D&TS GmbH
Wilhelmstraße 41-43, D-58332 Schwelm
T. +49 (0) 2336 42828-0 * F. +49 (0) 2336 42828-28
E-Mail: info@dundts.com * www.dundts.com

Anwenderdokumentation

KeyValuePairLib

Version: 1.0 (pdf)

Inhaltsverzeichnis

Historie der Versionen

Version	Datum	Autor	Änderungsgrund / Bemerkungen
1.0	27.09.2020	Patrick Hildebrandt	PDF-Funktionalität implementiert
1.1			
1.2			
1.3			

1 Allgemeines

1.1 Anwendungsszenarien

Diese Klassenbibliothek extrahiert Schlüssel-Wert-Paare aus semi- oder unstrukturierten Datenblättern und stellt die Ergebnisse als generische Eigenschaften zur weiteren Verarbeitung bereit. Zusätzlich können diese Eigenschaften als Objekte zurückgegeben werden.

Generell sind die integrierten Typen der Klassenbibliothek öffentlich gestaltet, um einfache Refaktorisierung und Steuerung durch übergeordnete Projekte zu ermöglichen.

1.2 Unterstützte Dateitypen

In der ersten Version unterstützt die Klassenbibliothek den Dateityp PDF. Der Algorithmus basiert auf dem Prinzip, dass Felder in PDF-Dateien anhand ihrer räumlichen Trennung voneinander wie Zellen interpretiert und extrahiert werden.

Dieses Prinzip grenzt sich klar ab von den im PDF-Standard vorgesehenen Formularfeld-Attributen und ist somit auch nicht von diesen abhängig.

1.3 Weiterentwicklung

Die Dateitypen CSV, XLSX und XML sind bereits in grundlegenden Strukturen berücksichtigt worden. Da PDF-Dateien bei der Verarbeitung in eine tabellarische Form überführt und als DataTable gespeichert werden, ermöglicht dies für zukünftige Erweiterungen eine verhältnismäßig einfache Adaption des Extraktionsalgorithmus auf die oben genannten Dateitypen.

Dies hat nur Gültigkeit, sofern die entsprechenden Dateien einen typografischen Aufbau von der Art eines Datenblattes aufweisen und einzelne Materialien mit ihren Eigenschaften beschreiben.

1.4 Abhängigkeiten

1.4.1 Backend

Komponente:	Abhängigkeiten:
KeyValuePairLib.sln	.NET 7.0
Config.cs	Json.NET 13.0.3 (MIT License)
PdfFileData.cs	Spire.PDF 9.7.17 (free trial without time limitation: https://github.com/e-iceblue/Spire.PDF-for-.NET/blob/master/README.md)

1.4.2 Frontend

Komponente:	Abhängigkeiten:

2 Anwendung

2.1 Schnellstart

Konfigurationsdatei:

```
string configDefence = @"..\..\..\..\ConsoleTest\config\configDefence.cfg";
```

Quelldatei:

```
string defence1 = @"C:\Users\PatrickHildebrandt\Desktop\[redacted].pdf";
```

Instanziierung:

```
KeyValuePairReader kvprDefence1 = new(configDefence, defence1);
```

```
string defence1 = @"C:\Users\PatrickHildebrandt\Desktop\[redacted].pdf";
string configDefence = @"..\..\..\..\ConsoleTest\config\configDefence.cfg";
KeyValuePairReader kvprDefence1 = new(configDefence, defence1);
```

2.2 CONSOLE

Diese Methode stellt ein Debugging- und Inspektions-Tool dar und gibt die Ergebnisse der jeweiligen Sub-Methoden in Reihenfolge der Verarbeitungsschritte in der Konsole aus. Ohne Parameter werden alle Methoden ausgegeben:

```
kvprDefence1.CONSOLE();
kvprDefence1.CONSOLE(0, 11);
```

```
kvprDefence1.CONSOLE();
kvprDefence1.CONSOLE(0, 11);
```

```
0 =   Config.CONSOLE_Config()
1 =   Separators.CONSOLE_SeparatorCollection()
2 =   _fileData.CONSOLE_pdfDocString()
3 =   _fileData.CONSOLE_pdfDocEmptyRowsRemoved()
4 =   _fileData.CONSOLE_pdfDocFieldsDelimited()
5 =   _fileData.CONSOLE_Rows()
6 =   _fileData.CONSOLE_fields()
7 =   _fileData.CONSOLE_indexedFields()
8 =   _fileData.CONSOLE_FieldTable()
9 =   CONSOLE_FieldTableMethods()
10 =  CONSOLE_FieldTableMethodsExpand()
11 =  KeyValuePairs.CONSOLE_GroupedKeyValuePairs()
12 =  KeyValuePairGroups.CONSOLE_GroupedKeyValuePairs()
```

```
CONSOLE_Eingang / Input::
CONSOLE_GroupedKeyValuePairs:
#############################

°Eingangsspannung (Ue) / Input voltage "Ein" / "On"°
                                °1[redacted]V°
°Eingangsstrom / Input current°
                                °C[redacted]V°
°Überspannungsschutz / Overvoltage protection°
                                °S[redacted]V°
°Ansteueranzeige / Control display°
                                °L[redacted]n°
°Polung / Polarity°
                                °b[redacted]n°
```

