Extending OpenOffice XHTML filter

Habib Louafi*and Stéphane Coulombe†

Department of IT and Software Engineering
École de Technologies Supérieure, University of Quebec, Canada

May 7, 2013

OpenOffice enterprise documents can be converted into XHTML-based Web pages. The components (e.g. text, images, and shapes) of the enterprise document are extracted and converted separately, and then wrapped in a XHTML document.

However, the XHTML-based Web page, which outputs only text, is very rudimentary and not reliable at all. It doesn't take into account the font issues; all the fonts used in the presentation are replaced by only one font (the Web browser's default font). The images, graphics and even the background are completely absent in the outputted Web pages, though, in the XTML source code, images' binary codes are still included. Besides, the layout aspect is not taken into account; all the objects (text boxes, images, etc) are overlaid on top of each other.

Therefore, we have extended the native OpenOffice XHTML filter by fixing the aforementioned drawbacks and adding some features that were not considered at all; such as the graphics. Note that, only the features which are important to our research are fixed. Figures 1, 2 and 3 depict the impact of the extensions we have achieved. The first figure shows a presentation slide containing text boxes, images and some graphics. The second figure shows the Web page as produced by the native OpenOffice XHTML filter, whereas the third one shows the Web page as produced by the extended version of the OpenOffice XHTML filter.

The following sections will be about the various identified bugs and how they were fixed as well as the added features.

1 Styles issues in the native XHTML filter

The following is tested using OpenOffice 3.1 version. The XHTML filters of the 3.1 and 2.4 versions have the same behavior, as described in the previous section.

1.1 How the styles are represented in the presentation document

After analyzing how the styling information, used in the presentation document, is represented in the back-end (XML content), we have realized that not all this information is included in the XML content. In fact, the styles used in the presentation are of two types:

1. *The default styles*: When the user writes a text in the presentation without modifying any styles, the default ones are used. In other words, the styling information is not specified by the user, but by the OpenOffice engine. On the back-end side, the styles' information is not included at all in the XML content "content.xml". In fact, they are included in the "styles.xml" file and referenced in the XML content of the presentation.

*Email: hlouafi@gmail.com, habib.louafi.1@ens.etsmtl.ca
†Email: coulombe.stephane@etsmtl.ca

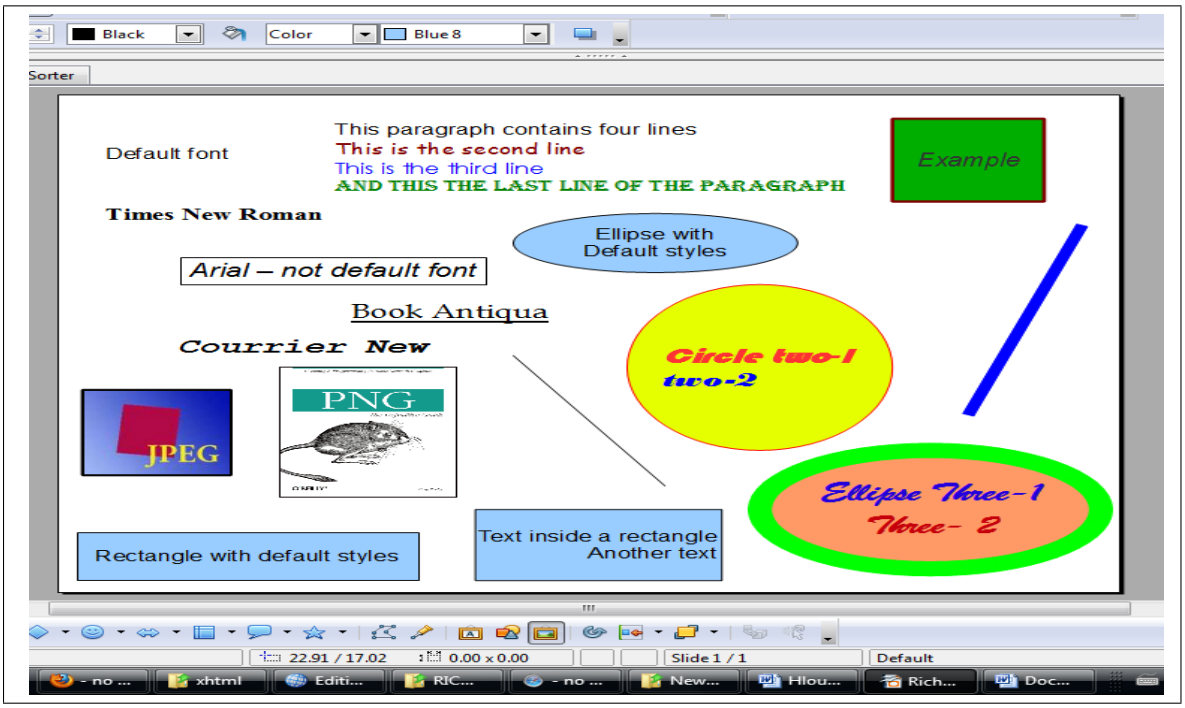

Figure 1: Example of a slide containing different objects.

