pagegen

every page is a file

Pagegen is a static web site generator, meaning it creates web sites from flat text files and directories. Great for managing sites from the command line.

There are many other static site generators (e.g. Webgen http://webgen.rubyforge.org/, WML http://thewml.org and Webby http://webby.rubyforge.org/). Pagegen tries to be different by making assumptions about the generated HTML and keeping the content files as simple as possible. This should mean less to learn and more time to focus on the content — which is king, after all.

Download Pagegen

tar.gz deb

v0.8.6 released 2010-11-19

Design goals

Pagegen is designed to allow focus on writing content and easily manipulating it using the GNU/Linux command line. Content is contained in directories and flat text files. Pagegen uses Pextile http://pagegen.phnd.net/user-manual/pextile-markup-reference markup (a variation of Textile http://en.wikipedia.org/wiki/Textile_(markup_language)). Sites are created by making folders and adding content files, then generating HTML.

Features

- Content is stored in flat text files, easily manipulated using mv, cp, vi, grep, sed and all your other fav GNU/Linux tools
- Concisely markup/format content using Pextile (variation of Textile)
- The content can be stored in infinitely nested directories, for which Pagegen will create navigational menus and breadcrumbs
- Specification of menu item order
- Web friendly URLs
- The web site only needs a regular web server to run (no script or database requirements)
- Content files may be output from any script or executable that is run at generation time
- Provides content variables and incremental variables to aid content maintenance
- Generate sitemap.xml

pagegen every page is a file

Quick start

This page shows how to install Pagegen and start managing content, for in depth documentation please see the user manual http://pagegen.phnd.net/user-manual.

Step 1. Setup

1. Download http://pagegen.phnd.net/download/
2. Install http://pagegen.phnd.net/user-manual/installation

Step 2. Write some content

1. Run `pagegen new my_site` to setup the required directories and files for the site
2. Add some text to the **[pagegen directory]/mysite/content/default**: `echo "This is the front page:)" > default`
3. Create a new directory called **My new page** in the **[pagegen directory]/my_site/content** directory `mkdir "My new page"`
4. Create a new file in the **My new page** directory, called **default**. Fill it with some text: `echo "This here is my page" > "My new page/default"`. Every time a directory is added a **default** file must be created as well.
5. Create another file in the **My new page** directory called **Another page**, again adding some text to it: `echo "Starting to get this?" > "My new page/Another page"`

Step 3. Customize site layout

Pagegen adds layout and navigation elements to each page when they are processed. To change the site look and feel, play with the following files:

- [site directory]/include/css/pagegen.css
- [site directory]/templates/header
- [site directory]/templates/footer

Step 4. Generate site

Run `pagegen gen my_site` to generate the site, see **[Pagegen directory]/my_site/site/live** for the freshly generated HTML files. To take the site live, simply copy the **live** directory to your web server. Enjoy!

pagegen every page is a file

User manual

Want to start playing straight away? Please see the quick start http://pagegen.phnd.net/quick-start.

pagegen every page is a file

Installation

Pagegen can be downloaded as a simple tar file or as a deb package.

Installing deb package

1. Download http://pagegen.phnd.net/download e.g. `wget http://pagegen.phnd.net/include/downloads/pagegen_0.8.6_all.deb`
2. Install the package, e.g. `dpkg -i pagegen_0.8.6_all.deb`
3. Setup Pagegen sites directory #setup_pagegen_sites_directory

Installing tar package

1. Download http://pagegen.phnd.net/download e.g. `wget http://pagegen.phnd.net/include/downloads/pagegen_0.8.6.tar.gz`
2. Extract tar archive `tar -zxf pagegen_0.8.6.tar.gz`
3. Ensure all files in the `pagegen_0.8.6` are executable e.g. `chmod 777 *`
4. Setup Pagegen sites directory #setup_pagegen_sites_directory

Setup Pagegen sites directory

Pagegen requires a sites directory to store the sites it manages (content, templates, generated HTML etc). It assumes the sites directory is located in **~/pagegen**, if this directory does not exist it must be created e.g. `mkdir ~/pagegen`.

