Editing XML files

There are at least 3 solutions for editing SGML and XML files with emacs.

Built in `sgml-mode` - major mode.

Working on this...

Additional package `xml-lite` providing minor mode working with built in SGML major mode.

Simple extension for built in emacs SGML mode. It provides support for indentation and colorizing edited files. It adds a few parameters and key bindings to major mode.

Table 1. xml-lite customization options.

Variable name	Default value	Description
`xml-lite-indent-offset`	`4`	Specifies the default indentation level for `xml-lite-indent-line'.
`xml-lite-in-dent-comment-offset`	`5`	Specifies the indentation level for XML comments.
`xml-lite-electric-slash`	`'close`	If non-nil, inserting a '/' after a '<' behaves electrically. If set to `indent', typing '</' just triggers reindentation. If set to `close', typing '</' inserts an end-tag for the enclosing XML element.

Table 2. xml-lite commands added to sgml-mode

Function name	Keys binding	Description
indent-for-tab-command	**`TAB`**	This is emacs built in function indeed. However it does not work in pure emacs sgml-mode. xml-lite adds support for intending for SGML and XML files and now it works in usual emacs way: “Indent line in proper way for current major mode or insert a tab.”
xml-lite-slash	**`/`**	Behaves electrically if `xml-lite-electric-slash' is non-nil. See xml-lite-electric-slash description for detailed info.
xml-lite-insert-end-tag	**`C-C /`**	Insert an end-tag for the current element.

Function name	Keys binding	Description
xml-lite-show-context	`C-c C-s`	Display the current context. Parse until it finds a start-tag as the first thing on a line. The context is a list of tag-info structures. The last one is the tag immediately enclosing the current position.

Future versions of emacs will contain xml-lite functions integrated into built in emacs sgml-mode. However at the moment you can download the last version of this package from this location [http://www.dogbiscuit.org/mdub/software/xml-lite.el].

Additional package `PSGML` providing major mode for editing XML and SGML files.

I started to work on this document using xml-lite mode. It works well and helps a lot if you don't spend too much time editing XML like documents. However, for large XML editing you definitely need more advanced library. PSGML gives you almost all you can need for your work with SGML/XML documents.

Installation and initial setup is easy. Just download the last available SGML [http://www.lysator.liu.se/projects/about_psgml.html] version. I use the last alpha 1.3.1 version with no problems. Unpack it somewhere and put to your `.emacs` file following lines:

Example 1. .emacs file code to load and activate PSGML library for XML and SGML files.

```
(add-to-list 'load-path "~/.emacs.d/site-lisp/psgml")
(require 'psgml)
(autoload 'sgml-mode "psgml" "Major mode to edit SGML files." t)
(autoload 'xml-mode "psgml" "Major mode to edit XML files." t)
```

There is one important trick for PSGML customization. Not all (if any) settings are visible in SGML, XML mode if you put them directly in **custom-set-variables** or **setq** expressions. To have them working they should be set at the time of switching to SGML or XML mode. The best and most common way is to add function setting all variables to `sgml-mode-hook` and `xml-mode-hook`. So they will be set correctly every time you will open SGML or XML file. The sample code doing this is in file available at following addresses.

Example 2. Sample settings for PSGML package.

sgml-adds.html [http://wttools.sourceforge.net/emacs-stuff/sgml-adds.html]

sgml-adds.el [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/*checkout*/wttools/wttools/emacs-stuff/sgml-adds.el?rev=HEAD&content-type=text/plain]

This file can be loaded from `.emacs` with command **(load-file "~/.emacs.d/site-lisp/sgml-adds.el")** or you can copy entire code to your `.emacs`. It contains not only my PSGML hook definition but also some more useful functions which can speed up XML/SGML coding. If you are interested only in

sample code of personal hook setting look for function definition **my-psgml-mode-hook**.

Table 3. The most important commands and their keys for package PSGML.

Function name	Keys binding	Description
sgml-insert-element	`C-c C-e`	Reads element name from mini-buffer and inserts start and end tags. Required elements in the content will be automatically inserted if the option `sgml-auto-insert-required-elements' is non-nil.
sgml-tag-region	`C-c C-r`	Makes the region into a new element. Reads element name from mini-buffer with completion.
sgml-insert-end-tag	`C-c /`	Inserts an end-tag for the current element.
sgml-split-element	`C-c RET`	Split the current element at point. If repeated, the containing element will be split before the beginning of then current element. Typical use is to start a new paragraph element when inside a paragraph.
sgml-insert-attribute	`C-c +`	Read attribute name and value from mini-buffer and insert attribute specification. If point is immediately after a start-tag, this command operates on that start-tag. Otherwise the command will operate on the element after point. The attribute name will be read with completion. If the attribute has a token list as declared value the attribute value will also be read with completion.
sgml-beginning-of-element	`C-M-a`	Move to the (content) beginning of the current element.
sgml-end-of-element	`C-M-e`	Move to the (content) end of the current element
sgml-forward-element	`C-M-f`	Move forward by element.
sgml-backward-element	`C-M-b`	Move backward by element.
sgml-down-element	`C-M-d`	Move down to the (content) beginning of the next element.
sgml-change-element-name	`C-c =`	Change the name of the current

Function name	Keys binding	Description
		element. Tries to translate attribute specifications. An attribute will be translated to an attribute with the same name.
sgml-kill-element	`C-M-k`	Kill the element following the cursor.
sgml-untag-element	`C-c -`	Remove tags from current element.
sgml-make-character-reference	`C-c #`	Convert character under point to a character reference. If called with a numeric argument, convert a character reference back to a normal character.
fill-paragraph	`M-q`	Fill paragraph at or after point. It is standard emacs function but can is useful for filling current paragraph. It does not takes a lot of time nor CPU but also does not work well when there is a mixed content in current tag.
sgml-fill-element	`C-c C-q`	Fills an element as a paragraph. This is a substitute for the normal `fill-paragraph'. The command uses heuristics to decide what should be a paragraph. 1. If point is in an element content, recursively fill the sub-elements. 2. Find the biggest element with mixed content containing point. 3. If the above element is mixed but contains elements with pure element content then fill what is between the pure elements as paragraphs and fill the pure elements recursively.
sgml-list-valid-tags	`C-c C-t`	List contextually valid tags.
sgml-next-trouble-spot	`C-c C-o`	The built-in parser can find some mark-up errors. To check the whole file go to the beginning of the buffer and use this command.
sgml-validate	`C-c C-v`	PSGML can not validate an SGML document. If you have a validating SGML parser, like

Function name	Keys binding	Description
		`sgmls` or `xmllint`, you can run the parser on your file with the command. Go to PSGML configuration section to see how to set your favorite XML parser.
sgml-normalize	**none**	Normalize the document in the buffer.