SiteLite 2.0

This print is generated by SiteLite as an automatic pageassembly from
http://www.ia.hiof.no/~borres/sitelite/ver20/.
No effort is done to make the print wellformed and "pretty".

b.stenseth

Welcome to SiteLite 2.0

SiteLite is a program that:

- lets you build and maintain simple sites without in-depth knowledge of html
- lets you build and maintain complex sites with a minimum of effort on the tedious taks
- has a completely open architecture without databases or proprietary file formats. All files, txt or html, may be inspected and edited by any editor at any time.
- is based on a script file that describes your site, rather than an integrated IDE
- splits content, structure and layout in a simple and efficient way.
- lets you build and maintain sites in cooperation with other people, the source files may be anywhere on the internet.

If you want to test it, you can download from the Download page .

Some sites built with SiteLite:

- Shakespeare
- Illustrate
- Catalog
- Frames
- Pedagogy and Tecnnology
- Grafisk databehandling

The Idea

[Content] [Structure] [Layout] [Elements]

SiteLite is based on the following design philosophy:

- Simple things should be simple and advanced things should be possible. SiteLite should match a wide range of users, from novice to expert, and offer all an easy way to build and maintain sites.
- SiteLite should concentrate on automating the trivial tasks of site maintainence.
- SiteLite is building ready made sites and is not involved neither on the server nor on the client side during use of the site.
- No proprietary file formats are involved. All files that are built by SiteLite is of type html or txt, and may thus be opened, inspected and edited in any editor. If you at any point would like to abandon SiteLite, all your material is avaliable for other web-tools.
- Content, structure and layout is treated as separate issues. This is described in more detail below.

In general SiteLite operates according to the following illustration when a page is built:
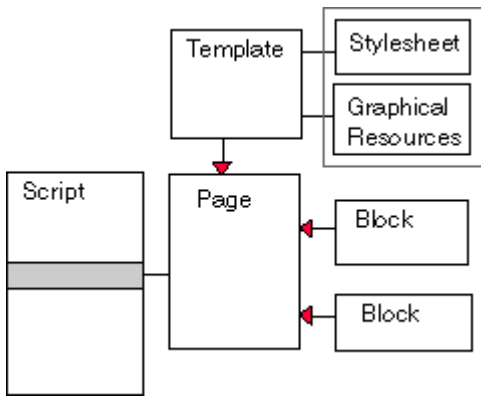
The script contains lines that associates a page with blocks of content, and a template.

Content

The content is written in the block-files. The content is any legal html text. The content may be prepared in any text-editor or html-editor. A page may be built with any number of blocks. A block-file may consist of one or more parts with an address label on each. The template is assumed to have matching addresses and the associated block parts are pasted into the template. Blocks may also be anonymous, in which case they are by default routed to the general BLOCK-address in the template.

Structure

The structure of the site is described in the script-file. The structure is basically a combination of hierarchy and sequence. The sequence is described by the sequence of the page definitions in the script, and the hierarchy by the levels of the pages, which is a part of the page description.

Layout

The layout of the pages are described by the templates and the associated style sheet and graphical resources. The template organise the overall structure of the page and routes the blocks to the appropriate places. The graphical resources are imagefiles like arrows, logo, background etc which are used by the templates. The style sheet defines mainly text styles, text colors etc.

Elements

To realise this strategy, and at the same time, give the user a large amount of freedom, SiteLite must rely on some information placed in the involved files. This information is in general described as elements. An element is a html-comment, for instance in a block a typical element is.

<!--BLOCK TARGET="ingress"--> this is the ingress<!--/BLOCK-->

and the matching element in the template is:

<!--ingress--> this is where the ingress will be put<!--/ingress-->

The elements has a form like HTML-elements, but are treated as comments, that is the <!-- and --> are enclosing the elements. This is chosen to avoid any conflicts with XML or HTML-elements, and different browser behaviour.

The page Elements describes the form of the elements in detail, and it describes the available elements in SiteLite.

Get Started

[..and go on]

The easiest way to get started is simply to start SiteLite and select menu *File-New*. This sets up a dialog box which lets you select where the site should be placed on your disk, where SiteLite should find the templates and where it should find all the resources, some gifs and a style sheet, that are necessary to realise the layout. The easiest thing to do is to accept the proposed defaults by clicking OK. When you build the site with menu *Tools-Build all pages* or by pressing the button with the hammer on, SiteLite will generate the necessary files in a reasonable directory structure.

This generates a script file that looks like this:

```
// Scriptfile for website MySite
// General site section

VERSION=2.0
CATALOG=d:\mysite2
TEMPLATES=templates
RESOURCES=gfx
KEYWORDS=sitelite,website
SITENAME=MySite
SITEAUTHOR=<a href="mailto:my.self@home.net">My Self</a>
SITEORG=http://www.hiof.no
MESSAGE=hello

// Page section

1,Home,I,index.html
BLOCKS:blocks\src_index.html
1,Sitemap,C,pages\toc.html
BLOCKS:blocks\src_toc.html
1,Page1,P,pages\pageone.html
BLOCKS:blocks\src_page1.html
1,Page2,P,pages\pagetwo.html
BLOCKS:blocks\src_page2.html
1,Siteindex,IX,pages\ix.html
BLOCKS:blocks\src_ix.html

// a print-page
0,All Pages,PP,pages\allpages.html
BLOCKS:blocks\src_allpages.html

//Template section
//.. no own templates defined


//Replace section
//.. is empty


//Collect section
//.. is empty
```

Six pages are prepared:

- A home page, called Home, which is stored in file index.html
- A site map page that lists all pages in the site
- Two general pages called Page1 and Page2 respectively
- An index page, which will show an index based on keywords in the other pages
- A page that collects the content of all the other pages.

One block of content is prepared for each of the pages. These are filled with preliminary content.

The selected pages are made to demonstrate components of a typical site. The pages "SiteMap", "SiteIndex" and "All Pages" may seem a little to ambitious for this simple site. You may want to delete the lines describing these pages and the associated blocks.

You will probably want to change the names of the pages as well as their filenames. You will probably open the block files in your favourite web editor, or text editor, and fill in some contents.

Adding new pages is simply done by inserting page lines after the pattern shown. The first parameter on the line, the digit, tells what level the page should be on. A section which looks like this:

```
1,Fruit,P,pages\fruit.html
BLOCKS:blocks\bfruit.html
2,Bananas,P,pages\banana.html
BLOCKS:blocks\bbanana.html
2,Apples,P,pages\apples.html
BLOCKS:blocks\bapple.html
```

will generate a section in a table of contents with fruit as a heading for bananas and apples.

You will see that SITEAUTHOR, SITENAME and ORG lines have a content that don't match your intentions and you will change these lines.

The script above generates the following catalog structure on your disk:
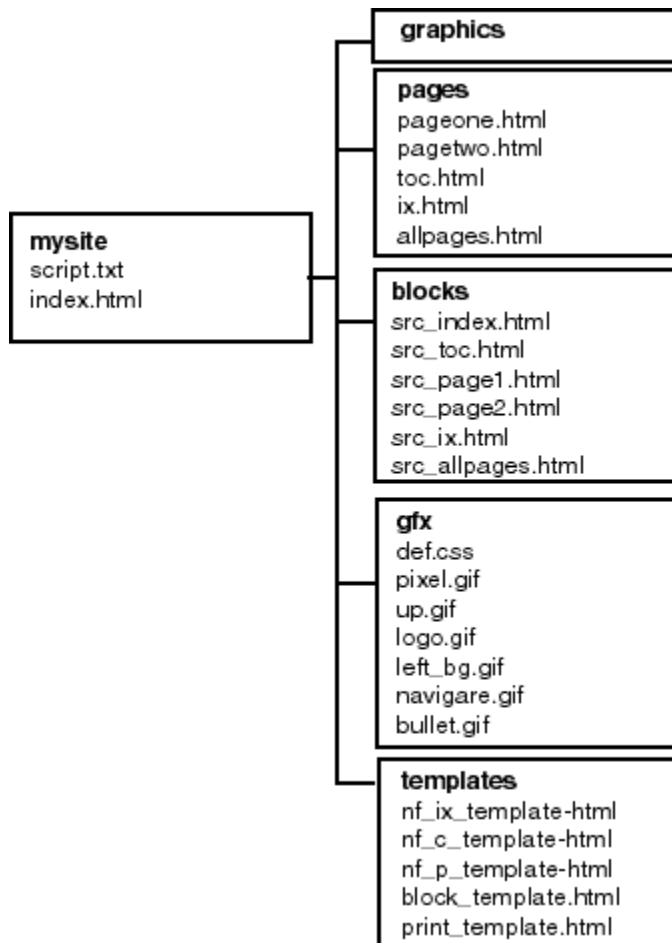
The catalog *pages* stores the pages as they are generated from SiteLite. The catalog *blocks* keeps the contents. The catalog *graphics* is empty and is prepared for you as a convenient place to put your graphics.

