PDF .NET C#

C# PDF Document Parser

Sergiy Stoyan

4.95/5 (14 votes)

Dec 26, 2018 GPL3 2 min read 37841

A .NET toolset for building PDF parsers

Download source code from Github

Read documentation.

Idea

The main approach of parsing by `PdfDocumentParser` is based on finding certain text or image fragments on a PDF page and then extracting text/images located and sized relatively to those fragments.

Within this scope, `PdfDocumentParser` is capable of the following:

- search/extract text represented by PDF entities
- search/extract text obtained by OCR
- search/compare/extract page fragments as images

As a part of parsing routine, `PdfDocumentParser` allows checking custom conditions on a PDF page to decide which actions should be taken on it.

`PdfDocumentParser` facilitates parsing graphical tables to data arrays.

For more details refer to the documentation.

Template Editor

To be able to parse a PDF document, `PdfDocumentParser` must be supplied with a parsing template corresponding to the document's layout. For this goal, `PdfDocumentParser` provides Template Editor that allows creating and debugging parsing templates in an easy manner in GUI. Template Editor should be invoked by the hosting application.

Application

An application based on `PdfDocumentParser` has to care about the following main aspects:

- provide storage and management of parsing templates
- allow a user to create and modify templates with Template Editor
- implement a custom algorithm of processing PDF files:
  - choose a template to be applied on a PDF page
  - process data parsed by the chosen template

An example of such an app is `SampleParser` project in `PdfDocumentParser` solution.

Algorithm

Some basic algorithm of processing a PDF file page by page would be the following:

C# Shrink ▲

```csharp
//Pseudo-code: processing a PDF file where every page requires choosing new template.
//Note: The classes and methods are not real and serve for simplicity and clarity only.

foreach(page in pdfFile)
{
    //find the right template for the page
    if(PdfDocumentParser.ActiveTemplate == null)
    {
        foreach(template in templates)
        {
            PdfDocumentParser.ActiveTemplate = template;
            if(PdfDocumentParser.IsCondition(page, "RightTemplateForPage"))
                break;
            PdfDocumentParser.ActiveTemplate = null;
        }
    }

    if(PdfDocumentParser.ActiveTemplate == null)
    {
        logWarning("No template matched to page: " + page.Number);
        continue;
    }

    //applying the chosen template to the page
    object value1 = PdfDocumentParser.GetValue(page, "field1");
    //doing something with value1...
    <...>
    object value2 = PdfDocumentParser.GetValue(page, "field2");
    //doing something with value2...
    <...>
}
```

Notice that conditions like '`RightTemplateForPage`' are introduced and predetermined by the custom application. `PdfDocumentParser` only provides the facility of checking them. Because of that, the parsing logic can be as complex as needed.

How exactly a condition is checked is up to the template because every template provides its own definition for it. A condition definition is a boolean expression of what was found and what was not found on PDF page.

For instance, when processing invoices, '`RightTemplateForPage`' might check if the company's name or logo is located on the PDF page and thus, detect if the page corresponds to the template.

Creating a VS Solution

Do not download the latest code as is in a branch because it may be in development. Instead, go to releases and download the latest (pre-)release source code. Find *SampleParser.sln* there and open it in Visual Studio. It will give a complete example of using `PdfDocumentParser` that you can modify according to your requirements.

Steps in Visual Studio if building from scratch without `SampleParser`:

- Create your project.
- Add `PdfDocumentParser` project to the solution.
- Reference `PdfDocumentParser` in your project.
- Update nuget packages for the solution.
- Start developing your parser using `PdfDocumentParser` API.

Enjoy!

History

- 12th February, 2020: Initial version