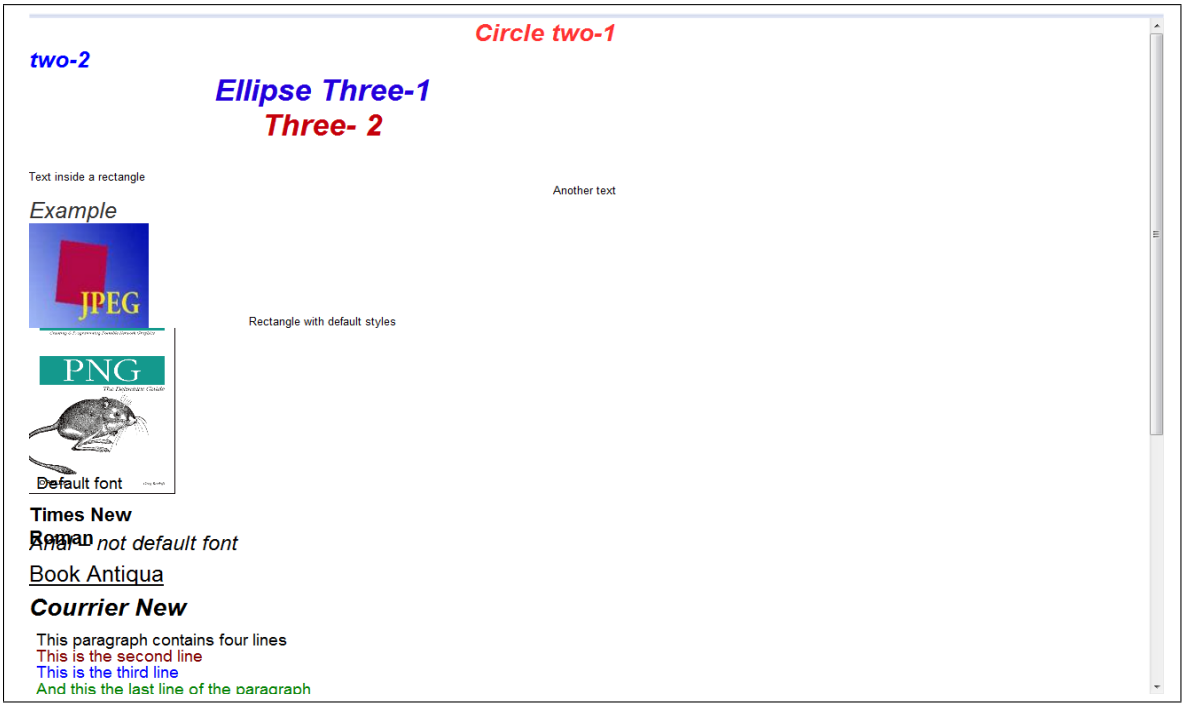

Figure 2: The Web page version of the previous slide as exported by the native OpenOffice XHTML filter.

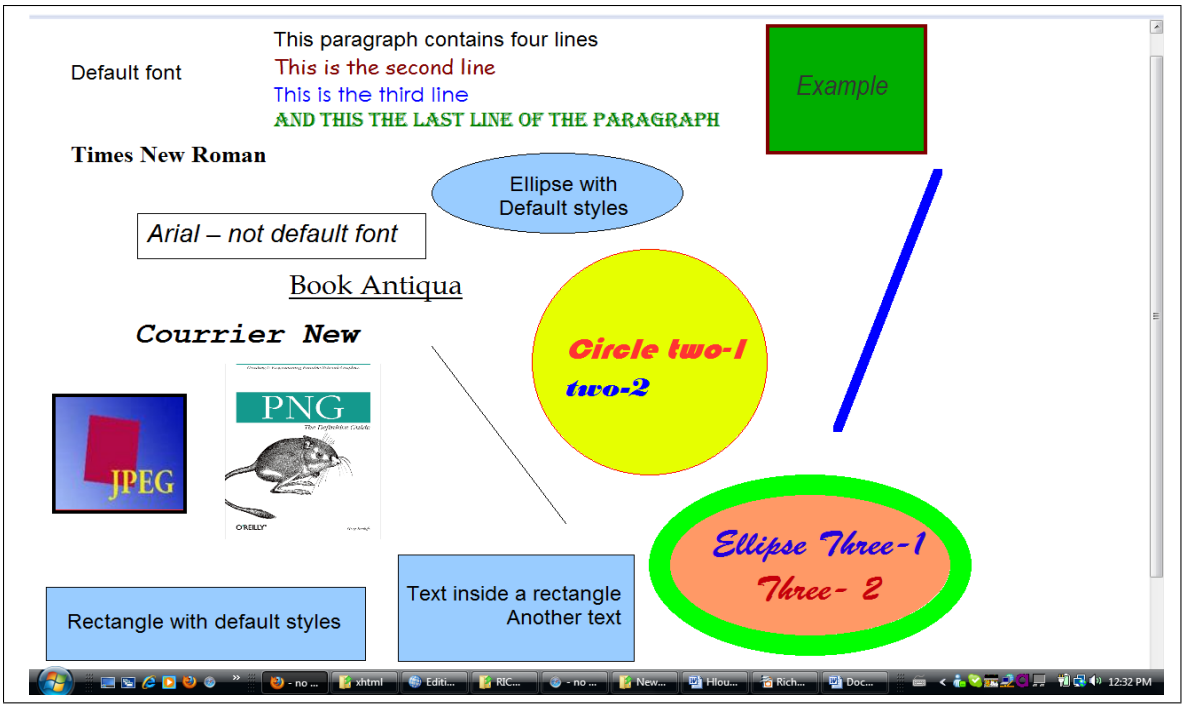

Figure 3: The Web page version of the previous slide as exported by the extended OpenOffice XHTML filter.

