

# Zend\_Pdf\_Cell - Logan Buesching

```
<ac:macro ac:name="info"><ac:parameter ac:name="title">Zend Pdf Cell</ac:parameter></ac:macro>
<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[
<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[
```

## Zend Framework: Zend\_Pdf\_Cell Component Proposal

Proposed Component Name	Zend_Pdf_Cell
Developer Notes	<a href="http://framework.zend.com/wiki/display/ZFDEV/Zend_Pdf_Cell">http://framework.zend.com/wiki/display/ZFDEV/Zend_Pdf_Cell</a>
Proposers	logan@agoln.net
Revision	1.0 - 19 March 2008: First Draft. (wiki revision: 9)

## Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

## 1. Overview

Zend\_Pdf\_Cell is an attempt to provide additional text formatting features to PDF documents. A cell will provide the ability to format, align and position text inside of a PDF document. One or more cells may be attached to one or more PDF documents.

## 2. References

- Bug # 1254 Improve Zend\_Pdf Text-Functions
- My Personal Blog, with an implementation and description

## 3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** provide the ability to position cells.
- This component **will** provide the ability to position text within cells.
- This component **will** provide the ability to format text (font, color, size) within cells.
- This component **will** provide the ability to align text within cells.
- This component **will** provide the ability to add borders to cells.
- This component **will** provide the ability to create multiple cells on a page.

- This component **will** provide the ability to attach a cell to one or more pages.
- This component **will not** provide the ability to put graphics or images within cells.
- This component **will not** provide the ability for a cell to span more than one page.

## 4. Dependencies on Other Framework Components

- Zend\_Pdf
- Zend\_Pdf\_Resource\_Font
- Zend\_Pdf\_Page
- Zend\_Pdf\_Cmap
- Zend\_Pdf\_Color

## 5. Theory of Operation

A developer creates a new cell and attaches it to a page. Then when the information is finished, a write() is called to write the text to the PDF document.

```
pages[] =new Zend_Pdf_Page(Zend_Pdf_Page::SIZE_A4);
$font=Zend_Pdf_Font::fontWithName(Zend_Pdf_Font::FONT_TIMES_ITALIC);
$pdf->pages[0]->setFont($font,12);
//Creates a cell in the specified page
$cell=new Zend_Pdf_Cell($pdf->pages[0]);

//adds a cell in the upper left with "Hello World"
$cell->addText("Hello World");
$cell->write();

//creates a cell in the center of the page
//To do top and right, then you would
//or together POSITION_RIGHT and
//POSITION_TOP.
$cell=new Zend_Pdf_Cell($pdf->pages[0],
                        Zend_Pdf_Cell::POSITION_CENTER_X |
                        Zend_Pdf_Cell::POSITION_CENTER_Y);

//add a 1 pixel border
$cell->setBorder(1);
//align to the right
$cell->addText("The quick brown fox jumped over the lazy dog.", Zend_Pdf_Cell::ALIGN_RIGHT);
$cell->write();
```

## 6. Milestones / Tasks

- Milestone 1: Acceptance of proposal
- Milestone 2: Complete code for use case #1, #2, #3, #4
- Milestone 3: Unit tests
- Milestone 4: Complete phpDoc documents
- Milestone 5: Complete "How to use" documentation

## 7. Class Index

- Zend\_Pdf\_Cell

## 8. Use Cases

- Position a cell and text within a cell
- Align text within cell and position cell in center
- Add border
- Word wrap text

