REPORT MANAGER 2.0 - DOCUMENTATION

DEVELOPED BY JOSHUA H. SANTIAGO

This Library utilizes iTextSharp to generate PDF documents. Please review iTextSharp documentation for further details about this library.

// Thank you for using Report Manger

This library accepts a data table and creates a PDF report in a variety of sizes and orientations to best suit the developer and application user. The following documentation should assist the developer in implementing the library into their system.

// Preparing your data

Report Manager receives data as a standard DataTable object, which is then passed directly into the constructor. In order to pass data properly the developer should ensure that a table's headers are human readable. Renaming column headers (if necessary) can help to make the reports both easier to read and more presentable to users who do not understand standard database column naming schemes.

// Creating report headers

Creating a report header is easy with Report Manager. The developer will pass a StringBuilder directly into the constructor. The StringBuilder makes designing a header simple, in that it allows for a line by line customization. There is no height limit for the header, and the body of the report will automatically lower itself to provide the necessary space for it to generate correctly. A header is presented at the top of the first page of a report and is centered to provide a polished look. The developer also has the option of requesting an automatic report-date line which will generate under the header. No need to implement this into your StringBuilder object.

Headers are usually created as follows:

```
        private StringBuilder Header()
        {
            StringBuilder innerHeader = new StringBuilder();
            innerHeader.AppendLine("Test Application Report");
            innerHeader.AppendLine("Acme Distribution Corporation");
            innerHeader.AppendLine("2017 Inventory Totals");
            innerHeader.AppendLine("************************************");
            return innerHeader;
        }
```

// Designing your report

The developer will need to decide what appearance the report should have in order to best present the data that will be passed to the report. There are several options to think about when performing the initial setup:

1. Whether or not the report should show a "Report Date: xxx" line below the header. This is activated by setting the bool value to true or false in the constructor or by later assigning it to the appropriate property.
2. Whether or not the report should show a "*** END OF REPORT ***" indicator at the end of the report data. This is activated by setting the bool value to true or false in the constructor or by later assigning it to the appropriate property.
3. The size and orientation of the page which holds the report. There are currently four options available for the developer to choose from. These options are passed into the constructor or assigned to the property as an integer between 1 and 4.
    a. 1. Standard LETTER size document, portrait orientation.
    b. 2. Standard LETTER size document, landscape orientation.
    c. 3. Standard LEGAL size document, portrait orientation.
    d. 4 Standard LEGAL size document, landscape orientation
    e. Additional sizes can be added to the library by modifying the ReportDocument class located within the "components" folder. The current sizes are activated within a switch statement.
4. The font family that will be used throughout the report document. It is recommended to set this value to "Courier" as it seems to be the best way to present data. The developer can use any font family that he/she wishes but should probably use a single-spaced font set for this purpose. This value is passed as a string into the constructor or later defined in the properties. Ensure that the font family name is properly defined to prevent errors.
5. Margins are set individually and are passed as floating-point values. These are set in the constructor or later passed into a property. The following are recommended:
    a. Left Margin: 25f
    b. Right Margin: 25f
    c. Top Margin: 35f (This gives room for a two or three ring binder punch)
    d. Bottom Margin: 25f

// Constructor

After importing Report Manager, the developer will create a new CreateReport object within the method of their choice. The CreateReport class has 6 constructors to choose from to make designing your report as simple as possible.

/// Standard Constructor:

The standard constructor allows the developer to set-up their report with granular detail. It requires all the variables indicated in the "Designing Your Report" section to be set.

```
        // Full constructor
        public CreateReport(DataTable reportData, StringBuilder documentHeader, bool includeReportDate, bool
includeEndOfRecordIndicator,
            int documentPageSizeCode, string fontFamily, float leftMargin, float rightMargin, float topMargin, float
bottomMargin)
```

/// All Values at Default

This constructor only takes the DataTable and the StringBuilder for the header and does all the other hard work for the developer. Default page size is Letter in Landscape orientation, date and end of record indicators visible, font family set to Courier and all margins set to optimal values.

```
        // All values at default
        public CreateReport(DataTable reportData, StringBuilder documentHeader)
```

/// Defaults with Date Option

This constructor is the same as above, with the addition of the Report Date option. This will allow the user to set all defaults while eliminating the report date field.

```
        // Defaults with Date option
        public CreateReport(DataTable reportData, StringBuilder documentHeader, bool
includeReportDate)
```

/// Defaults with Date Option and End of Record Indicator Option

This constructor is the same as above, with the addition of the End of Record Indicator option. This will allow the user to set all defaults while eliminating the End of Record Indicator field.

```
        // Defaults with Date and End of Record Indicator
        public CreateReport(DataTable reportData, StringBuilder documentHeader, bool
includeReportDate, bool includeEndOfRecordIndicator)
```

/// Defaults with Font Option

This constructor is the same as the “All Values as Default” constructor but adds the option to change the font family.

```
        // Defaults with font Option
        public CreateReport(DataTable reportData, StringBuilder documentHeader, string
fontFamily)
```

/// Defaults with Report Date, End of Record Indicator and Font Option

```
        // Defaults with font, date and end of record indicator
        public CreateReport(DataTable reportData, StringBuilder documentHeader, bool
includeReportDate, bool includeEndOfRecordIndicator, string fontFamily)
```

/// Setting Properties

The developer may also set properties individually. By using the default constructor (or any of the others available) the default values will be set. But if you wish to change just the page size code,, the font family, or any other document property, the option is there.

Within your method, create a CreateReport object and set its properties as follows:

```
        private void YourMethod()
        {
            CreateReport cReport = new CreateReport(_dataTable, Header())
            {
                DocumentPageSizeCode = 4
            };
            cReport.CreateDocument();

            PDFReader pdfr = new PDFReader(cReport.GetDocumentPath());
            pdfr.Show();
            this.Hide();
        }
```

(This example also shows how to return the document path which can be inserted into your PDF reader of choice. In this sample application, the document path is passed into a window containing a PDF reader. The string is t hen read by the reader which displays the completed PDF document.

// Generating the Report

As indicated above in the "Setting Properties" section, generating your report is an easy process. Within your method create a new CreateReport object, set any properties, call .CreateDocument() and the call .GetDocumentPath().

This document path will then be passed into the PDF reader of your choice.

```
        private void YourMethod()
        {
            CreateReport cReport = new CreateReport(_dataTable, Header())
            {
                DocumentPageSizeCode = 4
            };

```

```

            cReport.CreateDocument();
```

```
            PDFReader pdfr = new PDFReader(cReport.GetDocumentPath());
            pdfr.Show();
            this.Hide();
        }
```