2. *The user-specified styles*: In this case, the typed text is re-styled by the user, such as modifying the font-name, font-size, font-color, etc. In other words, the styles' information is explicitly specified by the user. In this case, this information is included in the XML content of the presentation

1.2 How the styles are converted by the native XHTML filter

The styles used in the presentation are first collected by the "style_collector.xsl" template. Then, they are converted to CSS styles using the "style_mapping_css.xsl" template. After analyzing how the fonts are collected and mapped to their corresponding CSS styles, we have realized the following:

1. *The default styles*: We said that the default styles used in the presentation are not explicitly written in the XML content of the presentation, but in the styles XML file. The filter is able to convert all the styles information, except the font name. In the version of OpenOffice we have installed, the default font name was "Arial". This information is lost during the export operation; it is replaced by "Times New Roman". The latter is no more than the default font name of the Web browser. Since no font name is specified by the filter in the XHTML output, the Web browser uses its default font name. But, the good thing is that all the other styles (color, italic, underline, etc) are taken into account by the filter.

2. *The user-defined styles*: When the styles are specified explicitly by the user, we have noticed the same phenomena; the font's names used in the presentation are not reproduced by the filter.

The problem is in the "style_mapping_css.xsl" template that doesn't perform the mapping operation correctly. It uses a syntax that is different from that used by OpenOffice in the "content.xml" and "styles.xml" files. After fixing this bug, all the font's names used in the presentation (the default one and those specified by the user) are reproduced with fidelity as illustrated by Figures 4, 5, and 6.

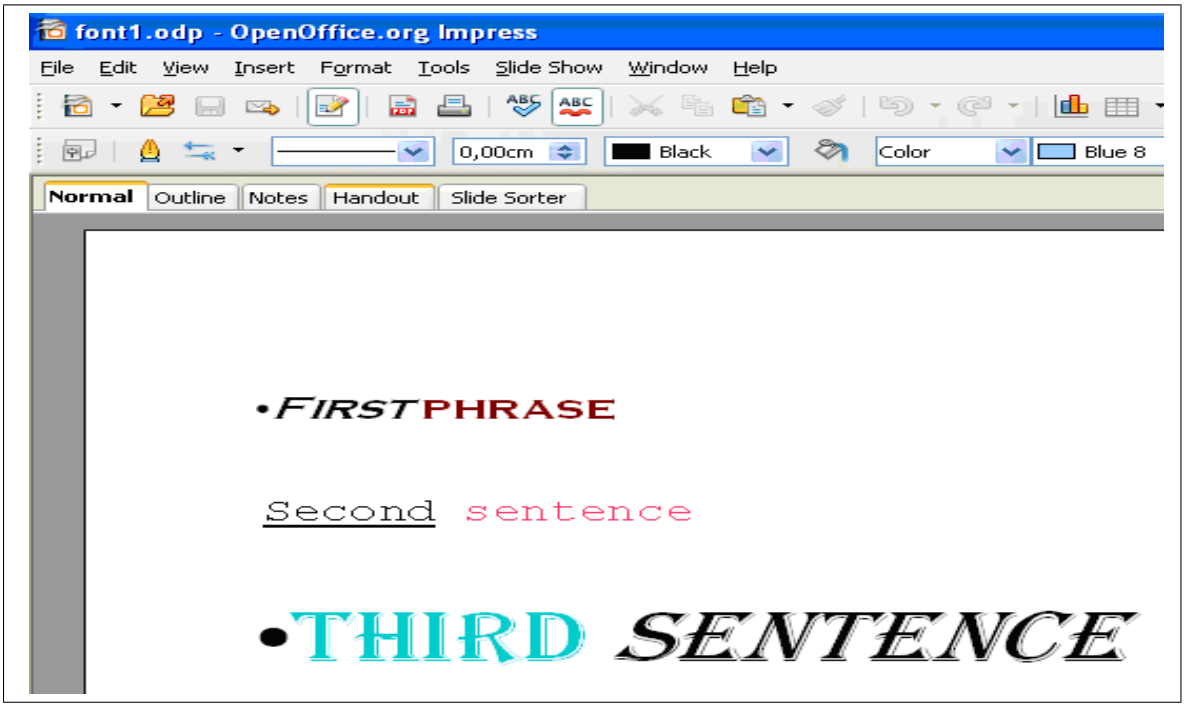

Figure 4: A slide that contains different fonts.

