Article custom

On this page:

Syntax

```
<txp:article_custom />
```

The **article_custom** tag is a *single* or a *container* tag that provides a variety of custom options for sorting, selecting, and displaying articles (the tag will be replaced with one or more articles).

If used as a container, it must be specified as an opening and closing pair of tags, like this:

```
<txp:article_custom>
    ...contained statements...
</txp:article_custom>
```

This is equivalent to putting the contained statements into a form named 'my_form' and using `<txp:article_custom form="my_form" />`.

Unlike the article tag, `<txp:article_custom>` will always return an article list and **is not context-sensitive**. This means while the article tag can only see posts within the currently viewed section/category/author and so forth, `<txp:article_custom />` can see all posts from all sections, categories and authors unless you restrict it via attributes (see below), thus context-sensitive navigation tags, such as older and newer, will not work.

`<txp:article_custom />` offers many additional attributes that `<txp:article />` does not.. but only `<txp:article />` will produce the full single article page. If you have only `<txp:article_custom />` on a page, rather than `<txp:article />`, then you will never reach the permalinked article page – you'll always get an article list page.

However, you can have the added features and functionality of `<txp:article_custom />`, while keeping the necessary `<txp:article />` in full force by using conditionals: `<txp:if_individual_article>` and `<txp:if_article_list>`, e.g.:

```
<txp:if_article_list>
    <txp:article_custom form="myform" limit="10" category="ideas" section="article" sortby="Posted" sortdir="desc" />
</txp:if_article_list>

<txp:if_individual_article>
    <txp:article />
</txp:if_individual_article>
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `allowoverride="boolean"`
  Whether to use override forms for the generated article list.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `author="author name"`
  Restrict to articles by specified author(s) **login name**.
  Values: (comma separated list of) author login name(s).
  Default: unset, retrieves from all authors.
- `category="category name"`
  Restrict to articles from specified category/categories. Note: the category names may be different to the title you typed when you created the category, as the names are sanitized for URL use. Check the Categories administration panel to ensure you are using the correct names.
  Values: (comma separated list of) category name(s).
  Default: unset, retrieves from all categories.
- `customfieldname="value"`
  Restrict to articles with specified value for specified custom field name. Replace *customfieldname* with the actual name of the custom field.
  Important: Using dashes `-` or spaces may cause errors or render this feature ineffectual. Underscores in both custom field names and values are confirmed to work.
  Default: unset.
- `excerpted="boolean"`
  Restrict to articles with/without an excerpt.
  Values: `0` (no, return all) or `1` (yes, return only those containing an excerpt).
  Default: `0`.
- `exclude="article id(s)"`
  Exclude a specific article or list of articles (each ID separated by a comma).
  Default: unset.
- `expired="boolean"`
  Whether to include articles that have expired or not.
  Values: `0` (no, don't include expired articles) or `1` (yes, include expired articles).

Default: Setting of preference ‘Publish expired articles’.

- `form="form name"`
  Use specified form template to process each article.
  Default: `default`.
- `id="article ID"`
  Display the specific article or list of articles (each ID separated by a comma).
  Important: When a list is supplied, this does **not** imply a sort order (see **Example 6** below).
  Default: unset.
- `keywords="keyword(s)"`
  Restrict to articles with specified keyword(s).
  Default: unset.
- `limit="integer"`
  The number of articles to display.
  Default: `10`.
- `month="yyyy"/"yyyy-mm"/"yyyy-mm-dd"`
  Restrict to articles posted within the specified year/month/day.
  Default: unset.
- `offset="integer"`
  The number of articles to skip.
  Default: `0`.
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: unset, retrieves from all sections.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `authorid` (author name).
  `category1`.
  `category2`.
  `comments_count`.
  `custom_1` through `custom_10` (from Textpattern 4.2.0 onwards: `custom_n` where ‘n’ is the number of your custom field – for numeric values use `(custom_n+0)`).
  `expires` (expiry date).
  `id` (article id#).
  `image` (article image id#).
  `keywords`.
  `lastmod` (date last modified).
  `lastmodid` (author name of last modification).
  `posted` (date posted).
  `rand()` (random).
  `section`.
  `status`.
  `title`.
  `url_title`.
  Each field in the `textpattern` database table can be used as a sort key.
  Default: `posted desc`.
- `status="status"`
  Restrict to articles with the specified `status`.
  Values: `live` or `sticky`.
  Default: `live`.
- `time="time"`
  Restrict to articles posted within specified timeframe.
  Values: `past`, `future` or `any` (both `past` and `future`).
  Default: `past`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display 5 articles from a specific section on the homepage

To show 5 articles from a ‘news’ section within the homepage `default` page template (or the template you use for displaying your homepage), add:

```
<txp:article_custom section="news" limit="5" form="homepage_articles" />
```

Then create an ‘article’ type form template called `homepage_articles`, containing:

```
<article>
    <h1>
        <txp:title />
    </h1>
    <txp:if_article_image>
        <p>
            <txp:article_image />
        </p>
    </txp:if_article_image>
    <txp:body />
</article>
```

Other tags used: article_image, body, if_article_image, title.

Example 2: List articles published in specified month

```
<txp:article_custom form="month_list" sort="Section asc" month="2004-14" />
```

This code will display a monthly list articles by section and in ascending order, between the years of 2004 and 2014. See **Example 5** below for more information on article sorting.

Example 3: Select by keyword

```
<txp:article_custom sort="Posted desc" keywords="One" />
```

This code will display articles that have a keyword with a value `One`.

Example 4a: Select by author (using single tag)

```
<txp:article_custom form="author_list" author="Parkling" />
```

This code will display articles that have a authored by author with login name of `Parkling`. The referenced `author_list` article form might go thus:

```
<p>
    <txp:permlink>
        <txp:title />
    </txp:permlink>
</p>
```

Other tags used: permlink, title.

Example 4b: Select by author (using container tag)

```
<txp:article_custom author="Parkling">
    <p>
        <txp:permlink>
            <txp:title />
        </txp:permlink>
    </p>
</txp:article_custom>
```

This is exactly equivalent to **Example 3a**, just using a container tag instead of a single tag with referenced form.

Other tags used: permlink, title.

Example 5: Combined with custom fields

```
<txp:article_custom colour="red" />
```

This code will display articles that have a custom field named `colour` with a value `red`.

Example 6: Article sorting

```
<txp:article_custom sort="AuthorID asc, Posted asc" />
```

Uses the `sort` attribute to define values by which to order article output. In this case two values are declared. `AuthorID asc` first orders articles alphabetically by author names, then `Posted desc` orders them by date published (`desc` meaning newest to oldest).

Why might you do it? Sorting is a powerful way to group articles (e.g. by author), and/or give priority to articles most recently published (typically better for your website visitors).

Example 7: Select by article ID

```
<txp:article_custom id="81,73" />
```

Outputs articles specified by list of IDs, in this example that would be articles `81` and `73`. Order of articles may not match the order of the IDs in the list, that depends on what `sort` ordering you have specified, for example:

```
<txp:article_custom id="81,73" sort="field(id,81,73)" />
```

Outputs articles specified by list of IDs, in the order given in the `sort` field.

Genealogy

Version 4.6.0

`exclude` attribute added.

Version 4.5.0

`expired` attribute added.

Version 4.0.7

Can be used as a container tag.
`id` attribute can take a comma-separated list of IDs.
`wraptag` and `break` attributes added.

Version 4.0.6

Support added for comma separated lists for `section`, `category` and `author` attributes.

Version 4.0.4

`listform` attribute deprecated (it never made a difference to article_custom anyway).
`sort` attribute added (replaces `sortby` and `sortdir` attributes).
`sortby` and `sortdir` attributes deprecated.

Article id

On this page:

Syntax

```
<txp:article_id />
```

The `article_id` tag is a *single* tag which returns the numeric ID of the article being displayed. This number will also be reflected as a part of the article permanent URL if it has been chosen as the ‘Permanent link mode’ in the Preferences administration panel.

Attributes

This tag has no attributes.

Examples

Example 1: Hyperlinked to the article

```
<txp:permlink>
    <txp:article_id />
</txp:permlink>
```

Other tags used: permlink.

Example 2: Conditional use

This will only display the hyperlinked article ID when on an individual article page.

```
<txp:if_individual_article>
    Article ID:
    <txp:permlink>
        <txp:article_id />
    </txp:permlink>
</txp:if_individual_article>
```

Other tags used: if_individual_article, permlink.

Article image

On this page:

Syntax

```
<txp:article_image />
```

The **article_image** tag is a *single* tag. Textpattern will replace this tag with the `<img src="...">` HTML tag matching the numeric ID or URL assigned when the article is posted.

The image to be associated with the tag is set in the Write administration panel. Click 'Advanced options' and enter either the URL of the image, or the Textpattern ID (a number set by Textpattern at upload) into the **Article image** field. Most of the time you will use the image ID here. Note that you can only assign a single image to each article.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&` for the image's `alt` and `title` attributes.
  Values: `html` or unset.
  Default: `html`.
- `height="integer"`
  Specify an image `height` which overrides the value stored in the database. Use `height="0"` to turn off the output of a width attribute in the `<img>` tag (thus the browser will scale the height if a width is used).
  Default: height of image stored in the database.
- `html_id="id"`
  The HTML `id` attribute assigned to the image (or to the `wraptag`, if set).
  Default: unset.
- `thumbnail="boolean"`
  Use the thumbnail rather than full-size image.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `width="integer"`
  Specify an image `width` which overrides the value stored in the database. Use `width="0"` to turn off the output of a width attribute in the `<img>` tag (thus the browser will scale the width if a height is used).
  Default: width of image stored in the database.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  CSS `class` attribute to apply to the image (or to the `wraptag`, if set).
  Default: unset (see [[class cross-reference]]).
- `style="style rule"`
  Inline CSS `style` rule. It's recommended that you assign CSS rules via `class` attribute instead.
  Default: unset.
- `wraptag="tag"`
  HTML tag to be used to wrap the `<img>` tag, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Use wraptag and class for styling

```
<txp:article_image wraptag="p" class="article-image" />
```

This will wrap the image in paragraph tags, applying the `class` to the paragraph:

```
<p class="article-image">
    <img src="...">
</p>
```

It gives you full control over the image appearance using CSS. Note: Without the wraptag, the class is applied directly to the `<img>` tag.

Example 2: Link thumbnail to the article

Used in an article list form, this will display an article list consisting of hyperlinked article images' thumbnails:

```
<txp:permlink>
    <txp:article_image thumbnail="1" />
</txp:permlink>
```

Other tags used: permlink.

Genealogy

Version 4.3.0

`width` and `height` attributes added.

Version 4.2.0

`align` attribute deprecated.

Version 4.0.7

Default value for `escape` attribute changed from 'unset' to `html`.

Version 4.0.4

`class`, `escape`, `;html_id`, `thumbnail` and `wraptag` attributes added.

Article url title

On this page:

Syntax

```
<txp:article_url_title />
```

The **article_url_title** tag is a *single* tag which returns the dumbed-down ‘URL title’ of the article being displayed. This URL title may also be part of the page’s address depending on the ‘Permanent link mode’ chosen in the Preferences administration panel.

Attributes

This tag has no attributes.

Genealogy

Version 4.0.5

Tag support added.

Article

On this page:

Syntax

```
<txp:article />
```

The basic **article** tag can be used as either a *single* tag or *container* tag and used to output one or more articles depending on the attributes used. Default attributes will be used when nothing specific is assigned.

It may be used as a *container* tag, in which case it must be specified as an opening and closing pair of tags, like this:

```
<txp:article>
    ...contained statements...
</txp:article>
```

This is equivalent to putting the contained statements into a form named 'my_form' and using `<txp:article form="my_form" />`.

The tag is context-sensitive, which means it will grab articles from the currently viewed section/category/author, etc.

When used on the front page, article's context will include articles from all sections set to display via 'Section appears on front page?' settings (see the Sections administration panel for more information).

`<txp:article />` is **not** the same as `<txp:article_custom />` – you can check out the differences of that tag if you're unsure of the differences!

Attributes

Tag will accept content/behaviour and presentation attributes (**case-sensitive**):

Content/behaviour attributes

- `allowoverride="boolean"`
  Whether to use override forms for the generated article list.
  Values: `0` (no) or `1` (yes).
  Default: `1`.
- `customfieldname="value"`
  Restrict to articles with specified value for specified custom field name. Replace *customfieldname* with the actual name of the custom field.
  Important: Using dashes `-` or spaces may cause errors or render this feature ineffectual. Underscores in both custom field names and values are confirmed to work.
- `form="form name"`
  Use specified form template to process each article.
  Default: `default`.
- `keywords="keyword(s)"`
  Restrict to articles with specified keyword(s).
- `limit="integer"`
  The number of articles to display.
  Default: `10`.
- `listform="form name"`
  Use specified form when page is displaying an article list.
  Default: `article_listing`.
- `offset="integer"`
  The number of articles to skip.
  Default: `0`.
- `pageby="integer"`
  The number of articles to jump forward or back when an older or newer link is clicked. Allows you to call the article tag several times on a page without messing up older/newer links.
  Default: value matches the value assigned to `limit`.

- `pgonly="boolean"`
  Do the article count, but do not display anything. Used when you want to show a search result count, or article navigation tags **before** the list of articles. Just make sure that, other than `pgonly`, both article tags are identical (form-related attributes are the exception, they do not need to be assigned).
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `searchall="boolean"`
  When outputting search results, include only those articles with 'Include in site search' set on the Sections administration panel. If set to `0`, only articles in the current section are displayed. See [[Fixing search results]] for more.
  Values: `0` (no) or `1` (yes).
  Default: `1`.
- `searchform="form name"`
  The form to be used for your customized search results output.
  Default: `search_results`.
- `searchsticky="boolean"`
  When outputting search results, include articles with status `sticky`.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `authorid` (author name).
  `category1`.
  `category2`.
  `comments_count`.
  `custom_1` through `custom_10` (from Textpattern 4.2.0 onwards: `custom_n` where 'n' is the number of your custom field – for numeric values use `(custom_n+0)`).
  `expires` (expiry date).
  `id` (article id#).
  `image` (article image id#).
  `keywords`.
  `lastmod` (date last modified).
  `lastmodid` (author name of last modification).
  `posted` (date posted).
  `rand()` (random).
  `section`.
  `status`.
  `title`.
  `url_title`.
  Each field in the `textpattern` database table can be used as a sort key.
  When viewing a search results list, `score` (how well the search terms match the article) is available as an additional value.
  Default: `posted desc` (`score desc` for search results).
- `status="status"`
  Restrict to articles with the specified `status`.
  Values: `live` or `sticky`.
  Default: `live`.
- `time="time"`
  Restrict to articles posted within specified timeframe.
  Values: `past`, `future` or `any` (both `past` and `future`).
  Default: `past`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).

- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Note on 'article list' vs. 'individual article' context

The **article** tag is context-sensitive. It will produce different results depending on whether the page being viewed is an article list or an individual article. Article-list context includes the default (home) page, section front pages, and category pages. Individual-article context applies on an article page (i.e. a page with a URL like `http://example.com/archives/24/my-article`).

Examples

Example 1: Basic use as single tag

Here is the **article** tag responsible for the main content of the home page on a new Textpattern installation:

```
<txp:article limit="5" />
```

Calls the `default` article form, which may contain any variation of article output you want to create. The `default` form cannot be deleted; it is the form you see on first viewing the Forms administration panel.

Uses the `limit` attribute to specify the maximum number of articles displayed in article list context (if not specified, this defaults to `10`).

Example 2: Specifying a form

Expanding on **Example 1**, here is the `article` tag responsible for showing lists of articles by category in the default page of a new Textpattern installation:

```
<txp:article form="article_listing" limit="5" />
```

In article list context, the form named `article_listing` will be processed and displayed for each article in the list. In individual article context, the `default` form would be used.

To see this in action, on a new Textpattern install, from the home page click on one of the category links near the bottom (right above the Comment link). Note the URL, similar to `http://example.com/category/meaningful-labor`. The `category` in the URL means this is a listing of articles by category. Here you see only the article title, posting date and article information (not the full article itself), because that is what is contained in the form named `article_listing`. Now click on the article title. Note the URL, similar to `http://example.com/articles/1/welcome-to-your-site`. This is an individual article page. Once again you can see the full article, this time with comments showing (if comments are enabled).

Example 3: Offsetting article display

Continuing from the previous examples:

```
<txp:article listform="article_listing" limit="5" offset="2" />
```

Here we include the `offset` attribute to offset article display by `2` articles. This means the five articles that will be displayed (i.e. `limit="5"`) in article list context will begin with the third most recent article published in the site (the offset will not be applied in individual article context).

Why might you do it? Offsetting articles is useful in situations where the most recent article(s) are already accessible in some way and you don't want them appearing again in normal article flow.

Example 4: Using pageby to split article output on a page

```
<div class="first">
    <txp:article limit="1" pageby"10" />
</div>
<div class="middle">
    <txp:article limit="8" offset="1" pageby="10" />
</div>
<div class="last">
    <txp:article limit="1" offset="9" pageby="10" />
```

```
</div>
```

Another:

```
<txp:article limit="5" pageby="10" />
<!-- google ad -->
<txp:article limit="5" offset="5" pageby="10" />
```

The `pageby` number should be the total number of articles displayed on the page. Without `pageby`, each article tag would page independently based on its own `limit`, as if it was the only article tag.

Example 5: Combined with custom fields

```
<txp:article colour="red" />
```

This code will display articles that have a custom field named `colour` with a value `red`.

Example 6: Article sorting

```
<txp:article sort="AuthorID asc, Posted desc" />
```

Uses the `sort` attribute to define values by which to order article output. In this case two values are declared. `AuthorID asc` first orders articles alphabetically by author names, then `Posted desc` orders them by date published (`desc` meaning newest to oldest).

Why might you do it? Sorting is a powerful way to group articles (e.g. by author), and/or give priority to articles most recently published (typically better for your website visitors).

Genealogy

Version 4.0.7

Can be used as a container.
`wraptag` and `break` attributes added.

Version 4.0.4

`sort` attribute added (replaces `sortby` and `sortdir`) attributes.
`sortby` and `sortdir` attributes deprecated.

Version 4.0.2

`pageby` attribute added.

Author email

On this page:

Syntax

```
<txp:author_email />
```

The **author_email** tag is a *single* tag that is used to return the email address of the author of the currently displayed article.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`.
- `link="boolean"`
  Make text a `mailto:` link.
  Values: `0` (no) or `1` (yes).
  Default: `0`.

Examples

Example 1: Display email address (but don’t link)

```
<p>
    <txp:author_email />
</p>
```

Example 2: Create a mailto: link

```
<txp:author_email link="1" />
```

Genealogy

Version 4.5.0

Tag support added.

Author

On this page:

Syntax

```
<txp:author />
```

The **author** tag is a *single* tag that is used to return the name of the author of the currently displayed article.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`;
- `format="boolean"`
  Display plain URL or full link.
  Values: `link`, `url` or unset.
  Default: unset (display title or full link, depending on `link`).
- `link="boolean"`
  Make text a link to the author's posts.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: unset, retrieves from all sections.
- `this_section="boolean"`
  If set to `1`, the linked author name will direct users to an author list in the current section, otherwise author list from all sections is displayed.
  Values: `0` (no, all sections) or `1` (yes, this section only).
  Default: `0`.
- `title="boolean"`
  Whether to display the author's real name or login name.
  Values: `0` (login name) or `1` (real name).
  Default: `1`.

Examples

Example 1: Link to list of author's articles

```
<h1>
    <txp:title />
</h1>
<txp:body />
<p class="author-date">
    Posted By: <txp:author link="1" /> at <txp:posted />
</p>
```

The author's name in this article form is a hyperlink to a list of articles by this author.

Other tags used: posted, title, body.

Example 2: Author landing page

```
<txp:if_author>
    <h1>
```

```
        Articles by author: <txp:author />
    </h1>
    <txp:article form="article_listing" limit="5" />
</txp:if_author>
```

Display the author's name above a list of articles by that author when visiting `example.com/author/Author+Name` URLs.

Other tags used: if_author, article.

Genealogy

Version 4.6.0

`escape` and `format` attributes added.

Version 4.5.0

Permitted the tag to be used on author list landing pages.

Version 4.3.0

`title` attribute added.

Version 4.0.4

`section` and `this_section` attributes added.

Authors

On this page:

Syntax

```
<txp:authors>
```

The **authors** tag is a *single* or *container* tag that Textpattern will use to gather a list of authors designated within the Textpattern Users administration panel.

If used as a *container* tag, it must be specified as an opening and closing pair of tags, like this:

```
<txp:authors>
    ...contained statements...
</txp:authors>
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `form="form name"`
  Use specified form template to process each author.
  Default: unset.
- `group="user group"`
  Comma-separated list of user groups (publishing roles that the authors belong to.
  Values: `publisher`, `managing_editor`, `copy_editor`, `staff_writer`, `freelancer`, `designer` or `privs_none`.
  Default: unset, retrieves from all groups.
- `limit="integer"`
  The number of authors to display.
  Default: `0` (no limit).
- `name="author"`
  Comma-separated list of author names.
  Default: unset, which determines whether 'any' author listing is being viewed.
- `offset="integer"`
  The number of authors to skip.
  Default: `0`.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `id` (author id#).
  `last_access` (most recent log in).
  `name` (login name).
  `realname` (real name)
  `privs` (user group).
  `rand()` (random).
  Each field in the `textpattern` database table can be used as a sort key.
  Default: `name asc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`

Label prepended to item.
Default: unset (but see [[label cross-reference]] for exceptions).

- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag=“h3”).
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: List all authors

```
<txp:authors />
```

Example 2: List authors only from certain groups, with link and email

```
<txp:authors group="publisher, managing_editor" wraptag="ul" break="li">
    <txp:author link="1" />
    (<txp:author_email />)
</txp:authors>
```

Other tags used: author, author_email.

Example 3: Render a drop-down list of authors

```
<select name="author">
    <txp:authors sort="realname asc">
        <option value="<txp:author />"><txp:author /></option>
    </txp:authors>
</select>
```

Other tags used: author.

Genealogy

Version 4.6.0

Tag support added.

Body

On this page:

Syntax

```
<txp:body />
```

The **body** tag is a *single* tag which is used to return the text, or content, of the article being displayed (the article itself). The tag can be used in a Textpattern ‘article’ type [[Form template]], or within a [[Page template]], either wrapped within a given article tag, or directly in the template itself so long as the context is with a single article (as opposed to an article list).

Attributes

This tag has no attributes.

Examples

Example 1: Display the article text

```
<h1>
    <txp:title />
</h1>
<p>
    <txp:author /> at <txp:posted />
</p>
<txp:body />
```

When used as part of your article form, this displays the article title, author and posted date, then the body text beneath that.

Other tags used: author, posted, title.

Breadcrumb

On this page:

Syntax

```
<txp:breadcrumb />
```

The **breadcrumb** tag is a *single* tag which is used to create breadcrumb navigation. It provides either hyperlinked navigation, or plain text positional display. Breadcrumbs are *not* displayed on the ‘default’ section (home page) of your site.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="value"`
  Whether to hyperlink breadcrumbs.
  Values: `0` (no) or `1` (yes).
  Default: `1`.
- `linkclass="class name"`
  HTML class attribute applied to the breadcrumb links.
  Default: unset.
- `separator="value"`
  Character to be used as the breadcrumb separator.
  Default: » .
- `title="boolean"`
  Whether to display the title or not.
  Values: `0` (no, display name) or `1` (yes).
  Default: `0`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `wraptag="element"`
  HTML element to wrap breadcrumb block, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display a hyperlinked breadcrumb trail

```
<txp:breadcrumb label="Navigation" separator="::" link="1" wraptag="p" />
```

Provides hyperlinks to sections or categories in a breadcrumb style, linking back to your home page.

Example 2: Display a text only breadcrumb trail

```
<txp:breadcrumb label="Navigation" separator="/" link="0" wraptag="p" />
```

Provides a breadcrumb guide that reflects where a user is within the site’s navigation.

Genealogy

Version 4.5.0

Default `class="noline"` for `linkclass` attribute removed (now unset).

Version 4.3.0

`sep` attribute deprecated and renamed `separator`.

Category list

On this page:

Syntax

`<txp:category_list />`

The **category_list** tag can be used as either a *single* tag or *container* tag which is used to produce a list of linked categories.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `active_class="class name"`
  HTML `class` attribute to be applied to the `active` or current link in a list. Only works in the *single* tag without the `form` attribute.
  Default: unset.
- `categories="category name(s)"`
  Comma-separated list of categories to include, displayed in the order specified (unless overridden by `sort` attribute). Use category names **not** titles here – note that Textpattern automatically converts the names to lowercase and converts spaces to hyphens when they are created.
  Default: unset (all categories).
- `children="boolean"`
  Can limit the list depth to one level below the parent category.
  Values: `0` (no children, i.e.,only show one level below the parent) or `1` (show all nested categories).
  Default: `1`.
- `exclude="category name(s)"`
  List of category names which will be excluded from the list. `categories` takes precendence over `exclude`.
  Default: unset.
- `form="form name"`
  Use specified form to process each included category.
  Default: unset.
- `html_id="id"`
  The HTML `id` attribute applied to the `wraptag`, if set.
  Default: unset.
- `limit="integer"`
  The number of articles to display.
  Default: `0` (no limit).
- `offset="integer"`
  The number of articles to skip.
  Default: `0`.
- `parent="category name"`
  Return only specified category and its children categories. Accepts comma-separated list of values.
  Default: unset.
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: unset, retrieves from all sections.
- `sort="sort value(s)"`
  How to sort the resulting list.
  Values:
  `id`.
  `name`.
  `parent`.
  `rand()` (random).
  `title`.
  `type`.
  Default: `name asc`.
- `this_section="boolean"`

Link to currently active section (overrides `section` attribute).
Values: `0` (no) or `1` (yes).
Default: `0`.
- `type="category type"`
  Values: `article`, `image`, `link`, `file`.
  Default: `article`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Labelled category list

```
<txp:category_list label="Categories" wraptag="p" break="br" />
```

Example 2: As an unordered list

```
<txp:category_list break="li" wraptag="ul" />
```

Example 3: Set active class using the container tag

```
<txp:category_list wraptag="ul" break="">
    <li<txp:if_category name="<txp:category />"> class="active"</txp:if_category>>
        <txp:category title="1" link="1" />
    </li>
</txp:category_list>
```

This code will add `class="active"` to the `<li>` element around the current viewed category in the list, allowing your to style it with CSS as desired.

Other tags used: category, if_category.

Genealogy

Version 4.6.0

`html_id`, `limit` and `offset` attributes added.

Version 4.0.7

Can be used as a container tag.
`form` and `children` attributes added.

Version 4.0.4

`active_class`, `categories`, `exclude`, `section` and `this_section` attributes added.

Category

On this page:

Syntax

`<txp:category />`

The **category** tag can be used as either a *single* or *containing* tag. It will display information of the category as defined by the `name` attribute, or the one currently being viewed. When used as a containing tag, it will turn the contents into a link to the category. Otherwise, it will return plain text.

May be used in any context.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="boolean"`
  Whether to display as link. Works only in the *single* tag, **not** in the *containing* tag variant.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `name="category name"`
  Display specific category. Note the category name is specified in lowercase regardless of how you typed its title in the Categories administration panel. Also note that if you had called your category `My Category Name` it becomes `my-category-name` when used in tags.
  Default: unset (use current category).
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: current section (for backwards compatibility).
- `this_section="boolean"`
  If set to `1`, the linked category name will direct users to an category list in the current section, otherwise category list from all sections is displayed.
  Only link to articles from the current section. The `section` attribute overrides this setting.
  Values: `0` (no, all sections) or `1` (yes, this section only).
  Default: `0`.
- `title="boolean"`
  Whether to display category's title instead of its name.
  Values: `0` (no, display name) or `1` (yes, display title).
  Default: `0`.
- `type="category type"`
  Values: `article`, `image`, `link` or `file`.
  Default: `article`.
- `url="boolean"`
  Display plain URL or full link.
  Values: `0` (no) or `1` (yes).
  Default: `0` (display title or full link, depending on `link`).

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` attribute, applied to `wraptag`. If no `wraptag` is supplied (and `link="1"`), the `class` is applied to the `<a>` tag instead.
  Default: unset (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag to wrap around output, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display the current category name

```
<txp:category />
```

Example 2: Display hyperlinked category title

```
<txp:category title="1" link="1" />
```

Example 3: Display a specific category's title, hyperlinked

```
<txp:category name="article" title="1" link="1" wraptag="p" />
```

Example 4: Container example

```
<txp:category name="book">My books</txp:category>
```

Genealogy

Version 4.0.7

Applies `class`; attribute to the `<a>` tag when the `wraptag` attribute is empty.
`url` attribute added.

Version 4.0.4

`this_section` attribute added.

Category1

On this page:

Syntax

```
<txp:category1 />
```

The **category1** tag can be used as either a *single* tag or *container* tag. It will display information of the category as defined by **Category 1** of the article being displayed. When used as a containing tag, it will turn the contents into a link to that category. Otherwise, it will return plain text.

This tag may be used within either an article form, or in a page, wrapped in an if_individual_article conditional tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="boolean"`
  Whether to link to articles from the same category. Works only in the *single* tag.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: unset, retrieves from all sections.
- `title="boolean"`
  Whether to output category title, rather than name.
  Values: `0` (no, use name) or `1` (yes, use title).
  Default: `0`.
- `this_section="boolean"`
  Whether to only link to articles from the section containing the current article.
  Values: `0` (no, allow from any section) or `1` (yes).
  Default: `0`

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class attribute` to be applied to `wraptag`.
  Default: unset (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag to wrap around output, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Category name in plain text

```
<p>
    <txp:category1 />
</p>
```

Example 2: Hyperlinked category title

```
<p>
    <txp:category1 link="1" title="1" />
</p>
```

If **category1** is ‘General stuff’, this tag will display the words ‘General stuff’ as a hyperlink to a list of articles in the same category.

Example 3: Container example

```
<p>
    <txp:category1>Other articles in category:
    <txp:category1 title="1" /></txp:category1>
</p>
```

Example 4: Category 1 and 2

```
<p>
    Filed in: <txp:category1 link="1" title="1" />
    <txp:if_article_category number="2">
        and <txp:category2 link="1" title="1" />
    </txp:if_article_category>
</p>
```

Shows a hyperlinked category 1, and also hyperlinked category 2 but only if it is used.

Other tags used: category2, if_article_category.

Genealogy

Version 4.0.4

`class`, `section`, `this_section` and `wraptag` attributes added.

Category2

On this page:

Syntax

```
<txp:category2 />
```

The **category2** tag can be used as either a *single* tag or *container* tag. It will display information of the category as defined by **Category 2** of the article being displayed. When used as a containing tag, it will turn the contents into a link to that category. Otherwise, it will return plain text.

This tag may be used within either an article form, or in a page, wrapped in an if_individual_article conditional tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="boolean"`
  Whether to link to articles from the same category. Works only in the *single* tag.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: unset, retrieves from all sections.
- `title="boolean"`
  Whether to output category title, rather than name.
  Values: `0` (no, use name) or `1` (yes, use title).
  Default: `0`.
- `this_section="boolean"`
  Whether to only link to articles from the section containing the current article.
  Values: `0` (no, allow from any section) or `1` (yes).
  Default: `0`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class attribute` to be applied to `wraptag`.
  Default: unset (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag to wrap around output, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Category name in plain text

```
<p>
    <txp:category2 />
</p>
```

Example 2: Hyperlinked category title

```
<p>
    <txp:category2 link="1" title="1" />
</p>
```

If **category2** is ‘Specific stuff’, this tag will display the words ‘Specific stuff’ as a hyperlink to a list of articles in the same category.

Example 3: Container example

```
<p>
    <txp:category2>Other articles in category:
    <txp:category2 title="1" /></txp:category2>
</p>
```

Example 4: Category 1 and 2

```
<p>
    Filed in: <txp:category1 link="1" title="1" />
    <txp:if_article_category number="2">
        and <txp:category2 link="1" title="1" />
    </txp:if_article_category>
</p>
```

Shows a hyperlinked category 1, and also hyperlinked category 2 but only if it is used.

Other tags used: category1, if_article_category.

Genealogy

Version 4.0.4

`class`, `section`, `this_section` and `wraptag` attributes added.

Comment anchor

On this page:

Syntax

```
<txp:comment_anchor />
```

The **comment_anchor** tag is a *single* tag which is used to produce an empty anchor tag with an id attribute reflecting the comment ID. Used in the form that renders your comments (the default form is named ‘comments’).

Attributes

This tag has no attributes.

Examples

Example 1: Generate current comment anchor

```
<txp:comment_anchor />
<txp:comment_message />
```

When the comment number is `000005` you will see:

```
<a id="c000005"></a>
```

Other tags used: comment_message.

Comment email input

On this page:

Syntax

```
<txp:comment_email_input />
```

The **comment_email_input** tag is a *single* tag which is used to display a text entry field to accept the commenter's email address. Used in the comment input form template.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `size="integer"`
  HTML `size` attribute to be applied to the HTML form text `input` field.
  Default: `25`.

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comments_help, comment_message_input, comment_name_input, comment_preview, comment_remember, comment_submit, comment_web_input.

Genealogy

Version 4.6.0

`size` attribute added (replaces functionality of deprecated `isize` attribute in comments_form tag).

Comment email

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:comment_email />
```

The **comment_email** tag is a *single* tag which is used to display the commenter's email address, if entered at the time of posting. Should be used in a Textpattern 'comment' type [[Form template]].

Attributes

This tag has no attributes.

Examples

Example 1: Comments display form with linked email and comment id

```
<txp:comment_message />
<p>
    <a href="mailto:<txp:comment_email />">Email</a> |
    <txp:comment_permlink>
        <txp:comment_id />
    </txp:comment_permlink>
</p>
```

Other tags used: comment_id, comment_message, comment_permlink.

Comment id

On this page:

Syntax

```
<txp:comment_id />
```

The **comment_id** tag is a *single* tag which is used to display the comment’s internal id as assigned by Textpattern at the time of posting. Used in a comments display form.

Attributes

This tag has no attributes.

Examples

Example 1: Comments display form with linked comment id

```
<txp:comment_message />
<p>
    By <txp:comment_name /> at <txp:comment_time />
    <txp:comment_permlink>
        <txp:comment_id />
    </txp:comment_permlink>
</p>
```

Other tags used: comment_message, comment_name, comment_permlink, comment_time.

Comment message input

On this page:

Syntax

```
<txp:comment_message_input />
```

The *comment_message_input tag is a *single* tag which is used to display a text entry field to accept the commenter's message text. Used in the comment input form template.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `cols="integer"`
  HTML `cols` attribute to be applied to HTML form `textarea` field.
  Default: `25`.
- `rows="integer"`
  HTML `rows` attribute to be applied to HTML form `textarea` field.
  Default: `5`.

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comment_email_input, comments_help, comment_name_input, comment_preview, comment_remember, comment_submit, comment_web_input.

Example 2: Text area changes in preview

Using some conditional tags, the size of the comment form `textarea` can be changed in the preview.

```
<txp:if_comments_preview>
    <p>
        This is just a preview of your comment!<br>
        <txp:comment_message_input cols="55" rows="5" /><br>
        <txp:comments_help />
    </p>
<txp:else />
    <p>
        Message (required)<br>
        <txp:comment_message_input cols="55" rows="15" /><br>
        <txp:comments_help />
    </p>
</txp:if_comments_preview>
```

Other tags used: comments_preview, if_comments_preview, else.

Genealogy

Version 4.6.0

`cols` and `rows` attributes added (replaces functionality of deprecated `msgcols` and `msgrows` attributes in comments_form tag).

Comment message

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:comment_message />
```

The **comment_message** tag is a *single* tag which is used to display the message text, or comment. Used in the form that renders your comments (the default form is named ‘comments’).

Attributes

This tag has no attributes.

Examples

Example 1: Comments display form

```
<txp:comment_message />
<p class="footnote">
    <a href="mailto:<txp:comment_email />">Email</a> |
    <txp:comment_permlink>
        <txp:comment_id />
    </txp:comment_permlink>
</p>
```

Other tags used: comment_email, comment_id, comment_permlink.

Comment name input

On this page:

Syntax

```
<txp:comment_name_input />
```

The **comment_name_input** tag is a *single* tag which is used to display a text entry field to accept the commenter’s name. Used in the comment input form template.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `size="integer"`
  HTML `size` attribute to be applied to the HTML form text `input` field.
  Default: `25`.

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comment_email_input, comments_help, comment_message_input, comment_preview, comment_remember, comment_submit, comment_web_input.

Genealogy

Version 4.6.0

`size` attribute added (replaces functionality of deprecated `isize` attribute in comments_form tag).

Comment name

On this page:

Syntax

```
<txp:comment_name />
```

The **comment_name** tag is a *single* tag which is used to display a link using the commenter's name as text. If an email address is supplied and allowed to be viewed, an email link is created. Otherwise, if a website is entered, the website URL is used. If neither is supplied, the name displays as plain text.

Commenter's name and/or email address can be set as a requirement. Should be used in a Textpattern 'comment' type [[Form template]].

Attributes

This tag will accept the following attributes (**case-sensitive**):

- `link="boolean"`
  Make text a link to the author's URL/email (depending on the information given).
  Values: `0` (no) or `1` (yes).
  Default: `0`.

Examples

Example 1: Comments display form

```
<txp:comment_message />
<p>
    <txp:comment_name />
    <txp:comment_time />
    <txp:comment_permlink>
        <txp:comment_id />
    </txp:comment_permlink>
</p>
```

Other tags used: comment_id, comment_message, comment_permlink, comment_time.

Comment permlink

On this page:

Syntax

```
<txp:comment_permlink>
```

The **comment_permlink** tag is a *container* tag which is used to return the permanent link of the article comment being displayed. The container tag wraps the text assigned to the link. Should be used in a Textpattern ‘comment’ type [[Form template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `anchor="boolean"`
  Whether to apply the comment’s ID number to the hyperlink tag (as the `id` attribute), setting this comment permanent link as the comment page anchor.
  Values: `0` (no) or `1` (yes).
  Default: `0`.

Examples

Example 1: Comments display form

```
<txp:comment_message />
<p>
    <txp:comment_name />
    <txp:comment_time />
    <txp:comment_permlink>
        <txp:comment_id />
    </txp:comment_permlink>
</p>
```

Other tags used: comment_id, comment_message, comment_name, comment_time.

Comment preview

On this page:

Syntax

```
<txp:comment_preview />
```

The **comment_preview** tag is a *single* tag which is used to display a ‘Preview’ button the user can use to preview the comment text. Used in the comment input form template.

Attributes

This tag has no attributes.

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comment_email_input, comments_help, comment_message_input, comment_name_input, comment_remember, comment_submit, comment_web_input.

Comment remember

On this page:

Syntax

```
<txp:comment_remember />
```

The **comment_remember** tag is a *single* tag which is used to display a check box input field. If checked the users details are remembered by the system the next time they open a comment form. Used in the comment input form template.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `forgetlabel="text"`
  Label that appears next to the 'Forget' `checkbox`.
  Default: `Forget` (localized).
- `rememberlabel="text"`
  Label that appears next to the 'Remember' `checkbox`.
  Default: `Remember` (localized).

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comment_email_input, comments_help, comment_message_input, comment_name_input, comment_preview, comment_submit, comment_web_input.

