

CMFReportTool

Version 0.3.0

by Ulrich Eck

net-labs Systemhaus GmbH, Germany

1. Introduction.

This product provides the capability of converting Content to PDF files via the click of a button. Secondly, it will allow you to create dynamic content in an XML type document (such as this) for PDF output.

Similar to HTML-Skins one can use Page Markup Language (PML) to describe Dynamic Content as PDF-Documents using the advanced layout-capabilities of reportlab's Platypus module. The PML was inspired by Markup Language used pythonpoint, but it provides all features of [ReportLabs](#) Platypus module.

CMFReportTool is an extension to Zope/CMF and Plone. It's a rich and flexible tool that provides a means of printing PDF documents. It can either print XML formatted (PML) PageTemplates, or CMF Content (e.g. Documents, News Items, etc.) as a PDF document. Since it incorporates PageTemplates as its *Documents* as well as its *Templates*, TAL is available to provide dynamic content.

2. Installing CMFReportTool

A. Dependencies:

- Reportlab PDF-Toolkit current version <http://www.reportlab.org/>
- Zope 2.5.1 or later <http://www.zope.org>
- CMF-1.3 (optionally Plone) <http://cmf.zope.org/>
- Python Image Library (PIL) <http://www.pythonware.com/products/pil/>
- PyXML Library <http://sourceforge.net/projects/pyxml/>

B. Installation:

To install CMFReportTool, uncompress the CMFReportTool product into your zope/Products directory (e.g. linux):

```
ln -s /path/to/installation /path/to/zope/Products
```

If QuickInstaller hasn't been installed, use the Zope Management Interface (ZMI) and navigate to the root of your CMFSite installation:

1. Add an external method to the root of the CMF Site.
2. Use the following configuration values for the External Method:
 - id:** install_reports
 - title:** Install ReportTool optional
 - module name:** CMFReportTool.Install
 - function name:** install
3. Go to the management screen for the newly added external method and click the 'Try it' tab.

The installed function will execute and give information about the steps it took to register and install the CMFReportTool into the CMF Site instance. Once it has completed, a CMF Tool ('portal_report') will be installed in CMF Root, and a 'zpt_reporttool' skin will be installed in the 'portal_skins' folder.

This Software has been tested with Windows 2K, Linux, and Python-2.2

3. Getting Started

The 'portal_report' Tool converts 'PML' Documents, HTML Content and StructuredText-Documents to PDF. This Tool utilizes two Python Scripts to render the content as PDF's (asPDF and asPML). The "rendering process " uses path traversal to designate the PDF-Templatename and the PDF-Documentname.

Example of a traversal path:

```
http:// <path-to-document >/asPDF/templatename/documentname/optionalPDFfilename.pdf
```

PML PageTemplate to PDF: Create a PageTemplate in the *portal_skins/custom* folder with the **Id** of 'default', and add or upload the contents of the *default_template.txt* file located in the 'examples' folder. Create an Image in the *portal_skins/custom* folder with the **Id** of 'net-labs.gif' and upload the *net-labs.gif* file into this object. Next, create a PageTemplate in the Root directory of the CMF site (for this example only) with the **Id** of 'document'. Upload or paste the contents of *testdocument_pdf.txt* into this object (Examples are located at *Products/CMFReportTool/doc/examples/*).

PML - > PDF

```
http:// <path-to-document >/asPDF/default/document
```

CMF Document as Structured Text to PDF: Create a CMF Document with and **Id** of 'structured_ex' and add or upload a Structured Text document such as *<Zope Site >/Products/PageTemplates/help/tal.stx* file. It doesn't matter...Any of the *.stx files located in Zope will work just fine.

STX - > PDF

```
http:// <path-to-document >/asPML/default/structured_ex
```

External Resources (Images) can be located in either Zope or the local file system. A filename-Attribute knows two url-schemes: *file:/ <local-file-system-path-to-image >* and *zodb: <zope-path-to-image-object >*.

Customized Templates: 'PML' formatted ZPT's should be located in the *portal_skins/custom* folder. They are designed to define the layout, style, fonts and formats available to the *Documents*.

Documents: There are two types of Documents.

PML Formatted Documents: These are ZPT's formatted with the PML content. These items are rendered using the PML-to-PDF path traversal method described above.

