteOnly Property LineLeftUp As ExcelCreator.xlLineStyle
```

[C#]
```
Public ExcelCreator.xlLineStyle LineTop [set]
Public ExcelCreator.xlLineStyle LineBottom [set]
Public ExcelCreator.xlLineStyle LineRight [set]
Public ExcelCreator.xlLineStyle LineLeft [set]
Public ExcelCreator.xlLineStyle LineRightUp [set]
Public ExcelCreator.xlLineStyle LineLeftUp [set]
```

設定値

xlLineStyle
xlLineStyle 列挙体のメンバで設定します。

定数	内容
lsNone	罫線なし(デフォルト値)
lsNormal	実線
lsThick	太線
lsBroken	破線
lsDot	点線
lsThick2	極太線
lsDouble	二重線
lsSlender	細実線
lsMidBroken	中破線
lsSlnChain1	細一点鎖線
lsMidChain1	中一点鎖線
lsSlnChain2	細二点鎖線
lsMidChain2	中二点鎖線
lsMidChains	中一点斜鎖線

解説

線種と色の両方を設定する場合は、「 + 」演算子で結合して設定します。
C# では、(ExcelCreator.xlLineStyle)((Int32)ExcelCreator.xlLineStyle.lsThick + (Int32)ExcelCreator.xlColor.xcRed)
の様に、列挙体の定数をそれぞれ Int32 型にキャストし、かつ、全体を列挙体にキャストして設定します。
引数 xlLineStyle に色を設定する場合の設定値は、色を示す定数を参照してください。
色を示す定数は ExcelCreator 5.0 for .NET ヘルプ [補足]-[色を示す定数] を参照してください。

参照

Box メソッド

コーディング例

セルに各罫線を設定します。

[VB.NET]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("C2:F2").Attr.LineTop = ExcelCreator.xlLineStyle.lsNormal      '実線
XlsCreator1.Cell("C5:F5").Attr.LineBottom = ExcelCreator.xlLineStyle.lsBroken  '破線
XlsCreator1.Cell("C2:C5").Attr.LineLeft = ExcelCreator.xlLineStyle.lsThick + ExcelCreator.xlColor.xcRed  '
赤太線
XlsCreator1.Cell("F2:F5").Attr.LineRight = ExcelCreator.xlLineStyle.lsDot + ExcelCreator.xlColor.xcGreen '
緑点線
XlsCreator1.CloseBook(True)
```

```
[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("C2:F2").Attr.LineTop = ExcelCreator.xlLineStyle.lsNormal;      //実線
xlsCreator1.Cell("C5:F5").Attr.LineBottom = ExcelCreator.xlLineStyle.lsBroken;  //破線
xlsCreator1.Cell("C2:C5").Attr.LineLeft = (ExcelCreator.xlLineStyle)((int)ExcelCreator.xlLineStyle.lsThick +
(int)ExcelCreator.xlColor.xcRed);  //赤太線
xlsCreator1.Cell("F2:F5").Attr.LineRight = (ExcelCreator.xlLineStyle)((int)ExcelCreator.xlLineStyle.lsDot +
(int)ExcelCreator.xlColor.xcGreen); //緑点線
xlsCreator1.CloseBook(true);
```

注意事項

LineLeftUp プロパティ、LineRightUp プロパティを同じセルに設定した場合、クロスした罫線になりますが、Excel の制限により両方の線種や色を別々に設定出来ないため、LineRightUp プロパティで設定した線種や色の値が有効になります。

Pattern プロパティ

パターンを設定します。

書式

[VB.NET]
Public WriteOnly Property Pattern As ExcelCreator.xlPattern

[C#]
Public ExcelCreator.xlPattern Pattern [set]

設定値

xlPattern
xlPattern 列挙体のメンバで設定します。

定数	内容	定数	内容
pn01	塗りつぶし(デフォルト値)	pn10	極太線左下がり斜線格子
pn02	50%灰色	pn11	実線 横縞
pn03	75%灰色	pn12	実線 縦縞
pn04	25%灰色	pn13	実線 右下がり斜線 縞
pn05	横縞	pn14	実線 左下がり斜線 縞
pn06	縦縞	pn15	実線 横 格子
pn07	右下がり斜線 縞	pn16	実線 左下がり斜線 格子
pn08	左下がり斜線 縞	pn17	12.5%灰色
pn09	左下がり斜線 格子	pn18	6.25%灰色

解説

パターンと色の両方を設定する場合は、「 + 」演算子で結合して設定します。
引数 xlPattern に色を設定する場合の設定値は、セルに対する色設定を参照してください。

コーディング例

B2 ～ D3 セルのパターン（左下がり斜線 縞）と背景色（青）を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("B2:D3").Attr.Pattern = ExcelCreator.xlPattern.pn08
XlsCreator1.Cell("B2:D3").Attr.BackColor = ExcelCreator.xlColor.xcBlue
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("B2:D3").Attr.Pattern = ExcelCreator.xlPattern.pn08;
xlsCreator1.Cell("B2:D3").Attr.BackColor = ExcelCreator.xlColor.xcBlue;
xlsCreator1.CloseBook(true);
```

BackColor プロパティ

背景色を設定します。

書式

[VB.NET]
Public WriteOnly Property BackColor As ExcelCreator.xlColor

[C#]
Public ExcelCreator.xlColor BackColor [set]

設定値

xlColor
セルの背景色を示す xlColor 列挙体のメンバ、または、色を示す定数で設定します。

定数	値	内容
xcDefault	0	自動(デフォルト値)
xcBlack	8	黒
xcWhite	9	白
xcRed	10	赤
xcGreen	11	明るい緑
xcBlue	12	青
xcYellow	13	黄色
xcPink	14	ピンク
xcCyan	15	水色

色を示す定数は ExcelCreator 5.0 for .NET ヘルプ [補足]-[色を示す定数] を参照してください。

コーディング例

B2 ～ D3 セルにパターン、色を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003)
XlsCreator1.Cell("B2:D3").Attr.BackColor = ExcelCreator.xlColor.xcBlue 'セルの背景色：青色
XlsCreator1.CloseBook(True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3, ExcelCreator.xlVersion.ver2003);
xlsCreator1.Cell("B2:D3").Attr.BackColor = ExcelCreator.xlColor.xcBlue; //セルの背景色：青色
xlsCreator1.CloseBook(true);
```