You may want to organise your material in any way you want. The catalog structure is built, and rebuilt, according to the filenames and paths for pages and blocks. **Note** however that SiteLite does not remove old versions if you change names of files, and SiteLite does not move files around within the site catalog. It does however help you to identify bad site-internal cross references if you redefine the file structure or

move pages.

The contents of the catalogs *gfx* and *templates* are filled from the programs resources and are the basic layout resources and templates which are distributed with SiteLite. You may edit any of those. Below is a short explanation of changes that you would probably like to do at once, in addition to changing the script file. If you want to learn more about resources and templates se pages Resources and Templates respectively.

Change the logo. As you can see on the page you are reading now, there is a logo in the page's top left corner. This logo is found in the file *logo.gif*. You will certainly want to edit this file to make it look like the logo of your organisation. You should keep your new logo the same size in the first attempt to customise the site. If you change the size you may have to edit some template files in addition.

Change the background colour of the left column. The file *left_bg.gif* should be edited.

You may inspect the other gif-files and see if you want to change them. Note that if you do anything with *navigare.gif* you may have to redefine the corresponding image map. The navigare.gif are the previous-home-next image in the pages lower-right and upper-right corners.

Don't touch the *pixel.gif*. It is used by SiteLite to create space on the pages where necessary.

▲

..and go on

Smart replaces

You can introduce general replaces in the script file and thus achieve a lot of effects, like changing an image all trough the site or automatic inclusion of html files or text files into pages. If you combine these replaces with your own tags in your material the freedom increases considerably. A tag is a html comment.

You can also make layout changes by editing the style sheet, def.css. You should have some familiarity with style sheets to do this, but the syntax is rather obvious and some educated guessing for simple changes like colors etc may do the trick. Browsers are not to be trusted when it comes to style sheets, so you should test such changes on the browsers that you consider important.

Modify templates

When you modify the template files you can really make things happen. You can change column widths and you can change headers and footers. You can easily introduce your own templates with your own tags and assign them to your own special page types. You should have some knowledge of the tags SiteLite recognises if you want to do major changes.

Redefine everything

You can throw away all templates, graphics and the style sheet and build everything according to your own head. SiteLite is basically a very simple program that in effect does replaces in html-pages. (In addition to composing structured lists like table of contents etc.) You may familiarise yourself with the present tags, introduce your own and put them into your templates. The only limitation is of course that you will not be able to make SiteLite produce composite data that it is not programmed for.

If you have suggestions for new data compositions that should be programmed and prepared for insertion I will be happy to hear about them.

The Script File

[A Sample] [Site part] [Page part] [Template part]
[Replace part] [Collect part] [Naming files] [Script for this site]

This file describes the structure of your web site and are the "glue" in the site. You may edit it directly in SiteLite. The script file is a text file that can as well be prepared and edited in any editor that saves your work in plain text.

The script consists of 4 parts:

- Site part
- Page part
- Template part, may be empty
- Replace part, may be empty
- Collect part, may be empty

These parts must occur in this order in the script, and:

- The sequence of the page-lines must reflect the sequence of pages, and blocks, you want in the site.
- The sequence of replace-lines are executed in the sequence they are given. If you, for some mysterious reason, plan cascading replace effects you should watch this.

A Sample

```
// a very simple script for a site with
// two pages
VERSION=2.0
CATALOG=c:\mysite\
SITENAME=Mini Site
1,First Page,P,pages\page1.html
BLOCKS:blocks\src_page1.html
1,Second Page,P,pages\page2.html
BLOCKS:blocks\src_page2.html
```

All blank lines are ignored. Lines that starts with "//" are considered as comments and are ignored by SiteLite.

Site part

Line	Description
VERSION=2.0	Must be exactly as stated so SiteLite 2.0 can appreciate this file as suited for version 2.0 of SiteLite
CATALOG=	full path to site catalog on disk
TEMPLATES=	full path, site local path or url to catalog with page templates. If the location is on the internet or on another disk volume than the site catalog, the templates are copied to a local catalog: **templates**. If the line is omitted, **templates** is assumed. If the indicated catalog is empty, SiteLite produces "standard" templates as needed.
RESOURCES=	full path, site local path or url to catalog with graphical resources and style sheet. If the location is on the internet or on another disk volume than the site catalog, the resources are copied to a local catalog: **gfx**. If the line is omitted, **gfx** is assumed.

	If the indicated catalog is empty, SiteLite produces a minimum set of "standard" resources.
KEYWORDS=	comma separated list of keywords for the site, on one line
MESSAGE=	any string or html-string. Inserted at bottom of pages.
SITENAME=	any string or html-string. Appears as top "banner"
SITEORG=	a complete URL to the organisation associated with the site. For instance: www.hiof.no
SITEAUTHOR=	any text, normally a complete reference to the authors home page or a mailto. For instance Børre S

Page part

consists of any number of page lines. All page lines may be followed by any number of **block** lines and **based_on** lines. A block-line identifies content files that will be filled into the page, and the based_on-lines are explicit cross references which are independent of the overall hierarchical structure of the site. These cross references will appear below table of content in the left column in the original templates.

In the description below [text], means that the brackets with content may be omitted.

A page line has this form:

indent,name[,type[,location[,author[,msg[,option]]]]]

- **indent** is the level in the site hierarchy, 1 is on top. Pages with level 0 is built but will not appear in table of contents etc. Pages with level 0 are typically used as collected pages for print, for frames etc.
- **name** is any name for the page. You may use this name as a reference within the site. A name must not contain any of the characters:/:.#\".
- **type** identifies the template to use. Predefined types are:
    - **P**: normal page. This will be the type for the majority of the pages in a site.
    - **I**: start page. This may be used as a special first page.
    - **C**: SiteLite will attempt to build a site map on this page.
    - **IX**: SiteLite will attempt to build a site index on this page.
    - **PP**: SiteLite will attempt to assemble the content of other pages on this site. For instance to make a printable selection of pages from the site.
  You may use any other text that identifies a template you have defined. If type is omitted or blank, option K is assumed and no template will be requested. See below. There is a set of template types reserved for frame based sites. The types are **FT,FL,FF,FT,FP,FC,FIX**. See page Templates for a full explanation of templates.
- **location** is one of four:
    - a filename relative to site catalog
    - an absolute filename to the same disk volume as the site catalog
    - an absolute filename to another disk volume than the site catalog
    - a complete URL.
  If location is omitted or blank, the name of the page will simply appear as a heading or separator in tables of contents. If the location is an URL or an other disk the page will be imported, copied to the local disk. See page Import for an explanation of import.
- **author** is any text, but is normally a complete link, href or mailto. If omitted or blank, the value of the SITEAUTHOR-line is assumed.
- **msg**: any string or html-string. Inserted at bottom of pages. If omitted or blank, the value of the SITEMESSAGE-line is assumed.
- **option** only two options are recognised at the moment:

- K: keep this page's body as is. No template, nor any blocks will be involved. Type is meaningless, and is ignored for pages with option K. All tables of contents, collections, replace and indexes are updated for the page.
- N: Works only when location is a complete URL. No import with copy. No template, nor any blocks will be involved. Type is meaningless, and is ignored for pages with option N. The page is referenced directly in all tables of contents. This option is best suited for sites with frames.