Genealogy

Version 4.6.0

`forgetlabel` and `rememberlabel` attributes added (replaces functionality of deprecated `forgetlabel` and `rememberlabel` attributes in comments_form tag).

Comment submit

On this page:

Syntax

```
<txp:comment_submit />
```

The **comment_submit** tag is a *single* tag which is used to display a ‘Submit’ button. Clicking the submit button writes the comment information to the database. Used in the comment input form template.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `label="text"`
  Label that appears on the ‘Submit’ button.
  Default: `Submit` (localized).

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comment_email_input, comments_help, comment_message_input, comment_name_input, comment_preview, comment_remember, comment_web_input.

Genealogy

Version 4.6.0

`label` attribute added (replaces functionality of deprecated `submitlabel` attribute in comments_form tag).

Comment time

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:comment_time />
```

The **comment_time** tag is a *single* tag which is used to display the time and date the comment was submitted. Should be used in a Textpattern ‘comment’ type [[Form template]].

Attributes

This tag will accept the following attributes (**case-sensitive**):

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values, `since`, `iso8601` (ISO 8601 reference), `w3cdtf` (W3CDTF reference), or `rfc822` (RFC 822 reference).
  Default: the ‘Date format’ set in preferences.
- `gmt="boolean"`
  Return either local time (according to the set time zone preferences) or GMT.
  Values: `0` (local time) or `1` (GMT).
  Default: `0`.
- `lang="ISO language code"`
  Format time string suitable for the specified language (locale).
  Values: locales adhere to ISO-639.
  Default: unset (time format set in the Preferences administration panel.

Examples

Example 1: Comments display form

```
<txp:comment_message />
<p>
    <txp:comment_name />
    <txp:comment_time />
    <txp:comment_permlink>
        <txp:comment_id />
    </txp:comment_permlink>
</p>
```

Other tags used: comment_id, comment_message, comment_name, comment_permlink.

Comment web input

On this page:

Syntax

```
<txp:comment_web_input />
```

The **comment_web_input** tag is a *single* tag which is used to display a text entry field to accept the commenter’s domain name. Used in the comment input form template.

Function assumes `http://` for all URLs.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `size="integer"`
  HTML `size` attribute to be applied to the HTML form text `input` field.
  Default: `25`.

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comment_email_input, comments_help, comment_message_input, comment_name_input, comment_preview, comment_remember, comment_submit.

Genealogy

Version 4.6.0

`size` attribute added (replaces functionality of deprecated `isize` attribute in comments_form tag).

Comment web

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:comment_web>
```

The **comment_web** tag can be used as either a *single* or a *container* tag. Thus it may be used as an opening and closing pair:

```
<txp:comment_web>
    ...containing statements...
</txp:comment_web>
```

It is used to display (a link to) the commenter’s web address, if entered at the time of posting. When used as a container tag, it will turn the contents into a link to that web address. Otherwise, it will return the web address. Should be used in a Textpattern ‘comment’ type [[Form template]].

Attributes

This tag has no attributes.

Examples

Example 1: Comments display form with linked website and comment anchor link

```
<txp:comment_message />
<p>
    By <txp:comment_name /> at <txp:comment_time />,
    <a href="<txp:comment_web />">Visit their website</a>
</p>
```

Other tags used: comment_message, comment_name, comment_time.

Example 2: Container example

```
<txp:comment_web>Website</txp:comment_web>
```

Comments count

On this page:

Syntax

```
<txp:comments_count />
```

The **comments_count** tag is a *single* tag which is used to display the number of comments associated with a particular article.

Though **comments_count** can be used independently, it is also called by comments_invite to append the comments count to the **comments_invite** link.

Used in an article form.

Attributes

This tag has no attributes.

Examples

Example 1: Display comment invitation and count

But only if any comments are associated with the current article.

```
<txp:if_comments>
    <p>
        <txp:comments_invite showcount="0" /> |
        <txp:comments_count /> respones received.
    </p>
</txp:if_comments>
```

Other tags used: comments_invite, if_comments.

Comments error

On this page:

Syntax

```
<txp:comments_error />
```

The **comments_error** tag is a *single* tag which is used to produce the current comments error.

Attributes

Tag will accept the following attributes (**case-sensitive**):

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: `comments_error` (see [[class cross-reference]]).
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display comments error when an error exists

```
<txp:if_comments_error>
    <txp:comments_error break="li" wraptag="ul" />
</txp:if_comments_error>
```

Other tags used: if_comments_error.

Comments form

On this page:

Syntax

```
<txp:comments_form />
```

The **comments_form** tag is a *single* tag which is used to display a comment form. Comments will be attached to present individual article as a default, or to the article set by the `id` attribute.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `form="form name"`
  Use specified form template.
  Default: `comment_form`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="CSS class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: `comments_form` (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag to wrap around output, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Give visitors indication of ‘Comments’ status

Comments for articles can be turned on or off at the author’s discretion for any article that is published; by using the following scheme in an article form, you can still have the on/off control over comments while still giving users indication of comment status:

```
<txp:if_comments_allowed>
    <txp:comments_form />
<txp:else />
    <p>Comments are turned off for this article.</p>
</txp:if_comments_allowed>
```

Other tags used: if_comments_allowed, else.

Example 2: Display conditional comments and form

```
<txp:if_comments_allowed>
    <txp:comments form="lineitem" break="li" wraptag="ul" />
    <txp:comments_form />
</txp:if_comments_allowed>
```

And then an example of what the form `lineitem` could contain:

```
<txp:comment_id />
```

For the current article, returns a list of id numbers for comments and a comment input form (but only if comments are currently allowed).

Other tags used: comment_id, comments, if_comments_allowed.

Genealogy

Version 4.6.0

`msgstyle` attribute deprecated.
`forgetlabel`, `isize`, `msgcols`, `msgrows`, `msgstyle`, `previewlabel`, `rememberlabel` and `submitlabel` attributes deprecated and functionality moved to individual `comment_*` input tags.

Version 4.5.0

Added `forgetlabel`, `previewlabel`, `rememberlabel` and `submitlabel` attributes.

Version 4.0.4

`show_preview` attribute removed.

Comments help

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:comments_help />
```

The **comments_help** tag is a *single* tag which is used to display a Textile help link. This tag can be used in a Textpattern page or a Textpattern form.

Attributes

This tag has no attributes.

Examples

Example 1: Comment form

```
<p>
    Name (required)<br>
    <txp:comment_name_input />
</p>
<p>
    Email (required)<br>
    <txp:comment_email_input />
</p>
<p>
    Website<br>
    <txp:comment_web_input />
</p>
<p>
    <txp:comment_remember />
</p>
<p>
    Message (required)<br>
    <txp:comment_message_input /><br>
    <txp:comments_help />
</p>
<p>
    <txp:comment_preview />
    <txp:comment_submit />
</p>
```

Other tags used: comment_email_input, comments_help, comment_message_input, comment_name_input, comment_preview, comment_remember, comment_submit, comment_web_input.

Comments invite

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:comments_invite />
```

The **comments_invite** tag is a *single* tag which is used to display a link to an article comment form. Text used for the link will be taken from the invitation field on the Textpattern Write administration panel.

This tag can be used in both a Textpattern [[Page template]] and a [[Form template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `showalways="boolean"`
  Whether to display invite on individual article page.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `showcount="boolean"`
  Whether to display comment count.
  Values: `0` (no) or `1` (yes).
  Default: `1`.
- `textonly="boolean"`
  Whether to display invite as text, rather than a hyperlink.
  Values: `0` (no) or `1` (yes).
  Default: `0`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: `comments_invite` (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag to wrap around invite text, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display comments invitation and comment count

```
<txp:if_comments>
    <txp:comments_invite wraptag="p" />
</txp:if_comments>
```

This will display the comments invitation and a comment count (but only if there are any comments associated with the current article).

Other tags used: if_comments.

Comments preview

On this page:

Syntax

`<txp:comments_preview />`

The **comments_preview** tag is a *single* tag which is used to display a preview of a visitor’s comment.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `form="form name"`
  Use specified form template to process each comment.
  Default: `comments`.
- `label="text"`
  Label that appears on the ‘Preview’ button.
  Default: `Preview` (localized).

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: `comments_preview` (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag to wrap around the list, specified without brackets (e.g. `wraptag="div"`).
  Default: depends upon ‘Present Comments as a Numbered List?’ preference setting – either `ol` or unset (but see [[wraptag cross-reference]] for exceptions).

Genealogy

Genealogy

Version 4.6.0

`label` attribute added (replaces functionality of deprecated `previewlabel` attribute in comments_form tag).

Version 4.0.4

Use is necessary in comments display form (`<comments_display />`, by default).

Version 4.0.3

Tag support added.

Comments

On this page:

Syntax

```
<txp:comments />
```

The **comments** tag is a *single* tag which is used to display the comments associated with a particular article. Comments will be displayed for the present individual article as a default, or to the article set by the `id` attribute.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `form="form name"`
  Use specified form template to process each comment.
  Default: `comments`.
- `limit="integer"`
  The number of comments to display.
  Default: `0` (no limit).
- `offset="integer"`
  The number of comments to skip.
  Default: `0`.
- `sort="sort value(s)"`
  How to sort the resulting list.
  Values:
  `discussid` (comment ID).
  `email`.
  `ip` (IP address).
  `message`.
  `name`.
  `parentid` (article ID).
  `posted`.
  `rand()` (random).
  `web`.
  Default: `posted asc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `li` or `div`, depends upon Preferences administration panel setting for ‘Present comments as a numbered list?’.
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: `comments` (see [[class cross-reference]]).
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: `ol` or unset, depends upon Preferences administration panel setting for ‘Present comments as a numbered list?’.

Examples

Example 1: Display comments and give an indication of comments status

```
<txp:comments />
```

```
<txp:if_comments_allowed>
    <txp:comments_form />
<txp:else />
    <p>Comments are turned off for this article.</p>
</txp:if_comments_allowed>
```

Comments for articles can be turned on or off at the author's discretion for any article that is published; by using the scheme above in an 'article' type form, you can still have the on/off control over comments while still giving users indication of comment status.

Other tags used: comments_form, else, if_comments_allowed.

Example 2: Conditional comments

```
<txp:if_comments_allowed>
    <txp:comments form="comments" break="li" wraptag="ul" />
    <txp:comments_form />
</txp:if_comments_allowed>
```

And the `comments` form (which is a 'comment' type form):

```
<txp:comment_message />
<p class="footnote">
    <a href="mailto:<txp:comment_email />">Email</a> |
    <txp:comment_permlink>
        <txp:comment_id />
    </txp:comment_permlink>
</p>
```

For the article, list id numbers and a comment input form; but only if comments are currently allowed.

Other tags used: comment_email, comments_form, comment_id, comment_permlink, if_comments_allowed.

Genealogy

Version 4.6.0

`breakclass` attribute deprecated.

Css

On this page:

Syntax

```
<txp:css />
```

The basic **css** tag is a *single* tag used to output the URL of the stylesheet assigned in the Textpattern Sections administration panel.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `format="value"`
  How to format output: either return complete HTML link tag with necessary HTML attributes, or only the stylesheet's URL.
  Values: `link` or `url`.
  Default: `url`.
- `media="value"`
  HTML media attribute to be applied to link tag (when invoked with `format="link"`).
  Default: `screen`.
- `name="style name"`
  Link to specified style.
- `rel="value"`
  HTML rel attribute to be applied to link tag (when invoked with `format="link"`).
  Default: `stylesheet`.
- `title="value"`
  HTML title attribute to be applied to link tag (when invoked with `format="link"`).
  Default: unset.

Examples

Example 1: Output just the stylesheet's URL

```
<head>
    <!-- ...tags... -->
    <link rel="stylesheet" href="<txp:css />" media="screen, projector">
    <!-- ...more tags... -->
</head>
```

Example 2: Output the link to the section's default stylesheet

```
<head>
    <!-- ...tags... -->
    <txp:css format="link" />
    <!-- ...more tags... -->
</head>
```

Example 3: Output the link to a named stylesheet

```
<head>
    <!-- ...tags... -->
    <txp:css format="link" name="style_name" />
    <!-- ...more tags... -->
</head>
```

Example 4: Output print and alternate stylesheets

```
<head>
    <!-- ...tags... -->
    <txp:css format="link" name="plain" rel="alternate" title="Plain and simple style" />
    <txp:css format="link" name="glossy" rel="alternate" title="Glossy style" />
    <txp:css format="link" name="print" media="print" />
    <!-- ...more tags... -->
</head>
```

Genealogy

Version 4.3.0

`n` attribute deprecated and renamed to `name`.

Version 4.0.4

`format`, `media`, `rel` and `title` attributes added.

Custom field

On this page:

Syntax

```
<txp:custom_field />
```

The basic **custom_field** tag is a *single* tag and used to display the contents of a custom field.

Custom fields are useful when you need to output content having a consistent structure, usually in context to a particular type of article. Custom fields are defined in the Preferences administration panel, and used in the Write administration panel. There are conditions to be aware of in each case, so be sure to read the following sections, respectively:

1. [[Defining custom fields]]
2. [[Adding custom field data]]

Also see the if_custom_field conditional tag, which provides more flexibility and power using custom fields.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `default="value"`
  Default value to use when field is empty.
- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&` prior to echoing the field contents.
  Values: `html` or unset.
  Default: `html`;
- `name="fieldname"`
  Display specified custom field.

Examples

Example 1: Book reviews

You might, for example, publish book reviews (for which you add the author, the title of the book, the publishing company and the year of publication), with:

1. a custom field named `Book_Author` containing `J.R.R. Tolkien`,
2. a custom field named `Book_Title` containing `The Lord of the Rings`,
3. a custom field named `Book_Publisher` containing `HarperCollins`,
4. a custom field named `Book_Year` containing `2004`.

and an ‘article’ type form like the following:

```
<p>
    <txp:custom_field name="Book_Author" />: <txp:custom_field name="Book_Title" /><br>
    Published by <txp:custom_field name="Book_Publisher" /> in <txp:custom_field name="Book_Year" />
</p>
```

HTML returned would be:

```
<p>
    J.R.R. Tolkien: The Lord of the Rings<br>
    Published by HarperCollins in 2004.
</p>
```

Example 2: Power A linklog

With an article title of `Textpattern`, an excerpt of `Textpattern is awesome!`, a custom field named `Link` containing `http://textpattern.com/`, and an ‘article’ type form like the following:

```
<article class="linklog-entry">
```

```
    <h1>
        <a href="<txp:custom_field name="Link" />"><txp:title /></a>
    </h1>
    <p>
        <time datetime="<txp:posted format="iso8601" />" itemprop="datePublished">
            <txp:posted format="%d %d %Y" />
        </time>
    </p>
    <txp:excerpt />
</article>
```

HTML returned would be:

```
<article class="linklog-entry">
    <h1>
        <a href="http://textpattern.com/">Textpattern</a>
    </h1>
    <p>
        <time datetime="2005-08-14T15:08:12Z" itemprop="datePublished">14 Aug 2005</time>
    </p>
    <p>Textpattern is awesome!</p>
</article>
```

Other tags used: title, posted, excerpt.

Example 3: Unescaping HTML output

With a custom field named `foo` containing:

```
<a href="../here/">
```

using the following:

```
<txp:custom_field name="foo" />
```

will return this hunk of HTML:

```
&amp;amp;#60;a href=&amp;amp;#34;../here/&amp;amp;#34;&amp;amp;#62;
```

whereas using:

```
<txp:custom_field name="foo" escape="" />
```

will render the URL as you’d expect, exactly as written in the custom field itself. Thus, it will be rendered as a link by the browser.

Else

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:else />
```

The **else** tag is a *single* tag that is used within a containing conditional tag to provide the means to assign default, or alternative, behaviour when the condition in the surrounding tag is **not** met.

Visually, this is the general structure in which it is used:

```
<txp:if_conditional_tag>
    ...Content if true...
<txp:else />
    ...Content if not true...
</txp:if_conditional_tag>
```

Attributes

This tag has no attributes.

Examples

Example 1: Display excerpt when available

```
<txp:if_excerpt>
    <txp:excerpt />
<txp:else />
    <txp:section link="1" />
</txp:if_excerpt>
```

When the excerpt is available it is displayed, but when it is missing a hyperlinked section name is displayed instead.

Other tags used: excerpt, if_excerpt, section.

Email

On this page:

Syntax

```
<txp:email />
```

The **email** tag is both a *single* tag and a *container* tag. Textpattern will replace it with a `mailto:` email link, according to the attributes set.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `email="value"`
  The email address from which to make the link.
  Values: Any valid email address.
  Default: unset.
- `linktext="value"`
  The displayed link text.
  Values: Any text.
  Default: `Contact`.
- `title="value"`
  The `title` attribute to assign to the link.
  Values: Any valid HTML title.
  Default: unset.

Examples

Example 1: Simple email link

```
<txp:email email="donald.swain@example.com" linktext="Contact me" title="Send me an email" />
```

Example 2: Pre-populate message subject and body

```
<txp:email email="donald.swain@example.com?subject=Lorem ipsum&body=Sit amet..." />
```

Example 3: Container example

```
<txp:email email="donald.swain@example.com" title="Email me!">
    <img src="/images/email.png" alt="Email">
</txp:email>
```

Example 4: With Symbolset's 'email' glyph

If you happen to use the 'email' glyph in the social media set of Symbolset, you can still use this tag. Let's say you're creating a social button bar using Symbolset glyphs in a list. The normal way to do this would be to set up your selectors on the individual anchor elements, like the first three list items show below. For the email glyph you need to put the selectors in the `<li>` since you can't put them in the `<a>`, as the last list item shows:

```
<ul class="socbar">
    <li>
        <a href"https://twitter.com/xxx" class="ss-icon twit">twitter</a>
    </li>
    <li>
        <a href="https://plus.google.com/xxx" class="ss-icon gplus">googleplus</a>
    </li>
    <li>
        <a href="http://www.linkedin.com/xxx" class="ss-icon in">linkedin</a>
    </li>
    <li class="ss-icon email">
        <txp:email email="donald.swain@example.com" linktext="email" />
    </li>
</ul>
```

If you’re using Symbolset, then you’ll know that the `linktext=""` attribute value in the last list item above **has** to be `email` for the glyph to work. Then the CSS must be like follows to target both instances of Symbolset glyph use:

```
/* Common rules */
a.ss-icon,
li.ss-icon a {
    /* design as you want */
}

/* Target each one if specific hover effect is wanted */
.twit:hover {
    /* design as you want */
}
... etc ...
li.email a:hover {
    /* design as you want */
}
```

See the feed_link tag for a similar solution for Symbolset’s ‘rss’ glyph.

Genealogy

Version 4.0.5

Can be used as a container.

Error message

On this page:

Syntax

```
<txp:error_message />
```

The **error_message** tag is a *single* tag that Textpattern will replace with the error message text for the error status as set by the server. Should be used in an `error_xxx` or `error_default` Textpattern [[Page template]].

Attributes

This tag has no attributes.

Examples

Example 1: Display error information

```
<h1>
    <txp:error_status />
</h1>
<p>
    <txp:error_message />
</p>
```

With the tags arranged like this (as they are in the `error_default` page template), they display the error status code as a heading and the relevant server message beneath it, usually to indicate to the visitor that something went wrong.

Other tags used: error_status.

Error status

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:error_status />
```

The **error_status** tag is a *single* tag that Textpattern will replace with the error status as set by the server. Should be used in an `error_xxx` or `error_default` Textpattern [[Page template]].

Attributes

This tag has no attributes.

Examples

Example 1: Display error information

```
<h1>
    <txp:error_status />
</h1>
<p>
    <txp:error_message />
</p>
```

With the tags arranged like this (as they are in the `error_default` page template), they display the error status code as a header and the relevant server message beneath it, usually to indicate to the visitor that something went wrong.

Other tags used: error_message.

