FastReport.Net Programmer's Manual

目次

第1章

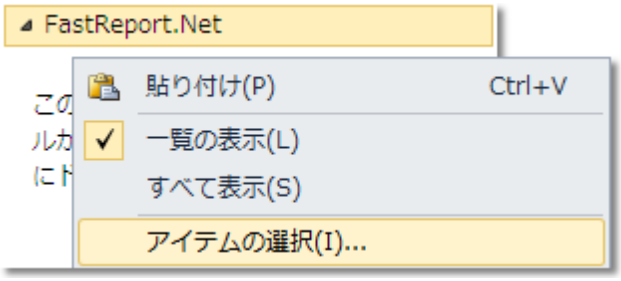
全般情報

全般情報

Visual Studio ツールボックスへのインストール

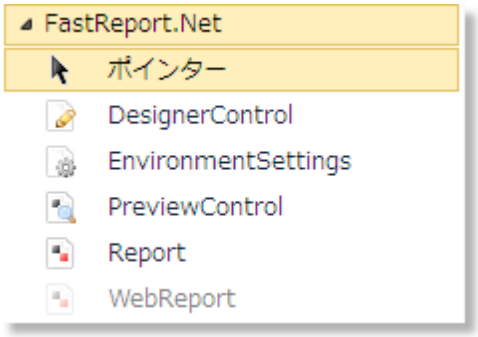
FastReport.Net インストール プログラムによって、FastReport コンポーネントが自動的に Visual Studio ツールボックスへ追加されます。インストール時にこの自動追加機能を無効にした場合は、後でこのコンポーネントを手動で追加することができます。これは次のように行います。

- ツールボックスに「FastReport.Net」タブがある場合はこれを削除します。
- 新しいタブを作成（ツールボックスを右クリックして「タブの追加」を選択）するか、FastReport コンポーネントを追加したい既存のタブを選択します。
- タブを右クリックして「アイテムの選択」を選択します。



- 表示されるダイアログで、「参照」ボタンをクリックし、「C:\Program Files\FastReports\FastReport.Net」フォルダーにある「FastReport.dll」、「FastReport.Web.dll」ファイルを選択します。
- 「OK」ボタンをクリックしてダイアログを閉じます。

これで、選択したタブには以下の FastReport.Net コンポーネントが表示されます。



- Report
- PreviewControl
- DesignerControl
- EnvironmentSettings
- WebReport（このコンポーネントは ASP.NET プロジェクトでのみ表示されます）。

トラブルシューティング

レポートデザイナーで作業しているときに問題（ツールバーやツール ウィンドウの破損など）が発生した場合は、構成ファイルを削除してください。このファイルは FastReport.Net の起動時に作成されます。このファイルは以下のフォルダーにあります。

Windows XP：

```
C:\Documents and Settings\user_name\Local Settings\Application Data\FastReport\FastReport.config
```

Windows Vista:

```
C:\Users\user_name\AppData\Local\FastReport\FastReport.config
```

構成ファイルには以下の情報が保存されています。

- ダイアログ ウィンドウのサイズと位置
- ツールバーの設定
- 最近使用されたデータ接続
- 電子メール設定（プレビュー ウィンドウで［電子メールで送信］機能を使用した場合）

配布

アプリケーションと一緒に以下のファイルを配布することができます。

- FastReport.dll - FastReport.Net のメイン ライブラリ
- FastReport.Web.dll - ASP.Net WebReport コンポーネントが含まれるライブラリ
- FastReport.Bars.dll - ツールバーおよびドッキング ウィンドウ ライブラリ
- FastReport.Editor.dll - 構文を強調表示するコードエディター。エンドユーザー デザイナーを提供しない場合、このライブラリは必要ありません。
- FastReport.xml - FastReport で使用されるクラス、プロパティおよびメソッド用のコメント。このファイルはスクリプトエディターで使用されます。また、［データ］ウィンドウや［プロパティ］ウィンドウのヒントパネルでも使用されます。このファイルの配布は必須ではありません。
- Japanese.frl - FastReport で使用される日本語リソース ファイル。このファイルは FastReport.Net をインストールしたマシンの Localization フォルダー（デフォルトで 32 ビット OS の場合は C:\Program Files\FastReports\FastReport.Net\Localization、64 ビット OS の場合は C:\Program Files (x86)\FastReports\FastReport.Net\Localization）に存在します。

『FastReport.Net User's Manual』ヘルプを配布することができます。これは FRNetUserManual.chm というファイルです。このファイルは、レポートデザイナーで［ヘルプ］メニューの［ヘルプの目次］を選択すると見ることができます。

レポートをファイルに保存した場合、それらのファイルも配布する必要があります。

ソース コードのコンパイル

ソース コードは FastReport.Net Professional エディションに付属しています。これには FastReport.dll、FastReport.Web.dll ライブラリのソース コードが含まれています。このコードは、ご自身のアプリケーションのソリューション ファイルに含めることができます。この手順を以下に示します。

- Visual Studio でプロジェクトを開きます。
- [ファイル] メニューを選択します。
- [追加 | 既存のプロジェクト] を選択します。
- FastReport.csproj ファイル (C:\Program Files\FastReports\FastReport.Net\Source\FastReport フォルダー内にあります) を追加します。
- FastReport.Web.csproj ファイル (C:\Program Files\FastReports\FastReport.Net\Source\FastReport.Web フォルダー内にあります) を追加します。

FastReport および FastReport.Web プロジェクトのアセンブリ署名をオフにします。これは次の手順で行います。

- ソリューション エクスプローラーで "FastReport" プロジェクトを右クリックします。
- [プロパティ] を選択します。
- [署名] タブで [アセンブリの署名] チェックボックスのチェックをオフにします。
- FastReport.Web プロジェクトについても同様の手順を行います。

その他の FastReport.Net アセンブリへの参照を更新します。これは次の手順で行います。

- ソリューション エクスプローラーで "FastReport" プロジェクトの "参照設定" を展開します。
- FastReport.Bars および FastReport.Editor の参照を削除します。
- "参照設定" を右クリックし、[参照の追加] を選択します。
- FastReport.Bars.dll および FastReport.Editor.dll ファイルへの参照を追加します。これらのファイルは C:\Program Files\FastReports\FastReport.Net フォルダー内にあります。

