pyxml2pdf Documentation

Björn Ludwig

Feb 09, 2021

Contents:

Convert XML input to PDF table. Since we forked the upstream this project has generalized quite a bit on the generation of a multipage PDF file containing a table with subtables each containing a subset of the xml tags based on the texts of some of their children tags.

For the *pyxml2pdf* homepage go to GitHub.

pyxml2pdf is written in Python 3 and currently requires Python 3.6 or later.

CHAPTER 1

pyxml2pdf

1.1 core

1.1.1 downloader

class pyxml2pdf.core.downloader.**Downloader**(*url*, *output_filename=None*)

Download a file and store the result

If no *output_filename* is specified, we extract the filename of the downloaded file and store the result in the *input* subfolder of the root directory.

Parameters

- **url** (*str*) – the full download link
- **output_filename** (*str*) – the local path where and under what name to store the downloaded file

_extract_filename()

Extract from a url the element after the last slash, i.e. the filename

Returns the filename in the url

Return type str

1.1.2 events

A wrapper *pyxml2pdf.core.events.Event* for xml extracted data

class pyxml2pdf.core.events.**Event**(*element*)

A wrapper class for xml.etree.ElementTree.Element

xml.etree.ElementTree.Element is augmented with the table row representation and the attributes and methods to manipulate everything according to the final tables needs. A *pyxml2pdf.core.events.Event* can only be initialized with an object of type xml.etree.ElementTree.Element.

Parameters **element** (*xml.etree.ElementTree.Element*) – the element to build the instance from

class EventParagraph(*text: str*)
A wrapper class for `reportlab.platypus.Paragraph`

`reportlab.platypus.Paragraph` is solely used with one certain style, which is handed over in the constructor.

Parameters **text** (*str*) – the text to write into row

_build_description(*link: str = ''*) → str
Build the description for the event

This covers all cases with empty texts in some of the according children tags and the full as well as the reduced version with just the reference to the subtable where the full version can be found. Since the title of the event is mandatory, and the beginning of the description is always filled by the same tags' texts those are not received as parameter but directly retrieved from the xml data.

Parameters **link** (*str*) – a link to more details like the trainer url or the subtable

Returns the full description including url if provided

Return type str

_build_type() → str
Build the type for the event

This assembles the type of the event from the different kinds.

Returns the entry in the type column of the event

Return type str

_concatenate_tags_content(*event_subelements: List[str], separator: str = ' - '*) → str
Form one string from the texts of a subset of an event's children tags

Form a string of the content for all desired event children tags by concatenating them together with a separator. This is especially necessary, since `reportlab.platypus.Paragraph` cannot handle *None's as texts but handles as well the concatenation of XML tags' content, if 'event_subelements* has more than one element. So we ensure the result to be at least an empty string.

Parameters

- **event_subelements** – list of all tags for which the descriptive texts is wanted, even if it is just one
- **separator** – the separator in between the concatenated texts

Returns concatenated, separated texts of all tags for the current event

_init_categories()
Initialize the list of categories from the according xml tag's content

_init_date()
Create a properly formatted string containing the date of the event

_init_full_row() → List[pyxml2pdf.core.events.Event.EventParagraph]
Initialize the single table row containing all information of the event

Extract interesting information from events children tags and connect them into a nicely formatted row of a table.

Returns the common starting columns of any table representation

Return type List[*EventParagraph*]

_init_reduced_row(*subtable_title*)
Initializes the reduced version of the event

Create a table row in proper format but just containing a brief description of the event and a reference to the fully described event at another place, namely the subtable with the given title.

Parameters **subtable_title** (*str*) – title of the subtable which contains the full event

> **Warning:** Do not call this function directly since it is automatically called right after *get_full_row()* is invoked.

static _parse_prerequisites(*personal: str*, *material: str*, *financial: str*, *offers: str*) → str
Determine all prerequisites and assemble a string accordingly.