2.3 Expliziter Zugriff auf Eigenschaften

Nach der Instanziierung können die Ergebnisse der Extraktion als Eigenschaften explizit angesprochen werden:

```
Console.WriteLine(kvprEnergie1.KeyValuePairs["Prozessauswahl *"]);
Console.WriteLine(kvprEnergie1.KeyValuePairs["Formular-ID"]);
Console.WriteLine(kvprEnergie1.KeyValuePairs["Materialnummer"]);
Console.WriteLine(kvprEnergie1.KeyValuePairs["Materialart *"]);
```

```
string energie1 = @"..\..\..\..\ConsoleTest\demoPDFs\energie1.Pdf";
string configEnergie = @"..\..\..\..\ConsoleTest\config\configEnergie.cfg";
KeyValuePairReader kvprEnergie1 = new(configEnergie, energie1);
Console.WriteLine(kvprEnergie1.KeyValuePairs["Prozessauswahl *"]);
Console.WriteLine(kvprEnergie1.KeyValuePairs["Formular-ID"]);
Console.WriteLine(kvprEnergie1.KeyValuePairs["Materialnummer"]);
Console.WriteLine(kvprEnergie1.KeyValuePairs["Materialart *"]);
```

2.4 Rückgabe

Über den expliziten Zugriff auf Eigenschaften hinaus stehen mehrere Möglichkeiten zur Rückgabe der Ergebnisse zur Verfügung. Zwecks Vollständichkeit werden hier alle Möglichkeiten anhand ihres Quelltextes aufgeführt:

```
public object this[string name]
{
      get => (_properties as IDictionary<string, object>)[name];
      set => (_properties as IDictionary<string, object>)[name] = value;
}
```

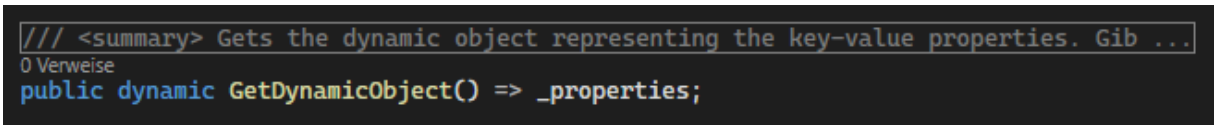
```
25 Verweise
public class GroupedKeyValuePairs
{
    // ! FIELDS
    fields

    // ! PROPERTIES
    #region properties
    /// <summary> Gets or sets the value as object associated with the specified key ...
    12 Verweise
    public object this[string name]
    {
        get => (_properties as IDictionary<string, object>)[name];
        set => (_properties as IDictionary<string, object>)[name] = value;
    }
    #endregion
```

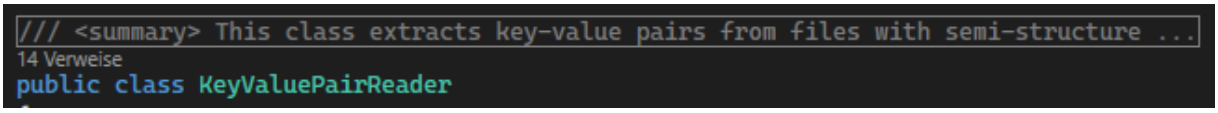
```
public IDictionary<string, object> GetProperties() => _properties as
IDictionary<string, object>;
```

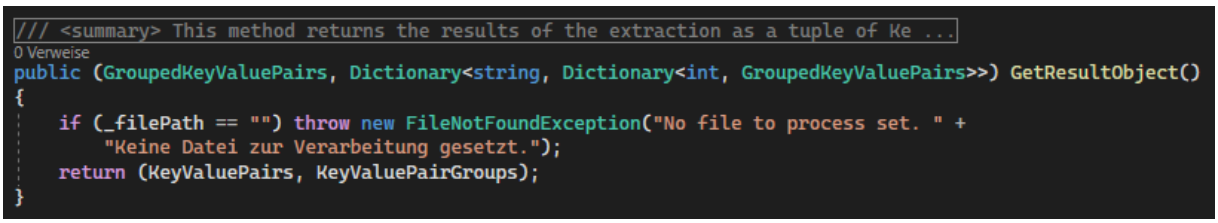
```
/// <summary> Gets a Dictionary of all key-value properties in the collection. G ...
1 Verweis
public IDictionary<string, object> GetProperties() => _properties as IDictionary<string, object>;
```

```
public dynamic GetDynamicObject() => _properties;
```

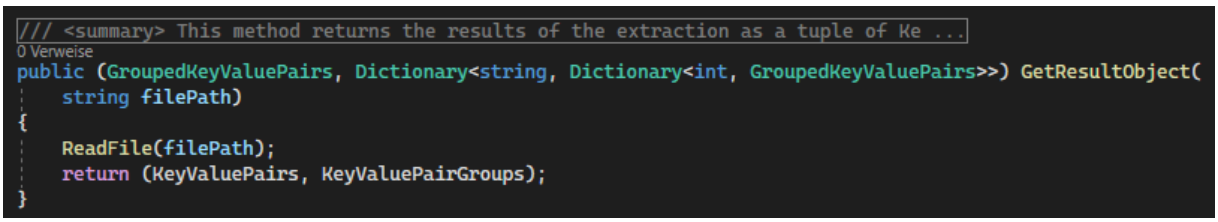



```
public (GroupedKeyValuePairs, Dictionary<string, Dictionary<int,
GroupedKeyValuePairs>>) GetResultObject()
{
        if (_filePath == "") throw new FileNotFoundException("No file to
        process set. Keine Datei zur Verarbeitung gesetzt.");
        return (KeyValuePairs, KeyValuePairGroups);
}
```

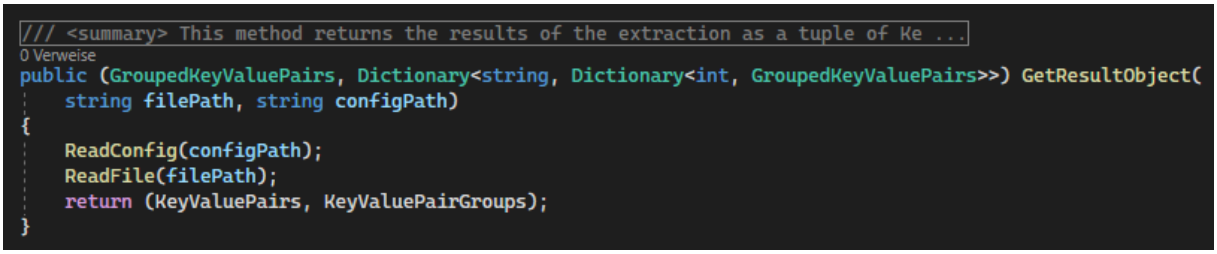





```
public (GroupedKeyValuePairs, Dictionary<string, Dictionary<int,
GroupedKeyValuePairs>>) GetResultObject(string filePath)
{
        ReadFile(filePath);
        return (KeyValuePairs, KeyValuePairGroups);
}
```



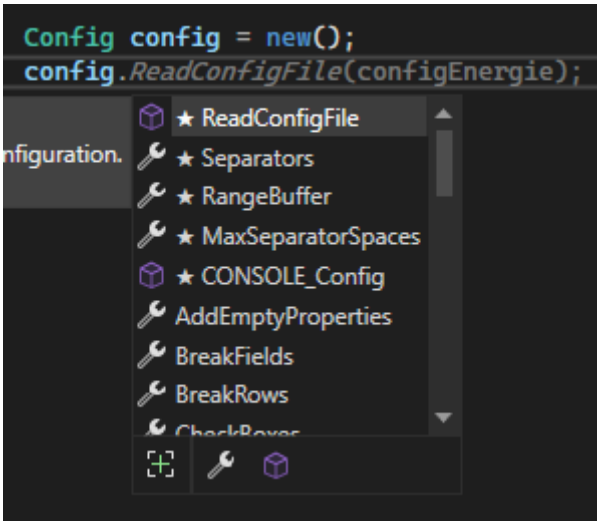

```
public (GroupedKeyValuePairs, Dictionary<string, Dictionary<int,
GroupedKeyValuePairs>>) GetResultObject(string filePath, string configPath)
{
        ReadConfig(configPath);
        ReadFile(filePath);
        return (KeyValuePairs, KeyValuePairGroups);
}
```



3 Konfiguration

Im Folgenden wird die Bedeutung der einzelnen Zeilen der Konfigurationsdatei anhand des Templates „configTemplate.cfg“ erläutert. Hierbei handelt es sich um eine Konfigurationsdatei im JSON-Format, welche durch Konstruktoren und Methoden eingelesen werden kann. Die hierbei erläuterten Zeilen bilden die Eigenschaften der Klasse Config.cs ab. Diese Eigenschaften können äquivalent hierzu ebenfalls mit der Punktnotation von übergeordneten Projekten angesprochen werden:

```
Config config = new();
```



configTemplate.cfg:

```
{
```

```
  "MaxWhiteSpaces": 3,
```

Hier wird definiert ab wie vielen Leerzeichen zwischen Feldern diese als leerer Raum interpretiert werden. Standard ist 3, Minimum ist 2, sollte jedoch vermieden werden, da dies auch Eingabefehler von doppelten Leerzeichen berücksichtigt.

```
  "RangeBuffer": 3,
```

Hier wird ein Puffer definiert, wie viele Zeichen bei horizontalem Versatz zwischen vertikalen Schlüssel- und Wert-Feldern toleriert werden. Standard ist 3. Ein zu großer Puffer kann benachbarte Felder miteinbeziehen.

```
  "MinSeparatorSpaces": 2,
```

Hier wird die minimale Anzahl von Trennzeichen definiert für Schlüssel-Wert-Paare, welche durch eine Sequenz von Trennzeichen in einem Feld verbunden sind. Standard ist 2. Höherer Wert steigert die Performance.

```
  "MaxSeparatorSpaces": 30,
```

Hier wird die maximale Anzahl von Trennzeichen definiert für Schlüssel-Wert-Paare, welche durch eine Sequenz von Trennzeichen in einem Feld verbunden sind. Standard ist 30. Niedrigerer Wert steigert die Performance.

```
  "CutTop": 0,
```

Hier wird definiert, wie viele Pixel von PDF-Seiten oberhalb abgeschnitten werden. Standard ist 0.

```
  "CutBottom": 0,
```

Hier wird definiert, wie viele Pixel von PDF-Seiten unterhalb abgeschnitten werden. Standard ist 0.

```
  "CutLeft": 0,
```

Hier wird definiert, wie viele Pixel von PDF-Seiten links abgeschnitten werden. Standard ist 0.

```
  "CutRight": 0,
```
Hier wird definiert, wie viele Pixel von PDF-Seiten rechts abgeschnitten werden. Standard ist 0.

```
  "AddEmptyProperties": true,
```
Hier wird definiert, ob Eigenschaften ohne gefundene Werte zu den Ergebnissen hinzugefügt werden sollen oder nicht. Standard ist true.

```
  "SearchKeys": [
```
Hier werden die SearchKey-Objekte definiert zu denen Werte aus Dateien extrahiert werden.

```
    {"Key": "Schlüssel1", "Direction": 0, "Appendix": ["Anhang"]},
```
Die Suche wird horizontal durchgeführt. Zugewiesen dem Wert 0.

```
    {"Key": "Schlüssel2", "Direction": 1, "Appendix": ["Anhang"]},
```
Die Suche wird vertikal durchgeführt. Zugewiesen dem Wert 1.

```
    {"Key": "Schlüssel3", "Direction": 2, "Appendix": ["Anhang"]},
```
Die Suche wird über mehrere Spalten durchgeführt. Zugewiesen dem Wert 2.

```
    {"Key": "Schlüssel4", "Direction": 4, "Appendix": ["Anhang"]},
```
Die Suche wird über mehrere Zeilen durchgeführt. Zugewiesen dem Wert 4.

```
    {"Key": "Schlüssel5", "Direction": 8, "Appendix": ["Anhang"]},
```
Die Suche erfolgt über Spalten und Zeilen, ähnlich einer Tabelle. Zugewiesen dem Wert 8.

```
    {"Key": "Schlüssel6", "Direction": 16, "Appendix": ["Anhang"]},
```
Die Suche erfolgt über die Erkennung von Checkboxen in und nach Schlüssel-Feldern. Zugewiesen dem Wert 16.

```
    {"Key": "Schlüssel7", "Direction": 32, "Appendix": ["Anhang"]},
```
Die Suche erfolgt über die Erkennung von Radio-Buttons in Wert-Feldern. Zugewiesen dem Wert 32.

```
    {"Key": "Schlüssel7", "Direction": 64, "Appendix": ["Anhang"]},
```
Die Suche erfolgt über die Erkennung von Trennzeichen-Sequenzen in Feldern. Zugewiesen dem Wert 64.

```
    {"Key": "Schlüssel8", "Direction": 128, "Appendix": ["Anhang"]}
```
Reserviert für die abstrakte Implementierung von Sonderfällen in abgeleiteter Klasse, die nicht mit existierenden Konfigurationsregeln abgebildet werden können. Zugewiesen dem Wert 128.

```
  "SearchKeyGroups": {
    "repetingProperty1": [
```
Namen für SearchKeyGroups müssen aus Feldern im Dokument ausgewählt werden, die wie eine Überschrift für ihre Schlüssel-Wert-Kombinationen fungieren. Außerdem sind sie notwendig, um identische SearchKeys zu unterscheiden und die Ergebnisse entsprechend zu separieren. Weiterhin lassen sich hiermit auch Kategorieren bzw. Abschnitte im Dokument voneinander abgrenzen.

```
      {"Key": "Schlüssel1:", "Direction": 0, "Appendix": ["Anhang", "über 2
Zeilen"]},
```
Unter Anhängen sind Unterschriften von Schlüsseln zu verstehen (z.B. englische Bezeichnung unter deutschem Schlüssel). Anhänge sind notwendig, wenn sie in Ergebnissen benötigt werden. Weiterhin werden sie benötigt, um sich selbst im Dokumentenfluss zu überspringen.

```
      {"Key": "Schlüssel8:", "Direction": 128, "Appendix": ["Anhang", "über 2
Zeilen"]}
    ],
    "repetingProperty2": [
      {"Key": "Schlüssel1:", "Direction": 0, "Appendix": ["Anhang","über","3
Zeilen"]},
```