CMF/Plone Content: These are Zope CMF Content items. For example, Documents and News Items containing HTML or Structured Text. These items are rendered using the following path traversal:

```
http:// <path-to-document >/asPDF/pdftemplate_default/document_pdf
```

An even easier method to render these documents as PDF is the action icon mentioned below.

Adding PDF ActionIcon: Plone 2.x has a *portal_actionicons* tool which makes it easy to add a "document_action" icon to the portal. You simply do the following:

1. Add to 'portal_action':

Name: Skin document as PDF

ID: pdf

Action: string:\${object_url}/asPDF/pdftemplate_default/document_pdf

Condition: python:test(hasattr(portal.portal_properties.site_properties, 'allow_pdf') and portal.portal_properties.site_properties.allow_pdf, 1, 0)

Permission: View

Category: document_actions

2. Add to 'portal_actionicon':

Category: plone

Action ID: pdf

Action Title: PDF

Priority: 0

Icon URL Expression: pdf_icon.gif

3. Add property to 'portal_properties/site_properties':

Name: allow_pdf

Type: boolean

Value: 1

4. The Page Markup Language (PML)

A. Templates:

Templates define static elements, stylesheets as well as Frames. It works like Quark-XPress or FrameMaker. Predefined and linked Frames are filled with Flowables from the Document. Think of these as a lot like slots in PageTemplates, but content is concated and nicely distributed across the frames. Templates are Zope PageTemplate (ZPT) objects containing PML data, and are located in the 'portal_skins/custom' folder.

<template > The **root element** of the Template XML format. The 'template' contains any number of <pagetemplate > elements and an optional <stylesheet > element.

Attribute	Description	Default
filename	Name of default file name	
pagesize	A0-A6, B0-B6, LETTER, LEGAL, ELEVENSEVENTEEN	A4
landscape		0
showboundary	If set draw a box around the frame boundaries	0
leftmargin		10
rightmargin		10
topmargin		10
bottommargin		10
allowsplitting	If set flowables (eg, paragraphs) may be split across frames or pages	1

PageTemplate

<pagetemplate > The page formatting element. This element must not be empty. A *Template* may contain one or more *pagetemplates*. Names assigned to the templates are used for referencing, therefore, no two templates should have the same *id*. For example, one template might be used as the title page (e.g. id="FirstPage "), and one or more for the remaining sections of the document.

The order of *pagetemplates* may be controlled with the *nextid* attribute. In other words, if an action is taken within a Document to end the page, the application will switch to the next *pagetemplate* defined in the *nextid*. This attribute is not required.

Attribute *startframe* is optional. However, if more than one frame elements are included in the *pagetemplate* it would be necessary to define which frame will be used first. The parent element is the <template >.

The 'pagetemplate' contains one or more <frame > element, and an optional <static > element.

Attribute	Description	Default
id	Name of the 'pagetemplate'.(Required)	
nextid	The 'id' name of the next <pagetemplate>.	None
startframe	The 'id' name of the <frame>.	None

Frame

<frame > Frames are containers or writing space used by the Document to populate with content. Just as the name implies, it frames the content on the page. The parent element is the `<pagetemplate >`.

Each frame is required to have a unique *id* within a `pagetemplate`. The same name may be used within a different `pagetemplate`. If more than one frame is available within a `pagetemplate` it will be necessary to define the *nextid* attribute in order to control the flow of frames.

Positioning the frame is controlled by the *x* and *y* attributes. These define the lower-left corner of the frame. The width and height of the frame are required attributes of the frame.

A two column *pagetemplate* for an 'A4' *pagesize* would look like the following:

```
<pagetemplate id="DemoPage" startframe="content">
  <static>

    <infostring align="left" x="10cm" y="28cm" size="10"
      font="Helvetica"
      color="(0.4,0.4,0.4)">Page %(page)s</infostring>
    <line x1="1cm" x2="20cm" y1="27.9cm" y2="27.9cm"
      stroke="(0.4,0.4,0.4)"
      linewidth="1"/>

  </static>
  <frame id="content"
    nextid="right"
    x="2cm"
    y="1.5cm"
    width="8cm"
    height="25.5cm"
    leftpadding="0.1cm"
    rightpadding="0.1cm"
    toppadding="0.5cm"
    bottompadding="0.5cm"
    showBoundary="0"/>
  <frame id="right"
    nextid="content"
    x="11cm"
    y="1.5cm"
    width="8cm"
    height="25.5cm"
    leftpadding="0.1cm"
    rightpadding="0.1cm"
    toppadding="0.5cm"
    bottompadding="0.5cm"
    showBoundary="0"/>
</pagetemplate>
```