Excerpt

On this page:

Syntax

```
<txp:excerpt />
```

The **excerpt** tag is a *single* tag which is used to return the excerpt text, if any, associated with the article being displayed.

The conditional tag if_excerpt can be used to check if there is an excerpt.

Attributes

This tag has no attributes.

Examples

Example 1: Excerpt and ‘read more’ button

This example explains how you could display the excerpt in an article list, and excerpt + body in an individual article. Use the following in an article form:

```
<txp:if_article_list>
    <txp:if_excerpt>
        <txp:excerpt />
        <p class="read-more">
            <a href="<txp:permlink />#body" title="<txp:title />">Full article</a>
        </p>
    <txp:else />
        <txp:body />
    </txp:if_excerpt>
<txp:else />
    <txp:if_excerpt>
        <txp:excerpt />
    </txp:if_excerpt>
    <div id="body">
        <txp:body />
    </div>
</txp:if_article_list>
```

Other tags used: body, else, if_article_list, if_excerpt, permlink, title.

Example 2: Display the excerpt text or a default link

Use the following within an article form:

```
<txp:if_excerpt>
    <txp:excerpt />
<txp:else />
    <p>
        Section:
        <txp:section title="1" link="1" />
    </p>
</txp:if_excerpt>
```

Other tags used: else, if_excerpt, section.

Expires

On this page:

Syntax

```
<txp:expires />
```

The **expires** tag is a *single* tag used to indicate when an article should no longer appear in a site, particularly when the information is date sensitive (e.g. events like conferences, meetings and so forth). The tag is defined by expiration date values that are set under the ‘’Date and time’ section of the Write administration panel.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values.
  Default: the ‘Archive date format’ set in preferences.
- `gmt="boolean"`
  Return either local time (according to the set time zone preferences) or GMT.
  Values: `0` (local time) or `1` (GMT).
  Default: `0`.
- `lang="ISO language code"`
  Format time string suitable for the specified language (locale).
  Values: locales adhere to ISO-639.
  Default: unset (time format set in the Preferences administration panel.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `wraptag="element"`
  HTML tag to wrap around output, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Custom format date setting

```
<p>
    Expires:
    <txp:expires format="%b %d, %Y" />
</p>
```

This would result in the following HTML output:

```
<p<Expires: Feb 03, 2014</p>
```

Example 2: Extended custom format date setting

```
<p>
    Expires:
    <time datetime="<txp:posted format="iso8601" />">
        <txp:posted class="time-day" wraptag="span" format="%d" />
        <txp:posted class="time-month" wraptag="span" format="%b" />
        <txp:posted class="time-year" wraptag="span" format="%Y" />
    </time>
</p>
```

This would result in the following HTML output:

```
<p>
    Expires:
    <time datetime="2014-02-03T10:43:39Z">
        <span class="time-day">03</span>
        <span class="time-month">Feb</span>
        <span class="time-year">2014</span>
    </time>
</p>
```

This provides styling hooks for each date part.

Genealogy

Version 4.0.7

Tag support added.

Feed link

On this page:

Syntax

`<txp:feed_link>`

The **feed_link** tag can be used as either a *single* or *container* tag and is used to output a link to the site's 'articles' RSS feed. When used as a container tag, it will turn the contents into a link to the feed, otherwise the value of `label` attribute will be used as link text. Should be used in a Textpattern [[Page template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `category="category name"`
  Restrict to articles from specified category/categories. Note: the category names may be different to the title you typed when you created the category, as the names are sanitized for URL use. Check the Categories administration panel to ensure you are using the correct names.
  Values: (comma separated list of) category name(s).
  Default: current category.
- `flavor="value"`
  Whether to output a link to the RSS or Atom version of the feed.
  Values: `rss` or `atom`.
  Default: `rss`.
- `format="value"`
  Whether to output HTML `<a>` tag or `<link>` tag.
  Values: `a` or `link`.
  Default: `a`.
- `limit="integer"`
  Number of articles to display in the feed.
  Default: depends upon Preferences administration panel setting.
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: current section.
- `title="value"`
  HTML title attribute to be applied to link tag.
  Default: depends upon `flavor` used, either `RSS feed` or `Atom feed`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `wraptag="element"`
  HTML element to wrap feed link, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Note: `wraptag` is applicable only when using `format` of `a`.

Examples

Example 1: Display an RSS feed link for specific section and category

```
<txp:feed_link flavor="rss" section="about" category="general" label="XML" wraptag="p" />
```

Example 2: Container example

```
<txp:feed_link wraptag="p">
    <img src="/path/to/rss-icon.png" alt="RSS">
</txp:feed_link>
```

Example 3: Site wide generic RSS feed

```
<txp:feed_link section="" category="" />
```

Creates a link to the site’s feed for articles in all sections and categories. If you omit the `section` and `category` attributes, the feed will default to the current section/category.

Example 4: With Symbolset’s ‘rss’ glyph

If you happen to use the ‘rss’ glyph in the social media set of Symbolset, you can still use this tag. Let’s say you’re creating a social button bar using Symbolset glyphs in a list. The normal way to do this would be to set up your selectors on the individual anchor elements, like the first three list items show below. For the RSS glyph you need to put the selectors in the `<li>` since you can’t put them in the `<a>`, as the last list item shows:

```
<ul class="socbar">
    <li>
        <a href"https://twitter.com/xxx" class="ss-icon twit">twitter</a>
    </li>
    <li>
        <a href="https://plus.google.com/xxx" class="ss-icon gplus">googleplus</a>
    </li>
    <li>
        <a href="http://www.linkedin.com/xxx" class="ss-icon in">linkedin</a>
    </li>
    <li class="ss-icon rss">
        <txp:feed_link flavor="rss" section="articles" category="" label="rss" />
    </li>
</ul>
```

If you’re using Symbolset, then you’ll know that the `label=""` attribute value in the last list item above **has** to be `rss` for the glyph to work. If you try and put the two Symbolset `class` attribute values in the **feed_link** tag using its `class` attribute, it won’t work, unfortunately. But putting them in the `<li>` element, like shown above, does work.

There is no `atom` trigger word in Symbolset! So while you can use `flavor="atom"` and create an Atom feed just fine, you still need to use `label="rss"` for the link label to call the Symbolset glyph. This shouldn’t be a problem because the glyph replaces the link text. You can then use `title=""` to provide a custom hover text, or leave it out for the default display: ‘Atom feed’.

Then the CSS must be like follows to target both instances of Symbolset glyph use:

```
/* Common rules */
a.ss-icon,
li.ss-icon a {
    /* design as you want */
}

/* Target each one if specific hover effect is wanted */
.twit:hover {
    /* design as you want */
}
... etc ...
li.email a:hover {
    /* design as you want */
}
```

See the email tag for a similar solution for Symbolset’s ‘email’ glyph.

Genealogy

Version 4.3.0

`class` attribute added.

Version 4.0.4

`format` attribute added.

File download author

On this page:

Syntax

```
<txp:file_download_author />
```

The **file_download_author** tag is a *single* tag that Textpattern will replace with the author’s name associated with the current download in a file_download. Can **only** be used inside `<txp:file_download />`.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="link type (boolean)"`
  Whether to hyperlink the author or not.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `section="section name"`
  Direct any linked author name to the nominated section instead of to the default (front) page.
  Default: unset.
- `this_section="boolean"`
  If set to `1`, the linked author name will direct users to an author list in the current section, otherwise author list from all sections is displayed.
  Values: `0` (no, all sections) or `1` (yes, this section only).
  Default: `0`.
- `title="boolean"`
  Whether to display the author’s real name or login name.
  Values: `0` (login name) or `1` (real name).
  Default: `1`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: unset (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag (specified without brackets) to wrap around the author name (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display and link the file author

```
<p>
    File author:
    <txp:file_download_author link="1" />
</p>
```

Genealogy

Version 4.3.0

Tag support added.

File download category

On this page:

Syntax

```
<txp:file_download_category />
```

The **file_download_category** tag is a *single* tag that Textpattern will replace with the category of the file to download. Should be used in a Textpattern ‘file’ type [[Form template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `title="boolean"`
  Whether to display the category name or its title.
  Values: `0` (name), or `1` (title).
  Default: `0`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: unset (see [[class cross-reference]]).
- `wraptag="element"`
  HTML tag to wrap around category text, specified without brackets (e.g. `wraptag="p"`).
  Default: unset.

Examples

Example 1: Display a file category name

```
<p>
    File category:
    <txp:file_download_category />
</p>
```

File download created

On this page:

Syntax

```
<txp:file_download_created />
```

The **file_download_created** tag is a *single* tag that Textpattern will replace with the upload date of the file to download. Should be used in a Textpattern ‘file’ type [[Form template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values, `since`, `iso8601` (ISO 8601 reference), `w3cdtf` (W3CDTF reference), or `rfc822` (RFC 822 reference).
  Default: the ‘Archive date format’ set in preferences.

Examples

Example 1: Display formated file upload date

```
<p>
    File created:
    <txp:file_download_created format="%D" />
</p>
```

Returns the file creation date in the format `mm/dd/yy`.

File download description

On this page:

Syntax

```
<txp:file_download_description />
```

The **file_download_description** tag is a *single* tag which Textpattern will replace with the description of the file to download, as defined when the file was uploaded. Should be used in a Textpattern ‘file’ type [[Form template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&` for the file’s `description` attributes.
  Values: `html` or unset.
  Default: `html`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: unset (see [[class cross-reference]]).
- `wraptag="element"`
  HTML tag to wrap around description text, specified without brackets (e.g. `wraptag="div"`).
  Default: unset.

Examples

Example 1: Display a file’s description

```
<p>
    File description:
    <txp:file_download_description />
</p>
```

Other tags used: text.

Genealogy

Version 4.0.7

Default value for `escape` attribute changed from ‘unset’ to `html`.

File download downloads

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:file_download_downloads />
```

The **file_download_downloads** tag is a *single* tag that Textpattern will replace with the number of times the current file has been downloaded. Should be used in a Textpattern ‘file’ type [[Form template]].

Attributes

This tag has no attributes.

Examples

Example 1: Display the number of downloads

```
<p>
    Downloads:
    <txp:file_download_downloads />
</p>
```

File download id

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:file_download_id />
```

The **file_download_id** tag is a *single* tag that Textpattern will replace with the internal ID number of the file to be downloaded. Should be used in a Textpattern ‘file’ type [[Form template]].

Attributes

This tag has no attributes.

Examples

Example 1: Display a file id

```
<p>
    File ID number:
    <txp:file_download_id />
</p>
```

File download link

On this page:

Syntax

```
<txp:file_download_link />
```

The **file_download_link** tag is both a *single* tag and a *container* tag. Thus it may be used as an opening and closing pair:

```
<txp:file_download_link>
    ...containing statements...
</txp:file_download_link>
```

When used as a single tag, Textpattern will replace the tag with a download link to the file being downloaded. As a container, it will assign the link to the given text or tag, while the single tag outputs the file’s plain URL.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `filename="text"`
  Specifies which file to display by its `filename` as shown on the Files administration panel.
- `id="integer"`
  Specifies the `id`, assigned at upload of the file, to display. Can be found on the Files administration panel.

Note: `id` takes precedence over `filename`. If neither is defined and the tag is not used within the context of a file, nothing is returned.

Examples

Example 1: Provide a link to download file #4

```
<txp:file_download_link id="4">
    <txp:file_download_name />
    [<txp:file_download_size format="mb" decimals="2" />]
</txp:file_download_link>
```

Makes a link to the given file (#4) comprising its file name and size.

Other tags used: file_download_name, file_download_size.

File download list

On this page:

Syntax

```
<txp:file_download_list />
```

The **file_download_list** tag is a *single* or a *container* tag which is used to produce a list of download links according to the given attributes. Each file in the list is formatted by the file tags used in the given form (default is the `files` form).

If used as a container, it must be specified as an opening and closing pair of tags, like this:

```
<txp:file_download_list>
    ...contained statements...
</txp:file_download_list>
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `author="author login name"`
  Restrict to files with the specified author.
  Default: unset.
- `auto_detect="string context"`
  List of Textpattern contexts to consider when automatically searching for files. If you wish to turn off the automatic check, set this to `auto_detect=""`.
  Values: `category` (to look in the URL for a category list) and/or `author` (to look in the URL for an author list).
  Default: `category, author`.
- `category="category name"`
  Restrict to files from the specified category. Allows a comma separated list of category names. Note: category names may be different to the Title you typed when you created the category, as the names are sanitized for URL use. Check the Categories administration panel to ensure you are using the correct names.
  Default: unset.
- `form="form name"`
  Use the specified form template to process the files.
  Default: `files`.
- `id="file ID"`
  Display the specific file or list of files.
  Values: (comma separated list of) file ID.
  Default: unset.
- `limit="integer"`
  Number of files to display.
  Default: `10`.
- `offset="integer"`
  Number of files to skip.
  Default: unset.
- `pageby="integer or limit"`
  Number of files to jump each page. Without this attribute, you cannot navigate using the newer and older tags. Usually you will want to track the `limit` attribute. Use `pageby="limit"` to do this, which means you will not have to amend two values if you subsequently decide to alter the `limit`.
  Default: unset.
- `realname="author real name"`
  Restrict to files with the specified author name.
  Default: unset.
- `sort="by what and order"`
  How to sort the resulting list.
  Values:
  `id`
  `filename`

    `title`
    `category`
    `description`
    `downloads`
    `created`
    `modifie`
    `rand()` (random).
    Adding a space and then one of either `asc` or `desc` orders by ascending or descending value, respectively.
    Default: `filename asc`.
- `status="file status"`
  Restrict to files with the specified status.
  Values: `hidden`, `pending`, `live`.
  Default: `live`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display a list of the ten most popular downloads, in descending order

```
<txp:file_download_list limit="10" break="li" wraptag="ol" sort="downloads desc" />
```

Genealogy

Version 4.3.0

`pageby` attribute added to enable paging via newer and older.
`author` attribute added.
`realname` attribute added.
`auto_detect` attribute added to allow automatic (URL-based) contextual listings.

Version 4.2.0

`id` attribute added.

Version 4.0.7

Can be used as a container tag.

Version 4.0.6

Support added for comma separated list for `category` attribute.

File download modified

On this page:

Syntax

```
<txp:file_download_modified />
```

The **file_download_modified** tag is a *single* tag that Textpattern will replace with the last modified date of the file to download. Should be used in a Textpattern ‘file’ type [[Form template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values, `since`, `iso8601` (ISO 8601 reference), `w3cdtf` (W3CDTF reference), or `rfc822` (RFC 822 reference).
  Default: the ‘Archive date format’ set in preferences.

Examples

Example 1: Display formated file modified date

```
<p>
    File last modified:
    <txp:file_download_modified format="%D" />
</p>
```

Returns the file creation date in the format `mm/dd/yy`.

File download name

On this page:

Syntax

```
<txp:file_download_name />
```

The **file_download_name** tag is a *single* tag that Textpattern will replace with the name of the file to download. Should be used in a Textpattern ‘file’ type [[Form template]]. or within a file_download_link tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `title="boolean"`
  Whether to display the file download name or its title.
  Values: `0` (name), or `1` (title).
  Default: `0`.

Examples

Example 1: Display the name of a file, linked to download

```
<txp:file_download_link filename="my-presentation.pdf">
    <txp:file_download_name />
    [<txp:file_download_size format="mb" decimals="2" />]
</txp:file_download_link>
```

Other tags used: file_download_link, file_download_size.

Genealogy

Version 4.3.0

`title` attribute added.

File download size

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:file_download_size />
```

The **file_download_size** tag is a *single* tag which Textpattern will replace with the formatted file size of the file to be downloaded. Should be used in a Textpattern 'file' type [[Form template]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `decimals="places"`
  Number of decimal places to format the value to.
  Default: `2`.
- `format="numbering style"`
  The way to represent the number, based on the file's expected size.
  Values:
  `b` (bytes)
  `k` (kilobytes)
  `m` (megabytes)
  `g` (gigabytes)
  `t` (terabytes)
  `p` (petabytes)
  `e` (exabytes)
  `z` (zettabytes)
  `y` (yottabytes)
  Default: unset (i.e.,the most appropriate units based on the file size).

Examples

Example 1: Display formatted file size in kilobytes

```
<txp:file_download_size format="k" />
```

File download

On this page:

Syntax

```
<txp:file_download />
```

The **file_download** tag is a *single* tag which Textpattern will replace with a Textpattern ‘file’ type [[Form template]]. Inside that form go the other [[file tags]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `filename="name"`
  Filename of the file to link to.
  Default: unset (nothing is returned).
- `form="form name"`
  Use specified form template to process the files.
  Default: `files`.
- `id="integer"`
  File `id` of the file to link to.
  Default: unset (nothing is returned).

Examples

Example 1: Display a download form

```
<txp:file_download form="new-files" id="1" />
```

Gets file with `id` of `1` and displays results using `new-files` form.

Hide

On this page:

Syntax

```
<txp:hide />
```

The **hide** tag is a *container* tag which is used to suppress the interpretation of all enclosed contents. Use it for comments, temporary concealment of article text parts or non-destructive form changes.

Attributes

This tag has no attributes.

Examples

Example 1: Insert a useful note in a template

```
<txp:hide>This is a work-around for a bug in Internet Explorer, again</txp:hide>
```

Example 2: Comment out part of a form for testing

If you want to try something out to see how it affects the layout without actually deleting the content, wrap it in **hide** tags:

```
<div class="entry-content">
    <txp:body />
</div>
<txp:hide>
    <address class="vcard author">
        <span class="fn">
            <txp:author />
        </span>
    </address>
    <txp:comments_invite wraptag="p" />
</txp:hide>
```

Renders the body text inside the `<div class="entry-content">` but skips the `<address>` and `<txp:comments_invite />` tags.

Other tags used: body, author, comments_invite.

If article author

On this page:

Syntax

```
<txp:if_article_author>
```

The **if_article_author** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_article_author>
    ...conditional statement...
</txp:if_article_author>
```

The tag will execute the contained statement if the author name associated with a particular article matches the value of the `name` attribute. Should be used in an 'article' type form.

The `name` attribute requires an author's login name **not** their real name.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="author"`
  Comma-separated list of author names.
  Default: unset (i.e.,any author at all).

Examples

Example 1: Display some text dependent on an article's author

```
<p>
    <txp:author />
    <txp:if_article_author name="admin">
        (Administrator)
    </txp:if_article_author>
</p>
```

Displays article author name, then displays the text "(Administrator)" if the article was written (posted) by the author `admin`.

If article category

On this page:

Syntax

```
<txp:if_article_category>
```

The **if_article_category** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_article_category>
    ...conditional statement...
</txp:if_article_category>
```

It will execute the contained statement if the category name associated with a particular article (Category1 or Category2) matches the values of the name and number attributes. Should be used in an 'article' type Textpattern [Form]] template.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="category"`
  Comma-separated list of category names (not titles) to match. Note the category name is specified in lower case regardless of how you typed its title in the Categories administration panel. Also note that if you had called your category 'My Category Name' it becomes `my-category-name` when used in tags.
  Default: unset.
- `number="number"`
  Match category in Category1 or Category2 (or both).
  Values: `1` or `2`.
  Default: unset, causing both categories to be matched against the specified name.

Examples

Example 1: Display matched category

```
<txp:if_article_category name="prose" number="1">
    <p>
        <txp:category1 />
    </p>
</txp:if_article_category>
```

If the Category1 assigned to the article is 'Prose', the category is displayed. Note that the category **name** is used in this tag, which may be different to its displayed category **Title**. When categories are created, Textpattern converts them to lowercase and replaces spaces with hyphens. So, for example, 'My Category' has a name `my-category`.

Other tags used: category1.

Example 2: Using the tag with else

```
<txp:if_article_category name="algebra" number="1">
    <p>Fun with algebra</p>
<txp:else />
    <p>
        <a href="/">Home</a>
    </p>
</txp:if_article_category>
```

Displays the welcome text if the category and category number match the given values, or shows a default link otherwise.

