PDF for .NET

2018.04.09 更新

グレープシティ株式会社

目次

はじめに 3
コンポーネントをプロジェクトに組み込む方法 3-4
コンポーネントのランタイムファイル 4
製品の概要 5
更新履歴 5
2.0J 5
新機能 5-6
クラスメンバ 6
サンプル 6-7
1.0J 7
新機能 7
クラスメンバ 7-8
主な特長 9-10
PDF for .NET の使い方 11
ドキュメントの作成 11-13
テキスト 13
テキストの描画 13-14
テキストの測定 14-15
テキストのフロー制御 15-16
RTF テキストの描画 16-17
画像の描画 17-18
グラフィックの描画 18-20
ページの追加とオーバーレイ描画 20-23
ブックマークやリンクの追加 23
ブックマークの追加 23-25
リンクの追加 25-26
ファイルの添付 26-28
セキュリティと制限の適用 28-29
メタファイル画像の描画 29-30
複雑な RTF テキストの描画 30-33
タスク別ヘルプ 34

Copyright © GrapeCity Inc. All rights reserved.

ドキュメントプロパティを設定する 34-38
ドキュメントに制限を設定する 38-41
ページサイズを設定する 41-42
ページの向きを設定する 42
画質を設定する 42-43
PDF ファイルを印刷する 43-44
他のコンポーネントとの連携 44
C1FlexGrid データを追加する 44
VSPrinter で表示したレポートからの PDF の作成 44-45
C1Report ドキュメントから PDF ブックマークを削除する 45-46

Copyright © GrapeCity Inc. All rights reserved.

はじめに

PDF for .NET は、.NET Framework アプリケーションから PDF ドキュメントを作成できるライブラリです。

PDF（ポータブルドキュメントフォーマット）形式は、1993 年に Adobe 社が開発して以来、電子システムの印刷表現の業界標準となっています。PDF は高品質で移植性が高く、デバイスに依存しません。また、PDF ドキュメントは無償のマルチプラットフォームドキュメントビューア Adobe Reader で閲覧できます。PDF ドキュメントの移植性と高品質により、PDF は Web パブリッシングの素晴らしい選択肢となっています。Adobe、およびその他のソースからの PDF 形式と PDF ユーティリティの詳細については、Adobe 社の Web サイト(www.adobe.com)をご覧ください。

コンポーネントをプロジェクトに組み込む方法

コンポーネントの組み込み

Visual Studio では、ツールボックスにコンポーネントを追加しただけでは、プロジェクトにコンポーネントを追加したことにはなりません。プロジェクトの参照設定へ追加された時点でコンポーネントが組み込まれます。

以下のいずれかの操作を行うとプロジェクトへコンポーネントが組み込まれます。

1. フォームにコンポーネントを配置する
2. ソリューションエクスプローラ上で参照の追加を行う

プロジェクトに組み込まれているコンポーネントの一覧は、ソリューションエクスプローラで確認できます。また、各コンポーネントが使用している DLL もソリューションエクスプローラに登録される場合があります。詳細については、Visual Studio の製品ヘルプを参照してください。

本製品で使用しているコンポーネントの一覧を以下に示します。

ファイル	内容
C1.C1Pdf.4.dll	本体アセンブリ（※）
C1.C1Pdf.4.Design.dll	デザイナアセンブリ（※）

※ .NET Framework 4 以上でご利用いただけます。

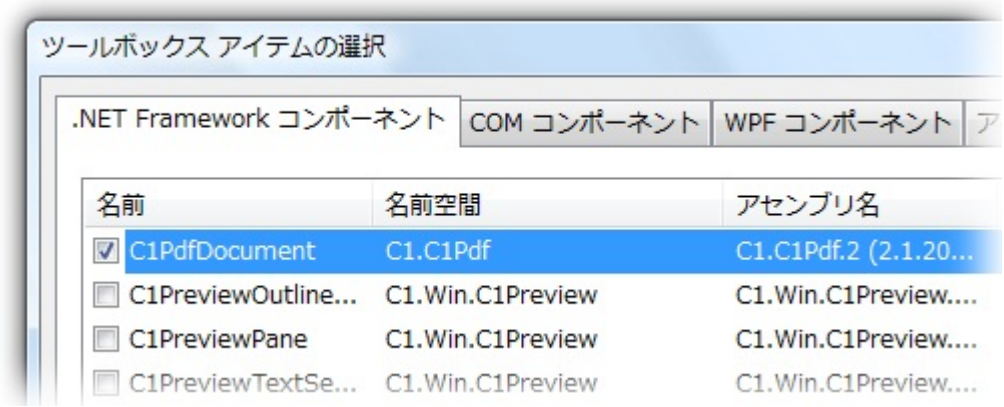
フォームにコンポーネントを配置する方法

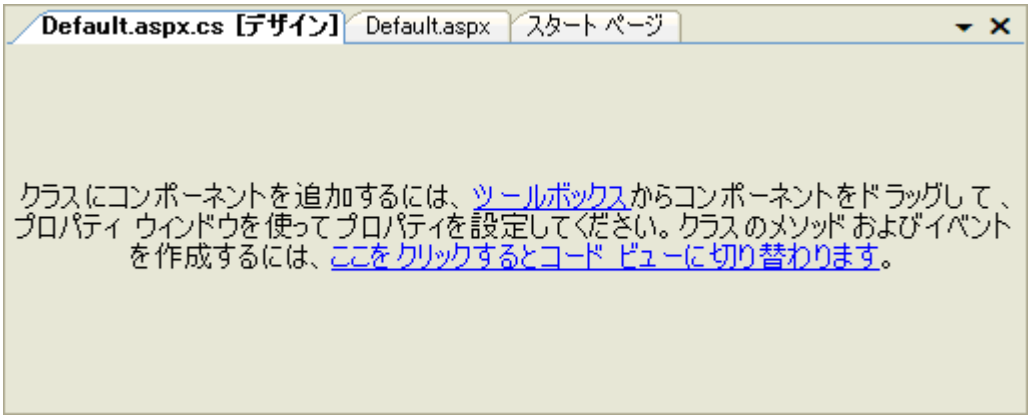
以下に、C1Pdf コントロールをツールボックスに追加し、フォームに配置する方法を示します。これにより、コンポーネントがプロジェクトに組み込まれます。

配置手順

1. [ツール]メニューから[ツールボックス アイテムの選択]を選択します。
2. [ツールボックス アイテムの選択]ダイアログの「.NET Framework コンポーネント」タブを選択します。
3. 使用するコンポーネントのチェックボックスを ON にして〈OK〉ボタンをクリックしてください。ツールボックスに指定したコントロールのアイコンが表示されます。

Copyright © GrapeCity Inc. All rights reserved.





4. ツールボックスから指定したコントロールのアイコンを選択してフォームに配置します。ソリューションエクスプローラの参照設定に指定したコントロールの名前空間が追加されます。

コンポーネントのランタイムファイル

Pdf for.NET のランタイムファイルは、C:\Program Files\ComponentOne\WinForms\bin\ フォルダにインストールされる次のファイルです。

ファイル	内容
C1.C1Pdf.4.dll	本体アセンブリ(※)

※ .NET Framework 4 以上でご利用いただけます。

注意:本製品に含まれているファイルのうち、上記以外のファイルは配布できません。

Copyright © GrapeCity Inc. All rights reserved.

製品の概要

PDF for .NETでは、PDF フォーム、電子署名、セキュリティ、圧縮、アウトライン、ハイパーリンク、添付ファイルといった、PDF の高度な機能の大部分をサポートしています。

しかも、**PDF for .NET** の主要な特長は使い方が簡単であることです。ドキュメントに内容を追加するための方法は、.NET の Graphics クラスと似ているため、.NET でのテキストとグラフィックスの表示方法を知っていれば、**PDF for .NET** の使い方がすぐ身につきます。

更新履歴

このセクションでは、旧バージョンのPDF for .NET の更新履歴について説明します。

現行バージョンの機能の改善、不具合の修正、対応環境の追加に関する情報については、リリースノートを参照してください。

2.0J

このセクションでは、PDF for .NET 2.0J の更新履歴について説明します。

新機能

PDF for .NET 2.0J では、以下の新機能が追加されました。

PDF フォーム

PDF ドキュメントに PDF フォームを追加できるようになりました。対応する PDF フォームは、チェックボックス、コンボボックス、リストボックス、ボタン、ラジオボタン、テキストボックスです。

使用例については、**PdfAcroform** サンプルを参照してください。

電子署名

PDF ドキュメントに電子署名を追加できるようになりました。

使用例については、**PdfSignature** サンプルを参照してください。

HTML 描画

PDF ドキュメントに HTML を描画できるようになりました。HTML を描画するための **MeasureStringHtml** と **DrawStringHtml** という2つの新しいメソッドが追加されました。

これらの新しいメソッドの文法は RTF テキスト描画メソッド（**MeasureStringRtf** と **DrawStringRtf**）と同じです。1ページ上の任意の矩形へ HTML を描画したり、複数ページへ HTML をフローイング描画したりすることができます。HTML 描画エンジンは CSS、テーブル、および画像をサポートします。フォーム、編集、またはハイパーリンクなどのインタラクティブな HTML 機能はサポートされません。

使用例については、**FlowHtml**サンプルを参照してください。

注釈

PDF ドキュメントに注釈を追加できるようになりました。

使用例については、**PdfAnnotations** サンプルを参照してください。

Copyright © GrapeCity Inc. All rights reserved.

テキストのクリッピング

テキスト描画で矩形領域を指定した際に、領域外のテキストが切り取られて表示されるようになりました。

クラスメンバ

PDF for .NET 2.0J では、以下のクラスメンバが追加されました。

PDF フォーム

メンバ	説明
C1PdfDocument.AddField メソッド	フォームフィールドを現在のページに追加します。
PdfCheckBox クラス	チェックボックスを表します。
PdfComboBox クラス	コンボボックスを表します。
PdfListBox クラス	リストボックスを表します。
PdfPushButton クラス	ボタンを表します。
PdfRadioButton クラス	ラジオボタンを表します。
PdfTextBox クラス	テキストボックスを表します。
PdfFieldBase クラス	他のすべての **PDF フォーム** フィールドクラス（**フォームフィールド**）の派生元となる抽象基本クラス。

署名

メンバ	説明
PdfSignature クラス	PDF フォームの電子署名フィールドを表します。

HTML 描画

メンバ	説明
C1PdfDocument.DrawStringHtml メソッド	HTML 文字列を表示します。
C1PdfDocument.MeasureStringHtml メソッド	指定された条件で表示されたときの HTML 文字列を計測します。

注釈

メンバ	説明
C1PdfDocument.AddNote メソッド	注釈項目を現在のページに追加します。

サンプル

Copyright © GrapeCity Inc. All rights reserved.

PDF for .NET 2.0J では、以下のサンプルが追加されました。

サンプル	説明
FlowHtml	HTML を PDF ドキュメントに表示する方法を示します。
PdfAcroform	様々な PDF フォームを含む PDFを作成する方法を示します。
PdfAnnotations	PDF ドキュメントに様々な形式の注釈を追加します。
PdfSignature	PDF ドキュメントに電子署名を追加します。
PrintDocument	PrintDocument を PDF ファイルに表示します。

1.0J

このセクションでは、PDF for .NET 1.0J の更新履歴について説明します。

新機能

PDF for .NET 1.0J では、以下の新機能が追加されました。

ビューアオプション

PDF ビューアで開いたときに、ツールバーを隠す、見開きページ形式で表示するなど、PDF ドキュメントを表示する方法を設定することが可能になりました。

テキスト描画メソッドの拡張

以下のようにテキスト描画が拡張されました。

- RTF（リッチテキスト形式）テキストの幅を測定するためのメソッドが追加されました。
- DrawString メソッドに初期文字値を受け取るオーバーロードメソッドが追加されました。これで、DrawString と DrawStringRtf の整合性が高まりました。
- DrawStringRtf メソッドに戻り値が追加され、描画する初期文字値を受け取るオーバーロードメソッドが追加されました。これらの追加により、複数ページドキュメントに RTF テキストをフローできるようになりました。例については、**C1PdfDocument.DrawStringRtf(String, Font, Brush, RectangleF, Int32) メソッド** を参照してください。

ドキュメントプロパティの追加

PDF ドキュメントの作成日時を設定できるようになりました。

日本語フォント処理

以下のような日本語フォント処理が追加されました。

- 外字フォントの埋め込みに対応しました。
- JIS2004 フォントに対応しました。制限についてはリリースノートを参照してください。

クラスメンバ

PDF for .NET 1.0J では、以下のクラスメンバが追加されました。

Copyright © GrapeCity Inc. All rights reserved.

ビューアオプション

メンバ	説明
C1PdfDocumentBase.ViewerPreferences プロパティ	ビューアでこのドキュメントを表示する方法に関する情報が含まれる PdfViewerPreferences オブジェクトを取得します。
PdfViewerPreferences クラス	ビューアでドキュメントを表示する時に使用されるオプションを表します。
PageLayoutEnum 列挙体	ビューアで PDF ファイルを表示するときに使用されるページレイアウトを指定します。
PageModeEnum 列挙体	ビューアで PDF ファイルを表示するときの表示方法を指定します。

テキスト描画メソッドの拡張

メンバ	説明
C1PdfDocument.MeasureStringRtf メソッド	指定された条件で描画された RTF 文字列のサイズを測定します。
C1PdfDocument.DrawString(String, Font, Brush, RectangleF, Int32) メソッド	指定された StringFormat オブジェクトの書式設定属性を使用して、指定されたテキスト文字列を指定された四角形内に指定された Brush オブジェクトと Font オブジェクトで描画します。
C1PdfDocument.DrawString(String, Font, Brush, RectangleF, Int32, StringFormat) メソッド	指定された StringFormat オブジェクトの書式設定属性を使用して、指定されたテキスト文字列を指定された四角形内に指定された Brush オブジェクトと Font オブジェクトで描画します。
C1PdfDocument.DrawStringRtf(String, Font, Brush, RectangleF, Int32) メソッド	RTF 文字列を指定された四角形内に指定された Brush オブジェクトと Font オブジェクトで、文字列内の指定されたオフセットから描画します。

ドキュメントプロパティの追加

メンバ	説明
PdfDocumentInfo.CreationDate プロパティ	PDF ドキュメントの作成日時を取得または設定します。

Copyright © GrapeCity Inc. All rights reserved.

