PHPWord Documentation

Release 0.13.0

The PHPWord Team

Jul 25, 2017

Contents

PHPWord is a library written in pure PHP that provides a set of classes to write to and read from different document file formats. The current version of PHPWord supports Microsoft Office Open XML (OOXML or OpenXML), OASIS Open Document Format for Office Applications (OpenDocument or ODF), and Rich Text Format (RTF).

CHAPTER 1

Introduction

PHPWord is a library written in pure PHP that provides a set of classes to write to and read from different document file formats. The current version of PHPWord supports Microsoft Office Open XML (OOXML or OpenXML), OASIS Open Document Format for Office Applications (OpenDocument or ODF), and Rich Text Format (RTF).

PHPWord is an open source project licensed under the terms of LGPL version 3. PHPWord is aimed to be a high quality software product by incorporating continuous integration and unit testing. You can learn more about PHPWord by reading this Developers' Documentation and the API Documentation.

Features

- Set document properties, e.g. title, subject, and creator.
- Create document sections with different settings, e.g. portrait/landscape, page size, and page numbering
- Create header and footer for each sections
- Set default font type, font size, and paragraph style
- Use UTF-8 and East Asia fonts/characters
- Define custom font styles (e.g. bold, italic, color) and paragraph styles (e.g. centered, multicolumns, spacing) either as named style or inline in text
- Insert paragraphs, either as a simple text or complex one (a text run) that contains other elements
- Insert titles (headers) and table of contents
- Insert text breaks and page breaks
- Insert right-to-left text
- Insert and format images, either local, remote, or as page watermarks
- Insert binary OLE Objects such as Excel or Visio
- Insert and format table with customized properties for each rows (e.g. repeat as header row) and cells (e.g. background color, rowspan, colspan)

- Insert list items as bulleted, numbered, or multilevel
- Insert hyperlinks
- Insert footnotes and endnotes
- Insert drawing shapes (arc, curve, line, polyline, rect, oval)
- Insert charts (pie, doughnut, bar, line, area, scatter, radar)
- Insert form fields (textinput, checkbox, and dropdown)
- Create document from templates
- Use XSL 1.0 style sheets to transform headers, main document part, and footers of an OOXML template
- ... and many more features on progress

File formats

Below are the supported features for each file formats.

Writers

Features		OOXML	ODF	RTF	HTML	PDF
Document Properties	Standard	✓	✓	✓	✓	✓
	Custom	✓	✓			
Element Type	Text	✓	✓	✓	✓	✓
	Text Run	✓	✓	✓	✓	✓
	Title	✓	✓		✓	✓
	Link	✓	✓	✓	✓	✓
	Preserve Text	✓				
	Text Break	✓	✓	✓	✓	✓
	Page Break	✓		✓		
	List	✓				
	Table	✓	✓	✓	✓	✓
	Image	✓	✓	✓	✓	
	Object	✓				
	Watermark	✓				
	Table of Contents	✓				
	Header	✓				
	Footer	✓				
	Footnote	✓			✓	
	Endnote	✓			✓	
Graphs	2D basic graphs	✓				
	2D advanced graphs					
	3D graphs	✓				
Math	OMML support					
	MathML support					
Bonus	Encryption					
	Protection					

Readers

Features		OOXML	DOC	ODF	RTF	HTML
Document Properties	Standard	✓				
	Custom	✓				
Element Type	Text	✓	✓	✓	✓	✓
	Text Run	✓				
	Title	✓		✓		
	Link	✓	✓			
	Preserve Text	✓				
	Text Break	✓	✓			
	Page Break	✓				
	List	✓		✓		✓
	Table	✓				✓
	Image	✓	✓			
	Object					
	Watermark					
	Table of Contents					
	Header	✓				
	Footer	✓				
	Footnote	✓				
	Endnote	✓				
Graphs	2D basic graphs					
	2D advanced graphs					
	3D graphs					
Math	OMML support					
	MathML support					
Bonus	Encryption					
	Protection					

Contributing

We welcome everyone to contribute to PHPWord. Below are some of the things that you can do to contribute.

- Read our contributing guide.
- Fork us and request a pull to the develop branch.
- Submit bug reports or feature requests to GitHub.
- Follow @PHPWord and @PHPOffice on Twitter.