Other tags used: else.

Example 3: Display a list of matching links

In an article form, put the following set of conditionals for each category you want to look for:

```
<txp:if_article_category name="yourcategory" number="1">
    <txp:article_custom category="yourcategory" sort="Posted asc" wraptag="ul" break="li">
        <txp:permlink>
            <txp:title />
        </txp:permlink>
    </txp:article_custom>
</txp:if_article_category>
```

Lists articles of the same category as the current article’s Category1.

Other tags used: article_custom, else, title, permlink.

If article id

On this page:

Syntax

```
<txp:if_article_id>
```

The **if_article_id** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_article_id>
    ...conditional statement...
</txp:if_article_id>
```

The tag will execute the contained statement if the article id associated with a particular article matches the id attribute. Should be used in an article form/container. The `id` attribute **must** be used in an **article list context** (when producing a page that displays more than one article) or the tag will do nothing.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `id="integer"`
  Comma delimited integer article ID list.
  Default: current article's ID if available (i.e.,on a page that displays a single article).

Examples

Example 1: Display info if the article id matches

```
<txp:if_article_id id="33">
    <p>
        <txp:title />
    </p>
</txp:if_article_id>
```

Displays the article title if the id of the current article is 33.

Other tags used: title.

Example 2: Display a list of articles omitting current article

```
<txp:article_custom label="related" labeltag="h4" section="<txp:section />" wraptag="ul">
    <txp:if_article_id>
    <txp:else />
        <li>
            <txp:permlink>
                <txp:title />
            </txp:permlink>
        </li>
    </txp:if_article_id>
</txp:article_custom>
```

Displays an unordered linked list of articles from the same section omitting the article currently viewed.

Other tags used: article_custom, section, else, permlink, title.

Genealogy

Version 4.0.7

Defaults to the current article's ID.

If article image

On this page:

Syntax

```
<txp:if_article_image>
```

The **if_article_image** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_article_image>
    ...conditional statement...
</txp:if_article_image>
```

The tag will execute the contained statements if an image is associated (through the Write administration panel 'Article image' field) with the article being displayed.

Attributes

This tag has no attributes.

Examples

Example 1: Display default image if no article image exists

```
<txp:if_article_image>
    <p>
        <txp:article_image />
    </p>
<txp:else />
    <p>
        <txp:image id="5" />
    </p>
</txp:if_article_image>
```

Other tags used: article_image, else, image.

Example 2: Integration with third-party PHP resizing script (TimThumb)

TimThumb is a simple, flexible, PHP script that resizes images directly on your web server. Read the TimThumb documentation for basic installation instructions (also requires the GD image library). Then, for example, you can use the following:

```
<txp:if_article_image>
    <p>
        <txp:images limit="1">
            <img src="<txp:site_url />timthumb.php?src=<txp:image_url />&amp;w=640" alt="<txp:image_info type='alt' />">
        </txp:images>
    </p>
</txp:if_article_image>
```

Checks an article image exists, then uses `<txp:images>` with `limit="1"` to display that image (because the `<txp:images>` tag takes the article image as a first priority). Uses the TimThumb script to proportionately resize a 640px wide version of the image automatically, and keep a cached version of the resized image for future visitors.

Other tags used: images, image_info, image_url, site_url.

Genealogy

Version 4.2.0

Tag support added.

If article list

On this page:

Syntax

```
<txp:if_article_list>
```

The **if_article_list** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_article_list>
    ...conditional statement...
</txp:if_article_list>
```

The tag will execute the contained statement if an article list is being displayed (i.e.,not showing an individual article).

Attributes

This tag has no attributes.

Examples

Example 1: Article/article list navigation

```
<txp:article />

<txp:if_individual_article>
    <p>
        <txp:link_to_prev>
            <txp:prev_title />
        </txp:link_to_prev>
        <txp:link_to_next>
            <txp:next_title />
        </txp:link_to_next>
    </p>
</txp:if_individual_article>

&lt;txp:if_article_list&gt;
    <p>
        <txp:older>Previous</txp:older>
        <txp:newer>Next</txp:newer>
    </p>
</txp:if_article_list>
```

This example shows how to setup article navigation so that link_to_prev and link_to_next are used at the individual article level *or* older and newer with article lists.

Other tags used: link_to_prev, link_to_next, prev_title, next_title, if_individual_article, older, newer.

Example 2: In combination with the ‘else’ tag

```
<txp:if_article_list>
    <p>
        <txp:site_name />
    </p>
<txp:else />
    <p>
        <img src="http://example.com/logo.png" alt="Logo">
    </p>
</txp:if_article_list>
```

This example shows the if_article_list in combination with else to display a site’s site_name or logo when an article list is displayed or not, respectively.

Other tags used: else, site_name.

If article section

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:if_article_section>
```

The **if_article_section** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_article_section>
    ...conditional statement...
</txp:if_article_section>
```

The tag will execute the contained statements if the section name associated with a particular article matches the value of the `name` attribute. Should be used in an article form.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="section"`
  Comma-separated list of section names.

Examples

Example 1: Check the article's section

```
<txp:if_article_section name="poetry">
    <p>by <txp:author /></p>
</txp:if_article_section>
```

Displays the author name if the current article belongs to the section named `poetry`.

Other tags used: author.

Example 2: Using the tag with else

```
<txp:if_article_section name"poetry">
    <p>Fun With poetry</p>
<txp:else />
    <p><a href="index.php">Home</a></p>
</txp:if_article_section>
```

Display the welcome text if the article's section matches `poetry`, or shows a default link otherwise.

Other tags used: else.

If author

On this page:

Syntax

```
<txp:if_author>
```

The **if_author** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_author>
    ...conditional statement...
</txp:if_author>
```

The tag will execute the contained statement **if** the called page is the result of an article search by a specific author's name.

This is **not** the same as checking if the current article was written (posted) by the given author. Use if_article_author for that situation.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="author"`
  Comma-separated list of author names.
  Default: unset, which determines whether 'any' author listing is being viewed.
- `type="context"`
  Textpattern context to check against. You can choose from the following contexts (set to empty to include all contexts):
  `article` is this an article author list?
  `image` is this an image author list?
  `file` is this a file author list?
  `link` is this a link author list?
  Default: `article`.

Examples

Example 1: Select a stylesheet based on author

Selects a stylesheet named `author_list` when a list by author `admi` is being displayed, or a stylesheet determined by the active section for normal page display.

```
<txp:if_author name="admin">
    <link rel="stylesheet" href="<txp:css name="author_list" />" media="screen">
<txp:else />
    <link rel="stylesheet" href="<txp:css />" media="screen">
</txp:if_author>
```

Other tags used: else, css.

Genealogy

Version 4.3.0

`type` attribute added.

If category

On this page:

Syntax

```
<txp:if_category>
```

The **if_category** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_category>
    ...conditional statement...
</txp:if_category>
```

The tag will execute the contained statements if the `name` attribute matches a category search value, or the list is an article list by category.

Should be used in a page template; if checking the category in an article form, use if_article_category.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="category"`
  Comma-separated list of category names. Note the category name is specified in lower case regardless of how you typed its title in the Categories administration panel. Also note that if you had called your category 'My Category Name' it becomes 'my-category-name' when used in tags.
  Default: unset, which determines whether "any" category listing is being viewed.
- `type="context"`
  Textpattern context to check against. You can choose from the following contexts (set to empty to include all contexts):
  `article` is this an article category list?
  `image` is this an image category list?
  `file` is this a file category list?
  `link` is this a link category list?
  Default: `article`.

Examples

Example 1: Display info depending on list category

```
<txp:if_category name="prose">
    <p><txp:author /></p>
</txp:if_category>
```

Displays the author's name if the article list is of category `prose`.

Other tags used: author.

Example 2: Use tag with else

```
<txp:if_category name="prose">
    <p><txp:category /></p>
<txp:else />
    <h3><txp:site_name /></h3>
</txp:if_category>
```

Displays the category name if the article list is of category `prose`, otherwise show the site's name.

Other tags used: category, else, site_name.

Example 3: Display an appropriate heading

```
<;txp:if_category>
   <h3>Articles in category <txp:category title="1" /></h3>
<txp:else />
   <h3>All articles</h3>
</txp:if_category>
```

Displays an appropriate heading for both category and non-category pages.

Other tags used: category, else.

Example 4: Display a category/article list

Given the defined article categories: `prose`, `poetry`, and `opinions`.

```
<txp:category_list label="Category Navigation" wraptag="p" />

<txp:if_category name="prose">
    <txp:recent_articles limit="25" break="li" wraptag="ol" label="Prose" category="prose" />
</txp:if_category>

<txp:if_category name="poetry">
    <txp:recent_articles limit="25" break="li" wraptag="ol" label="Poetry" category="poetry" />
</txp:if_category>

<txp:if_category name="opinions">
    <txp:recent_articles limit="25" break="li" wraptag="ol" label="Opinions" category="opinions" />
</txp:if_category>
```

Shows a category list and, underneath it, a list of related articles in the currently selected category. Changing the category using the list changes the related articles underneath.

Other tags used: category_list, recent_articles.

Genealogy

Version 4.3.0

`type` attribute added.

If comments allowed

On this page:

Syntax

```
<txp:if_comments_allowed>
```

The **if_comments_allowed** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_comments_allowed>
    ...conditional statement...
</txp:if_comments_allowed>
```

The tag will execute the contained statements if comments are allowed for a given article.

This tag can be used in pages, or in article or comment forms; though it will allow you to use an `id` attribute in a comment form, the default behaviour (no attribute) will ensure consistency in comment/article matching when viewing an individual article.

When used in a page template, it will test the article identified by the attribute for status and act, or not, according to that status. It will not pass the `id` attribute to the contained statement, such as comments or comments_form, they must be added as attributes to the contained tag.

This tag is mainly used in combination with if_comments_disallowed.

Attributes

This tag has no attributes.

Examples

Example 1: Give an indication of comments status

```
<txp:if_comments_allowed>
    <txp:comments_form />
</txp:if_comments_allowed>

<txp:if_comments_disallowed>
    <p>Comments are turned off for this article.</p>
</txp:if_comments_disallowed>
```

Comments for articles can be turned off or on at the author’s discretion for any article that is published; by using the following scheme in an article form, you can still have the on/off control over comments while also giving an indication of the comment status.

Other tags used: comments_form, if_comments_disallowed.

Example 2: Display a list of IDs

```
<txp:if_comments_allowed>
    <txp:comments form="lineitem" break="li" wraptag="ul" />
<txp:else />
    <p>Comments closed.</p>
</txp:if_comments_allowed>
```

And the form ‘lineitem’ (type: comment):

```
<txp:comment_id />
```

Displays a list of id numbers for comments on the current article, if comments are currently allowed.

Other tags used: comments, else, comment_id.

If comments disallowed

On this page:

Syntax

```
<txp:if_comments_disallowed>
```

The **if_comments_disallowed** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_comments_disallowed>
    ...conditional statement...
</txp:if_comments_disallowed>
```

The tag will execute the contained statements if comments are disallowed for a given article.

The **if_comments_disallowed** tag can be used in pages, and in article and comment forms. When used in a page template, it will test the article identified by the attribute for status and act, or not, according to that status. It will not pass the `id` attribute to the contained statement, such as comments or comments_form; they must be added as attributes to the contained tag.

Although you can use an `id` attribute in a comment form, the default behaviour (no attribute) will ensure consistency in comment/article matching when viewing an individual article.

This tag is mainly used in combination with if_comments_allowed.

Attributes

This tag has no attributes.

Examples

Example 1: Give an indication of comments status

```
<txp:if_comments_allowed>
    <txp:comments_form />
</txp:if_comments_allowed>

<txp:if_comments_disallowed>
    <p>Comments are turned off for this article.</p>
</txp:if_comments_disallowed>
```

Comments for articles can be turned off or on at the author’s discretion for any article that is published; by using the following scheme in an article form, you can still have the on/off control over comments while also giving an indication of the comment status.

Other tags used: comments_form, if_comments_disallowed.

Example 2: List of comment links

```
<txp:if_comments_disallowed>
    <txp:comments form="lineitem" break="li" wraptag="ul" />
</txp:if_comments_disallowed>
```

And the form ‘lineitem’ (type: comment):

```
<txp:comment_permlink>
    <txp:comment_id />
</txp:comment_permlink>
```

Displays a list of links to comments for the current article, using the comment `id` as text, but only if comments are currently **not allowed**.

Other tags used: comments, comment_permlink, comment_id.

If comments error

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:if_comments_error>
```

The **if_comments_error** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_comments_error>
    ...conditional statement...
</txp:if_comments_error>
```

The tag will execute the contained statements when an error exists with the comments.

Attributes

This tag has no attributes.

Examples

Example 1: Display comments error when an error occurs

```
<txp:if_comments_error>
    <txp:comments_error />
</txp:if_comments_error>
```

Other tags used: comments_error.

If comments preview

On this page:

Syntax

```
<txp:if_comments_preview>
```

The **if_comments_preview** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_comments_preview>
    ...conditional statement...
</txp:if_comments_preview>
```

The tag will execute the contained statements if a comment is being previewed.

Attributes

This tag takes no attributes:

Examples

Example 1: Display text prompt when comment is previewed

```
<txp:if_comments_preview>
    <p>
        <strong>Please review your comment before submitting.</strong>
    </p>
</txp:if_comments_preview>
```

If comments

On this page:

Syntax

```
<txp:if_comments>
```

The **if_comments** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_comments>
    ...conditional statement...
</txp:if_comments>
```

The tag will execute the contained statements if there are one or more comments associated with a particular article. Should be used in an ‘article’ type form.

Attributes

This tag has no attributes.

Examples

Example 1: Number of comments associated with an article

```
<txp:if_comments>
    <p>
        Comments:
        <txp:comments_count />
    <p>
<txp:else />
    <p>No comments.</p>
</txp:if_comments>
```

Displays the number of comments written against the current article, otherwise display text to indicate there are no comments.

Other tags used: comments_count, else.

If custom field

On this page:

Syntax

```
<txp:if_custom_field>
```

The **if_custom_field** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_custom_field>
    ...conditional statement...
</txp:if_custom_field>
```

The tag will execute the contained statements if one or more custom fields for a given article have content. The contents of a custom field can be displayed with the custom_field tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `match="match type"`
  How you wish your value to be tested. Choose from:
  `exact`: value text must exactly match the custom field.
  `any`: checks if any of the given comma-separated list of @value@s occur anywhere in the custom field.
  `all`: checks if all of the given comma-separated list of @value@s occur anywhere in the custom field.
  `pattern`: allows you to specify a regular expression in your `value` attribute to match against the custom field.
  Default: `exact`.
- `name="field name"`
  The custom field name you wish to check.
- `separator="character"`
  If you wish to treat your custom field as a list of items – so that each item is a discrete entity and tested separately when using `any` or `all` matching – specify the delimiter that you use in the custom field. This attribute is ignored if using `exact` or `pattern` matching.
- `value="field value"`
  The custom field content you want to check for a match.

Examples

Example 1: Display contents of custom fields

```
<txp:if_custom_field name="subtitle">
    <txp:custom_field name="subtitle" />
</txp:if_custom_field>
```

Checks if a custom field has any content (at all) and display it.

Why might you do it? Say, you are publishing book reviews on your site and you use custom fields to enter the author, title, publisher and year of publication. Some of the books have a subtitle, others don’t, so the conditional checks if the custom field you named `subtitle` holds any content and if it does, it will be displayed. If it’s empty, the field won’t turn up on the page.

The whole set of custom fields could look like this:

```
<h4>
    <txp:custom_field name="author" />:
    <txp:custom_field name="title" />
</h4>
<txp:if_custom_field name="subtitle">
    <p>
        <txp:custom_field name="subtitle" />
    </p>
```

```
</txp:if_custom_field>
<p>
    Published by
    <txp:custom_field name="publisher" />
    in
    <txp:custom_field name="year" />
</p>
```

For a **book that has a subtitle**, this may be seen:

```
<h4>Stephen Covey: The Seven Habits of Highly Effective People</h4>
<p>Powerful lessons in personal change</p>
<p>Published by Simon &amp; Schuster in 2002.</p>
```

For a **book without a subtitle**, this might be shown:

```
<h4>J.R.R. Tolkien: The Lord of the Rings</h4>
<p>Published by HarperCollins in 2004.</p>
```

Other tags used: custom_field.

Example 2: Check custom field value

A mood indicator:

```
<txp:if_custom_field name="mood" value="happy">
    <img src="happy-face.png" alt="Happy">
</txp:if_custom_field>

<txp:if_custom_field name="mood" value="sad">
    <img src="sad-face.png" alt="Sad">
</txp:if_custom_field>
```

Checks the content of the custom field named `mood` to see if it matches the text `happy` or `sad`. Depending which one it matches determines which of the emoticon images is displayed.

Why might you do it? If you define a custom field `mood`, you can enter a word to indicate your mood while writing an article. You enter either `happy` or `sad`.

Example 3: Use the tag with else

```
<txp:if_custom_field name="website">
    <p>
        <txp:custom_field name="website" />
    </p>
<txp:else />
    <p>Unfortunately, this band hasn't got a website.</p>
</txp:if_custom_field>
```

If the custom field named ‘website’ has some content, display it, otherwise display a standard message.

Why might you do it? If you publish music reviews and you’ve set up some custom fields for the band name, the album title and the band’s website. But not all bands have a website and you want to display a standard message if a band hasn’t got one.

Other tags used: custom_field, else.

Example 4: Display a conditional statement based on a comma separated value

```
<txp:if_custom_field name="animals" separator="," match="any" value="monkeys">
    <p>The monkeys are eating bananas.</p>
<txp:else />
    <p>The bears ate all the monkeys. How sad.</p>
</txp:if_custom_field>
```

Checks the content of the custom field named `animals` which has a comma separated list of animals. It checks if it contains the text `monkeys`, and displays a conditional statement if it does.

Other tags used: else.

Genealogy

Version 4.3.0

`val` attribute deprecated and renamed to `value`.
`matc` and `separator` attributes added.

If description

On this page:

Syntax

```
<txp:if_description>
```

The **if_description** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_description>
    ...conditional statement...
</txp:if_description>
```

The tag will execute the contained statement if the current article/section/cateogry's 'Description' field has an entry.

Attributes

This tag has no attributes.

Examples

Example 1: Supply meta tag if description exists

```
<head>
    ....
    <txp:if_description>
        <txp:meta_description />
    <txp:else />
        <meta name="description" content="A generic description fallback, possibly about bacon." />
    </txp:if_description>
    ....
</head>
```

Other tags used: meta_description.

Genealogy

Version 4.6.0

Tag support added

If different

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:if_different>
```

The **if_different** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_different>
    ...contained statements...
</txp:if_different>
```

The tag will execute the contained statement when the value of the contained statement differs from the preceding value for that contained statement. Can be used in Textpattern 'article', 'link', 'comment', and 'file' type [[Form templates]].

`if_different` can contain several HTML tags but only **one** Textpattern tag.

Attributes

This tag has no attributes.

Examples

Example 1: Display posting time per article once per day

```
<txp:if_different>
    <h3>
        <txp:posted format="%d %B %Y" />
    </h3>
</txp:if_different>
```

To be used inside a Textpattern 'article' type [[Form template]] or an article tag container (such as article or article_custom).

Other tags used: posted.

Example 2: Build an indented list of article titles grouped by section

Intention:

- Display a list of all articles' titles grouped by sections and ordered by article title.
- Add headings from section titles to designate an article's context.
- Sort alphabetically by section name, then by article title.

Desired result:

- about (section title)
    - 1st Article from about section
    - 2nd Article from about section
    - …another article
- family (section title)
    - 1st Article from family section
    - 2nd Article from family section
    - …another article
- people (section title)
    - 1st Article from people section
    - 2nd Article from people section
    - …another article

In a Textpattern [[Page template]], add this tag to loop through all articles from all sections:

```
<txp:article_custom sort="Section asc, Title asc">
    <txp:if_different>
```

```
        <h2>
            <txp:section title="1" />
        </h2>
    </txp:if_differen>
    <h3>
        <txp:title />
    </h3>
</txp:article_custom>
```

The snippet above lists all article titles and renders an intermittent heading element whenever a **different** section is encountered while the articles loop through.

Other tags used: article_custom, section, title.

If excerpt

On this page:

Syntax

```
<txp:if_excerpt>
```

The **if_excerpt** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_excerpt>
    ...conditional statement...
</txp:if_excerpt>
```

The tag will execute the contained statements if an excerpt is associated with the article being displayed.