Parameters

- **material** (*str*) – material prerequisite xml text
- **personal** (*str*) – personal prerequisite xml text
- **financial** (*str*) – financial prerequisite xml text
- **offers** (*str*) – xml text of what is included in the price

Returns the text to insert in prerequisite column the current event

Return type str

static _remove_century(*four_digit_year*)
Remove the first two digits of the string representing the year

Parameters **four_digit_year** (*typing.Match*) – the result of `re.sub()`

Returns the last two digits of the string representing the year

Return type str

_table_builder = <pyxml2pdf.tables.builder.TableBuilder object>

_table_style = <pyxml2pdf.styles.table_styles.TableStyle object>

categories
Return the event's categories

Returns a list of the event's categories

Return type List[str]

create_reduced_after_full()
Decorator to execute *_init_reduced_row()* with *get_full_row()*

Returns the return value of *get_full_row()*

Return type Table

date
Return the date of the event

Returns date

Return type str

get_full_row(**args*, ***kwargs*)
Exchange a table row with all the event's information against a subtable's title

This ensures, that after handing over the full information, the reduced version with a reference to the subtable containing the full version is created.

Note: This is ensured by a decorator, which is why the function signature on ReadTheDocs.org is displayed incorrectly. The parameter and return value are as follows...

Parameters **subtable_title** (*str*) – the title of the subtable in which the row will be integrated

Returns a table row with all the event's information

Return type Table

get_table_row(*subtable_title*)

Return the table row representation of the event

This is the API of *pyxml2pdf.core.events.Event* for getting the table row representation of the event. It makes sure, that on the first call *get_full_row()* is invoked and otherwise `pyxml2pdf.core.events.Event._reduced_row` is returned.

Parameters **subtable_title** (*str*) – the title of the subtable in which the row will be integrated

Returns a table row representation of the event

Return type Table

responsible

Return the name of the person being responsible for the event

Returns first and last name

Return type str

1.1.3 initializer

class pyxml2pdf.core.initializer.**Initializer**(*input_path*, *output_path*)

Coordinate the construction of the pdf result

Parameters

- **input_path** (*str*) – path to input xml-file
- **output_path** (*str*) – path to pdf file containing result

1.1.4 parser

pyxml2pdf.core.parser is the interface between xml input and table

class pyxml2pdf.core.parser.**Parser**(*elements: List[reportlab.platypus.flowables.KeepTogether]*)

XML parser to extract all interesting information from xml input

Parameters **elements** – cells to populate the Parser

collect_xml_data(*events*)

Traverse the parsed xml data and gather collected event data

The collected xml data then is passed to the table_manager and all arranged data is return.

Parameters **events** (*List[Event]*) – a list of the items from which the texts shall be extracted into a nicely formatted table

Returns list of all table rows containing the relevant event data

Return type List[KeepTogether]

1.1.5 post_processor

class pyxml2pdf.core.post_processor.**PostProcessor**(*path*)
Arrange for needed modifications of the result to prepare for printing

This creates an instance of a *pyxml2pdf.core.post_processor* for a multipage PDF file to automate splitting and rotating.

Parameters **path** (*str*) – path to the PDF file which shall be processed

finalize_print_preparation()
Take the resulting multi page PDF and split into rotated single pages

Taken from pythonlibrary.org in combination with johndcook.com

1.1.6 sorter

class pyxml2pdf.core.sorter.**Sorter**(*courses*)
Provides a method to sort from xml extracted data by a tag containing a date

We took effbot.org and adapted the code to our needs of sorting a list of `xml.etree.ElementTree.Element` by the texts of one of their tags containing a string representation of a date.

Parameters **courses** (*List[xml.etree.ElementTree.Element]*) – events that where extracted from an xml source

sort_parsed_xml(*sort_key*)
Sort a list of `xml.etree.ElementTree.Element` by their date

Taken from effbot.org and adapted.

Parameters **sort_key** (*str*) – the xml tag which contains the data

1.2 tables

1.2.1 tables

class pyxml2pdf.tables.tables.**EventTable**(*title*, *locations*, *activities*)
An EventTable contains a subset of the xml inputs

Contains all events which cover the mentioned activities at the mentioned locations. Every event listed covers at least one of the listed activities at one of the listed locations.