Pagegen is now ready to roll.

> If you want to specify another location for this directory create **~/.pagegen.conf** and add `PAGEGENDIR=[path to your Pagegen sites directory]`.

Further configuration of the installation

> This information is for customizing the installation further and is probably not relevant to most users.

If **/etc/pagegen.conf** exists it will be used, it may contain the following settings.

Name	Value	Description
PAGEGEN_LIB_DIR	path_to_executables	The **pagegen** executable may be moved to any location (e.g. `/usr/bin`), as long as this setting points to the directory containing the other executables

Managing content

Content is stored as regular files and is organized by putting the files in directories.

Content structure

> All directories must contain a file called **default**

All content is stored in the **content directory** (`[Pagegen directory]/[site directory]/content`).

To and organize content pages add new text files and directories to the **content** directory. The name of the file/directory will be the page title, so use human readable names. All directories must contain a file called **default**.

> Pagegen will generate nice URLs from the file and directory names. For instance the directory and file 'About us/Contact' will get the URL 'about_us/contact'.

> The front page is **content/default**

Page title and URL

When Pagegen generates pages the page title becomes the file or directory name (without the sort prefix **XXX_** if present, see below), this means the names used is important, if you want a title 'Apples are nice' the file must be called 'Apples are nice' (optionally 'XXX_Apples are nice').

Sorting content in menus

Pagegen sorts the items in the menus by listing them alphabetically, to override this default sorting the files and directories can be prefixed by **XXX_** (3 digits and an underscore). The sort prefix will be striped from the title and URL

The **XXX_** prefix for all file and directory names allows content to be sorted by changing the prefix. E.g. To sort 3 items, set the prefix to 001_, 002_ and 003_.

> A good practice is to use increments of 10 when setting the sort prefix (010_, 020_, 030_ etc.). This means an item can be added in the sequence with for instance 011_ without having to rename all items.

Hiding pages from menu listing

If a directory or file is hidden (just prefix with a '.' (period), as you normally would on Linux/Unix, it will not be shown in the menus.

Site configuration

There are two configuration files that affect the way Pagegen generates files, they are:

- **[Pagegen directory]/bin/pagegen.conf**
- **[Site directory]/site.conf**

The format is **Key=Value**, and **#** to comment. The files accept the same settings. To set options globally use **pagegen.conf** and for site specific use **site.conf**. If settings are set in both files the setting from the **site.conf** will be used. Currently the following settings are available.

Setting name	Setting values	Description
add_accessibility_bookmarks	1 or 0	If **1** then bookmarks are added to the main menu and content, to facilitate accessibility
create_sitemap_xml	1 or 0	If **1** a sitemap.xml will be created and added to the web root folder. Must also set **sitemap_page_base_url**
filter_svn_dirs	1 or 0	If **1** then .svn directories will not be included in output. Note that if used with the **symlink_site_include_dir** setting any .svn directories in the include directory will not be removed (that would mean loosing any revision control that was put on the include directory)
omit_crumb_trail	1 or 0	If **1** then the crumb trail is not included in output
omit_home_from_main_menu	1 or 0	If **1** then the Home link is not included in the main menu
omit_last_changed	1 or 0	If **1** then the last changed string is not included in output
omit_sub_menu	1 or 0	If **1** the section menu is not included in output
pextile_print_urls	1 or 0	If **1** all links in the content will have a span tag with class **pg_print_url**. Using CSS the URL can be visible only when printing
place_navigation_after_content	1 or 0	If **1** then the navigational elements are placed after the content in the HTML file, useful for creating accessible pages
site_base_url	Site URL	Used to generate proper links for the sitemap.xml and for print urls
symlink_site_include_dir	1 or 0	If **1** then instead of copying the **[site directory]/include** directory to the output folder, a symlink is made instead

pagegen every page is a file

Generate site

Run **pagegen gen [site name]** (use **pagegen list** to see which sites are available) the generated HTML files will be in the **[Pagegen directory]/[site directory]/site/live**, simply copy this directory to your web root to go live:)

Pagegen file structure

Each generated page has the following structure.