CHAPTER 2

Installing/configuring

Requirements

Mandatory:

- PHP 5.3.3+
- XML Parser extension
- Zend\Escaper component
- Zend\Stdlib component
- Zend\Validator component

Optional:

- Zip extension
- GD extension
- XMLWriter extension
- XSL extension
- dompdf library

Installation

PHPWord is installed via Composer. You just need to add dependency on PHPWord into your package.

Example:

```
{
    "require": {
       "phpoffice/phpword": "v0.13.*"
```

```
    }
}
```

If you are a developer or if you want to help us with testing then fetch the latest branch for developers. Notice: all contributions must be done against the developer branch.

Example:

```
{
    "require": {
        "phpoffice/phpword": "dev-develop"
    }
}
```

Using samples

After installation, you can browse and use the samples that we've provided, either by command line or using browser. If you can access your PHPWord library folder using browser, point your browser to the `samples` folder, e.g. `http://localhost/PhpWord/samples/`.

CHAPTER 3

General usage

Basic example

The following is a basic example of the PHPWord library. More examples are provided in the samples folder.

```
<?php
require_once 'bootstrap.php';

// Creating the new document...
$phpWord = new \PhpOffice\PhpWord\PhpWord();

/* Note: any element you append to a document must reside inside of a Section. */

// Adding an empty Section to the document...
$section = $phpWord->addSection();
// Adding Text element to the Section having font styled by default...
$section->addText(
    '"Learn from yesterday, live for today, hope for tomorrow. '
        . 'The important thing is not to stop questioning." '
        . '(Albert Einstein)'
);

/*
 * Note: it's possible to customize font style of the Text element you add in three␣
↪ways:
 * - inline;
 * - using named font style (new font style object will be implicitly created);
 * - using explicitly created font style object.
 */

// Adding Text element with font customized inline...
$section->addText(
    '"Great achievement is usually born of great sacrifice, '
        . 'and is never the result of selfishness." '
        . '(Napoleon Hill)',
```

```
    array('name' => 'Tahoma', 'size' => 10)
);

// Adding Text element with font customized using named font style...
$fontStyleName = 'oneUserDefinedStyle';
$phpWord->addFontStyle(
    $fontStyleName,
    array('name' => 'Tahoma', 'size' => 10, 'color' => '1B2232', 'bold' => true)
);
$section->addText(
    '"The greatest accomplishment is not in never falling, '
        . 'but in rising again after you fall." '
        . '(Vince Lombardi)',
    $fontStyleName
);

// Adding Text element with font customized using explicitly created font style
↪object...
$fontStyle = new \PhpOffice\PhpWord\Style\Font();
$fontStyle->setBold(true);
$fontStyle->setName('Tahoma');
$fontStyle->setSize(13);
$myTextElement = $section->addText('"Believe you can and you\'re halfway there."
↪(Theodor Roosevelt)');
$myTextElement->setFontStyle($fontStyle);

// Saving the document as OOXML file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'Word2007');
$objWriter->save('helloWorld.docx');

// Saving the document as ODF file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'ODText');
$objWriter->save('helloWorld.odt');

// Saving the document as HTML file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'HTML');
$objWriter->save('helloWorld.html');

/* Note: we skip RTF, because it's not XML-based and requires a different example. */
/* Note: we skip PDF, because "HTML-to-PDF" approach is used to create PDF documents.
↪*/
```

Settings

The `PhpOffice\PhpWord\Settings` class provides some options that will affect the behavior of PHPWord. Below are the options.

XML Writer compatibility

This option sets XMLWriter::setIndent and XMLWriter::setIndentString. The default value of this option is `true` (compatible), which is required for OpenOffice to render OOXML document correctly. You can set this option to `false` during development to make the resulting XML file easier to read.

```
\PhpOffice\PhpWord\Settings::setCompatibility(false);
```

Zip class

By default, PHPWord uses Zip extension to deal with ZIP compressed archives and files inside them. If you can't have Zip extension installed on your server, you can use pure PHP library alternative, PclZip, which included with PHPWord.

```
\PhpOffice\PhpWord\Settings::setZipClass(\PhpOffice\PhpWord\Settings::PCLZIP);
```

Output escaping

Writing documents of some formats, especially XML-based, requires correct output escaping. Without it your document may become broken when you put special characters like ampersand, quotes, and others in it.

