PhpPresentation Documentation

Release 0.12.0

The PhpPresentation Team

Jul 15, 2017

Contents

PHPPresentation is a library written in pure PHP that provides a set of classes to write to different presentation file formats, i.e. OpenXML (.pptx) and OpenDocument (.odp). PHPPresentation is an open source project licensed under LGPL.

CHAPTER 1

Introduction

PHPPresentation is a library written in pure PHP that provides a set of classes to write to different presentation file formats, i.e. Microsoft *Office Open XML <http://en.wikipedia.org/wiki/Office_Open_XML>* (.pptx) and OASIS Open Document Format for Office Applications (.odp).

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

Features

- Create an in-memory presentation representation
- Set presentation meta data (author, title, description, etc)
- Add slides from scratch or from existing one
- Supports different fonts and font styles
- Supports different formatting, styles, fills, gradients
- Supports hyperlinks and rich-text strings
- Add images with different styles (positioning, rotation, shadow)
- Set printing options (header, footer, page margins, paper size, orientation)
- Output to different file formats: PowerPoint 2007 (.pptx), OpenDocument Presentation (.odp), Serialized Spreadsheet)
- ... and lots of other things!

File formats

Below are the supported features for each file formats.

Writers

Features		PPTX	ODP	HTML	PDF
Document	Mark as final	✓			
Document Properties	Standard	✓	✓		
	Custom				
Slides		✓	✓		
	Name		✓		
Element Shape	Image	✓	✓		
	Hyperlink	✓	✓		
	Line	✓	✓		
	MemoryImage	✓	✓		
	RichText	✓	✓		
	Table	✓	✓		
	Text	✓	✓		
Charts	Bar3D	✓	✓		
	Line	✓	✓		
	Pie3D	✓	✓		
	Scatter	✓	✓		

Readers

Features		PPTX	ODP	HTML	PDF	PPT
Document	Mark as final	✓				
Document Properties	Standard	✓	✓			
	Custom					
Slides		✓	✓			
	Name		✓			
Element Shape	Image	✓	✓			✓
	Hyperlink	✓	✓			✓
	RichText	✓	✓			✓
	Table					
	Text	✓	✓			✓
Charts	Bar3D					
	Line					
	Pie3D					
	Scatter					

Contributing

We welcome everyone to contribute to PHPPresentation. 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 @PHPOffice on Twitter

CHAPTER 2

Installing/configuring

Requirements

Mandatory:

- PHP 5.3+
- PHP Zip extension
- PHP XML Parser extension

Optional PHP extensions:

- XMLWriter

Installation

There are two ways to install PHPPresentation, i.e. via Composer or manually by downloading the library.

Using Composer

To install via Composer, add the following lines to your `composer.json`:

```
{
    "require": {
       "phpoffice/phppresentation": "dev-master"
    }
}
```

Manual install

To install manually, download PHPOfficePHPPresentation package from GitHub and download PHPOfficeCommon package from GitHub. Extract the package and put the contents to your machine.

```
require_once 'path/to/PhpPresentation/src/PhpPresentation/Autoloader.php';
\PhpOffice\PhpPresentation\Autoloader::register();

require_once 'path/to/PhpOffice/Common/src/Common/Autoloader.php';
\PhpOffice\Common\Autoloader::register();
```

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 PHPPresentation library folder using browser, point your browser to the `samples` folder, e.g. `http://localhost/PhpPresentation/samples/`.

CHAPTER 3

General usage

Basic example

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

```
require_once 'src/PhpPresentation/Autoloader.php';
\PhpOffice\PhpPresentation\Autoloader::register();

$objPHPPresentation = new PhpPresentation();

// Create slide
$currentSlide = $objPHPPresentation->getActiveSlide();

// Create a shape (drawing)
$shape = $currentSlide->createDrawingShape();
$shape->setName('PHPPresentation logo')
      ->setDescription('PHPPresentation logo')
      ->setPath('./resources/phppresentation_logo.gif')
      ->setHeight(36)
      ->setOffsetX(10)
      ->setOffsetY(10);
$shape->getShadow()->setVisible(true)
                   ->setDirection(45)
                   ->setDistance(10);

// Create a shape (text)
$shape = $currentSlide->createRichTextShape()
      ->setHeight(300)
      ->setWidth(600)
      ->setOffsetX(170)
      ->setOffsetY(180);
$shape->getActiveParagraph()->getAlignment()->setHorizontal( Alignment::HORIZONTAL_
↪CENTER );
$textRun = $shape->createTextRun('Thank you for using PHPPresentation!');
$textRun->getFont()->setBold(true)
```

```
                    ->setSize(60)
                    ->setColor( new Color( 'FFE06B20' ) );

$oWriterPPTX = IOFactory::createWriter($objPHPPresentation, 'PowerPoint2007');
$oWriterPPTX->save(__DIR__ . "/sample.pptx");
$oWriterODP = IOFactory::createWriter($objPHPPresentation, 'ODPresentation');
$oWriterODP->save(__DIR__ . "/sample.odp");
```

Document information

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

```
$properties = $objPHPPresentation->getProperties();
$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');
```

Presentation Properties

You can define some properties which are relative to the presentation, like the zoom or the thumbnail.

Comments

You can define if the presentation display or not the comments with the method `setCommentVisible`.

```
$oPresentation = new PhpPresentation();
$oProperties = $oPresentation->getPresentationProperties();
// Get the display for comment
var_export($oProperties->isCommentVisible());
// Output : false
// Enable the display for comment
$oProperties->setCommentVisible(true);
// Get the display for comment
var_export($oProperties->isCommentVisible());
// Output : true
```

Last View

You can define the last view of the presentation with the method `setLastView`.

```
$oPresentation = new PhpPresentation();
$oProperties = $oPresentation->getPresentationProperties();
// Get the last view of the presentation
```

```
echo $oProperties->getZoom();
// Output : PresentationProperties::VIEW_SLIDE
// Set the last view of the presentation
$oProperties->setLastView(PresentationProperties::VIEW_NOTES);
// Get the last view of the presentation
echo $oProperties->getZoom();
// Output : PresentationProperties::VIEW_NOTES
```

Thumbnail

You can define the thumbnail of the presentation with the method `setThumbnailPath`.

```
$oPresentation = new PhpPresentation();
$oProperties = $oPresentation->getPresentationProperties();
// Set path of the thumbnail
$oProperties->setThumbnailPath(__DIR__.'\resources\phppowerpoint_logo.gif');
// Get path of the thumbnail
echo $oProperties->getThumbnailPath();
```

Zoom

You can define the zoom of the presentation with the method `setZoom`.

```
$oPresentation = new PhpPresentation();
$oProperties = $oPresentation->getPresentationProperties();
// Get zoom of the presentation
echo $oProperties->getZoom();
// Output : 1
// Set zoom of the presentation (3 = 300%)
$oProperties->setZoom(3);
// Get zoom of the presentation
echo $oProperties->getZoom();
// Output : 3
```

CHAPTER 4

Slides

Slides are pages in a presentation. Slides are stored as a zero based array in `PHPPresentation` object. Use `createSlide` to create a new slide and retrieve the slide for other operation such as creating shapes for that slide.

Name

By default, a slide has not a name. You can define it with the method `setName`.

```
$oSlide = $oPHPPresentation->createSlide();
$oSlide->setName('Title of the slide');
```

Visibility

By default, a slide is visible. You can define it with the method `setIsVisible`.

```
$oSlide = $oPHPPresentation->createSlide();
$oSlide->setIsVisible(false);
var_dump($oSlide->isVisible());
```

CHAPTER 5

Shapes

Shapes are objects that can be added to a slide. There are five types of shapes that can be used, i.e. [rich text](#rich-text), [line](#line), [chart](#chart), [drawing](#drawing), and [table](#table). Read the corresponding section of this manual for detail information of each shape.

Every shapes have common properties that you can set by using fluent interface.

- `width` in pixels
- `height` in pixels
- `offsetX` in pixels
- `offsetY` in pixels
- `rotation` in degrees
- `fill` see *[Fill](#fill)*
- `border` see *[Border](#border)*
- `shadow` see *[Shadow](#shadow)*
- `hyperlink`

Example:

```
$richtext = $slide->createRichTextShape()
        ->setHeight(300)
        ->setWidth(600)
        ->setOffsetX(170)
        ->setOffsetY(180);
```

Line

To create a line, use *createLineShape* method of slide.

CHAPTER 6

Styles

Fill

Use this style to define fill of a shape as example below.

```
$shape->getFill()
        ->setFillType(Fill::FILL_GRADIENT_LINEAR)
        ->setRotation(270)
        ->setStartColor(new Color('FFCCCCCC'))
        ->setEndColor(new Color('FFFFFFFF'));
```

Properties:

- `fillType`
- `rotation`
- `startColor`
- `endColor`

Border

Use this style to define border of a shape as example below.

```
$shape->getBorder()
        ->setLineStyle(Border::LINE_SINGLE)
        ->setLineWidth(4)
        ->getColor()->setARGB('FFC00000');
```

Properties:

- `lineWidth`
- `lineStyle`

- dashStyle
- color

Shadow

Use this style to define shadow of a shape as example below.

```
$shape->getShadow()
        ->setVisible(true)
        ->setDirection(45)
        ->setDistance(10);
```

Properties:

- visible
- blurRadius
- distance
- direction
- alignment
- color
- alpha

Alignment

- horizontal
- vertical
- level
- indent
- marginLeft
- marginRight

Font

- name
- bold
- italic
- superScript
- subScript
- underline
- strikethrough
- color

Bullet

- bulletType
- bulletFont
- bulletChar
- bulletNumericStyle
- bulletNumericStartAt

Color

Colors can be applied to different objects, e.g. font or border.

```
$textRun = $shape->createTextRun('Text');
$textRun->getFont()->setColor(new Color('C00000'));
```

CHAPTER 7

Writers

ODPresentation

The name of the writer is `ODPresentation`.

```
$oWriter = IOFactory::createWriter($oPhpPresentation, 'PowerPoint2007');
$oWriter->save(__DIR__ . '/sample.pptx');
```

PowerPoint2007

The name of the writer is `PowerPoint2007`.

```
$oWriter = IOFactory::createWriter($oPhpPresentation, 'PowerPoint2007');
$oWriter->save(__DIR__ . '/sample.pptx');
```

You can change the ZIP Adapter for the writer. By default, the ZIP Adapter is ZipArchiveAdapter.

```
use PhpOffice\Common\Adapter\Zip\PclZipAdapter;
use PhpOffice\Common\Adapter\Zip\ZipArchiveAdapter;

$oWriter = IOFactory::createWriter($oPhpPresentation, 'PowerPoint2007');
$oWriter->setZipAdapter(PclZipAdapter);
$oWriter->save(__DIR__ . '/sample.pptx');
```

Serialized

The name of the writer is `Serialized`.

```
$oWriter = IOFactory::createWriter($oPhpPresentation, 'Serialized');
$oWriter->save(__DIR__ . '/sample.phppt');
```

You can change the ZIP Adapter for the writer. By default, the ZIP Adapter is ZipArchiveAdapter.

```
use PhpOffice\Common\Adapter\Zip\PclZipAdapter;
use PhpOffice\Common\Adapter\Zip\ZipArchiveAdapter;

$oWriter = IOFactory::createWriter($oPhpPresentation, 'Serialized');
$oWriter->setZipAdapter(PclZipAdapter);
$oWriter->save(__DIR__ . '/sample.phppt');
```

CHAPTER 8

Readers

ODPresentation

The name of the reader is `ODPresentation`.

```
$oReader = IOFactory::createReader('ODPresentation');
$oReader->load(__DIR__ . '/sample.odp');
```

PowerPoint97

The name of the reader is `PowerPoint97`.

```
$oReader = IOFactory::createReader('PowerPoint97');
$oReader->load(__DIR__ . '/sample.ppt');
```

PowerPoint2007

The name of the reader is `PowerPoint2007`.

```
$oReader = IOFactory::createReader('PowerPoint2007');
$oReader->load(__DIR__ . '/sample.pptx');
```

Serialized

The name of the reader is `Serialized`.

```
$oReader = IOFactory::createReader('Serialized');
$oReader->load(__DIR__ . '/sample.phppt');
```

CHAPTER 9

Recipes

How to define the zoom of a presentation ?

You must define the zoom of your presentation with the method `setZoom()`

```
// Default
$zoom = $oPHPPresentation->getZoom();
// $zoom = 1

// Without parameter
$oPHPPresentation->setZoom();
$zoom = $oPHPPresentation->getZoom();
// $zoom = true

// Parameter = false
$oPHPPresentation->setZoom(2.8);
$zoom = $oPHPPresentation->getZoom();
// $zoom = 2.8
```

How to mark a presentation as final ?

You must define your presentation as it with the method `markAsFinal()`

```
// Default
$state = $oPHPPresentation->isMarkedAsFinal();
// $state = false

// Without parameter
$oPHPPresentation->markAsFinal();
$state = $oPHPPresentation->isMarkedAsFinal();
// $state = true
```

```php
// Parameter = false
$oPHPPresentation->markAsFinal(false);
$state = $oPHPPresentation->isMarkedAsFinal();
// $state = false

// Parameter = true
$oPHPPresentation->markAsFinal(true);
$state = $oPHPPresentation->isMarkedAsFinal();
// $state = true
```

CHAPTER 10

Frequently asked questions

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

No. This one is much better with tons of new features that you can't find in PHPPowerPoint 0.1. 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 PHPPowerPoint from CodePlex flawlessly, but I can't use the latest PHPPresentation from GitHub. Why?

PHPPresentation requires PHP 5.3+ since 0.2, while PHPPowerPoint 0.1 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 PHPPresentation 0.2+.

Why am I getting a class not found error?

If you have followed the instructions for either adding this package to your `composer.json` or registering the autoloader, then perhaps you forgot to include a `use` statement for the class(es) you are trying to access.

Here's an example that allows you to refer to the `MemoryDrawing` class without having to specify the full class name in your code:

```
use PhpOffice\PhpPresentation\Shape\MemoryDrawing as MemoryDrawing;
```

If you *have* followed the installation instructions and you *have* added the necessary `use` statements to your code, then maybe you are still referencing the `PHPPowerPoint` classes using the old PEAR/PSR-0 approach. The 0.1 approach to naming classes used verbose class names to avoid namespace collisions with other libraries. For example, the `MemoryDrawing` class was actually called `PHPPowerPoint_Shape_MemoryDrawing`. Version 0.2 of the library renamed the classes, moved to a namespaced approach and switched to the PSR-0 autoloader. Interestingly, old code that was still referencing classes using the verbose approach *still worked* (which was pretty cool!). This

is because the PSR-0 autoloader was correctly translating the verbose class references into the correct file name and location. However, `PHPPowerPoint` now relies exclusively on the PSR-4 autoloader, so old code that may have been referencing the classes with the verbose class names will need to be updated accordingly.

Why PHPPowerPoint become PHPPresentation ?

As Roman Syroeshko noticed us, PowerPoint is a trademark. For avoiding any problems with Microsoft, we decide to change the name to a more logic name, with our panel of readers/writers.

CHAPTER 11

Credits

Images from chart page come from the *LibreOffice Core* <*https://github.com/LibreOffice/core/tree/master/icon-themes/galaxy/chart2/res*>.

Some definitions come from the *Office Open XML* <*http://officeopenxml.com/*>.

CHAPTER 12

References

OpenXML

Known as “ISO/IEC 29500, Third edition, 2012-09-01”

ISO :

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

MSDN :

- PowerPoint Viewer
- DocumentFormat.OpenXml.Presentation Namespace on MSDN
- Open XML SDK 2.5 with Validator

OpenDocument

- Oasis OpenDocument Standard Version 1.2

PowerPoint 97

- [MS-PPT]: PowerPoint (.ppt) Binary File Format
- OffVis : Microsoft Tool for reading PPT files

CHAPTER 13

Animation

You can create multiples animations in a slide.

```
use PhpOffice\PhpPresentation\Slide\Animation;

$oAnimation1 = new Animation();
$oAnimation1->addShape($oDrawing);
$oSlide->addAnimation($oAnimation1);

$oAnimation2 = new Animation();
$oAnimation2->addShape($oRichText);
$oSlide->addAnimation($oAnimation2);
```

CHAPTER 14

Charts

To create a chart, use *createChartShape* method of Slide.

Example:

```
$chartShape = $slide->createChartShape();
```

Parts

Axis

You can define gridlines (minor and major) for each axis (X & Y). For each gridline, you can custom the width (in points), the fill type and the fill color.

```
use \PhpOffice\PhpPresentation\Shape\Chart\Gridlines;

$oLine = new Line();

$oGridLines = new Gridlines();
$oGridLines->getOutline()->setWidth(10);
$oGridLines->getOutline()->getFill()->setFillType(Fill::FILL_SOLID)->
↪setStartColor(new Color(Color::COLOR_BLUE));

$oShape = $oSlide->createChartShape();
$oShape->getPlotArea()->setType($oLine);
$oShape->getPlotArea()->getAxisX()->setMajorGridlines($oGridLines);
```

For Axis, you can define the min & max bounds with *setMinBounds* & *setMaxBounds* methods. For resetting them, you pass null as parameter to these methods.

```
use \PhpOffice\PhpPresentation\Shape\Chart\Gridlines;

$oLine = new Line();
```

```
$oShape = $oSlide->createChartShape();
$oShape->getPlotArea()->setType($oLine);
$oShape->getPlotArea()->getAxisX()->setMinBounds(0);
$oShape->getPlotArea()->getAxisX()->setMaxBounds(200);
```

You can define outline for each axis (X & Y).

```
$oLine = new Line();

$oShape = $oSlide->createChartShape();
$oShape->getPlotArea()->setType($oLine);
$oShape->getPlotArea()->getAxisX()->getOutline()->setWidth(10);
$oShape->getPlotArea()->getAxisX()->getOutline()->getFill()->setFillType(Fill::FILL_
↪SOLID)->setStartColor(new Color(Color::COLOR_BLUE));
```

For Axis Y, you can define tick mark with *setMinorTickMark* & *setMajorTickMark* methods. For resetting them, you pass Axis::TICK_MARK_NONE as parameter to these methods.

```
use \PhpOffice\PhpPresentation\Shape\Chart\Axis;

$oLine = new Line();

$oShape = $oSlide->createChartShape();
$oShape->getPlotArea()->setType($oLine);
$oShape->getPlotArea()->getAxisY()->setMinorTickMark(Axis::TICK_MARK_NONE);
$oShape->getPlotArea()->getAxisY()->setMajorTickMark(Axis::TICK_MARK_INSIDE);
```

For Axis Y, you can define unit with *setMinorUnit* & *setMajorUnit* methods. For resetting them, you pass null as parameter to these methods.

```
use \PhpOffice\PhpPresentation\Shape\Chart\Axis;

$oLine = new Line();

$oShape = $oSlide->createChartShape();
$oShape->getPlotArea()->setType($oLine);
$oShape->getPlotArea()->getAxisY()->setMinorUnit(null);
$oShape->getPlotArea()->getAxisY()->setMajorUnit(0.05);
```

You can define visibility for each axis (X & Y).

```
$oLine = new Line();

$oShape = $oSlide->createChartShape();
$oShape->getPlotArea()->setType($oLine);
$oShape->getPlotArea()->getAxisX()->setIsVisible(false);
```

Title

By default, the title of a chart is displayed. For hiding it, you define its visibility to false.

```
$oLine = new Line();
$oShape = $slide->createChartShape();
$oShape->getPlotArea()->setType($oLine);
```

```
// Hide the title
$oShape->getTitle()->setVisible(false);
```

Series

You can custom the font of a serie.

You can custom the marker of a serie, for Line & Scatter charts.

You can custom the line of a serie, for Line & Scatter charts.

You can define the position of the data label. Each position is described in MSDN

```
$oSeries = new Series('Downloads', $seriesData);
$oSeries->setLabelPosition(Series::LABEL_INSIDEEND);
```

You can define if some informations are displayed.

```
$oSeries = new Series('Downloads', $seriesData);
$oSeries->setSeparator(';');
$oSeries->setShowCategoryName(true);
$oSeries->setShowLeaderLines(true);
$oSeries->setShowLegendKey(true);
$oSeries->setShowPercentage(true);
$oSeries->setShowSeriesName(true);
$oSeries->setShowValue(true);
```

View3D

For enabling the autoscale for a shape, you must reset the height percent.

```
$oShape->getView3D()->setHeightPercent(null);
```

Types

Area

TODO

Bar & Bar3D

Gap Width

You can define the gap width between bar or columns clusters. It is defined in percent. The default value is 150%. The value must be defined between 0 and 500.

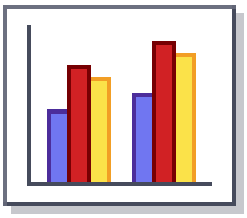
```
$oBarChart = new Bar();
$oBarChart->setGapWidthPercent(250);
```

Stacking

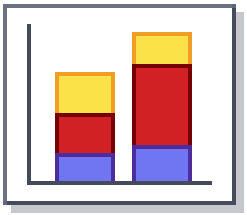
You can stack multiples series in a same chart. After adding multiples series, you can define the bar grouping with *setBarGrouping* method of AbstractTypeBar.

```
$oBarChart = new Bar();
$oBarChart->addSeries($oSeries1);
$oBarChart->addSeries($oSeries2);
$oBarChart->addSeries($oSeries3);
$oBarChart->setBarGrouping(Bar::GROUPING_CLUSTERED);
// OR
$oBarChart->setBarGrouping(Bar::GROUPING_STACKED);
// OR
$oBarChart->setBarGrouping(Bar::GROUPING_PERCENTSTACKED);
```

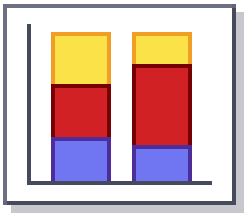
- Bar::GROUPING_CLUSTERED



- Bar::GROUPING_STACKED



- Bar::GROUPING_PERCENTSTACKED



Line

TODO

Pie & Pie3D

TODO

Scatter

TODO

CHAPTER 15

Comments

To create a comment, create an object *Comment*.

Example:

```
use PhpOffice\PhpPresentation\Shape\Comment;

$oComment = new Comment();
$oSlide->addShape($oComment);
```

You can define text and date with setters.

Example:

```
use PhpOffice\PhpPresentation\Shape\Comment;

$oComment = new Comment();
$oComment->setText('Text of the Comment');
$oComment->setDate(time());
$oSlide->addShape($oComment);
```

Author

For a comment, you can define the author.

Example:

```
use PhpOffice\PhpPresentation\Shape\Comment;
use PhpOffice\PhpPresentation\Shape\Comment\Author;

$oAuthor = new Author();
$oComment = new Comment();
$oComment->setAuthor($oAuthor);
$oSlide->addShape($oComment);
```

You can define name and initials with setters.

Example:

```
use PhpOffice\PhpPresentation\Shape\Comment;
use PhpOffice\PhpPresentation\Shape\Comment\Author;

$oAuthor = new Author();
$oAuthor->setName('Name of the author');
$oAuthor->setInitals('Nota');
$oComment = new Comment();
$oComment->setAuthor($oAuthor);
$oSlide->addShape($oComment);
```

CHAPTER 16

Drawing

To create a drawing, you have four sources : File, GD, Base64 and ZipFile.

File

To create a drawing, use *createDrawingShape* method of slide.

```
$oShape = $oSlide->createDrawingShape();
$oShape->setName('Unique name')
        ->setDescription('Description of the drawing')
        ->setPath('/path/to/drawing.filename');
```

It's an alias for :

```
use PhpOffice\PhpPresentation\Shape\Drawing\File;

$oShape = new File();
    $oShape->setName('Unique name')
            ->setDescription('Description of the drawing')
            ->setPath('/path/to/drawing.filename');
$oSlide->addShape($oShape);
```

GD

```
use PhpOffice\PhpPresentation\Shape\Drawing\Gd;

$gdImage = @imagecreatetruecolor($width, $height);

$oShape = new Gd();
$oShape->setName('Sample image')
    ->setDescription('Sample image')
```

```
    ->setImageResource($gdImage)
    ->setRenderingFunction(Drawing\Gd::RENDERING_JPEG)
    ->setMimeType(Drawing\Gd::MIMETYPE_DEFAULT);
$oSlide->addShape($oShape);
```

Base64

```
use PhpOffice\PhpPresentation\Shape\Drawing\Base64;

$oShape = new Base64();
$oShape->setName('Sample image')
    ->setDescription('Sample image')
    ->setImageResource($gdImage)
    ->setData('..........');
$oSlide->addShape($oShape);
```

ZipFile

```
use PhpOffice\PhpPresentation\Shape\Drawing\ZipFile;

$oShape = new ZipFile();
$oShape->setName('Sample image')
    ->setDescription('Sample image')
    ->setPath('zip://myzipfile.zip#path/in/zip/img.ext')
$oSlide->addShape($oShape);
```

CHAPTER 17

Media

To create a video, create an object *Media*.

Example:

```
use PhpOffice\PhpPresentation\Shape\Media;

$oMedia = new Media();
$oMedia->setPath('file.wmv');
// $oMedia->setPath('file.ogv');
$oSlide->addShape($oMedia);
```

You can define text and date with setters.

Example:

```
use PhpOffice\PhpPresentation\Shape\Media;

$oMedia = new Media();
$oMedia->setName('Name of the Media');
$oSlide->addShape($oMedia);
```

Quirks

For Windows readers, the prefered file format is WMV. For Linux readers, the prefered file format is OGV.

CHAPTER 18

RichText

Rich text shapes contain paragraphs of texts. To create a rich text shape, use `createRichTextShape` method of slide.

Below are the properties that you can set for a rich text shape.

- `wrap`
- `autoFit`
- `fontScale` : font scale (in percentage) when autoFit = RichText::AUTOFIT_NORMAL
- `lnSpcReduction` : line spacing reduction (in percentage) when autoFit = RichText::AUTOFIT_NORMAL
- `horizontalOverflow`
- `verticalOverflow`
- `upright`
- `vertical`
- `columns`
- `bottomInset` in pixels
- `leftInset` in pixels
- `rightInset` in pixels
- `topInset` in pixels
- `autoShrinkHorizontal` (boolean)
- `autoShrinkVertical` (boolean)

Properties that can be set for each paragraphs are as follow.

- `alignment` see *[Alignment](#alignment)*
- `bulletStyle` see *[Bullet](#bullet)*
- `lineSpacing` see *[LineSpacing](#linespacing)*

- `font` see *[Font](#font)*

Bullet

For a paragraph, you can define the bullet style.

Example:

```
use PhpOffice\PhpPresentation\Shape\RichText\Paragraph;
use PhpOffice\PhpPresentation\Style\Bullet;

$oParagraph = new Paragraph();
$oParagraph->getBulletStyle();
```

With the bullet style, you can define the char, the font, the color and the type.

```
use PhpOffice\PhpPresentation\Shape\RichText\Paragraph;
use PhpOffice\PhpPresentation\Style\Bullet;
use PhpOffice\PhpPresentation\Style\Color;

$oParagraph = new Paragraph();
$oParagraph->getBulletStyle()->setBulletChar('-');
$oParagraph->getBulletStyle()->setBulletType(Bullet::TYPE_BULLET);
$oParagraph->getBulletStyle()->setBulletColor(new Color(Color::COLOR_RED));
```

LineSpacing

For a paragraph, you can define the line spacing.

Example:

```
use PhpOffice\PhpPresentation\Shape\RichText\Paragraph;

$oParagraph = new Paragraph();
$oParagraph->setLineSpacing(200);
$iLineSpacing = $oParagraph->getLineSpacing();
```

Run

For a run, you can define the language.

Example:

```
use PhpOffice\PhpPresentation\Shape\RichText\Run;

$oRun = new Run();
$oRun->setLanguage('fr-FR');
```

CHAPTER 19

Tables

To create a table, use *createTableShape* method of slide.

Example:

```
$tableShape = $slide->createTableShape($columns);
```

Rows

A row is a child of a table. For creating a row, use *createRow* method of a Table shape.

```
$tableShape = $slide->createTableShape($columns);
$row = $tableShape->createRow();
```

Cells

A cell is a child of a row.

You can access cell objects with *nextCell* method of a Row object.

```
$tableShape = $slide->createTableShape($columns);
$row = $tableShape->createRow();
// Get the first cell
$cellA1 = $row->nextCell();
// Get the second cell
$cellA2 = $row->nextCell();
```

You can access cell object directly.

```
$tableShape = $slide->createTableShape($columns);
$row = $tableShape->createRow();
```

```
// Get the first cell
$cellA1 = $row->getCell(0);
// Get the second cell
$cellA2 = $row->getCell(1);
```

Define margins of a cell

Margins of cells are defined by margins of the first paragraph of cell. Margins of cells are defined in pixels.

For defining margins of cell, you can use the *setMargin** method of a Alignment object of the active paragraph of a Cell object.

```
$tableShape = $slide->createTableShape($columns);
$row = $tableShape->createRow();
$cellA1 = $row->nextCell();
$cellA1->getActiveParagraph()->getAlignment()
    ->setMarginBottom(20)
    ->setMarginLeft(40)
    ->setMarginRight(60)
    ->setMarginTop(80);
```

Define the text direction of a cell

For defining the text direction of cell, you can use the *setTextDirection* method of the *getAlignment* method of a Cell object. The width is in pixels.

```
$tableShape = $slide->createTableShape($columns);
$row = $tableShape->createRow();
$cellA1 = $row->nextCell();
$cellA1->getAlignment()->
↪setTextDirection(\PhpOffice\PhpPresentation\Style\Alignment::TEXT_DIRECTION_
↪VERTICAL_270);
```

Define the width of a cell

The width of cells are defined by the width of cell of the first row. If not defined, all cells widths are calculated from the width of the shape and the number of columns.

For defining the width of cell, you can use the *setWidth* method of a Cell object. The width is in pixels.

```
$tableShape = $slide->createTableShape($columns);
$row = $tableShape->createRow();
$cellA1 = $row->nextCell();
$cellA1->setWidth(100);
```

CHAPTER 20

Indices and tables

- genindex
- modindex
- search