Name	Source	Description
Header	[site directory]/templates/header	Start of HTML document
Main menu	Generated HTML	List of links to pages found in all immediate child directories from the **content** directory (one directory below)
Sub menu	Generated HTML	For each child directory of **content** show links from the current page to the top child directory
Crumb trail	Generated HTML	Crumb trail for each page
Last changed date	Generated HTML	Show modification time of content files
Content files	[site directory]/content/*	The actual site content
Footer	[site directory]/templates/footer	End of HTML document

Site navigation

When the site is generated each page will contain a main menu, sub menu and crumb trail (unless the sub menu and/or crumb trail are disabled in the **site.conf**).

The generated menus can be tweaked using CSS and/or Javascript and allow for some of standard navigational layouts. All examples below refer to the following directory structure.

> A directory must always contain a file called default.

> All items, except the default files, can be prefixed with 3 digits and an underscore, e.g. **010_** to specify order in menus. All menu items are sorted alphabetically, so the 3 digit prefix and underscore allows items to be sorted as the user wants.

Example site(directory) structure

Level 1	Level 2	Level 3	Level 4
Directory 1			
	default		
Directory 2			
	default		
	File 2.1		
	File 2.2		
Directory 3			
	default		
	Directory 3.1		
		default	
		File 3.1.1	
		Directory 3.1.2	
			default
			File 3.1.2.1
		Directory 3.1.3	
			default
			File 3.1.3.1

Main menu

The main menu consists of all directories found in level 1 and all items found in level 2. For each page the appropriate section and menu item is indicated by a class attribute.

The main menu that would be generated page **File 2.1** in the example above would be as follows.

```
1  <div id="mainmenu">
2    <ul>
3      <li class="section"><a href="/directory_1/">Directory 1</a></li>
4      <li class="section_selected"><a href="/directory_2/">Directory 2</a>
5        <ul>
6          <li><a class="item_selected" href="/directory_2/file_2.1">File 2.1</a></li>
7          <li><a href="/directory_2/file_2.2">File 2.2</a></li>
8        </ul>
9      </li>
10     <li class="section"><a href="/directory_3/">Directory 3</a>
11       <ul>
12         <li><a href="/directory_3/directory_3.1/">Directory 3.1</a></li>
13       </ul>
14     </li>
15   </ul>
16 </div>
```

Notice the href attribute for the **Directory links**, they assume that the default index file on the web server is called **default**. See Web server setup http://pagegen.phnd.net/user-manual/web-server-setup for more information.

Please note that class attribute of the <li> tag on line 4 is **section_selected** whilst for the other items at this level it is **section**. Similarly on line 6 the class attribute is **item_selected**. The **section_selected** class will always be set on the appropriate tag, whilst the **item_selected** will only be set if the page is in either level 1 or 2 (see example above).

Sub menu

The sub menu lists items from level 1, but only listing the contents of the directories required to display the current item.

The sub menu that would be generated for page **File 3.1.2.1** in the example above would be as follows.

```
1  <ul id="submenu">
2    <li>
3      <a href="/directory_3/">Directory 3</a>
4      <ul>
5        <li>
6          <a href="/directory_3/directory_3.1/">Directory 3.1</a>
7          <ul>
8            <li>
9              <a href="/directory_3/directory_3.1/file_3.1.1">File 3.1.1</a>
10           </li>
11           <li>
12             <a href="/directory_3/directory_3.1/directory_3.1.2/">Directory 3.1.2</a>
13             <ul>
14               <li>
15                 <a class="item_selected" href="/directory_3/directory_3.1/directory_3.1.2/file_3.1.2.1">Fi
16               </li>
17             </ul>
18           </li>
19           <li>
20             <a href="/directory_3/directory_3.1/directory_3.1.3/">Directory 3.1.3</a>
21           </li>
22         </ul>
23       </li>
24     </ul>
25   </li>
26 </ul>
```

Please note that on line 15 the <a> tag class attribute is **item_selected** and that Directory 3.1.3 has not been expanded.