Escaping can be performed in two ways: outside of the library by a software developer and inside of the library by built-in mechanism. By default, the built-in mechanism is disabled for backward compatibility with versions prior to v0.13.0. To turn it on set `outputEscapingEnabled` option to `true` in your PHPWord configuration file or use the following instruction at runtime:

```
\PhpOffice\PhpWord\Settings::setOutputEscapingEnabled(true);
```

Default font

By default, every text appears in Arial 10 point. You can alter the default font by using the following two functions:

```
$phpWord->setDefaultFontName('Times New Roman');
$phpWord->setDefaultFontSize(12);
```

Document information

You can set the document information such as title, creator, and company name. Use the following functions:

```
$properties = $phpWord->getDocInfo();
$properties->setCreator('My name');
$properties->setCompany('My factory');
$properties->setTitle('My title');
$properties->setDescription('My description');
$properties->setCategory('My category');
$properties->setLastModifiedBy('My name');
$properties->setCreated(mktime(0, 0, 0, 3, 12, 2014));
$properties->setModified(mktime(0, 0, 0, 3, 14, 2014));
$properties->setSubject('My subject');
$properties->setKeywords('my, key, word');
```

Measurement units

The base length unit in Open Office XML is twip. Twip means "TWentieth of an Inch Point", i.e. 1 twip = 1/1440 inch.

You can use PHPWord helper functions to convert inches, centimeters, or points to twips.

```
// Paragraph with 6 points space after
$phpWord->addParagraphStyle('My Style', array(
    'spaceAfter' => \PhpOffice\PhpWord\Shared\Converter::pointToTwip(6))
);

$section = $phpWord->addSection();
$sectionStyle = $section->getStyle();
// half inch left margin
$sectionStyle->setMarginLeft(\PhpOffice\PhpWord\Shared\Converter::inchToTwip(.5));
// 2 cm right margin
$sectionStyle->setMarginRight(\PhpOffice\PhpWord\Shared\Converter::cmToTwip(2));
```

CHAPTER 4

Containers

Containers are objects where you can put elements (texts, lists, tables, etc). There are 3 main containers, i.e. sections, headers, and footers. There are 3 elements that can also act as containers, i.e. textruns, table cells, and footnotes.

Sections

Every visible element in word is placed inside of a section. To create a section, use the following code:

```
$section = $phpWord->addSection($sectionStyle);
```

The `$sectionStyle` is an optional associative array that sets the section. Example:

```
$sectionStyle = array(
    'orientation' => 'landscape',
    'marginTop' => 600,
    'colsNum' => 2,
);
```

Page number

You can change a section page number by using the `pageNumberingStart` style of the section.

```
// Method 1
$section = $phpWord->addSection(array('pageNumberingStart' => 1));

// Method 2
$section = $phpWord->addSection();
$section->getStyle()->setPageNumberingStart(1);
```

Multicolumn

You can change a section layout to multicolumn (like in a newspaper) by using the `breakType` and `colsNum` style of the section.

```
// Method 1
$section = $phpWord->addSection(array('breakType' => 'continuous', 'colsNum' => 2));

// Method 2
$section = $phpWord->addSection();
$section->getStyle()->setBreakType('continuous');
$section->getStyle()->setColsNum(2);
```

Line numbering

You can apply line numbering to a section by using the `lineNumbering` style of the section.

```
// Method 1
$section = $phpWord->addSection(array('lineNumbering' => array()));

// Method 2
$section = $phpWord->addSection();
$section->getStyle()->setLineNumbering(array());
```

Below are the properties of the line numbering style.

- `start` Line numbering starting value
- `increment` Line number increments
- `distance` Distance between text and line numbering in twip
- `restart` Line numbering restart setting continuous|newPage|newSection

Headers

Each section can have its own header reference. To create a header use the `addHeader` method:

```
$header = $section->addHeader();
```

Be sure to save the result in a local object. You can use all elements that are available for the footer. See "Footer" section for detail. Additionally, only inside of the header reference you can add watermarks or background pictures. See "Watermarks" section.

Footers

Each section can have its own footer reference. To create a footer, use the `addFooter` method:

```
$footer = $section->addFooter();
```

Be sure to save the result in a local object to add elements to a footer. You can add the following elements to footers:

- Texts `addText` and `createTextrun`

- Text breaks
- Images
- Tables
- Preserve text

See the “Elements” section for the detail of each elements.

Other containers

Textruns, table cells, and footnotes are elements that can also act as containers. See the corresponding “Elements” section for the detail of each elements.

CHAPTER 5

Elements

Below are the matrix of element availability in each container. The column shows the containers while the rows lists the elements.