PDF クラス

出力する PDF ファイルの各種設定を行います。

書式

[VB.NET]
Public ReadOnly Property PDF()

[C#]
public PDF [get]

解説

このクラスの全てのメンバの一覧については、以下の PDF メンバを参照してください。

パブリックプロパティ

Title	文書プロパティの概要［タイトル］を設定します。	P142
Subject	文書プロパティの概要［サブタイトル］を設定します。	P143
Producer	文書プロパティの概要［PDF 変換］を設定します。	P144
Creator	文書プロパティの概要［アプリケーション］を設定します。	P145
Author	文書プロパティの概要［作成者］を設定します。	P146
EmbedFonts	埋め込むフォントを設定します。	P147
ImageType	イメージの種類を設定します。	P148
ProgressDialog	処理経過を表示するプログレスダイアログの表示 / 非表示を設定します。	P149

Title プロパティ

PDF ファイルの文書プロパティの概要［タイトル］を設定します。

書式

[VB.NET]
Public WriteOnly Property PDF.Title As String

[C#]
Public string PDF.Title [set]

解説

デフォルトは「ExcelCreator 5.0 for .NET」です。

コーディング例

PDF ファイルの文書プロパティの概要［タイトル］を設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.Title = "2005 年度売り上げ一覧"
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PDF.Title = "2005 年度売り上げ一覧";
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

Subject プロパティ

PDF ファイルの文書プロパティの概要［サブタイトル］を設定します。

書式

[VB.NET]
Public WriteOnly Property PDF.Subject As String

[C#]
Public string PDF.Subject [set]

解説

デフォルトは「ExcelCreator 5.0 for .NET」です。

コーディング例

PDF ファイルの文書プロパティの概要［サブタイトル］を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.Subject = "第一営業部"
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PDF.Subject = "第一営業部";
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

Producer プロパティ

PDF ファイルの文書プロパティの概要［PDF 変換］を設定します。

書式

[VB.NET]
Public WriteOnly Property PDF.Producer As String

[C#]
Public string PDF.Producer [set]

解説

デフォルトは「ExcelCreator 5.0 for .NET」です。

コーディング例

PDF ファイルの文書プロパティの概要［PDF 変換］を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.Producer = "ExcelCreator 5.0 for .NET"
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PDF.Producer = "ExcelCreator 5.0 for .NET";
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

Creator プロパティ

PDF ファイルの文書プロパティの概要［アプリケーション］を設定します。

書式

[VB.NET]
Public WriteOnly Property PDF.Creator As String

[C#]
Public string PDF.Creator [set]

解説

デフォルトは「ExcelCreator 5.0 for .NET」です。

このプロパティで設定した内容は、Acrobat Reader 6 / 7 で開いた場合、文書プロパティの［アプリケーション］の項目として扱われます。Acrobat Reader 5 では［作成］の項目として扱われます。

コーディング例

PDF ファイルの文書プロパティの概要［アプリケーション］を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.Creator = "ExcelCreator 5.0 for .NET"
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PDF.Creator = "ExcelCreator 5.0 for .NET";
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

Author プロパティ

PDF ファイルの文書プロパティの概要［作成者］を設定します。

書式

[VB.NET]
Public WriteOnly Property PDF.Author As String

[C#]
Public string PDF.Author [set]

解説

デフォルトは「Advance Software Corp」です。

コーディング例

PDF ファイルの文書プロパティの概要［作成者］を設定します。

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.Author = "作成者の名前"
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PDF.Author = "作成者の名前";
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

EmbedFonts プロパティ

PDF ファイルに埋め込むフォントを設定します。

書式

[VB.NET]
Public WriteOnly Property PDF.EmbedFonts As String()

[C#]
Public string[] PDF.EmbedFonts [set]

解説

PDF ファイルに埋め込むフォント名を設定します。
設定したフォントが Excel ファイル内で使用されていない場合は、PDF ファイルには埋め込まれません。
"*" を設定した場合、Excel ファイル内で使用しているフォントを全て埋め込みます。

コーディング例

「MS ゴシック」「MS P ゴシック」を、PDF ファイルの文書で使用しているフォントとして設定します。

```
[VB.NET]
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.EmbedFonts = New String() {"MS ゴシック", "MS P ゴシック"}
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)


[C#]
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PDF.EmbedFonts = new string[]{"MS ゴシック", "MS P ゴシック"};
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

ImageType プロパティ

PDF ファイルのイメージの種類を設定します。

書式

[VB.NET]
Public WriteOnly Property PDF.ImageType As ExcelCreator.xlImageType

[C#]
Public ExcelCreator.xlImageType PDF.ImageType [set]

設定値

xlImageType 列挙体のメンバで設定します。

定数	内容
BMP	ビットマップ形式
GIF	GIF 形式
JPEG	JPEG 形式（デフォルト値）

コーディング例

PDF ファイルのイメージの種類を設定します。

[VB.NET]

```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.ImageType = ExcelCreator.xlImageType.BMP
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)
```

[C#]

```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
xlsCreator1.PDF.ImageType = ExcelCreator.xlImageType.JPEG;
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

ProgressDialog プロパティ

PDF ファイル出力時の処理経過を表示するプログレスダイアログの表示 / 非表示を設定します。

書式

[VB.NET]
Public Property ProgressDialog As Boolean

[C#]
public bool ProgressDialog [get, set]

