Zend_Pdf text drawing improvements - Nico Edtinger

<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 Component Proposal

Proposed Component Name	Zend_Pdf
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Pdf
Proposers	Nico Edtinger
Zend Liaison	TBD
Revision	1.0 - 5 May 2009: Initial Draft. 1.1 - 24 June 2009: Added Docblocks. 1.2 - 24 June 2009: Moved to Ready for Review (wiki revision: 7)

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

It's currently quite complicated to output text with Zend_Pdf, because the positioning and wrapping of every text piece has to be done manually. Instead this proposal introduces a text cursor and a method to add a text block, which enables to Zend_Pdf to handle the basic text flow operations.

2. References

- Zend_Pdf requirments

3. Component Requirements, Constraints, and Acceptance Criteria

Most requirements take the form of "foo will do ...." or "foo will not support ...", although different words and sentence structure might be used. Adding functionality to your proposal is requirements creep (bad), unless listed below. Discuss major changes with your team first, and then open a "feature improvement" issue against this component.

- This proposal **will not** reengineer Zend_Pdf.
- This proposal **will** instead add some new basic methods, that extend the current functionality.
- This proposal **will not** hinder reengineering Zend_Pdf. Everything that gets added is an API improvement, designed from a component user point of view.
- The new methods **will** be optional.
- Therefore, Zend_Pdf **will** stay completely backward compatible.

4. Dependencies on Other Framework Components

- Zend_Pdf

5. Theory of Operation

The basic method to add text is still drawText(), but the parameters $x and $y will be optional, if a text cursor is set. The cursor runs from left to right or right to left. Every time a new text fragment is drawn the cursor advances automatically. If a width is set, the cursor will automatically add a new text line wrapping words as needed. It's possible to change the text style between text fragments.
In addition a new method drawTextBlock() is added that allows drawing text left or right aligned, centered or justified. The later two alignments are only possible with this method, but the whole text block can only have one text style.

6. Milestones / Tasks

- Milestone 1: design notes will be published here [DONE]
- Milestone 2: Published working prototype.
- Milestone 3: Working prototype checked into the incubator.
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

7. Class Index

- Zend_Pdf
- Zend_Pdf_Page

8. Use Cases

UC-01

```
// write text using a cursor
$page = new Zend_Pdf_Page();
$page->setTextCursor(200, 200);
$page->setTextCursorWidth(100);
$page->drawText('This is a very long text block ');
$page->drawText('to test the ');
$page->setFillColor(new Zend_Pdf_Color_Html('#5d5e60'));
$page->drawText('automatic line breaks ');
$page->setFillColor(new Zend_Pdf_Color_Html('#bd1d35'));
$page->drawText('if the text is to long to fit a line.')
```

UC-02

```
// output a random text block using alignment justified
$page = new Zend_Pdf_Page();
$text = '';
for ($i = 0; $i < 500; ++$i) {
    $length = rand(4, 10);
    for ($j = 0; $j < $length; ++$j) {
      $text .= rand('a', 'z');
    }
    $text .= ' ';
}
$page->drawTextBlock($text, 200, 200, 90, null, Zend_Pdf_Page::ALIGN_JUSTIFY);
```

9. Class Skeletons

```
class Zend_Pdf_Page {
    /**
     * Draw a line of text at the specified position or current cursor position.
     *
     * @see Zend_Pdf_Page::drawPage()
     * @param string $text
     * @param float|null $x (optional) if null cursor position is taken and advanced afterwards
     * @param float|null $y (optional) if null cursor position is taken and advanced if text wrapped
     * @param string $charEncoding (optional) Character encoding of source text.
     *   Defaults to current locale.
     * @param bool $ignore_width (optional) draw text, but don't wrap
     * @throws Zend_Pdf_Exception
     * @return Zend_Pdf_Page
     */
    public function drawText($text, $x = null, $y = null) {
        // ...
    }

    /**
     * Set position of text cursor.
     *
     * @param float $x x position
     * @param float $y y position
     * @return Zend_Pdf_Page fluid interface
     */
    public function setTextCursor($x, $y) {
        // ...
    }

    /**
```

```
     * Set width of block used for cursor. If the width is reached a newline is started.
     * Wrapping happens at whitespace.
     *
     * @param float|null $width block width. null to deactivate
     * @return Zend_Pdf_Page fluid interface
     */
    public function setTextCursorWidth($width) {
        // ...
    }

    /**
     * Set if cursor moves from left to right or from right to left
     *
     * @param string $direction self::CURSOR_DIRECTION_LEFT or self::CURSOR_DIRECTION_RIGHT
     * @return Zend_Pdf_Page fluid interface
     */
    public function setTextCursorDirection($direction) {
        // ...
    }

    /**
     * Start a newline. The x position is reset and line height is added to the y position
     *
     * @return Zend_Pdf_Page fluid interface
     */
    public function textCursorNewline() {
        // ...
    }

    /**
     * Move text cursor in relation to current position
     *
     * @param float $x_offset x offset
     * @param float $y_offset y offset
     * @return Zend_Pdf_Page fluid interface
     */
    public function textCursorMove($x_offset = null, $y_offset = null) {
    }

    /**
     * Draw a text in a block with a fixed width and an optional fixed height. The text can be left or
right
     * aligned, centered of justified. If height is given, but the text would be longer an exception
is thrown.
     *
     * This method may also be called as drawTextBlock($text, $width). The block is then drawn to the
current
     * cursor position.
     *
     * @param string $text
     * @param float $x x position
     * @param float $y y position
     * @param float $width block width
     * @param float|null $height optional height to check for
     * @param string $align one of: self::ALIGN_LEFT, self::ALIGN_RIGHT, self::ALIGN_CENTER,
self::ALIGN_JUSTIFY
     * @throws Zend_Pdf_Exception
     * @return Zend_Pdf_Page fluid interface
     */
    public function drawTextBlock($text, $x, $y, $width, $height = null, $alignment =
self::ALIGN_LEFT) {
```

```
        // ...
    }
}
```

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