Num	Element	Section	Header	Footer	Cell	Text Run	Footnote
1	Text	v	v	v	v	v	v
2	Text Run	v	v	v	v	•	•
3	Link	v	v	v	v	v	v
4	Title	v	?	?	?	?	?
5	Preserve Text	?	v	v	v*	•	•
6	Text Break	v	v	v	v	v	v
7	Page Break	v	•	•	•	•	•
8	List	v	v	v	v	•	•
9	Table	v	v	v	v	•	•
10	Image	v	v	v	v	v	v
11	Watermark	•	v	•	•	•	•
12	Object	v	v	v	v	v	v
13	TOC	v	•	•	•	•	•
14	Footnote	v	•	•	v**	v**	•
15	Endnote	v	•	•	v**	v**	•
16	CheckBox	v	v	v	v	•	•
17	TextBox	v	v	v	v	•	•
18	Field	v	v	v	v	v	v
19	Line	v	v	v	v	v	v

Legend:

- `v`. Available.
- `v*`. Available only when inside header/footer.
- `v**`. Available only when inside section.
- `-`. Not available.
- `?`. Should be available.

Texts

Text can be added by using `addText` and `addTextRun` method. `addText` is used for creating simple paragraphs that only contain texts with the same style. `addTextRun` is used for creating complex paragraphs that contain text with different style (some bold, other italics, etc) or other elements, e.g. images or links. The syntaxes are as follow:

```
$section->addText($text, [$fontStyle], [$paragraphStyle]);
$textrun = $section->addTextRun([$paragraphStyle]);
```

- `$text`. Text to be displayed in the document.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

For available styling options see *Font* and *Paragraph*.

Titles

If you want to structure your document or build table of contents, you need titles or headings. To add a title to the document, use the `addTitleStyle` and `addTitle` method.

```
$phpWord->addTitleStyle($depth, [$fontStyle], [$paragraphStyle]);
$section->addTitle($text, [$depth]);
```

- `depth`.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.
- `$text`. Text to be displayed in the document.

It's necessary to add a title style to your document because otherwise the title won't be detected as a real title.

Links

You can add Hyperlinks to the document by using the function addLink:

```
$section->addLink($linkSrc, [$linkName], [$fontStyle], [$paragraphStyle]);
```

- `$linkSrc`. The URL of the link.
- `$linkName`. Placeholder of the URL that appears in the document.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

Preserve texts

The `addPreserveText` method is used to add a page number or page count to headers or footers.

```
$footer->addPreserveText('Page {PAGE} of {NUMPAGES}.');
```

Breaks

Text breaks

Text breaks are empty new lines. To add text breaks, use the following syntax. All parameters are optional.

```
$section->addTextBreak([$breakCount], [$fontStyle], [$paragraphStyle]);
```

- `$breakCount`. How many lines.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

Page breaks

There are two ways to insert a page breaks, using the `addPageBreak` method or using the `pageBreakBefore` style of paragraph.

:: code-block:: php

    \$section->addPageBreak();

Lists

To add a list item use the function `addListItem`.

Basic usage:

```
$section->addListItem($text, [$depth], [$fontStyle], [$listStyle], [$paragraphStyle]);
```

Parameters:

- `$text`. Text that appears in the document.
- `$depth`. Depth of list item.
- `$fontStyle`. See *Font*.
- **`$listStyle`. List style of the current element TYPE_NUMBER,** TYPE_ALPHANUM, TYPE_BULLET_FILLED, etc. See list of constants in PHPWord_Style_ListItem.
- `$paragraphStyle`. See *Paragraph*.

Advanced usage:

You can also create your own numbering style by changing the `$listStyle` parameter with the name of your numbering style.

```
$phpWord->addNumberingStyle(
    'multilevel',
    array(
        'type' => 'multilevel',
        'levels' => array(
            array('format' => 'decimal', 'text' => '%1.', 'left' => 360, 'hanging' => 360, 'tabPos' => 360),
            array('format' => 'upperLetter', 'text' => '%2.', 'left' => 720, 'hanging' => 360, 'tabPos' => 720),
        )
    )
);
$section->addListItem('List Item I', 0, null, 'multilevel');
$section->addListItem('List Item I.a', 1, null, 'multilevel');
$section->addListItem('List Item I.b', 1, null, 'multilevel');
$section->addListItem('List Item II', 0, null, 'multilevel');
```

For available styling options see *Numbering level*.