設定値

値	内容
True	プログレスダイアログを表示する
False	プログレスダイアログを表示しない（デフォルト値）

解説

アプリケーション中で処理の経過を制御する場合は、このプロパティを False に設定し、Progress イベントで処理の経過を制御します。
ProgressDialog プロパティが True の場合は、Progress イベントは発生しません。

参照

Progress イベント

コーディング例

PDF ファイル出力時にプログレスダイアログを表示しないようにするため、プログレスダイアログを非表示に設定します

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003)
XlsCreator1.PDF.ProgressDialog = False
XlsCreator1.CloseBook(True, Application.StartupPath & "¥PdfFile.pdf", True)
```

[C#]
```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 1, ExcelCreator.xlVersion.ver2003);
XlsCreator1.PDF.ProgressDialog = false;
xlsCreator1.CloseBook(true, Application.StartupPath + @"¥PdfFile.pdf", true);
```

第 5 章 コンポーネントの登録について

ExcelCreator 5.0 for .NET が提供する以下のコンポーネントのグローバルアセンブリキャッシュへの登録は、ExcelCreator 5.0 for .NET のインストーラが自動で行います。また、アンインストール時には自動で登録の解除を行います。

- ExcelCreator.dll　　コンポーネント本体
- ExcelCreator.XlsCrt.dll　　ExcelCreator 生成エンジン
- ExcelCreator.XlsPDF.dll　　PDF ファイル生成エンジン
- XlsCrtNet5.dll　　旧バージョン ExcelCreator .NET 下位互換エンジン

ExcelCreator 5.0 for .NET のインストーラによるグローバルアセンブリキャッシュへの登録が自動で行われなかった場合は、次の方法でコンポーネントのグローバルアセンブリキャッシュへの登録 / 解除を行ってください。
グローバルアセンブリキャッシュへの登録 / 解除の詳細については、Visual Studio .NET のオンラインヘルプをご参照ください。

・グローバルアセンブリキャッシュへ登録する方法

エクスプローラを使って登録します。
エクスプローラ上で、コンポーネントをグローバルアセンブリキャッシュフォルダ（ 例: C:¥WINDOWS¥assembly、Windows2000 の場合は C:¥WINNT¥assembly ）にドラッグ &ドロップします。

・グローバルアセンブリキャッシュから登録を解除する方法

エクスプローラを使って登録を解除します。
エクスプローラ上で、グローバルアセンブリキャッシュフォルダ（例 : C:¥WINDOWS¥assembly、Windows2000 の場合は C:¥WINNT¥assembly）を開きます。次に、コンポーネントを選択し、右メニューの[削除]をクリックしてください。

・以前のバージョンのコンポーネントを使用する方法

ExcelCreator 5.0 for .NET のアップデートモジュールでは、常に最新のコンポーネントを利用出来る様、最新のコンポーネントの再配置とグローバルアセンブリキャッシュへの再登録を自動的に行います。
しかし、過去のプロジェクトをリビルドするなど、以前のバージョンのコンポーネントを使用する必要がある場合には、以下の手順でコンポーネントを切り替えてください。

変更の操作において、Visual Studio .NET の［ツールボックス］へ、ExcelCreator 5.0 for .NET コンポーネントの削除と追加を行います。
［ツールボックス］の削除と追加を行うための、Visual Studio .NET 2002 / Visual Studio .NET 2003 の［ツールボックスのカスタマイズ］画面、Visual Studio 2005 の［ツールボックス アイテムの選択］画面を開く操作は、以下の方法で行ってください。

・Visual Studio 2005 の場合
メニュー［ツール］-［ツールボックス アイテムの選択］をクリックして［ツールボックス アイテムの選択］画面を開きます。

・Visual Studio .NET 2003 の場合
メニュー［ツール］-［ツールボックスのカスタマイズ］をクリックして［ツールボックスのカスタマイズ］画面を開きます。

・Visual Studio .NET 2002 の場合
メニュー［ツール］-［ツールボックス アイテムの追加と削除］をクリックし、［ツールボックスのカスタマイズ］画面を開きます。

ここでは、Ver5.0.0.1716 から Ver5.0.0.1628 の ExcelCreator 5.0 for .NET コンポーネントへ変更すると仮定しています。

1) Visual Studio .NET の［ツールボックス］から、ExcelCreator 5.0 for .NET コンポーネントを削除します。
［ツールボックスのカスタマイズ］画面、もしくは、［ツールボックス アイテムの選択］画面の［.NET Framework コンポーネント］タブの［XlsCreator］のチェックをオフにし、Visual Studio .NET を終了してください。
Visual Studio .NET 2002 、Visual Studio .NET 2003 、Visual Studio 2005 が共存する場合は、全バージョンのチェックボックスのチェックをオフにしてください。

※Visual Studio 2005 の画面です。

2) ExcelCreator 5.0 for .NET インストールフォルダ内の Bin フォルダにインストールされている unregGAC.bat を実行します。
ExcelCreator 5.0 for .NET コンポーネントが、グローバルアセンブリキャッシュから削除されます。

3) ExcelCreator 5.0 for .NET インストールフォルダ内の Bin フォルダの直下にある以下のファイルを削除します。

ExcelCreator.dll
ExcelCreator.XlsCrt.dll
ExcelCreator.XlsPDF.dll

4) 使用するバージョンのバックアップフォルダ内にある下記ファイルを、ExcelCreator 5.0 for .NET インストールフォルダ内の Bin フォルダへコピーします。

ExcelCreator.dll
ExcelCreator.XlsCrt.dll
ExcelCreator.XlsPDF.dll

5) ExcelCreator 5.0 for .NET インストールフォルダ内の Bin フォルダにインストールされている regGAC.bat を実行します。
ExcelCreator 5.0 for .NET コンポーネントが、グローバルアセンブリキャッシュへ登録されます。

6) Visual Studio .NET の［ツールボックス］に、ExcelCreator 5.0 for .NET コンポーネントを登録します。［ツールボックスのカスタマイズ］画面、もしくは、［ツールボックス アイテムの選択］画面の［.NET Framework コンポーネント］タブの［XlsCreator］のチェックをオンにします。
Visual Studio .NET 2002 、Visual Studio .NET 2003 、Visual Studio 2005 が共存する場合は、全バージョンのチェックボックスのチェックをオンにしてください。

※Visual Studio 2005 の画面です。

・注意事項 ・ コンポーネントの切り替えは、1 の操作を必ず行い、ExcelCreator 5.0 for .NET コンポーネントが Visual Studio .NET の［ツールボックス］に登録されていない状態（チェックがオフ）で行ってください。

・コンポーネントが［ツールボックス］に登録されている状態でアセンブリの切り替えを行った場合、プロジェクトのビルド時にエラーが発生する場合があります。その場合、［ツールボックスのカスタマイズ］画面、もしくは、［ツールボックス アイテムの選択］画面の［.NET Framework コンポーネント］タブにて、使用するコンポーネントのチェックをオン、使用しないコンポーネントのチェックをオフにしてください。

