Artefact

Yann Baës - Pierre Corbel - Amandine Watrelos

Université de Lille 1 SCIENCES ET TECHNOLOGIES

UFR IEEA
Informatique, Electronique
Electrotechnique, Automatique

Phar

Plan

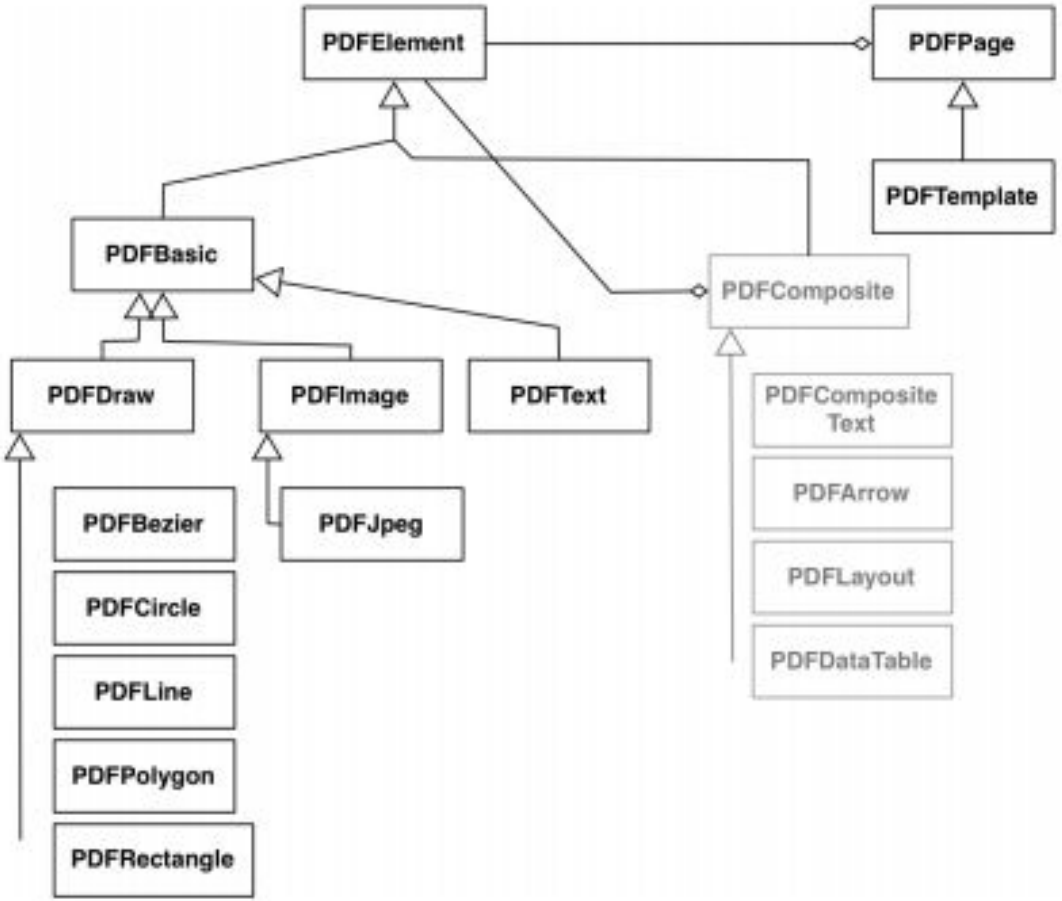
1) Le projet
2) Les classes de base
3) Exemples
4) Ce que nous pouvons apporter

Crée en février 2013 par Olivier Auverlot
et Guillaume Larcheveque (INRIA Lille).

> Puissant framework qui génère des PDFs.

> Simple à comprendre et utiliser.

> Personnalisable à souhaits et possibilités infinies.