If you use both K and N, KN, N will be effected.

A block line has this form:

BLOCKS:location[,location]

- **BLOCKS:** indicates that this is a list of block files, and that their content will be extracted and inserted into the nearest page above in the script.
- **location** is one of four:
  - a filename relative to site catalog
  - an absolute filename to the same disk volume as the site catalog
  - an absolute filename to another disk volume than the site catalog
  - a complete URL.

If the location is an URL or an other disk the block will be imported, copied to the local disk. See page Import for an explanation of import.

A based-on line has the following form:

BASED_ON:pagename[,pagename]

- **BASED_ON:** indicates that this is a cross reference list for the nearest page above in the script.
- **pagename** is exactly the name given in the referenced pages page line.

Template part

is optional and may consist of any number of lines of type:

ATEMPLATE=pagetype,location

- **pagetype**: A text that will be used on page lines to identify this template
- **location** is one of four:
  - a filename relative to site catalog
  - an absolute filename to the same disk volume as the site catalog
  - an absolute filename to another disk volume than the site catalog
  - a complete URL.

If the location is an URL or an other disk the template will be imported, copied to the local disk under the catalog templates.

Following types and associated files are predefined:

```
// common
PP - print_template.html
B  - block_template.html

//for noframe sites
P  - nf_p_template.html
I  - nf_i_template.html
C  - nf_c_template.html
IX - nf_ix_template.html
W  - wrapper_template.html

//for framebased sites
FF - f_f_template.html
FT - f_t_template.html
FL - f_l_template.html
FI - f_i_template.html
```

```
FP - f_p_template.html
FIX- f_ix_template.html
FC - f_c_template.html
```

If you reuse any of the above types you override the default template definition associated with that type. See page Templates for a full description.

▲

Replace part

is optional an may consist of any number of lines of one of the four following types.

RSIMPLE=sout,sin	replaces *sout* with *sin*
RDOUBLE=left,right,sin	replaces any text between *left* and *right* with *sin*
R1QUOTE=key,sin	replaces first quoted text after *key* with *sin*
R2QUOTE=key1,key2,sin	replaces first quoted text after *key2* after *key1* with *sin*

Note that these replaces are done in all published pages as they are generated. They have no effect on the source blocks. If you want to make global changes in your site, including blocks, you may use the general replace tool in SiteLite.

The *sin* parameter may be any text or it may be the name of a .html or .txt file that will be inserted. If the *sin* parameter starts with **#** the rest of the string is interpreted as one of three:

- an absolute filename.
- a filename relative to the site catalog
- an URL

▲

Collect part

is optional.

Any number of lines of the following type.

COLLECT:name,leftpar,rightpar

- **COLLECT:**Tells SiteLite to collect a list of all text enclosed in parentheses leftpar - rightpar.
- **name** is the name of the collection. The collected list will be inserted at COLLECTION-elements with this name as a property.
- **leftpar** and **rightpar** may be any text, comments or html.

Any number of lines of the following type.

INDEX:location

- **INDEX:**Tells SiteLite to collect a list of keyword occurences.
- **location** is the name of the file with a list of keywords, one keyword on each line. The keyword may be followed by any number of synonyms. The lines:

```
Hallo,HALLO,HI,Hi,hi
Welcome,Villkommen,velkommen
```

is a legal (but not very useful) file-content. The location may be an absolute path, a site local path

or an URL.

Naming files

NOTE that all filenames in a script should use backslash (\\) as path separator, while all URLs should use forward slash (/). Using a forward slash (/) in a filepath makes SiteLite try to interpret the filename as an URL, sometimes with unpredictable results, and allways with unwanted results.

A filename local to the sitecatalog	templates\mytemplate.html
An absolute filename	d:\othersite\templates\mytemplate.html
An URL	http://www.ia.hiof.no/~borres/gb/templates/mytemplate.html

Script for this site

This SiteLite documentation is made by the following script: SiteLite script File

Elements and tags

Templates and blocks have elements that control how contents should be formatted or routed. An element has the basic form:

`<!--TAG ATTRIBUTE="value"--> content <!--/TAG-->`

There may be many possible attribute="value" pairs in each element, separated by a space. Note that the value-part must be quoted. If the attributes are omitted there is always a set of default values that will be used. The content-part may be empty, even if that may be rather meaningless i some cases.

Elements recognised by SiteLite are:

`<!--BLOCK TARGET="nnn" WRAPME="OFF"--> content <!--/BLOCK-->`

> This element is used in block files to identify the parts that should be extracted when assembling a page. The attribute TARGET instructs SiteLite to put this part as content of an element called nnn in the associated template:
> `<!--nnn-->content <!--/nnn-->`
> When the attribute TARGET is not present, SiteLite expect to find:
> `<!--BLOCK-->content <!--/BLOCK-->`
> in the template. Content in the template is overwritten in both cases
>
> The WRAPME- attribute instructs SiteLite to take this part from the block and wrap it in the template *wrapper_template.html* before it is transferred to the pages target template. Default is OFF.

`<!--CONLIST BULLET="ON"--> content <!--/CONLIST-->`

> This element is the place where SiteLite builds a page relevant table of contents. This table includes all pages at level 1, all ancestors of the page, all siblings and all children. This is the normal left-column page list. If the attribute BULLET is ON, the current page is marked with a bullet. If it is OFF, the current page is only marked with different style.
>
> If a attribute APPEND="ON" is included, the toc is appended to existing content in element. Otherwise this content is overwritten.

`<!--TOC COLS="3" MIN_LEVEL="1" MAX_LEVEL="1000"--> content <!--/TOC-->`

This element is the place where SiteLite builds a complete site map. A complete site map is a table of contents for all pages with levels indicated by the MIN_LEVEL and MAX_LEVEL attributes. You may control this by telling which levels to include. Default is all pages with level greater than 0. The attribute COLS instructs SiteLite to build the site map in a table with columns. 3 columns is default.

If a attribute APPEND="ON" is included, the toc is appended to existing content in element. Otherwise this content is overwritten.

<!--SINGLETOC COLS="3" MEMBERS="SIBLINGS" STYLE="tlevel1"--> content <!--/SINGLETOC-->

This element is the place where SiteLite builds a table of contents which includes all siblings of a page, or all direct children (MEMBERS="CHILDREN"). The table is built with the value of the style-attribute, which may be any style found in the style sheet. The default values are as above. The attribute COLS instructs SiteLite to build the site map in a table with columns. 3 columns is default.

If a attribute APPEND="ON" is included, the toc is appended to existing content in element. Otherwise this content is overwritten.

<!--LOCALTOC LEVEL="2" SHOW="ON"--> content <!--/LOCALTOC-->

This element is the place where SiteLite builds a page local table of content, normally in the start of the page. Attribute LEVEL instructs SiteLite to include HTML-headings (h1,h2,h3,h4,h5,h6) of LEVEL's value in the table of content. Values of LEVEL must be in the range 1..6. If attribute SHOW is OFF, no table is built at all.

If a attribute APPEND="ON" is included, the toc is appended to existing content in element. Otherwise this content is overwritten.

If this element is missing, SiteLite inserts an element with attributes as shown above immediately after first header 1, h1-element, in the page.

<!--BASED_ON--> content <!--/BASED_ON-->

This element is the place where SiteLite builds a list of page names given in the BASED_ON-line(s) in the script.

If a attribute APPEND="ON" is included, the list is appended to existing content in element. Otherwise this content is overwritten. If the list is empty, the content is always removed.

<!--INDEX COLS="3" REFERENCE="ON" SORT="OFF"--> content <!--/INDEX-->

This is where SiteLite place an index table with keywords that are inserted by the INX-tag below and/or from the file given in an INDEX-line in the script. If the attribute REFERENCE is ON, the entries in the index table links back to the actual page. ON is default. The attribute COLS instructs SiteLite to build the index in a table with columns. 3 columns is default. If SORT is ON the page entries for each keyword is sorted alphabetically, otherwise they are presented as found, ie. page sequence.