Attributes

This tag has no attributes.

Examples

Example 1: Display the excerpt if it exists

```
<txp:if_excerpt>
    <txp:excerpt />
</txp:if_excerpt>
```

Other tags used: excerpt.

If expired

On this page:

Syntax

```
<txp:if_expired>
```

The **if_expired** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_expired>
    ...conditional statement...
</txp:if_expired>
```

The tag will execute the contained statements, if a particular article is expired. Should be used in an ‘article’ type form.

Attributes

This tag has no attributes.

Examples

Example 1: Show when article has (or will) expire

```
<txp:if_expired>
    This article is already expired. It expired <txp:expires />.
<txp:else />
    <txp:if_expires>
        This page isn't expired. It expires <txp:expires />.
    <txp:else />
        This page doesn't expire.
    </txp:if_expires>
</txp:if_expired>
```

Other tags used: else, expires, if_expires.

Genealogy

Version 4.0.7

Tag support added.

If expires

On this page:

Syntax

```
<txp:if_expires>
```

The **if_expires** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_expires>
    ...conditional statement...
</txp:if_expires>
```

The tag will execute the contained statements, if a particular article has an expiry date set. Should be used in an 'article' type form.

Attributes

This tag has no attributes.

Examples

Example 1: Show when an article is to expire

```
<txp:if_expires>
    This article expires on <txp:expires />.
</txp:if_expires>
```

Other tags used: expires.

Genealogy

Version 4.0.7

Tag support added.

If first article

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:if_first_article>
```

The **if_first_article** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_first_article>
    ...conditional statement...
</txp:if_first_article>
```

The tag will execute the contained statements if the displayed article is the first in the currently displayed list. It will display in both single article and article list modes. Should be used in an Textpattern ‘article’ type [[Form template]].

Attributes

This tag has no attributes.

Examples

Example 1: Add a linked section by title

```
<h3>
    <txp:permlink>
        <txp:title />
    </txp:permlink>
    |
    <txp:posted />
    by
    <txp:author />
    <txp:if_first_article>
        | Section:
        <txp:section link="1" title="1" />
    </txp:if_first_article>
</h3>
<txp:body />
<txp:comments_invite wraptag="p" />
```

Displays a link to the header of the first article in an article list.

Other tags used: permlink, title, author, posted, section, body, comments_invite.

Example 2: Add a class to a list item

```
<li<txp:if_first_article> class="first"</txp:if_first_article>>
    <txp:permlink>
        <txp:title />
    </txp:permlink>
</li>
```

Adds a CSS class to the first article in an article list.

Other tags used: permlink, title.

If first category

On this page:

Syntax

```
<txp:if_first_category>
```

The **if_first_category** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_first_category>
    ...conditional statement...
</txp:if_first_category>
```

The tag will execute the contained statements if the current category (usually one inside the container or form of a category_list) is the first in the currently displayed list.

Attributes

This tag has no attributes.

Examples

Example 1: Identify first category in a category list

```
<txp:category_list parent="group-1" children="0">
    <txp:if_first_category>
        <h3>
            <txp:category />
        </h3>
    <txp:else />
        <txp:category link="1" />
    </txp:if_first_category>
</txp:category_list>
```

Prevents the first category in the list from being hyperlinked to a category page. Why you might do it? If you nest categories under a ‘header’ category you might want to show the header of the group but not allow people to link to its category page.

Other tags used: category, category_list, else.

Genealogy

Version 4.0.7

Tag support added.

If first file

On this page:

Syntax

```
<txp:if_first_file>
```

The **if_first_file** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_first_file>
    ...conditional statement...
</txp:if_first_file>
```

The tag will execute the contained statements if the displayed file is the first in the currently displayed file_download_list. The tag supports else.

Attributes

This tag has no attributes.

Examples

Example 1: Identify first file in a file list

[todo:pw]

Other tags used:

Genealogy

Version 4.6.0

Tag support added.

If first image

On this page:

Syntax

```
<txp:if_first_image>
```

The **if_first_image** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_first_image>
    ...conditional statement...
</txp:if_first_image>
```

The tag will execute the contained statements if the displayed image is the first in the currently displayed images list. The tag supports else.

Attributes

This tag has no attributes.

Examples

Example 1: Identify first image in an image list

[todo:pw]

Other tags used:

Genealogy

Version 4.6.0

Tag support added.

If first link

On this page:

Syntax

```
<txp:if_first_link>
```

The **if_first_link** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_first_link>
    ...conditional statement...
</txp:if_first_link>
```

The tag will execute the contained statements if the displayed link is the first in the currently displayed linklist. The tag supports else.

Attributes

This tag has no attributes.

Examples

Example 1: Identify first link in an link list

[todo:pw]

Other tags used:

Genealogy

Version 4.6.0

Tag support added.

If first section

On this page:

- Syntax
- Attributes
- Examples
- Genealogy

Syntax

```
<txp:if_first_section>
```

The **if_first_section** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_first_section>
    ...conditional statements...
</txp:if_first_section>
```

The tag will execute the contained statements if the current section (usually one inside the container or form of a section_list) is the first in the currently displayed list.

Attributes

This tag has no attributes.

Examples

Example 1: Assign a specific id to the first item in the list

```
<txp:section_list wraptag="ul" break="">
    <li<txp:if_first_section> id="first"</txp:if_first_section>>
        <txp:section title="1" link="1" />
    </li>
</txp:section_list>
```

Genealogy

Version 4.0.7

Tag support added.

If individual article

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:if_individual_article>
```

The **if_individual_article** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_individual_article>
    ...conditional statement...
</txp:if_individual_article>
```

The tag will execute the contained statements if an individual article is being displayed (i.e. not an article list).

Please note, that article_custom always displays an article list, even when you set it to display only one article. Thus the **if_individual_article** tag will not work with article_custom, you'll have to use the article tag instead.

Attributes

This tag has no attributes.

Examples

Example 1: Select next-prev or older-newer navigation

```
<txp:article />

<txp:if_individual_article>
    <p>
        <txp:link_to_prev>
            <txp:prev_title />
        </txp:link_to_prev>
        <txp:link_to_next>
            <txp:next_title />
        </txp:link_to_next>
    </p>
</txp:if_individual_article>

<txp:if_article_list>
    <p>
        <txp:olde>
            Previous
        </txp:older>
        <txp:newer>
            Next
        </txp:newer>
    </p>
</txp:if_article_list>
```

Shows links to the next/previous article if the current page is an article, or shows links to the next/previous page of results if the current page is an article list.

Other tags used: link_to_prev, link_to_next, next_title, prev_title, if_article_list, older, newer.

Example 2: Use the tag with else

```
<txp:if_individual_article>
    <p>
        <txp:site_name />
    </p>
<txp:else />
```

```
    <p>
        <img src="http://example.com/logo.png" alt="Logo">
    </p>
</txp:if_individual_article>
```

Displays the site’s name when showing a single article, and a logo when **not** displaying a single article.

Other tags used: else, site_name.

If keywords

On this page:

Syntax

```
<txp:if_keywords>
```

The **if_keywords** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_keywords>
    ...conditional statement...
</txp:if_keywords>
```

The tag will execute the contained statement if the current article's 'Keywords' field has one or more entries.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `keywords="keywords"`
  Comma-separated list of keywords.
  Default: unset, which determines whether any keywords are assigned to the article.

Examples

Example 1: Supply meta tag if keywords exist

```
<head>
    ....
    <txp:if_individual_article&gt;
        <txp:if_keywords>
            <txp:meta_keywords />
        <txp:else />
            <meta name="keywords" content="Apple, Orange, Pear, Foo, Bar" />
        </txp:if_keywords>
```

….

Other tags used: meta_keywords, if_individual_article, else.

Genealogy

Version 4.0.7

Tag support added

If last article

On this page:

Syntax

```
<txp:if_last_article>
```

The **if_last_article** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_last_article>
    ...conditional statement...
</txp:if_last_article>
```

The tag will execute the contained statements if the displayed article is the last in the currently displayed list. It will display in both single article and article list modes. Should be used in an Textpattern ‘article’ type [[Form template]].

Attributes

This tag has no attributes.

Examples

Example 1: Add an image after the last article in a list

```
<h3>
    <txp:permlink>
        <txp:title />
    </txp:permlink>
    |
    <txp:posted />
    by
    <txp:author />
</h3>

<txp:body />
<txp:comments_invite wraptag="p" />

<txp:if_last_article>
    <img src="http://example.com/footer.jpg" alt="Footer">
</txp:if_last_article>
```

Other tags used: "permlink""permlink, title, posted, author, body, comments_invite.

If last category

On this page:

Syntax

```
<txp:if_last_category>
```

The **if_last_category** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_last_category>
    ...conditional statements...
</txp:if_last_category>
```

The tag will execute the contained statements if the current category (usually one inside the container or form of a category_list) is the last in the currently displayed list.

Attributes

This tag has no attributes.

Examples

Example 1: Identify last category in a category list

```
<txp:category_list parent="group-1" children="0">
    <txp:if_last_category>
        <h3>
            <txp:category />
        </h3>
    <txp:else />
        <txp:category link="1" />
    </txp:if_last_category>
</txp:category_list>
```

Prevents the last category in the list from being hyperlinked to a category page. Why you might do it? If you nest categories under a ‘header’ category you might want to show the header of the group but not allow people to link to its category page.

Other tags used: category, category_list, else.

Genealogy

Version 4.0.7

Tag support added.

If last file

On this page:

Syntax

```
<txp:if_last_file>
```

The **if_last_file** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_last_file>
    ...conditional statement...
</txp:if_last_file>
```

The tag will execute the contained statements if the displayed file is the last in the currently displayed file_download_list. The tag supports else.

Attributes

This tag has no attributes.

Examples

Example 1: Identify last file in a file list

[todo:pw]

Other tags used:

Genealogy

Version 4.6.0

Tag support added.

If last image

On this page:

Syntax

```
<txp:if_last_image>
```

The **if_last_image** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_last_image>
    ...conditional statement...
</txp:if_last_image>
```

The tag will execute the contained statements if the displayed image is the last in the currently displayed images list. The tag supports else.

Attributes

This tag has no attributes.

Examples

Example 1: Identify last image in an image list

[todo:pw]

Other tags used:

Genealogy

Version 4.6.0

Tag support added.

If last link

On this page:

Syntax

```
<txp:if_last_link>
```

The **if_last_link** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_last_link>
    ...conditional statement...
</txp:if_last_link>
```

The tag will execute the contained statements if the displayed link is the last in the currently displayed linklist. The tag supports else.

Attributes

This tag has no attributes.

Examples

Example 1: Identify last link in an link list

[todo:pw]

Other tags used:

Genealogy

Version 4.6.0

Tag support added.

If last section

On this page:

Syntax

```
<txp:if_last_section>
```

The **if_last_section** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_last_section>
    ...conditional statements...
</txp:if_last_section>
```

The tag will execute the contained statements if the current section (usually one inside the container or form of a section_list) is the last in the currently displayed list.

Attributes

This tag has no attributes.

Examples

Example 1: Assign a specific id to the last item in the list

```
<txp:section_list wraptag="ul" break="">
    <li<txp:if_last_section> id="last"</txp:if_last_section>>
        <txp:section title="1" link="1" />
    </li>
</txp:section_list>
```

Genealogy

Version 4.0.7

Tag support added.

If plugin

On this page:

Syntax

`<txp:if_plugin>`

The **if_plugin** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_plugin>
    ...conditional statement...
</txp:if_plugin>
```

The tag will execute the contained statements if the name attribute matches a currently installed and enabled plugin, and the current version number is equal to or greater than the `version` attribute assigned (if used).

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="text"`
  Plugin name as defined on the Plugins administration panel.
- `version="number"`
  Minimum plugin version number.
  Default: unset.

Examples

Example 1: Check plugin exists before using a tag

```
<txp:if_plugin name="zem_plugin_lang" version="4">
    <txp:zem_contact to="dest@example.com" />
</txp:if_plugin>
```

Applies the tag `<txp:zem_contact />` if the ‘zem_contact_lang’ plugin is installed, activated, and the version number is equal to or greater than 4.

Genealogy

Version 4.3.0

`ver` attribute deprecated and renamed `version`.

If search results

On this page:

Syntax

```
<txp:if_search_results>
```

The **if_search_results** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<;txp:if_search_results>
    ...conditional statement...
</txp:if_search_results>
```

The tag will execute the contained statements if the current article list contains a certain amount of entries matching the search term – mostly more than zero.

A typical application of this tag is the conditional output of a “Sorry, we found no items matching your search request.” message, but the `min` and `max` attributes allow for a finer grained reaction to search queries.

Note: You cannot use this tag directly inside an if_search tag without using an article tag first to actually perform the search! See Example 2 below for clarification.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `max="number"`
  If the search results count is no higher than `max`, the tags enclosed by this conditional tag are rendered.
  Default: unset (which results in no upper limit).
- `min="number"`
  If the search results count is at least equal to `min`, the tags enclosed by this conditional tag are rendered.
  Default: `1`.

Examples

Example 1: More informative search result output

```
<txp:if_search_results>
    <p>These articles match your search request:</p>
<txp:else />
    <p>Sorry, we were not able to find a page matching your search request <q><txp:search_term /></q>.</p>
</txp:if_search_results>
```

Ensures the visitor does not see a blank page when no articles match the search request.

Other tags used: else, search_term.

Example 2: In context within if_search

```
<txp:if_search>
    <txp:article pgonly="1" limit="10" />
    <txp:if_search_results>
        <p>These articles match your search request:</p>
        <txp:article limit="10" searchform="search_results" />
    <txp:else />
        <p>Sorry, we were not able to find a page matching your search request <q><txp:search_term /></q>.</p>
    </txp:if_search_results>
</txp:if_search>
```

Detects if a search is in progress, calls the article tag to perform the search but **inhibits display via the `pgonly` attribute**. Once the search has been performed (internally) and Textpattern knows how many search results there are, you can then use if_search_results to detect whether there were any or not. Why you have to do this? Because it’s the only way to use the tag! Trying to use it without first calling an article tag will give unexpected results and, more often than not, a “Page template … does not contain a txp:article tag” warning.

Note: You must ensure that all attributes used in your two article tags are identical (except for any `form` attributes, which can safely be omitted when using `pgonly`). Failure to keep the tags in sync will result in strange article counts or odd behaviour.

Other tags used: if_search, article, else, search_term.

Example 3: Take action when there are too many hits

```
<txp:if_search_results max="500">
    <p>These articles match your search request:</p>
<txp:else />
    <p>Seems like you are looking for a very common search term. Try using a more specific search phrase.</p>
</txp:if_search_results>
```

Advises the visitor to search for something more specific in the case where their search term generated an excessive amount of hits.

Other tags used: else.

Genealogy

Version 4.0.6

Tag support added.

If search

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:if_search>
```

The **if_search** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<;txp:if_search>
    ...conditional statement...
</txp:if_search>
```

The tag will execute the contained statements if the called page is the result of a search.

Attributes

This tag has no attributes.

Examples

Example 1: Select a different stylesheet during search

```
<txp:if_search>
    <link rel="stylesheet" href="<txp:css name="search" />">
<txp:else />
    <link rel="stylesheet" href="<txp:css />">
</txp:if_search>
```

Selects a stylesheet named ‘search’ when search results are being displayed, or a stylesheet determined by the active section for normal page display.

Other tags used: else, css.

If section

On this page:

Syntax

```
<txp:if_section>
```

The **if_section** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<;txp:if_section>
    ...conditional statement...
</txp:if_section>
```

The tag will execute the contained statements if the called page is part of the section specified with the name attribute.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="section"`
  Comma-separated list of section names. For the default section, either use the text `default` or a single comma `,` (for example, both `name=", other_section"` and `name="default, other_section"` are equivalent).

Examples

Example 1: Conditionally display text for a section

```
<txp:if_section name="about">
    <p>Danger, ego pages ahead!</p>
<txp:else />
    <p>Nothing. Just nothing. Any ideas? Anybody?</p>
</txp:if_section>
```

Other tags used: else.

Example 2: Add a special class to mark the currently active section

```
<nav>
    <ul>
        <li<txp:if_section name=", article"> class="active"</txp:if_section>>
            <txp:section link="1" title="1" name="" />
        </li>
        <li<txp:if_section name="portfolio"> class="active"</txp:if_section>>
            <txp:section link="1" title="1" name="portfolio" />
        </li>
        <li<txp:if_section name="about"> class="active"</txp:if_section>>
            <txp:section link="1" title="1" name="about" />
        </li>
    </ul>
</nav>
```

A different way of marking the active section can be accomplished by using section_list and its attribute `active_class`. While the above snippet will mark the list item, section_list will mark solely the link.

Other tags used: section.

If status

On this page:

Syntax

```
<txp:if_status>
```

The **if_status** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_status>
    ...conditional statement...
</txp:if_status>
```

The tag will execute the contained statements depending on the requested page’s HTTP status condition. Normal pages result in a status code of ‘200’, while missing pages set Textpattern’s status to ‘404’.

This tag provides a method of sharing one page template between common pages and error pages, but including different output depending on the page’s HTTP status.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `status="number"`
  Numerical HTTP status code.
  Default: `200` (OK).

Examples

Example 1: Conditionally display text on missing pages

```
<txp:if_status status="404">
    <h2>The page you requested could not be found.</h2>
</txp:if_status>
```

If thumbnail

On this page:

Syntax

```
<txp:if_thumbnail>
```

The **if_thumbnail** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_thumbnail>
    ...conditional statement...
</txp:if_thumbnail>
```

The tag will execute the contained statements **if** the current image (from an images tag) has a thumbnail assigned to it. Must always be used in an image context.

Attributes

This tag has no attributes.

Examples

Example 1: Only show a thumbnail if one exists

```
<txp:images author="neo" wraptag="figure" class="author-gallery">
    <txp:if_thumbnail>
        <txp:thumbnail />
        <txp:image_info wraptag="figcaption" />
    <txp:else />
        <p>No thumbnail</p>
    </txp:if_thumbnail>
</txp:images>
```

For each image uploaded by author 'neo', display its thumbnail if it has one, or the text 'No thumbnail' if it doesn't. Add a caption beneath the thumbnail using image_info.

Other tags used: images, else, image_info, thumbnail.

Genealogy

Version 4.3.0

Tag support added.

If variable

On this page:

Syntax

```
<txp:if_variable>
```

The **if_variable** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_variable>
    ...conditional statement...
</txp:if_variable>
```

It tests the existence and/or value of a global variable set with the variable tag.

Note: In case you are getting unexpected results in an **if_variable** evaluation, check whether you have entered additional white space or invisible characters in your variable declarations and remove those.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="variable name"`
  The name of the variable you wish to check.
- `value="value"`
  The (optional) value which the named variable must match in order for the contained statements to be executed. If this attribute is omitted, the tag returns ‘true’ if the named variable is defined. If this attribute’s value is omitted (i.e. `value=""`), the tag returns ‘true’ if the variable is defined, but has no value.

Examples

See the variable page for examples.

Genealogy

Version 4.0.7

Tag support added.