- 12 packages et 106 classes
- Aucun ordre à respecter sur l'ordre de création des éléments
- Chaque élément est réutilisable
- De nombreux objets complexes déjà présents (courbes, tableaux…)
- Toutes les pages sont des streams

Classes de base

PDFAlignment

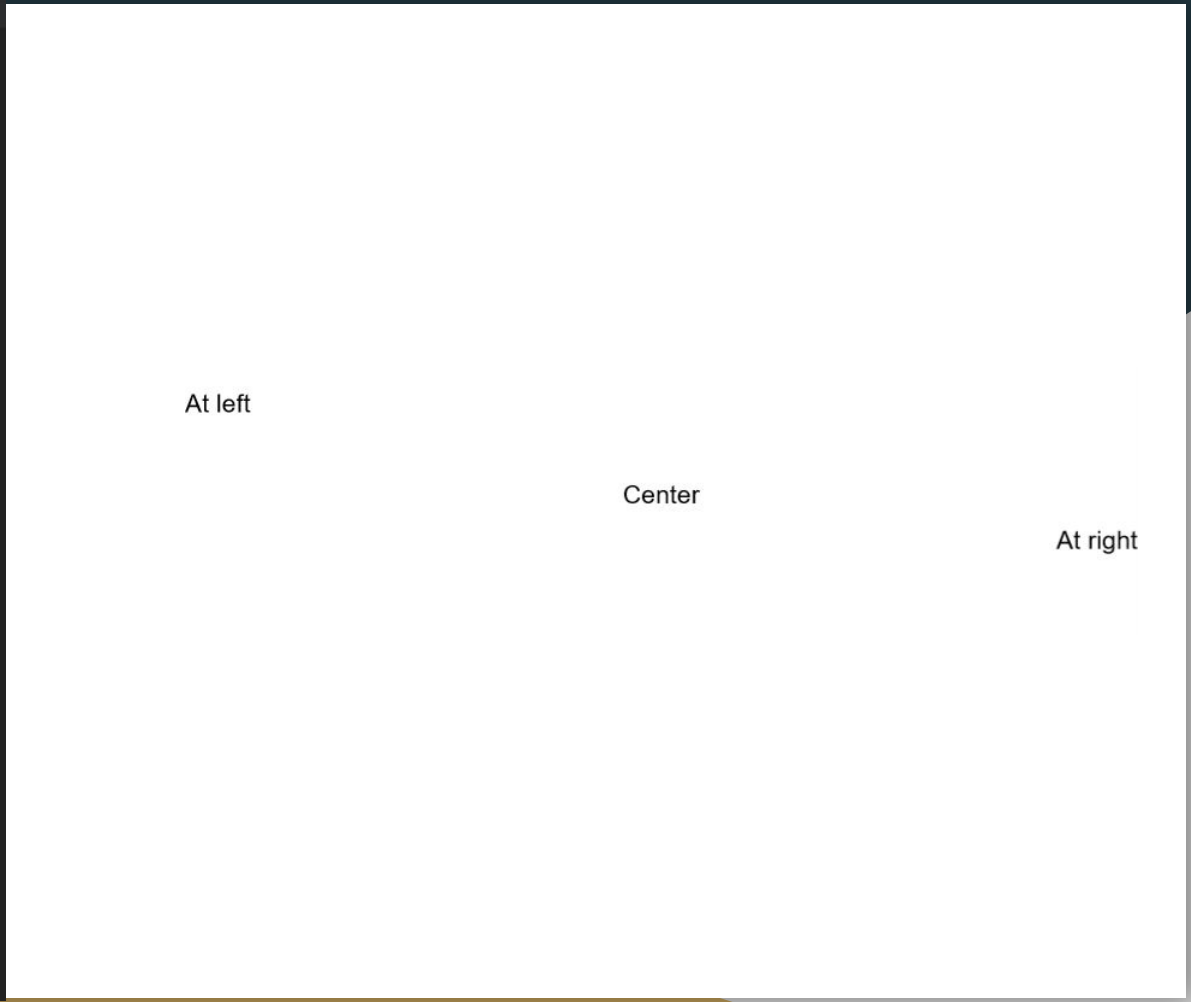
PDFColor

PDFDocument

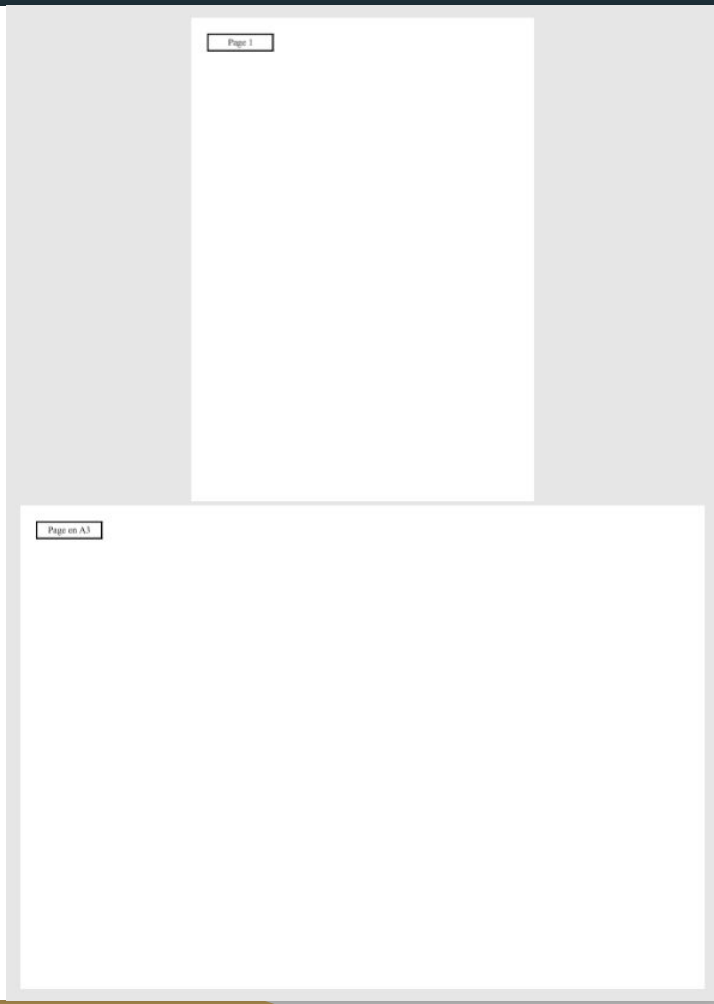
PDFMetaData

Exemples

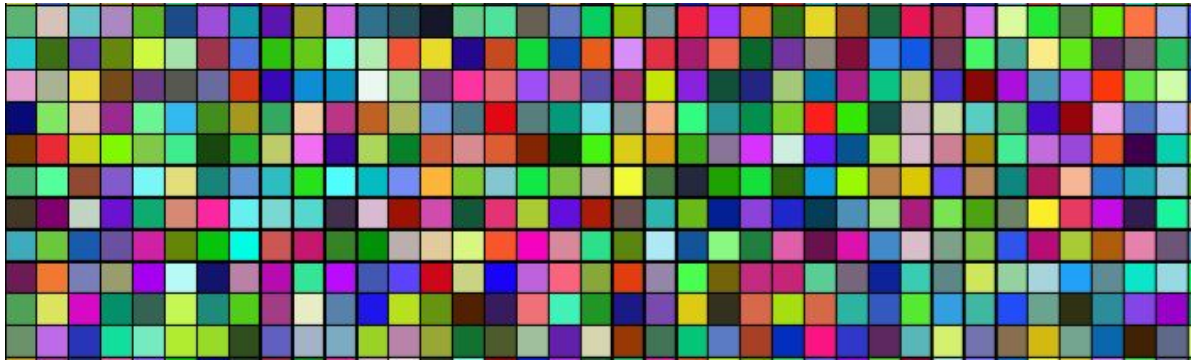
```
alignmentsTest: aStream
    "a simple demonstration of Artefact"

    | pdfdoc myFont aPage |
    pdfdoc := PDFDocument new.
    myFont := PDFHelveticaFont new fontSize: 16 pt.
    aPage := PDFPage new.
    aPage
        add:
            (PDFFormattedTextElement new
                font: myFont;
                dimension: 210 mm @ 10 mm;
                from: 0 mm @ 0 mm;
                text: 'At left').

        aPage
        add:
            (PDFFormattedTextElement new
                font: myFont;
                dimension: 210 mm @ 10 mm;
                from: 0 mm @ 20 mm;
                text: 'Center';
                alignment: (PDFAlignment center)).
                    aPage
        add:
            (PDFFormattedTextElement new
                font: myFont;
                dimension: 210 mm @ 10 mm;
                from: 0 mm @ 30 mm;
                text: 'At right';
                alignment: (PDFAlignment right)).
    pdfdoc add: aPage.
    pdfdoc exportTo: aStream
```