主な特長

PDF for .NET は、セキュリティ、圧縮、アウトライン、ハイパーリンク、および添付ファイルを含む、PDF 仕様に含まれる高度な機能の大部分をサポートします。

しかし、**PDF for .NET** の大きな特長は使い方が簡単である点です。コンテンツをドキュメントに追加するために備わっている各コマンドは .NET の **Graphics** クラスで使用可能なコマンドと似ています。.NET のテキストとグラフィックスを表示する方法を理解していれば、**PDF for .NET** を使用する方法も既に理解されています。

便利さを実感できる **PDF for .NET** の特長として、次の事項が挙げられます。

- **メタファイル画像の高速描画と圧縮**
  メタファイルはビットマップに変換されません。これらは、解析されてベクターグラフィックコマンドに変換され、可能な範囲で最良の解像度を保持します。グラフや図面を PDF ドキュメントに追加する場合は、メタファイルの方がビットマップ画像よりも適しています。
- **AcroForms サポート**
  新しい AddField メソッドを使用して、PDF ドキュメントに Acrobat フォームフィールドを追加します。PDF for .NET は、次のフィールドタイプをサポートします。テキストボックス、チェックボックス、ラジオボタン、プッシュボタン、コンボボックス、およびリストボックス。
- **HTML のサポート**
  新しい DrawStringHtml メソッドを使用して、PDF ドキュメントに HTMLをレンダリングします。HTML コンテンツを複数のページまたは列に流し込み、既存のスタイルシートを使用して、HTML を他のタイプのコンテンツ（画像、RTF、プレーンテキスト、フォームフィールドなど）と組み合わせることができます。
- **DrawImage メソッドを使用したなじみある構文**
  PDF ドキュメントへの画像の追加は簡単です。すべて DrawImage メソッドを使用して行われます。DrawImage は、指定された画像を指定された位置に描画し、画像の配置とスケール調整を制御するパラメータを備えています。メタファイルを含む、任意の標準 .NET Image オブジェクトを表示できます。
- **ドキュメントの制限の管理**
  ユーザーによるコンテンツのコピーおよび編集を許可したり、ユーザーによるドキュメントの印刷を制限したり、ユーザーに対して注釈の編集権限を設定したりできます。
- **PDF ファイルへの添付ファイルの追加**
  添付データには、メインドキュメントが雑多に含まれている詳細情報を含むスプレッドシード、ムービーやサウンドを含むマルチメディアファイル、サンプルコードなど、どんな種類のファイルも含めることができます。PDF ファイルへの添付データの追加は簡単です。添付するファイル、添付データを取り込むページの領域、およびオプションで添付データの外観を指定するだけです。
- **オーナーとユーザーのパスワード保護**
  PDF ファイルに機密情報を含める場合は、権限のあるユーザーだけがアクセスできるように、これらの情報を暗号化できます。ドキュメントのオーナーとその他すべてのユーザーには別個のパスワードが設定されます。ユーザーのアクセスは、ドキュメントの表示、印刷、または編集など、特定の操作のみを許可するように選択して制限できます。
- **グラフィカル要素の追加**
  線、矩形、省略記号、円、弧、角が丸い矩形、多角形、ベジエ曲線などを追加します。
- **アウトライン構造の作成**
  ほとんどの長文の PDF ドキュメントは、リーダの左側のペインに表示されるアウトライン構造を備えています。アウトラインを使用して、ドキュメントの構造を簡単にブラウズし、特定のトピックを見つけることができます。PDF for .NET では、アウトラインのエントリ（ブックマーク）を追加してこのアウトライン構造をビルドできます。
- **ハイパーリンクとローカルリンクの追加**
  PDF には、ハイパーリンクとハイパーリンクターゲットを PDF ドキュメントに追加するメソッドが用意されています。クリックすると、同じ PDF ドキュメント内の他の場所に移動できるローカルリンクを追加することもできます。このタイプのリンクは、目次や索引など、ドキュメント内に一種のクロス参照を実装する場合に便利です。
- **ドキュメント情報およびビューア環境設定の制御**
  PDF では、作成する PDF ドキュメントにメタデータを追加できます。作成者、作成日、キーワードなどを指定します。Adobe Reader でドキュメントを開いたときに適用されるデフォルトのビューア環境設定も提供できます。初期ページレイアウト、ウィンドウの位置およびリーダのツールバーとメニューの表示／非表示を指定します。
- **PDF/A のサポート**
  PDF/A は、通常、請求書、パンフレット、マニュアル、研究報告書などを作成するユーザーが、レポートを PDF/A 形式で保存するために使用します。また、JPEG2000 画像のエクスポート、電子署名の提供、および埋め込みフォントのサ

Copyright © GrapeCity Inc. All rights reserved.

ポートを可能にします。

Copyright © GrapeCity Inc. All rights reserved.

PDF for .NET の使い方

PDF for .NET の使い方

ドキュメントの作成

このセクションでは、**PDF for .NET** の使用方法について説明します。

このセクションでは、C1Pdf を使用した最もシンプルなアプリケーションを作成します。C1Pdf を初めて使う人でも、手順に従って簡単に PDF ファイルを作成することができます。このトピックでは、"Hello World" というテキストを含む PDF ファイルを作成する方法について説明します。

1. Windows フォームアプリケーションを作成して、フォームに C1PdfDocument コントロールを追加します。

   Visual Basic コードの書き方

   Visual Basic
   ```
   Imports System.Drawing
   ```

   C# コードの書き方

   C#
   ```
   using System.Drawing;
   ```

2. フォームのソースコードに次の **Imports** または **using** ディレクティブを追加します。

   Visual Basic コードの書き方

   Visual Basic
   ```
   ' コンテンツをドキュメントに追加します。
   Dim font As Font = New Font("Arial",12)
   C1PdfDocument1.DrawString("Hello World!", font, Brushes.Black, PointF.Empty)
   ```

   C# コードの書き方

   C#
   ```
   //コンテンツをドキュメントに追加します。
   Font font = new Font("Arial", 12);
   c1PdfDocument1.DrawString("Hello World!", font, Brushes.Black, PointF.Empty);
   ```

3. デザイン画面でフォームをダブルクリックして、**Form_Load** イベントハンドラに次のコードを記述します。これでテキストが PDF ドキュメントに追加されます。

   Visual Basic コードの書き方

   Visual Basic
   ```
   ' コンテンツをドキュメントに追加します。


   Dim font As Font = New Font("Arial",12)
   ```

   C# コードの書き方

Copyright © GrapeCity Inc. All rights reserved.

```
C#
// コンテンツをドキュメントに追加します。
RectangleF rect = pdf.PageRectangle;

Font font = new Font("Arial", 12);
```

4. **Save** メソッドを使用して、ドキュメントをファイルまたはストリームに保存します。

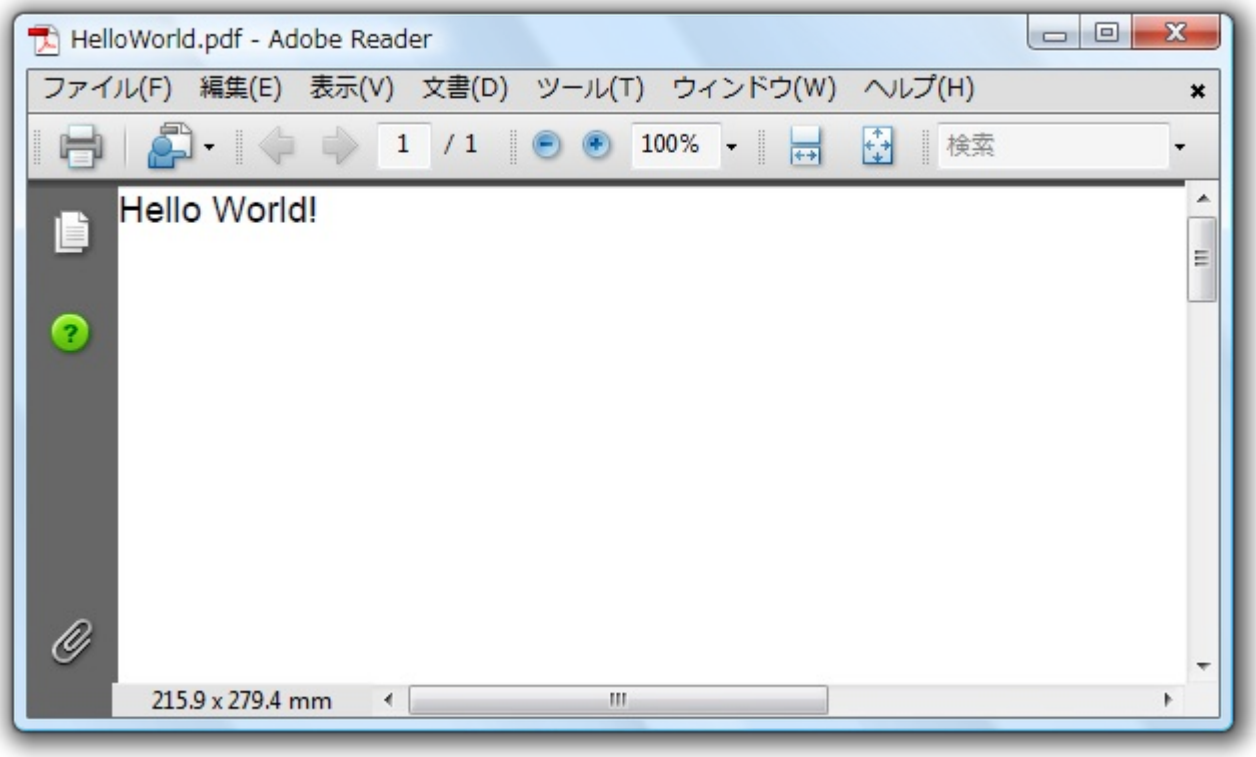
Visual Basic コードの書き方

```
Visual Basic
'ドキュメントをファイルに保存します。
C1PdfDocument1.Save("c:\HelloWorld.pdf")
```

C# コードの書き方

```
C#
//ドキュメントをファイルに保存します。
c1PdfDocument1.Save(@"c:\HelloWorld.pdf");
```

アプリケーションを実行すると、C ドライブの直下に「HelloWorld.pdf」という PDF ファイルが作成されます。



手順 3 では、**Font** オブジェクトを作成して DrawString メソッドを呼び出し、「Hello World!」というテキストを描画しています。これは、.NET Framework の **Graphics** オブジェクトとまったく同じ操作方法であるため、C1Pdf は非常に使いやすくなっています。

覚えておかなければならない重要なことは、**PDF for .NET** ではページの左上隅に起点を持つポイントベースの座標システムを使用することです。これは、.NET で使用されるデフォルトの座標システムと似ており、デフォルトの PDF 座標システム（起点はページの左下隅）とは異なります。この例では、「Hello World」の「H」の左上ポイントは [1,1] にあります。

座標システムがピクセルでなく、ポイントに基づくため、**PDF for .NET** は float 型のメンバを持つ **RectangleF**、**SizeF**、および

Copyright © GrapeCity Inc. All rights reserved.

PointF 構造体を使用し、**int** 型のメンバを持つ Rectangle、Size、および Point は使用しません。

テキスト

このセクションでは、テキストの描画、測定、フロー制御のほか、RTF テキストの描画などの方法について説明します。

テキストの描画

C1Pdf ドキュメントにテキストを描画するためには、**DrawString** メソッドを使用します。

DrawString は、指定のフォントとブラシを使用し、指定した位置に指定のテキストを描画します。たとえば、以下のように記述します.

Visual Basic コードの書き方

Visual Basic

```
C1PdfDocument1.DrawString("Hello World!", font, Brushes.Black, rect)
```

C# コードの書き方

C#

```
c1PdfDocument1.DrawString("Hello World!", font, Brushes.Black, rect);
```

デフォルトでは、DrawString はテキストを指定の四角形の左上に配置し、四角形内でテキストを折り返します。これらのオプションを変更するには、Graphics オブジェクトと同様に DrawString のパラメータに StringFormat オブジェクトを使用します。**StringFormat** は、水平方向の配置(**Alignment**)、垂直方向の配置(**LineAligmnent**)、折り返しなどを制御するフラグ(**FormatFlags**)を指定できるメンバを持ちます。

たとえば以下のコードでは、**StringFormat** オブジェクトを作成し、このオブジェクトを使用してテキストを四角形の垂直と水平の両方向の中央に配置します。

Visual Basic コードの書き方

Visual Basic

```
Dim font As New Font("MS PGothic", 12)
Dim rect As New RectangleF(72, 72, 100, 100)
Dim text As String = "小さい矩形領域内に描画するための長いテキスト。"
' テキストを中央配置します。
Dim sf As New StringFormat()
sf.Alignment = StringAlignment.Center
sf.LineAlignment = StringAlignment.Center
C1PdfDocument1.DrawString(text, font, Brushes.Black, rect, sf)
C1PdfDocument1.DrawRectangle(Pens.Gray, rect)
```

C# コードの書き方

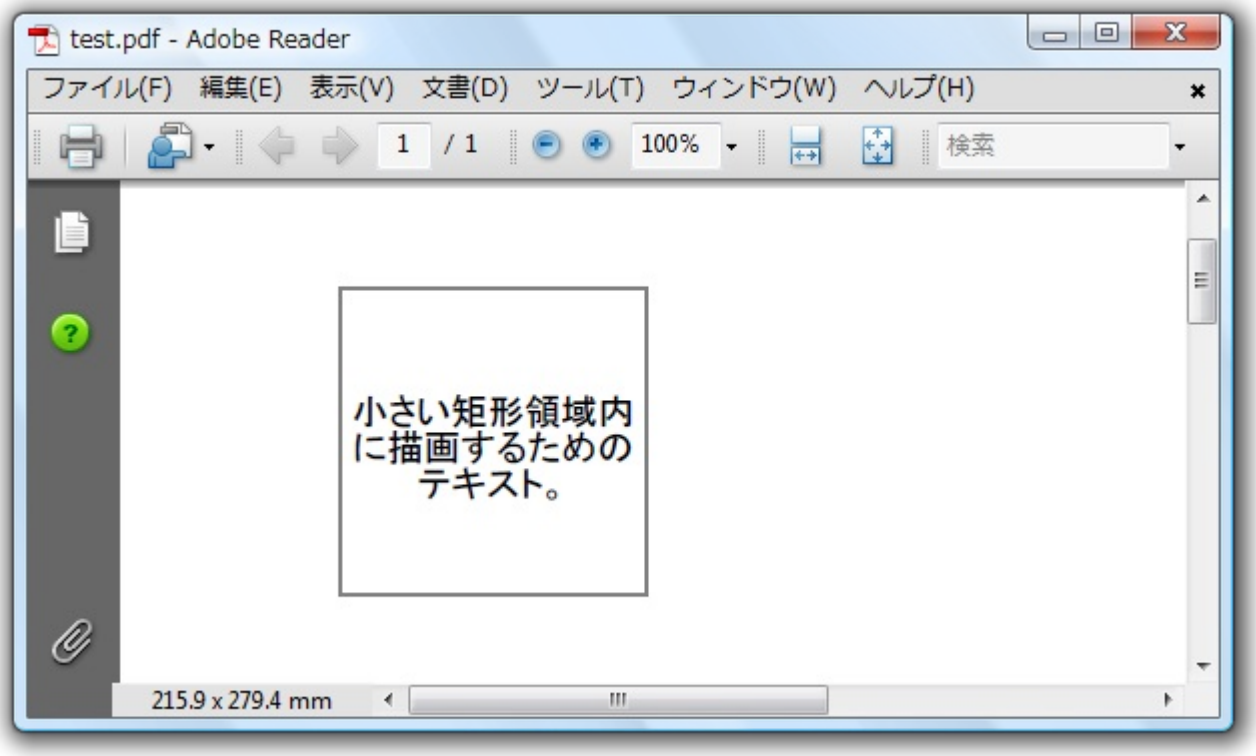
C#

```
Font font = new Font("MS PGothic", 12);
RectangleF rect = new RectangleF(72, 72, 100, 100);
string text = "小さい矩形領域内に描画するための長いテキスト。";
// テキストを中央配置します。
StringFormat sf = new StringFormat();
```