- ソリューション エクスプローラーで "FastReport.Web" プロジェクトの "参照設定" を展開します。
- FastReport および FastReport.Bars の参照を削除します。
- "参照設定" を右クリックし、[参照の追加] を選択します。
- FastReport.dll および FastReport.Bars.dll ファイルへの参照を追加します。これらのファイルは C:\Program Files\FastReports\FastReport.Net フォルダー内にあります。

Windows.Forms での作業

Windows.Forms での作業

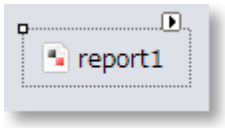
Visual Studio での Report コンポーネントの使用

ここでは、Visual Studio で Report コンポーネントを使用する一般的な方法を説明します。型指定されたデータセットのデータを使用します。

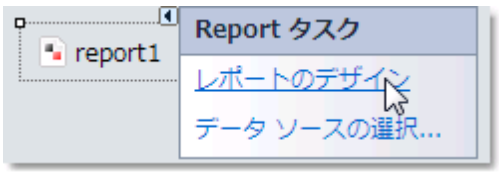
- 新しい Windows フォーム アプリケーションを作成します。
- そのアプリケーションにデータセットを追加します（[データ] メニューの [新しいデータ ソースの追加] を選択）。
- フォーム デザイナーを表示します。
- フォーム上に DataSet コンポーネントを追加し、作成した型指定のデータセットにそのコンポーネントを接続します。

レポートを作成するには以下の手順を実行します。

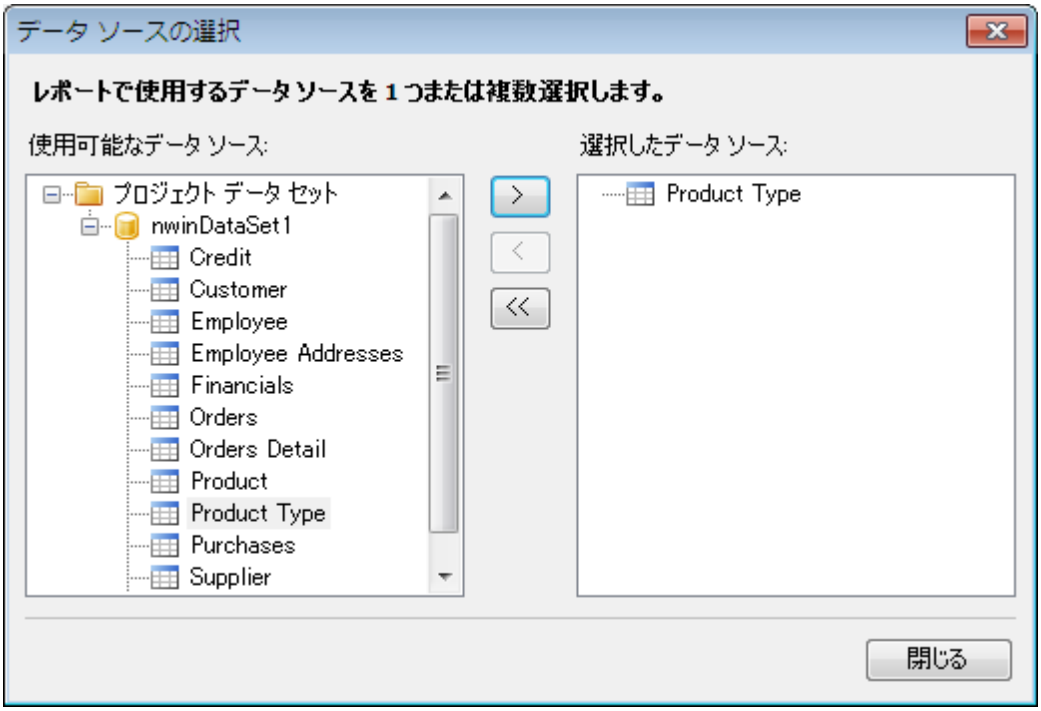
- フォーム上に Report コンポーネントを置きます。



- コンポーネントを右クリック（またはスマート タグ ボタンをクリック）し、[レポートのデザイン...] を選択します。



- レポートで使用するデータ ソースを選択します。



- レポートを作成します。この詳細については、『FastReport.Net User's Manual』をお読みください。
- レポートデザイナーを閉じます。
- フォーム上に Button コントロールを追加します。
- そのコントロールをダブルクリックし、button_Click イベントハンドラーに次のコードを記述します。

```
report1.Show();
```

- プロジェクトを保存し、実行します。ボタンをクリックすると、実行されたレポートが表示されます。

コード内でのレポート作業

コード内で Report コンポーネントを使って作業するには、以下のことを行う必要があります。

- レポートインスタンスを作成する
- そのインスタンスにレポートファイルを読み込む
- アプリケーション定義のデータをレポートに登録する
- 必要に応じて、値をレポートパラメーターに渡す
- レポートを実行する

上記の手順を次のコード例で示します。

```
using (Report report = new Report())
{
  report.Load("report1.frx");
  report.RegisterData(dataSet1, "NorthWind");
  report.Show();
}
```

