Editing SGML with Emacs and PSGML

Lennart Staflin

Published by ...

1 Introduction

PSGML is a major mode for editing SGML documents. It works with GNU Emacs 19.19 and later or with Lucid Emacs 19.9.

This (draft) manual is for PSGML version 0.4.

There are several functions for editing SGML documents. Indentation according to element nesting depth and identification of structural errors (but it is not a validating SGML parser). Menus and commands for inserting tags with only the contextually valid tags. Attribute values can be edited in separate window with information about types and defaults. Structure based editing includes movement and killing; and also several commands for folding editing.

2 Limitations

- Can not handle SHORTREF or variant concrete syntaxes, though it does allow unlimited length names. I suppose you can say that it accepts the "Core concrete syntax".
- Assumes that general entity references can be treated as data.
- The features CONCUR, DATATAG, LINK and RANK are not supported.

3 Installing PSGML

Place the `*.el` and the `*.elc` files in a directory where Emacs can find it. If you obtained PSGML without `*.elc` files, you can create them by letting Emacs byte compile the `*.el` files (use `M-x byte-compile-file`). Parsing SGML in Emacs Lisp is slow and needs all the speed it can get.

Put the following line in your .emacs:

```
(autoload 'sgml-mode "psgml" "Major mode to edit SGML files." t )
```

3.1 Modify search path for external entities

SGML can refer to external files (really entities) either with public identifiers or with system identifiers. PSGML uses the variable `sgml-public-map` to map public identifiers to file names.

A public identifier looks like

```
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN"
```

where "ISO 8879:1986" is the owner, "ENTITIES" is the text class and "Added Latin 1" is the text description (and "EN" is language). The default path would look for this file in

```
/usr/local/lib/sgml/ISO_8879:1986/entities/Added_Latin_1
```

Note: blanks are transliterated to '`_`' (and also '`/`' to '`%`') and the text class is down cased.

The default for `sgml-public-map` giving the above path is

```
("/usr/local/lib/sgml/%o/%c/%d")
```

The `sgml-public-map` variable can contain a list of file name templates where '`%P`' will be substituted with the whole public identifier, owner is substituted for '`%O`', public text class for '`%C`', and public text description for '`%D`'. The text class will be converted to lower case and the owner and description will be transliterated according to the variable `sgml-public-transliterations`. The templates in the list is tried in order until an existing file is found. An element can also be a dotted pair `(`*regexp* `.` *filename*`)`, the *filename* is a string treated as above, but only if the regular expression, *regexp*, matches the public identifier.

A system identifier looks like

```
SYSTEM "htmlplus.dtd"
```

where "htmlplus.dtd" is a system specific identifier. PSGML treats system identifiers as file names and looks for them in the directories listed in the variable `sgml-system-path` (default is either in current directory or in '`~/sgml`').

4 General

Files with extensions '`.sgml`', '`.sgm`' or '`.dtd`' will automatically be edited in SGML mode. To edit some other file in sgml mode, type `M-x sgml-mode RET` after finding the file.

If you can modify the file you can add a *Local Variables* list (see section 'Local Variables in Files' in *The Emacs Editor*) to the end of the file. This can make Emacs automatically set sgml mode and user options when the file is loaded. The simplest Local Variables list would look like:

```
<!--
Local Variables:
mode: sgml
End:
-->
```

You can also put a line at the top of the file to tell emacs to use sgml mode:

```
<!-- -*- sgml -*- -->
```

But remember that you can't have a comment before the *SGML declaration*.

You can use PSGML without specifying a DTD, it will the accept any tags and assume a content model of `ANY` and no tags can be omitted.

When PSGML needs contextual information it parses the document up to the point. During the parsing, it builds a parse tree. The parse tree is used to initialise the next parse, to avoid having to parse things already parsed. Changes to the buffer is supposed to prune the tree of all outdated information. But if you get strange complaints from the parser, try and back up a bit and use `C-c C-o` (`sgml-next-trouble-spot`).

5 Running an external SGML parser

PSGML can not validate a SGML document (see below what it can and can't do). If you have a validating SGML parser, like '`sgmls`', you can run the parser on your file with the command `C-c C-v` (`sgml-validate`).