Tables

To add tables, rows, and cells, use the `addTable`, `addRow`, and `addCell` methods:

```
$table = $section->addTable([$tableStyle]);
$table->addRow([$height], [$rowStyle]);
$cell = $table->addCell($width, [$cellStyle]);
```

Table style can be defined with `addTableStyle`:

```
$tableStyle = array(
    'borderColor' => '006699',
    'borderSize'  => 6,
    'cellMargin'  => 50
);
$firstRowStyle = array('bgColor' => '66BBFF');
$phpWord->addTableStyle('myTable', $tableStyle, $firstRowStyle);
$table = $section->addTable('myTable');
```

For available styling options see *Table*.

Cell span

You can span a cell on multiple columns by using `gridSpan` or multiple rows by using `vMerge`.

```
$cell = $table->addCell(200);
$cell->getStyle()->setGridSpan(5);
```

See `Sample_09_Tables.php` for more code sample.

Images

To add an image, use the `addImage` method to sections, headers, footers, textruns, or table cells.

```
$section->addImage($src, [$style]);
```

- `$src`. String path to a local image or URL of a remote image.
- `$style`. See *Image*.

Examples:

```
$section = $phpWord->addSection();
$section->addImage(
    'mars.jpg',
    array(
        'width'         => 100,
        'height'        => 100,
        'marginTop'     => -1,
        'marginLeft'    => -1,
```

```
        'wrappingStyle' => 'behind'
    )
);
$footer = $section->addFooter();
$footer->addImage('http://example.com/image.php');
$textrun = $section->addTextRun();
$textrun->addImage('http://php.net/logo.jpg');
```

Watermarks

To add a watermark (or page background image), your section needs a header reference. After creating a header, you can use the `addWatermark` method to add a watermark.

```
$section = $phpWord->addSection();
$header = $section->addHeader();
$header->addWatermark('resources/_earth.jpg', array('marginTop' => 200, 'marginLeft'␣
↪=> 55));
```

Objects

You can add OLE embeddings, such as Excel spreadsheets or PowerPoint presentations to the document by using `addObject` method.

```
$section->addObject($src, [$style]);
```

Table of contents

To add a table of contents (TOC), you can use the `addTOC` method. Your TOC can only be generated if you have add at least one title (See "Titles").

```
$section->addTOC([$fontStyle], [$tocStyle], [$minDepth], [$maxDepth]);
```

- `$fontStyle`. See font style section.
- `$tocStyle`. See available options below.
- `$minDepth`. Minimum depth of header to be shown. Default 1.
- `$maxDepth`. Maximum depth of header to be shown. Default 9.

Options for `$tocStyle`:

- `tabLeader`. Fill type between the title text and the page number. Use the defined constants in PHPWord_Style_TOC.
- `tabPos`. The position of the tab where the page number appears in twips.
- `indent`. The indent factor of the titles in twips.

Footnotes & endnotes

You can create footnotes with `addFootnote` and endnotes with `addEndnote` in texts or textruns, but it's recommended to use textrun to have better layout. You can use `addText`, `addLink`, `addTextBreak`, `addImage`, `addObject` on footnotes and endnotes.

On textrun:

```
$textrun = $section->addTextRun();
$textrun->addText('Lead text.');
$footnote = $textrun->addFootnote();
$footnote->addText('Footnote text can have ');
$footnote->addLink('http://test.com', 'links');
$footnote->addText('.');
$footnote->addTextBreak();
$footnote->addText('And text break.');
$textrun->addText('Trailing text.');
$endnote = $textrun->addEndnote();
$endnote->addText('Endnote put at the end');
```

On text:

```
$section->addText('Lead text.');
$footnote = $section->addFootnote();
$footnote->addText('Footnote text.');
```

The footnote reference number will be displayed with decimal number starting from 1. This number use `FooterReference` style which you can redefine by `addFontStyle` method. Default value for this style is `array('superScript' => true);`

Checkboxes

Checkbox elements can be added to sections or table cells by using `addCheckBox`.

```
$section->addCheckBox($name, $text, [$fontStyle], [$paragraphStyle])
```

- `$name`. Name of the check box.
- `$text`. Text to be displayed in the document.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

Textboxes

To be completed

Fields

To be completed

Line

Line elements can be added to sections by using `addLine`.

```
$lineStyle = array('weight' => 1, 'width' => 100, 'height' => 0, 'color' => 635552);
$section->addLine($lineStyle)
```

Available line style attributes:

- `weight`. Line width in twips.
- `color`. Defines the color of stroke.
- `dash`. Line types: dash, rounddot, squaredot, dashdot, longdash, longdashdot, longdashdotdot.
- `beginArrow`. Start type of arrow: block, open, classic, diamond, oval.
- `endArrow`. End type of arrow: block, open, classic, diamond, oval.
- `width`. Line-object width in pt.
- `height`. Line-object height in pt.
- `flip`. Flip the line element: true, false.