Copyright © GrapeCity Inc. All rights reserved.

```
sf.Alignment = StringAlignment.Center;
sf.LineAlignment = StringAlignment.Center;
c1PdfDocument1.DrawString(text, font, Brushes.Black, rect, sf);
c1PdfDocument1.DrawRectangle(Pens.Gray, rect)
```

このトピックの作業結果



テキストの測定

多くの場合、テキストを描画する前にテキストがページに収まるかどうかをチェックする必要があります。そのためには、**MeasureString** メソッドを使用できます。MeasureString は、指定のフォントで描画するときのテキストの幅と高さ(ポイント単位)を含む **SizeF** 構造を返します。

たとえば以下のコードでは、段落が現在のページに収まるかどうかを調べ、必要に応じてページ区切りを挿入します。これにより、段落を1ページ内にまとめることができます。

Visual Basic コードの書き方

Visual Basic

```
Private Function RenderParagraph(text As String, font As Font, rect As RectangleF,
rectPage As RectangleF) As Rectangle
    '必要な高さを計算します。
     Dim sz As SizeF = C1PdfDocument1.MeasureString(text, font, rect.Width)
     rect.Height = sz.Height
    'このページに収まらない場合、ページ区切りを挿入します。
     If rect.Bottom > rectPage.Bottom Then
      C1PdfDocument1.NewPage()
      rect.Y = rectPage.Top
     End If
```


Copyright © GrapeCity Inc. All rights reserved.

```
    'テキストを描画します。
     C1PdfDocument1.DrawString(text, font, Brushes.Black, rect)
     '次回の四角形を更新します。
    rect.Offset(0, rect.Height)
    Return rect
End Function
'RenderParagraph メソッドを使用します。
Dim font As New Font("MS PGothic", 10)
Dim rectPage As RectangleF = C1PdfDocument1.PageRectangle
rectPage.Inflate(-72, -72)
Dim rect As RectangleF = rectPage
Dim s As String
For Each s In myStringList
  rect = RenderParagraph(s, font, rect, rectPage)
Next s
```

C# コードの書き方

C#

```
private RectangleF RenderParagraph(string text, Font font, RectangleF rect,
RectangleF rectPage)
{
    //必要な高さを計算します。
    SizeF sz = C1PdfDocument1.MeasureString(text, font, rect.Width);
   rect.Height = sz.Height;
    //このページに収まらない場合、ページ区切りを挿入します。
    if (rect.Bottom > rectPage.Bottom)
    {
     c1PdfDocument1.NewPage();
     rect.Y = rectPage.Top;
    }
    //テキストを描画します。
    c1PdfDocument1.DrawString(text, font, Brushes.Black, rect);
    //次回の四角形を更新します。
    rect.Offset(0, rect.Height);
    return rect;
}
//RenderParagraph メソッドを使用します。
Font font = new Font("MS PGothic", 10);
RectangleF rectPage = c1PdfDocument1.PageRectangle;
rectPage.Inflate(-72, -72);
RectangleF rect = rectPage;
foreach (string s in myStringList)
{
 rect = RenderParagraph(s, font, rect, rectPage);
}
```

テキストのフロー制御

DrawStringメソッドは、整数を返します。これは、四角形に収まらなかったために、出力されなかった最初の文字のインデックスです。この値を使用し、ページからページへ、またはページ内のフレームからフレームへのテキストフローを制御することが

Copyright © GrapeCity Inc. All rights reserved.

できます。たとえば、以下のように記述します。

Visual Basic コードの書き方

Visual Basic

```
'複数のページにまたがるテキストを描画します。
While True
  '四角形に収まるだけのテキストを描画します。
   Dim nextChar As Integer
   nextChar = C1PdfDocument1.DrawString(text, font, Brushes.Black, rectPage)
   '完了したら終了します。
   If nextChar >= text.Length Then
    Exit While
   End If
   '描画された部分を削除します。
   text = text.Substring(nextChar);
   '次のページに移動します。
    C1PdfDocument1.NewPage()
End While
```

C# コードの書き方

C#

```
//複数のページにまたがるテキストを描画します。
while (true)
{
  //四角形に収まるだけのテキストを描画します。
  int nextChar = c1PdfDocument1.DrawString(text, font, Brushes.Black, rectPage);
  //完了したら終了します。
  if (nextChar >= text.Length)
  {
   break;
  }
  // 描画された部分を削除します。
   Text = text.Substring(nextChar);
  // 次のページに移動します。
   c1PdfDocument1.NewPage();
}
```

MeasureString メソッドと **DrawString** メソッドを組み合わせることにより、描画のルーチンを開発して、段落をどのように描画するかを自由に制御できます。

RTF テキストの描画

DrawString は、単一のフォントと単色を使用して段落を描画するときに必要なすべての機能を提供します。

複数のフォントと複数の色を組み合わせて、リッチテキスト形式で段落を描画する場合、代わりに DrawStringRtf メソッドを使用する必要があります。DrawStringRtf は DrawString と似ていますが、テキストを RTF として解釈する点が異なります。

たとえば以下のコードでは、太字と斜体文字を含む1行のテキストを描出します。

Visual Basic コードの書き方

Visual Basic

Copyright © GrapeCity Inc. All rights reserved.

```
Dim font As New Font("Arial", 12)
Dim rect As RectangleF = pdf.PageRectangle
rect.Inflate(- 72, - 72)
Dim s As String
s = "To {\b boldly} go where {\i no one} has gone before!"
C1pdf.DrawStringRtf(s, font, Brushes.Black, rect)
```

C# コードの書き方

C#

```
Font font = new Font("MS PGothic", 12);
RectangleF rect = c1PdfDocument1.PageRectangle;
rect.Inflate(-72, -72);
c1PdfDocument1.DrawStringRtf(@"To {\b boldly} go where {\i no one} has gone before!",
font, Brushes.Black, rect);
```

このテキストは、完全な RTF ドキュメントを含まないことに注意してください。このテキストは、\rtf1 のヘッダ、フォント、色のいずれのテーブルも持っていません。DrawStringRtf メソッドはそれを認識し、提供された *Font* 引数と *Brush* 引数に基づいて必要な RTF ヘッダを自動的に構築します。

テキストが完全な RTF ヘッダを含んでいた場合、その RTF ヘッダを DrawStringRtf への呼び出しで指定した Font 引数と Brush 引数の代わりに使用します。これが当てはまるのは、テキストを **RichTextBox** コントロールからコピーしたか、RTF ファイルから読み込んだ場合です。

画像の描画

C1**Pdf** ドキュメントに画像を追加するのは簡単であり、すべての操作は**DrawImage** メソッドで行います。

DrawImage メソッドは、指定した位置に指定した画像を描画し、さらに画像の配置とスケーリングの制御を提供する引数を備えています。たとえば以下のコードでは、同じ画像を3種類の方法で描画します。

Visual Basic コードの書き方

Visual Basic

```
Dim rect As RectangleF = C1PdfDocument1.PageRectangle
rect.Inflate(- 72, - 72)
'四角形に合わせて画像を引き伸ばします。
C1PdfDocument1.DrawImage(pictureBox1.Image, rect)
'画像を四角形の中央に配置し、縦横比を維持しながらスケーリングします。
C1PdfDocument1.DrawImage(pictureBox1.Image, rect, ContentAlignment.MiddleCenter,
C1.C1Pdf.ImageSizeModeEnum.Scale)
'画像を四角形の左上に描画します。
C1PdfDocument1.DrawImage(pictureBox1.Image, rect, ContentAlignment.TopLeft,
C1.C1Pdf.ImageSizeModeEnum.Clip)
```

C# コードの書き方

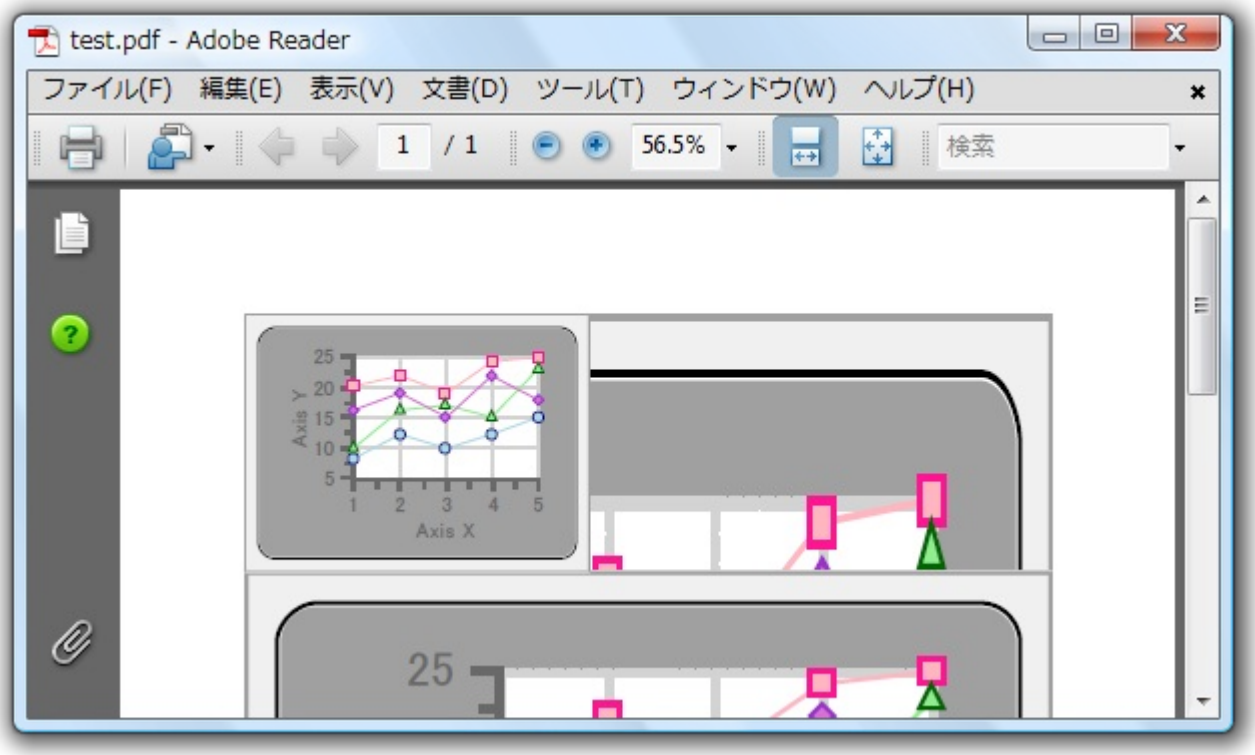
C#

```
RectangleF rect = c1PdfDocument1.PageRectangle;
rect.Inflate(-72, -72);
//四角形に合わせて画像を引き伸ばします。
c1PdfDocument1.DrawImage(pictureBox1.Image, rect);
//画像を四角形の中央に配置し、縦横比を維持しながらスケーリングします。
```


Copyright © GrapeCity Inc. All rights reserved.

```
c1PdfDocument1.DrawImage(pictureBox1.Image, rect, ContentAlignment.MiddleCenter,
C1.C1Pdf.ImageSizeModeEnum.Scale);
//画像を四角形の左上に描画します。
c1PdfDocument1.DrawImage(pictureBox1.Image, rect, ContentAlignment.TopLeft,
C1.C1Pdf.ImageSizeModeEnum.Clip);
```

このトピックの作業結果



C1Pdf では、ビットマップ画像やメタファイル画像などの .NET Image オブジェクトを描画できます。メタファイルは、ビットマップに変換せずにドキュメント内に描画されるため、可能な限りの最高の解像度を保持します。グラフまたは図面を PDF ドキュメントに追加する場合、メタファイルのほうがビットマップ画像より優れています。なお、**C1PDF**では、EMF+ メタファイルに対応しています。

ビットマップ画像は、**C1Pdf** により自動的に管理されます。ページヘッダなどに同じ画像を複数回描画するとき、画像の1コピーのみを PDF ファイルに保存します。画質とファイルサイズをトレードオフできる **C1PdfDocumentBase.ImageQuality** プロパティを使用し、画質を制御できます。

グラフィックの描画

C1PdfDocument クラスは、線、四角形、楕円、扇形、円弧、丸角の四角形、ポリゴン、ベジェ曲線などのグラフィカル要素をドキュメントに追加するいくつかのメソッドを提供します。

これらのメソッドは、.NET の **Graphics** クラスにあるメソッドのサブセットであり、同じ **Brush** クラスと **Pen** クラスを使用して線や塗り潰し領域の色とスタイルを制御します

C1Pdf では、ポイント単位の座標システムを使用しており、原点はページの左上に配置されることに注意してください(.NET の **Graphics** クラスのデフォルトの座標システムはピクセルベースです)。

以下の例では、**C1Pdf と** .NET の **Graphics** クラスのグラフィックスメソッドがどの程度類似しているかを示しています。サンプルコードは、g という C1PdfDocument クラスを宣言し、扇形、スプライン、およびその他のグラフィカル要素を描画するメソッドを呼び出します。

Copyright © GrapeCity Inc. All rights reserved.