Crumb trail

The crumb trail lists the "directory path" to the current item. The crumb trail to **File 2.2** would be as follows.

```
1 <div id="crumbtrail">
2   <a href="/">Home</a> >
3   <a href="/directory_2/">Directory 2</a> >
4   <a href="/directory_2/file_2.2">File 2.2</a>
5 </div>
```

Customize site design

HTML templates

There are two HTML template files, **[site directory]/templates/header** and **[site directory]/templates/footer**, upon generation of the site the header is added before the content, and the footer naturally after.

If the header and footer are executable their output will be used instead of the actual file contents. Executable template files have any variables set in **[site directory]/content_variables/*** , the **page_title** and/or any variables set in the content page header available for use (see Dynamic content http://pagegen.phnd.net/user-manual/dynamic-content for more information).

In addition to the template files Pagegen adds a ** ** tag around the page content. The tag's **id** attribute will be **frontpage_content** for the front page and for all other pages the **id** will be **content**.

CSS, Javascript and images

When generating the site all contents in the **[site directory]/include** directory are either copied(default) or symlinked to the output folder (see Site configuration http://pagegen.phnd.net/user-manual/site-configuration). Add CSS, Javascript and images (or anything else) to this directory to have them included with the generated site.

Dynamic content

Though the final site will be static HTML, the content may be dynamically created when it is generated.

Executable content pages

When generating content Pagegen will first check if the content file is executable, if it is the file will be run and the output used, if not the file will be parsed as pextile markup. I.e. the content file could be a shell script that produces HTML.

Incremental variables

For non-executable content files it is possible to add incremental variables. An incremental variable will increment by 1 each time it is used. To use enter **%[single letter]%** anywhere in the content file.

The following content file..

```
Hello I'm an incremental variable %a%, and I'm %b%
I started as 1, but now I've become %a%
```

..will become:

```
Hello I'm an incremental variable 1, and I'm 1
I started as 1, but now I've become 2
```

Content variables

On site generation all files in **[site directory]/content_variables** will be parsed and the variables made available for use in all content pages that are not executable. The format is **key=value**.

For instance if the file **[site directory]/content_variables/company_names** includes a line consisting of **my_company=My Very Own Inc.** then all occurrences of **$my_company** in the content files will be replaced by **My Very Own Inc.**.

Page variables

Content pages that are not scripts/executables may contain an optional header part in order to set certain attributes. The header section consists of key/values followed by a line containing only **=====** (five equals signs).

The key/values can be used by the header, footer or in the content page itself by specifying **$[key name]**.

```
var1=Hello
var2=World
=====
${var1} ${var2}!!
```

When the page is generated it will become:

```
Hello World!!
```

The variables defined in the page header are available as environment variables to the header and footer templates. For the template to make use of the variable it must be executable and setup to do something with the variable.

Page title

Default is to set the page title to the same as the file/directory name, this may be overridden by specifying a value for **page_title** in the page header section, e.g.:

```
page_title=Override the default title with this one:)
=====
h2 Normal content
```

The **$page_title** variable is also available as an environment variable for executable header and footer templates. If the template

is executable it could be used to e.g. set the page title tag to the **$page_title** variable.

pagegen every page is a file

Pextile markup reference

Pagegen's markup language, Pextile, is pretty much a rip off/subset of Textile, a brilliant markup created by Dean Allen.