これらの手順の詳細については、本マニュアルの後のセクションで説明しています。

レポートの保存と読み込み

レポートは以下の方法で保存できます。

方法	説明
アプリケーションのリソース内	前のセクションで示したレポートの一般的な使用シナリオで、この方法が使用されています。これを行うには Report オブジェクトの StoreInResources プロパティを使用します。このプロパティの既定値は True です。この方法には、次のようなプラス面とマイナス面があります。 + レポートはアプリケーションに埋め込まれ、別途ファイルを配布する必要はありません。 - レポートを変更する必要がある場合は、アプリケーションを再コンパイルする必要があります。 レポートの読み込みは自動的に行われます。これを実行するため、FastReport.Net はフォームの InitializeComponent メソッドにコードを追加します。
.FRX ファイル内	この方法は、ユーザーがレポートを変更できるようにする場合に役立ちます。この場合、レポートの StoreInResources プロパティには False を設定します。 ファイルからレポートを読み込む場合は、次のように Report オブジェクトの Load メソッドを使用します。 report1.Load("filename.frx");
データベース内	レポートを文字列または Blob ストリームとして、データベースに保存することができます。 文字列からレポートを読み込む場合は、Report オブジェクトの LoadFromString メソッドを使用します。ストリームからレポートを読み込む場合は、次のように Load メソッドのオーバーロードバージョンを使用します。 report1.Load(stream); レポートデザイナーで読み込み/保存の操作をサポートするには、デザイナーのファイルを開く/ファイルの保存用ダイアログを置き換える必要があります。これを行うには、「開く]および [名前を付けて保存]ダイアログの置き換え」をお読みください。
C#/ VB.NET クラスとして	レポートをクラスとして扱うには、レポートを設計して .cs/ .vb ファイルとして保存します。これには、[名前を付けて保存]ダイアログの [ファイルの種類]フィールドを使用します。選択できるファイルの種類は .cs または .vb です。これはレポートのスクリプト言語によって決まります [スクリプト言語は [レポート]メニューの [オプション]で変更できます)。そのファイルをプロジェクトへ含めます。この方法には、次のようなプラス面とマイナス面があります。 + レポートをクラスとして作業できます。 + レポートをデバッグできます。 + これは、中程度の信頼環境で実行する ASP.NET プロジェクトでレポートを使用する唯一の方法です。 - このレポートは編集できません。編集するには元の .FRX ファイルが必要です。 - レポートを変更する必要がある場合は、アプリケーションを再コンパイルする必要があります。

	レポートを使って作業するには、次のようにレポートのクラスのインスタンスを作成します。 SimpleListReport report = new SimpleListReport(); report.Show();

データの登録

レポートがアプリケーションのデータ（型指定されたデータセットやビジネス オブジェクトなど）を使用する場合は、そのデータをレポートに登録する必要があります。これは、Report オブジェクトの RegisterData メソッドを使用して行うことができます。

「Visual Studio での Report コンポーネントの使用」セクションで説明されているようなレポートを使用する場合は、データを登録する必要がありません。データ登録は FastReport.Net が自動的に行います（フォームの InitializeComponent メソッドに RegisterData 呼び出しを追加します）。

RegisterData メソッドはレポートを読み込んだ後に呼び出す必要があります。

```
report1 = new Report();
report1.Load("report.frx");
report1.RegisterData(dataSet1, "NorthWind");
```

RegisterData メソッドはオーバーロードされ、以下のようなデータを登録することができます。

メソッド	説明
void RegisterData(DataTable data)	データセットを登録します。このメソッドはすべてのテーブル、ビューおよび関係も登録します。 注意：2 つ以上のデータセットを登録する場合は、これではなくRegisterData（DataTable data、string name）メソッドを使用します。
void RegisterData(DataTable data, string name)	データセットを登録します。name パラメーターに名前を指定します（2 つ以上のデータセットを登録する場合、この名前は永続的かつ一意である必要があります）。
void RegisterData(DataTable data, string name)	データ テーブルを登録します。
void RegisterData(DataView data, string name)	データ ビューを登録します。
void RegisterDataAsp(IDataSource data, string name)	AccessDataSource などの ASP.NET データ ソースを登録します。
void RegisterData(DataRelation data, string name)	関係を登録します。
void RegisterData(IEnumerable data, string name, BOConverterFlags flags, int maxNestingLevel)	ビジネス オブジェクトを登録します。使用する項目（プロパティ、フィールド）を flags パラメーターに指定します。maxNestingLevel パラメーターに入れ子の最大レベルを指定します（通常は 3 レベル以下にします）。入れ子のオブジェクトは、場合によってはレポートの実行速度を低下させることがあります。

レポートパラメーターへ値を渡す

レポートはパラメーターを持つことができます。詳細については、『FastReport.Net User's Manual』をお読みください。パラメーターへ値を渡すには、Report オブジェクトの SetParameterValue メソッドを使用します。

```
report1.Load("report.frx");
report1.SetParameterValue("MyParam", 10);
report1.Show()
```

このメソッドは次のように宣言されます。

```
public void SetParameterValue(string complexName, object value)
```

パラメーターの名前を complexName パラメーターに指定します。入れ子のパラメーターにアクセスするには、次のように完全な名前を使用します。

```
" ParentParameter.ChildParameter"
```

レポートの実行

レポートを実行するには、以下に示す Report オブジェクトのいずれかのメソッドを使用します。

メソッド	説明
void Show()	レポートを実行してそれをプレビュー ウィンドウに表示します。このメソッドは 次のメソッドと同じです。 report1.Load("report1.frx"); if (report1.Prepare()) report1.ShowPrepared();
bool Prepare()	レポートを実行します。レポートが正常に実行された場合は、true が返ります。このメソッドの後で、ShowPrepared、PrintPrepared、SavePrepared、Export のうちいずれかのメソッドを呼び出す必要があります。 report1.Load("report1.frx"); if (report1.Prepare()) report1.ShowPrepared();
bool Prepare(bool append)	レポートを実行します。append パラメーターに true が設定された場合、その実行されたレポートは既存のレポートに追加されます。これにより、いくつかのレポートを作成し、それらを 1 つのレポートとしてプレビューに表示することができます。 report1.Load("report1.frx"); report1.Prepare(); report1.Load("report2.frx"); report1.Prepare(true); report1.ShowPrepared();
void ShowPrepared()	実行したレポートをプレビュー ウィンドウに表示します。このレポートは、Prepare メソッドを使用して実行されたレポート、または LoadPrepared メソッドを使用して .FPX ファイルから読み込んだレポート (データ保存レポート) のいずれかです。

	report1.Load("report1.frx"); if (report1.Prepare()) report1.ShowPrepared(); またはreport1.LoadPrepared("report1.frx"); report1.ShowPrepared();
void ShowPrepared(bool modal)	実行したレポートをプレビュー ウィンドウに表示します。modal パラメーターはプレビューをモーダルに表示するかどうかを決定します。
void ShowPrepared(bool modal, Form owner)	前のメソッドと同じです。owner パラメーターはプレビュー ウィンドウを所有するウィンドウを決定します。
void ShowPrepared(Form mdiParent)	前のメソッドと同じです。mdiParent パラメーターはメイン MDI ウィンドウを決定します。