CHAPTER 6

Styles

Section

Available Section style options:

- `borderBottomColor`. Border bottom color.
- `borderBottomSize`. Border bottom size (in twips).
- `borderLeftColor`. Border left color.
- `borderLeftSize`. Border left size (in twips).
- `borderRightColor`. Border right color.
- `borderRightSize`. Border right size (in twips).
- `borderTopColor`. Border top color.
- `borderTopSize`. Border top size (in twips).
- `breakType`. Section break type (nextPage, nextColumn, continuous, evenPage, oddPage).
- `colsNum`. Number of columns.
- `colsSpace`. Spacing between columns.
- `footerHeight`. Spacing to bottom of footer.
- `gutter`. Page gutter spacing.
- `headerHeight`. Spacing to top of header.
- `marginTop`. Page margin top (in twips).
- `marginLeft`. Page margin left (in twips).
- `marginRight`. Page margin right (in twips).
- `marginBottom`. Page margin bottom (in twips).
- `orientation`. Page orientation (`portrait`, which is default, or `landscape`).

- `pageSizeH`. Page height (in twips). Implicitly defined by `orientation` option. Any changes are discouraged.
- `pageSizeW`. Page width (in twips). Implicitly defined by `orientation` option. Any changes are discouraged.

Font

Available Font style options:

- `allCaps`. All caps, *true* or *false*.
- `bgColor`. Font background color, e.g. *FF0000*.
- `bold`. Bold, *true* or *false*.
- `color`. Font color, e.g. *FF0000*.
- `doubleStrikethrough`. Double strikethrough, *true* or *false*.
- `fgColor`. Font highlight color, e.g. *yellow*, *green*, *blue*.
- `hint`. Font content type, *default*, *eastAsia*, or *cs*.
- `italic`. Italic, *true* or *false*.
- `name`. Font name, e.g. *Arial*.
- `rtl`. Right to Left language, *true* or *false*.
- `size`. Font size, e.g. *20*, *22*.
- `smallCaps`. Small caps, *true* or *false*.
- `strikethrough`. Strikethrough, *true* or *false*.
- `subScript`. Subscript, *true* or *false*.
- `superScript`. Superscript, *true* or *false*.
- `underline`. Underline, *dash*, *dotted*, etc.

Paragraph

Available Paragraph style options:

- `alignment`. Supports all alignment modes since 1st Edition of ECMA-376 standard up till ISO/IEC 29500:2012.

See `\PhpOffice\PhpWord\SimpleType\Jc` class for the details. - `basedOn`. Parent style. - `hanging`. Hanging by how much. - `indent`. Indent by how much. - `keepLines`. Keep all lines on one page, *true* or *false*. - `keepNext`. Keep paragraph with next paragraph, *true* or *false*. - `lineHeight`. Text line height, e.g. *1.0*, *1.5*, etc. - `next`. Style for next paragraph. - `pageBreakBefore`. Start paragraph on next page, *true* or *false*. - `spaceBefore`. Space before paragraph. - `spaceAfter`. Space after paragraph. - `tabs`. Set of custom tab stops. - `widowControl`. Allow first/last line to display on a separate page, *true* or *false*.

Table

Available Table style options:

- `alignment`. Supports all alignment modes since 1st Edition of ECMA-376 standard up till ISO/IEC 29500:2012.

See `\PhpOffice\PhpWord\SimpleType\JcTable` and `\PhpOffice\PhpWord\SimpleType\Jc` classes for the details. - `bgColor`. Background color, e.g. '9966CC'. - `border(Top|Right|Bottom|Left)Color`. Border color, e.g. '9966CC'. - `border(Top|Right|Bottom|Left)Size`. Border size in twips. - `cellMargin(Top|Right|Bottom|Left)`. Cell margin in twips. - `width`. Table width in percent.

Available Row style options:

- `cantSplit`. Table row cannot break across pages, *true* or *false*.
- `exactHeight`. Row height is exact or at least.
- `tblHeader`. Repeat table row on every new page, *true* or *false*.