> **^** denotes the start of line and **$** denotes the end of line

> To escape parsing of content file start with **^/*$** and end with **^*/$**. All lines between the start and end escape codes will be outputted verbatim

Acronym
AGA((A good acronym)) → <acronym title="A good acronym">AGA</acronym>

Emphasise (italics)
__text__ → text

Footnote reference (X = number)
[[X]] → X

Footnote (X = number)
^fnX Text$ → X. Text

Heading (h1-6)
^h1 Text$ → <h1>Text</h1>

Horizontal rule
^---$ → <hr />

Image (after first exclamation can use <, > or - to align)
!img_src Image alternative text! →

Link (anchor point on same page)
"The apple section":"name:apples" → The apple section

Link (normal)
"This is a link":"http://phnd.net" → This is a link

Monospace
@@text@@ → <tt>text</tt>

Notice
^notice Text$ → Text

Numbered list
^# Numberd list item$ (nest lists by prefixing additional '#' (hash)) → Numbred list item

Preformatted text
^> Text$ → <pre>Text</pre>

Quote
^quote Text$ → Text

Strikethrough
--text-- → ~~text~~

Strong (bold)
text → text

Table header row (use %<, %> or %- to align cell content)
^%< Cell1 % Cell2 %$ → <table><tr><th class="pg_th_left">Cell1</th><th>Cell2</th></tr></table>

Table row (use |<, |> or |- to align cell content)
^| Cell1 |> Cell2 |$ → <table><tr><td>Cell1</td><td class="pg_td_right">Cell2</td></tr></table>

Unordered list (nest lists by prefixing additional '*' (asterix))
^* Unordered list item$ → Unordered list item

Warning
^warning Text$ → Text

Web server setup

After Pagegen has generated the site, all necessary files are found in the **[site directory]/site/live** directory. To publish, this directory must be copied to the web server.

> All links generated by Pagegen are relative and assume the site is located directly on the web root folder (i.e. **http://mysite.com/pagegen files**). If this is no good, consider adding an appropriate <base> tag ⇨ http://www.w3schools.com/TAGS/tag_base.asp to the **[site directory]/templates/header** file.

> If using Apache the .htaccess file can be saved in the **directory/templates** directory, on generation it will be detected and copied to the web root folder. This allows both config and content to be saved locally and easily version controlled.

Force correct mime type

To provide SEO friendly URLs Pagegen does not add file extensions to the files it generates, therefore it is necessary for the web server to tell the browsers what content type the file is.

To do this for Apache add the following to your configuration (e.g. .htaccess).

```
# Force text/html for all files
<FilesMatch "\.*">
  ForceType text/html
</FilesMatch>

# Use default type for files that have an extension
<FilesMatch "\.[a-zA-Z0-9]{1,4}$">
  ForceType none
</FilesMatch>
```

The above Apache configuration tells Apache to set the HTTP **Content-Type** header to **text/html** for all files, exept if the file has an extension, in wich case do not force the type.

Serve default file if no file specified in request

Content files for directories are named **default**, for these to be returned when the user requests a directory the server must be configured.

For Apache this means setting the **DirectoryIndex** in the appropriate configuration file (e.g. .htaccess).

```
DirectoryIndex default
```

Custom error pages

Pagegen automatically creates default error pages that the web server can be configured to show (if, for instance, a file was not found). The templates for the error pages can be found in the **[site directory]/templates** directory. The defaults templates are:

- **error_page_400** – Bad request
- **error_page_403** – Access forbidden
- **error_page_404** – Page not found
- **error_page_500** – Internal server error

> If other error pages are required, put them in the template directory and prefixing the file name with **error_page_**

The error pages are placed in **[site directory]/site/live**. The web server must be configured to show them if an error occurs.

For Apache the following will specify which error page to show. Add the following to the appropriate configuration file (e.g. .htaccess).

```
ErrorDocument 400 /error_page_400
ErrorDocument 403 /error_page_403
ErrorDocument 404 /error_page_404
ErrorDocument 500 /error_page_500
```

sitemap.xml and robots.txt

Pagegen can generate a sitemap.xml http://www.sitemaps.org/ file and handle a robots.txt http://www.robotstxt.org/ file. Both files are for use by web crawlers/spiders and are useful for SEO.

sitemap.xml

Through configuration settings it is possible to setup generation of a sitemap.xml file. The **sitemap.xml** will be placed in the web root folder.