If a attribute APPEND="ON" is included, the index list is appended to existing content in element. Otherwise this content is overwritten.

<!--INX NAME="name"/-->

You may enter this tag anywhere in the text: blocks, pages or templates. SiteLite consider name as a indexable keyword and produces a list of those keywords associated with the pages where the INDEX-element occurs.

<!--WRAP--> content <!--/WRAP-->

This used only one place: In the template wrapper_template.html, which has type W. It marks

where the wrapped block should be inserted. No attributes. See the WRAPME-attribute of the BLOCK-element. Content is overwritten.

<!--PRINTLIST PAGES="xx" TEXTONLY="OFF"--> content <!--/PRINTLIST-->

This instructs SiteLite to collect content from a number of pages and put it here. xx is interpreted as:
- A list of page names separated by ,. If a page name has the extension '+', all pages that are children of this page is included.
- A list of pages file names separated by ,. Same '+'-extension is possible.
- ALL : all pages with level greater than 0. This is default

If the attribute TEXTONLY is ON, all text from the specified pages are extracted. All tags are removed and no formatting is preserved. Default is OFF.
If a attribute APPEND="ON" is included, the material is appended to existing content in element. Otherwise it is overwritten.

<!--PRINTABLE--> content <!--/PRINTABLE-->

This tag is used to identify the parts of a page that should be included in a make-a-print-page operation, see PRINT-element above. SiteLite inspects a ready-built page and extracts content between elements <!--PRINTABLE--> and <!--/PRINTABLE-->. SiteLite does not identify possible nested tags. The safest thing is probably to include this tags in the template-files only, to avoid nesting. No attributes possible.

<!--COLLECTION NAME="name" LINESEPARATOR="<br>" REFERENCE="ON"--> content <!--/COLLECTION-->

SiteLite reports its collections according to COLLECT-lines in the script file here. name is the name given in COLLECT-line in the script. LINESEPARATOR instructs SiteLite to put the associated value in front of each item in the collection. Default is <br>. If REFERENCE is ON a link back to the page where the item was collected is inserted. ON is default.

If a attribute APPEND="ON" is included, the collected list is appended to existing content in element. Otherwise this content is overwritten.

<!--BLOCKFILENAME/-->

This tag is used only one place: in the template block_template.html. It is used to set in the blocks filename when a new block is created, to increase readability.

<!--NEXT/-->

A reference to the next page is inserted as a replace of the first quoted string after this element.

<!--PREV/-->

A reference to the previous page is inserted as a replace of the first quoted string after this element.

<!--HOME/-->

A reference to the home page is inserted as a replace of the first quoted string after this element. A home page is identified as the first page with level greater than 0.

<!--SITEORG/-->

The value from the SITEORG-line in the script is inserted as a replace of the first quoted string after this element.

<!--SITEAUTHOR--> content <!--/SITEAUTHOR-->

The value from the SITEAUTHOR-line in the script is inserted as the content of this element.

<!--PAGEAUTHOR--> content <!--/PAGEAUTHOR-->

The author-value from the page-line in the script is inserted as the content of this element. If no page author is given, the site author is inserted.

<!--SITENAME--> content <!--/SITENAME-->

The value from the SITENAME-line in the script is inserted as the content of this element.

<!--PAGENAME--> content <!--/PAGENAME-->

The value from the name in the page-line in the script is inserted as the content of this element.

<!--AMESSAGE--> content <!--/AMESSAGE-->

The value from the message in the page-line in the script is inserted as the content of this element. If no message is given, the value of the MESSAGE-line is inserted.

<!--DATE--> content <!--/DATE-->

SiteLite inserts today's date as the content of this element.

<!--STAMP--> content <!--/STAMP-->

SiteLite places is "fingerpint" here. SiteLite insists on leaving a fingerprint in the site. You can control where with this element. A good idea is to place it in the first, or home, page in the site.

<!--TOTOP/-->

SiteLite inserts an arrow and a reference to the top of the page as replace for this element. An anchor: is anticipated at the top of the page, normally as part of the template.

<!--PAGENO--> content <!--/PAGENO-->

SiteLite inserts the page's sequence number here. Just a simple sequence number without any consideration of level or visibility. Usefull in "slides-sites".

<!--PAGECOUNT--> content <!--/PAGECOUNT-->

SiteLite inserts the number of pages here, without any consideration of level or visibility. Usefull in "slides-sites".

<!--TRAIL--> content <!--/TRAIL-->

SiteLite inserts an entry of this kind here
home > chapter > page
That is: all direct ancestors, including home and current page.
No attributes.

<!--PATCH FILENAME="filename" STRUCTURE="ON"/-->

Collects content from a file according to the same priority as normal blockextraction: First SiteLite looks for a set of BLOCK-tags, then for a set of BODY-tags, then for a set of HTML-tags. If none of these is found, the files contents is copied as is.
If STRUCTURE="OFF" or the structure attribute is omitted, the file is copied as is.
The file, or extraction from file, replaces the PATCH-element.
Filename is interpreted as relative to sitecatalog, full filepath or url.

Templates

[Standard] [Advanced] [Frames]

The number of templates and their use is a compromise between efficiency and ease of use.

On one hand it should be as easy as possible to establish and maintain sites for users who are not fluent in html and style sheets. This indicates that SiteLite should propose a "reasonable" solution without too much adjustment. The first section below describes the templates in a "standard" solution.

On the other hand SiteLite should offer great freedom for users who want to design their own solution from the bottom. This suggests that SiteLite should not force any special solutions. The section at the end of this page, Advanced, suggests an alternative way to think templates.

If you don't start out with a full set of templates, copied from an other site, SiteLite generates templates when needed. You will only see the templates that you have requested from page lines in the script. Types that SiteLite are enable to generate are: P,I,C,IX,B,PP,W,FF,FT,FL,FP,FI,FC,FIX. These are explained below.

If you plan to build a site with frames, you should have a look at the Frames section below.

Standard

The basic templates are:

- **nf_p_template.html**, for pages of type **P**.
  This is the normal templates which describes the majority of pages in a site.
- **nf_i_template.html**, for pages of type **I**.
  This is identical to the **P** template. It is included to make it easy to produce a different first page, or entry page in the site.
- **nf_c_template.html**, for pages of type **C**.
  This is a page that looks in general like the the P-type template, but it has an element for generating a site map, a TOC-element.
- **nf_ix_template.html**, for pages of type **IX**.
  This is a page that looks in general like the the P-type template, but it has an element for generating a site index, an INDEX-element.
- **block_template.html**, for generating empty new blocks, type **B**.
- **print_template.html**, for pages of type **PP**.
  This is a page that wraps a collection of one or more pages, referred to as a print page. It expects a block with an element of type PRINT.
- **wrapper_template.html**, for wrapping blocks, type **W**.
  This is a template that is used if it is necessary to wrap a block in a table structure, for instance to achieve an empty left column. You will probably not find it meaningful to use the W-type on a page line. It is associated with the WRAP- property of the BLOCK-elements. One of the examples that you can download demonstrates its use. If you find the use of little interest, just forget it.

The general layout of the P,I,C and IX templates are like this:
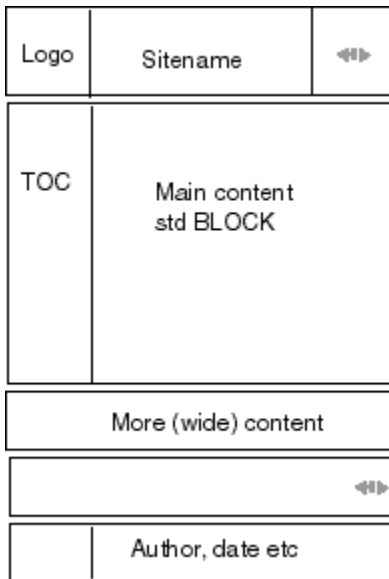
Advanced

Consider the following :

- If you inspect the four templates, types P,I,C,IX, you will see that they are almost identical. The only difference is that the C-template contains a TOC-element, and that the IX-template contains an INDEX-element, while P and I are identical.

  If you want to do extended experimenting with the layout, it may be better to include the elements in the associated blocks and simply use only P-templates. This limits the templates to edit down to one.

- You are free to introduce any template you want, with a matching type. If you reuse one of the types reserved by SiteLite: P, I, C, IX, B, PP, W, FF, FT, FL, FP, FI, FC, FIX your definition overrules the original. The way to define a new template is to introduce a ATEMPLATE-line in the script. For instance:
      ATEMPLATE=Q,templates\nf_q_template.html
  and you may use it on a page line, like this:
      1,thePage,Q,pages\thepage.html

- When you customise your templates or build new ones, you should always refer to common resources as if they were in the gfx folder, even if you have given an other name to the resource folder in the script. SiteLite will adjust this addressing when pages are built. It increases safety and speed to actually use this name also in the script.

- The block template may be customised to automatically prepare blocks with more BLOCK-elements of different type that matches the templates you use. For instance you may design a block like this

```
...
<!--BLOCK TARGET="HEADING"--><!--/BLOCK -->
<!--BLOCK TARGET="INGRESS"--><!--/BLOCK -->
<!--BLOCK TARGET="ARTICLE"--><!--/BLOCK -->
...
```

  with a matching template:

```
...
<!--HEADING--><!--/HEADING -->
<!--INGRESS--><!--/INGRESS -->
<!--ARTICLE--><!--/ARTICLE -->
...
```

Frames

SiteLite can easily be used to build and maintain frame based sites. You have all the freedom you would like to set up the framelayout. The easiest way to investigate frames in SiteLite is to build a new site with frames, Menu *File - New*, and check the *Use Frames-* box in the dialog box.

SiteLite suggests a simple solution with a frame structure:

```
        TOP
LEFT
        HOME
```

SiteLite produces 4 page lines which describes the frame structure in the script:

```
0,Frameset,FF,index.html
0,Topframe,FT,top.html
0,Leftframe,FL,left.html
1,Home,FI,home.html
  BLOCKS:blocks\src_home.html
```

The three first page lines has level 0, and will thus never appear in tables of contents. The best thing is to investigate the four pages. It is important that the name given to the home-frame in the page Frameset matches the name of the page: home.

You will also see that SiteLite has a separate set of templates for frame-based sites. Their meaning and intended use should be evident from the names and the way they are used in the site SiteLite generates. Templates meant for frames have names starting with f_ as opposed to the templates for non-frame sites which begin with nf_. The predefined templates are:

- **f_f_template.html** for defining the framesets, type **FF**
- **f_t_template.html** for defining the top frame with the site title, type **FT**
- **f_l_template.html** for defining the left frame with table of contents, type **FL**
- **f_i_template.html** for defining the home page in the main frame, type **FI**
- **f_c_template.html** for defining a site map page in the main frame, type **FC**
- **f_ix_template.html** for defining a site index page in the main frame, type **FIX**
- **f_p_template.html** for defining any other page in the main frame, type **FP**

You are of course free to define your own templates, and your own frame structure.

Resources

[Gifs] [Stylesheet]

Resources are used as a general name on the images and the style sheet that realises the layout. The resources are normally kept in the catalog **gfx**. SiteLite depends on some gif's and a style sheet. When you modify templates or roll your own, you may find that you will add resources. When you refer to resources, always refer to them as if they were in a catalog called gfx, even if you have given a different name in the RESOURCE-line in the sript. SiteLite will identify the references to resources and correct them. It is safest and fastest to use the standard name gfx as the name on the catalog you specify in the RESOURCES=-line in the script.

Below is an explanation of the resources that should be present in gfx in a "standard" site, as SiteLite makes them without any changes in templates. It may be useful to have a look at the source for, for instance this page.

Gifs

bullet.gif is a small (3x5 pixels) image that is used to mark the current page in a table of contents. See element CONLIST for a description of how to turn it on or off. SiteLite hardcodes use of this image. May be edited, but should not be removed if you don't abandon the idea of a marker once and for all.

logo.gif is the logo (130x30 pixels) that appears in the top-let corner of a "standard" page. May be, and will most certainly be, edited. If you change the size of the logo, you should inspect the relevant templates to make sure it does not corrupt the layout.

navigare.gif is the small (50x30 pixels) cluster of arrows that appears in the upper-right and lower-right corners of a page. If you change the dimensions of this, you must also change the corresponding image map in the relevant templates.

up.gif is the little (17x9 pixels) arrow that brings you to the top of the page. You may edit this, but you should not delete it. SiteLite hardcodes this arrow together with a reference as a substitution for the <!--TOTOP/--> element. The reference depends on an anchor in the top of the template, or page.

pixel.gif is a small (1x1 pixels) transparent image that may be used to pad spaces in the pages. SiteLite does not hardcode any use this image, but it may be used in the templates. You may find it useful, since it is not possible to realise all layout plans with style sheets.

left_bg.gif is an image (2500x20 pixels) that makes the background of the pages, including the left column. SiteLite does not hardcode any use of this and it is up to your layout plans if you need this image or if you want to change colors and/or dimensions.

Stylesheet

[Hardcoded styles] [Template styles] [General styles]

SiteLite relies on cascading style sheets, and the file **def.css** is important. You will find the file in the catalog gfx, and there should be a reference to the file in all templates, typically by the line:
<link rel=STYLESHEET href="gfx/def.css">
(the actual reference is correct by SiteLite during build.)

If you are not familiar with style sheets, you should not be too worried. You may achieve great results with some educated guessing. There are lots of sources for Cascading Style Sheets (css) on the Internet. One is: http://builder.cnet.com/Authoring/CSS/ss02.html

The default styles distributed by SiteLite are chosen to be "reasonable", but you will almost certainly want to do some adjustments to reflect your preferences; colors, textsize etc.

The styles defined in def.css falls in three categories.

1. Styles that are hard coded in SiteLite. Use of these styles are inserted by SiteLite in a way that cannot be controlled by the user. The user may however change the style definitions.
2. Styles that is used by SiteLite templates. These styles are used in "standard" templates. User may inspect and edit their use in the templates or blocks, as well as the styledefinitions in the stylesheet.
3. General styles for standard html-tags. Some general styles are included for users convenience, so they easily may be edited in the style sheet.

Hardcoded styles

When SiteLite assembles lists and table of contents, it inserts styles as an integrated part of the lists. These styles are chosen and described in a way that allows the user maximum control of the appearance. Below is a list that explain their use and refer to the actual styles names in the style sheet.

Page relevant table of contents

marked as CONLIST in the templates, is the table of content that are made for each page, and which normally appears in the left bar of the page. Each of the possible levels in this table is described with its own style. A page with level 3 is for instance described by SiteLite as:
<div class="**clevel3**">pagename and reference</div>
There are defined 11 clevel-types to match page levels. These are found in the style sheet as:
.clevel0
.clevel1
etc

There are a matching set of styles **aclevel** which are used to mark the current page, the page that owns the table of content.

Site map

The site map, marked as TOC in the templates, have a structure that are like the CONLIST. To make the freedom of choice as great as possible there are defined a parallel set of styles for this table of content. The styles are found as **tlevel**, 0 to 10.

PageLocal TOC

The table of content that normally appears in the head of a page, with references to subsections in square brackets are controlled by the style: **localtoc**

Index

The index is marked as INDEX in the templates. An index consists of a series of keywords, each with a list of page references. The actual styles that are inserted by SiteLite are
<div class=**"indexkeyword">**keyword</div>
<div class=**"indexentry">**pageref1</div>
<div class=**"indexentry">**pageref2</div>
etc

Based on

If you use BASED_ON:-lines in the script, SiteLite expects to find a <!--BASED_ON--> -tag in the template. The actual list of pages are hard coded by SiteLite as:
<div class=**"basedon">**page name or reference</div>

Template styles

These styles are introduced to match specific needs of SiteLite. Their use is not hardcoded, or forced, but they are generally useful. The indicated use refers to the "standard" templates. You are of course free to use them in another way, simply discard them or introduce new styles, when you construct your own templates.

basedhead

Used to wrap the BASEDON-element, for instance
<p class="basedhead">
<!--BASEDON APPEND="ON"-->Based on pages:
<!--/BASEDON--></p>

sitename

Used to wrap the SITENAME-element

mainfooter

Used to wrap the footer, including SITEAUTHOR, PAGEAUTHOR, DATE, AMESSAGE elements. Offsets the content to match the main area on the page.

leftfooter

Used to wrap the footer, including SITEAUTHOR, PAGEAUTHOR, DATE, AMESSAGE elements. No offset, used by frame pages and print pages.

If you inspect def.css you will find that there are some styles used to wrap predefined (PRE) contents. They are defined to document program code. You may use them, modified or not, for other purposes. They are included to show samples of some useful styling with blocks and background color.

You may find it usefull to make your own styles. Probably the best candidates are (sub)classes of **div**, **p** or **pre**.

General styles

To control the overall appearance of your pages, it is useful to be able to manipulate the appearance of general html-elements. The html-elements that are defined in the std style sheet are:

- BODY, defining styles for the page as a whole
- P, defining the style for general paragraphs
- A:link, defining style for links
- A:visited, links that have been visited
- A:active, the active link
- A:hover, when you move cursor over a link (Does not work for all browsers)
- H1, defining the style for heading, level 2
- H2, for heading level 2
- H3, for heading level 3
- H4, for heading 4
- UL, for unordered lists
- OL, for ordered lists
- TD, for table elements
- PRE, defining style for predefined text

You may choose to include more definitions at will.

Addressing

[Images] [Pages]

SiteLite does a considerable amount of recalculation of references. Two items is of special interest: referencing of images and referencing directly from one page to another within the site.

Images

Images may be addressed relative to the block the reference appears in or relative to the page that the block will be pasted into. A block based reference may be smart if you use some WYSIWYG-html editor. A page based reference is faster since you may turn off the check-images option while building. See menu *Properties-When Building*.

If you organise the catalog structure with this in mind, you may get both advantages. Two organisations may be worth consideration:

- Put page and belonging blocks in the same catalog.
- Put all images in a catalog that has the same relative address from the block-catalog(s) as from the page-catalog(s).

When blocks and pages are imported, SiteLite attempts to copy images at the same time. Their addresses are calculated as relative to the page they will appear in. Images that are referenced by a complete URL

or a complete file path are not copied. SiteLite 2.0 does not copy other files, like applets, sounds or multimedia files.

Pages

SiteLite offers the possibility to refer pages within the site by name in stead of by filename. This may be a good idea if you don't intend to change pagenames.

If you choose to use filenames, that is normal referencing, you **must** make the reference relative to the page the reference will appear in, not the block it is edited in. If you run a check on all pages, with menu *Tools-Check all Pages* or click the magnifying glass, SiteLite will detect and report bad addressing within the site.

Import

[Templates] [Resources] [Blocks] [Pages]

SiteLite 2.0 allows import of all kinds of components. The reason for this is twofold:

- You may build a site with contributions from many people on the Internet.
- You may reuse material from other sites or pages.

You may import from the Internet, using a http-protocoll, or you may import from a disk address, a file path.

SiteLite is built with the strategy that all components in a site should always be stored on one disk volume. All components that are located on other volumes or on the internet are imported, that is copied, to the local disk during build. A consequence of this strategy is that changes you do to the copies will be erased the next time you do an effective import, that is the next time you build. You may of course change the script to avoid this.

The strategy is chosen to support the most common situations, where a site is mainly selfcontained and 99% of the work is on your local machine. The imports you do are mainly contributions from other writers on the internet. The strategy also makes it easy to share resources and templates to keep up a standard layout and appearance within a (virtual) community.

The strategy allows a site to be distributed in many catalogs on one disk volume. This may be a good thing while testing and refining your site, but it makes it complicated to move the site to another volume, for instance for publishing. Unless you work directly on the server that will host the site, it is a good idea to keep all components of a site completely within one catalog. This does not prevent you from importing during build, and it makes it easy to move a site by simply moving the catalog.

When you plan a site, it is wise, but in most cases not necessary, to use the name standards for catalogs with templates and resources. Templates should be placed in a catalog *templates* and resources in a catalog *gfx*.

Below is a description of the import of different kinds of components and some comments and advises to their use.

Templates

Templates may be imported from:

- a complete URL (*www-ia.iof.no/~borres/gb/templates/*)
- a complete disk path (*d:\mysstdsite\templates*)

or templates may reside on a disk path relative to the site catalog (*..\mysstdsite\templates* or simply: *templates*). In this case they are not copied, but used as is.

SiteLite copies all imported templates to the the catalog **templates** which is a catalog directly below the site catalog. If the TEMPLATES= -line in the script contains a remote location (other disk volume or internet address), **all** files in that location are copied to the templates-catalog.

If, after the copying, some "standard" templates are missing, these are produced by SiteLite if needed. If the TEMPLATES= -line is missing, the necessary templates are produced by SiteLite and are put in the local *templates*-catalog as needed. This way SiteLite always have access to the basic template scheme.

If the script contains any ATEMPLATE= -lines, the same copy procedure are exercised; all templates ends up in the same templates catalog.

The disadvantage of importing is that if you do changes to the local copies of the templates, these changes are disregarded next time you build since the local copies are overwritten. The only remedy for this is to edit only the originals or change the script.

▲

Resources

By resources we understand all graphics in the templates, and the style sheet file.

Resources may be imported from

- a complete URL (*www-ia.iof.no/~borres/gb/gf x/*),
- a complete disk path (*d:\mysstdsite\gf x*)

or they may reside on a disk path relative to the site catalog (*..\mysstdsite\gf x* or simply *gf x*). In this case they are not copied, but used as is.

SiteLite makes a cop of imported resources in the catalog **gfx** which is a catalog directly below the site catalog.

If, after the copying, some "standard" resources are missing, these are produced by SiteLite. If the RESOURCES= -line is missing, all resources are produced by SiteLite and are put in the local *gfx*-catalog. This way SiteLite always have access to the basic resources.

NOTE that templates (and pages and blocks) should always refer to resources as if they were directly accessible in a catalog *gfx*. SiteLite will correct the references.

▲

Blocks

Blocks may be imported from

- a complete URL (*www-ia.iof.no/~borres/gb/ma-2d/matrix.html*),
- a complete disk path (*d:\mysstdsite\blocks\cv.html*)

or they may reside on the same disk volume as the site catalog (*..\mysstdsite\blocks\cv.html* or *blocks\cv.html*). In this case they are not copied, but used as is.

When blocks are copied, SiteLite uses the following naming strategy: SiteLite establish a local catalog called *imported*, and all imports (except templates and resources) are stored within this catalog. Subdirectories are established to match the structure the blocks are copied from. For instance the source
   www-ia.hiof.no/~borres/gb/ma-2d/matrix.html
is copied to:
   c:\mysite\imported\borres\gb\ma-2d\matrix.html
assuming that c:\mysite is the site catalog. This is not a full guarantee against name conflicts, but it makes them fairly unlikely.

SiteLite attempts to copy images which are referenced in a block. They are copied with the same naming strategy as for the block, and their addresses are calculated so it fits the page the block are pasted in.

SiteLite does not copy other files, like applets, sounds or multimedia files.

▲

Pages

Pages may be imported from

- a complete URL (*www-ia.iof.no/~borres/gb/ma-2d/page-2d.html*),
- a complete disk path (*d:\mysstdsite\pages\p-cv.html*)

or they may reside on a disk path relative to the site catalog (*..\mysstdsite\pages\p-cv.html*). In this case they are not copied.

The same naming strategy applies for pages as for blocks. Imported pages are always considered readymade, that is they are not built. The K-option (Keep) is assumed by SiteLite, and BLOCK-lines that belong to the page are ignored. This means that the page is not assembled from blocks but all other substitution and building of index and tables of content are done. Updates to the page are only done on the local copy.