Parameters

- **title** (*str*) – name of the table describing where and what is done in the included events
- **locations** (*List[str]*) – list of locations where the activities take place
- **activities** (*List[str]*) – list of activities covered by the listed events

append(*event: reportlab.platypus.tables.Table*)
Append an event to the end of the table

Parameters **event** – a single event that should be appended to the table's list of events

extend(*event_list*)
Append a a list of events to the end of the table

Parameters **event_list** – a list of events that should be appended to the table's list of events

1.2.2 builder

class pyxml2pdf.tables.builder.**TableBuilder**

collect_subtables() → List[reportlab.platypus.tables.Table]
Collect all subtables at once

Returns the subtables

create_fixedwidth_table(*cells: List[List[reportlab.platypus.flowables.Flowable]], widths: Union[float, List[float], None] = None, style: Optional[List[Tuple[Union[str, Tuple[int]]]]] = None*) → reportlab.platypus.tables.Table
Create a table with specified column widths

Create a table from specified cells with fixed column widths and a specific style.

Parameters

- **cells** – cells wrapped by a list representing the columns wrapped by a list representing the lines
- **widths** – Optional column widths. The default results in reasonable settings based on experience.
- **style** – Optional table's style. The default results in reasonable settings based on experience.

Returns A table containing specified cells in fixed width, styled columns.

create_subtables()
Create subtables for all different kinds of events

Return List[EventTable] a list of all subtables

distribute_event(*event*)
Distribute an event to the subtables according to the related categories

Parameters **event** (*Core.events.Event*) – event to distribute

make_header(*title: str*) → List[reportlab.platypus.tables.Table]
Build the first two rows of a subtable

Build the first two rows of a subtable with its title and column headings taken from the properties file.

Parameters **title** (*str*) – the title of the subtable

Returns two line table with title and headings

Return type List[reportlab.platypus.Table]

1.3 styles

1.3.1 cell_formattings

class pyxml2pdf.styles.cell_formattings.**CellFormattingCommands**

This class contains some cell formatting commands explained from page 85 on in the reportlab user guide.

A command is structured as a tuple containing a descriptive string representing the command class, the first and the last cell on which the command should be invoked and the parameter value for the command. For details see the link.

align_center = ('ALIGN', (0, 0), (-1, -1), 'CENTER')

align_left = ('ALIGN', (0, 0), (-1, -1), 'LEFT')

static background(*colors*)

static box(*colors*)

static font_size(*size*)

static inner_grid(*colors*)

leftpadding_reduce = ('LEFTPADDING', (0, 0), (-1, -1), 3)

rightpadding_reduce = ('RIGHTPADDING', (0, 0), (-1, -1), 3)

valign_middle = ('VALIGN', (0, 0), (-1, -1), 'MIDDLE')

valign_top = ('VALIGN', (0, 0), (-1, -1), 'TOP')

1.3.2 table_styles

class pyxml2pdf.styles.table_styles.**TableStyle**

Create a collection of styling information about the table to create

Beautiful colors are:

- aliceblue (nicht mit azure)
- azure (nicht mit aliceblue)
- honeydew
- ...

_custom_styles = {'heading': [('VALIGN', (0, 0), (-1, -1), 'MIDDLE'), ('BACKGROUND',

static _init_font_family()

Register the desired font with reportlab

This ensures that and as cell content work well.

column_widths

Return the column widths for the tables

Returns the list of column widths in mm

Return type List[float]

custom_styles

Return the custom styles and stylesheet for the tables

Returns

the custom styles and stylesheet in the Form

```
custom_styles = {
    "heading": List[Styles],
    "normal": List[Styles],
    "sub_heading": List[Styles],
    "stylesheet": StyleSheet1
}
```

table_width

Return the full table width

Returns the sum of all column widths in mm

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

C

D

E