Site settings

The following settings must be configured to turn on sitemap generation, set these either globally in **pagegen.conf** or per site in **site.conf**.

```
# This turns on sitemap generation
create_sitemap_xml=1

# This is required for the sitemap URLs to make sense
site_base_url=http://mysite.com
```

Page settings

In addition to the site settings each content page may specify certain variables to affect its listing in the sitemap.xml. These are set in the page header.

Variable name	Values	Description
omit_page_from_sitemap	1	If set the page will not be listed in the sitemap
page_modified_date	YYYY-MM-DD	Specifies the date the content page was modified, default will use the actual date of the file, this variable is useful for overriding, for instance if the content file is a script it could set the date to the date the script ran, instead of the last time the script file was modified
sitemap_page_change_freq	always, hourly, daily, weekly, monthly, yearly or never	How often the page is likely to change
sitemap_page_priority	0.0 – 1.0	The priority of this page relative to other pages on the site. Default is 0.5

Example page header

```
page_modified_date=2009-05-11
sitemap_page_priority=0.2
=====
h2 This page is just for show
```

robots.txt

Simply create a robots.txt file (see robotstxt.org http://www.robotstxt.org/) and place it in the **directory/templates** directory, on generation the file will be detected and copied to the web root folder, which is where it belongs.

Examples

This section provides examples for using Pagegen to manage a web site.

Content formating

Pextile markup

Pagegen uses Pextile for marking up content, it is a variant of Textile please see the Pagegen Pextile reference http://pagegen.phnd.net/user-manual/pextile-markup-reference, the following is an example of content source file and the HTML that is generated from it.

Pextile is based on Textile, according to Wikipedia ⇨ http://en.wikipedia.org/wiki/Textile_(markup_language) "Textile is a lightweight markup language originally developed by Dean Allen. Textile converts its marked-up text input to valid, well-formed XHTML".

Pagegen content file

```
h2 This is the main heading
A paragraph which is just some text really. But can include "links":"http://google.com".

# A list item
# Another list item
```

Pagegen generated HTML

```
<h2>This is the main heading</h2>
<p>A paragraph which is just some text really. But can include <a href="http://google.com">links</a>.</p>

<ol>
<li>A list item</li>
<li>Another list item</li>
</ol>
```

Content variables

Content variables can be used to ensure uniformity across the site and ease maintenance of text snippets that occur more than once. After defining a content variable it can be referenced in all non-executable content files and on generation will be replaced by it's value.

If needed to repeat the same snippet of text several places across the site content variables can save a lot of work if that text at some point needs to be updated.

Suppose we are writing tutorials for a graphics editor and want to start each tutorial with instructions on how to open a file. Instead of repeating the instructions in each file, we could define a content variable and reference it from each tutorial. If the procedure needed to be changed, for instance a new version of the graphics editor is released, we would only need to update the content variable.

1. Create a file in **[site directory]/content_variables** called, for example, **procedures**
2. Add the following to the **procedures** file (all on one line):
   ```
   open_file_procedure='<ol><li>Click File > Open</li><li>Find file and double click it</li></ol>'
   ```
3. Add the text **$open_file_procedure** to a content file

When the site is generated the variable reference in the content file will be replaced with the variable value.

We can add as many files to **[site directory]/content_variables** directory's as we want and each file may contain as many content variables as we like.

Custom meta description tag

This example describes how to set a meta description tag per page.

First create a new page as follows.

```
page_description=This is the description
=====
h2 My page
Some content
```

When the site is generated Pagegen will first load all variables found in the content page headers, in this case it will find one called page_description. These variables will be made available to the header executable which uses it to create the HTML title tag.

> This example uses a bash script, but any executable that can read environment variables can be used to generate content.

The following header will only print the description meta tag if the page_description variable is defined for the current page. Make the header template executable `chmod +x [site dir]/templates/header`.

```
#!/bin/bash
echo "<html>"
echo "<head>"

if ! [ "$page_description" = "" ]; then
  echo "<meta name="description" content="${page_description}"/>"
fi

echo "</head>"
echo "<body>"
```

Custom page title tag

If the **[site directory]/templates** files are executable, they will be run when Pagegen generates the site and their output used.