レポートのデザイン

レポートデザイナーをご自分のアプリケーションで使用することができます。これは、FastReport.Net Basic 以外のすべてのエディションで機能します。これを行うには、Report オブジェクトの Design メソッドを使用します。

```
report1 = new Report();
report1.Load("report1.frx");
report1.Design();
```

Design メソッドはオーバーロードされます。

メソッド	説明
bool Design()	デザイナーを表示します。
bool Design(bool modal)	デザイナーを表示します。modal パラメーターはデザイナーをモーダルに表示する必要があるかどうかを決定します。
bool Design(Form mdiParent)	デザイナーを表示します。mdiParent パラメーターはメイン MDI ウィンドウを定義します。

レポートのエクスポート

実行したレポートは、サポートされる形式のいずれか 1 つにエクスポートできます。現時点で、以下の形式が使用できます。

- PDF
- HTML
- RTF
- Excel XML（Excel 2003 以上）
- Excel 2007

- CSV
- TXT
- OpenOffice Calc
- 図 (BMP、PNG、JPEG、GIF、TIFF、メタファイル)

エクスポートはエクスポートフィルターで実行します。これは次の手順で行います。

- Prepare メソッドを使用してレポートを実行します。
- エクスポートフィルターのインスタンスを作成して、そのプロパティを設定します。
- Report オブジェクトの Export メソッドを呼び出します。

次のコード例では、実行したレポートを HTML 形式にエクスポートします。

```
// レポートを実行する
report1.Prepare();

// HTML エクスポートフィルターのインスタンスを作成する
FastReport.Export.Html.HTMLExport export = new FastReport.Export.Html.HTMLExport();

// エクスポートオプション ダイアログを表示してエクスポートを行う
if (export.ShowDialog())
  report1.Export(export, "result.html");
```

この例では、エクスポート設定をダイアログ ウィンドウで行っています。

FastReport.Net 環境の構成

ツールボックスで使用可能な EnvironmentSettings コンポーネントを使用して、いくつかの FastReport.Net 環境設定を制御することができます。これを行うには、このコンポーネントをフォーム上に置き、[プロパティ] ウィンドウでそのプロパティを設定します。

EnvironmentSettings.ReportSettings プロパティには以下のようなレポート関連の設定があります。

プロパティ	説明
Language DefaultLanguage	新しいレポートの既定のスクリプト言語。
bool ShowProgress	進行状況ウィンドウを表示するかどうかを決定します。
bool ShowPerformance	レポートパフォーマンスに関する情報 (レポート生成時間、メモリの消費) をプレビューウィンドウの右下隅に表示するかどうかを決定します。

EnvironmentSettings.DesignerSettings プロパティには以下のデザイナー関連の設定があります。

プロパティ	説明
Icon Icon	デザイナー ウィンドウのアイコン。
Font DefaultFont	レポートで使用される既定のフォント。

EnvironmentSettings.PreviewSettings プロパティには以下のようなプレビュー関連の設定があります。

プロパティ	説明
PreviewButtons Buttons	プレビューのツールバーに表示されるボタンのセット。
int PagesInCache	プレビュー時にメモリ キャッシュに格納できる保存ページ数。
bool ShowInTaskbar	プレビュー ウィンドウが Windows のタスクバーに表示されるようにするかどうかを決定します。
bool TopMost	プレビュー ウィンドウを最上位フォームとして表示させるかどうかを決定します。
Icon Icon	プレビュー ウィンドウのアイコン。
string Text	プレビュー ウィンドウのテキスト。テキストが設定されない場合は、既定のテキスト"プレビュー" が使用されます。

EnvironmentSettings.EmailSettings プロパティには電子メールのアカウント設定があります。これらの設定は、プレビュー ウィンドウの [電子メールで送信] 機能で使用されます。

プロパティ	説明
string Address	送信元アドレス (例 :電子メール アドレス)。
string Name	送信者名 (例 :名前)。
string MessageTemplate	新しいメッセージを作成するために使用するメッセージ テンプレート。たとえば、"拝啓、... 敬具" など。
string Host	SMTP ホストアドレス。
int Port	SMTP ポート (既定値は 25)。
string UserName, string Password	ユーザー名とパスワード。サーバーが認証を必要としない場合、このプロパティは空のままにしてください。
bool AllowUI	[電子メールの送信] ダイアログでこれらの設定を変更できるようにします。設定は FastReport.Net 構成ファイルに保存されます。

UI スタイルの設定は EnvironmentSettings コンポーネントの以下のプロパティで使用できます。

プロパティ	説明
UIStyle UIStyle	デザイナーとプレビュー フォームのスタイル。VisualStudio2005、Office2003、Office2007Blue、Office2007Silver、Office2007Black、VistaGlass の 6 つのスタイルが使用できます。 既定のスタイルは "Office2007Black" です。
bool UseOffice2007Form	このプロパティはデザイナーやプレビューのフォームに影響します。Office2007Blue、Office2007Silver、Office2007Black、VistaGlass のいずれかのスタイルが選択された場合に、Office2007 スタイル フォームを使用するかどうかを決定します。 既定値は True です。

上記のプロパティに加えて、EnvironmentSettings コンポーネントにはいくつかのイベントがあります。それらのイベントを使用して以下のことが行えます。

- デザイナーの [ファイルを開く] および [ファイルの保存] の標準ダイアログの置き換え
- 標準の進行状況ウィンドウの置き換え
- 独自の接続文字列を、レポートで定義された接続へ渡す

