Zpdf

1.1 Introduction

Native python library for generating Portable Document Format (PDF) documents. Provided mainly in support of StruturedText (see the documentation string of StructuredText.py) and is therefore limited to text. However it provides a nice class interface to the library.

The main structure in a PDF document is a Document which contains a number of elements defined in the elements.py module. The pages are collected in a hierarchical manner under a root object and can share common resources through shared dictionaries. The PDF language is, like postscript a general page description language, and provides no tools for higher level document formatting. The object model of this module was designed for simple applications and facilitates easy document formatting. Document formating is a complex and difficult subject and thus users who demand a complete system should probably look at Lout or TeX.

1.2 PDF Objects

All PDF objects inherit from the PDFObject base class. It provides all objects with a __getitem__, __setitem__ special method and thus instances act like dictionaries with a limited functionality. The PDFObject base class in conjunction with the AcquisitionDict class provide for a simple acquisition mechanism where element properties that do not exist in the instance's local properties dictionary are retrieved from parent objects. This allows global changes to document properties like font_family, font_size etc.

All elements (with the exception of the basic phrase unit, and UL, OL) conform to the following interface:

- format: formats elements based on the document style.
- more: This method should return two objects, the first is the part of the element that fits in the current page and a second containing any leftovers.
- to_PDF: returns a PDF representation of the element.
- get_bbox: returns the bounding box of the object.

1.2.1 Text Elements

The major text element is a Paragraph. It accepts blocks of text and returns PDF formatted output depending on the Document style. Classes in the elements module correspond loosely to the HTML elements:

- P: Paragraph element
- H[1-6]: Headings
- HR: Horizontal Line

- PRE: Preformatted text
- EM: Emphasized text (italics)
- STRONG: Bold text
- CODE: Fixed width text
- UL: Bulleted list
- OL: Ordered list
- Header: Header info common to all pages

Style information can be provided to individual elements but the general approach is to provide a style dictionary to the root document.

[More documentation should go here ...]

1.2.2 Examples

Lets jump to some examples.

Converting this document to HTML and then PDF:

```
python StructuredText.py doc/Zpdf.stx > doc/Zpdf.html

python html2pdf doc/Zpdf.html > doc/Zpdf.pdf
```

You can change the style by adding a new style section in the styles.py module (or in your own styles module). The effect of the different style parameters should be documented but I hope their purpose is obvious.

Here is a more complete listing of the previous example:

```
import styles,elements,pdflib,html2pdf

text=open('doc/Zpdf.html').read()

out=[]
p = html2pdf.HTMLParser(out)
p.feed(data)
p.close()

doc=pdflib.Document()
doc.initialize()
doc.set_style(styles.zpdf_doc)

lt=elements.HeaderLine()
lt['position']='T'

lb=elements.HeaderLine()
```

To create a simple paragraph of text:

```
import styles,elements,pdflib

doc=pdflib.Document()
doc.initialize()
doc.set_style(styles.zpdf_doc)

par=elements.P()
par.add_text('Zpdf v0.1')

doc.add_elements([par])
doc.format()
print doc.to_PDF()
```

Try it on the many structured-text documents available in the Zope source distribution!