このサンプルコードの要点は、C1PdfDocument クラスを通常の .NET **Graphics** オブジェクトと置き換えたとき、コードをコンパイルして同じ結果を得られるという点にあります。

Visual Basic コードの書き方

Visual Basic

```
'PDF ドキュメントを作成します。
Dim g As New C1.C1Pdf.C1PdfDocument()
'描画を設定します。
Dim rect As New RectangleF(0, 0, 300, 200)
Dim text As String = "Hello world of .NET Graphics and PDF."  ControlChars.Cr +
ControlChars.Lf + "Nice to meet you."
Dim font As New Font("Times New Roman ", 12, FontStyle.Italic Or FontStyle.Underline)
Dim bezierPoints() As PointF = { _
 New PointF(10F, 100F), New PointF(20F, 10F),, _
 New PointF(35F, 50F), New PointF(50F, 100F), _
 New PointF(60F, 150F), New PointF(65F, 100F), _
 New PointF(50F, 50F)}
'いくつかのパイスライスを描画します。
Dim startAngle As Single
Dim penWidth As Integer = 0
Dim penRGB As Integer = 0
g.FillPie(Brushes.Red, rect, 0, 20F)
g.FillPie(Brushes.Green, rect, 20F, 30F)
g.FillPie(Brushes.Blue, rect, 60F, 12F)
g.FillPie(Brushes.Gold, rect, - 80F, - 20F)
'いくつかの円弧を描画します。
Dim startAngle As Single
For startAngle = 0 To 360 - 40 Step 40
  Dim penColor As Color = Color.FromArgb(penRGB, penRGB, penRGB)
  penWidth = penWidth + 1
  penWidth = penWidth + 1
  penRGB = penRGB + 20
  g.DrawArc(pen, rect, startAngle, 40F)
}
'1つの四角形といくつかのベジェスプライン曲線を描画します。
g.DrawRectangle(Pens.Red, rect);
g.DrawBeziers(Pens.Blue, bezierPoints);
g.DrawString(text, font, Brushes.Black, rect);
```

C#コードの書き方

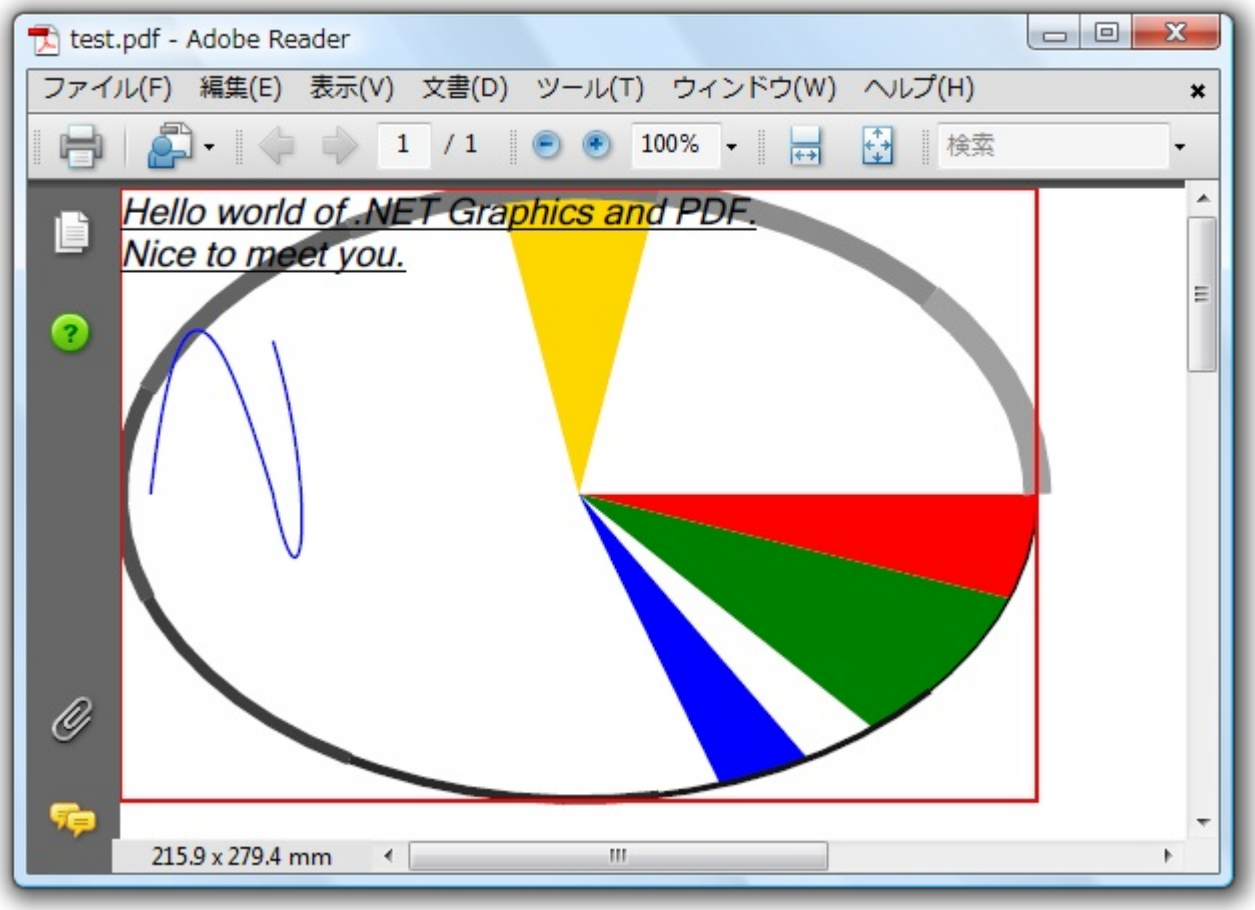
C#

```
// PDF ドキュメントを作成します。
C1.C1Pdf.C1PdfDocument g = new C1.C1Pdf.C1PdfDocument();
// 描画を設定します。
RectangleF rect = new RectangleF(0,0,300,200);
 string text = "Hello world of .NET Graphics and PDF.\r\n" + "Nice to meet you.";
 Font font = new Font("Times New Roman", 12, FontStyle.Italic
 PointF[] bezierPoints = new PointF[]
 {
 new PointF(10f, 100f), new PointF(20f, 10f),
```


Copyright © GrapeCity Inc. All rights reserved.

```
 new PointF(35f, 50f), new PointF(50f, 100f),
 new PointF(60f, 150f), new PointF(65f, 100f),
 new PointF(50f, 50f)
 };
// いくつかのパイスライスを描画します。
int penWidth = 0;
 int penRGB = 0;
 g.FillPie(Brushes.Red, rect, 0, 20f);
 g.FillPie(Brushes.Green, rect, 20f, 30f);
 g.FillPie(Brushes.Blue, rect, 60f, 12f);
 g.FillPie(Brushes.Gold, rect, -80f, -20f);
// いくつかの円弧を描画します。
for (float startAngle = 0; startAngle 360; startAngle += 40)
 {
 Color penColor = Color.FromArgb(penRGB, penRGB, penRGB);
 Pen pen = new Pen(penColor, penWidth++);
 penRGB = penRGB + 20;
 g.DrawArc(pen, rect, startAngle, 40f);
 }
// 1つの四角形といくつかのベジェスプライン曲線を描画します。
g.DrawRectangle(Pens.Red, rect);
 g.DrawBeziers(Pens.Blue, bezierPoints);
 g.DrawString(text, font, Brushes.Black, rect);
```

実行結果は以下のようになります。



Copyright © GrapeCity Inc. All rights reserved.

ページの追加とオーバーレイ描画

これまでの例では、C1PdfDocument オブジェクトを作成したら直ぐにドキュメントに内容の追加を開始しています。これが可能となる理由は、**C1PdfDocument** を作成すると、空白ページがドキュメントに自動的に追加され、任意の種類の内容を受け取る準備ができるからです。

最初のページがいっぱいになったとき、**NewPage** メソッドを使用し、新しいページを追加できます。

デフォルトでは、ドキュメントのすべてのページは同じサイズと向きを持ちます。これらの引数は、C1PdfDocument コンストラクタ内で指定できます。**PaperKind**、**PageSize**、**Landscape** の各プロパティを設定することにより、ページサイズと向きをいつでも変更できます。たとえば以下のコードでは、**PaperKind** 列挙体で定義した各用紙サイズを持つドキュメントを作成します。

Visual Basic コードの書き方

Visual Basic

```
Dim font As New Font("MS PGothic", 9)
Dim sf As New StringFormat()
sf.Alignment = StringAlignment.Center
sf.LineAlignment = StringAlignment.Center
'各用紙サイズを持つページを作成します。
Dim firstPage As Boolean = True
Dim pk As PaperKind
For Each pk In System.Enum.GetValues(GetType(PaperKind))
 'カスタムサイズをスキップします。
  If pk = PaperKind.Custom Then
   GoTo ContinueForEach1
  End If
  '最初のページの後に各ページの新しいページを追加します。
    If Not firstPage Then
   C1PdfDocument1.NewPage()
  End If
   firstPage = False
  '用紙の種類を設定します。
    C1PdfDocument1.PaperKind = pk
   'ページにコンテンツを描画します。
    C1PdfDocument1.DrawString("PaperKind: " + pk.ToString(), font, Brushes.Black,
C1PdfDocument1.PageRectangle, sf)
CoContinueForEach1:
Next pk
```

C# コードの書き方

C#

```
Font font = new Font("MS PGothic", 9);
StringFormat sf = new StringFormat();
sf.Alignment = StringAlignment.Center;
sf.LineAlignment = StringAlignment.Center;
//各用紙サイズを持つページを作成します。
bool firstPage = true;
foreach (PaperKind pk in Enum.GetValues(typeof(PaperKind)))
{
  //カスタムサイズをスキップします。
```


Copyright © GrapeCity Inc. All rights reserved.

```
   if (pk == PaperKind.Custom) continue;
   {
        continue;
   }
  //最初のページの後に各ページの新しいページを追加します。
   if (!firstPage) c1PdfDocument1.NewPage();
   {
        _c1pdf.NewPage();
   }
    firstPage = false;
    // 用紙の種類を設定します。
    c1PdfDocument1.PaperKind = pk;
    // ページにコンテンツを描画します。
    c1PdfDocument1.DrawString("PaperKind: " + pk.ToString(), font, Brushes.Black,
c1PdfDocument1.PageRectangle, sf);
}
```

書き込みはドキュメントに追加した最後のページだけに制限されません。**CurrentPage** プロパティを使用し、書き込むページを選択してから、通常の描画コマンドを通常どおりに使用できます。これは、ドキュメントの描画後、ページに内容を追加するときに便利です。たとえば以下のコードでは、各ページに現在のページ番号とドキュメントのページ総数「n ／ m ページ」を含むフッタを追加します。

Visual Basic コードの書き方

Visual Basic

```
Private Sub AddFooters()
    Dim font As New Font("Tahoma", 7, FontStyle.Bold)
    Dim sf As New StringFormat()
    sf.Alignment = StringAlignment.Center
    Dim page As Integer
    For page = 0 To C1PdfDocument1.Pages.Count - 1
     'ページを選択します。
     C1PdfDocument1.CurrentPage = page
     'フッタを描画する四角形を構築します。
     Dim rect As RectangleF = C1PdfDocument1.PageRectangleF
    rect.Y = rect.Bottom
     'フッタを書きます。
     Dim text As String
     text = String.Format("{0} ／ {1} ページ", page + 1, C1PdfDocument1.Pages.Count)
     pdfC1PdfDocument1.DrawString(text, font, Brushes.Gray, rect, sf)
    Next page
End Sub
```

C# コードの書き方

C#

```
private void AddFooters()
{
    Font font = new Font("Tahoma", 7, FontStyle.Bold);
    StringFormat sf = new StringFormat();
    sf.Alignment = StringAlignment.Center;
    for (int page = 0; page  c1PdfDocument1.Pages.Count; page++)
```


Copyright © GrapeCity Inc. All rights reserved.

```
    {
    //ページを選択します。
     c1PdfDocument1.CurrentPage = page;
    //フッタを描画する四角形を構築します。
     RectangleF rect = c1PdfDocument1.PageRectangle;
     rect.Y = rect.Bottom - 36;
     //フッタを書きます。
      string text = string.Format("{0} ／ {1} ページ", page+1,
c1PdfDocument1.Pages.Count);
     c1PdfDocument1.DrawString(text, font, Brushes.Gray, rect, sf);
  }
}
```

このコードでは **Pages** プロパティを使用してページカウントを取得していることに注意してください。**Pages** プロパティは ArrayList クラスのコレクションであり、ページをカウントおよび一覧にできるメソッドや、指定の位置のページを追加および削除できるメソッドを持ちます。**Pages コレクション**を使用すると、ページをドキュメント内の特定の位置から削除し、それらを任意の位置に再度挿入できます。

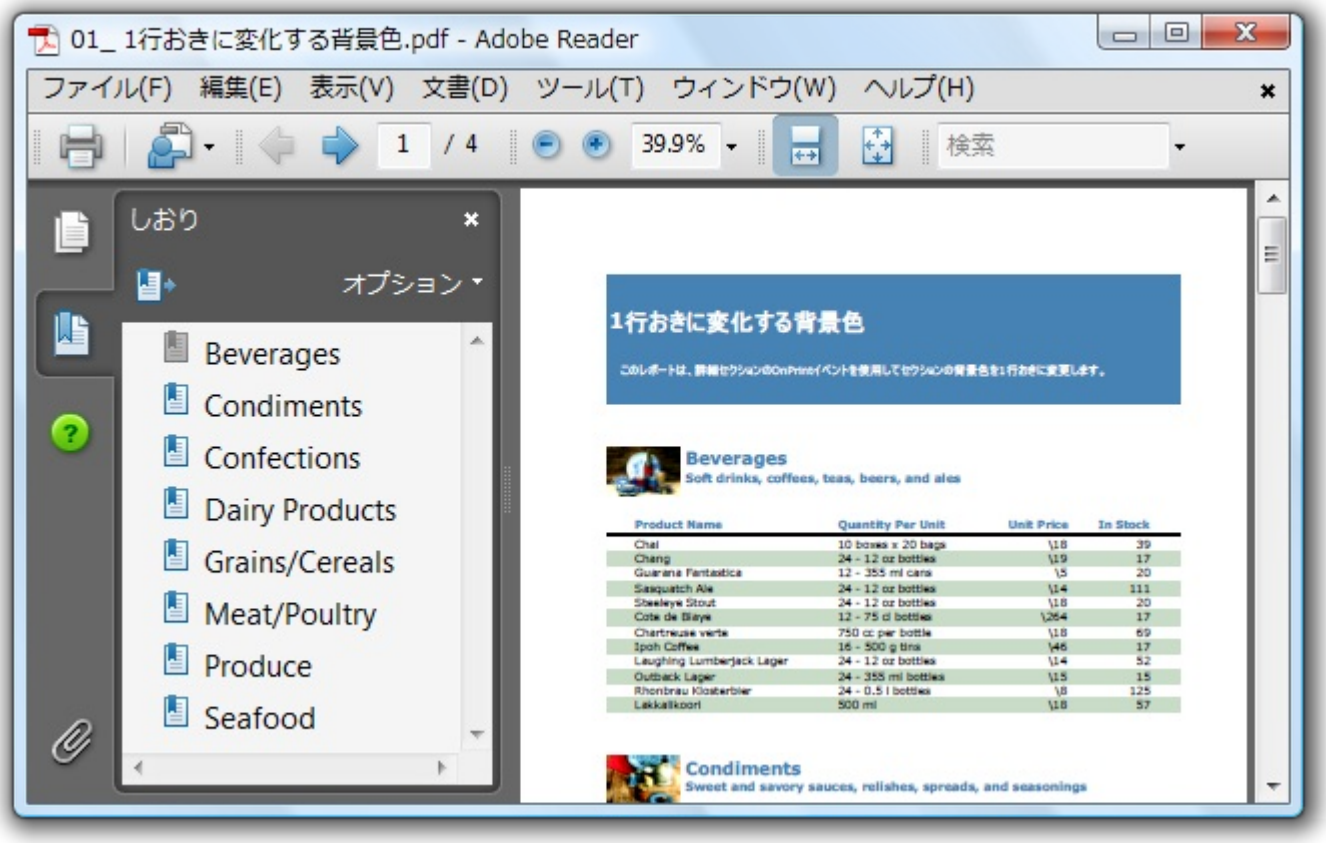
ブックマークやリンクの追加

次のトピックでは PDF ドキュメントへブックマークやハイパーリンクを追加する方法を説明します。

ブックマークの追加

Adobe Reader アプリケーションを使用して PDF ドキュメントを開くと、大部分の長いドキュメントではリーダーの左ペインにアウトライン構造が表示されることがわかります。このアウトラインにより、ドキュメントの構造全体を見渡して容易に特定のトピックを検索できます。以下の図は、アウトラインを持つ PDF ドキュメントの例を示しています。

Copyright © GrapeCity Inc. All rights reserved.



