HTML Export

The are many ways of writing HTML and it is impossible to please everyone. The exporter has been extensively rewritten several times in order to provide HTML code as clean as possible and there are a number of options, described below, controlling document export.
In addition to the XHTML exporter there is a `Multipart HTML' exporter which converts the XHTML output (possibly consisting of several files, i.e., images, style sheet) into a single MIME-encoded file (traditionally with the extension .mht). Some web browsers save web pages in this format.

Contents

Options

Export as HTML 4.01
Export with PHP instructions
Declare as XML
Allow extra markup in AWML namespace
Embed (CSS) style sheet
Embed images in URLs (Base64-encoded)

Export as HTML 4.01

AbiWord's HTML exporter has always given the user a choice between HTML and XHTML, the latter being XML and thus easier to handle (and ideally to re-import, although this has been an elusive goal). Some people still prefer the traditional HTML, since some browsers don't understand that XHTML is HTML and not just XML (which can almost anything), but increasingly XHTML is preferred and best-understood.
If exporting as HTML then the document will not be declared as XML and markup in the AWML namespace will not be possible, since these options are specific to XHTML.

Export with PHP instructions

This option is seldom (if ever) used but may be of interest. Many websites these days use PHP to give webpages a `corporate look and feel' or to provide a dynamic element to layout. HTML documents have simple `<?php` (or the older `<?`) instructions embedded at the top and bottom and these replace all the site-specific markup. (Often the resulting document is barely recognisable as HTML!)
If this option is selected then AbiWord inserts `<?php include ...` statements in the head and at the beginning and end of the body of the HTML document. NOTE: Since some versions of PHP get confused by other processing instructions, such as `<?xml` , it may be necessary to uncheck the `Declare as XML' option.
Unfortunately this is not very customisable and AbiWord's new XHTML template system, available only through command-line conversion of documents to XHTML, can be used to much greater effect.

Declare as XML

XML documents, including XHTML ones, normally start with a line declaring them to be XML. This line is something like:
`<?xml version="1.0"?>`
Some (old) web browsers see this and get confused, and some versions of PHP also get confused because `<?` used to be regarded as a PHP insertion point.
If saving as HTML then this option is irrelevant; if as XHTML then uncheck this option to suppress the declaration.

Allow extra markup in AWML namespace

This option is specific to XHTML since it uses XML namespaces to add extra information to the HTML document. However, *this is not correctly functional at the moment and its use is not recommended*.

Embed (CSS) style sheet

AbiWord generates a lot of style-sheet information related to document styles and this can be embedded within the HTML document in a `<style><!-- --></style>` section or saved to an external style sheet (*filename*.html_files/style.css). The former is generally more useful; if an external style sheet is used on the website then it may be best to use the PHP option or, if converting from the command line, an XHTML template.

Embed images in URLs (Base64-encoded)

Images in the AbiWord document are normally saved in a subdirectory (*filename*.html_files/) and referenced by hyperlink (`<img src="filename.html_files/ ...`) but there is a mechanism for saving images within the HTML file itself (although this is generally frowned upon).

Templates & Command-Line Conversion

AbiWord-2.0 has a (still experimental) XHTML template mechanism that can be used when using AbiWord as a command-line tool to convert documents to HTML.
The simplest way to convert a file to HTML using AbiWord, on Unix systems at least, is:

```
AbiWord --to=html file.abw
```

which will import *file.abw* and export as HTML to *file.html* using default options.
It is possible, however, to specify export options on the command line using `--exp-props`:

```
AbiWord --to=html file.abw --exp-props="html4: yes"
```

The above command specifies that the output is to be HTML 4.01 and not XHTML. All six export options can be specified in this manner: `html4`, `php-includes`, `declare-xml`, `use-awml`, `embed-css` and `embed-images`. Each takes a `yes` or `no`, and multiple options can specified, e.g.:

```
--exp-props="html4: no; declare-xml: no; embed-css: yes"
```

The `--exp-props` argument can be used to define an arbitrary number of property name-value pairs. The above six options are *reserved* property names; there are two other reserved property names associated with XHTML templates: `html-template` and `href-prefix`.

XHTML Templates (EXPERIMENTAL)

The XHTML template is a skeleton XHTML document which the exporter first imports and then exports again with modifications. The template contains *Processing Instructions* which instruct the exporter to insert, e.g., the document title or the document text.
All of the exporter's processing instructions have the form `<?abi-xhtml- ... ?>`, e.g.:

```
<?abi-xhtml-insert title?>
<?abi-xhtml-insert meta?>
<?abi-xhtml-insert body?>
```

In the output, these instructions are replaced with, respectively, the document's title (from the document properties, as text), the document's metadata (document properties, etc., as HTML `<meta ... />`), and the document's content (as HTML, inside of the usual `<body>...</body>`).
The template system was written primarily for the AbiWord documentation. The goal was to provide a navigation menu for the AbiWord bundled help documentation. This consists of about 100 AbiWord documents, in several folders, which need to be converted to HTML. The navigation menu needed submenus, and the menu item corresponding to the current webpage needed to be highlighted, etc., and all using only HTML, CSS and relative links between webpages, images and style sheets.
To achieve this with a single template an element of flow control is necessary, and this is achieved by defining variables in the `--exp-props` argument, e.g.:

```
--exp-props="active-menu: howto"
```

One conditional mechanism is substitution in comments. The following in the template:

```
<style>
<?abi-xhtml-comment-replace property="active-menu"
comment="
tr.menu td.$$ div {
background-color: #0000ff;
text-decoration: underline;
}
"?>
</style>
```

will produce the following in the exported XHTML:

```
<style>
<!--
tr.menu td.howto div {
background-color: #0000ff;
text-decoration: underline;
}
-->
</style>
```

since the `$$` in the processing instruction comment text is replaced by he value (`howto`) of the specified property (`active-menu`).
The principal flow-control mechanism, however, is the use of `<?abi-xhtml-if` *`condition`*`?> ... <?abi-xhtml-fi ?>` where everything between the two processing instructions is ignored if *`condition`* is false. Unfortunately, the condition syntax is currently very limited (e.g., `<?abi-xhtml-if active-menu==howto?>`) and is likely to be rewritten. (These should be nestable.)
Another processing instruction is `<?abi-xhtml-menuitem ...?>` which creates a table cell of one sort or another with the specified label text.
Finally, to make relative links between resources possible, link addresses which start with a `$`

(e.g., `<a href="$/about.html">` and `<img src="$/images/logo.jpg" />`) will have the `$` substituted with the value of property `href-prefix`, e.g.:

```
--exp-props="html-template: normal.html; href-prefix: .."
```

This tells Abiword to use the template *normal.html* in the current directory and to prefix all links (marked with an initial `$`) with the path `..` so `<a href="$/about.html">` will change to `<a href="../about.html">`, and so on.

The best example of XHTML template usage is in AbiWord's abiword-docs CVS module in the directory abiword-docs/ABW/en-US/template/.