The attributes and default values are listed below:

Attribute	Description	Default
id	Name of the 'frame'.(Required)	
nextid	The 'id' name of the next <frame>.	
x	The 'x' position of lower-left corner.(Required)	
y	The 'y' position of lower-left corner.(Required)	
width	Width of frame.(Required)	
height	Height of frame.(Required)	
leftpadding	Left padding within the frame.	10
rightpadding	Right padding within the frame.	10
toppadding	Top padding within the frame.	10
bottompadding	Bottom padding within the frame.	10
showboundary	Draw a box around the frame boundaries.	0

Static

<static > Static is a CMFReportTool element used to anchor the area of a 'pagetemplate' that contains graphic elements such as <line >, <rectangle >, <roundrect >, <polygon >, <string >, <infostring >, <customshape >, <fixedimage > or <ellipse >. This element has no attributes. The parent element is the <pagetemplate >.

<line > The parent element is the <static >.

Attribute	Description	Default
x1	The 'x' starting point.	
y1	The 'y' starting point.	
x2	The 'x' ending point.	
y2	The 'y' ending point.	
stroke	Color of line.	(0,0,0)
width	Width of line.	

<rectangle >

Attribute	Description	Default
x	The 'x' location of the lower-left corner.	
y	The 'y' location of the lower-left corner.	
width	Width of rectangle.	
height	Height of rectangle.	
fill	Fill color.	
stroke	Color of border.	(0,0,0)
linewidth	Width of border.	

<roundrect >

Attribute	Description	Default
x	The 'x' location of the lower-left corner.	
y	The 'y' location of the lower-left corner.	
width	Width of rectangle.	
height	Height of rectangle.	
radius	Radius of corners.	
fill	Fill color.	
stroke	Color of border.	(0,0,0)
linewidth	Width of border.	

<ellipse >

Attribute	Description	Default
x1		
y1		
x2		
y2		
fill	Fill color.	
stroke	Color of border.	(0,0,0)
linewidth	Width of border.	

<polygon >

Attribute	Description	Default
points		
linewidth		
fill	Fill color.	
stroke	Color of border.	(0,0,0)

<string >

Attribute	Description	Default
x		0
y		0
color		(0,0,0)
font		Times-Roman
size		12
align		left

<infostring >

Attribute	Description	Default
x		0
y		0
color		(0,0,0)
font		Times-Roman
size		12
align		left

<customshape >

Attribute	Description	Default
path		
module		
class		
initargs		

<fixedimage >

Attribute	Description	Default
x	The 'x' location of the lower-left corner.	
y	The 'y' location of the lower-left corner.	
width	Width of image.	
height	Height of image.	

StyleSheet

<stylesheet > This element contains an optional number of **<paragraphstyle >** and **<tablestyle >** elements defining the style of paragraphs and/or tables. The parent element is the **<template >**.

ParagraphStyle

<paragraphstyle > Defines the style of the paragraph flowables. Several examples of paragraphstyles can be seen in the **Paragraph Examples** section of this document. The parent element is the **<stylesheet >**.

Each paragraphstyle requires a *name*, *fontname* and *fontsize*. The *name* attribute is referenced by the **<para >**, **<prefmt >** and **<xprefmt >** elements of the Document.

Optionally, *paragraphstyles* may inherit attributes from a *parent paragraphstyle* item. This can save a lot of extra typing when designing a *paragraphstyle*.

Top and bottom padding is controlled with the *spacebefore* and *spaceafter* attributes. The left padding is controlled with the *leftindent* attribute.

Attribute	Description	Default
name	Name of the 'paragraphstyle'.(Required)	
fontname	Name of the font name. (Required)	Times-Roman
fontsize	Size of the font. (Required)	10
alignment	left right center justify	left
firstlineindent	Spaces to indent first line.	0
bulletfontname	Bullet font name.	Times-Roman
bulletfontsize	Bullet font size.	10
leftindent	Left padding.	0
leading	Spacing between adjacent lines of text.	12
parent	Parent paragraphstyle item.	None
spacebefore	Space before the paragraph.	0
spaceafter	Space after the paragraph.	0
textcolor	Color of text.	black

Fonts available for use in the paragraphstyle:

Times-Roman|Times-Bold|Times-Italic|Times-BoldItalic

Courier|Courier-Bold|Courier-Oblique|Courier-BoldOblique

Helvetica|Helvetica-Bold|Helvetica-Oblique|Helvetica-BoldOblique

Symbol|ZapfDingbats

TableStyle

<tablestyle > Defines the 'table' style. Contained in the <stylesheet > element. The only (Required) argument in the 'tablestyle' tag is the 'name' attribute. This is the 'style' attribute associated with the <table > tag. The parent element is the <stylesheet >.