アウトラインのエントリはブックマークと呼ばれ、AddBookmark メソッドを使用して **C1Pdf** ドキュメントに追加できます。**AddBookmark** メソッドは3つの引数を取ります。すなわち、アウトラインエントリのタイトル、アウトラインレベル、および現在のページ上のエントリの y 位置(ページの上部からポイントで測定)です。

たとえば以下のルーチンでは段落をドキュメントに追加し、さらにその段落をレベル0のアウトラインエントリとしてマークします。

Visual Basic コードの書き方

Visual Basic

```
Private Function RenderParagraph(text As String, font As Font, rect As RectangleF,
rectPage As RectangleF, outline As Boolean) As RectangleF
     ' このページに収まらない場合、ページを改行します。
    rect.Height = C1PdfDocument1.MeasureString(text, font, rect.Width).Height
    If rect.Bottom > rectPage.Bottom Then
        C1PdfDocument1.NewPage()
       rect.Y = rectPage.Top
    End If
     ' テキストを描画します。
    C1PdfDocument1.DrawString(text, font, Brushes.Black, rect)
     ' アウトラインに見出しを追加します。
    If outline Then
        C1PdfDocument1.DrawLine(Pens.Black, rect.X, rect.Y, rect.Right, rect.Y)
        C1PdfDocument1.AddBookmark(text, 0, rect.Y)
    End If
     ' 次回の四角形を更新します。
    rect.Offset(0, rect.Height)
    Return rect
End Function
```


Copyright © GrapeCity Inc. All rights reserved.

C# コードの書き方

C#

```
private RectangleF RenderParagraph(string text, Font font, RectangleF rect,
RectangleF rectPage, bool outline)
{
    // このページに収まらない場合、ページを改行します。
    rect.Height = c1PdfDocument1.MeasureString(text, font, rect.Width).Height;
    if (rect.Bottom > rectPage.Bottom)
    {
        c1PdfDocument1.NewPage();
        rect.Y = rectPage.Top;
    }
     // テキストを描画します。
    c1PdfDocument1.DrawString(text, font, Brushes.Black, rect);
     // アウトラインに見出しを追加します。
    if (outline)
    {
        c1PdfDocument1.DrawLine(Pens.Black, rect.X, rect.Y, rect.Right, rect.Y);
        c1PdfDocument1.AddBookmark(text, 0, rect.Y);
    }
     // 次回の四角形を更新します。
    rect.Offset(0, rect.Height);

    return rect;
}
```

メモ:ブックマークの子は初期化のとき表示されるかどうかを指定するためのブール値やドキュメントのターゲット名を渡すには、AddBookmark()メソッドに対して上記に記載したの他も2つのオーバーロードを使用できます。

リンクの追加

PDF 仕様により、複数のタイプの注釈をドキュメントに追加できます。多くの場合、注釈は強調表示やメモとして手動で追加されますが、プログラムでも追加できます。**C1PdfDocument** はハイパーリンク、ハイパーリンクターゲット、および添付ファイルを PDF ドキュメントに追加するメソッドを提供します。

ドキュメントにハイパーリンクを追加するには、**AddLink** メソッドを使用します。AddLink メソッドは、2つの引数 URL と RectangleF を取ります。URL はテキストを指定し、RectangleF はリンクとして機能する現在のページ上の領域を指定します。

AddLink メソッドは可視の内容をページに追加しないため、通常、AddLink と共に別のコマンドを使用して、ユーザーに表示できるテキストや画像を指定する必要があることに注意してください。たとえば以下のコードでは、「GrapeCity Web サイト」というテキストと、GrapeCity Web サイトを開くリンクを追加します。

Visual Basic コードの書き方

Visual Basic

```
Dim rect As New RectangleF(50, 50, 100, 15)
Dim font As New Font("MS UI Gothic", 10, FontStyle.Underline)
_c1PdfDocument1.AddLink("https://www.grapecity.co.jp/developer/", rect)
_c1PdfDocument1.DrawString("GrapeCity Web サイト", font, Brushes.Blue, rect)
```

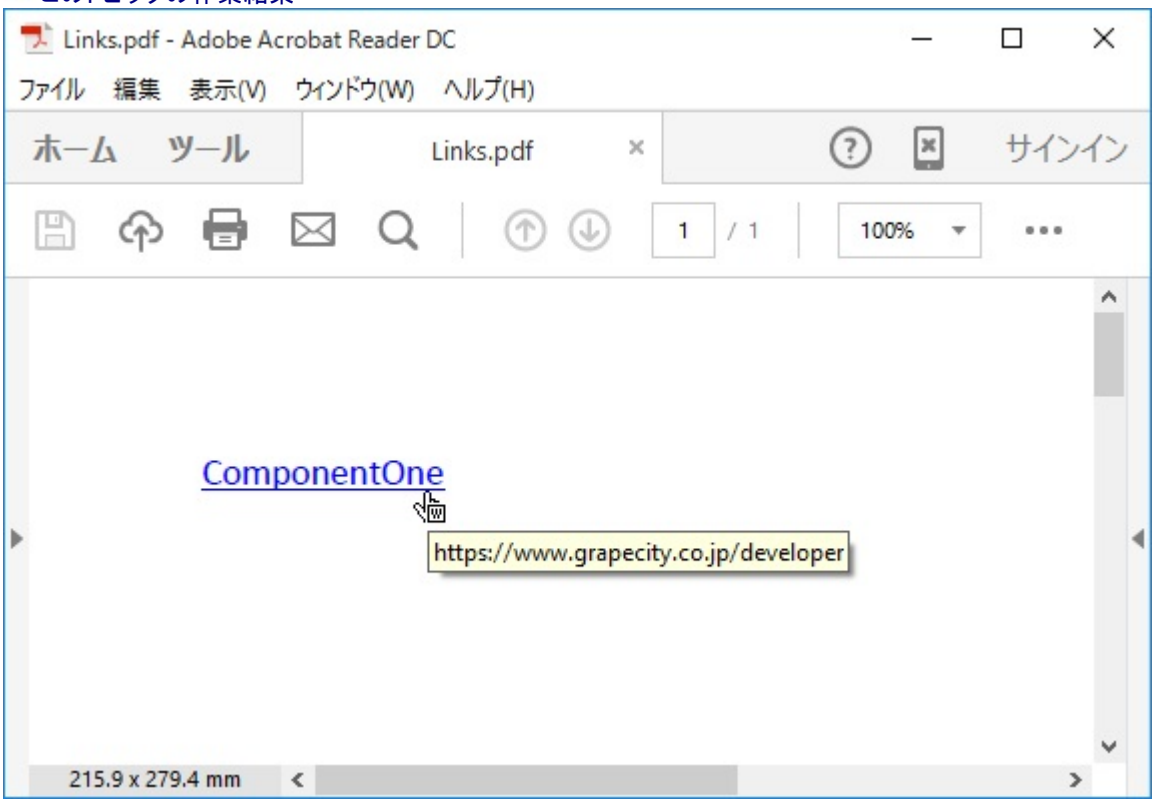
C# コードの書き方

Copyright © GrapeCity Inc. All rights reserved.

C#

```
RectangleF rect = new RectangleF(50, 50, 100, 15);
Font font = new Font("MS UI Gothic", 10, FontStyle.Underline);
_c1PdfDocument1.AddLink("https://www.grapecity.co.jp/developer/", rect);
_c1PdfDocument1.DrawString("GrapeCity Web サイト", font, Brushes.Blue, rect);
```

このトピックの作業結果



また、Web サイトだけでなく、同じ PDF ドキュメント内の別の位置を開くローカルリンクを追加することもできます。このタイプのリンクは、ドキュメント内に目次やインデックスなどのタイプの相互参照を実装する場合に便利です.

ローカルリンクは以下の2つのことを除いて通常のハイパーリンクと同じです

- *URL* 引数の先頭を" #" にする必要があります。
- **AddTarget**メソッドを使用し、リンクターゲットの位置を指定する必要があります。AddTarget メソッドは、AddLink と同じ引数、すなわちターゲットの名前と、ユーザーによるリンクのクリック時に表示するページ上の領域をマークする四角形を指定するテキストを取ります。

ファイルの添付

PDF ファイルに添付ファイルを追加する機能は便利なことがよくあります。添付ファイルは、詳細情報のスプレッドシートや、マルチメディア(ムービー、サウンド、サンプルコードなど)といったあらゆる種類のファイルを含むことができます。

ファイル添付を **C1Pdf** ドキュメントに追加するのは簡単です。行う操作は、**AddAttachment** メソッドを呼び出し、添付するファイル、添付ファイルを含むページの領域、およびオプションとして、添付ファイルの外観を指定するだけです。

添付ファイルはアイコンとして表示されます(アイコンは、AttachmentIconEnum 列挙体内の4つの定義済みのアイコンの中から選択でき、アイコンの色を選択することもできます)。ユーザーが添付ファイルトの上にマウスを移動させると、ファイル名が表示され、カーソルが切り替わって使用可能な添付ファイルが存在することを示します。これでユーザーは、マウスを右クリックして添付ファイルを開いたり、添付ファイルをファイルに保存したりできます。

たとえば以下のコードでは、指定したディレクトリ内の全ファイルを PDF ドキュメントに添付します.

Visual Basic コードの書き方

Copyright © GrapeCity Inc. All rights reserved.

Visual Basic

```
Dim rect As New RectangleF(100, 100, 60, 10)
Dim font As New Font("MS UI Gothic", 9)
 'いくつかのファイルを添付します。
Dim path As String = "c:\temp\files"
Dim file As String
For Each file In System.IO.Directory.GetFiles(path)
    Dim width As Single = rect.Width
    rect.Width = rect.Height
    _c1PdfDocument1.FillRectangle(Brushes.Gray, rect)
    _c1PdfDocument1.AddAttachment(file, rect)
    rect.Width = width
    rect.X += rect.Height
    _c1PdfDocument1.DrawString(System.IO.Path.GetFileName(file), font, Brushes.Black,
rect)
    rect.X -= rect.Height
    rect.Y += 2 * rect.Height
Next file
```

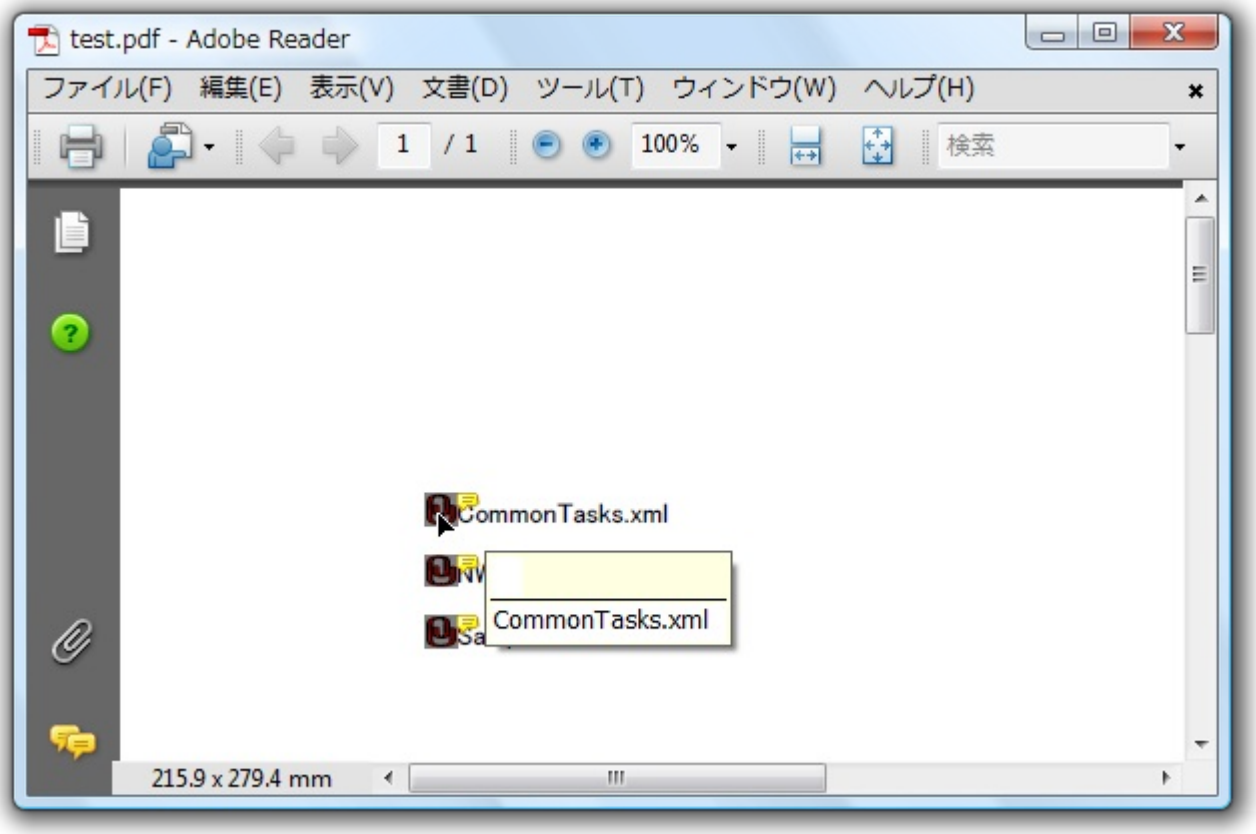
C# コードの書き方

C#

```
RectangleF rect = new RectangleF(100, 100, 60, 10);
Font font = new Font("MS UI Gothic", 9);
 //いくつかのファイルを添付します。
string path = @"c:\temp\files";
string[] files = System.IO.Directory.GetFiles(path);
foreach (string file in files)
{
    float width = rect.Width;
    rect.Width = rect.Height;
    _c1PdfDocument1.FillRectangle(Brushes.Gray, rect)
    _c1PdfDocument1.AddAttachment(file, rect);
    rect.Width = width;
    rect.X += rect.Height;
    _c1PdfDocument1.DrawString(System.IO.Path.GetFileName(file), font, Brushes.Black,
rect);
    rect.X -= rect.Height;
    rect.Y += 2 * rect.Height;
}
```

このトピックの作業結果

Copyright © GrapeCity Inc. All rights reserved.