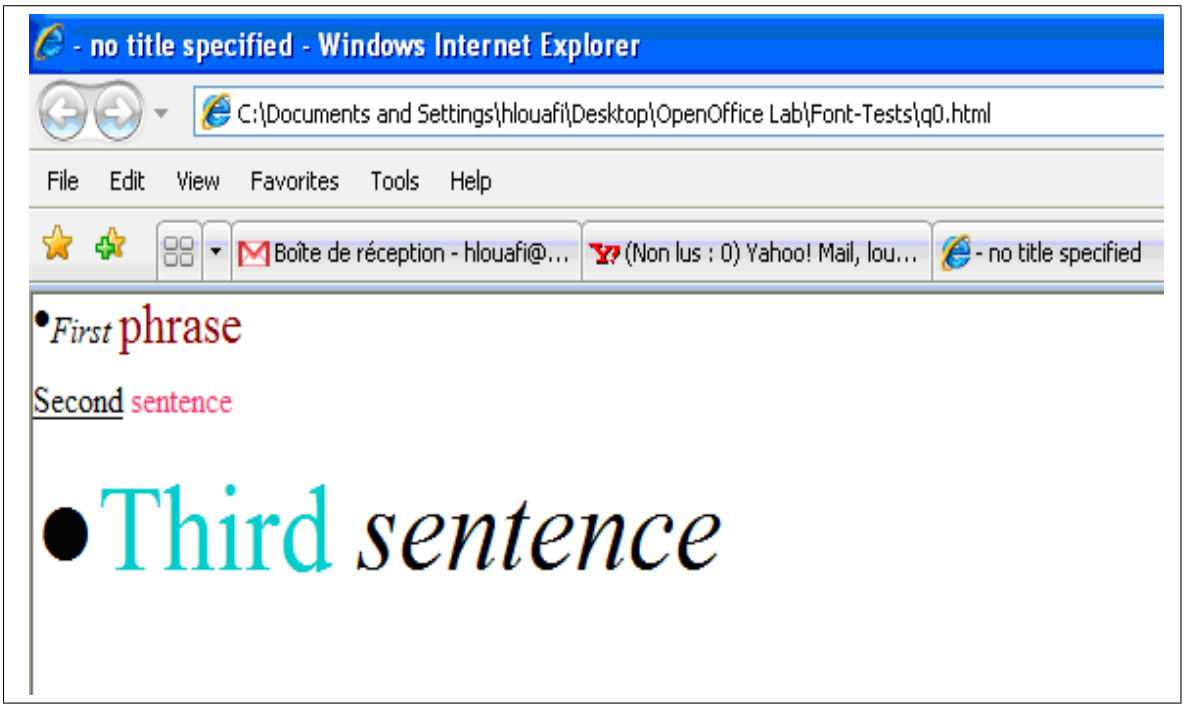

Figure 5: The XHTML Web page version of the previous slide as exported by the native OpenOffice XHTML filter.

Figure 6: The XHTML Web page version of the previous slide as exported by the extended OpenOffice XHTML filter.

2 Layout issues

Figure 7 shows a slide comprised of seven text boxes. One of them is situated opposite to the others on the right side. The native filter is not able to preserve this layout as shown by Figure 8; all is aligned to the left. The problem is that the filter doesn't convert the coordinates of each text box. That's why they are serialized in the XHTML output. In OpenOffice, these coordinates are referred by x and y; which represent the distances of the text box from the left and top of the slide respectively. They should be converted to their corresponding CSS left and top positions. Figure 9 shows the XHTML output after fixing this issue. The modifications have been done in the template: "..\OpenOffice.org 3\Basis\share\xslt\export\xhtml\body.xsl"

3 Images issues

The OpenOffice native XHTML filter converts images into binary codes and incorporates them in the XHTML code. It doesn't use the traditional technique that consists in putting all the images in a folder and including URL references to them in the XHTML code. Figure 11 shows the XHTML source code of the example shown by Figure 10. Albeit, the embedded images are binary coded, they can be rendered by the majority of the Web browsers, such as Firefox, Safari, etc. The problem is with the MS-IE, which is not able to render binary coded images.

In fact, the images are coded using the base-64 encoding [1]. This way of coding images has a negative impact on the file size of the presentation. On average, an image encoded using base-64 is 33% larger than a binary image [2]. Besides, there are pros and cons to using this kind of representation; file size and bandwidth versus network latency.

Two options are offered when the image is to be inserted in the presentation. When the user is asked to enter or select the file name of the image to insert, a check box, named *Link*, can be checked to indicate whether the physical URL of the image should be inserted or not.

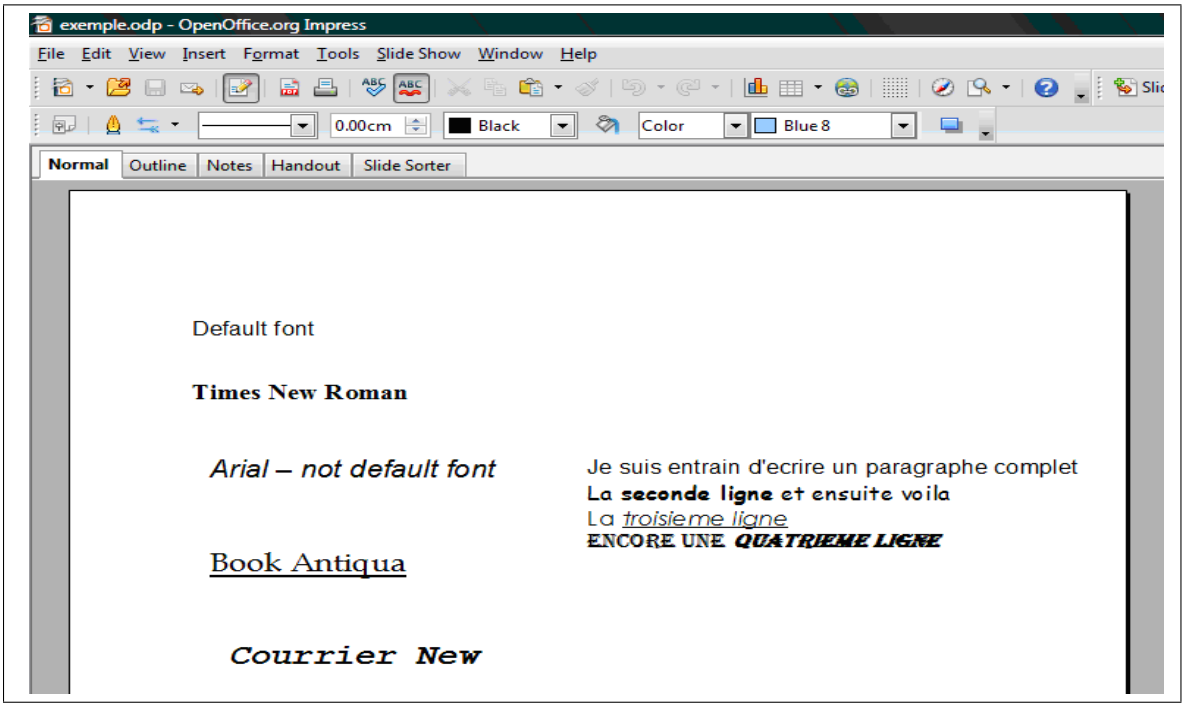

Figure 7: A slide that contains different textboxes dispersed.

