**Purpose**

HouseRentalContract project is an example of the home rent contract generation. The example demonstrates a simple textual multi-page document that includes repeating footer with automatic page numbering and consists of two distinctive sections with different content of a footer area (initials area on the all pages of the contract except the page with signatures). Example also includes demonstration of using Word-like tabulation and working with JSON.

The example source is available in repo <https://github.com/gehtsoft-usa/PDF.Flow.Examples/tree/master/Examples/HouseRentalContract>.

**Prerequisites**

1) **Visual Studio 2017** or above is installed.

To install a community version of Visual Studio use the following link: <https://visualstudio.microsoft.com/vs/community/>

Please make sure that the way you are going to use Visual Studio is allowed by the community license. You may need to buy Standard or Professional Edition.

2) **.NET Core Framework SDK 2.1** or above is installed.

To install the framework use the following link: <https://dotnet.microsoft.com/download>

**Description**

**Contract data**

The contract content is located in the file Content/contract.txt. This is a template with the contract text and the fields, which will be substituted with the values from the dictionary. Such fields should be taken in braces:

*A deposit in the amount of {deposit} will be held.*

Breaks between the paragraphs have to be indicated with {paragraphbreak}.

**Dictionary data**

The values for the contract fields are located in the file Content/contract\_dictionary.json:

*[*

*{*

*"Key": "renter",*

*"Value": "John Smith"*

*},*

*…*

*…*

*{*

*"Key": "guests",*

*"Value": "4"*

*}*

*]*

**Output file**

The example creates the file HouseRentalContract.pdf in the output (bin/(debug|release)/netcoreapp2.1 folder.

**Writing the source code**

**1. Create new console application.**

* Run Visual Studio
* File -> Create -> Console Application (.Net Core)

**2. Modify class Program.**

2.1. In the function Main() call WriteLine() to output the text you want to see in the console:

*Console.WriteLine("Gehtsoft.PDFFlow.Demos");*

*Console.WriteLine("----------------------");*

*Console.WriteLine("CONTRACT");*

*Console.WriteLine("----------------------");*

Call method for contract generation:

*GenerateExample();*

After generation is completed, ask user to press any key:

*Console.WriteLine("");*

*Console.WriteLine("Press any key for exit...");*

*Console.ReadKey();*

2.2. In the same class Program define the expression for GenerateExample() method, so it will call GenerateExample() from ContractExample class:

*private static void GenerateExample() => ContractExample.GenerateExample();*

**3. Generate Contract**

3.1. Under the Model subfolder create new **class ContractDictionary**:

*namespace Gehtsoft.PDFFlow.Contract.Model*

*{*

*[DataContract]*

*public class ContractDictionary*

*{*

*[DataMember]*

*public string Key { get; set; }*

*[DataMember]*

*public string Value { get; set; }*

*}*

*}*

3.2. Create new **class ContractExample**.

3.3. **Define fields** of the ContractExample class.

Path to the output file:

*private static readonly string PdfFile;*

Path to the project directory:

*private static readonly string ProjectDir;*

Path to the input template text of the contract:

*private static readonly string ContractTextFile;*

Path to the JSON file, containing values to be replaced in the template:

*private static readonly string DictionaryJsonFile;*

JSON content:

*private static readonly string DictionaryJsonContent;*

Different fonts:

*private static readonly Font DocumentFont;*

*private static readonly Font ItalicFont;*

*private static readonly Font TitleFont;*

Contract text:

*public static List<string> ContractData { get; set; }*

Dictionary data:

*public static List<ContractDictionary> DictionaryData { get; }*

3.4. Create constructor *static ContractExample()*

and **initialize fields**:

Initialize path variables:

*PdfFile = Path.Combine(Environment.CurrentDirectory, "Contract.pdf");*

*ProjectDir = Directory.GetCurrentDirectory();*

*ContractTextFile = Path.Combine(ProjectDir, "Content", "contract.txt");*

*DictionaryJsonFile = Path.Combine(ProjectDir, "Content", "contract\_dictionary.json");*