Available Cell style options:

- `bgColor`. Background color, e.g. '9966CC'.
- `border(Top|Right|Bottom|Left)Color`. Border color, e.g. '9966CC'.
- `border(Top|Right|Bottom|Left)Size`. Border size in twips.
- `gridSpan`. Number of columns spanned.
- `textDirection(btLr|tbRl)`. Direction of text. You can use constants `\PhpOffice\PhpWord\Style\Cell::TEXT_DIR_BTLR` and `\PhpOffice\PhpWord\Style\Cell::TEXT_DIR_TBRL`
- `valign`. Vertical alignment, *top*, *center*, *both*, *bottom*.
- `vMerge`. *restart* or *continue*.
- `width`. Cell width in twips.

Image

Available Image style options:

- `alignment`. See `\PhpOffice\PhpWord\SimpleType\Jc` class for the details.
- `height`. Height in pixels.
- `marginLeft`. Left margin in inches, can be negative.
- `marginTop`. Top margin in inches, can be negative.
- `width`. Width in pixels.
- `wrappingStyle`. Wrapping style, *inline*, *square*, *tight*, *behind*, or *infront*.

Numbering level

Available NumberingLevel style options:

- `alignment`. Supports all alignment modes since 1st Edition of ECMA-376 standard up till ISO/IEC 29500:2012.

See `\PhpOffice\PhpWord\SimpleType\Jc` class for the details. - `font`. Font name. - `format`. Numbering format bullet|decimal|upperRoman|lowerRoman|upperLetter|lowerLetter. - `hanging`. See paragraph style. - `hint`. See font style. - `left`. See paragraph style. - `restart`. Restart numbering level symbol. - `start`. Starting value. - `suffix`. Content between numbering symbol and paragraph text tab|space|nothing. - `tabPos`. See paragraph style. - `text`. Numbering level text e.g. %1 for nonbullet or bullet character.

CHAPTER 7

Templates processing

You can create an OOXML document template with included search-patterns (macros) which can be replaced by any value you wish. Only single-line values can be replaced.

To deal with a template file, use `new TemplateProcessor` statement. After TemplateProcessor instance creation the document template is copied into the temporary directory. Then you can use `TemplateProcessor::setValue` method to change the value of a search pattern. The search-pattern model is: `${search-pattern}`.

Example:

```
$templateProcessor = new TemplateProcessor('Template.docx');
$templateProcessor->setValue('Name', 'John Doe');
$templateProcessor->setValue(array('City', 'Street'), array('Detroit', '12th Street
↪'));
```

It is not possible to directly add new OOXML elements to the template file being processed, but it is possible to transform headers, main document part, and footers of the template using XSLT (see `TemplateProcessor::applyXslStyleSheet`).

See `Sample_07_TemplateCloneRow.php` for example on how to create multirow from a single row in a template by using `TemplateProcessor::cloneRow`.

See `Sample_23_TemplateBlock.php` for example on how to clone a block of text using `TemplateProcessor::cloneBlock` and delete a block of text using `TemplateProcessor::deleteBlock`.

CHAPTER 8

Writers & readers

OOXML

The package of OOXML document consists of the following files.

- _rels/
  - .rels
- docProps/
  - app.xml
  - core.xml
  - custom.xml
- word/
  - rels/
    - document.rels.xml
  - media/
  - theme/
    - theme1.xml
  - document.xml
  - fontTable.xml
  - numbering.xml
  - settings.xml
  - styles.xml
  - webSettings.xml
- [Content_Types].xml

OpenDocument

Package

The package of OpenDocument document consists of the following files.

- META-INF/
  - manifest.xml
- Pictures/
- content.xml
- meta.xml
- styles.xml

content.xml

The structure of `content.xml` is described below.

- office:document-content
  - office:font-facedecls
  - office:automatic-styles
  - office:body
    - office:text
      - draw:*
      - office:forms
      - table:table
      - text:list
      - text:numbered-paragraph
      - text:p
      - text:table-of-contents
      - text:section
    - office:chart
    - office:image
    - office:drawing

styles.xml

The structure of `styles.xml` is described below.

- office:document-styles
  - office:styles
  - office:automatic-styles

  - office:master-styles
    * office:master-page

RTF

To be completed.

HTML

To be completed.

PDF

To be completed.