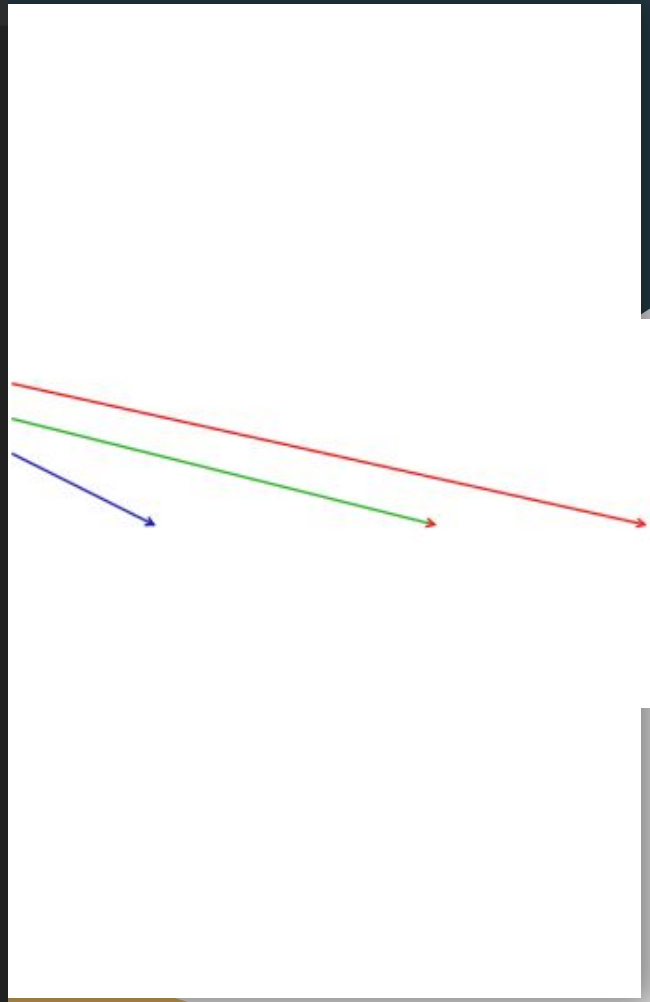
```
multiOrientationsTest: aStream
    "Create a document with two pages and two different orientations"

    | pdfdoc myBigFont firstPage secondPage myLandscapeFormat |
    pdfdoc := PDFDocument new.

    myLandscapeFormat := PDFA3Format new setLandscape.

    myBigFont := PDFTimesFont new fontSize: 16 pt.

    firstPage := PDFPage new.
    firstPage add: (PDFCellElement new
        from: 10mm@10mm;
        font: myBigFont;
        dimension: 40 mm @ 10 mm;
        text: 'Page 1').

    secondPage := PDFPage new format: myLandscapeFormat.

    secondPage add: (PDFCellElement new
        from: 10mm@10mm;
        font: myBigFont;
        dimension: 40 mm @ 10 mm;
        text: 'Page en A3').

    pdfdoc add: firstPage.
    pdfdoc add: secondPage.

pdfdoc exportTo: aStream
```



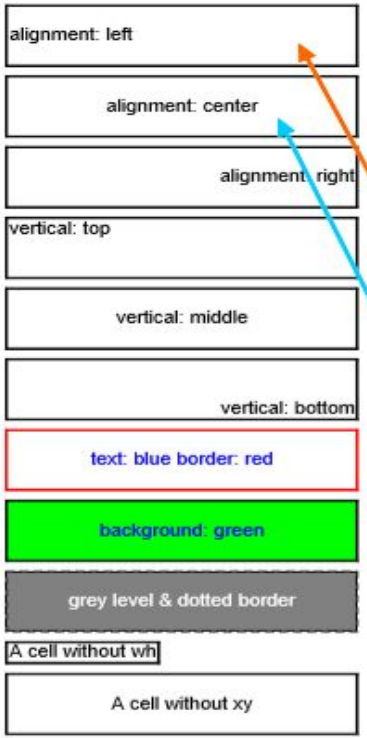
```
mosaiqueTest: aStream
   "generate a sample document with colored squares"

   | tileSize plage pdfdoc aPage |
   pdfdoc := PDFDocument new.
   tileSize:= 5mm@5mm.
   plage := 0 to: 255.
   aPage := PDFPage new.
   pdfdoc add: aPage.
   0 to: (aPage format width convertTo: tileSize x unit) / tileSize x do: [ :x |
      0 to: (aPage format height convertTo: tileSize y unit) / tileSize y do: [ :y |
         aPage
            add:
               ((PDFRectElement from: (x * tileSize x) @ (y * tileSize y) dimension: tileSize)
                  fillColor: (PDFColor r: plage atRandom g: plage atRandom b: plage atRandom)) ] ].
   pdfdoc exportTo: aStream
```



```
multiStyleArrowsTest: aStream
    "drawing arrows with differents styles"

    | pdfdoc aPage |
    pdfdoc := PDFDocument new.
    pdfdoc styleSheet > #redArrow at: #drawColor put: (PDFColor r: 243 g: 24 b: 24).
    pdfdoc styleSheet  at: #drawColor put: (PDFColor r: 24 g: 24 b: 180).
    pdfdoc styleSheet > #greenArrow at: #drawColor put: (PDFColor r: 24 g: 180 b: 24).
    pdfdoc styleSheet > #greenArrow > #head at: #drawColor put: (PDFColor r: 255 g: 0 b: 0).
    aPage := PDFPage new.

    aPage add: ((PDFArrowElement
        from: 10 mm @ 10 mm
        to: 100 mm @30 mm)
        style: #redArrow
    ).

    aPage add: (PDFArrowElement
        from: 10 mm @20 mm
        to: 30 mm @30 mm
    ).

    aPage add: ((PDFArrowElement
        from: 10 mm @ 15 mm
        to: 70 mm @30 mm)
        style: #greenArrow
    ).

    pdfdoc add: aPage.

pdfdoc exportTo: aStream
```