Read contract data:

*ContractData = File.ReadAllLines(ContractTextFile).ToList();*

Read dictionary data:

*DictionaryJsonContent = File.ReadAllText(DictionaryJsonFile);*

*DictionaryData = JsonConvert.DeserializeObject<List<ContractDictionary>>(DictionaryJsonContent);*

Initialize fonts:

*DocumentFont = new Font*

*{*

*Name = FontNames.Helvetica,*

*Size = 16f,*

*Color = Color.Black*

*};*

*ItalicFont = DocumentFont.Clone(); ItalicFont.Name = "Times-Italic";*

*TitleFont = DocumentFont.Clone(); TitleFont.Name = "Courier-Bold"; TitleFont.Size = 28f;*

3.5. Create method **GenerateExample**().

Create new document by calling method New() of the DocumentBuilder. Then call method AddContractSection, which will be described later. After the document structure is finished, call Build() with the desired file path as a parameter:

*internal static void GenerateExample()*

*{*

*Console.WriteLine("Generating file " + PdfFile);*

*if (File.Exists(PdfFile))*

*File.Delete(PdfFile);*

*DocumentBuilder*

*.New()*

*.AddContractSection()*

*.Build(PdfFile);*

*}*

3.6. **Add contract sections.** Create extention method of the builder:

*internal static DocumentBuilder AddContractSection(this DocumentBuilder builder)*

*{*

*}*

Call AddDefaultSection() which will create the first section with default section’s parameters and then change needed parameters (section margins in our case):

*builder.AddDefaultSection(s =>*

*{*

*s.SetMargins(40);*

Then call sub-methods which will generate parts of our document in the current section:

*AddContractTitle(s);*

*ReplaceKeyWordsFromDictionary();*

*AddContractText(s);*

*AddFooter(s);*

*});*

Add second section for the last page with signatures:

*builder.AddDefaultSection(s =>*

*{*

*s.SetMargins(40);*

*AddSignatures(s);*

*AddLastFooter(s);*

*});*

Method AddContractSection() should return DocumentBuilder, so we could call builder’s method Build() as described in 3.4:

*return builder;*

3.7. **Add contract title** as a Paragraph. Just pass paragraph’s text as a string parameter of AddParagraph(). It returns Paragraph, so you can then set paragraph’s alignment, margins and font:

*internal static void AddContractTitle(Section s)*

*{*

*s.AddParagraph("House Rental Contract")*

*.SetAlignment(HorizontalAlignment.Center)*

*.SetMargins(0, 0, 0, 20)*

*.SetFont(TitleFont);*

*}*

3.8. **Replace keywords in the template** text with the values from dictionary:

*internal static void ReplaceKeyWordsFromDictionary()*

*{*

*ContractData = ContractData*

*.ConvertAll(s => s.Replace("{paragraphbreak}", " "));*

*foreach (ContractDictionary item in DictionaryData)*

*{*

*ContractData = ContractData.ConvertAll(s =>*

*s.Replace("{" + item.Key + "}", item.Value));*

*}*

*}*

3.9. **Add contract text to the current section**. Add Paragraph to a current section, set it’s font, alignment and margins.

Then call one of the AddText() methods which receive array of strings as a parameter. In this case each string will be started from the new line, so you don’t need to add several parameters to render several lines of text:

*internal static void AddContractText(Section s)*

*{*

*foreach (var paragraph in ContractData)*

*{*

*s.AddParagraph(paragraph)*

*.SetFont(DocumentFont)*

*.SetJustifyAlignment(true);*

*}*

*}*

3.10. **Add footer with page number and tabulation**.

Method AddSinglePage() of the Layout adds footer to every page. You could also call AddOddPage() to add this footer only to odd pages, or AddEvenPage() for only even pages. Method AddRepeatingArea() adds area to the current layout (footer). You can add as many areas as you need. We need one here.