これらのタスクについては、本マニュアルの後のセクションで説明しています。

[開く] および [名前を付けて保存] ダイアログの置き換え

データベースにレポートを保存することにした場合、データベースからレポートを開く、またはデータベースへレポートを保存できるよう デザイナーを変更する必要があります。つまり、標準の [開く] および [名前を付けて保存] ダイアログを、データベースを扱える独自のダイアログに置き換える必要があります。これを行うには、EnvironmentSettings コンポーネント (前のセクションを参照) を使用します。このコンポーネントには以下のイベントがあります。

イベント	説明
CustomOpenDialog	レポート デザイナーで [開く] ダイアログを表示しようとしたときに発生します。このイベント ハンドラーで、ユーザーがレポート ファイルを選択できるダイアログ ウィンドウを表示する必要があります。ダイアログが実行できたら、e.Cancel に false を返し、選択されたファイル名を e.FileName に設定する必要があります。 以下のコード例ではこのイベントの使用方法を示します。 `private void CustomOpenDialog_Handler(` `  object sender, OpenSaveDialogEventArgs e)` `{` `  using (OpenFileDialog dialog = new OpenFileDialog())` `  {` `  dialog.Filter = "Report files (*.frx)|*.frx";` `  // ダイアログが実行できたら` `  // e.Cancel に false を設定する` `  e.Cancel = dialog.ShowDialog() != DialogResult.OK;` `  // 選択されたファイル名を e.FileName に設定する` `  e.FileName = dialog.FileName;` `  }` `}`
CustomSaveDialog	レポート デザイナーで [名前を付けて保存] ダイアログを表示しようとしたときに発生します。このイベント ハンドラーで、ユーザーがレポート ファイルを選択できるダイアログ ウィンドウを表示する必要があります。ダイアログが実行できたら、e.Cancel に false を返し、選択されたファイル名を e.FileName に設定する必要があります。 以下のコード例ではこのイベントの使用方法を示します。 `private void CustomSaveDialog_Handler(` `  object sender, OpenSaveDialogEventArgs e)` `{` `  using (SaveFileDialog dialog = new SaveFileDialog())` `  {` `  dialog.Filter = "Report files (*.frx)|*.frx";` `  // e.FileName から既定のファイル名を取得する` `  dialog.FileName = e.FileName;`

	`` // ダイアログが実行できたら // e.Cancel に false を設定する e.Cancel = dialog.ShowDialog() != DialogResult.OK; // 選択されたファイル名を e.FileName に設定する e.FileName = dialog.FileName; } } ```
CustomOpenReport	レポートデザイナーでレポートを読み込もうとしたときに発生します。このイベントハンドラーで、e.Report プロパティで指定されたレポートを、e.FileName プロパティで指定された場所から読み込む必要があります。後者のプロパティには CustomOpenDialog イベントハンドラーで返された名前が含まれます。これはファイル名、データベース キー値などです。 以下のコード例ではこのイベントの使用方法を示します。 ``` private void CustomOpenReport_Handler(object sender, OpenSaveReportEventArgs e) { // 指定した e.FileName からレポートを読み込む e.Report.Load(e.FileName); } ```
CustomSaveReport	レポートデザイナーでレポートを保存しようとしたときに発生します。このイベントハンドラーで、e.Report プロパティで指定されたレポートを、e.FileName プロパティで指定された場所に保存する必要があります。後者のプロパティには CustomSaveDialog イベントハンドラーで返された名前が含まれます。これはファイル名、データベース キー値などです。 以下のコード例ではこのイベントの使用方法を示します。 ``` private void CustomSaveReport_Handler(object sender, OpenSaveReportEventArgs e) { // 指定した e.FileName にレポートを保存する e.Report.Save(e.FileName); } ```

標準の進行状況ウィンドウの置き換え

進行状況ウィンドウは以下の作業時に表示されます。

- レポートの実行
- 印刷
- エクスポート

EnvironmentSettings コンポーネントの ReportSettings.ShowProgress プロパティを False に設定すると、この標準の進行状況を表示しないようにすることができます。また、その標準の進行状況ウィンドウの代わりに独自のウィンドウを表示させることもできます。これを行うには、EnvironmentSettings コンポーネントの以下のイベントを使用します（「FastReport.Net 環境の構成」セクションを参照）。

イベント	説明
StartProgress	操作の前に 1 回発生します。このイベント内で、独自の進行状況ウィンドウを作成し、表示します。

Progress	現在のレポートページが処理されるたびに発生します。このイベント内で、独自のウィンドウに進行状況を表示する必要があります。
FinishProgress	操作の後に 1 回発生します。このイベント内で、進行状況ウィンドウを破棄する必要があります。

Progress イベントは ProgressEventArgs タイプの e パラメーターを使用します。これには以下のプロパティがあります。

プロパティ	説明
string Message	メッセージ テキスト。
int Progress	現在処理されているレポートページのインデックス。
int Total	レポートの総ページ数。総ページ数が不明であるため、レポートの実行時にこのパラメーターを 0 にすることができます。

ほとんどの場合、Progress イベントハンドラーで、e.Message プロパティによるテキストを表示する必要があります。進行状況バーを表示する場合は、他のパラメーターが役立ちます。

独自の接続文字列を渡す

レポート内に定義されたデータ ソースを使用する場合は、アプリケーション定義の接続文字列をレポートへ渡す必要があります。これを行うには 3 つの方法があります。

第 1 の方法 :レポートの Connection オブジェクトへ接続文字列を直接渡します。次のように実行します。

```
report1.Load(...);
// レポートの読み込み後、そのレポートを実行する前に
// そのレポートに 1 つの接続があることを前提とする
report1.Dictionary.Connections[0].ConnectionString = my_connection_string;
report1.Show()
```

第 2 の方法 :レポートパラメーターを使用して接続文字列を渡します。次のように実行します。

- レポートデザイナーを実行します。
- [データ] ウィンドウで、新しいレポートパラメーターを (たとえば、"MyParameter" という名前で) 作成します。詳細については、『FastReport.Net User's Manual』を参照してください。
- [データ] ウィンドウで、データ ソースを含む Connection オブジェクトを選択します。
- [プロパティ] ウィンドウで、ConnectionStringExpression プロパティに次の値を設定します。

```
[MyParameter]
```

- 接続文字列を MyParameter パラメーターに渡します。

```
report1.SetParameterValue("MyParameter", my_connection_string);
```

第 3 の方法 :EnvironmentSettings コンポーネントの DatabaseLogin イベントを使用します (「FastReport.Net 環境の構成」セクションを参照)。このイベントは FastReport が接続を開くたびに発生します。このイベントハンドラーの例を以下に示します。

```
private void environmentSettings1_DatabaseLogin(
  object sender, DatabaseLoginEventArgs e)
```

```
{
  e.ConnectionString = my_connection_string;
}
```

DatabaseLogin イベントはグローバルであることを覚えておいてください。これはすべてのレポートで動作します。

カスタム SQL を渡す

レポートには、データウィザード（[データ] メニューの [データソースの追加] を選択）を使用して追加されたデータソースを含めることができます。場合によっては、アプリケーションからそのデータソースに対してカスタム SQL を渡す必要があります。これを行うには、以下のコードを使用します。

```
using FastReport.Data;

report1.Load(...);
// レポートの読み込み後、そのレポートを実行する前に
// テーブルをそのエイリアスで検索する
TableDataSource table = report1.GetDataSource("MyTable") as TableDataSource;
table.SelectCommand = "new SQL text";
report1.Show()
```

レポートオブジェクトへの参照

レポートをクラスとして扱う場合（「レポートの保存と読み込み」セクションを参照）、レポートオブジェクトを直接参照することができます。次のコード例では、レポートに含まれている Text1 オブジェクトのフォントを変更する方法を示しています。

```
SimpleListReport report = new SimpleListReport();
report.Text1.Font = new Font("Arial", 12);
```

別のケースで、オブジェクトを参照しなければならない場合は、Report オブジェクトの FindObject メソッドを使用する必要があります。

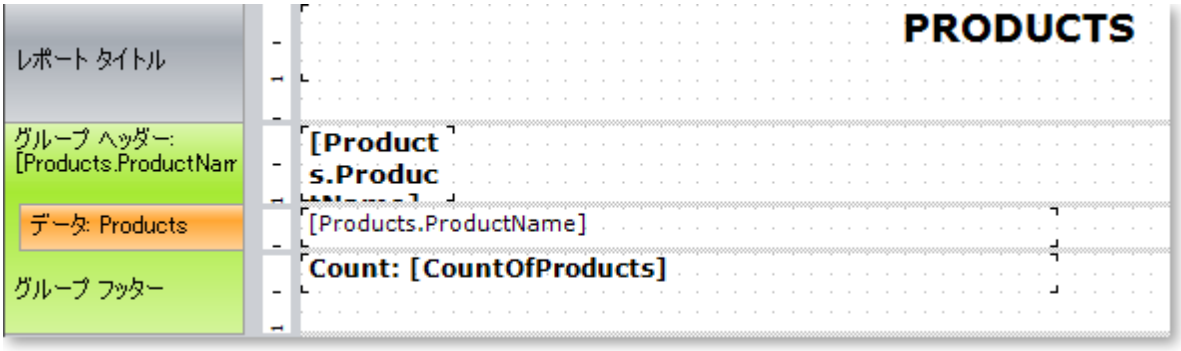
```
TextObject text1 = report1.FindObject("Text1") as TextObject;
text1.Font = new Font("Arial", 12);
```

レポートに定義されたデータソースを参照するには、Report オブジェクトの GetDataSource メソッドを使用します。このメソッドは、パラメーターとしてデータソースのエイリアスを使用します。

```
DataSourceBase ds = report1.GetDataSource("Products");
```

コードを使用したレポートの作成

コードを使ってレポートを作成する方法を見てみましょう。次のようなレポートを作成します。




```
Report report = new Report();

// "Products" テーブルを登録する
report.RegisterData(dataSet1.Tables["Products"], "Products");
// レポートでこのテーブルを使用できるようにする
report.GetDataSource("Products").Enabled = true;

// レポート ページを作成する
ReportPage page1 = new ReportPage();
page1.Name = "Page1";
report.Pages.Add(page1);

// レポート タイトル バンドを作成する
page1.ReportTitle = new ReportTitleBand();
page1.ReportTitle.Name = "ReportTitle1";
// 高さを 1.5cm に設定する
page1.ReportTitle.Height = Units.Centimeters * 1.5f;

// グループ ヘッダーを作成する
GroupHeaderBand group1 = new GroupHeaderBand();
group1.Name = "GroupHeader1";
group1.Height = Units.Centimeters * 1;
// グループの条件を設定する
group1.Condition = "[Products.ProductName].Substring(0, 1)";
// グループを page1.Bands コレクションに追加する
page1.Bands.Add(group1);
```

```
// グループ フッターを作成する
group1.GroupFooter = new GroupFooterBand();
group1.GroupFooter.Name = "GroupFooter1";
group1.GroupFooter.Height = Units.Centimeters * 1;

// データ バンドを作成する
DataBand data1 = new DataBand();
data1.Name = "Data1";
data1.Height = Units.Centimeters * 0.5f;
// データ ソースを設定する
data1.DataSource = report.GetDataSource("Products");
// データ バンドをグループへ接続する
group1.Data = data1;

// テキスト オブジェクトを作成する

// レポート タイトル
TextObject text1 = new TextObject();
text1.Name = "Text1";
// 境界を設定する
text1.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 19, Units.Centimeters * 1);
// テキストを設定する
text1.Text = "PRODUCTS";
// 外観を設定する
text1.HorzAlign = HorzAlign.Center;
text1.Font = new Font("Tahoma", 14, FontStyle.Bold);
// レポート タイトルに追加する
page1.ReportTitle.Objects.Add(text1);

// グループ
TextObject text2 = new TextObject();
text2.Name = "Text2";
text2.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 2, Units.Centimeters * 1);
text2.Text = "[[Products.ProductName].Substring(0, 1)]";
text2.Font = new Font("Tahoma", 10, FontStyle.Bold);
// グループ ヘッダーに追加する
group1.Objects.Add(text2);
```

```
// データ バンド
TextObject text3 = new TextObject();
text3.Name = "Text3";
text3.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 10, Units.Centimeters * 0.5f);
text3.Text = "[Products.ProductName]";
text3.Font = new Font("Tahoma", 8);
// データ バンドに追加する
data1.Objects.Add(text3);

// グループ フッター
TextObject text4 = new TextObject();
text4.Name = "Text4";
text4.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 10, Units.Centimeters * 0.5f);
text4.Text = "Count:[Count Of Products]";
text4.Font = new Font("Tahoma", 8, FontStyle.Bold);
// グループ フッターに追加する
group1.GroupFooter.Objects.Add(text4);

// 合計を追加する
Total groupTotal = new Total();
groupTotal.Name = "Count Of Products";
groupTotal.TotalType = TotalType.Count;
groupTotal.Evaluator = data1;
groupTotal.PrintOn = group1.Footer;
// レポート合計に追加する
report.Dictionary.Totals.Add(groupTotal);

// レポートを実行する
report.Show();
```