## 9. Class Skeletons

```
class Zend_Pdf_Cell {

    const ALIGN_LEFT=0;
    const ALIGN_RIGHT=1;
    const ALIGN_CENTER=2;
    const ALIGN_JUSTIFY=3;

    const POSITION_LEFT=0;
    const POSITION_RIGHT=1;
    const POSITION_TOP=2;
    const POSITION_BOTTOM=4;
    const POSITION_CENTER_X=8;
    const POSITION_CENTER_Y=16;

    /**
     * Width of the cell
     *
     * @var int
     */
    private $_width;

    /**
     * Height of the cell
     *
     * @var int
     */
    private $_height;

    /**
     * Upper left X coordinate
     *
     * @var int
     */
    private $_x;

    /**
     * Upper left Y coordinate
     *
     * @var int
     */
    private $_y;

    /**
     * Current page for the cell to belong to.
     *
     * @var Zend_Pdf_Page
     */
    private $_page;
```

```

/**
 * How the cell should be positioned on the page
 *
 * @var int
 */
private $_position;

/**
 * 3 dimensional array that stores the text in the cell.
 * The first diminsion is for the line number.
 * The second diminsion is for the section number.
 * The third diminsion is the properties for that section.
 *
 * The only properties (3rd diminsion) valid for a non-zero section (second diminsion)
 * are text, font and encoding.
 *
 * @var array
 */
private $_text;

/**
 * Keeps track of the current line number
 *
 * @var int
 */
private $_lineNumber;

/**
 * Current text section number.
 *
 * @var int
 */
private $_section;

/**
 * Current font that is being used.
 *
 * @var Zend_Pdf_Font
 */
private $_font;

/**
 * Current font's height
 *
 * @var int
 */
private $_fontSize;

/**
 * When we want to auto-calculate the height, this is the
 * height's current value.
 *
 * @var int
 */
private $_autoHeight;

/**
 * When we want to aut-calculate the width, this is the
 * width's current value.
 *
 * @var int
 */
private $_autoWidth;

/**
 * Border around the cell.
 *
 * The border has three properties associated with it:
 *

```

```

* $_border['pattern']: The type of pattern for the border.
* $_border['size']: The size, in pixels, for the border.
* $_border['color']: Color for the border.
*
* By default, there is no border.
*
* @var array
*/
private $_border;

/**
 * Adds text to the cell.
*
* The text is not written to the PDF until write() is called. Every time addText is called,
* a new text section is created.
*
* If this text is not the first text in a line, then the alignment, x and y variables are ignored
* as it makes no sense to align a bit of text from the middle of a line.
*
* @param string $text Text to add to the section
* @param mixed $alignment (Optional) How to align the text in the cell. If no alignment is
* specified, then it uses the previous line's alignment. If this is the first line in the
* cell, then it uses Zend_Pdf_Cell::ALIGN_LEFT as default.
* @param int $x (Optional) Offset of X from the relative position of this line in the cell.
* Defaults to 0
* @param string $charEncoding (Optional) Encoding of this particular section of text.
* Defaults to current locale.
* @return void
*/
public function addText($text, $alignment=null, $offset=0, $charEncoding='') {}

/**
* Adds a new line to the cell.
*
* For every new line that you want in the cell, this function must be called.
* @return void
*/
public function newLine() {}

/**
* Sets the border around the cell.
*
* A border may be placed around the cell. If the border is greater than
* one pixel in width, then the border grows outwards away from the cell.
*
* For Example:
*
* If you create a cell with a width of 100 pixels, and you have a border of
* 10 pixels, then the total width of the cell is 120 pixels.
*
* @todo Implement more borders.
* @param int $pattern Type of border to draw.
* @param int $size Size of border, in pixels.
* @param Zend_Pdf_Color $color Color for the border. Defaults to
* Zend_Pdf_Color_RGB(0,0,0)
*/
public function setBorder($size=1,$pattern=null,$color=null) {}

/**
* Sets the location of where the cell's upper left corner should be placed.
* If the alignment is set, then x and y are offsets to the alignment.
*
* @param int $x X offset for the cell.
* @param int $y Y offset for the cell.
* @param mixed $alignment (Optional) How to align the cell with the X and Y as offsets.
* Defaults to none.
*/
public function setLocation($x, $y, $alignment=Zend_Pdf_Cell::ALIGN_LEFT) {}

/**

```

```
* Draws the cell to the PDF page.  
*  
* This function will parse the internal $_text array and draw the cell to the PDF page.  
*  
* @return void  
*/
```

```
public function write() {}  
}
```

]]></ac:plain-text-body></ac:macro>  
]]></ac:plain-text-body></ac:macro>