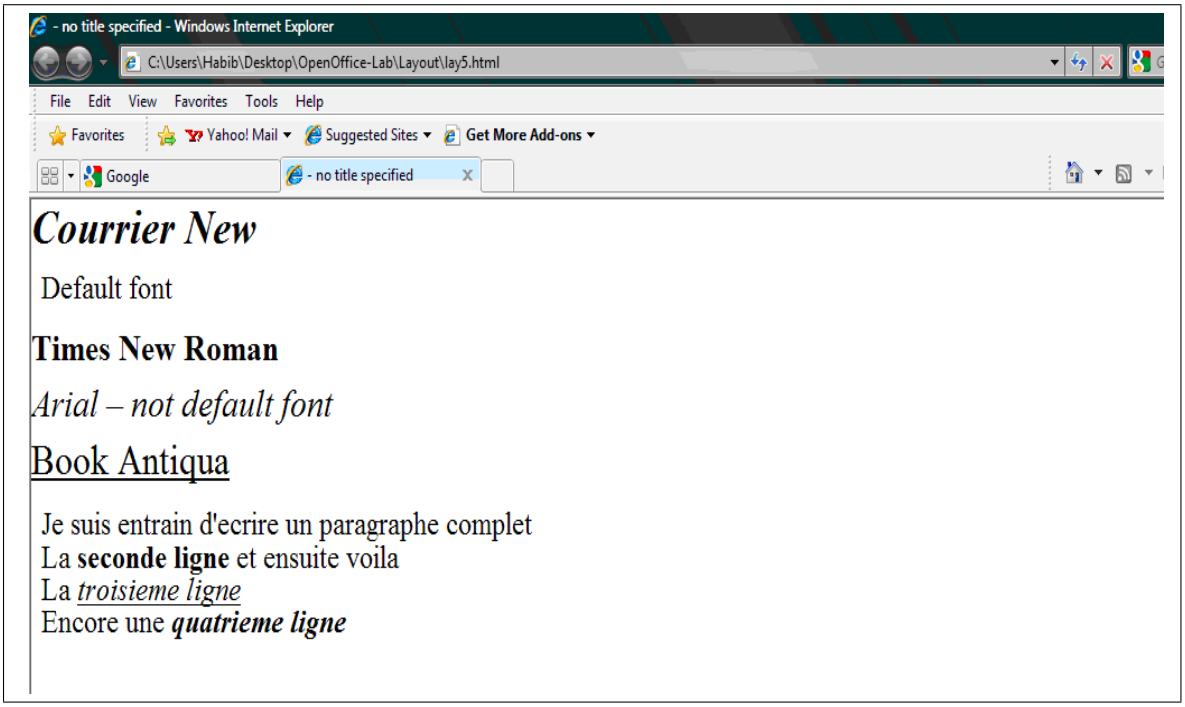

Figure 8: The XHTML Web page version of the previous slide as exported by the native OpenOffice XHTML filter.

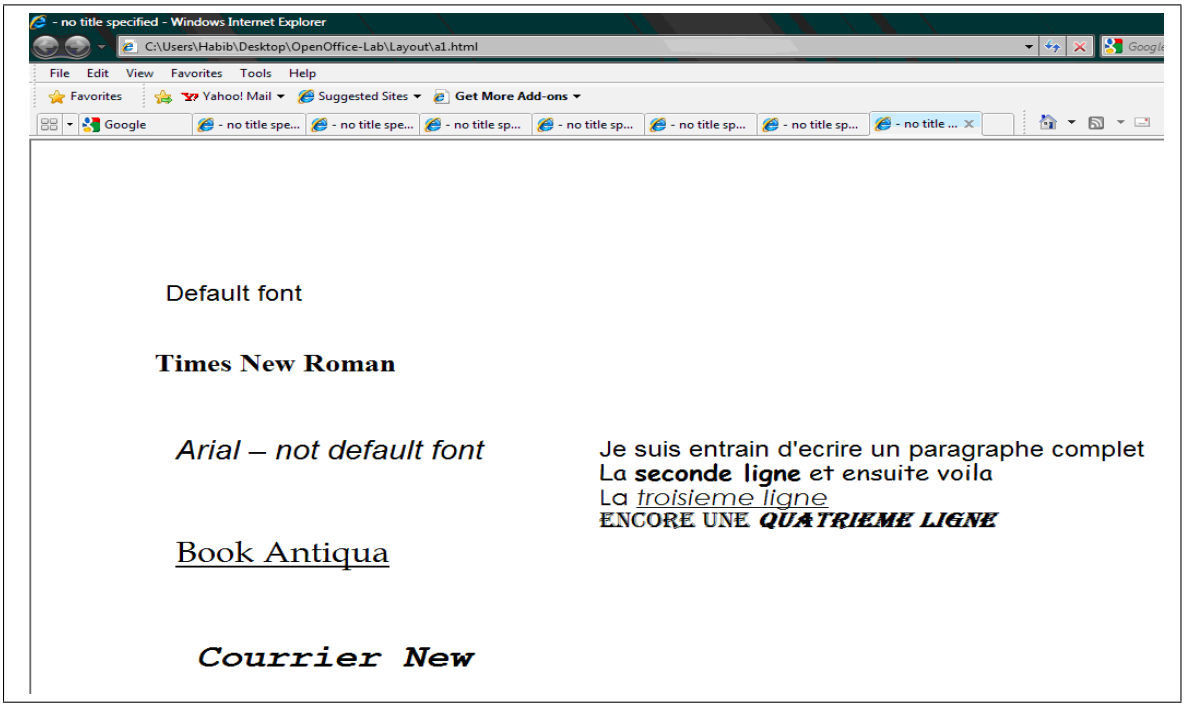

Figure 9: The XHTML Web page version of the previous slide as exported by the extended OpenOffice XHTML filter.

