Transforming Open XML WordprocessingML to XHTML Using the Open XML SDK 2.0

# Benefits to Converting WordprocessingML to XHTML

Converting WordprocessingML documents to XHTML has several interesting uses.

* You may want to provide a simple preview of Open XML WordprocessingML documents in your application. By converting to XHTML, you can use one of many HTML viewers to integrate this functionality.
* Another interesting application is interoperating with other software systems where you can use HTML to achieve rich functionality. For example, you could populate a SharePoint wiki from a set of word-processing documents.
* For developers, a transform to XHTML is especially interesting. If you need a simple query of a word-processing document, perhaps selecting paragraphs by content or style name, it is easier to write the query for XHTML than it is for WordprocessingML.

The converter presented in this article has some limitations. It does not transform formatting such as styles and fonts. Instead it transforms the actual textual content and images. Although the example does not transform formatting from the source document, you can supply a cascading style sheet that defines appropriate classes for paragraph styles. The cascading style sheet can then supply formatting at a paragraph level.

# Using the HtmlConverter Class

To download the HtmlConverter class and examples, see [PowerTools for Open XML](http://www.codeplex.com/powertools), click the Downloads tab, and download HtmlConverter.zip.

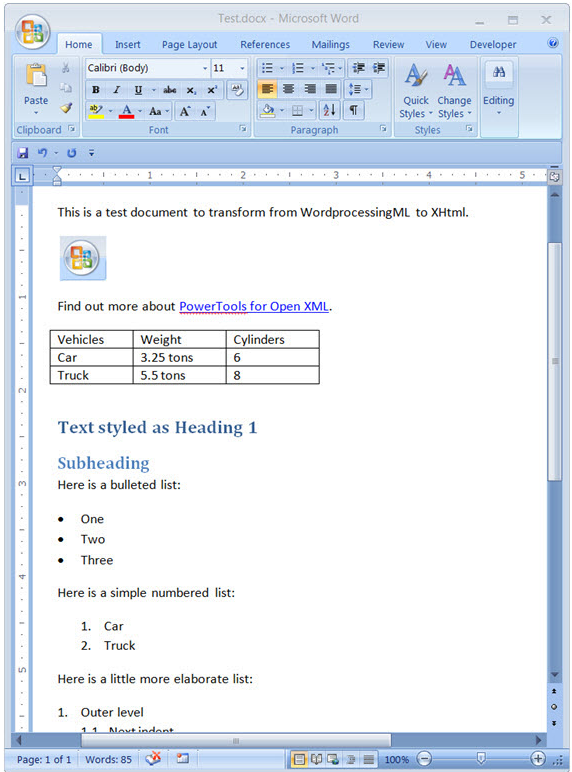
To build HtmlConverter, you must also download and install the [Open XML SDK 2.0 for Microsoft Office](http://www.microsoft.com/downloads/details.aspx?FamilyId=C6E744E5-36E9-45F5-8D8C-331DF206E0D0).

If you are setting up your own project to include the HtmlConverter class, you need to add references to the following four assemblies:

* DocumentFormat.OpenXml
* OpenXmlPowerTools
* System.Drawing
* Windows.Base

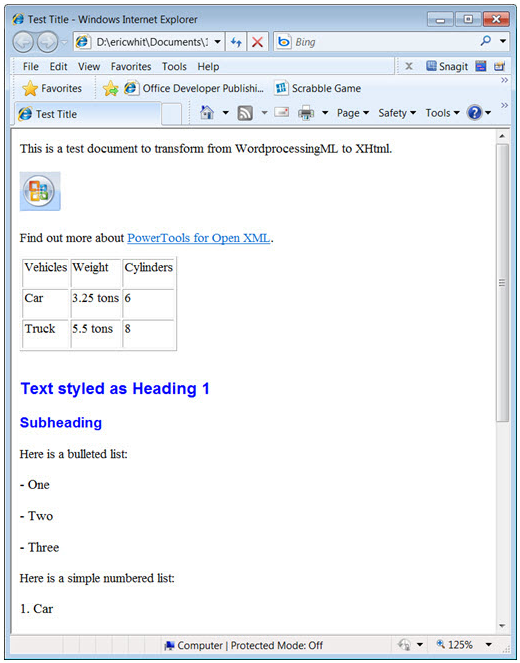
Figure 1 shows a WordprocessingML document that I used as a test source.

**Figure 1. Test source WordprocessingML document**



When transformed to XHTML, the document looks like Figure 2.

**Figure 2. Rendered XHTML**



This first version of this example has the following goals.

* **Accurately transform the text of the document.** It transforms the text of each paragraph to an appropriate XHTML element—either a paragraph element (x:p) or an appropriate heading element (such as x:h1 or x:h2). The textual transform is robust. It accurately transforms the text if the WordprocessingML contains revision tracking, content controls, or various annotations such as fields.
* **Does not convert formatting, such as paragraph fonts, or character fonts.** However, the HtmlConverter class optionally adds a class attribute (derived from the paragraph style name) for each paragraph or heading element. You can then supply a cascading style sheet that applies basic formatting at the paragraph level.
* **Transform WordprocessingML tables to simple XHTML tables.** The transform is recursive. It handles tables that are embedded in cells in other tables.
* **Transform hyperlinks appropriately.** This is somewhat more complex than first seems, because there are two forms of markup for hyperlinks.
* **Transform numbered lists and bulleted lists.** The article [Working with Numbered Lists in Open XML WordprocessingML](http://msdn.microsoft.com/en-us/library/ee922775.aspx) explains the semantics of the markup for numbered and bulleted lists. The code in this article includes a new class, the ListItemRetriever class, which retrieves the text of the list item for any paragraph.
* **Correctly transform images.** This issue is somewhat more difficult because images in WordprocessingML are embedded in the package (the .zip file), but in XHTML, images have their own URL. You must write images to separate files on the disk, or upload them to a server depending on how you want to use the XHTML. The browser that displays the XHTML needs access to those images. The HtmlConverter class enables you to supply an event handler that takes an image as an argument and returns the markup to insert in the XHTML. In that event handler, you can upload or save the image as appropriate, and then return an XHTML IMG element that contains the appropriate URI or URL that points to the image.

The most important goal of the HtmlConverter class is to give you source code and guidance on how to perform this kind of transformation and to provide a flexible starting point so that you can code your own conversion to HTML that meets your goals.