Example of a 'tablestyle':

```
<tablestyle name="StandardTable">
  <stylecmd expr="( 'LEFTPADDING', (0,0), (-1,-1), 5)"/>
  <stylecmd expr="( 'RIGHTPADDING', (0,0), (-1,-1), 5)"/>
  <stylecmd expr="( 'TOPPADDING', (0,0), (-1,-1), 3)"/>
  <stylecmd expr="( 'BOTTOMPADDING', (0,0), (-1,-1), 7)"/>
  <stylecmd expr="( 'ALIGN', (0,0), (-1,-1), 'LEFT')"/>
  <stylecmd expr="( 'GRID', (0,0), (-1,-1), 0.5, colors.black)"/>
  <stylecmd expr="( 'BOX', (0,0), (-1,-1), 1, colors.black)"/>
  <stylecmd expr="( 'FONT', (0,0), (-1,-1), 'Helvetica', 10)"/>
</tablestyle>
```

<stylecmd >The commands passed to argument 'expr' of the 'stylecmd' come in three main groups which affect the table background, draw lines, or set cell styles.

The first element of each command is its identifier, the second and third arguments determine the cell coordinates of the box of cells which are affected with negative coordinates counting backwards from the limit values as in **Python** indexing. The coordinates are given as (column, row) which follows the spreadsheet 'A1' model, but not the more natural (for mathematicians) 'RC' ordering. The top left cell is (0,0) the bottom right is (-1,-1).

Example of cell coordinates:

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)

Another example using a mix of negative and positive coordinates:

(0,0)	(1,0)	(2,0)	(3,0)	(-1,0)
(0,1)	(1,1)	(-3,-3)	(-2,1)	(-1,1)
(0,2)	(1,2)	(2,2)	(-2,-2)	(4,2)
(0,-1)	(1,3)	(2,3)	(3,3)	(-1,-1)

The cell formatting commands all begin with an identifier, followed by the start and stop cell definitions and the perhaps other arguments. the cell formatting commands are:

Cell Formatting Commands

Attribute	Description	Default
ALIGNMENT (or ALIGN)	LEFT RIGHT CENTRE (or CENTER)	

Attribute	Description	Default
BACKGROUND	takes a color name or (R,G,B) tuple.	
BOTTOMPADDING	takes an integer.	3
FONT	takes fontname, optional fontsize and optional leading.	
FONTNAME (or FACE)	takes fontname	
FONTSIZE (or SIZE)	takes fontsize in points; leading may get out of sync.	
LEADING	takes leading in points.	
LEFTPADDING	takes an integer.	6
RIGHTPADDING	takes an integer.	6
TEXTCOLOR	takes a color name or (R,G,B) tuple.	
TOPPADDING	takes an integer.	3
VALIGN	TOP MIDDLE BOTTOM	BOTTOM

Line Formating Commands: The line command names are: GRID, BOX, OUTLINE, INNERGRID, LINEBELOW, LINEABOVE, LINEBEFORE and LINEAFTER. BOX and OUTLINE are equivalent, and GRID is the equivalent of applying both BOX and INNERGRID.

Here's an example using the Line Formating Commands:

```
<tablestyle name="LineCommandTable">
  <stylecmd expr="( 'GRID' ,(0,0) ,(-1,-1) ,0.5 ,colors.grey)"/>
  <stylecmd expr="( 'GRID' ,(1,1) ,(-2,-2) ,1 ,colors.green)"/>
  <stylecmd expr="( 'BOX' ,(0,0) ,(1,-1) ,2 ,colors.red)"/>
  <stylecmd expr="( 'BOX' ,(0,0) ,(-1,-1) ,2 ,colors.black)"/>
  <stylecmd expr="( 'LINEABOVE' ,(1,2) ,(-2,2) ,1 ,colors.blue)"/>
  <stylecmd expr="( 'LINEBEFORE' ,(2,1) ,(2,-2) ,1 ,colors.pink)"/>
  <stylecmd expr="( 'BACKGROUND' ,(0,0) ,(0,1) ,colors.pink)"/>
  <stylecmd expr="( 'BACKGROUND' ,(1,1) ,(1,2) ,colors.lavender)"/>
  <stylecmd expr="( 'BACKGROUND' ,(2,2) ,(2,3) ,colors.orange)"/>
</tablestyle>
```

produces:

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
(0,2)	(1,2)	(2,-2)	(-2,-2)	(4,2)
(0,3)	(1,-1)	(2,3)	(3,3)	(-1,-1)

B. Documents:

Documents define dynamic Content, Images, Vectorgraphics as well as Metadata. This part is usually generated when the client requests a view as PDF and it is filled in (using PageTemplates) with the objects/lists values and rendered. Documents may be stored anywhere within the CMF site.

<document > The **root element** of the Document XML format. The optional attribute 'filename' defines the name of the displayed PDF file. Contains <title > and <content > elements.

```
<document filename=[None] >Document Elements </document >
```

<title > Document title. The parent element is the <document >.

```
<title >Title </title >
```

<author > Document author. The parent element is the <document >.

```
<author >Author </author >
```

<subject > Document subject. The parent element is the <document >.

```
<subject >Subject </subject >
```

<content > Contains any number of <para >, <prefmt >, <image >, <table >, <spacer > and <action > elements. The parent element is the <document >.

A document may have any number of *content* elements.

```
<content >Content Elements <content >
```

<para > The paragraph XML container. The attribute 'style' defines the 'paragraphstyle' to use.

Paragraphstyles are defined in the 'stylesheet' section of the Template. The parent element is the <content >.

```
<para style=[Normal] [bullettext={Font Character}] >Text </para >
```

<prefmt > Creates a preformatted paragraph which does no wrapping, line splitting or other manipulations. No XML style tags are taken account of in the text. If dedent is non zero dedent common leading spaces will be removed from the front of each line. The parent element is the <content >.

```
<prefmt style=[Normal] dedent=[inch|cm] >Text </prefmt >
```

<xprefmt > This is a cross between <para > and <prefmt >. If dedent is non zero dedent common leading spaces will be removed from the front of each line. The parent element is the <content >.

```
<xprefmt style=[Normal] dedent=[inch|cm] >Text </xprefmt >
```

<image > An image file located in the ZODB or on the host file system. The attributes 'width' and 'height' define the displayed size of the image. Units may be expressed in 'inches' (inch) or 'centimeters' (cm). The default unit is inches. The parent element is the <content >.

```
<image filename=[file:|zodb:] width=[inch|cm] height=[inch|cm] / >
```

<shape > Much the same as the shapes defined in the Template section. However, it will require the <parameter > element to define the characteristics of the shape. The parent element is the <content >.

```
<shape factory=? > <parameter >Parameter Info </parameter > </shape >
```

<spacer > An element used to provide vertical spacing between other elements. The height units may be expressed in 'inches' (inch) or 'centimeters' (cm). The default unit is inches. The parent element is the <content >.

```
<spacer height=[inch|cm] / >
```

<parameter > This element is used to pass parameters to the *action* element. The parent element is the *<action >*.

```
<parameter >... </parameter >
```

<action > Provides control of the 'template' within the 'content' section of the *<document >*. This element may also contain a *<parameter >* element which is used to pass parameters to Platypus (see example below). The following example will change the pagetemplate and end the page (pagebreak). The parent element is the *<content >*.

```
<action name="nextPageTemplate">
<parameter>PageTempateID</parameter>
</action>

<action name="pageEnd"/>
```

Other action types:

Attribute	Description	Parameter
documentBegin	Implement actions at beginning of document	
pageBegin	Perform actions required at beginning of page. Shouldn't normally be called directly	
pageEnd	Show the current page check the next page template hang a page begin	
frameBegin	What to do at the beginning of a frame	
frameEnd	Handles the semantics of the end of a frame. This includes the selection of the next frame or if this is the last frame then invoke pageEnd.	
flowable	Try to handle one flowable from the front of list [flowables]	flowables
nextPageTemplate	On endPage change to the page template with name or index [pt]	pagetemplate
currentFrame	Change to the frame with name or index [fx]	frame
nextFrame	On endFrame change to the frame with name or index [fx]	frame

<table > This element is almost identical to its HTML counterpart. Contains the row *<tr >* and data cells *<td >*. The parent element is the *<content >*.

Attribute	Description	Default
rowheight	The 'rowheight' argument specifies the height of all rows in the Table. If this attribute is omitted the height will be calculated automatically. This argument may be overridden by specifying the 'rowheight' in the row <i><tr ></i> tag.	Optional
colwidth	The 'colwidths' argument specifies the default width of all columns. Omitting this attribute will result in the automatic calculation of column width. This argument may be overridden by specifying the 'colwidth' in the data cell <i><td ></i> tag.	Optional

Attribute	Description	Default
slitbyrow	The 'splitbyrow' argument is only needed for tables both too tall and too wide to fit in the current context. In this case you must decide whether to 'tile' down and across, or across and then down. This parameter is a Boolean indicating that the Table should split itself by row before attempting to split itself by column	1
repeatrows	The 'repeatrows' and 'repeatcols' arguments specify the number of leading rows and columns that should be repeated when the Table is asked to split itself.	1
repeatcols	See 'repeatrows'.	0
style	Specifies the 'tablestyle' to use in the <stylesheet> section of the <template>.	None

<tr > This is similar to the HTML tag <tr >. Acceptable attributes are 'rowheight' and 'stylecmd'. If the 'rowheight' is omitted from the <table > tag, the 'rowheight' will be calculated automatically. The parent element is the <table >.

Attribute	Description	Default
rowheight	The 'rowheight' argument specifies the height of all rows in the Table. If this attribute is omitted the height will be calculated automatically. This argument may be overridden by specifying the 'rowheight' in the row <tr> tag.	Optional
stylecmd	See the 'tablestyle' section.	Optional

<td > This is similar to the HTML <td > data cell tag. Acceptable attributes are 'colwidth' and 'stylecmd'. If the 'colwidth' is omitted from the <table > tag, the 'colwidth' will be calculated automatically. The parent element is the <tr >.

Attribute	Description	Default
colwidth	The 'colwidths' argument specifies the default width of all columns. Omitting this attribute will result in the automatic calculation of column width. This argument may be overridden by specifying the 'colwidth' in the data cell <td> tag.	Optional
stylecmd	See the 'tablestyle' section.	Optional

5. Paragraph Examples

The `<para >` and `<prefmt >` flowables can format a block of text into a paragraph with a given style.

The paragraph Text can contain HTML-like markup including the following tags:

Tag	Description
<code><a></code>	anchor or link
<code><link></code>	link
<code><bullet></code>	bullet
<code></code>	bold
<code><i></code>	italics
<code><u></code>	underline
<code><dl></code>	data list
<code><dt></code>	data title
<code><dd></code>	data
<code><super></code>	superscript
<code><sub></code>	subscript
<code></code>	unordered list
<code></code>	ordered list
<code></code>	list item
<code></code>	font
<code><onDraw name=callable label="a label"></code>	onDraw event

Note: With the exception of `` and `<onDraw >` These HTML like tags cannot contain any attributes. In other words, you may not use a *style* attribute with these tags. An error may occur if you attempt to nest these tags.

A. Header

Header Example

template

```
<paragraphstyle name="Heading4" parent="Normal"
    fontname="Helvetica-Bold" fontsize="11"
    spaceafter="10" spacebefore="5"/>
```

document

```
<para style= "Heading4 " >Header Example </para >
```

B. Normal:

This is an example of a **'Normal'** paragraphstyle in CMFReportTool.

template

```
<paragraphstyle name="Normal"
    fontname="Helvetica" fontsize="11"
    alignment="left" firstlineindent="0"
    leftindent="0" spaceafter="5"/>
```

document

```
<para style= "Normal " >This is an example of a <B >'Normal' </B > paragraphstyle
in CMFReportTool. </para >
```

C. Bullet:

- This is an example of a *'Bullet'* paragraphstyle in CMFReportTool.

template

```
<paragraphstyle name="Bullet" parent="Normal"
    fontname="Helvetica" fontsize="11"
    bulletfontname="Symbol" bulletfontsize="10"
    bulletindent="15" leftindent="20"
    spaceafter="5"/>
```

document

```
<para style= "Bullet " >This is an example of a <I >'Bullet' </I > paragraphstyle
in CMFReportTool. </para >
```

D. Code:

This is an example of a *'Code'* paragraphstyle in CMFReportTool.

