Collect Parser and Supporting Macros

1. Collect parser

http://rileylink.net/moin.cgi/collect.py

2. AnchorLink macro

http://rileylink.net/moin.cgi/AnchorLink.py

3. EmbeddedPage macro

http://rileylink.net/moin.cgi/EmbeddedPage.py

4. FormattedPage macro

http://rileylink.net/moin.cgi/FormattedPage.py

5. GetTOC macro

http://rileylink.net/moin.cgi/GetTOC.py

6. HttpLink macro

http://rileylink.net/moin.cgi/HttpLink.py

7. PageBreak macro

http://rileylink.net/moin.cgi/PageBreak.py

[Top]

Contents

collect

Background and rationale

Collecting a number of MoinMoin pages for printing or saving as a single document could be useful in a range of circumstances. Although a single composite page could be constructed, an automated collection of existing pages makes more sense. Such automation avoids duplication, and having a set of separate pages is better for on-line delivery. The Collect parser has been written to combine of pages into a single document with consistent internal linking and a cover page with hyperlinks to pages within the collection. If saved by the PDF.py action, the internal links also work within a PDF reader.

Description

Collect is a simple parser that reads key:value pairs from a dictionary to combine pages into a collection for printing or saving as a booklet.

The recognised key::value pairs are:

heading:: ***text***

Text to be inserted at the top of the cover page as a level 1 heading

paged:: Yes | No

Yes = insert a page break between collected pages, which has effect in print view and PDF.py action. No = insert a horizontal rule between collected pages.

contents:: List | Contents

List = insert s numbered list of collected pages on the cover page.
Contents = insert table of contents for each collected pages under level 2 headings on the cover page.

contents-depth:: 1...6

Integer from 1 to 6 to specify the heading depth when tables of contents are inserted on the cover page.

page:: ***pagename*** **[|** ***text*** **]**

Name of page to add to the collection, with an optional alternative text delimited by a vertical bar

page-heading:: Yes | No

Yes = insert page name or alternative text (if specified) at the top of each collected page as a level 1 heading.

page-url:: Yes | No

Yes = insert an external link on the cover page for page just specified.

The keys **page**, **page-heading** and **page-url** can be repeated as a set to include more pages in the collection (see the example below).

This parser uses six plugin macros, AnchorLink.py, EmbeddedPage.py, FormattedPage.py, GetTOC.py, HttpLink.py and PageBreak.py, which also need to be installed.

Usage

As processing instruction

```
#format collect [raw]

 key:: value
 key:: value
 ...
```

As parser section

```
{{{#!collect [raw]
 key:: value
 key:: value
 ...
}}}

The 'raw' argument is optional. When set, the output is the unformatted MoinMoin markup.
```

Example

The following example collects this page along with those of the helper macros into a single document. PiAction is used to automatically provide the content as a PDF. The link to the example page followings the listing below of its raw content.

```
#action PDF
#format collect

 heading:: Collect Parser and Supporting Macros
 paged:: Yes
 contents:: List
 page:: collect.py|Collect parser
 page-url:: Yes
 page:: AnchorLink.py|AnchorLink macro
 page-url:: Yes
 page:: EmbeddedPage.py|EmbeddedPage macro
 page-url:: Yes
 page:: FormattedPage.py|FormattedPage macro
 page-url:: Yes
 page:: GetTOC.py|GetTOC macro
 page-url:: Yes
```

```
page:: HttpLink.py|HttpLink macro
page-url:: Yes
page:: PageBreak.py|PageBreak macro
page-url:: Yes
```

collect.py/Example - formatted example as a PDF.

collect.py/ExampleRaw - intermediate MoinMoin markup generated by the Collect parser for this example.

Download & Release Notes

Download	Release Version	Moin Version	Release Notes
collect-1.0.py	1.0	1.9.2	

Copyright

License

Known issues and limitations

- So that internally linked hypertext is distinguished, print.css on this site has been changed to give a and a:visited a text colour of blue.

Hits

31

[Top]

Contents

AnchorLink

Background and rationale

MoinMoin provides a the macro <<Anchor(anchorname)>> to insert anchors into a page. To link to anchors in the current page the MoinMoin markup is [[#anchorname]] or [[#anchorname|label text]]. This inserts a link using a call to the pagelink function of the formatter, with the link created in the form <a href="/pagename#achorname">anchorname</a>. This works fine but it isn't interpreted as a internal page link when converted to a PDF by wkhtmltopdf as used in the PDF.py action.

However, the internal page links to title anchors inserted by the <<TableOfContents>> macro are interpreted correctly by wkhtmltopdf. This occurs because <<TableOfContents>> calls the anchorlink function of the formatter. This function creates a link in the form <a href="#achorname">anchorname</a>.

Therefore, a macro to create a link to an anchor in the current page in the same way links are created by the <<TableOfContents>> macro is needed to provide compatibility with the PDF.py action.

Description

AnchorLink is a macro to insert a link to an anchor in the current page using HTML that is correctly interpreted when the page is converted to a PDF by wkhtmltopdf as used in the PDF.py action.