Some variables control this function:

sgml-validate-command User Option

The command to validate an SGML document. The file name of current buffer file name will be appended to this, separated by a space. Default is `sgml -s`.

sgml-declaration User Option

If this is a string this will be inserted before the file name when running the `sgml-validate-command`.

sgml-offer-save User Option

If non-nil, `C-c C-v` (`sgml-validate`) will ask about saving modified buffers before running the validate command.

The built-in parser can find some markup errors. The command `C-c C-o` (`sgml-next-trouble-spot`) is the best way to use the built-in parser for this. To check the whole file go to the beginning of the buffer and use `C-c C-o`.

Some of the markup errors not found are:

- Errors in the SGML declaration.
- Errors in attribute specifications.
- Markup errors in entity replacement text.

6 Using a SGML declaration

PSGML does not understand the SGML declaration, it accepts one in the file but it is ignored. If you have the SGML declaration in another file you can make 'sgmls' use it when you use the C-c C-v (sgml-validate) command (see Chapter 5 [Validating], page 5).

PSGML has some options in what features it uses and what markup it creates. You have to set these options to make PSGMLs behaviour consistent with your SGML declaration and personal preferences.

sgml-omittag User Option

Set this to t if the SGML declaration has 'OMITTAG YES' and to nil otherwise.

sgml-shorttag User Option

Set this to t if the SGML declaration has 'SHORTTAG YES' and to nil otherwise.

sgml-always-quote-attributes User Option

If non-nil, quote all attribute values inserted after finishing edit attributes. If this variable is nil and sgml-shorttag is non-nil, attribute values that consists of only name characters will not be quoted.

sgml-minimize-attributes User Option

Determines minimization of attributes inserted by edit-attributes. If non-nil, omit attribute name, if attribute value is from a token group. If max, omit attributes with default value. Minimization will only be done if they produce legal SGML (assuming sgml-omittag and sgml-shorttag are set correctly).

7 Managing the DTD

PSGML needs to know about the DTD you are using for many of its commands. If you have a '`DOCTYPE`' declaration in your file, PSGML will try to parse this. Either the first time you do something that needs to parse the document or with the command `sgml-parse-prolog`. Big DTDs take some time to parse.

You can have the '`DOCTYPE`' declaration in another file either by setting `sgml-parent-document` to the other file or by creating a saved DTD and setting `sgml-default-dtd-file` to that file.

If parsing the DTD takes long time you can save the parsed DTD in a file using the command `M-x sgml-save-dtd`. Next time you can have PSGML load that file instead of parsing the DTD. For Emacs to find the saved DTD you must either save the DTD using the default name or do a `M-x sgml-save-options` after saving the DTD.

sgml-default-dtd-file User Option

> This is the default file name for saved DTD. This is set by `sgml-mode` to the buffer file name less extension plus the extension `.ced`, if that file exists. Can be changed in the Local variables section of the file.

sgml-parent-document User Option

> This can be set to a file name (a string) of a file containing the '`DOCTYPE`' declaration to use, or a list `(`*filename doctypename*`)`, where *filename* is a file name of a file containing then '`DOCTYPE`' declaration to use with the modification that the document type name is *doctypename*.

If you change the doctype you must execute `sgml-parse-prolog`, changes in the doctype are not automatically recognised.

8 Editing commands

See also the mode documentation (`C-h m`).

8.1 Inserting Markup

The commands that insert start-tags works only if the document has an associated DTD.

A list of valid tags is computed for a position in the buffer and is used for a menus and completion. The list will contain:

1. The end-tag for the current element unless `sgml-balanced-tag-edit` is non-nil. Furthermore it will contain end-tags for enclosing elements if the necessary omissible end-tag declarations has been made in the DTD.
2. The start-tags of all elements that could occur after point. If `sgml-omittag-transparent` is nil, the above will be limited to the elements that can occur within the current element.

Another type of computed list is a list of valid element names. This list is also used for completion. If `sgml-omittag-transparent` is nil, the list will only contain the elements that can be in the content of the current element.

The commands are:

`C-c <`
: Will ask for the tag to insert in the mini-buffer with completion on the the tags that are valid at point (`sgml-insert-tag`).