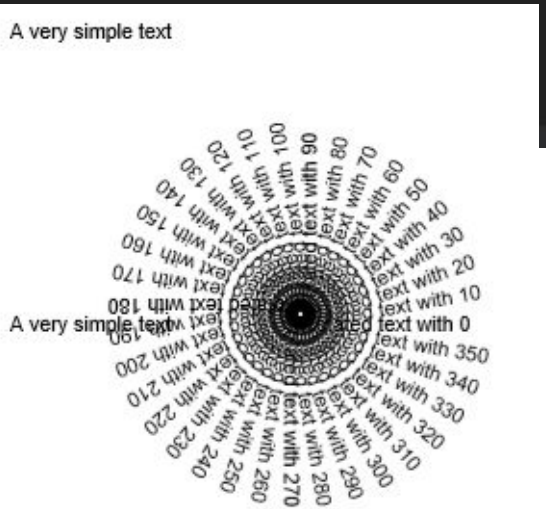
Demonstration of PDFCellStyle




```
verticalLayoutTest: aStream
   "self verticalLayoutTest"

   | pdfdoc myTitleFont aPage |
   pdfdoc := PDFDocument new.
   myTitleFont := PDFTimesFont new
      fontSize: 24 pt;
      bold: true.
   aPage := PDFPage new
      add:
         (PDFVerticalLayout new from: 10mm@10mm;
            add:
               (PDFCellElement new
                     from: 10 mm @ 5 mm;
                     dimension: 190 mm @ 10 mm;
                     fillColor: (PDFColor new setGreyLevel: 224);
                     text: 'Demonstration of PDFCellStyle';
                     font: myTitleFont;
                     alignment: PDFAlignment center);
            add:
               (PDFCellElement new
                     from: 10 mm @ 20 mm;
                     dimension: 50 mm @ 10 mm;
                     text: 'alignment: left';
                     alignment: PDFAlignment left);
            add:
               (PDFCellElement new
                     from: 80 mm @ 20 mm;
                     dimension: 50 mm @ 10 mm;
                     text: 'alignment: center');
```

Demonstration of PDFCellStyle

```
rotationTest: aStream
   "PDFDemos rotationTest"

   | pdfdoc aPage  |
   pdfdoc := PDFDocument new compression: false.

   aPage := PDFPage new.
   aPage add: (PDFTextElement from: 10 mm @ 10 mm text: 'A very simple text').

   0 to: 35do: [ :a |
      aPage add: (PDFTextElement new from: 50 mm @ 50 mm; text: ('Rotated text with ' , (a * 10) asString); rotation: (a * 10)).
   ].

   aPage add: (PDFTextElement from: 10 mm @ 50 mm text: 'A very simple text').
   pdfdoc  add: aPage.
   pdfdoc exportTo: aStream
```