The label text is inserted without a # prefix. This prefix is included for links created by [[#anchorname]], which is ideal. So if the anchorname is meaningful in its own right, no alternative text needs to be provide, keeping the markup more readable.

Usage

<<AnchorLink(text)>>

```
<<AnchorLink(anchorname) - inserts anchor link using anchorname as the label text.

<<AnchorLink(anchorname|description>> - inserts anchor link using description as the label text.
```

To be functional an anchor must be created by <<Anchor(anchorname)>> elsewhere in the page.

Alternatively anchorname can be the same as a heading, thereby creating a link to that heading because MoinMoin creates anchors for all headings.

Example

```
<<AnchorLink(ExampleAnchor|Click to go to Example Anchor)>>

<<AnchorLink(Example Heading|Click to go to Example Heading)>>

<<Anchor(ExampleAnchor)>> Example anchor here.

=== Example Heading ===
```

Click to go to Example Anchor

Click to go to Example Heading

Example anchor points here.

Example Heading

Example anchor link point to this section.

Download & Release Notes

Download	Release Version	Moin Version	Release Notes
AnchorLink-1.0.py	1.0	1.9.2	

Copyright

License

Known issues and limitations

- If <<AnchorLink(anchorname)>> is used after <<PageBreak()>> (see PageBreak.py), even though wkhtmltopdf creates a new page in the PDF the anchor link doesn't reliably point to that page. When the new page starts with a heading, an anchor link to that heading seems to be more reliable.

Hits

16

[Top]

Contents

EmbeddedPage

Background and rationale

MoinMoin page content included from other pages using the FormattedPage.py macro or other similar macros (e.g. <<Include()>> and <<IncludePage()>>) contain links that potentially cause the browser to leave the current page or that are inoperative. However, the Collection.py parser seeks to combine several pages under a cover page and have all links to the collected pages operate within the context of the current page. This is achieved using the helper macro, EmbeddPage.

Description

EmbeddedPage post-processes HTML from a formatted page to convert URL references to anchor links for the sister pages provided in the arguments. It also changes HTML heading tag ids to match the anchor links. This macro imports the FormattedPage.py marco and is used when embedding one or more wiki pages into the current page. It is a helper macro for the Collection.py parser.

The post processing converts URL references to the target or sister pages provided to anchor links relative to the host page. It also deletes any suffixes added to anchors, heading ids and anchor links added by the formatter, so that they operate effectively from the host page.

Usage

<<EmbeddedPage(pagelist)>>

```
Arguments "targetindex" and "pagelist" are both required.

<<EmbeddedPage(targetindex, pagelist)>>

"targetindex" is an 0-based index for the target page in the page list.

"pagelist" is a comma-delimited list of pages included in the collection.
```

If no index or list is provided, the macro returns nothing. No error message is returned.

Examples

See collect.py parser for examples of it use.

Download & Release Notes

Download	Release Version	Moin Version	Release Notes
EmbeddedPage-1.0.py	1.0	1.9.2	

Copyright

License

Known issues and limitations

- None known

Hits

8

[Top]

Contents

FormattedPage

Background and rationale

MoinMoin provides the Include marco, with its range of options, to insert content from another page into the current page. The included content is fully integrated into the host page rather than as it might appear alone. MoinMoin MacroMarket (http://moinmo.in/MacroMarket) offerings the IncludePage macro, which displays content from other pages within a frame. Neither quite satisfied the needs of the Collection.py parser, which needs to be able to include the full content of pages formatted as they would appear alone, and without affecting the heading structure of the host page. So this macro has been written as a helper macro for that parser, and the content it returns is further processed by the EmbeddedPage.py macro.

Description

FormattedPage returns HTML for a whole page formatted according to the processing instructions of that page. It does not include the options of in-built Include macro, but allows the formatted page to use footnotes, particularly because these are used by the Ref.py macro. Also, it leaves the heading structure for the host page unaltered. It uses MoinMoin.Page.send_page with output redirected so that the HTML can be returned.

Usage

<<FormattedPage(pagename)>>

```
Argument "pagename" is required.

<<FormattedPage(pagename)>>
```

If the page specified does not exist or is empty, the macro returns nothing. No error message is returned.

Examples

See EmbeddedPage.py macro and collect.py parser for examples of it use.

Download & Release Notes

Download	Release Version	Moin Version	Release Notes
FormattedPage-1.0.py	1.0	1.9.2	

Copyright

License

Known issues and limitations

- None known

Hits

10

[Top]

Contents

GetTOC

Background and rationale

The MoinMoin macro TableOfContents creates a contents list for the current page, but not for another page, which of course is totally sensible. However, the collect.py parser combines pages into a single page and being able to included a contents list for each of these in a cover page would be useful. So this GetTOC macro was written as a helper macro for that parser. However, it might have some, albeit limited, applicaiton in other contexts.

Description

GetTOC gets a Table of Contents for the current or another page by calling the MoinMoin macro TableOfContents. It then converts the table heading, "Contents", to a hyperlink to a page anchor with a suffix of '.top', before inserting it in the current page. For this anchor link to be useful, the anchor needs to be inserted on the current page using the Anchor macro. This is done automatically by the collect.py parser. If done directly for pages with name containing non-alphanumeric characters, the name needs to be converted to text in the same way this is done by MoinMoin.wikiutil.wikiutil.anchor_name_from_text.