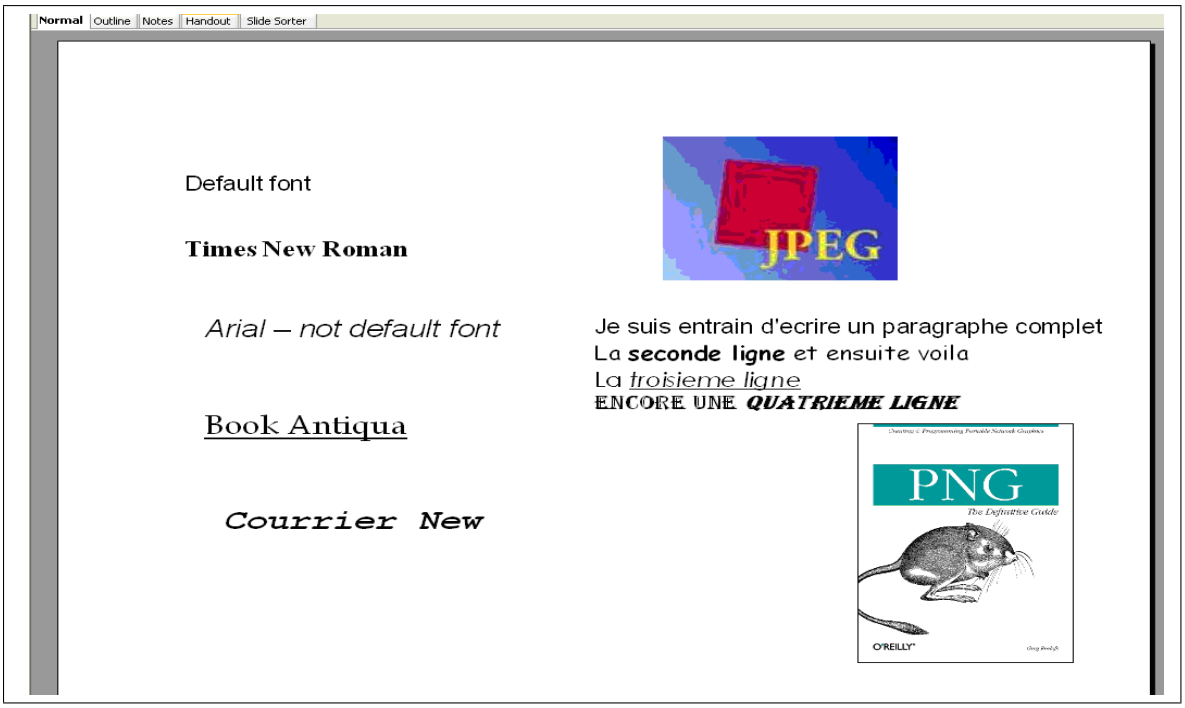

Figure 10: A slide that contains different text boxes and images.

Figure 11: The source code of the XHTML Web page version of the previous slide as exported by the native OpenOffice XHTML filter (The images are binary coded).

1. If *Link* is not checked, the image is converted into base-64 and included in the content XML file. In the latter, a relative link to PNG files is used as a URL to the image.

2. If *Link* is checked, the image is not converted and an absolute URL link is inserted in the XML content file. That URL represents the physical location, in the disk, of the inserted image. This way, the URL and type of the image (e.g., JPEG, GIF,...) are preserved.

Using the second option, it is possible to use the traditional way of representing images in Web pages. This way, it is possible to insert the URL of the image in the XHTML code and adding an extra folder; which includes the embedded images.

4 Graphics issues

In order to enhance the existing OpenOffice XHTML filter, we have added the possibility to export the following graphics: rectangles, circles, ellipse and lines. To this end, we have added a Javascript library that enables us to draw different shapes on the Web browser. This library is free of use and open source code [3]. Figure 12 shows an example of a slide containing rectangles, an ellipse, a circle and two lines. The existing filter doesn't output any graphics. As illustrated by Figure 13, the modifications added to the filter are able to render those shapes on Web pages. The modifications have been achieved as follows:

1. The "..\OpenOffice.org 3\Basis\share\xslt\common\measure_conversion.xsl" is included in the template:
"..\OpenOffice.org 3\Basis\share\xslt\export\common\styles\style_mapping_css.xsl".

2. The templates used to capture the styles used by the shapes; such as the color of the line and the background color are added to the template:
"..\OpenOffice.org 3\Basis\share\xslt\export\common\styles\style_mapping_css.xsl".