実行されたレポートは次のように表示されます。

独自のプレビュー ウィンドウの使用

EnvironmentSettings コンポーネントを使用すると（「FastReport.Net 環境の構成」セクションを参照）、標準のプレビュー ウィンドウを調整することができます。関連するプロパティは EnvironmentSettings.PreviewSettings プロパティ内部に含まれています。

何らかの理由で標準のプレビュー ウィンドウを使用したくない場合は、独自のウィンドウを作成することができます。これを行うには 、フォーム上に追加できる PreviewControl コントロールを使用します。このコントロールにレポートを表示するには、以下のコードを使用して Report オブジェクトにそのコントロールを接続します。

```
report1.Preview = previewControl1;
```

レポートを実行し PreviewControl に表示するには、Report オブジェクトの Show メソッドを使用します。

```
report1.Show()
your_form.ShowDialog();
```

または、次のコードを使用します。

```
if (report1.Prepare())
{
  report1.ShowPrepared();
  your_form.ShowDialog();
}
```

上記のコード例で、your_form には PreviewControl が含まれるフォームを指定します。

PreviewControl コンポーネントのメソッドを使用する場合、コードからそのフォームを処理できます。さらに、ToolbarVisible や StatusbarVisible プロパティを使用すれば、標準のツールバーやステータスバーではなく独自のものを使用することもできます。これは Demos\ C#\ CustomPreview サンプル プロジェクトで例示しています。

データ ウィザードでテーブルをフィルター処理する

データ ウィザードは [データ] メニューの [データ ソースの追加] を選択して呼び出すことができます。このウィザードでは、接続の設定とデータ テーブルの選択を行うことができます。既定では、選択した接続で使用できるすべてのテーブルが表示されます。不要なテーブルを除外する場合は、Config.DesignerSettings.FilterConnectionTables イベントを使用します。次のコード例では、テーブル一覧から "Table 1" というテーブルを取り除く方法を示します。

```
using FastReport.Utils;
Config.DesignerSettings.FilterConnectionTables += FilterConnectionTables;

private void FilterConnectionTables(
  object sender, FilterConnectionTablesEventArgs e)
{
  if (e.TableName == "Table 1")
    e.Skip = true;
}
```

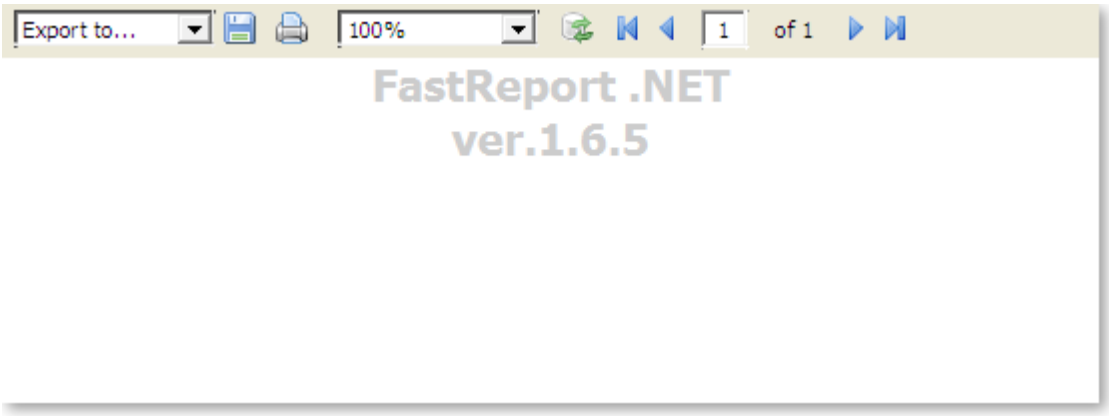
ASP.NET での作業

ASP.NET での作業

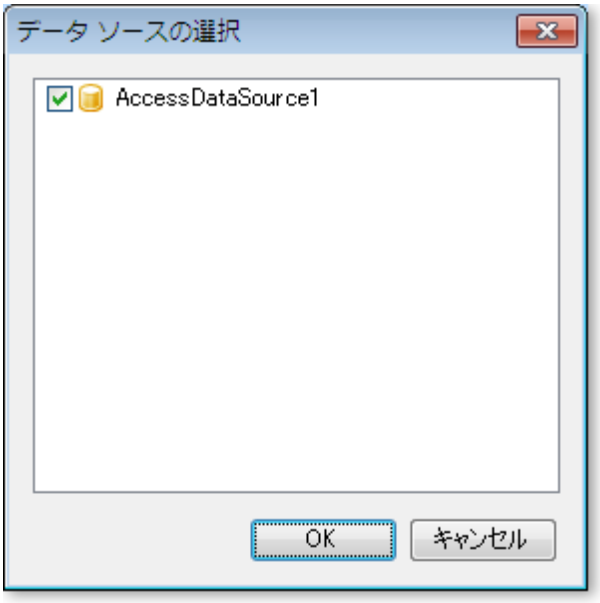
WebReport コンポーネントの使用

WebReport コンポーネントの一般的な使用法について見てみましょう。

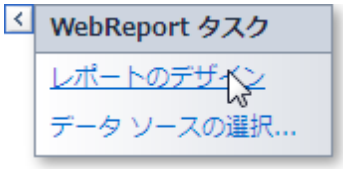
- 必要なすべてのデータ ソース (AccessDataSource など) を持つ Web プロジェクトがあることを前提とします。
- Web フォームに WebReport コンポーネントを置きます。



