Redmine - Defect #8850

Change PDF column width

2011-07-19 16:20 - Paps 1

Status:	New	**Start date:**	2011-07-19
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	PDF export	**Estimated time:**	0.00 hour
Target version:			
Resolution:		**Affected version:**	1.2.1

Description

Hi,

We have a problem with the pdf export for some custom queries.

The column width is not optimal for the "N° Prime" and "Fournisseur" fields.

For some fields, the width are too big for the information present in to it and too small for others.

How to change the width of the column to automatically adapt to the content in to it?

Thanks

History

#1 - 2025-02-04 16:59 - popy popy

Better later to the party than never :-)
I have implemented a rudimentary html tr, td, th parser in pdf.rb which calculates/set's the widths of each cell depending on the max text length of each column.
Also colspan & rowspan are supported.
It works way better than the original code, which just uses the same width for every column.
Yeah, it's not perfect! It should be implemented in rbpdf directly, but that's beyond my knowledge of ruby :-)

Here is the implementation for lib/redmine/export/pdf.rb, tested on redmine 5.0.5.stable.
Replace the function "RDMwriteFormattedCell" with my code and add the new function "parse_and_calculate_column_widths" below it.

```
        def RDMwriteFormattedCell(w, h=0, x='', y='', txt='', attachments=[], border=0, ln=1, fill=0, reseth=
true, align='', autopadding=true)
          require 'nokogiri'
          @attachments = attachments

          css_tag = ' <style>
          table, td {
                  border: 1px #ff0000 solid;
                  background-color: #fafafa;
          }
          th {  background-color:#EEEEEE; padding: 4px; white-space:nowrap; text-align: center;  font-style: b
old;}
          pre {
                  background-color: #fafafa;
          }
          h2, h3, h4, h5, h6 {
                  color:#333333;
          }
          </style>'

          # Remove {{toc}} tags
          txt = txt.gsub(/<p>\{\{(.*?)toc\}\}<\/p>/i, '')

          doc = Nokogiri::HTML.fragment(txt)
          tables = doc.css('table')

          #logger.info "Tables found: #{tables.size}"

          tables.each do |table|
```

```
            # Calculate column widths and insert them into the HTML
            table_new_html = parse_and_calculate_column_widths(table)

            # Replace the original table element with the modified one
            table.replace(Nokogiri::HTML.fragment(table_new_html).at('table'))
          end

          #logger.info "Parsed HTML: #{doc.to_html}"

          writeHTMLCell(w, h, x, y, css_tag + doc.to_html, border, ln, fill, reseth, align, autopadding)
        end

        def parse_and_calculate_column_widths(table)
          rows = table.css('tr')
          column_widths = []
          total_columns = 0

          # Count the maximum number of columns and calculate text lengths
          rows.each do |row|
            columns = row.css('td', 'th')
            column_index = 0

            columns.each do |col|
              colspan = col['colspan'].to_i
              colspan = 1 if colspan == 0 # If no colspan or 0, use 1
              text_lines = col.text.split(/\r?\n|\r|\n/)
              max_text_length = 1
              max_text_length = text_lines.map { |line| line.strip.length }.max + 3 if text_lines.length > 0

              # Calculate maximum width for each column
              colspan.times do |i|
                column_widths[column_index + i] ||= 0
                column_widths[column_index + i] = [column_widths[column_index + i], max_text_length].max if
colspan == 1
              end

              column_index += colspan
              total_columns = [total_columns, column_index].max
            end
          end

          # Apply square root normalization
          normalized_widths = column_widths.map { |w| Math.sqrt(w) }
          total_width = normalized_widths.sum.to_f
          column_widths = normalized_widths.map { |width| (width / total_width * 100).round(2) }


#logger.info "Calculated normalized column widths: #{column_widths.inspect} - columns: #{column_widths.length}
"

          # Create rowspan array
          rowspan_current = Array.new(column_widths.length, 0)
          rowspan_colspan = Array.new(column_widths.length, 1)
          #logger.info "Rowspan init: #{rowspan_current.inspect}"

          # Calculate & set cell widths, also take rowspan & colspan into account
          rows.each do |row|
            columns = row.css('td', 'th')
            column_index = 0

            columns.each do |col|
              colspan = col['colspan'].to_i
              colspan = 1 if colspan == 0
              rowspan = col['rowspan'].to_i
              rowspan = 1 if rowspan == 0

              if (rowspan > 1)
                rowspan_current[column_index] = rowspan - 1
                rowspan_colspan[column_index] = colspan
              else
                if (rowspan_current[column_index] > 0)
                  rowspan_current.drop(column_index).each do |rowspan_col|
                    if (rowspan_col > 0)
                      rowspan_current[column_index] -= 1 if (rowspan_current[column_index] > 0)
                      column_index += rowspan_colspan[column_index]
```

```
                end
              end
            end
          end

          width = 0
          colspan.times do |i|
            width += column_widths[column_index]
            column_index += 1
          end

          col.set_attribute('style', "width: #{width - 0.12}%")
        end
      end

      table.to_html
    end
```

Files

column_width.pdf	317 KB	2011-07-19	Paps 1