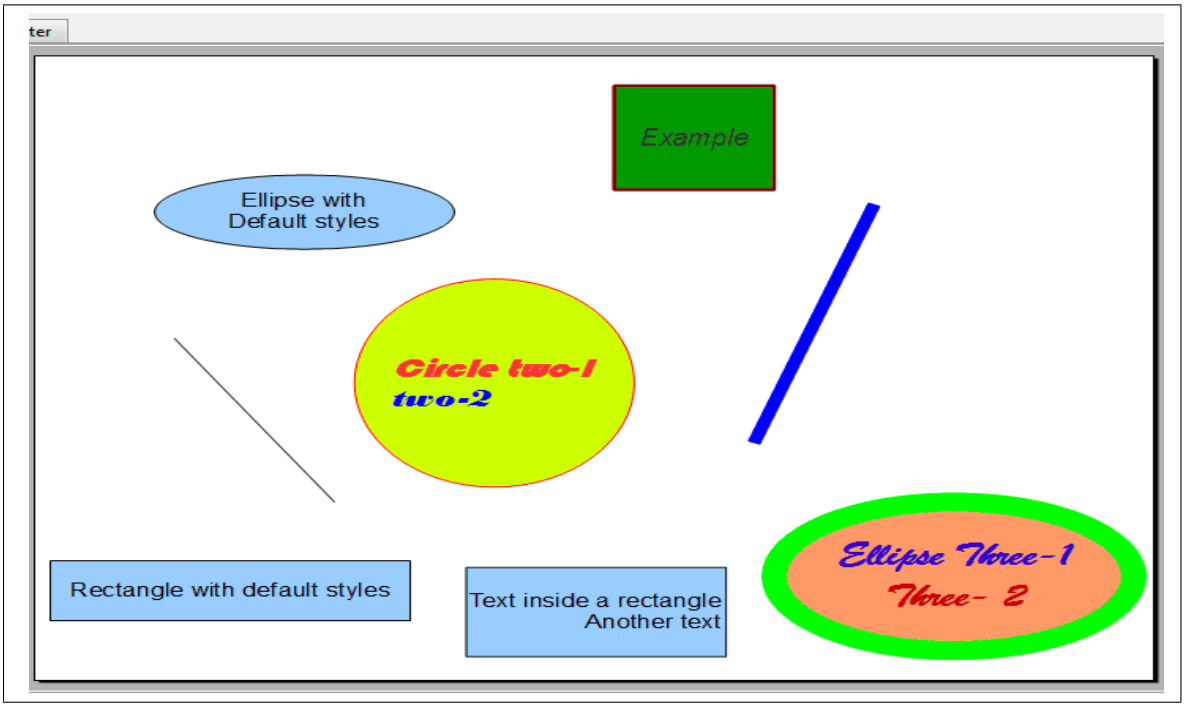

Figure 12: A slide that contains different shapes.

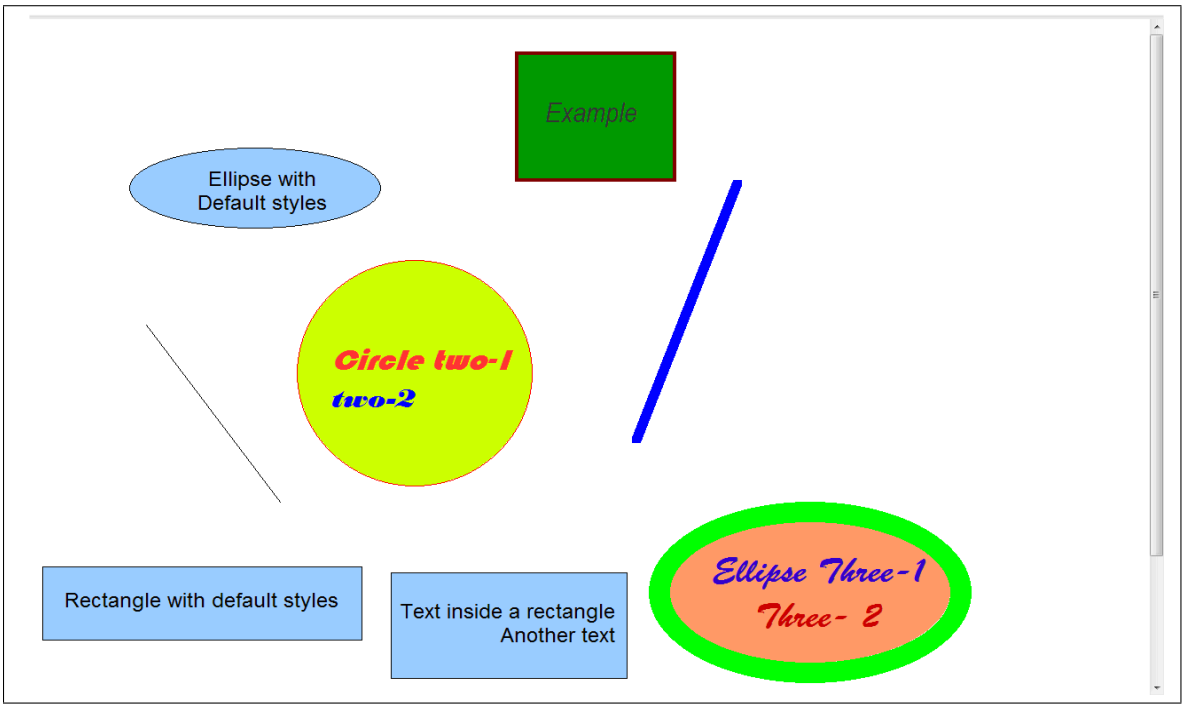

Figure 13: The XHTML Web page version of the previous slide as exported by the actual OpenOffice XHTML filter.

3. The templates used to draw the rectangle, ellipse, circle and line shapes are added to the template: "..\OpenOffice.org 3\Basis\share\xslt\export\xhtml\body.xsl".