- スマート タグ メニューで [データ ソースの選択] を選択し、レポートで使用するデータ ソースを 1 つまたは複数選択します。



- スマート タグ メニューで [レポートのデザイン] を選択し、レポート デザイナーを実行します。



- レポートを作成します。詳細については、『FastReport.Net User's Manual』をお読みください。
- デザイナーを閉じます。
- 変更を保存し、プロジェクトを実行します。実行されたレポートのウィンドウが表示されます。

レポートの保存と読み込み

レポートは以下の方法で保存できます。

方法	説明
Web フォーム内	先のセクションで示した一般的なシナリオでこの方法を使用します。レポートは、WebReport コンポーネントの ReportResourceString プロパティに格納されます。この方法には、次のようなプラス面とマイナス面があります。 + これは FastReport.Net を使って作業する最も簡単な方法です。 - レポートテンプレートは Web フォームの ViewState に格納されます。これはクライアント側に転送されます。レポートのサイズが大きい場合は作業の速度が低下する可能性があります。 - この方法は "中程度の信頼" モードと互換性がありません。 レポートは自動的に読み込まれます。
.FRX ファイル内	この方法は、レポートが App_Data という特別なフォルダーにあるファイルに保存されていることを前提とします。これは次の手順で行います。 • レポートデザイナーを実行します。 • レポートを作成し、それを .FRX ファイルに保存します。 • ソリューション エクスプローラーで "App_Data" フォルダーを選択し、右クリックして 追加｜既存の項目] を選択します。先ほど保存したレポートファイルを選択します。 • WebReport コンポーネントを選択し、その ReportResourceString プロパティをクリアします。 • ReportFile プロパティを選択し、そのエディターを起動して App_Data フォルダーからレポート選択します。 この方法には、次のようなプラス面とマイナス面があります。 + レポートはクライアントマシンへ転送されません。 - この方法は "中程度の信頼" モードと互換性がありません。 レポートは自動的に読み込まれます。
C#/ VB.NET クラスと	このモードでは、レポートをクラスとして扱います。これは次の手順で行います。

して

- レポートを設計し、それを .cs または .vb ファイルとして保存します。これには、"名前を付けて保存"ダイアログの "ファイルの種類"フィールドを使用します。選択できるファイルの種類は .cs または .vb です。これはレポートのスクリプト言語によって決まります (スクリプト言語は "レポート"メニューの "オプション"で変更できます)。
- そのファイルをプロジェクトへ含めます。ファイルは App_Code フォルダーに保存するようにしてください。
- WebReport コンポーネントの ReportResourceString および ReportFile properties をいずれもクリアします。

この方法には、次のようなプラス面とマイナス面があります。

+ レポートを標準クラスとして扱うことができます。
+ レポートは Visual Studio でデバッグできます。
+ これは "中程度の信頼" モードでレポートを使用する唯一の方法です。
- このレポートは編集できません。編集するには元の .FRX ファイルが必要です。

レポートを使って作業するには、WebReport.StartReport イベントハンドラーを作成します。このハンドラーで、以下のことを行います。

- レポートクラスのインスタンスを作成します。
- データを登録します。
- レポートを WebReport コンポーネントの Report プロパティに設定します。

StartReport イベントハンドラーの例：

```
SimpleListReport report = new SimpleListReport();
report.RegisterDataAsp(your_data, "your_data_name");
WebReport1.Report = report;
```

データの登録

WebReport コンポーネントのスマートタグ メニューを使用してデータソースを選択する場合は、データを手動で登録する必要はありません。この場合、FastReport.Net は WebReport コンポーネントの ReportDataSources プロパティにデータソースの名前を格納します。

このようなデータ登録の方法を使用しない場合は、手動で登録する必要があります。これには WebReport コンポーネントの StartReport イベントを使用します。このイベントハンドラーで、レポートの RegisterData および RegisterDataAsp メソッドを呼び出します。このレポートは WebReport.Report プロパティを使ってアクセスできます。

```
webReport1.Report.RegisterData(myDataSet);
```

データの登録については、"Windows.Forms での作業"の章の "データの登録"セクションもお読みください。

レポートパラメーターへ値を渡す

レポートパラメーターへ値を渡すには、Report オブジェクトの SetParameterValue メソッドを使用します。このメソッドについては、「Windows.Forms での作業」の章の「レポートパラメーターへ値を渡す」セクションで詳しく説明しています。

このメソッドを ASP.NET で使用するには、WebReport コンポーネントの StartReport イベント用にイベントハンドラーを作成する必要があります。このレポートは WebReport.Report プロパティを使ってアクセスできます。

```
webReport1.Report.SetParameterValue("MyParam", 10);
```

"中程度の信頼" モードでの作業

このモードは多くの共有ホスティング プロバイダーで使用されています。このモードでは、以下の行為が制限されています。

- レポートのコンパイルを行うことができない
- MS Access データソースを使用できない
- RichObject を使用できない
- WinAPI 呼び出しまたは一時ファイル（PDF、Open Office）を使用するいくつかのエクスポートフィルターが使用できない
- プロバイダーによっては、上記以外にも制限される行為があるかもしれません。

このモードでレポートの作業を行うには、「レポートの保存と読み込み」セクションで説明しているように、レポートを C#/ VB.NET クラスとして保存する必要があります。この場合、レポートのコンパイルは必要ありません。

また、System.Windows.Forms.DataVisualization.dll アセンブリを GAC に追加する必要があります。このアセンブリは、Microsoft Chart コントロールの一部であり、グラフを描画するために FastReport で使用されます。このアセンブリを GAC に追加する場合は、ご利用の共有ホスティング プロバイダーにご相談ください。