添付ファイルはアイコン（**AttachmentIconEnum** 内の4つの定義済みアイコンから選択できアイコンの色も選択可能）として表示されます。ユーザーがマウスを添付ファイル上に移動すると、ファイル名が表示されカーソルが変化して、使用可能な添付ファイルが存在することを示します。ユーザーは添付ファイル名を右クリックし、添付ファイルを開くか、それをファイルに保存できます.

セキュリティと制限の適用

デフォルトでは、誰でも PDF ファイルを開いたり、コピーしたり、印刷したり、編集したりできます。ただし、PDF ドキュメントが機密情報を含む場合は、PDF ドキュメントを暗号化して権限を持ったユーザーのみがアクセスできるように設定できます。

ドキュメントの所有者とその他のすべてのユーザー用として別個のパスワードがあります。ユーザーのアクセスを選択的に制限し、ドキュメントの閲覧、印刷、編集などの特定の操作のみを許可できます。

C1Pdf は、**PdfSecurity** オブジェクトを返す Security プロパティを提供します。このオブジェクトは、「所有者パスワード」（ドキュメントのパスワードと権限を変更するために必要）と「ユーザーパスワード」（ドキュメントを開くために必要）を指定できるプロパティを持っています。さらに、**PdfSecurity**オブジェクトでは、通常ユーザーが持つことができる権限を指定できます。

たとえば、ユーザーにドキュメントの閲覧だけを許可して、印刷や編集は許可しないように指定できます。

Visual Basic コードの書き方

Visual Basic

```
 ' ドキュメントを通常どおりに作成します。
CreateDoc()
  ' パスワードを設定します。
_c1pdf.Security.OwnerPassword = "2mds%dffgd"
_c1pdf.Security.UserPassword = "anyone"
_c1pdf.Security.AllowEditAnnotations = False
_c1pdf.Security.AllowEditContent = False
_c1pdf.Security.AllowPrint = False
```


Copyright © GrapeCity Inc. All rights reserved.

```
  ' 暗号化したドキュメントを保存します。
_c1pdf.Save("")
```

C# コードの書き方

C#

```
 // ドキュメントを通常どおりに作成します。
CreateDoc();
 // パスワードを設定します。
_c1pdf.Security.OwnerPassword = "2mds%dffgd";
_c1pdf.Security.UserPassword = "anyone";
_c1pdf.Security.AllowEditAnnotations = false;
_c1pdf.Security.AllowEditContent = false;
_c1pdf.Security.AllowPrint = false;
 // 暗号化したドキュメントを保存します。
_c1pdf.Save(@"c:\reports\secure.pdf");
```

メモ:ドキュメント制限は、PDF ドキュメントの[ドキュメントプロパティ]ダイアログボックスに表示されます。

権限を指定し、所有者パスワードのみを設定してユーザーパスワードは空のままにしておけることに注意してください。この場合、誰でもドキュメントを開くことができますが、権限を変更できるのは所有者のみとなります。

PDF で使用している暗号化スキームは一般的なもので、100% 安全ではないことに注意してください。提供されたセキュリティは、大部分の通常の攻撃からドキュメントを保護するのに適していますが、データが本当に機密情報である場合、PDF 暗号化のみに頼るべきではありません。

メタファイル画像の描画

C1Pdf では、非常に簡単にドキュメントを作成できます。その主な理由は、オブジェクトモデルがよく知られた .NET の **Graphics** モデルを模倣しているためです。ただし、**Graphics** クラスで使用できるすべてのメソッドが **C1Pdf** でも使用できるわけではありません。さらに、**Graphics** オブジェクトを描画する既存のコードがあり、メソッドが非常に似ている場合でも書き直したくないときがあります。

こうした場合には、**Metafile** オブジェクトから派生した Graphics オブジェクトに描画して、作成した **Metafile** を**DrawImage**メソッドを使用して **C1Pdf** に描画することにより、既存の .NET コードを再利用できます。この方法では、画像または PDF ドキュメントとして作成する任意のグラフィックスを提供できます。

たとえばアプリケーションが、各ページを **Graphics** オブジェクトに描画する **PrintDocument** パターンを使用し、ドキュメントを生成する場合を想定します。その場合、同じメソッドを使用し、1ページに1つずつメタファイルのコレクションを作成し、以下のコードを使用してその一覧を PDF ドキュメントに変換できます。

Visual Basic コードの書き方

Visual Basic

```
 ' ドキュメントをメタファイルの一覧として、1ページに1つずつ取得します。
Dim pages As ArrayList = GetMetafiles()
 ' 各ページをループし、PDF ドキュメントを作成します。
_c1PdfDocument1.Clear()
Dim i As Integer
for i = 0 i <= pages.Count
   ' i 番目のページを取得します。
    Dim page As Metafile = CType(Metafile.FromFile(pages[i]), Metafile)
    If Not (page Is Nothing) Then
         ' ページサイズを計算します。
        Dim sz As SizeF = page.PhysicalDimension
```


Copyright © GrapeCity Inc. All rights reserved.

```
        sz.Width = Math.Round(sz.Width * 72.0F / 2540.0F, 2)
        sz.Height = Math.Round(sz.Height * 72.0F / 2540.0F, 2)
         ' ページを追加し、サイズを設定します。
        If i > 0 Then C1PdfDocument1.NewPage()
            _c1PdfDocument1.PageSize = sz
        End If
        _c1pdf.PageSize = sz
         ' ページを PDF ドキュメントに描画します。
        _c1PdfDocument1.DrawImage(page, C1PdfDocument1.PageRectangle)
    End If
Next
 ' ファイルに保存します。
_c1PdfDocument1.Save("c:\temp\mydoc.pdf")
```

C# コードの書き方

C#

```
// ドキュメントをメタファイルの一覧として、1ページに1つずつ取得します。
ArrayList pages = GetMetafiles();
 // 各ページをループし、PDF ドキュメントを作成します。_c1pdf.Clear();
c1PdfDocument1.Clear();
for (int i = 0; i < pages.Count; i++)
{
     // i 番目のページを取得します。
    Metafile page = (Metafile)Metafile.FromFile(pages[i]);
    if (page == null)
    {
        continue;
    }
     // ページサイズを計算します。
    SizeF sz = page.PhysicalDimension;
    sz.Width    = (float)Math.Round(sz.Width    * 72f / 2540f, 2);
    sz.Height = (float)Math.Round(sz.Height * 72f / 2540f, 2);
     // ページを追加し、サイズを設定します。
    if (i > 0)
    _c1PdfDocument1.NewPage();
     // ページを PDF ドキュメントに描画します。
    _c1PdfDocument1.DrawImage(page, c1PdfDocument1.PageRectangle);
}
 // ファイルに保存します。
_c1PdfDocument1.Save(@"c:\temp\mydoc.pdf");
```

コードでは、一覧の各メタファイルを取得し、そのサイズをポイントで計算し（各ページは異なるサイズを持つことができます）、メタファイルをページに描画しています。メタファイルは、レポートエンジン、描画またはグラフ作成プログラム、またはメタファイル画像を作成できるアプリケーションによって生成できます。

なお、**C1PDF**では、EMF+ メタファイルに対応しています。According to Microsoft、EMF+ は EMF の拡張であり、GDI+ レコードを格納することができます。以前、すべてのメタファイルが EFM に変換されていましたが、EMF+ ファイルは変換されないので、変換によってコンテキストが失われることはありません。

複雑な RTF テキストの描画

Copyright © GrapeCity Inc. All rights reserved.

PDF for .NET のもう1つの強力な機能は、**DrawStringRtf** メソッドを使用してリッチテキストフォーマット(RTF)テキストを表示する機能です。主に RTF を作成するツールが非常に多く、RTF によるテンプレートの作成、アプリケーションのカスタマイズ、RTF 文字列の表示が容易なので、この機能は非常に便利です。これにより、十分に書式設定された実用的なドキュメントを、コードを記述することなく簡単に作成できるようになります。

たとえば、以下のコードはワードパッドによって作成された RTF テンプレートを使用しています。このコードは、ディレクトリをスキャンし、テンプレートに基づいて RTF 文字列をビルドし、DrawStringRtf メソッドを使用して RTF を表示します。

Visual Basic コードの書き方

Visual Basic

```
' RTF テンプレートを取得します。
Dim rtfHdr As String = "" & _
   "{\rtf1\ansi\ansicpg1252\deff0\deflang1033" &
   "{\fonttbl{\f0\fswiss\fcharset0 " & _
   "Arial;}{\f1\froman\fprq2\fcharset0 Book Antiqua;}}" &  _
   "{\colortbl ;\red0\green0\blue0;}" & _
   "\viewkind4\uc1\pard\f0\fs20\par" & _
   "\pard\tx1440\tx2880\tx4320\tx5760\cf1\b\f1\fs24 " & _
   "Directory Report created on <<TODAY>>\par" & _
   "\ul\par Name\tab Extension\tab Size\tab Date\tab " & _
   "Attributes\par"
Dim rtfEntry As String = "" & _
   "\cf0\ulnone\b0\f0\fs16 <<NAME>>\tab " & __
   "<<EXT>>\tab <<SIZE>>\tab <<DATE>>\tab <<ATTS>>\par"
   ' RTF 文字列をビルドします。
Dim sb As New StringBuilder()
sb.Append(rtfHdr.Replace("<<TODAY>>", DateTime.Today.ToShortDateString()))
For Each file in Directory.GetFiles(@"c:\winnt", "*.bmp")
For Each file in Directory.GetFiles(@"c:\winnt", "*.bmp")
   Dim s As String = rtfEntry
   Dim fi As FileInfo = New FileInfo(file)
   s = s.Replace("<<NAME>>", Path.GetFileNameWithoutExtension(file));
   s = s.Replace("<<EXT>>",  fi.Extension);
   s = s.Replace("<<SIZE>>", string.Format("{0:#,##0}", fi.Length));
   ss = s.Replace("<<DATE>>", fi.LastWriteTime.ToShortDateString());ring())
   s = s.Replace("<<ATTS>>", fi.Attributes.ToString())
   sb.Append(s)
Next
sb.Append("}")
 ' RTF 文字列を表示します。
Dim _c1pdf As New C1PdfDocument()
Dim rect As RectangleF = _c1pdf.PageRectangle
rect.Inflate(-72, -72)
_c1pdf.DrawStringRtf(sb.ToString(), Font, Brushes.Black, rect)
 ' 保存します。
_c1pdf.Save("c:\temp\dir.pdf")
```

C# コードの書き方

C#

```
// RTF テンプレートを取得します。
string rtfHdr = @"{\rtf1\ansi\ansicpg1252\deff0\deflang1033" +
```