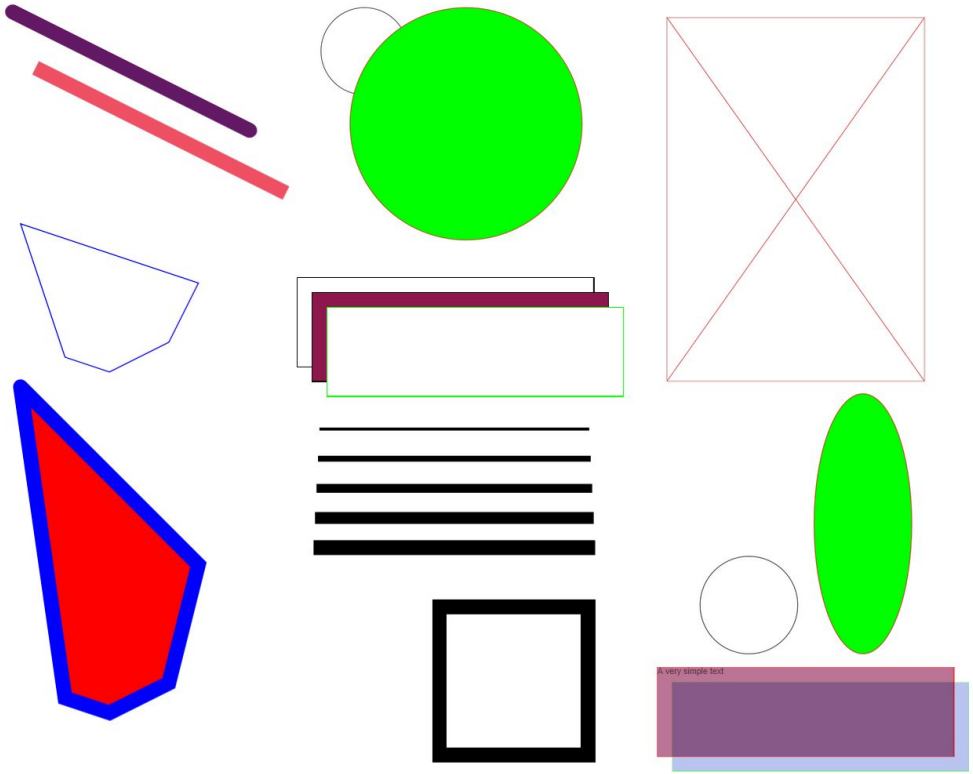


```
datatableWithCaptionsTest: aStream
   "generate a datatable with captions"

   | pdfdoc aPage |
   pdfdoc := PDFDocument new.
   pdfdoc styleSheet > #dataTableWithColoredCaption > #caption
      fillColor: (PDFColor r: 158 g: 158 b: 79);
      drawColor: (PDFColor r: 158 g: 158 b: 79).
   pdfdoc styleSheet > #dataTableWithColoredCaption margin: 4 pt.
   pdfdoc styleSheet > #dataTableWithColoredCaption > #cell alignment: PDFAlignment right.
   pdfdoc setFormat: PDFA4Format new.
   pdfdoc metaData title: 'Users report'.
   aPage := PDFPage new.
   aPage
      add:
         (PDFDataTableWithColumnsCaptionElement new
            captions: #('Name' 'Surname' 'email');
            data:
                  #(#('Smith' 'Peter' 'peter.smith@mail.org') #('Jones' 'Mickael' 'mickael.
'robert.washington@blif.com'));
            from: 10 mm @ 20 mm;
            dimension: 190 mm @ 60 mm;
            style: #dataTableWithColoredCaption;
            yourself).
   aPage
      add:
         (PDFDataTableWithRowsCaptionElement new
            captions: #('Name' 'Surname' 'email');
            data:
                  #(#('Smith' 'Jones' 'washington') #('Peter' 'Mickael' 'robert') #('peter.
'robert.washington@blif.com'));
            from: 10 mm @ 90 mm;
            dimension: 190 mm @ 60 mm).
   pdfdoc add: aPage.
   pdfdoc exportTo: aStream
```

Name	Surname	email
Smith	Peter	peter.smith@mail.org
Jones	Mickael	mickael.jones@epr.com
washington	robert	robert.washington@blif.com

Name	Smith	Jones	washington
Surname	Peter	Mickael	robert
email	peter.smith@mail.org	mickael.jones@epr.com	robert.washington@blif.com

Et encore plein de possibilités !



Ce que nous pourrions apporter au projet ?

- Prise en compte des caractères unicodes (chinois, arabe..)

- Amélioration de la documentation

- Compléter les tests non finis

En résumé :

Un package complet est autonome permettant la génération de pdf

Une utilisation intuitive et libre

Un grand nombre de classes objets prêtes à l'emploi

Une prise en main simple grâce à un grand nombre d’exemples simples