グローバルアセンブリキャッシュへの登録や解除の詳細については、Visual Studio .NET のオンラインヘルプを参照してください。

第 6 章 アプリケーションの配布について

◆再配布可能なファイル

ExcelCreator 5.0 for .NET を組み込んだアプリケーションを配布する場合、以下のファイルを実行環境に配布する必要があります。
※これらのファイルは、〈ExcelCreator 5.0 for .NET インストールフォルダ〉¥bin フォルダにあります。

・ ExcelCreator 5.0 for .NET コンポーネント
　・ExcelCreator.dll　　コンポーネント本体
　・ExcelCreator.XlsCrt.dll　　ExcelCreator 生成エンジン
　・ExcelCreator.XlsPDF.dll　　PDF ファイル生成エンジン

・ ExcelCreator .NET クラス
　・XlsCrtNet5.dll　　クラスライブラリ(旧バージョン ExcelCreator .NET 下位互換エンジン)

ExcelCreator 5.0 for .NET の使用許諾に基づき、アプリケーションの種類によってライセンス体系が異なります。詳しくはアプリケーションのライセンス体系をご参照ください。

◆配布先の環境

・.NET Framework 再配布可能パッケージのインストール
.NET Framework 対応のアプリケーションを実行するために必要な .NET Framework 再配布可能パッケージが、インストールされている必要があります。パッケージの入手・配布方法等の詳細は、Visual Studio .NET のオンラインマニュアルを参照してください。

・C ランタイムファイル (MSVCR71.dll) の配布
.NET Framework 1.0 (SP3)、もしくは .NET Framework 2.0 のみがインストールされた環境の場合、C ランタイムファイルを開発環境からコピーして配布する必要があります。
C ランタイムファイル は、アプリケーションと同一フォルダ、またはシステムフォルダに配布してください。

◆配布可能ファイルの登録

通常、配布先のマシンに ExcelCreator 5.0 for .NET コンポーネントおよび ExcelCreator .NET クラスを登録する必要はありません。
ただし、これらの再配布可能ファイルを複数のプログラムから参照する場合、下記のいずれかの方法で登録する必要があります。

1) グローバルアセンブリキャッシュに登録する
2) コンフィグレーションファイルに登録する
ASP.NET 2.0 を使用する Web アプリケーションを配布する場合、再配布可能なファイルの配布は 1 の方法で行ってください。

1) グローバルアセンブリキャッシュに登録する
.NET Framework SDK のグローバルアセンブリキャッシュツール (gacutil.exe) を使用して、グローバルアセンブリキャッシュに ExcelCreator 5.0 for .NET コンポーネント等を登録します。
コマンドプロンプトまたは[ファイル名を指定して実行]ダイアログボックスで以下のように実行します。

・登録例:
C:¥gacutil.exe /i "C:¥Program Files¥Application¥ExcelCreator.dll"

※ExcelCreator .NET クラス (XlsCrtNet5.dll) を登録や登録解除する場合も、同様の手順で行います。

登録例は、gacutil.exe が C ドライブのルートに存在し、ExcelCreator 5.0 for .NET コンポーネント (ExcelCreator.dll) が C:¥Program Files¥Application ディレクトリにあると仮定しています。

また、登録の解除は、下記のようにします。

・登録の解除例：
C:¥gacutil.exe /i "C:¥Program Files¥Application¥ExcelCreator"

登録の解除は、dll ファイル名ではなくアセンブリ名を指定することに注意してください。
登録時は、Administrator 権限でログオンしておく必要があります。

2) コンフィグレーションファイルに登録する
アプリケーションコンフィギュレーションファイルの 〈probing〉 タグにコンポーネントを配置するサブフォルダ名を記述します。

gacutil.exe、コンフィグレーションファイルの詳細については、Visual Studio .NET のオンラインマニュアルを参照してください。

第 7 章　サポートサービスについて

◆ユーザー登録について

サポートサービスをご利用いただくにあたり、製品のユーザー登録が必要になります。登録されていない場合は、サポートサービスを利用できませんので、お問い合わせいただく前に必ずユーザー登録を行ってください。

◆ユーザー登録方法

製品に同梱されているユーザー登録カードに、必要事項を記入してご郵送ください。
また、弊社ホームページ（http://www.adv.co.jp/）のユーザー登録ページからもご登録いただくことができます。

◆サポートサービスについて

本製品の ReadMe、および、ヘルプに記述されていない事項、または動作に関するご不明な点などについては、サポートサービスをご利用ください。
本製品のサポートサービスは以下のサービスで構成されています。

■E-Mail サポート
■電話サポート
■FAX サポート

◆お問い合わせについて

ExcelCreator 5.0 for .NET 以外の事柄やプログラミング技法など一般的なご質問、または、動作保証環境外での使用によるご質問にはお答えいたしかねます。あらかじめ、ご了承ください。

◆お問い合わせになる前に

サポートサービスでは下記の情報が必要になります。
お問い合わせいただく前に、あらかじめこれらの情報をご用意ください。

・ユーザー情報の内容

(1) ExcelCreator 5.0 for .NET のシリアルナンバー（製品 CD-ROM ケース裏面のシールをご覧ください）
(2) 会社名
(3) 部署名
(4) 担当者名
(5) 連絡先（電話番号）
(6) FAX 番号
(7) E-Mail アドレス

・お問い合わせ内容

(8) 製品名（ExcelCreator 5.0 for .NET）
(9) ExcelCreator 5.0 for .NET の 4 桁の詳細バージョン
(10) 使用マシンの OS
(11) サービスパック（WindowsNT4.0/2000/XP の場合）
(12) .NET Framework のバージョン
(13) Visual Studio .NET のバージョン
(14) 問題となっている現象の内容
　① 現象の詳細内容

② 現象発生までの経緯
③ 現象の発生箇所
④ 現象を再現できる簡単なプロジェクト一式、もしくは、コーディングの一部

なお、お問い合わせの前に、以下の資料をご確認ください。回答となる情報が含まれている場合もあります。

・FAQ (ExcelCreator 5.0 for .NET ヘルプ 目次 [技術サポート情報])
・弊社ホームページ (http://www.adv.co.jp/support.htm) のサポートページ

◆お問い合わせ窓口

製品の操作方法や仕様に関するお問い合わせ窓口
電話番号　0776-21-9172
FAX 番号　0776-21-9022
E-Mail　xlscrt@adv.co.jp
受付時間　10:00 ～ 12:00 / 13:30 ～ 17:00

製品の営業に関するお問い合わせ窓口
電話番号　0776-21-9008
FAX 番号　0776-21-9022
E-Mail　info@adv.co.jp
受付時間　9:00 ～ 12:00 / 13:00 ～ 18:00

月曜日～金曜日 (祝祭日、弊社特別休業日を除く) にご利用いただけます。
※営業日以外にいただいた FAX/E-Mail でのお問い合わせは、翌営業日に回答致します。

第 8 章　技術サポート情報

1.　セル情報の設定と取得について

◆　Cell クラスの場合

Cell クラスにて対象セルの範囲指定を行い、各メソッドやプロパティに設定した場合、Cell クラスの範囲指定方法により属性や値の設定されるセル、値の取得されるセルに注意してください。

✧　設定例 1.

複数の範囲セルを指定し、属性と値の設定を行った場合、次のような結果となります。
セル（A1 ～ B2 と D3 ～ C3 と D4）を対象セルとします。このセルに対してフォント名を「MS ゴシック」、フォントサイズを「12 ポイント」、背景色を「水色（xcCyan)」、文字列「"abc"」を設定します。