template

```
<paragraphstyle name="Code" parent="Normal"
    fontname="Courier" fontsize="11"
    align="None" textcolor="(0,0,0.6)"
    spaceafter="5"/>
```

document

```
<para style= "Code " >This is an example of a <I >'Code' </I > paragraphstyle in
CMFReportTool. </para >
```

E. Preformatted:

This is an example
of a *'Preformatted'* paragraphstyle
in CMFReportTool.

template

```
<paragraphstyle name="Normal"
    fontname="Helvetica" fontsize="11"
    alignment="left" firstlineindent="0"
    leftindent="0" spaceafter="5"/>
```

document

```
This is an example
of a 'Preformatted' paragraphstyle
in CMFReportTool.</prefmt>
```

F. Sequencer:

Example #1 of a *'Sequencer'*.

Example #2 of a 'Sequencer'.

Example #3 of a 'Sequencer'.

template

Not applicable

document

```
<para style="Normal"><seqreset id="spam"/>Example #<seq id="spam"/> of a <I>'Sequencer'
```

```
<para style="Normal">Example #<seq id="spam"/> of a <I>'Sequencer'</I>.</para>
```

```
<para style="Normal">Example #<seq id="spam"/> of a <I>'Sequencer'</I>.</para>
```

6. Dynamic XML with Page Templates

Using Zope Page Templates (ZPT) provides the ability to generate dynamic XML. This is accomplished with the Tag Attribute Language (TAL), TAL Expression Syntax (TALES), and Macro Expansion TAL (METAL). Refer to ZPT documentation for additional information on this subject. This functionality is available in the PML-Templates as well as the PML-Documents.

Below is an example of a dynamically generated <table > using TAL:

Amount	Description	Price
0	Some Description	65.0,- EUR
1	Some Description	65.0,- EUR
2	Some Description	65.0,- EUR
3	Some Description	65.0,- EUR
4	Some Description	65.0,- EUR
5	Some Description	65.0,- EUR
6	Some Description	65.0,- EUR
7	Some Description	65.0,- EUR
8	Some Description	65.0,- EUR
9	Some Description	65.0,- EUR
	Total	650.0,- EUR

Source code for above example:

```
<content tal:define="price python:65.0; count python:10">
  <para style="Heading3">Dynamic Table Example</para>
  <spacer height="15" />
  <table rowheight="0.7cm"
    colwidth="2cm"
    splitbyrow="1"
    repeatrows="1"
    repeatcols="0"
    style="StandardTable">
    <tr>
      <td colwidth="2cm">Amount</td>
      <td colwidth="11cm">Description</td>
      <td colwidth="3cm">Price</td>
    </tr>
    <tal:block tal:repeat="item python:range(count)">
      <tr>
        <td tal:content="item"/>
        <td>Some Description</td>
        <td tal:content="string:${price},- EUR"/>
      </tr>
    </tal:block>
```

```
<tr tal:define="sum python:count*price">
  <td> </td>
  <td>Total</td>
  <td tal:content="string:${sum},- EUR"/>
</tr>
</table>
</content>
```

7. Definitions

Document: PML formatted Zope Page Template (ZPT) containing the content of the PDF document.

Styles: Paragraph- and Table-Styles can be preconfigured within the Template-Header.

Static-Elements: FixedImage,InfoString,Line,Rectangle,..all these Static-Element-Types are usable within the Templates Pagedefinition to setup static content.

Template: PML formatted Zope Page Template (ZPT) containing the output template of the PDF document. Includes the <stylesheet >, <pagetemplate > and <frame > elements.

Frames: Frames hold Flowables (Dynamic-Elements). They are linked via Names to enable things like: different Startpage, left/right-page of books, etc.

Flowables: Dynamic-Elements are called Flowables. They are aranged within the predefined order in Frames. Available Types are: Paragraph, Preformatted, Table, Image, Line, Rectangle, ...

Zope Page Template (ZPT): Zope Page Templates are an HTML/XML generation tool. These ZPT's include Tag Attribute Language (TAL), TAL Expression Syntax (TALES), and Macro Expansion TAL (METAL).

8. References

ReportLab User Guide: <http://www.reportlab.org>

Zope Page Templates Reference: <http://zope.org/Documentation/Books/ZopeBook/>

9. Credits

Thanks to the following people who contributed to CMFReportTool:

Stephen B. Kirby for contributing most of the UserGuide.