A good use of this is to setup a header that can set the HTML title tag for each page, which is good for search engines. Do this by setting the **[site directory]/templates/header** to the following.

```
#!/bin/bash
echo "<html>"
echo "<head>"
echo "<title>${title}</title>"
echo "<meta name="description" content="${page_description}"/>"
echo "</head>"
echo "<body>"
```

Make the header template executable `chmod +x [site dir]/templates/header`. When the site is generated the page title tag will contain the title of the page. This works because Pagegen populates the `$page_title` and makes it available for the header template.

Dynamic content page

If a page is executable Pagegen will execute the file and use the output in the generated file. See **man chmod** for how to set files executable.

The output of the executable will not be filtered by Pagegen, so it is the executables responsibility to output valid HTML.

The following is a simple hello world example, in **[site directory]/content** create a new file called, for instance, **000_Hello world**, and make it executable (e.g. **chmod +x "000_Hello world"**.

```
#!/bin/bash
echo "<h1>Hello world:)</h1>"
```

After generating the site, the page **[site url]/hello_world** will contain a Hello world title.

Dynamic templates

If the **[site directory]/templates** files are executable, they will be run when Pagegen generates the site and their output used.

A good use of this is to setup a footer which has a copyright notice showing the current year. Do this by setting the **[site directory]/templates/footer** to the following.

```
#!/bin/bash
echo "<div id="copyright">Copyright © $(date +%Y)</div>"
echo "</body>"
echo "</html>"
```

Make the footer template executable `chmod +x [site dir]/templates/footer`

When the site is generated each page will have a footer with the current year.

> This example uses a bash script, but any executable can be used to generate content.

Images

1. Copy the image to the **[site directory]/include/images** directory
2. Add the following markup to the content page, where you want the image **!/include/images/[image name] Image alternate text!**.

   You can float the image left or right using **<** or **>** immediately after the first exclamation mark, e.g. to left align: **!</include/images/[image name] Image alternate text!**.

Incremental variables

Incremental variables are variables that increment by one each time they are referenced. Consider the following example content file:

```
I'm an incremental variable: %a%
Here I am again: %a%
And finally: %a%
```

The generated page will be:

```
I'm a incremental variable: 1
Here I am again: 2
And finally: 3
```

This can be used when creating footnotes or for numbering headings.

Using incremental variables for footnotes

Standard notation for footnotes reference in text is `[[X]]` (where **X** is a number) and the actual footnote is `fnX Text`. The clue is that the footnote number (**X**) must match the reference in the text and the actual footnote.

To ease maintenance the **X** can be replaced by two incremental variables. E.g. `[[%a%]]` and `fn%b% Text`.

Using incremental variables for header numbering

For numbering section headers, try the following:

```
h2 %a% The first title
Some text

h2 %a% The second title
Some more text
```

It will produce:

```
<h2>1 The first title</h2>
<p>Some text</p>

<h2>2 The second title</h2>
<p>Some more text</p>
```

Sample site structure

To create a simple web site about, say, an ecological farm, perhaps the following file hierarchy would be appropriate:

```
Home
  About
  Produce
    Apples
    Carrots
  Contact
```

This would be represented by the following in the Pagegen **content directory**, for instance called **eco_farm**. The following files and directories would be setup int the **eco_farm/content** directory:

File/directory name	Title	Type
default	Home	File
000_About	About	File
001_Produce	–	Directory
001_Produce/default	Produce	File
001_Produce/001_Apples	Apples	File
001_Produce/002_Carrots	Carrots	File
002_Contact	–	Directory
002_Contact/default	Contact	File

Contact

Pagegen is developed by Oliver Fields. If you have comments, suggestions, bugs etc. please drop me a line at pagegen@phnd.net mailto:pagegen@phnd.net.

Hope you enjoy Pagegen!