  If the option `sgml-balanced-tag-edit` is non-nil, inserting a start-tag will also insert the corresponding end-tag. If in addition `sgml-auto-insert-required-elements` is non-nil, tags for elements required between the inserted tags will also be inserted.

`C-c C-e`
: Insert start and end-tags for an element (`sgml-insert-element`). The name of the element is read from the mini-buffer with completion on valid elements.

  Required elements in the content will be automatically inserted if the option `sgml-auto-insert-required-elements` is non-nil.

`C-c C-r`
: Makes the region into a new element (`sgml-tag-region`). Reads element name from mini-buffer with completion.

`C-c /`
: Inserts an end-tag for the current element (`sgml-insert-end-tag`).

Menu bar:

'`Markup`'
: Selecting from this menu will insert markup. The menu contains two sub-menus with tags and with entities, some other markup and a user defined section.

  Sub menus:

'`Insert element`'
: Will pop-up a menu of valid elements and insert start and end-tags for the selected element. Selections from the menu works like the `C-c C-e` command.

'`Insert start-tag`'
: Will pop-up a menu of valid start-tags and insert the selected tag.

'`Insert end-tag`'
: Will pop-up a menu of valid end-tags and insert the selected tag.

'`Tag region`'
: Will pop-up a menu of valid elements and tag the region with the selection. Selections from the menu works like the `C-c C-r` command.

'`Insert entity`'
: Menu of all general entities defined in the DTD.

A menu is also available directly with a mouse button click in the buffer. In GNU Emacs it is the first mouse button combined with shift (`S-MOUSE-1`). In Lucid Emacs it is bound to the third mouse button. The mouse button click will pop-up a menu of valid tags. The tags list is as described above. Selections from the menu works like the `C-c <` command. With the following exception:

You can tag a region, with start and end-tag. There are two ways to indicate the region to mark:

1. Use the normal mouse commands to mark region.

   For this to work you muse either use *transient mark mode* (see section 'Transient Mark Mode' in *The Emacs Editor*.) or set the option `sgml-tag-region-if-active` to non-nil (don't set this unless you are sure that you want it).

2. Alternatively make a secondary selection, this is done by holding down the meta key and using the mouse buttons. See section 'Secondary selection' in *The Emacs Editor*. Some window managers intercepts these events, which make it hard use the secondary selection in Emacs.

User Options:

`sgml-balanced-tag-edit`
: If non-nil, inserting a start-tag will also insert the corresponding end-tag.

`sgml-auto-insert-required-elements`
: If non-nil, automatically insert required elements in the content of an inserted element.

`sgml-omittag-transparent`
: If non-nil, will show legal tags inside elements with omissible start-tags and legal tags beyond omissible end-tags.

`sgml-tag-region-if-active`
: If non-nil, the '`Insert tags`' menu will tag a region if the region is considered active by emacs. If nil, region must be active and `transient-mark-mode` must be on for the region to be tagged.

8.2 Markup completion

If you are typing in markup directly `M-TAB` will help you by completing a tag name, an entity name or a markup declaration name. If you type `M-TAB` after a plain word, `ispell-complete-word` will be invoked instead.

If you have typed

    &At⋆

and type `M-TAB` (assuming you use the '`ISOLat1`' entity set) you get:

    &Atilde⋆

8.3 Showing information

Commands for showing information obtained by parsing the buffer.

`C-c C-c` Shows, in the message area, context at point: if in a tag or in mixed content and the open elements (`sgml-show-context`).

`C-c C-w` Shows what element the character after point (under the cursor) belongs to; also shows context of element (`sgml-what-element`).

`C-c C-t` List contextually valid tags (`sgml-list-tags`).

8.4 Indentation

You can indent a line according to the depth of element nesting at the beginning of the line. To indent the current line use `TAB` you can also use `LFD` (`newline-and-indent`) to start a new line with correct indentation.

sgml-indent-step User Option

How much to increment indent for every element level. If nil, no indentation.

If this is nil, `TAB` will insert a tab instead of indenting.

sgml-indent-data User Option

If non-nil, indent in data/mixed context also.

8.5 Moving about

These commands move in the element structure. The commands uses knowledge of SGML syntax, and if available the specific DTD.

`C-M-a` Move to the (content) beginning of the current element (`sgml-beginning-of-element`).

`C-M-e` Move to the (content) end of the current element (`sgml-end-of-element`).