```
XlsCreator1.CreateBook(Applicaiotn.StartupPath & "¥ExcelFile.xls", 3, _
                       ExcelCreator.xlVersion.ver2002)

'セル範囲指定し、書式情報を設定します
XlsCreator1.Cell("A1:B2, D3:C3, D4").Attr.FontName = "MS ゴシック"
XlsCreator1.Cell("A1:B2, D3:C3, D4").Attr.FontPoint = 12
XlsCreator1.Cell("A1:B2, D3:C3, D4").Attr.BackColor = ExcelCreator.xlColor.xcCyan
XlsCreator1.Cell("A1:B2, D3:C3, D4").Str = "abc"

XlsCreator1.CloseBook(True)
```

実行結果、下記の Excel ファイルが作成されます。
・ExcelFile.xls

	A	B	C	D	E
1	abc				
2					
3			abc		
4				abc	
5					
6					

属性については、セル（A1 ～ B2 と C3 ～ D3 と D4）に設定されます。
値については、セル（A1 と C3 と D4）に設定されます。
尚、セル範囲指定時に XlsCreator1.Cell("B2:A1, C3:D3, D4") と指定範囲を逆に記述しても同じ結果となります。

✧　設定例 2.

複数の範囲セルを指定し、値の取得を行った場合、次のような結果となります。
次のようなシートに対して、セル（A1 ～ B2 と C3 ～ D3 と D4）を対象セルとします。

・ExcelFile.xls

	A	B	C	D	E
1	aaa1	bbb1	ccc1	ddd1	
2	aaa2	bbb2	ccc2	ddd2	
3	aaa3	bbb3	ccc3	ddd3	
4	aaa4	bbb4	ccc4	ddd4	
5					
6					

```
Dim szData1 As String
Dim szData2 As String
Dim szData3 As String

'範囲（A1 ～ B2 と C3 ～ D3 と D4）を指定します
szData1 = XlsCreator1.Cell("A1:B2, C3:D3, D4").Str      'szData1 には、aaa1 が入ります

'前記と範囲は同じですが、指定順序を変えると取得される値が変わります
'範囲（C3 ～ D3 と A1 ～ B2 と D4）を指定します
szData2 = XlsCreator1.Cell("C3:D3, A1:B2, D4").Str      'szData2 には、ccc3 が入ります

'前記と指定順序は同じで、範囲指定記述を変えても取得される値は同じです
'範囲（C3 ～ D3 と A1 ～ B2 と D4）を指定します
szData3 = XlsCreator1.Cell("D3:C3, B2:A1, D4").str      'szData3 には、ccc3 が入ります