Method AddPageNumber() will render automatically calculated number of the page.

As we need a line for the signature and a text “(Initials)” under it, center-aligned regarding to the line, we use **word-like tabulation** for this purpose. There are two methods of indication tabulation positions in the Paragraph: AddTabulation (sets absolute position in pixels) and AddTabulationInPercent (sets position in percents). In the first row of the paragraph we need underlined place for the signature, so we jump to the position of 60 % with empty leading and then jump to the position of 100 % using BottomLine leading. As we need text “(Initials)” to be center-aligned regarding to this line, we jump to center-type tabulation positioned at 80 %:

*internal static void AddFooter(Section s)*

*{*

*s.Layout*

*.AddSinglePage()*

*.AddRepeatingArea(s.Page, 45, true, areaConfig: (area) =>*

*{*

*area.AddItem<Paragraph>(p =>*

*{*

*p.SetMargins(20, 0, 0, 0)*

*.AddPageNumber()*

*.AddTabSymbol().AddTabSymbol()*

*.AddTabulationInPercent(60, TabulationType.Left)*

*.AddTabulationInPercent(100, TabulationType.Right, TabulationLeading.BottomLine);*

*});*

*area.AddItem<Paragraph>(p =>*

*{*

*p.SetFont(ItalicFont)*

*.AddTabSymbol().AddTextToParagraph("(Initials)")*

*.AddTabulationInPercent(80, TabulationType.Center);*

*});*

*});*

*}*

3.11. Add signatures in the **second section**. Use **tabulation in percents** as described above. In the first row of paragraph we will have a line from the beginning to 20% and another line from 60% to 100%. In the second row of paragraph there will be a text "(Date)" left-aligned around position of 10% and a text "(Renter)" center-aligned around 80 %:

*internal static void AddSignatures(Section s)*

*{*

*s.AddParagraph()*

*.AddTabSymbol()*

*` .AddTabulationInPercent(20, TabulationType.Right, TabulationLeading.BottomLine)*

*.AddTabSymbol().AddTabSymbol()*

*.AddTabulationInPercent(60, TabulationType.Left)*

*.AddTabulationInPercent(100, TabulationType.Right, TabulationLeading.BottomLine);*

*s.AddParagraph()*

*.SetFont(ItalicFont)*

*.AddTabSymbol().AddTextToParagraph("(Date)")*

*.AddTabulationInPercent(10, TabulationType.Center)*

*.AddTabSymbol().AddTextToParagraph("(Renter)")*

*.AddTabulationInPercent(80, TabulationType.Center);*

*s.AddParagraph(" ");*

*s.AddParagraph()*

*.AddTabSymbol()*

*.AddTabulationInPercent(20, TabulationType.Right, TabulationLeading.BottomLine)*

*.AddTabSymbol().AddTabSymbol()*

*.AddTabulationInPercent(60, TabulationType.Left)*

*.AddTabulationInPercent(100, TabulationType.Right, TabulationLeading.BottomLine);*

*s.AddParagraph()*

*.SetFont(ItalicFont)*

*.AddTabSymbol().AddTextToParagraph("(Date)")*

*.AddTabulationInPercent(10, TabulationType.Center)*

*.AddTabSymbol().AddTextToParagraph("(Landlord)")*

*.AddTabulationInPercent(80, TabulationType.Center);*

*}*

3.12. Add **footer with page number** in the second section:

*internal static void AddLastFooter(Section s)*

*{*

*s.Layout.AddSinglePage().AddRepeatingArea(s.Page, 55, true,*

*areaConfig: (area) =>*

*{*

*area.AddItem<Paragraph>(p =>*

*{*

*p.SetMargins(20, 0, 0, 0).AddPageNumber();*

*});*

*});*

*}*

4. Generated **PDF file** should look as shown below:

