DEMONSTRATION 1 - WRITING TEXT

This is a simple paragraph of text that is being used to demonstrate the automatic line wrapping that is implemented within the Wrap_Pdf class. If you check the source text (located in resources/p1.txt), you will notice that while it does have line breaks, they are not at the same point in the PDF file as they are in the source text. That's because the writeText() method replaces all line breaks with spaces. Well, it replaces Mac-style line feeds anyway... if your results are different it might be because line 355 requires some tweaking.

This snippet demonstrates how the writeText() method can deal with changing font styles and still manage to wrap things properly. **This text is bold.** *And this text is italic*. And, as you can see, the word wrapping still works properly. Yay! :-)

You will also notice that the generated PDF file has a table of contents, each entry mapping to the corresponding page within the document. In Wrap_Pdf, these entries are logged if a section name is provided to the addPage() method and the actual table of contents is constructed by the output() method based on these entries.

And while we're on the topic of groovy PDF features, you will also notice that the document properties have been set as well. Although this is fairly straightforward with Zend_Pdf, finding information on exactly what each field is supposed to contain can be surprisingly difficult. I have included a comment in the source that points to a PDF reference manual that provides the exact definitions of each field.

DEMONSTRATION 2 – AUTOMATIC PAGE BREAKING

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse rutrum volutpat erat at porta. Duis quis ipsum a erat pulvinar pellentesque. Phasellus ullamcorper semper sem, et tristique ante pharetra ac. Nam dictum enim libero. Morbi pellentesque molestie nulla, eu vulputate lacus dignissim vel. Praesent in erat ipsum. Praesent condimentum vulputate tellus nec interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Proin lacinia fermentum vulputate. Cras et purus vitae arcu gravida accumsan nec vitae risus. Ut diam lacus, mollis quis tincidunt accumsan, consequat et dui. Vestibulum nec mauris neque, vel lacinia ante. Donec posuere diam ac tellus convallis id vulputate neque sagittis. In vel lorem consequat mi condimentum auctor.

Sed consectetur hendrerit elit, a commodo velit rutrum sit amet. Fusce vitae felis eros. Nullam sit amet massa velit, at commodo nibh. Fusce eu libero ut lorem lobortis rhoncus fringilla nec lacus. Donec varius, diam et tincidunt malesuada, erat magna lacinia sapien, quis porttitor lacus nunc nec tellus. In lacinia pretium imperdiet. Vivamus ac ipsum sit amet leo feugiat posuere ut sit amet nulla. Nunc ut enim id eros tincidunt auctor. Proin nibh libero, dictum vel tempus sed, rhoncus eget sem. Praesent accumsan risus at elit dignissim eget accumsan nibh varius. Ut suscipit posuere metus, a mollis tortor porta ac. Donec fringilla imperdiet sapien, pulvinar accumsan tortor suscipit quis. Curabitur eget dui lectus. Nam orci risus, ornare vel rhoncus eu, molestie ut magna. Aenean ligula purus, imperdiet et vehicula id, vehicula quis sapien. Aliquam eu mattis velit. Mauris eget tellus arcu, vitae commodo mauris. Integer condimentum, metus ut luctus lobortis, dolor augue sollicitudin neque, vitae convallis est neque sed dui. Donec condimentum leo in lectus malesuada accumsan. Proin ac rutrum erat.

Integer adipiscing blandit tristique. Pellentesque venenatis aliquam laoreet. Duis in felis est, vitae congue orci. Nullam bibendum, urna non rhoncus hendrerit, diam velit cursus augue, id pharetra orci libero ac lectus. In ut nibh turpis. Mauris sollicitudin tincidunt nisi eget cursus. Cras tempus turpis a metus pulvinar nec mollis velit facilisis. Integer varius mauris ut purus varius laoreet. Cras imperdiet dapibus dictum. Quisque elementum risus non elit luctus fermentum vehicula elementum dui. Sed vel enim enim, eget laoreet erat.

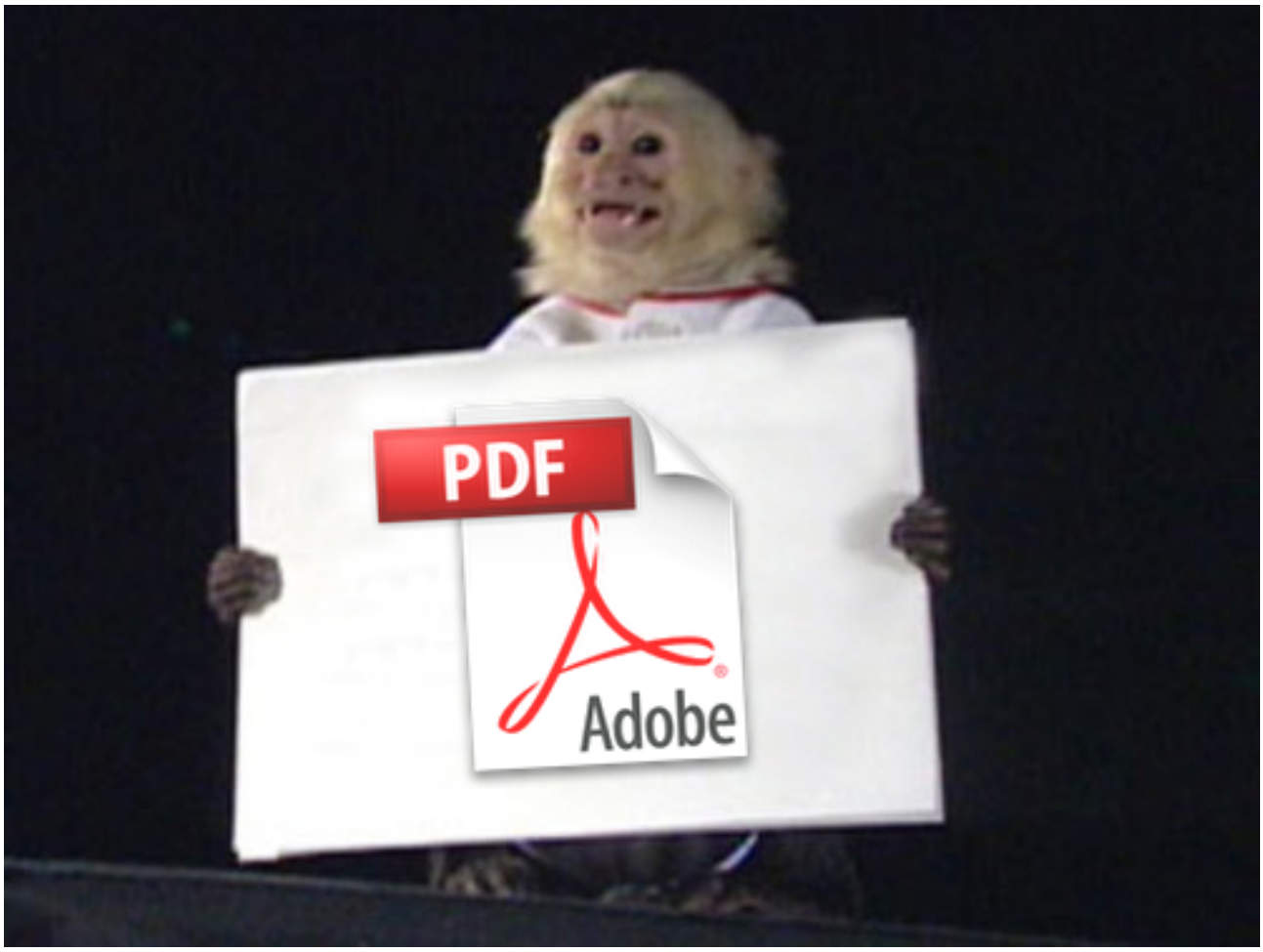
Suspendisse non justo odio. In vulputate nulla non sapien feugiat non volutpat enim bibendum. Curabitur ac porttitor neque. Cras malesuada vehicula elit, vitae facilisis mauris mattis ac. Vestibulum non lacus diam, ac pulvinar elit. Suspendisse fermentum risus luctus justo ultricies eu interdum ante eleifend. Suspendisse rutrum, libero ultrices dictum consectetur, elit enim dapibus nunc, nec ultrices arcu tortor id dui. Suspendisse mattis vestibulum viverra. Ut lobortis eleifend eros, quis malesuada leo egestas vel. Cras euismod justo aliquam ligula sodales tempor. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Integer aliquet, erat nec convallis sagittis, quam purus mollis nulla, a dictum leo mi vitae ipsum. Phasellus ut justo a enim lacinia posuere nec id arcu. Fusce ultricies nulla quis leo consectetur ut rutrum nibh bibendum. Cras egestas molestie sagittis. Proin tincidunt sagittis sapien.

Suspendisse non justo odio. In vulputate nulla non sapien feugiat non volutpat enim bibendum. Curabitur ac porttitor neque. Cras malesuada vehicula elit, vitae facilisis mauris mattis ac. Vestibulum

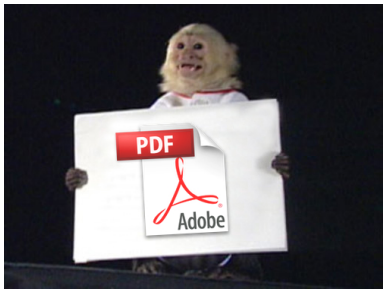
non lacus diam, ac pulvinar elit. Suspendisse fermentum risus luctus justo ultricies eu interdum ante eleifend. Suspendisse rutrum, libero ultrices dictum consectetur, elit enim dapibus nunc, nec ultrices arcu tortor id dui. Suspendisse mattis vestibulum viverra. Ut lobortis eleifend eros, quis malesuada leo egestas vel. Cras euismod justo aliquam ligula sodales tempor. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Integer aliquet, erat nec convallis sagittis, quam purus mollis nulla, a dictum leo mi vitae ipsum. Phasellus ut justo a enim lacinia posuere nec id arcu. Fusce ultricies nulla quis leo consectetur ut rutrum nibh bibendum. Cras egestas molestie sagittis. Proin tincidunt sagittis sapien.

DEMONSTRATION 3 – INCLUDING IMAGES

If we don't provide the width parameter, drawGraphic() will simply render the graphic the entire width of the page (ie. inside the margins).



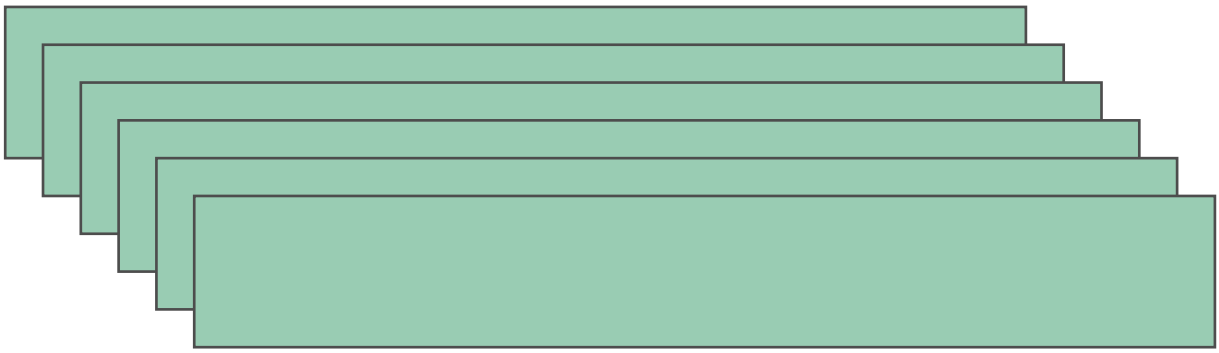
Alternatively, if we include a width (in millimetres of course), then drawGraphic() will render the image accordingly. Obviously this method could be further extended to include centering, etc.



It is also worth noting that Sane_Pdf keeps track of which images have already been added to the document and avoids loading up duplicate images in order to minimise the file size. As an example, the incremental cost of rendering the monkey a second time is only 103 bytes, as the document simply contains an additional reference to the original image with new placement information.

DEMONSTRATION 4 - DRAWING BOXES

This is a very rudimentary example of drawing boxes.



DEMONSTRATION 5 - DRAWING LINES

A common requirement is to be able to draw ruled lines on a section of the page for notes.

DEMONSTRATION 6 - ROTATED TEXT

Rotating text can be tricky, unless you rotate the page around the point at which you are intending to write the text - then its easy! :-)

This text is going up at 20 degrees

This text is going down at 20 degrees

And this is upside down!!!

This text is going straight down

This text is going straight up

DEMONSTRATION 7 - ADDING LINKS

Unfortunately when adding a link with Zend_Pdf you end up with a black border around the area that was defined as clickable. I am yet to find a way of simply rendering a snippet of text as a link. You also need to know the precise co-ordinates of the area you would like to make clickable, which kind of breaks the encapsulation I was trying to achieve, but anyway...

DEMONSTRATION 8 – DUMPING FILES

```
<?php
define( 'ZEND_PATH', '../zf1/library' ); // Obviously you'll want to set ZEND_PATH to point to the folder in which your Zend folder resides.
set_include_path( ZEND_PATH . PATH_SEPARATOR . get_include_path());
require_once ZEND_PATH . '/Zend/Loader/Autoloader.php';
Zend_Loader_Autoloader::getInstance();

require_once 'Wrap_Pdf.php';

$pdf = new Wrap_Pdf();
$pdf->setProperties( 'Sane_Pdf Example Document', $subject = 'Simplifying PDF Production with Zend_Pdf', $author = 'James Gordon' );

$pdf->addPage( 'Writing Text' );
$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 1 – WRITING TEXT' );
$pdf->ln();
$pdf->setStyle(2);
$pdf->writeText( file_get_contents( 'resources/p1.txt' ));
$pdf->ln();
$pdf->ln();
$pdf->writeText( file_get_contents( 'resources/p2.txt' ));
$pdf->setStyle(3);
$pdf->writeText( file_get_contents( 'resources/p3.txt' ));
$pdf->setStyle(4);
$pdf->writeText( file_get_contents( 'resources/p4.txt' ));
$pdf->setStyle(2);
$pdf->writeText( file_get_contents( 'resources/p5.txt' ));
$pdf->ln();
$pdf->ln();
$pdf->writeText( file_get_contents( 'resources/p6.txt' ));
$pdf->ln();
$pdf->ln();
$pdf->writeText( file_get_contents( 'resources/p7.txt' ));

$pdf->addPage( 'Automatic Page Breaking' );
$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 2 – AUTOMATIC PAGE BREAKING' );
$pdf->ln();
$pdf->setStyle(2);
$pdf->writeText( file_get_contents( 'resources/l1.txt' ));
$pdf->ln();
```

```
$pdf->ln();
$pdf->writeText( file_get_contents( 'resources/l2.txt' ));
$pdf->ln();
$pdf->ln();
$pdf->writeText( file_get_contents( 'resources/l3.txt' ));
$pdf->ln();
$pdf->ln();
$pdf->writeText( file_get_contents( 'resources/l4.txt' ));
$pdf->ln();
$pdf->ln();
$pdf->writeText( file_get_contents( 'resources/l4.txt' ));

$pdf->addPage( 'Including Images' );
$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 3 - INCLUDING IMAGES' );
$pdf->ln();
$pdf->setStyle(2);
$pdf->writeText( "If we don't provide the width parameter, drawGraphic() will simply render the
graphic the entire " .
                 'width of the page (ie. inside the margins).' );
$pdf->ln();

$pdf->drawGraphic( 'resources/GeoMonkey-PDF.jpg' );
$pdf->ln();
$pdf->writeText( 'Alternatively, if we include a width (in millimetres of course), then drawGraphic()
will render the ' .
                 'image accordingly. Obviously this method could be further extended to include
centering, etc.' );
$pdf->ln();
$pdf->drawGraphic( 'resources/GeoMonkey-PDF.jpg', 50 );
$pdf->ln();
$pdf->writeText( 'It is also worth noting that Sane_Pdf keeps track of which images have already been
added to the ' .
                 'document and avoids loading up duplicate images in order to minimise the file size.
As an example, ' .
                 'the incremental cost of rendering the monkey a second time is only 103 bytes, as the
document ' .
                 'simply contains an additional reference to the original image with new placement
information.' );

$pdf->addPage( 'Other Neat Stuff' );
$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 4 - DRAWING BOXES' );
```

```
$pdf->ln();
$pdf->setStyle(2);
$pdf->writeText( 'This is a very rudimentary example of drawing boxes.' );
$pdf->ln();
$y = $pdf->getY();

for( $c = 0; $c < 30; $c +=5 )
{
    $pdf->drawRectangle( 20 + $c, $y + $c, 155 + $c, $y + 20 + $c, 'grey', 'green' );
}

$pdf->setY( $y + 30 + $c );

$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 5 - DRAWING LINES' );
$pdf->ln();
$pdf->setStyle(2);
$pdf->writeText( 'A common requirement is to be able to draw ruled lines on a section of the page for
notes.' );
$pdf->ln();
$pdf->drawLines( 10 );
$pdf->ln();

$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 6 - ROTATED TEXT' );
$pdf->ln();
$pdf->setStyle(2);
$pdf->writeText( 'Rotating text can be tricky, unless you rotate the page around the point at which you
are intending ' .
                    'to write the text - then its easy! :-)' );
$pdf->ln();
$pdf->writeRotatedText( 'This text is going up at 20 degrees', 30, 225, 20 );
$pdf->writeRotatedText( 'This text is going down at 20 degrees', 120, 205, -20 );
$pdf->writeRotatedText( 'This text is going straight up', 200, 240, 90 );
$pdf->writeRotatedText( 'This text is going straight down', 10, 190, -90 );
$pdf->writeRotatedText( 'And this is upside down!!!', 120, 230, 180 );

$pdf->setY( 250 );
$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 7 - ADDING LINKS' );
$pdf->ln();
$pdf->setStyle(2);
$pdf->writeText( 'Unfortunately when adding a link with Zend_Pdf you end up with a black border
```

```
around the area that ' .
                'was defined as clickable. I am yet to find a way of simply rendering a snippet of text
as a link. ' .
                'You also need to know the precise co-ordinates of the area you would like to make
clickable, ' .
                'which kind of breaks the encapsulation I was trying to achieve, but anyway...' );
$pdf->addLink( 83, 255, 99, 260, 'http://framework.zend.com/manual/en/zend.pdf.html' );


$pdf->addPage( 'Dumping Files' );
$pdf->setStyle(1);
$pdf->writeText( 'DEMONSTRATION 8 - DUMPING FILES' );
$pdf->ln();
$pdf->setStyle(0);
$pdf->writeLines( file_get_contents( __FILE__ ));

$pdf->output( 'pdf_demo.pdf' );
passthru( 'open -a Preview pdf_demo.pdf' ); // For Mac users only
```