4. In the XSLT source code, these templates can be located by searching their corresponding keyword. For example, to locate the template that draw the ellipse, you can use the "draw:ellipse" keyword.

5 Text-boxes and images adaptation

This section is about conveting enterprise documents (Impres slides, for example) to be visualized on Web-enabled mobile terminal, which ahev not the same screen resolution. It is part of our research project.

The extended version of the XHTML filter allows us to adapt presentation slides into XHTML web pages comprised of text-boxes and images. The next step is to give this filter the possibility to adapt the presentation document components using a scaling parameter (which can be applied to both text boxes and images) and a quality factor (applied to embedded JPEG images).

To allow the XHTML filter resizing text boxes using a scaling parameter z, we modified the template: "..\OpenOffice.org 3\Basis\share\xslt\export\xhtml\body.xsl". For embedded JPEG images, we developed a Java-based application that uses imageMagick tools to convert images using z and QF. Then we registered this application with OpenOffice by providing the classpath in which the application is stored on the disk. As a result, using OpenOffice APIs, it is possible now to adapt presentation document, using the desired scaling parameter and quality factor, into Web pages renderable on Web-enabled mobile devices. Figure 14 shows a slide as exported by the OpenOffice XHTML extended version using different combinations of z and QF.

6 Conclusion

The extensions have been achieved in two steps. In the first one, we fixed the various bugs mentioned above and added the possibility to export images and shapes (other shapes can be added following the same logic we used). In the second step, we added the possibility to export the document elements (text, iamges and shapes) using a scaling parameter z, which is applied to all document's components, and a quality factor QF applied to JPEG images (this is used to control the resulting image visual quality and file size).

The extended version of the XHTML filter has been tested using OpenOffice Impress slides, and future work can be carried out to validate its applicability to the other types of documents, such as Writer and calc.

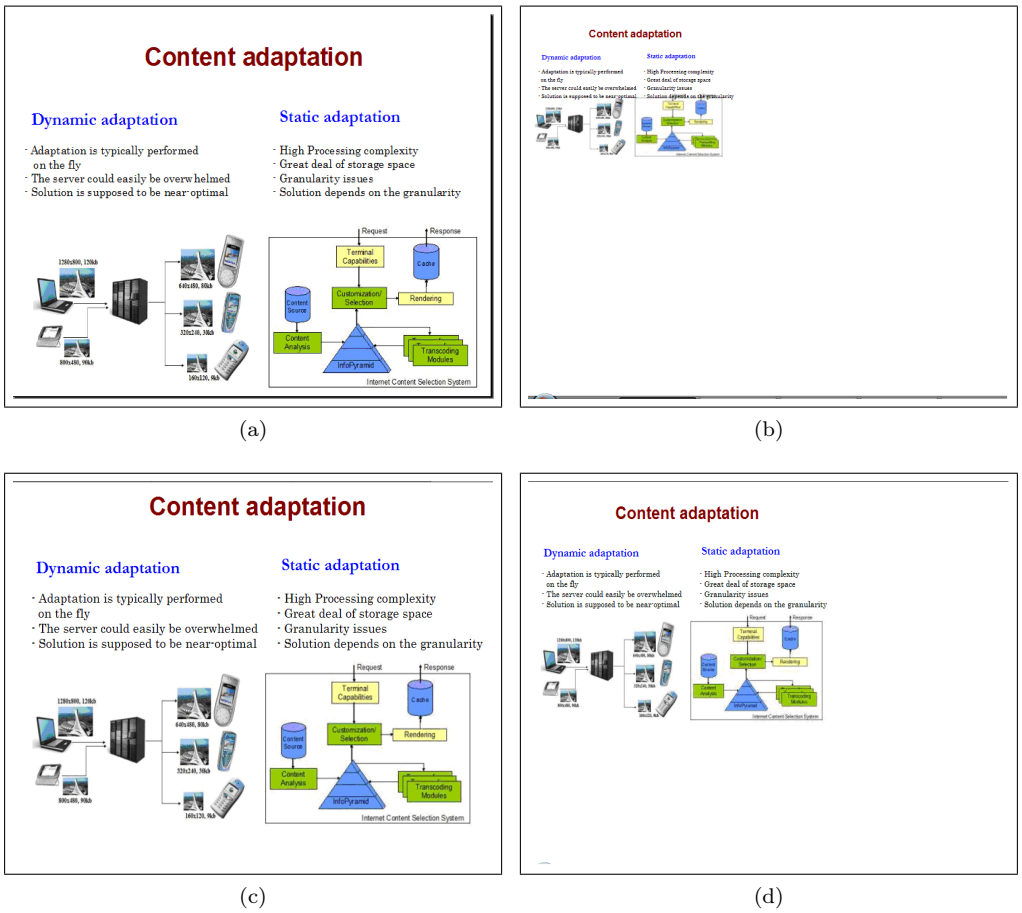

Figure 14: A slide as exported by our extended OpenOffice XHTML filter: (a) the original slide, (b) transcoded using $z = 30\%$ and $QF = 80$, (c) transcoded using $z = 80\%$ and $QF = 80$, (d) transcoded using $z = 50\%$, $QF = 60$

References

[1] M. S. Kolich. High Performance JavaScript Vector Graphics Library. On line, 2009. Accessed on 04 April 2013.

[2] Wikipedia. Data URI scheme. On line, 2013. Accessed on 04 April 2013.

[3] W. Zorn. High Performance JavaScript Vector Graphics Library. On line, 2008. Accessed on 04 April 2013.