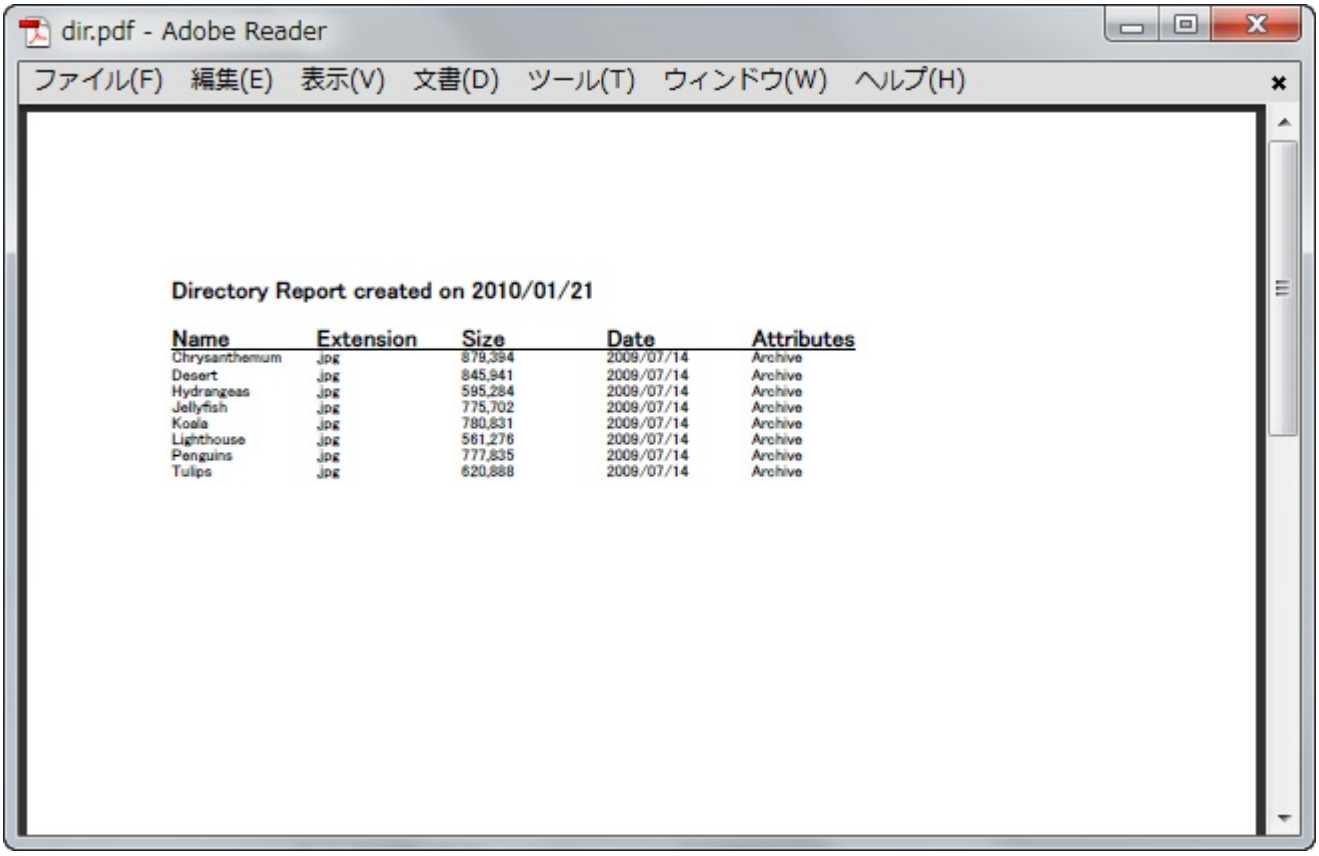

Copyright © GrapeCity Inc. All rights reserved.

```
@"{\fonttbl{\f0\fswiss\fcharset0 " +
@"Arial;}{\f1\froman\fprq2\fcharset0 Book Antiqua;}}" +
@"{\colortbl ;\red0\green0\blue0;}" +
@"\viewkind4\uc1\pard\f0\fs20\par" +
@"\pard\tx1440\tx2880\tx4320\tx5760\cf1\b\f1\fs24 " +
@"Directory Report created on <<TODAY>>\par" +
@"\ul\par Name\tab Extension\tab Size\tab Date\tab " +
@"Attributes\par";
string rtfEntry = @"\cf0\ulnone\b0\f0\fs16 <<NAME>>\tab " +
@"<<EXT>>\tab <<SIZE>>\tab <<DATE>>\tab <<ATTS>>\par";
// RTF 文字列をビルドします。
StringBuilder sb = new StringBuilder();
sb.Append(rtfHdr.Replace("<<TODAY>>", DateTime.Today.ToShortDateString()));
foreach (string file in Directory.GetFiles(@"c:\winnt", "*.bmp"))
{
string s = rtfEntry;
FileInfo fi = new FileInfo(file);
s = s.Replace("<<NAME>>", Path.GetFileNameWithoutExtension(file));
s = s.Replace("<<EXT>>", fi.Extension);
s = s.Replace("<<SIZE>>", string.Format("{0:#,##0}", fi.Length));
s = s.Replace("<<DATE>>", fi.LastWriteTime.ToShortDateString());
s = s.Replace("<<ATTS>>", fi.Attributes.ToString());
sb.Append(s);
}
sb.Append("}");
// RTF 文字列を表示します。
C1.C1Pdf.C1PdfDocument _c1pdf = new C1.C1Pdf.C1PdfDocument();
RectangleF rect = _c1pdf.PageRectangle;
rect.Inflate(-72, -72);
_c1pdf.DrawStringRtf(sb.ToString(), Font, Brushes.Black, rect);
// 保存します。
_c1pdf.Save(@"c:\temp\dir.pdf");
```

コードはとても簡単です。唯一複雑な部分は RTF 定義ですが、これはワードパッドで作成された RTF ファイルからコピー＆ペーストされました。RTF テンプレートの文字列には、後で実際のコンテンツに置き換えられるタグ（「<<NAME>>」など）が含まれています。

RTF 文字列がビルドされると、DrawStringRtf コマンドを使用して PDF ドキュメントに表示されます。結果は次のようになります。

Copyright © GrapeCity Inc. All rights reserved.



Copyright © GrapeCity Inc. All rights reserved.

タスク別ヘルプ

タスク別ヘルプは、ユーザーが .NET のプログラミングに精通しており、コントロールの一般的な使用方法を理解していることを前提にしています。

ヘルプに示される手順に従って作業を進めるだけで、**PDF for .NET** のさまざまな機能を具体的に紹介するプロジェクトを作成しながら、**PDF for .NET** の主要な機能と特長を理解できます。

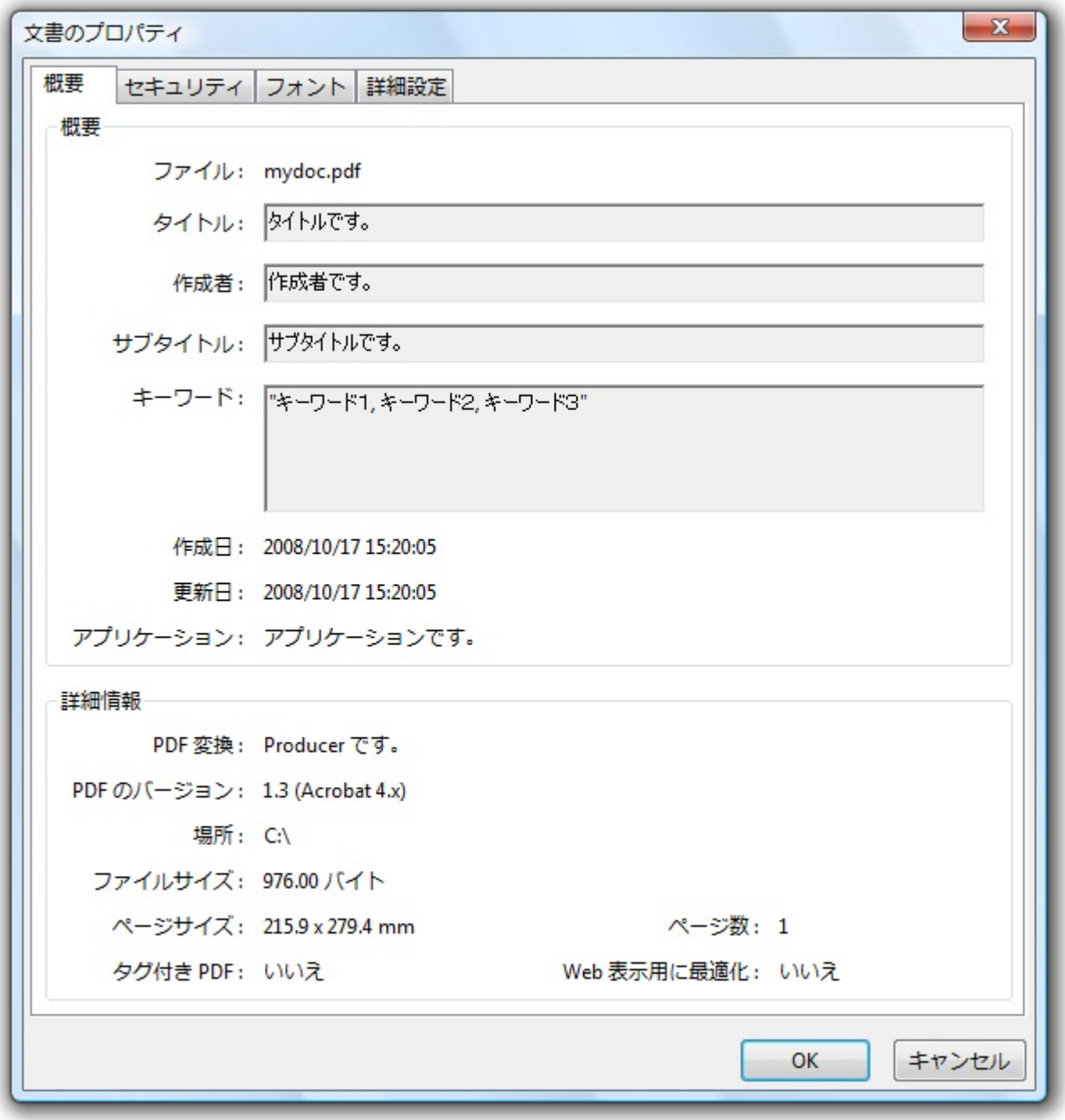
このセクションの各トピックは、**PDF for .NET** 製品を使用して特定のタスクを実行するためのソリューションを提供します。ヘルプに記載されている手順に従って、**PDF for .NET**の機能を利用することができるようになります。

また、タスクベースのヘルプトピックは、新しいプロジェクトが既に作成されていることを前提としています。

ドキュメントプロパティを設定する

PDF ドキュメントのドキュメントプロパティを設定すると、タイトル、作成者、サブタイトル、キーワード、アプリケーション、PDF Producer をカスタマイズできます。これらのドキュメントプロパティを設定するには、デザイナまたはコードで **Title**, **Author**, **Subject**, **Keywords**, **Creator**, **Producer** の各プロパティに情報を入力します。

Copyright © GrapeCity Inc. All rights reserved.

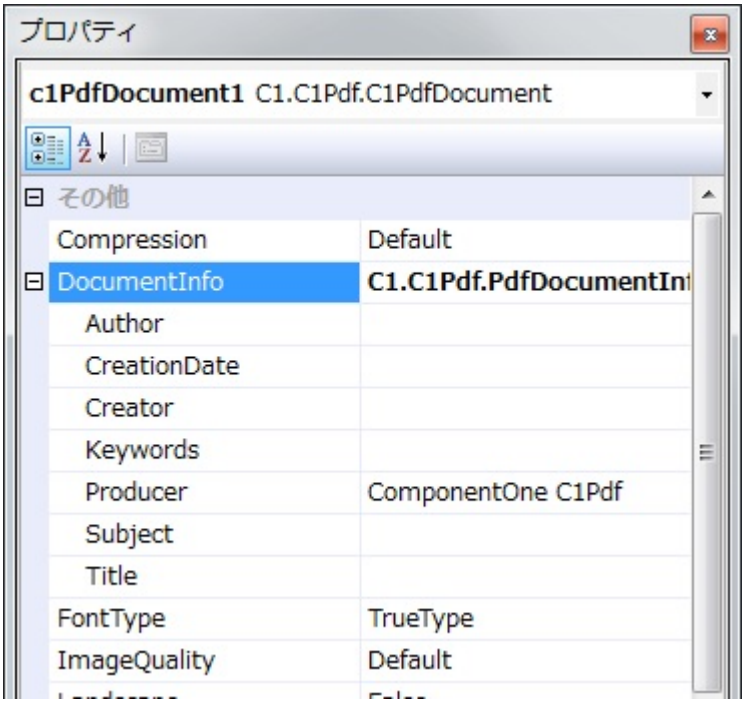


メモ: [説明]の下にある[アプリケーション]を設定するには、Creator プロパティを使用します。ドキュメントプロパティは、PDF ドキュメントの[文書のプロパティ]ダイアログボックスに表示されます。

デザイナの場合

プロパティウィンドウで、**DocumentInfo** プロパティを探し、プロパティノードを展開します。

Copyright © GrapeCity Inc. All rights reserved.



プロパティの情報を入力し、プロパティの設定が完了したら[Enter]キーを押します。

コードの場合

次のコードを **Form_Load** イベントに追加します。

1. **C1PdfDocument**オブジェクトを作成します。

   Visual Basic コードの書き方

   Visual Basic
   ```
   ' C1PdfDocument オブジェクトを作成します。
   Dim pdf As New C1.C1Pdf.C1PdfDocument()
   ```

   C# コードの書き方

   C#
   ```
   // C1PdfDocument オブジェクトを作成します。
   C1.C1Pdf.C1PdfDocument pdf = new C1.C1Pdf.C1PdfDocument();
   ```

2. ドキュメントプロパティを設定します

   Title プロパティを設定するには、次のコードを追加します。

   Visual Basic コードの書き方

   Visual Basic
   ```
   pdf.DocumentInfo.Title = "PDF Creation"
   ```

   C# コードの書き方

   C#
   ```
   pdf.DocumentInfo.Title = "PDF Creation";
   ```


Copyright © GrapeCity Inc. All rights reserved.

Author プロパティを設定するには、次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
pdf.DocumentInfo.Author = "I. Author"
```

C# コードの書き方

```
C#
pdf.DocumentInfo.Author = "I. Author"
```

Subject プロパティを設定するには、次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
pdf.DocumentInfo.Subject = "PDF"
```

C# コードの書き方

```
C#
pdf.DocumentInfo.Subject = "PDF";
```

Keywords プロパティを設定するには、次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
pdf.DocumentInfo.Keywords = "PDF, creating PDF, PDF files, eBooks"
```

C# コードの書き方

```
C#
pdf.DocumentInfo.Keywords = "PDF, creating PDF, PDF files, eBooks";
```

Creator プロパティを設定するには、次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
pdf.DocumentInfo.Creator = "My New Application"
```

C# コードの書き方

```
C#
pdf.DocumentInfo.Creator = "My New Application";
```

Producer プロパティを設定するには、次のコードを追加します。デフォルトは、「C1Pdf」です。

Visual Basic コードの書き方

Copyright © GrapeCity Inc. All rights reserved.

```
Visual Basic
pdf.DocumentInfo.Producer = "PDF Creation Industries"
```

C# コードの書き方

```
C#
pdf.DocumentInfo.Producer = "PDF Creation Industries";
```

3. ドキュメントをファイルに保存します。

Visual Basic コードの書き方

```
Visual Basic
' ドキュメントをファイルに保存します。
pdf.Save("c:\mydoc.pdf")
```