Usage

<< GetTOC(pagename, maxdepth)>>

```
Both arguments, "pagename" and "maxdepth", are optional.

<<GetTOC(pagename, maxdepth)>> - inserts a Table of Contents for pagename to depth of maxdepth

<<GetTOC>> or <<HttpLink()>> - inserts a Table of Contents for the current page

<<GetTOC(, maxdepth) - inserts a Table of Contents for the current page to depth of maxdepth
```

Examples

```
<<GetTOC(MoinMoin Matters, 1)>>
```

```
<<GetTOC>>

<<GetTOC(, 2)>>
```

Contents

Contents

Contents

The links in the first example above have no targets on this page. See collect.py for details of how it can be used.

The target for Contents link in the other two examples has been inserted at the top of this page using <<Anchor(GetTOC.py.top)>>, and the heading links point to the headings on this page.

Download & Release Notes

Download	Release Version	Moin Version	Release Notes
GetTOC-1.0.py	1.0	1.9.2	

Copyright

License

Known issues and limitations

- None known

Hits
82

[Top]

Contents

HttpLink

Background and rationale

MoinMoin inserts internal links as hypertext, either the page name (usually a WikiName) or as alternative text, if provided. This is suitable for pages viewed on-line, but the link is lost in printed or saved pages. For situations where the link needs be retained off-line, displaying the full URL would be an appropriate solution. However, MoinMoin does not currently facilitate the automatic generation of a full URL for pages within the wiki. If links are entered explicitly, there is a risk of link rot and therefore require more maintenance.

Description

HttpLink is a macro to insert an external link to a local wiki page link for stituations where it would be helpful to show the full URL, such as pages intended to be printed or saved.

HttpLink replaces spaces in page names with underscores to avoid the space character code (%20) being inserted into the URL, as such codes degrades URL readability.

Usage

<<HttpLink(pagename)>>

```
Argument "pagename" is optional.

<<HttpLink(pagename)>> - inserts an external link to `pagename`, or nothing if the page does not exist.

<<HttpLink>> or << HttpLink()>> - insert an external link to the current page.
```

Examples

```
<<HttpLink>>

<<HttpLink()>>

<<HttpLink(MoinMoin Matters)>> - Note: space replaced with underscore.
```

http://rileylink.net/moin.cgi/HttpLink.py

http://rileylink.net/moin.cgi/HttpLink.py

http://rileylink.net/moin.cgi/MoinMoin_Matters - Note: space replaced with underscore.

Download & Release Notes

Download	Release Version	Moin Version	Release Notes
HttpLink-1.0.py	1.0	1.9.2	

Copyright

@copyright: 2014 Ian Riley < ian@riley.asia>

License

GNU GPL, see COPYING for details.

Known issues and limitations

- None known

Hits
41

[Top]

Contents

PageBreak

Background and rationale

MoinMoin renders pages in HTML for display in browsers without pagination, as expected. When printing, different browsers will format the page differently, so sometimes the page breaks will be in awkward places (e.g. widowed headings, split images) and there is no way of controlling where these breaks occur. Also, it can be useful to save the print preview as a PDF (which is easily done under OS X), but likewise the page breaks will be undesirable. A macro to force a page break on printing would be useful, perhaps not for permanent inclusion in the page, but at least for rendering when printing or creating a PDF.

Description

PageBreak is a macro to insert a HTML division with a forced page break on printing.

Including an argument allows that arguments text to be printed on the page before the break. The argument text is escaped to make sure it is benign.

The argument text is only included in print view, and paging only occurs when printing from print view.

Usage

<<PageBreak(text)>>

```
Argument "text" is optional.

<<PageBreak>> or <<PageBreak()>> - forces a page break on printing.

<<PageBreak(text)>> - prints the text provided before forcing the page break.
```

In practice, the macro might need to be included before the first awkward break, then the page saved and checked for subsequent awkward breaks.

For a public page, it is recommended that any instances of the macro be removed after printing, because they might not work appropriately on another browser, or if the content of the page is changed. Also, not to clutter the page edit history with such changes, it might be wise to create a temporary version of the page to be printed in the SandBox or under the user's WikiHomePage.

For a private page, that you will be using with the same operating system and browser, and for which you are responsible of any changes to the page content, the instances of the macro could remain.

Example

```
...
blah blah
<<PageBreak(continued over)>>
blah blah
...
```

Page 1	Page 2
... blah blah continued over	blah blah ...

Download & Release Notes

Download	Release Version	Moin Version	Release Notes
PageBreak-1.2.py	1.2	1.9.2	

Copyright

License

Known issues and limitations

- ~~Used before headings incompatible with wkhtmltopdf in PDF action. Only bottom half of heading displayed.~~ Fixed with inclusion of non-breaking space in Version 1.2.
- ~~The text provided in the argument is always included in the page. It would be better if this could be restricted to print contexts only.~~ Added with Version 1.1.

Hits

36

collect.py/Example (last edited 2014-06-27 23:59:49 by IanRiley)