SiteLite attempts to copy images which are referenced in a page. They are copied with the same naming strategy as for the page. SiteLite does not copy other files, like applets, sounds or multimedia files.

Pages with the N-option is not copied, they are addressed directly from the site.

▲

Possible Asked Questions

...how do I:

[make a new site] [make a new page] [move a page]
[make a block] [edit a block] [move a block] [freeze a page]
[change a style] [print the site] [speed up SiteLite]
[use the Internet] [use frames] [use dynamic pages]

make a new site

If you want to make a completely new site you have three options:

1. You may start SiteLite and select new from the filemenu. This produces a simple script with some basic values, that you can correct directly in SiteLite. This is probably the easiest thing to do as a first SiteLite test.
2. You may start SiteLite and start editing the blank script area.
3. You may start any editor that produces pure ascii-code, write in your script, save it on your disk, and open it in SiteLite.

You can always test your script by pressing the interpret-script button, the glasses.

It is a good idea to keep your script in the catalog you specifies in the CATALOG= - line in the script. You don't have to do this, but it helps you keep things organised, until you get a better idea.

▲

make a new page

Use SiteLite or your favourite text editor and edit in a new page-line in the script. You should also edit in a block-line immediately after the page-line. You don't have to edit the block at the moment. If you build the site, the hammer-button, and inspect the page, you will fine that it is built with some preliminary content in the block.

move a page

Use SiteLite or your favourite text editor and move the page-line and the associated block-lines (and possible based_on-lines) to the place you want, copy and paste. You don't have to move the page file.

make a block

Use SiteLite or your favourite text editor and insert a block-line after the page you want it to appear in. Unless you have a non-trivial block routing plan, the blocks will be inserted in the sequence they appear.

A non-trivial block routing involves the element <!--BLOCK TARGET=nn--> in the block and an element <!--nn--> in the template.

edit a block

You may use any editor that produces html-code to edit a block. You may use a simple text editor like Notes, you may use the edit mode in a browser like Netscape, you may use a specialised editor like PageMill. This is all up to your liking. A personal advice is to use an editor that gives you control of the details, and it is great if the editor supports html-syntaks by som ecolor coding.

It is important that you are able to control and edit the SiteLite elements which appear as (non-visible) html-comments in the html-text.

move a block

Use SiteLite or your favourite text editor and move the block-line after the page you want it to appear in. You don't have to move the block-file.

A well planned division of the material into blocks, makes it easy to reorganise your material at any time.

freeze a page

This may mean at least two things:

- You don't want the main contents of the page to change. The only things that should be changed are references, tables of contents, index etc. In that case you mark the page with option 'K' in the page line. The page will appear in tables of contents and it will contribute to indexes etc. and general replaces will occur. If the page contains no elements that will influence itself or the other pages this is a straight way to include an "alien" page.

  You may achieve the same effect by deleting the type specification in the page-line. The page is then, so to speak, its own template.

- You don't want anything to happen to the page at all. In that case you may simply comment it out of the script by placing // in the start of the page-line. (If the page has blocks, these are ignored automatically). The page will not appear on tables of contents etc. after the next build.

change a style

Use your favourite text editor and open the def.css-file, edit the style in question and save the file. You don't have to rebuild the page to see the effects of this change, simply ask the browser to reload the actual page.

It is beyond this text to explain Cascading Style Sheets in any level of detail.

A word of advice: Test your styles with all the browsers you expect your users to use. Cascading Style Sheets is (still) a fairly shaky concept and many details are implemented in differently ways, or simply ignored, in different browsers.

print the site

A website is primarily meant to be read from the screen, and printing is no key issue. Still experience shows that many users print sites they find interesting, and your site will of course fall into this category. The amount of paper used easily becomes enormous if all your users should print all your pages.

SiteLite lets you assemble selected pages into what is referred to as "print-pages". This makes it possible to offer the reader a print of essentials. It also makes it possible for you to assemble the whole site or parts of the site for processing in an other program, typically a word processor that can read html.

What you do is to insert a page-line with type PP, and an associated bloc-line. Build the site and edit the <!--PRINT...--> element in the bloc. If you want many selective print-pages, you repeat this operation.

speed up SiteLite

SiteLite works fairly efficient for normal computers and "not-too-large" sites. You will however experience that there are some operations that may slow down sitebuilding. There are a lot of things you may do to gain speed:

- Turn off Internet-access. If you use any files (blocks, pages, templates, resources) that are located somewhere else on the Internet, SiteLite will always try to get the latest version of these files. On the other hand, SiteLite always make and keep local copies of such files on your disk. So if you don't expect any updates on your Internet-based materiel, you may "turn off the Internet".
- If you are concentrated on getting one, or a few, pages right, you should use the "Build selected"-button or menu.
- You may enclose a set of lines in the script by /* and */ lines to mask out part of the script temporarily.
- You may introduce // in the start of a single line to mask out this line temporarily.
- There are a few operations that may take some time, depending on your specifications. The important ones are available for temporary exclusion in the menu "Properties-When building".

use the Internet

SiteLite accepts templates, resources, and pages from an URL.

SiteLite will always make local copies of the things it is instructed to get from the net. Templates are copied to the "template"-catalog and resources are copied to the "gfx"-catalog. If necessary these catalogs are established.

Blocks and pages are copied to catalogs that are placed locally to the sitecatalog, in a catalog called "imported" with the same structure as they have on the original server.

If an imported block or page contains references to images, these images are copied according to the same naming strategy and the addresses are recalculated according to the page where the images is inserted.

All access to the Internet is via the http-protocol. SiteLite prompts user before it accepts redirections and

it gives up if the page requires a password. In technical terms SiteLite works smooth if it gets a 200-response to its request.

▲

use frames

SiteLite lends itself easily to support a frame based solution. The key is to introduce the frame describing pages as pages with level 0. SiteLite comes with a set of templates that describe a simple frame-solution. The easiest way to get control of frames is to start SiteLite and make a new site with File-New menu. Select frame based in the dialog box that follows. You can inspect the files and templates and modify them at will.

▲

use dynamic pages

SiteLite has no explicit support for dynamic pages, in any interpretation of the word "dynamic". SiteLite does however not bother about the content of the pages, and it should be rather straightforward to support a combination of a fixed structure in SiteLite with page dynamics in any number of frames.

▲

Download and updates

[Downloads] [Updates] [Converting]

Downloads

The program ~256kb	Download SiteLite20.exe and run.
This documentation ~230kb	Download allpages.pdf. This documentation as a pdf-file. You can download Acrobate Reader free from Adobe if you dont have it installed. Since Microsoft Explorer seems to have doubts about pdf, use **right** button and "Save as".
Demo sites ~256kb	Download a package of demo sites. Unzip, browse, inspect and change. You may have a look at them here: - Shakespeare - Illustrate - Catalog - Frames
Contributions	If you have interesting sites an/or templates and/or resources, let me know and I will leave a reference or a zipped copy here for download.

Neither me nor Ostfold College take any responsibility for any damage caused by the program or the use of it. SiteLite is downloaded and used completely at your own risk. Reasonable measures has been taken to ensure that what you download is what it is meant to be.