CHAPTER 9

Recipes

Create float left image

Use absolute positioning relative to margin horizontally and to line vertically.

```
$imageStyle = array(
    'width' => 40,
    'height' => 40,
    'wrappingStyle' => 'square',
    'positioning' => 'absolute',
    'posHorizontalRel' => 'margin',
    'posVerticalRel' => 'line',
);
$textrun->addImage('resources/_earth.jpg', $imageStyle);
$textrun->addText($lipsumText);
```

Download the produced file automatically

Use `php://output` as the filename.

```
$phpWord = new \PhpOffice\PhpWord\PhpWord();
$section = $phpWord->createSection();
$section->addText('Hello World!');
$file = 'HelloWorld.docx';
header("Content-Description: File Transfer");
header('Content-Disposition: attachment; filename="' . $file . '"');
header('Content-Type: application/vnd.openxmlformats-officedocument.wordprocessingml.
↪document');
header('Content-Transfer-Encoding: binary');
header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
header('Expires: 0');
```

```
$xmlWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'Word2007');
$xmlWriter->save("php://output");
```

Create numbered headings

Define a numbering style and title styles, and match the two styles (with `pStyle` and `numStyle`) like below.

```
$phpWord->addNumberingStyle(
    'hNum',
    array('type' => 'multilevel', 'levels' => array(
        array('pStyle' => 'Heading1', 'format' => 'decimal', 'text' => '%1'),
        array('pStyle' => 'Heading2', 'format' => 'decimal', 'text' => '%1.%2'),
        array('pStyle' => 'Heading3', 'format' => 'decimal', 'text' => '%1.%2.%3'),
        )
    )
);
$phpWord->addTitleStyle(1, array('size' => 16), array('numStyle' => 'hNum', 'numLevel
↪' => 0));
$phpWord->addTitleStyle(2, array('size' => 14), array('numStyle' => 'hNum', 'numLevel
↪' => 1));
$phpWord->addTitleStyle(3, array('size' => 12), array('numStyle' => 'hNum', 'numLevel
↪' => 2));

$section->addTitle('Heading 1', 1);
$section->addTitle('Heading 2', 2);
$section->addTitle('Heading 3', 3);
```

Add a link within a title

Apply 'HeadingN' paragraph style to TextRun or Link. Sample code:

```
$phpWord = new \PhpOffice\PhpWord\PhpWord();
$phpWord->addTitleStyle(1, array('size' => 16, 'bold' => true));
$phpWord->addTitleStyle(2, array('size' => 14, 'bold' => true));
$phpWord->addFontStyle('Link', array('color' => '0000FF', 'underline' => 'single'));

$section = $phpWord->addSection();

// Textrun
$textrun = $section->addTextRun('Heading1');
$textrun->addText('The ');
$textrun->addLink('https://github.com/PHPOffice/PHPWord', 'PHPWord', 'Link');

// Link
$section->addLink('https://github.com/', 'GitHub', 'Link', 'Heading2');
```

Remove [Compatibility Mode] text in the MS Word title bar

Use the `Metadata\Compatibility\setOoxmlVersion(n)` method with `n` is the version of Office (14 = Office 2010, 15 = Office 2013).

```
$phpWord->getCompatibility()->setOoxmlVersion(15);
```

CHAPTER 10

Frequently asked questions

How contribute to PHPWord?

- Improve the documentation (Sphinx Format)

Is this the same with PHPWord that I found in CodePlex?

No. This one is much better with tons of new features that you can't find in PHPWord 0.6.3. The development in CodePlex is halted and switched to GitHub to allow more participation from the crowd. The more the merrier, right?

I've been running PHPWord from CodePlex flawlessly, but I can't use the latest PHPWord from GitHub. Why?

PHPWord requires PHP 5.3+ since 0.8, while PHPWord 0.6.3 from CodePlex can run with PHP 5.2. There's a lot of new features that we can get from PHP 5.3 and it's been around since 2009! You should upgrade your PHP version to use PHPWord 0.8+.

CHAPTER 11

Credits

CHAPTER 12

References

ISO/IEC 29500, Third edition, 2012-09-01

- Part 1: Fundamentals and Markup Language Reference
- Part 2: Open Packaging Conventions
- Part 3: Markup Compatibility and Extensibility
- Part 4: Transitional Migration Features

Formal specifications

- Oasis OpenDocument Standard Version 1.2
- Rich Text Format (RTF) Specification, version 1.9.1

Other resources

- DocumentFormat.OpenXml.Wordprocessing Namespace on MSDN

CHAPTER 13

Indices and tables

- genindex
- modindex
- search