`C-M-f` Move forward by element (`sgml-forward-element`).

`C-M-b` Move backward by element (`sgml-backward-element`).

`C-M-u` Move up to before current element (`sgml-backward-up-element`).

`C-c C-n` Move up to after current element (`sgml-up-element`).

`C-M-d` Move down to the (content) beginning of the next element (`sgml-down-element`).

`C-c C-d` Move to the next place where data is allowed (`sgml-next-data-field`).

You can also move to the next place where there is some structural error with `C-c C-o` (see Chapter 5 [Validating], page 5).

8.6 Editing attributes

If you want to change the attributes of start-tag you can simply edit them directly in the buffer. Or you can place the cursor at or after the start-tag and use the `sgml-edit-attributes` command, available from the 'SGML'-menu or on `C-c C-a`. This will create a new Emacs window with all possible attributes listed in the form

attribute name `=` *current value*.

The *current value* may be shown as '`#DEFAULT`' if the attribute has not been given a value in the start-tag. The list also contains the attributes declaration as a comment. Note also that the *current value* is show without eventual quotes.

It is now possible to edit the attribute values. You can move to the next attribute with `TAB`. If you want to let an attribute have its default value use `C-c C-d`, this will insert a '`#DEFAULT`' in the value field.

If Emacs is running in a X window, the '`#DEFAULT`' will be underlined to distinguish it from normal values.

Finish the editing with `C-c C-c`; this will replace the attribute values in the main buffer with those edited. Note that values will be quoted as needed.

If you want to abort the editing, you can remove the window with `C-x 0` or if you want it neat, kill the buffer and remove the window.

Some other keys are:

`C-a` Go to the beginning of the value field. (`sgml-edit-attrib-field-end`)

`C-e` Go to the end of the value field. (`sgml-edit-attrib-field-end`)

`C-c C-k` Clear the value field. (`sgml-edit-attrib-clear`)

8.7 Changing and deleting

`C-c =` Change the name of the current element (`sgml-change-element-name`). Tries to translate attribute specifications. An attribute will be translated to an attribute with the same name. If the new element has no attribute with the same name, the attribute will be ignored. If there is an attribute with the same name but different declared content, a warning is given.

ID attributes are handled special, an attribute with declared value ID will always be translated to the attribute with declared value ID.

`C-c C-k` Kill next tag, markup declaration or process instruction (`sgml-kill-markup`).

`C-M-k` Kill the element following the cursor (`sgml-kill-element`).

`C-c -` Remove tags from current element (`sgml-untag-element`).

PSGML has a substitute for the normal `fill-paragraph`. The command `C-c C-q` (`sgml-fill-element`) uses heuristics to decide what should be a paragraph.

1. If point is in a 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.

The command `sgml-normalize` will

1. insert missing tags,
2. replace minimized tags with full tags,
3. fix attribute specification lists according to options set.

There is one option for the normalize command. With its default value, normalize may actually change the data content of some elements. But only by removing some white-space from the end of elements with omitted end-tags.

sgml-normalize-trims User Option

> If non-nil, `sgml-normalize` will trim off white space from end of element when adding end-tag.
> Default: `t`.

9 Folding editing

With these commands you can make parts of the text temporarily invisible to make it easier to see the overall structure of your text.

When folding a region all the lines but the first will be invisible. The first line of the region will still be visible with an ellipsis at the end.

See section 'Outline Mode' in *The Emacs Editor*.

`C-c C-f C-r`
: The region between point and mark will be folded (`sgml-fold-region`).

`C-c C-f C-e`
: The region between the start and end of the current element will be folded. (`sgml-fold-element`)

`C-c C-f C-s`
: Fold all the sub elements of the current element (`sgml-fold-subelement`).

`C-c C-s`
: Unfold the current line, assuming it is the first line of a folded region (`sgml-unfold-line`).

`C-c C-u C-e`
: Make all lines in current element visible (`sgml-unfold-element`).

`C-c C-u C-a`
: Make all lines in current buffer visible (`sgml-unfold-all`).

`C-c C-f C-x`
: Unfold current element and then fold the subelements (`sgml-expand-element`). If the current element is folded this expands what is visible.

10 Index

Variables

Functions

Key commands