Updates

Date	What is new
2002-06-19	Bugfix

	Indexes are sorted independant of upper/lower case. Multiple collections are allowed on one page.
2002-02-27	Bugfix SiteLite has in some special cases generated erronous image refs in printpages (aggregated pages). The problem occurred when two images on the same page had names of type: undef.gif and def.gif, that is one files name was the same as the end of another files name. This is now fixed.
2002-01-07	Bugfix SiteLite has generated erronous list of children in response to the SINGLETOC element. This is now fixed.
2001-08-23	Bugfix/Enhancement SiteLite has had problems with "mixed" imageref's. That is some refs from block and some prepared as from from page. These problems are now removed.
2001-07-25	Bugfix Patching with STRUCTURE removes attributes, and does thus perform as expected.
2001-06-11	Bugfix Printpages has not referenced two ocurrances of same image in one page correctly. This is now corrected.
2001-05-09	Small format adjustment SiteLites stamp on a site is reduced from "Built by SiteLite 20" to "SiteLite 20".
2001-04-26	Extended feature: The element **PATCH** has proved usefull and has been expanded with one option. It is now possible to write: <!--PATCH FILENAME="file" STRUCTURE="ON"/-->. The attribute STRUCTURE instructs SiteLite to collect from file according to the following priority: If a BLOCK-element exists, its content is chosen, if a BODY element its content is chosen, if a HTML-element exists its content is chosen. Else the whole file is chosen (same as with no STRUCTURE attribute or STRUCTURE ="OFF")
2001-04-06	Direct links: SiteLite will now allow relative filenames combined with build-option 'N'. This means that it is possible to include an "alien" page in tables of contents. Since the referenced page is not touched by SiteLite, user is responsible for making a "return-link" if so needed. This possibility has only been available for absolute URL's.
2001-01-30	Changed build strategy: SiteLite will no parse the script prior to all build (and check) operations. This will reduce the performance slightly, but will assure that changes in referenced files i REPLACE commands are up to date. This simplified logic will also assure that collections and indexes are built fresh for each pagebuild.
2001-01-30	Safer input: More robust parsing of script. No changes in allowed formats, but a better strategy for eliminating obvious meaningless whitespace.
2001-01-08	Bugfix: The sequence of page-parameters in the helpbox was wrong. Author comes before Message.
2000-12-23	New Feature: The replace tool (menu: **Tools-> Replace**) now takes a file as input text. Filename is interpreted as relative to sitecatalog, full filepath or url

2000-12-22	New Feature: New element introduced. PATCH. Collects a file "as-is" and places its content as a replacement for the tag. My name:<!--PATCH FILENAME="myname.txt"/-->at all times will be: My name:Børre Stenseth at all times Filename is interpreted as relative to sitecatalog, full filepath or url
2000-12-18	Bugfix: Unknown pagename in a BASED_ON line in the script caused SiteLite to crash. Such is no longer the case. The situation is reported as error.
2000-12-15	New feature: Element TRAIL introduced. Makes entries of this type: home > chapter > page Use: <!--TRAIL--><!--/TRAIL-->
2000-12-11	Bug fix: Possible loop-forever situation in replace command from the script is corrected.
2000-12-08	Increased flexibility in the script: Since SiteLite runs on Windows the native fileseparator is backslash (\). SiteLite 2.0 has been very strict on this point and has tried to interpret all locations in a script with a forward slash (/) as an URL. It si now possible to work with forward slashes also in filepaths. Use menu: **Attributes** > **When building**, and check the attribute: Allow forward slashes. A consequence of using this option is that all URL's must be stated explicitely, that is starting with http://.
2000-11-07	Bug fix: Address calculation for images in page resulting from "Make a printable Page" corrected.
2000-11-06	Bug fix: Type in pagerelevant table of contents corrected.
2000-10-16	New Feature: Two new elements are introduced: PAGENO and PAGECOUNT. Usefull when making "slides"-series with SiteLite. See bottom of page Elements.
2000-10-16	BugFix: Indexes was reported wrong in certain combinations of building multiple sites in sequence. Corrected.
2000-09-26	BugFix: Printpage corrected.
2000-09-20	BugFix: Building is corrected so that Indexes and Collections are not accumulating from one build to another. You must build all to make correct (complete) indexes and collections.
2000-09-20	BugFix: Correction style used by SiteLite to stamp a site.
2000-09-15	The possibility to use c-type comments, /* */,to mask multiple lines is removed due to loss of data on save.
2000-09-03	SiteLite 2.0 launched.

Converting

SiteLite 2.0 offers an automatic upgrade of your site from version 1.5. When SiteLite reads a script without the line: VERSION=2.0, it assumes that the script is a description of a site built with version 1.5. SiteLite offers to convert the site. In this case SiteLite inspects and attempts to update all templates and all blocks in the site, in addition to the script itself. **It is a good idea** to make a backup copy of the site before you allow an automatic update.

SiteLite does not do anything with the style sheet file during update. You should in particular consider copying the section below into your style sheet:

```
/* == styles hardcoded by SITELITE =====*/

/* Page relevant toc (left bar) */
.clevel0,.clevel1,.clevel2,
.clevel3,.clevel4,.clevel5,
.clevel6,.clevel7,.clevel8,
.clevel9,.clevel10
{
font-size: small;
font-weight:200;
background-color:transparent;
text-decoration:none;
line-height:120%;
}

.clevel0{ margin-left:0px;}
.clevel1{ margin-left:7px;
          font-weight:600;
          line-height:150%;}
.clevel2{ margin-left:14px;}
.clevel3{ margin-left:21px;}
.clevel4{ margin-left:28px;}
.clevel5{ margin-left:35px;}
.clevel6{ margin-left:42px;}
.clevel7{ margin-left:49px;}
.clevel8{ margin-left:49px;}
.clevel9{ margin-left:49px;}
.clevel10{ margin-left:49px;}
/*-active-*/
.aclevel0,.aclevel1,.aclevel2,
.aclevel3,.aclevel4,.aclevel5,
.aclevel6,.aclevel7,.aclevel8,
.aclevel9,.aclevel10
{
font-size: small;
font-weight:200;
background-color:transparent;
text-decoration:none;
line-height:120%;
color:#999999;
}

.aclevel0{ margin-left:0px;}
.aclevel1{ margin-left:7px;
           font-weight:600;
           line-height:150%;}
.aclevel2{ margin-left:14px;}
.aclevel3{ margin-left:21px;}
.aclevel4{ margin-left:28px;}
.aclevel5{ margin-left:35px;}
.aclevel6{ margin-left:42px;}
.aclevel7{ margin-left:49px;}
.aclevel8{ margin-left:49px;}
.aclevel9{ margin-left:49px;}
.aclevel10{ margin-left:49px;}


/*  on sitemap pages */
.tlevel0,.tlevel1,.tlevel2,
.tlevel3,.tlevel4,.tlevel5,
.tlevel6,.tlevel7,.tlevel8,
.tlevel9,.tlevel10
{
font-size: small;
font-weight:200;
background-color:transparent;
text-decoration:none;
line-height:120%;
}

.tlevel0{ margin-left:0px;}
```

```
.tlevel1{ margin-left:7px;
          font-weight:600;
          line-height:150%;}
.tlevel2{ margin-left:14px;}
.tlevel3{ margin-left:21px;}
.tlevel4{ margin-left:28px;}
.tlevel5{ margin-left:35px;}
.tlevel6{ margin-left:42px;}
.tlevel7{ margin-left:49px;}
.tlevel8{ margin-left:49px;}
.tlevel9{ margin-left:49px;}
.tlevel10{ margin-left:49px;}

/* local toc*/
.localtoc{ font-size:smallest;}

/**/
.basedon{ font-size:small;
          font-style: italic;
          margin-left:7px;}

/* on index pages */
.indexkeyword{  font-size:small;
                font-weight: bold;
                line-height:120%;}
.indexentry{    font-size:small;
                font-weight: normal;
                 margin-left:7px;}

/* ====== END OF styles hardcoded by SITELITE =====*/
/* heading for */
.basedhead{
  font-family: "Geneva", "Verdana", "Arial", sans-serif;
  font-size:small;
  font-weight: normal;
  color:#000077;
  }

/* bottom of pages, author, date, message etc. */
.mainfooter{
  font-family: "Geneva", "Verdana", "Arial", sans-serif;
  font-size:8pt;
  margin-left:140px;
  color:#AAAAAA;
}

/* bottom without offset. */
.leftfooter{
  font-family: "Geneva", "Verdana", "Arial", sans-serif;
  font-size:8pt;
  margin-left:10px;
  color:#AAAAAA;
}
```

References

SiteLite has so far had a small but very competent and active user community. Special thanks to collegae Gunnar Misund and student Laila Kynningsrud who have contributed to the development of SiteLite with well documented error reports and creative suggestions for new functionality.

B Stenseth
generated: June 19, 2002

SiteLite 2.0