C# コードの書き方

```
C#
// ドキュメントをファイルに保存します。
pdf.Save(@"c:\mydoc.pdf");
```

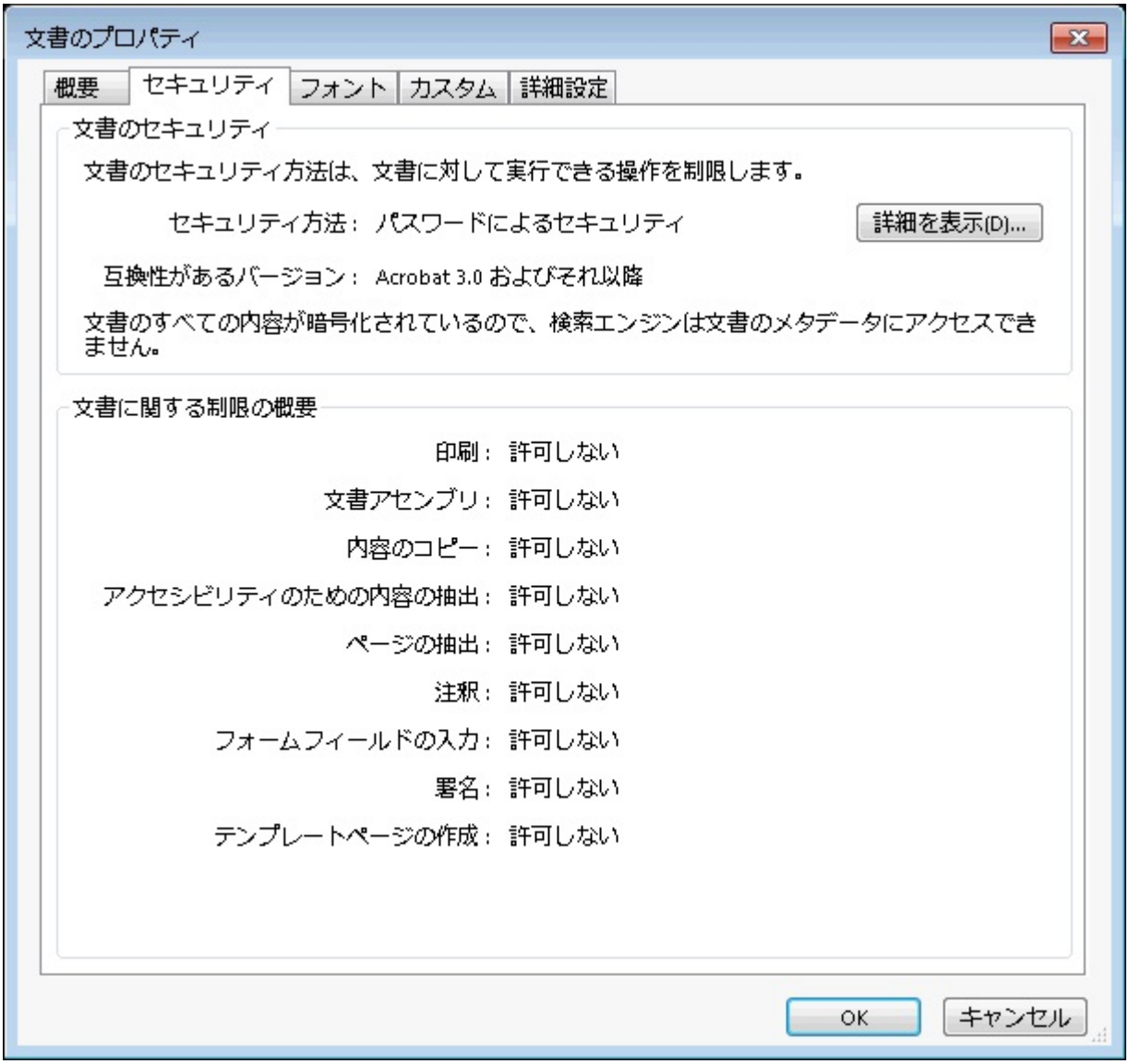
ドキュメントに制限を設定する

ドキュメント制限の設定では、ユーザーがコンテンツのコピー、注釈の編集、コンテンツの編集、または PDF ファイルの印刷を実行できるかどうかをカスタマイズできます。ドキュメント制限を有効にするには、デザイナまたはコードのいずれかで以下のプロパティを **False** に設定します。

- **AllowCopyContent**
- **AllowEditAnnotations**
- **AllowEditContent**
- **AllowPrint**

パスワードとアクセス権の設定についての詳細は、「セキュリティと制限の適用」を参考してください。

Copyright © GrapeCity Inc. All rights reserved.

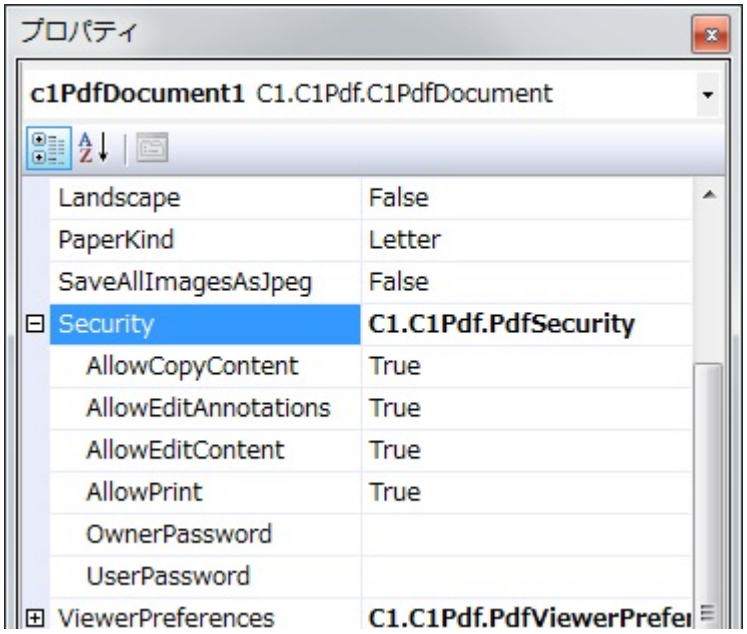


注意:ドキュメント制限は、PDF ドキュメントの[**ドキュメントプロパティ**]ダイアログボックスに表示されます。

デザイナの場

1. プロパティウィンドウで、**Security** プロパティを探し、プロパティノードを展開します。

Copyright © GrapeCity Inc. All rights reserved.



2. パスワードを入力し、終了したら[Enter]キーを押して、**OwnerPassword** プロパティを設定します。
3. AllowCopyContent、AllowEditAnnotations、AllowEditContent、および AllowPrint プロパティを **False** に設定して、コピー、編集、および印刷を不許可にします。

コードの場合

次のコードを Form_Load イベントに追加します。.

1. **C1PdfDocument** オブジェクトを作成します。

Visual Basic コードの書き方

```
Visual Basic
' C1PdfDocument オブジェクトを作成します。
Dim pdf As New C1.C1Pdf.C1PdfDocument()
```

C# コードの書き方

```
C#
// C1PdfDocument オブジェクトを作成します。
C1.C1Pdf.C1PdfDocument pdf = new C1.C1Pdf.C1PdfDocument();
```

2. 次のコードを追加し、OwnerPassword プロパティを設定します。

Visual Basic コードの書き方

```
Visual Basic
pdf.Security.OwnerPassword = "Password"
```

C# コードの書き方

```
C#
pdf.Security.OwnerPassword = "Password";
```


Copyright © GrapeCity Inc. All rights reserved.

3. 次のコードを追加し、AllowCopyContent、AllowEditAnnotations、AllowEditContent、および AllowPrint プロパティを **False** に設定して、コピー、編集、および印刷を不許可にします。

Visual Basic コードの書き方

```
Visual Basic
pdf.Security.AllowCopyContent = False
pdf.Security.AllowEditAnnotations = False
pdf.Security.AllowEditContent = False
pdf.Security.AllowPrint = False
```

C# コードの書き方

```
C#
pdf.Security.AllowCopyContent = false;
pdf.Security.AllowEditAnnotations = false;
pdf.Security.AllowEditContent = false;
pdf.Security.AllowPrint = false;
```

4. ドキュメントをファイルに保存します。

Visual Basic コードの書き方

```
Visual Basic
' ドキュメントをファイルに保存します。
pdf.Save("c:\mydoc.pdf")
```

C# コードの書き方

```
C#
// ドキュメントをファイルに保存します。
pdf.Save(@"c:\mydoc.pdf");
```

ページサイズを設定する

ページサイズを設定するには、**PaperKind** プロパティを使用します。このプロパティは、デザイナまたはコードのいずれかで設定できます。

デザイナの場合

プロパティウィンドウで PaperKind プロパティを探し、新しい用紙サイズを選択します。

コードの場合

次のコードを **Form_Load** イベントに追加し、PaperKind プロパティをリーガルサイズ用紙に設定します。

Visual Basic コードの書き方

```
Visual Basic
```


Copyright © GrapeCity Inc. All rights reserved.

```
' Legal サイズ用紙を使用します。
C1PdfDocument1.PaperKind = Printing.PaperKind.Legal
 ' ドキュメントをファイルに保存します。
C1PdfDocument1.Save("c:\mydoc.pdf")
 ' Save the document to a file.
pdf.Save("c:\mydoc.pdf")
```

C# コードの書き方

C#

```
// Legal サイズ用紙を使用します。
c1PdfDocument1.PaperKind = Printing.PaperKind.Legal;
 // ドキュメントをファイルに保存します。
c1PdfDocument1.Save(@"c:\mydoc.pdf");
 // Save the document to a file.
pdf.Save(@"c:\mydoc.pdf");
```

ページの向きを設定する

用紙の方向を設定するには、**Landscape** プロパティを使用します。このプロパティは、デザイナまたはコードのいずれかで設定できます。

デザイナの場合

プロパティウィンドウで Landscape プロパティを探し、**True** に設定します .

コードの場合

次のコードを **Form_Load** イベントに追加し、Landscape プロパティを設定します。

Visual Basic コードの書き方

Visual Basic

```
' 横の用紙方向を使用します。
C1PdfDocument1.Landscape = True
 ' ドキュメントをファイルに保存します。
C1PdfDocument1.Save("c:\mydoc.pdf")
```

C# コードの書き方

C#

```
// 横の用紙方向を使用します。
c1PdfDocument1.Landscape = true;
// ドキュメントをファイルに保存します。
c1PdfDocument1.Save(@"c:\mydoc.pdf");
```


Copyright © GrapeCity Inc. All rights reserved.

画質を設定する

PDF ドキュメントの画像品質を設定するには、ImageQuality プロパティを使用します。このプロパティは、デザイナまたはコードのいずれかで設定できます。

デザイナの場合

プロパティウィンドウで **ImageQuality** プロパティを探し、それを以下の **ImageQualityEnum** 設定のいずれかに設定します。Low、Medium、Default、または High。

コードの場合

次のコードを Form_Load イベントに追加し、**ImageQuality** プロパティを高品質画像に設定します。

Visual Basic コードの書き方

Visual Basic

```
' C1PdfDocumentオブジェクトを作成します。
Dim pdf As New C1.C1Pdf.C1PdfDocument()
' 高品質画像を使用します。
pdf.ImageQuality = C1.C1Pdf.ImageQualityEnum.High
' ドキュメントをファイルに保存します。
pdf.Save("c:\mydoc.pdf")
```

C# コードの書き方

C#

```
// C1PdfDocumentオブジェクトを作成します。
C1.C1Pdf.C1PdfDocument pdf = new C1.C1Pdf.C1PdfDocument();
// 高品質画像を使用します。
pdf.ImageQuality = C1.C1Pdf.ImageQualityEnum.High;
// ドキュメントをファイルに保存します。
pdf.Save(@"c:\mydoc.pdf");
```

PDF ファイルを印刷する

次のコードを使用すると、**C1Pdf** に印刷機能を追加できます。この例では、**Button1_Click** イベントにコードを追加します。

Visual Basic コードの書き方

Visual Basic

```
Process.Start("C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe", "/p /h " &
"c:\HelloWorld.pdf")
```

C# コードの書き方

C#

```
Process.Start(@"C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe", "/p /h " &
```


Copyright © GrapeCity Inc. All rights reserved.

```
@"c:\HelloWorld.pdf");
```

他のコンポーネントとの連携

次のトピックでは、ComponentOne に収録される他のコンポーネントを C1Pdf で使用する方法について説明します。

C1FlexGrid データを追加する

C1FlexGrid コントロールのデータを C1Pdf に追加するには、**C1FlexGrid** の **CreateImage** メソッドを使用し、**DrawImage** メソッドを使用して画像をレンダリングします。グリッドで作成される画像は、メタファイルです。

VSPrinter で表示したレポートからの PDF の作成

VSPrinter ActiveX コントロールで作成したドキュメントを PDF に変換するには、**VSPrinter.Picture** プロパティを使用してページ画像を取得し、**DrawImage** メソッドを使用して画像を **PDF for .NET** にレンダリングします。

次のコードでは、VSPrinter ドキュメントを作成し、それを PDF に変換します。

Visual Basic コードの書き方

Visual Basic

```
' PDF ドキュメントを初期化します。
_pdf.Clear()
 ' VSPrinter ドキュメントを作成します(または、既存のドキュメントからロードします)。
.StartDoc()
For i = 1 To 1000
    .Paragraph = "This is paragraph "  i.ToString()
Next
.EndDoc()
 ' ドキュメントを1ページずつスキャンします。
For i = 1 To .PageCount
 ' 現在のプレビューページを設定します。
   .PreviewPage = i
  ' 画像を取得します。
    Dim img As Image = .Picture
   ' 画像を PDF ドキュメントに保存します。
    If i > 1 Then
     _pdf.NewPage()
    End If
    _pdf.DrawImage(img, Me._pdf.PageRectangle)
Next
 ' プレビューを最初のページに戻します。
.PreviewPage = 0
```

C# コードの書き方

C#

```
// PDF ドキュメントを初期化します。
_pdf.Clear();
// VSPrinter ドキュメントを作成します(または、既存のドキュメントからロードします)。
.StartDoc();
```


Copyright © GrapeCity Inc. All rights reserved.

```
for (i = 1; i <= 1000; i++)
{
    .Paragraph = "This is a paragraph " + i.ToString() ;
}
.EndDoc();
 // ドキュメントを1ページずつスキャンします。
for (i = 1; i <= .PageCount; i++)
{
     // 現在のプレビューページを設定します。
    .PreviewPage = i;
     // 画像を取得します。
    Image img = .Picture;
     // 画像を PDF ドキュメントに保存します。
    if (i > 1)
    {
        _pdf.NewPage();
    }
    _pdf.DrawImage(img, this._pdf.PageRectangle)
}
 // プレビューを最初のページに戻します。
.PreviewPage = 0;
```

C1Report ドキュメントから PDF ブックマークを削除する

C1Report ドキュメントから PDF ブックマークを削除するには

- OutlineRootLevel プロパティを -1 に設定するか

  または

- PDF フィルタを作成してアウトラインを無効にします。

Visual Basic コードの書き方

Visual Basic

```
' OutlineRootLevel を -1 に設定します。
Me.C1Report1.OutlineRootLevel = - 1
Me.C1Report1.RenderToFile("report.pdf", C1.Win.C1Report.FileFormatEnum.PDF)
' または、PDF フィルタを作成し、アウトラインを無効にします。
Dim f As New C1.Win.C1Report.PdfFilter("report.pdf")
f.Outline = False
Me.C1Report1.RenderToFilter(f)
```

C# コードの書き方

C#

```
// OutlineRootLevel を -1 に設定します。
this.c1Report1.OutlineRootLevel = -1;
this.c1Report1.RenderToFile("report.pdf", C1.Win.C1Report.FileFormatEnum.PDF);
// または、PDF フィルタを作成し、アウトラインを無効にします。
C1.Win.C1Report.PdfFilter f = new C1.Win.C1Report.PdfFilter("report.pdf");
f.Outline = false;
```


Copyright © GrapeCity Inc. All rights reserved.

```
this.c1Report1.RenderToFilter(f)
```


Copyright © GrapeCity Inc. All rights reserved.