Zeilenumbrüche in einem Anhang müssen mit separaten Einträgen für jede Zeile abgebildet werden.

```
      {"Key": "Schlüssel8:", "Direction": 128, "Appendix": ["Anhang","über","3
Zeilen"]}
    ]
  },
```

HINWEIS: `Ignorieren` sollte vorrangig vor `Unterbrechungen` verwendet werden. `Unterbrechungen` beenden Operationen, während `Ignorieren` fortfährt und nach weiteren Werten sucht.

```
  "IgnoreFields": [
    " are ignored by skipping "
```

Diese Felder werden dadurch ignoriert, dass ihre enthaltenden Zeilen bzw. ihre Zeichenkette enthaltende Felder übersprungen werden. Speziell für Felder gedacht, die nicht eindeutig identifiziert werden können bzw. ein Vorkommen der Zeichenkette enthalten.

```
  ],
  "IgnoreRows": [
    "These rows are ignored by skipping them as a whole."
```

Diese Zeilen werden dadurch ignoriert, dass sie als Ganzes übersprungen werden. Speziell für Überschriften oder Felder gedacht, welche sich durch ihren Versatz zwischen Key- und Value-Zeilen schieben. Diese Zeichenketten müssen aus dem .CONSOLE(5)-Report ausgewählt werden um regulären Ausdrücken zu entsprechen.

```
  ],
  "BreakFields": [
    " stop fields are defined here "
```

Hier werden absolute Stoppfelder definiert für Szenarien, bei denen der Algorithmus über mehrfache Zeilen- oder Spalten-Werte hinaus iteriert mangels einer anderen identifizierbaren Abgrenzung.

```
  ],
  "BreakRows": [
    "Absolute stop rows are defined here for scenarios where the algorithm
iterates beyond multiple row values."
```

Hier werden absolute Stoppzeilen definiert für Szenarien, bei denen der Algorithmus über mehrfache Zeilenwerte hinaus iteriert mangels einer anderen identifizierbaren Abgrenzung. Diese Zeichenketten müssen aus dem .CONSOLE(5)-Report ausgewählt werden um regulären Ausdrücken zu entsprechen.

```
  ],
  "CheckBoxes": [
    "✓"
  ],
  "RadioButtons": [
    "◉"
  ],
```

Checkboxen und Radiobuttons können nur erkannt werden, wenn ihre Zeichen lesbar und hier definiert sind. Checkboxen sind für ein Vorkommen im oder nach dem Schlüssel gedacht, während Radio-Buttons für mehrere Werte gedacht sind, die das Radio-Button-Zeichen enthalten.

```
  "Separators": [
    "=",
    ":",
    "."
  ]
```

Separatoren stellen die Basis dar für die unter SearchKeys genannten Trennzeichen-Sequenzen. Diese werden automatisch generiert aus den Separatoren und den Konfigurationswerten „`MaxWhiteSpaces`“ „`MinSeparatorSpaces`“ und “`MaxSeparatorSpaces`“.

Beispiele für Trennzeichen-Sequenzen und Separatoren:

'Key: Value' => ':' = Separator

'Key ..... Value' => '.' = Separator

```
}
```