XlsCreator1.CloseBook(True)
```

実行結果、Excel ファイルの各セルから、取得できる値は以下の通りです。
Cell("A1:B2, C3:D3, D4") と指定した場合、取得対象セルは A1 セルとなり、aaa1 を取得します。
Cell("C3:D3, A1:B2, D4") と指定した場合、取得対象セルは C3 セルとなり、ccc3 を取得します。
Cell("D3:C3, B2:A1, D4") と指定した場合、取得対象セルは C3 セルとなり、ccc3 を取得します。

◆　Pos クラスの場合

Pos クラスにて対象セルの範囲指定を行い、各メソッドやプロパティに設定した場合、Pos クラスの範囲指定方法により属性や値の設定されるセル、値の取得されるセルに注意してください。

✧　設定例 1.

複数の範囲セルを指定し、属性と値の設定を行った場合、次のような結果となります。
セル（A1～C3）を対象セルとします。このセルに対してフォント名を「MS ゴシック」、フォントサイズを「12 ポイント」、背景色を「水色（xcCyan)」、文字列「″abc″」を設定します。

```
XlsCreator1.CreateBook(Applicaiotn.StartUpPath & "¥ExcelFile.xls", 3, _
                       ExcelCreator.xlVersion.ver2002)

'セル範囲指定し、書式情報を設定します
XlsCreator1.Pos(0, 0, 2, 2).Attr.FontName = "MS ゴシック"
XlsCreator1.Pos(0, 0, 2, 2).Attr.FontPoint = 12
XlsCreator1.Pos(0, 0, 2, 2).Attr.BackColor = ExcelCreator.xlColor.xcCyan
XlsCreator1.Pos(0, 0, 2, 2).Str = "abc"

XlsCreator1.CloseBook(True)
```

実行結果、下記の Excel ファイルが作成されます。
・ExcelFile.xls

	A	B	C	D
1	abc			
2				
3				
4				
5				

属性については、セル（A1 ～ C3）に設定されます。
値については、セル（A1）に設定されます。

✧　設定例 2.

複数の範囲セルを指定し、値の取得を行った場合、次のような結果となります。
次のようなシートに対して、セル(A1～D4)を対象セルとします。

・ExcelFile.xls

	A	B	C	D
1	aaa1	bbb1	ccc1	ddd1
2	aaa2	bbb2	ccc2	ddd2
3	aaa3	bbb3	ccc3	ddd3
4	aaa4	bbb4	ccc4	ddd4
5				

```
Dim szData1 As String
Dim szData2 As String

XlsCreator1.ReadBook(Applicaiotn.StartUpPath & "¥ExcelFile.xls", "")

'範囲（A1 ～ D4）を指定し、文字列を取得します
szData1 = XlsCreator1.Pos(0, 0, 3, 3).Str       'szData1 には、aaa1 が入ります
```

```
'前記と範囲は同じで、指定順序を変えても、取得される値は変わりません
szData2 = XlsCreator1.Pos(3, 3, 0, 0).Str       'szData2 には、aaa1 が入ります

XlsCreator1.CloseBook(True)
```

実行結果、Excel ファイルの各セルから、取得できる値は以下の通りです。
XlsCreator1.Pos(0, 0, 3, 3) と指定した場合、取得対象セルの範囲は A1 ～ D4 セルとなり、aaa1 を取得します。
XlsCreator1.Pos(3, 3, 0, 0) と指定した場合、取得対象セルの範囲は D4 ～ A1 セルとなり、aaa1 を取得します。

なお、Pos(0, 0, 3, 3) と指定しても、Pos(3, 3, 0, 0) と指定しても、取得対象セルは A1 セルとなり、aaa1 を取得します。

2. C# .NET で引数に 2 つ以上の列挙体のメンバを設定する方法

C# .NET で LineTop プロパティの様に、引数に 2 つ以上の列挙体のメンバを設定する場合、System.Int32 型にキャストした各列挙体のメンバを + 演算子で組み合わせ、全体を対応する列挙体のメンバにキャストします。
下記に LineTop プロパティを使用して、引数に罫線の種類と色を示す各列挙体のメンバを組み合わせて設定する方法を記載します。

◆コーディング例

[C#]

```
xlsCreator1.CreateBook(Application.StartupPath + @"¥ExcelFile.xls", 3,
ExcelCreator.xlVersion.ver2003);

// セル C2 から C5 に赤色の左罫線（太線）を設定します
xlsCreator1.Cell("C2:C5").Attr.LineTop =
(ExcelCreator.xlLineStyle)((Int32)ExcelCreator.xlLineStyle.lsThick +
                           (Int32)ExcelCreator.xlColor.xcRed);
// セル F2 から F5 に緑色の右罫線（点線）を設定します
xlsCreator1.Cell("F2:F5").Attr.LineTop =
(ExcelCreator.xlLineStyle)((Int32)ExcelCreator.xlLineStyle.lsDot +
                           (Int32)ExcelCreator.xlColor.xcGreen);

xlsCreator1.CloseBook(true);
```

3. 日付形式のセル値の取得と設定について

Excel ファイルの日付形式のセル値は、Excel ファイル内部でシリアル値（数値）として管理されているため、そのままの日付形式で取得することはできません。
日付の表示形式で取得を行う場合、ExcelCreator 5.0 for .NET で取得した値を、各言語のサポートする書式設定関数を使用して、任意の表示形式へ変更してください。

◆取得するコーディング例

[VB.NET]
```
Dim dtDate As DateTime
Dim strDate As String

XlsCreator1.ReadBook(Application.StartupPath & "¥SrcDate.xls")

' 数値を DateTime クラスに変換
dtDate = DateTime.FromOADate(Convert.ToDouble(XlsCreator1.Cell("A1").Value))
' DateTime クラスを文字列に変換
strDate = dtDate.ToString("yyyy/MM/dd")

XlsCreator1.CloseBook(True)

MsgBox("日付:" & strDate)
```

[C#]
```
DateTime dtDate;
string strDate;

xlsCreator1.ReadBook(Application.StartupPath + @"¥SrcDate.xls");

// 数値を DateTime クラスに変換
dtDate = DateTime.FromOADate(Convert.ToDouble(xlsCreator1.Cell("A1").Value));
// DateTime クラスを文字列に変換
strDate = dtDate.ToString("yyyy/MM/dd");

xlsCreator1.CloseBook(true);

MessageBox.Show("日付:" + strDate);
```

また、日付形式のセルに値を設定する場合も、各言語のサポートするデータ変換関数を使用して、日付型のデータを設定してください。

◆設定するコーディング例

[VB.NET]
```
XlsCreator1.CreateBook(Application.StartupPath & "¥SrcDate.xls",1,
                       XlsCreator.xlVersion.ver2002)

' 日付を Double クラスへ変換し、設定
XlsCreator1.Cell("A1").Value = Convert.ToDateTime("2004/5/20").ToOADate()
' 書式へ、日付型を設定
XlsCreator1.Cell("A1").Format = "yyyy/m/d"

XlsCreator1.CloseBook(True)
```

[C#]

```
xlsCreator1.CreateBook(Application.StartupPath + @"¥SrcDate.xls", 1,
                       XlsCreator.xlVersion.ver2002);

// 日付を double クラスへ変換し、設定
xlsCreator1.Cell("A1").Value = Convert.ToDateTime("2004/5/20").ToOADate();
// 書式へ、日付型を設定
xlsCreator1.Cell("A1").Format = "yyyy/m/d";

xlsCreator1.CloseBook(true);
```

◆日付書式を年号（和暦）で設定する際の注意点

実行環境にインストールされている Excel のバージョンが 2002(XP) 以降の場合、表示書式を「ee/mm/dd」と定義し、値を設定しても、「2004/02/25」のように西暦で表示されます。
Excel2002(XP) 以降の場合、日付の書式にて年号「e」を使用する場合に、日付の言語・地域を指定する必要がありますが、ExcelCreator 5.0 for .NET では、Excel ファイルの言語・地域の指定に対応していないため、書式設定時に「日本 ([$-411]) 」を指定してください。

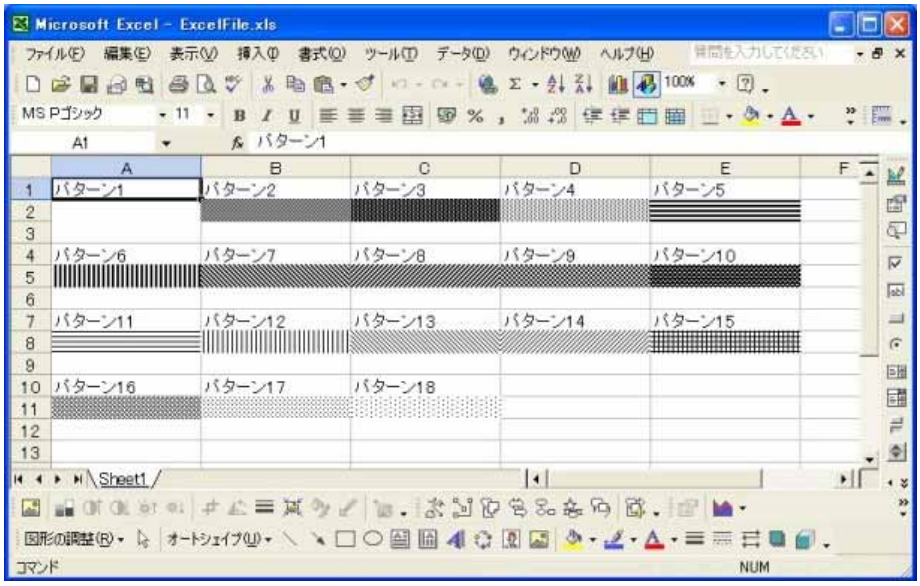
例 : [$-411]ee/mm/dd

4. PDF 出力時について

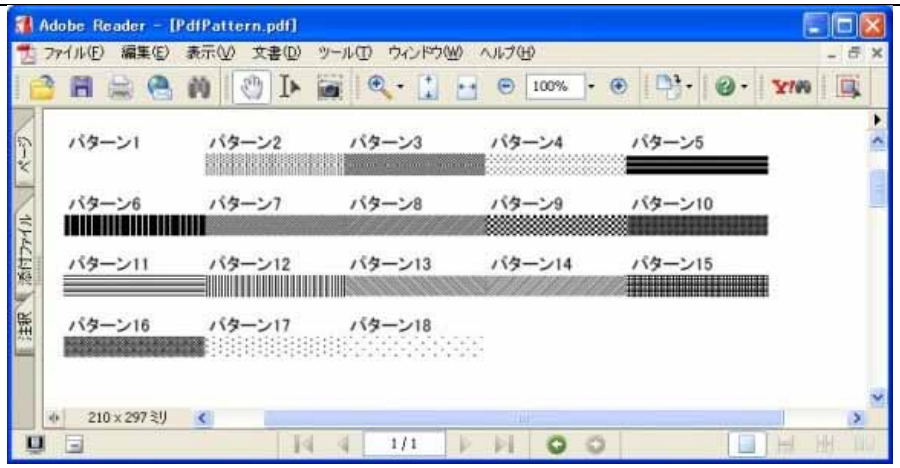
◆パターンを設定した場合の PDF ファイル出力結果について

Pattern プロパティでパターンを設定した PDF ファイルでは、以下の様にイメージが異なって出力されます。

・Excel ファイル出力結果



・PDF ファイル出力結果



◆プリンタのインストールされていない環境で PDF ファイルへの保存について

ExcelCreator 5.0 for .NET の PDF ファイルの出力時、実行環境にインストールされている「通常使用するプリンタ」の解像度を使用します。
出力する PDF ファイルに Pattern プロパティでパターンを設定した場合、前述の「通常使用するプリンタ」の解像度を基準としますが、プリンタのインストールされていない環境では画面の解像度 96dpi を基準としますので、パターンは粗く出力されます。
また、「通常使用するプリンタ」の解像度が低い（例:120dpi）場合でも同様にパターンは粗く出力されます。

第 9 章　互換性について

1.　ExcelCreator .NET との互換性について

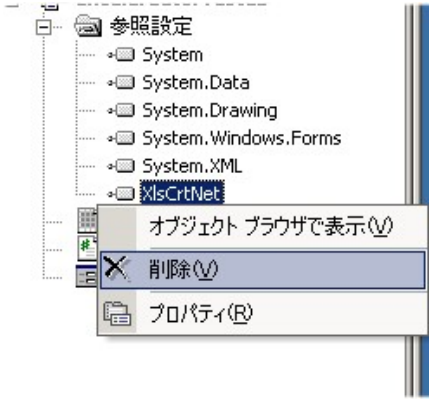
◆互換性について

ExcelCreator 5.0 for .NET コンポーネントは、ExcelCreator .NET クラスとの互換性はありません。
なお、ExcelCreator .NET クラスをご使用のお客様用に、ExcelCreator 5.0 for .NET では新しいバージョンの ExcelCreator .NET クラスを同梱しています。ExcelCreator .NET クラスを新しいバージョンへの変更方法は下記の説明を参照してください。

◆新バージョン ExcelCreator .NET クラスへの変更方法

プログラム中で使用している旧バージョン ExcelCreator .NET コントロールを新バージョン ExcelCreator .NET クラスに置き換えます。
プログラムのアセンブリ名は変更する必要はありません。

1)　プロジェクトの参照設定から、旧バージョン ExcelCreator .NET コンポーネント（XlsCrtNet.dll）を削除します。



2)　プロジェクトの参照設定へ、新バージョン ExcelCreator .NET クラスを追加します。

(1)　Visual Studio .NET のメニュー［プロジェクト］-［参照の追加］で［参照の追加］画面を開きます。

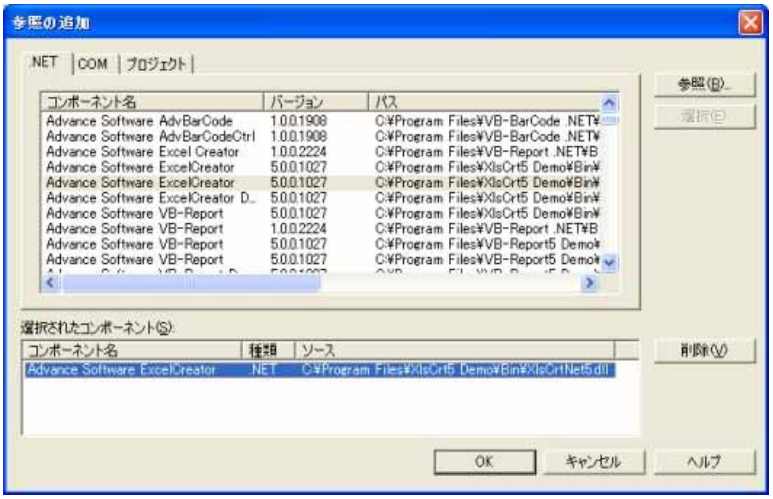
(2)　［参照の追加］画面の［.NET］タブを選択します。

(3)　下記の ExcelCreator .NET クラスを選択します。

コンポーネント名	パス
Advance Software ExcelCreator	C:¥Program Files¥XlsCrt5¥Bin¥XlsCrtNet5.dll

※パスは ExcelCreator 5.0 for .NET のデフォルトインストールフォルダの場合です。

(4)　上記のコンポーネントを選択した状態で［選択］ボタンをクリックし、［選択したコンポーネント］に追加します。



(5)　[OK]ボタンをクリックし、[参照の追加]画面を閉じます。
ExcelCreator .NET クラスが参照設定に追加されます。

・参照設定

2.　ActiveX 版との互換性について

Excel クリエーター Ver2，ExcelCreator Ver3.0（ActiveX 版）の ActiveX 版対応製品につきましては、共に ExcelCreator 5.0 for .NET との互換性はありません。
また、ActiveX 版のコーディングを、ExcelCreator 5.0 for .NET 対応のコーディングへ変換するツールはありません。

ExcelCreator 5.0 for .NET 用　問い合わせ用紙

送信枚数(当紙含む)　　枚
送信日付　　年　　月　　日

宛先 アドバンスソフトウェア㈱ ユーザーサポート係 行き

会社名			
所　属			
お名前			
連絡方法	電子メール ・ 郵送 ・ TEL ・ FAX		
ご住所			
ＴＥＬ		ＦＡＸ	
E－Mail			

シリアルNo.		使用機種	
使用言語(サービスパックの情報)	(　　)	使用OS(サービスパックの情報)	(　　)
メモリ容量			
プリンタ・ドライバの情報、ネットワークの環境			

不明内容をできるだけ詳しくご記入ください。なお、不具合時には現象を確認する必要があるため、『確実な再現手順』をお書き添え下さい。必要最低限のソースコード(データベースなど環境に依存しないもの)Excelファイルを添付していただきますと、問題解決に有効です。

不明な点がございましたら、この用紙をコピーして、必要事項をご記入の上、下記のFAX番号宛、また、郵送の場合は下記の住所宛にお送りください。
なお、電子メールでのお問い合わせも受け付けております。(E-mail アドレス : xlscrt@adv.co.jp)

アドバンスソフトウェア㈱ 〒918-8237 福井県福井市和田東 1 丁目 222 番地 TEL(0776)21-9008 FAX(0776)21-9022