If yield

On this page:

Syntax

```
<txp:if_yield>
```

The **if_yield** tag is a *conditional* tag and always used as an opening and closing pair, like this…

```
<txp:if_yield>
    ...conditional statement...
</txp:if_yield>
```

It tests the existence and/or value of a content set with the yield tag inside of an output_form container tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `value="value"`
  The (optional) value which the yield content must match in order for the contained statements to be executed. If this attribute is omitted, the tag returns ‘true’ if the yield content is defined. If this attribute’s value is omitted (i.e. `value=""`), the tag returns ‘true’ if yield content is defined, but has no value.

Examples

Example 1: Check if yield is defined

```
<txp:if_yield>
    Output_form was used as a container.
<txp:else />
    No yield defined.
</txp:if_yield>
```

Example 2: Check if yield content is set

```
<txp:if_yield value="">
    No yield, is empty.
<txp:else />
    Yield set: <txp:yield />
</txp:if_yield>
```

Other tags used: yield.

Example 3: Check against a specific yield content

```
<txp:if_yield value="red">
 Yield is red.
</txp:if_yield>
```

Genealogy

Version 4.6.0

Tag support added.

Image author

On this page:

Syntax

```
<txp:image_author />
```

The **image_author** tag is a *single* tag that Textpattern will replace with the author's name associated with the current image in an images list. It can **only** be used inside `<txp:images />`.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="link type (boolean)"`
  Whether to hyperlink the author or not.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `section="section name"`
  Direct any linked author name to the nominated section instead of to the default (front) page.
  Default: unset.
- `this_section="boolean"`
  If set to 1, the linked author name will direct users to an author list in the current section.
  Values: `0` (all sections) or `1` (this section only).
  Default: `0`.
- `title="boolean"`
  Whether to display the author's real name or login name.
  Values: `0` (login name) or `1` (real name).
  Default: `1`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: unset (see [[class cross-reference]]).
- `wraptag="element"`
  HTML element to wrap around the author name, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Add image author to gallery

```
txp:images category="mammals">
    <a href="<txp:image_url />">
        <txp:thumbnail />
    </a>
    <div class="img-info">
        by <txp:image_author />
    </div>
</txp:images>
```

Other tags used: images, thumbnail.

Example 2: Link to author list

```
<txp:images category"fish">
```

```
    <a href="<txp:image_url />">
        <txp:thumbnail />
    </a>
    <div class="img-info">
        by <txp:image_author link="1" />
    </div>
</txp:images>
```

Displays thumbnails and author info for each image in the `fish` category. The authors' names are hyperlinked to `example.com/author/image/User+Name`.

Other tags used: images, thumbnail.

Genealogy

Version 4.3.0

Tag support added.

Image date

On this page:

Syntax

```
<txp:image_date />
```

The **image_date** tag is a *single* tag that Textpattern will replace with the uploaded date of the current (or given) image. Should usually be used in an [[image form]], although it may be used on its own providing you specify an `id` or `name`.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `id="integer"`
  An `id` assigned at upload of an image to display. The IDs can be found on the Images administration panel.
  Default: unset.
- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values, `since`, `iso8601` (ISO 8601 reference), `w3cdtf` (W3CDTF reference), or `rfc822` (RFC 822 reference).
  Default: the ‘Archive date format’ set in preferences.
- `name="image name"`
  An image to display, given by its image name as shown on the Images administration panel. If both `name` and `id` are specified, the `id` takes precedence.
  Default: unset.

Examples

Example 1: Display additional image information

```
<txp:images category="mammals">
    <a href="<txp:image_url />">
        <txp:thumbnail />
    </a>
    <div class="img-info">
        <txp:image_info type="caption, author" break=" by " />
        <txp:image_date format="%e %b %Y" />
    </div>
</txp:images>
```

Other tags used: images, image_url, image_info, thumbnail.

Genealogy

Version 4.3.0

Tag support added.

Image display

On this page:

Syntax

```
<txp:image_display />
```

The **image_display** tag is a *single* tag that is intended to be used in tandem with image_index.

The **image_display** tag displays an image specified by the page URL which in turn is built by its tandem tag image_index. To use this tag successfully, it has to be placed either inside an article which shares a common category with the images to display (thereby linking article and image categories), or in a location at the page template which is displayed without any special article context.

If this tag seems to display no image at all, it probably resides inside an article which is never rendered as it does not belong to the currently active category.

Attributes

This tag has no attributes.

Examples

Example 1: Display a single image as chosen by image_index

```
<txp:image_display />
```

See the image_index tag documentation for details on how to populate this tag with content.

Image index

On this page:

Syntax

```
<txp:image_index />
```

The **image_index** tag is a *single* tag that is intended to be used in tandem with image_display. It renders thumbnails of all images contained in an image category. This category can be specified as an attribute to the tag and defaults to the current site category as given in the page's URL.

The thumbnail images are linked to an address which will pass the image ID plus the active category on to the tandem image_display tag. It is up to the user to include this tandem tag at an appropriate place inside the page template.

Note: As the image category is passed into image_display, it requires to either place the 'receiving' image_display on an article independent portion of the page (i.e. outside of the article form), or otherwise both the article used for display and the images have to share a 'common' category.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `category="category name"`
  Category of images to display.
  Values: (comma separated list of) category name(s).
  Default: presently viewed category.
- `limit="integer"`
  The number of images to display.
  Default: `0` (no limit).
- `offset="integer"`
  The number of images to skip.
  Default: `0`.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `alt` (image alt text)
  `author`
  `caption` (image caption text)
  `category` (image category)
  `date` (date posted)
  `ext` (image extension)
  `h` (image height)
  `id` (image id#)
  `name` (image name)
  `rand()` (random).
  `thumbnail`
  `w` (image width)
  Each field in the `txp_image` database table can be used as a sort key.
  Default: `name asc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.

Default: tag name or unset (see [[class cross-reference]]).

- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Create a list of images in a category

```
<txp:image_index category="personal" wraptag="ol" break="li" />
<txp:image_display />
```

Shows the thumbnail images from the category ‘personal’.

Other tags used: image_display.

Genealogy

Version 4.3.0

`c` attribute deprecated and renamed `category`.

Image info

On this page:

Syntax

```
<txp:image_info />
```

The **image_info** tag is a *single* tag that Textpattern will replace with the relevant image data from the current image. Should usually be used in an ‘image’ type form, although it may be used on its own providing you specify an **id** or **name**.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as <, > and `&`.
  Values: `html` or unset.
  Default: `html`;
- `id="integer"`
  An `id` assigned at upload of an image to display. The IDs can be found on the Images administration panel.
  Default: unset.
- `name="image name"`
  An image to display, given by its image name as shown on the Images administration panel. If both `name` and `id` are specified, the `id` takes precedence.
  Default: unset.
- `type="information type"`
  One or more of the following values to display the particular pieces of information from the current image.
  Values:
  `alt`: image `alt` content.
  `author`: image author’s login name (see image_author to display the author’s real name).
  `caption`: image `caption` content.
  `category`: image category name.
  `category_title`: image category title.
  `date`: timestamp of image upload (this is not very useful, so consult image_date for a better alternative).
  `ext`: image extension.
  `h`: image height.
  `id`: image id.
  `name`: image name.
  `thumb_w`: image thumbnail width.
  `thumb_h`: image thumbnail height.
  `w`: image width.
  Default: `caption`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: unset.
- `class="class name"`
  CSS `class` attribute to apply to the `wraptag`, if set.
  Default: unset (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML element to wrap the items grabbed from the `type` attribute, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Gallery thumbnail and caption

```
<txp:images category="mammals">
    <txp:thumbnail />
    <txp:image_info type="caption" wraptag="div" />
</txp:images>
```

Grabs all images from the ‘mammals’ category and displays the image thumbnail itself along with the image caption surrounded with HTML `<div>` tags. Note that the image IDs/names are not specified inside the container because they are automatically assigned from the `<txp:images>` tag for each image in the given category.

Other tags used: images, thumbnail.

Example 2: Multiple pieces of information at once

```
<txp:images category="birds, mammals" thumbnail="1" sort="category asc">
    <txp:if_different>
        <h4>
            <txp:image_info type="category_title" />
        </h4>
    </txp:if_different>
    <txp:thumbnail wraptag="div" />
    <txp:image_info type="w, h" wraptag="div" class="img-dimensions" break=" x " />
    by
    <txp:image_info type="author" />
</txp:images>
```

Shows the thumbnail of each image that has an assigned thumbnail image from the ‘mammals’ and ‘birds’ categories and, beneath each, show its dimensions ‘width’ x ‘height’ along with the author of the image. Since the list has been sorted by category, the `<txp:if_different>` conditional can be used to output the category title at the top of the list of images each time it changes.

Other tags used: images, thumbnail, if_different.

Example 3: Specific image information

```
<txp:image_info id="5" type="category_title" />
```

Displays the category_title of the category assigned to image ID ‘5’.

Genealogy

Version 4.6.0

`breakclass` attribute deprecated.

Version 4.3.0

Tag support added.

Image list DEPRECATED [todo:pw]

mininav

Do not use this tag, it was renamed to images!

```
<txp:image_list />
```

The image_list tag is a *single* or *container* tag that Textpattern will use to gather a list of matching images uploaded via the TXP images panel tab. Utilising the other image tags in the suite image_info, image_url, image_date and if_thumbnail) you can display simple image galleries from this list.

If used as a *container* tag, it must be specified as an opening and closing pair of tags, like this:

```
<txp:image_list>
...contained statements...
</txp:image_list>
```

This is equivalent to putting the contained statements into a form named “my_form” and using

```
<txp:image_list form="my_form" />
```

.

By default, the tag is context-sensitive, which means that in the absence of any other filter attributes (**id**, **name**, **category**, **author**, **realname**, **extension**, **thumbnail**), it will grab a list of image IDs from the currently viewed article’s **article_image** field. If it finds nothing there, it will check the URL to see if there is a category list in progress. If it finds nothg there too, the tag will display nothing. See the **auto_detect** attribute for further information.

Attributes

Tag will accept the following attributes (**case-sensitive**):

id
Filter the images by this list of

```
id
```

s assigned at upload. The IDs can be found on the images tab. Default: unset. **name** **name”**
Filter the images by this list of image names as shown on the images tab. Default: unset. **category** **category”**
Filter the images by this list of category names as defined in the categories tab. **author** **(login) ID”**
Filter the images by this list of author IDs who uploaded the pictures to Textpattern. Default: unset. **realname** **name”**
Filter the image list so it only includes images uploaded by this list of author real names. The author names may be URL encoded (e.g. **realname=“John+Smith”**) and thus could be read from the current site.com/author/author+name URL. Note that this attribute may incur one extra query per name, so if it is possible to use the raw author instead it will be faster. Default: unset. **extension**
Filter the images by this list of image extensions, including the leading dot. Example:

```
extension=".jpg, .png"
```

Default: unset. **thumbnail**
Filter the image list to only include images that have a thumbnail (

```
1
```

) or not (

```
0
```

). Default: unset (i.e.,all images). **label**
Text to display above the list of images as a label. Default: unset. **labeltag** **text”**
HTML tag to be used to wrap the label, specified without brackets. Default: unset. **wraptag** **text”**
HTML tag to be used to wrap the

```
img
```

tag, specified without brackets. Default: unset. **html_id** **number”**
The HTML

```
id
```

attribute applied to the

`wraptag`

, if set, otherwise to the

`img`

tag. Default: unset. **class name"**
CSS

`class`

attribute applied to the

`wraptag`

, if set, otherwise to the

`img`

tag. Default: unset. **break text"**
The (X)HTML tag (without brackets) or string to separate each image. Default: unset. **limit**
The number of images to display per page. Default:

`0`

(unlimited). **offset**
The number of images to skip. Default:

`0`

. **pageby (or _"limit"_*)***
The number of images to jump forward or back when an older or newer link is clicked. Without this attribute, pagination is not available; you will simply see **limit** images. You may specify

`pageby="limit"`

to allow pagination to automatically follow the value of the

`limit`

attribute. NOTE: newer and older will paginate all content types at once. Default: unset. **sort value(s)"**
How to sort the resulting image list. Specify an image attribute from the ones below and add either **asc** or **desc** to sort in ascending or descending order, respectively. Values:

`id`

(image id#)

`name`

(image name)

`category`

`extension`

(image extension)

`author`

`alt`

`caption`

`date`

`w`

(image width)

`h`

(image height)

```
thumb_w
```

(image thumbnail width)

```
thumb_h
```

(image thumbnail height)

```
rand()
```

(random) Default:

```
name asc
```

form name"
Use specified form for each image. If not used, and the container is empty, the tag will output a list of images that are compatible with image_display. Default: unset. **auto_detect context"**
List of Textpattern contexts to consider when automatically searching for images. If you wish to turn off the automatic check, set this to **auto_detect=""**. You can choose from the following contexts: **article** to look in the article_image field **category** to look in the URL for a category list **author** to look in the URL for an author list Default: **article, category, author**

Examples

Example 1: varying attributes

This example shows the outcome of various attribute configurations to give you an idea of what to expect from the tag. More concrete examples follow.

NB: THESE MAY NOT BE CORRECT ANY MORE DUE TO TAG CHANGES SINCE THE EXAMPLES WERE WRITTEN. NEED VERIFICATION.

<code><txp:image_list /></code>
uses the current article image field as a list of image IDs. If the article image field is empty, it contains names/URLs, or the tag is used on an article list page, it returns no images. If, however, the tag is used on a category list page (e.g.

```
category/image/photos
```

) then the **photos** category would be used **<code><txp:image_list auto_detect="" /></code>**
displays all images in the database. **<code><txp:image_list id="" /></code> <code><txp:image_list name="" /></code> <code><txp:image_list category="" /></code>**
no images displayed. This means that if you did some tag-in-tag magic such as :

```
category='<txp:custom_field name="my_cats" />'
```

it will show no images if the custom field is empty. **<code><txp:image_list id="2,3,6" /></code>**
display images 2, 3, and 6. **<code><txp:image_list name="lion.jpg, zebra.jpg" /></code>**
the named images are displayed. **<code><txp:image_list name="pengiun.jpg" /></code>**
no images are displayed (mis-spelled image name). **<code><txp:image_list category="mammals, birds" /></code>**
all images in the named categories are displayed. **<code><txp:image_list category=", mammals, birds" /></code>**
all images in the named categories and any uncategorized images are displayed. **<code><txp:image_list category=" " /></code>**
just uncategorized images are displayed (note that

```
category=","
```

also works, but a space looks better). **<code><txp:image_list author="attenborough, morris" /></code>**
all images by author (ID) **attenborough** and **morris** are displayed. **<code><txp:image_list realname="David+Attenborough" /></code>**
all images by author **David Attenborough** are displayed. This incurs one extra query to look up the author's ID from the given real name. **<code><txp:image_list category="mammals, birds" author="attenborough, morris" /></code>**
all images in the named categories that are assigned to the named authors are displayed. **<code><txp:image_list category="mammals, birds" extension=".jpg" /></code>**
all jpg images in the named categories are displayed. **<code><txp:image_list category="mammals, birds" extension=".jpg" author="attenborough, morris" /></code>**
all jpg images in the named categories that are assigned to the named authors are displayed. **<code><txp:image_list extension=".gif" /></code>**
all GIF images are displayed. **<code><txp:image_list category="mammals, birds" thumbnail="1" /></code>**
all images in the named categories that have thumbnails assigned to them are displayed. **<code><txp:image_list thumbnail="1" /></code>**

all images that have thumbnails assigned to them are displayed. **<code><txp:image_list thumbnail="0" /></code>** all images that do not have thumbnails assigned to them are displayed.

Example 2: description goes here

What it does…
…

Genealogy

Version 4.3.0

- tag introduced

Image url

On this page:

Syntax

```
<txp:image_url />
```

The **image_url** tag is a *single* or a *container* tag that Textpattern will replace with the URL of the current image in an images list, or the specific image if given an `id` or `name`.

If used as a container, it must be specified as an opening and closing pair of tags, like this:

```
<txp:image_url>
    ...link contents...
</txp:image_url>
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `id="integer"`
  An `id` assigned at upload of an image to display. The IDs can be found on the Images administration panel.
  Default: unset.
- `link="link type"`
  Whether to hyperlink the URL or not.
  Values:
  `1`: hyperlink the URL (if used as a single tag) or the container content.
  `0`: don't hyperlink the URL/container.
  `auto`: only apply the hyperlink if the tag is used as a container.
  Default: `auto`.
- `name="image name"`
  An image to display, given by its image name as shown on the Images administration panel. If both `name` and `id` are specified, the `id` takes precedence.
  Default: unset.
- `thumbnail="boolean"`
  Display the link to the image's thumbnail instead of the full size image.
  Values: `0` (display the full size image) or `1` (display the thumbnail).
  Default: `0`.

Examples

Example 1: Directly link gallery thumbs to main image

Used as a single tag:

```
<txp:images category="mammals">
    <a href="<txp:image_url />">
        <txp:thumbnail />
    </a>
</txp:images>
```

Used as a container:

```
<txp:images category="mammals">
    <txp:image_url>
        <txp:thumbnail />
    </txp:image_url>
</txp:images>
```

Other tags used: "images"images, thumbnail.

Genealogy

Version 4.3.0

Tag support added.

Image

On this page:

Syntax

`<txp:image />`

The **image** tag is a *single* tag that Textpattern will replace with the `<img src="...">` HTML tag matching the image of the numeric `id` assigned by Textpattern when the image was uploaded via the Textpattern Images administration panel.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&` for the image's `alt` and `title` attributes.
  Values: `html` or unset.
  Default: `html`.
- `height="integer"`
  Specify an image `height` which overrides the value stored in the database. Use `height="0"` to turn off the output of a width attribute in the `<img>` tag (thus the browser will scale the height if a width is used).
  Default: height of image stored in the database.
- `html_id="id"`
  The HTML `id` attribute applied to the `wraptag`, if set, otherwise to the `<img>` tag.
  Default: unset.
- `id="integer"`
  Specifies the `id`, assigned at upload of the image, to display. Can be found on the Images administration panel. If both `name` and `id` are specified, `name` is used while `id` is ignored.
- `name="image name"`
  Specifies which image to display by its image `name` as shown on the Images administration panel.
- `width="integer"`
  Specify an image `width` which overrides the value stored in the database. Use `width="0"` to turn off the output of a width attribute in the `<img>` tag (thus the browser will scale the width if a height is used).
  Default: width of image stored in the database.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value, if set, otherwise to the `<img>` tag.
  Default: unset (see [[class cross-reference]]).
- `style="style rule"`
  Inline CSS `style` rule. It's recommended that you assign CSS rules via `class` attribute instead.
  Default: unset.
- `wraptag="element"`
  HTML tag to be used to wrap the `<img>` tag, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display the given image

```
<txp:image id="42" />
```

Displays the image uploaded as ID #42.

Example 2: Apply a CSS class

```
<txp:image name="chickens.jpg" class="promoted" />
```

Displays the image named `chickens.jpg` and assigns a `class="promoted"` attribute/value to the `<img>` tag.

Had the `wraptag` attribute been used, the `class="promoted"` attribute/value would have been applied to that instead of directly to the image tag.

Genealogy

Version 4.3.0

`width` and `heigh` attributes added.

Version 4.2.0

`align` attribute deprecated.

Version 4.0.7

Default value for `escape` attribute changed from ‘unset’ to `html`.

Version 4.0.4

`html_id`, `escape` and `wraptag` attributes added.

Images

On this page:

- Syntax
- Attributes
- Examples
- Genealogy

Syntax

```
<txp:images />
```

The **images** tag is a *single* or *container* tag that Textpattern will use to gather a list of matching images uploaded via the Textpattern [[Images page]]. Utilising the other image tags in the suite image_info, image_url, image_date and if_thumbnail) you can display simple image galleries from this list.

If used as a *container* tag, it must be specified as an opening and closing pair of tags, like this:

```
<txp:images>
    ...contained statements...
</txp:images>
```

This is equivalent to putting the contained statements into a form named 'my_form' and using `<txp:images form="my_form" />`.

By default, the tag is context-sensitive, which means that in the absence of any filter attributes (`id`, `name`, `category`, `author`, `realname`, `extension`, `thumbnail`), it will return image IDs from the first of:

1. The currently viewed article's 'Article image' field;
2. Images matching the global category context;
3. Images matching the global author context;
4. All images.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `author="author (login) ID"`
  Filter the images by this list of author IDs who uploaded the pictures to Textpattern.
  Default: unset.
- `auto_detect="string context"`
  List of Textpattern contexts to consider when automatically searching for images. If you wish to turn off the automatic check, set this to `auto_detect=""`. You can choose from the following contexts:
  `article` to look in the article_image field.
  `category` to look in the URL for a category list.
  `author` to look in the URL for an author list.
  Default: `article, category, author`.
- `category="image category"`
  Filter the images by this list of category names as defined in the Categories administration panel.
- `extension=".extension"`
  Filter the images by this list of image extensions, including the leading dot. Example: `extension=".jpg, .png"`.
  Default: unset.
- `form="form name"`
  Use specified form template to process each image. If not used, and the container is empty, the tag will output a list of images that are compatible with image_display.
  Default: unset.
- `html_id="id number"`
  The HTML `id` attribute applied to the `wraptag`, if set, otherwise to the `img` tag.
  Default: unset.
- `id="integer"`
  Filter the images by this list of @id@s assigned at upload. The IDs can be found on the Images administration panel. The order of the ids overrides the default `sort` attribute.
  Default: unset.
- `limit="integer"`
  The number of images to display per page.
  Default: `0` (unlimited).
- `name="image name"`
  Filter the images by this list of image names as shown on the Images administration panel.
  Default: unset.
- `offset="integer"`
  The number of images to skip.
  Default: `0` (only effective if `limit` is set).
- `pageby="integer" (or "limit")`
  The number of images to jump forward or back when an older or newer link is clicked. Without this attribute, pagination is not available; you will simply see `limit` images. You may specify `pageby="limit"` to allow pagination to automatically follow the value of the `limit` attribute. Note: newer and older will paginate all content types at once.
  Default: unset.
- `realname="author name"`
  Filter the image list so it only includes images uploaded by this list of author real names. The author names may be

URL encoded (e.g. `realname="John+Smith"`) and thus could be read from the current `example.com/author/author+name` URL. Note that this attribute may incur one extra query per name, so if it is possible to use the raw author instead it will be faster.
Default: unset.

- `sort="sort value(s)"`
  How to sort the resulting image list. Specify an image attribute from the ones below and add either `asc` or `desc` to sort in ascending or descending order, respectively.
  Values:
  `alt`
  `author`
  `caption`
  `category`
  `date`
  `extension` (image extension)
  `h` (image `height` attribute)
  `id` (image id#)
  `name` (image name)
  `rand()` (random).
  `thumb_h` (image thumbnail `height` attribute)
  `thumb_w` (image thumbnail `width` attribute)
  `w` (image `width` attribute)
  Default: `name asc`.
- `thumbnail="boolean"`
  Filter the image list to only include images that have a thumbnail, or not.
  Values: unset (i.e.,all images), `1` (images that have a thumbnail) or `0` (images that do not have a thumbnail).
  Default: unset.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  CSS `class` attribute to apply to the image (or to the `wraptag`, if set).
  Default: tag name **or** unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="tag"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Varying attributes

This example shows the outcome of various attribute configurations to give you an idea of what to expect from the tag. More concrete examples follow.

```
<txp:images auto_detect="" />
```

Displays all images in the database.

```
<txp:images auto_detect="" sort="id desc" />
```

Display all images in the database, sorted by `id` in descending order.

```
<txp:images />
```

Context-sensitivity mode. Returns an image list based on the first of:

1. Article image field, if on an individual article page;
2. Images matching category, if on a category list page;
3. Images matching author, of on an author list page;
4. All images in the database.

```
<txp:images id="" />
<txp:images name="" />
<txp:images category="" />
```

No images displayed. This means that if you did some tag-in-tag magic such as: `category="<txp:custom_field name="my_cats" />"` it will show no images if the custom field is empty.

```
<txp:images id="2,3,6" />
```

Display images 2, 3, and 6.

```
<txp:images name="lion.jpg, zebra.jpg" />
```

The named images are displayed.

```
<txp:images name="pengiun.jpg" />
```

No images are displayed (mis-spelled image name).

```
<txp:images category="mammals, birds" />
```

All images in the named categories are displayed.

```
<txp:images category=", mammals, birds" />
```

All images in the named categories and any uncategorized images are displayed.

```
<txp:images category=" " />
```

Just uncategorized images are displayed (note that `category=","` also works, but a space looks better!).

```
<txp:images author="attenborough, morris" />
```

All images by author (ID) ‘attenborough’ and ‘morris’ are displayed.

```
<txp:images realname="David+Attenborough" />
```

All images by author ‘David Attenborough’ are displayed. This incurs one extra query to look up the author’s ID from the given real name.

```
<txp:images category="mammals, birds" author="attenborough, morris" />
```

All images in the named categories that are assigned to the named authors are displayed.

```
<txp:images category="mammals, birds" extension=".jpg" />
```

All JPEG images in the named categories are displayed.

```
<txp:images category="mammals, birds" extension=".jpg" author="attenborough, morris" />
```

All JPEG images in the named categories that are assigned to the named authors are displayed.

```
<txp:images extension=".gif" />
```

All GIF images are displayed.

```
<txp:images category="mammals, birds" thumbnail="1" />
```

All images in the named categories that have thumbnails assigned to them are displayed.

```
<txp:images thumbnail="1" />
```

All images that have thumbnails assigned to them are displayed.

```
<txp:images thumbnail="0" />
```

All images that do not have thumbnails assigned to them are displayed.

Example 2: Multiple pieces of information at once, using images tag as wrapper

```
<txp:images category="birds, mammals" thumbnail="1" sort="category asc">
    <txp:if_different>
        <h4>
            <txp:image_info type="category_title" />
        </h4>
    </txp:if_different>
    <txp:thumbnail wraptag="div" />
    <txp:image_info type="w, h" wraptag="div" break=" × " />
    by <txp:image_info type="author" />
</txp:images>
```

Shows the thumbnail of each image that has an assigned thumbnail image from the ‘mammals’ and ‘birds’ categories and, beneath each, show its dimensions ‘width’ x ‘height’ along with the author of the image. Since the list has been sorted by category, the `<txp:if_different>` conditional can be used to output the category title at the top of the list of images each time it changes.

Other tags used: if_different, image_info, thumbnail.

Example 3: Integration with third-party PHP resizing script (TimThumb)

TimThumb is a simple, flexible, PHP script that resizes images directly on your web server. Read the TimThumb documentation for basic installation instructions (also requires the GD image library). Then, for example, you can use the following:

```
<txp:images limit="6" category="gallery">
    <p>
        <a href="<txp:image_url />" title="Click to view original">
            <img src="<txp:site_url />timthumb.php?src=<txp:image_url />&amp;w=160" alt="<txp:image_info type='alt' />">
        </a>
    </p>
    <p>
        Author: <txp:image_author />
    </p>
</txp:images>
```

Creates a small gallery of 6 images from the category ‘gallery’. Uses the TimThumb script to proportionately resize a thumbnail version (160px wide) of the image automatically, and keep a cached version of the thumbnail for future visitors. Links the thumbnail to the original image, and lists the image author name below each thumbnail.

Other tags used: image_author, image_info, image_url, site_url.

Example 4: Using offset and limit for news pages

You can use the `offset` attribute to slice up your article_image list. By specifying a comma-separated list of image IDs in your ‘Article image’ field, this tag can iterate over them in groups. So, if your ‘Article image’ list contained four IDs you could treat your first image as the ‘Hero’ at the top of the article, displayed using:

```
<txp:images limit="1">
    <txp:article_image />
</txp:images>
```

And then later on you could drop in…

```
<txp:images offset="1" limit="3" wraptag="div" class="gallery">
    <txp:thumbnail />
</txp:images>
```

…to display the three remaining supporting images as thumbnails in a gallery, all taken from the Article Image field.

Other tags used: article_image, thumbnail.

Genealogy

Version 4.5.0

Sort order of `id` attribute maintained, unless overridden with `sort` attribute.

Version 4.3.0

Tag support added.

Tag reference index

Each Textpattern tag is described in detail on its respective reference page. See Tag basics for overview of tag types and modes of use.

Subcategories

TODO

Complete list of Textpattern tags

A

- article
- article custom
- article id
- article image
- article url title
- author
- author email
- authors

B

- body
- breadcrumb

C

- category
- category list
- category1
- category2
- comment anchor
- comment email
- comment email input
- comment id
- comment message
- comment message input
- comment name
- comment name input
- comment permlink
- comment preview
- comment remember
- comment submit
- comment time
- comment web
- comment web input
- comments
- comments count
- comments error
- comments form
- comments help
- comments invite
- comments preview
- css
- custom field

E

- else
- email
- error message
- error status
- excerpt
- expires

Keywords DEPRECATED [todo:pw]

```
<txp:keywords />
```

The keywords tag is a *single* tag that Textpattern will replace with the keywords associated with the article being displayed. The tag can be used in an article Form, or within pages (templates), either wrapped within a given article tag, or directly in the template itself so long as the context is with a single article (as opposed to an article list). For keywords metadata, see meta_keywords tag.

Attributes

This tag has no attributes.

Examples

Example 1: Display keywords in context of an article Form

In this example, keywords are used in an article Form along with other article components. The keywords themselves are used like a list of topical “tags”, *e.g.*, like you would use for more granular searching. The keywords would be presented (via CSS) horizontally (ideally) above the article’s excerpt.

```
<h3><txp:permlink><txp:title /></txp:permlink></h3>
<p><txp:posted /></p>
<p><txp:keywords /></p>
<txp:excerpt />
<txp:body />
```

Other tags used in example: permlink, title, posted, excerpt, and body

Example 2: Use keywords to fill <code>meta</code> element values

In this example, keywords are used directly in a *Page* (template) to insert

```
content=""
```

values in a

```
meta
```

element.

```
<meta name="keywords" content="<txp:keywords />" />
```

This works if the context is a single article being displayed.

The same result can be done by putting the keywords inside an if_individual_article tag (though a bit more code than necessary).

```
<txp:if_individual_article>
    <meta name="keywords" content="<txp:keywords />" />
</txp:if_individual_article>
```

In both of the above, if a given article has keywords associated with it, they will fill the meta value. If not, the meta element remains with no

```
content=""
```

values. This might be okay if you know for sure all articles will have keywords assigned.

If not, a metadata tag in your template with no value might not be desired. In which case you could set it up as a condition; if there are no keywords on the article, the entire meta element is not added to the template.

```
<txp:if_keywords>
    <meta name="keywords" content="<txp:keywords />" />
</txp:if_keywords>
```

Either of these uses might work well for certain pages where a single article is in context, but it doesn’t account for other page contexts, like home pages with complex content layouts, or pages displaying article lists (rather than single articles). In other words, these are not copy/paste solutions for every template in your site. You may, in fact, create some metadata values manually if the context of certain pages doesn’t change much.

Other tags used in example 2: if_individual_article and if_keywords.

Related Plugins

cbe_keywords uses keywords to offer tagging features.

chh_keywords provides tags to create a link list of an articleâ€™s keywords, browse lists of articles by keyword, and list all keywords in a tag cloud.

rah_metas, an SEO- and meta-tool, includes several keyword related attributes.

ras_if_article_keywords is a conditional tag that compares a keyword list as an attribute against the keywords associated with a particular article.

tru_tags uses keywords to offer tagging features.

wet_haystack allows site publishers to modify the default search behaviour by adding additional article fields to the set of indexed content, including keywords.

Lang

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:lang />
```

The **lang** tag is a *single* tag. Textpattern will replace this tag with the 2-letter code of the language which is set as the site’s language preference in the Languages administration panel, according to RFC 1766.

Attributes

This tag has no attributes.

Examples

Example 1: Define a document’s language

```
<!DOCTYPE html>
<html lang="<txp:lang />" dir="<txp:text item="lang_dir" />">
<head>
    <meta charset="utf-8">
    <title>
        <txp:page_title />
    </title>
</head>
```

Why you might do this? When declaring a DTD, namespace and language that a site is served, the `lang` attribute is useful for ensuring translators, search engines and content parsers handle the document in the correct manner.

Other tags used: page_title, text.

Link author

On this page:

Syntax

`<txp:link_author />`

The **link_author** tag is a *single* tag that Textpattern will replace with the author's name associated with the current link in a linklist. It can **only** be used inside `<txp:linklist />`.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="link type (boolean)"`
  Whether to hyperlink the author or not.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `section="section name"`
  Direct any linked author name to the nominated section instead of to the default (front) page.
  Default: unset.
- `this_section="boolean"`
  If set to `1`, the linked author name will direct users to an author list in the current section, otherwise author list from all sections is displayed.
  Values: `0` (no, all sections) or `1` (yes, this section only).
  Default: `0`.
- `title="boolean"`
  Whether to display the author's real name or login name.
  Values: `0` (login name) or `1` (real name).
  Default: `1`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: unset (see [[class cross-reference]]).
- `wraptag="tag"`
  HTML tag to wrap around the author name, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Genealogy

Version 4.3.0

Tag support added.

Link category [todo:pw]

```
<txp:link_category />
```

The link_category tag is a *single* tag which returns the link category as text. This tag is used in a “link” form or inside the linklist container tag to return information about the current link in a linklist.

Attributes

class name”
CSS class to apply to the wraptag Default: unset. **label**
Label for the top of the list. Default: unset. **labeltag text”**
HTML tag to wrap the label at the top of the list. Default: unset. **title**
Display link category name or its title. Values:

```
0
```

(category name) or

```
1
```

(category title). Default:

```
0
```

. **wraptag text”**
HTML tag to be used to wrap the category, without brackets. Default: unset.

Examples

Example 1: Display a link with class attribute

```
<a class="thislink" href="<txp:link_url />">
<txp:link_name escape="html" />
</a> : <txp:link_category title="1" />
```

Other tags used: link_url, link_name

Link date

On this page:

Syntax

```
<txp:link_date />
```

The **link_date** tag is a *single* tag which returns the date the link was created as text. This tag is used in a ‘links’ type form or inside the linklist container tag to display information about the current link.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values, `since`, `iso8601` (ISO 8601 reference), `w3cdtf` (W3CDTF reference), or `rfc822` (RFC 822 reference).
  Default: the ‘Date format’ set in preferences.
- `gmt="boolean"`
  Return either local time (according to the set time zone preferences) or GMT.
  Values: `0` (local time) or `1` (GMT).
  Default: `0`.
- `lang="ISO language code"`
  Format time string suitable for the specified language (locale).
  Values: locales adhere to ISO-639.
  Default: unset (time format set in the Preferences administration panel.

Examples

Example 1: Display a link with date and a class attribute

```
<a class="awesome-links" href="<txp:link_url />">
    <txp:link_name escape="html" />:
</a>
(<txp:link_date />)
```

Other tags used: link_name, link_url.

Link description

On this page:

Syntax

```
<txp:link_description />
```

The **link_description** tag is a *single* tag which is used to return the text from the ‘Description’ field as defined within the Links administration panel. This tag is used in a ‘links’ type form or inside a linklist container tag to display information about the current link.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`;

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap around description, specified without brackets (e.g. `wraptag="div"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display a link and its description field contents

Other tags used: link.

Genealogy

Version 4.0.7

Default value for `escape` attribute changed from unset to `html`.

Link feed link

On this page:

Syntax

```
<txp:link_feed_link />
```

The **link_feed_link** tag is a *single* tag. Textpattern will replace this tag with an anchor to the site’s ‘links’ RSS feed.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `category="category name"`
  Restrict to specified category. Note: the category name may be different to the title you typed when you created the category, as the names are sanitized for URL use. Check the Categories administration panel to ensure you are using the correct name.
  Value: category name.
  Default: current category.
- `flavor="value"`
  Whether to output a link to the RSS or Atom version of the feed.
  Values: `rss` or `atom`.
  Default: `rss`.
- `format="value"`
  Whether to output HTML `<a>` tag or `<link>` tag.
  Values: `a` or `link`.
  Default: `a`.
- `title="value"`
  HTML title attribute to be applied to link tag.
  Default: depends upon `flavor` used, either `RSS feed` or `Atom feed`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Note: `label` and `wraptag` attributes are applicable only when using `format` of `a` (`label` used as link text).

Examples

Example 1: Atom feed link with custom label

```
<txp:link_feed_link flavor="atom" label="Commerce links" />
```

Genealogy

Version 4.3.0

`class` attribute added.

Version 4.0.4

`format` attribute added.

Link id

On this page:

Syntax

```
<txp:link_id />
```

The **link_id** tag is a *single* tag which returns the numerical ID of the link. This tag is used in a ‘links’ type form or inside the linklist container tag to show information about the current link in the list.

Attributes

This tag has no attributes.

Examples

Example 1: Display link information

```
<txp:linklist category="dogs">
    Link:
    <a href="<txp:link_url />">
        <txp:link_name />
    </a>
    (id: <txp:link_id /> | category: <txp:link_category />)
</txp:linklist>
```

Other tags used: linklist, link_name, link_category.

Genealogy

Version 4.2.0

Tag support added.

Link name

On this page:

Syntax

```
<txp:link_name />
```

The **link_name** tag is a *single* tag which returns the name of the link as assigned on the links pane as text. This tag is used in a ‘links’ type form or inside the linklist container tag to display information about the current link.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`;

Examples

Example 1: Display a link with class attribute

```
<a class="awesome-links" href="<txp:link_url />">
    <txp:link_name />
</a>
```

Other tags used: link_url.

Genealogy

Version 4.0.7

Default value for `escape` attribute changed from unset to `html`.

Link to home

On this page:

Syntax

```
<txp:link_to_home>
```

The **link_to_home** tag is primarily a *container* tag that returns a link to the site’s home page. It will apply a hyperlink to whatever it wraps.

The tag can, however, be used as a *single* tag to generate a raw base URL of the site:

```
<txp:link_to_home />
```

In this mode it operates identically to site_url.

Attributes

Tag will accept the following attributes (**case-sensitive**):

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  CSS class attribute to apply to the anchor. Will be ignored if used as a single tag.
  Default: unset.

Examples

Example 1: Home page link with site’s name

```
<txp:link_to_home>
    <txp:site_name />
</txp:link_to_home>
```

Other tags used: site_name.

Link to next

On this page:

Syntax

```
<txp:link_to_next>
```

The **link_to_next** tag can be used as a *single* tag or a *container* tag to return the permanent URL of the next article by posting date.

If used as a container tag, the HTML required to output a hyperlink is returned; if used as a single tag, only the URL itself is returned.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `showalways="boolean"`
  Show the wrapped value even when no next article exists.
  Values: `0` (no) or `1` (yes).
  Default: `0`.

Examples

Example 1: Link to next article using the article title

```
<txp:link_to_next>
    <txp:next_title />
</txp:link_to_next>
```

Other tags used: next_title.

Example 2: Link to next article using static text

```
<txp:link_to_next showalways="1">
    Next
</txp:link_to_next>
```

This will always display the text ‘Next’, even when there is no next article.

Note: While `showalways` will enable this tag to display what is wrapped inside it, next_title":next-title returns nothing if there is no next title, so nothing is displayed. Use text, or the returned value, that you need displayed.

Example 3: Customising links

The container tag returns only a very basic link, which doesn’t allow for customising the link title, or adding a CSS class, etc. Using the tag in its single tag capacity opens up a lot more possibilities.

For example, to give the link an HTML `title` attribute of the next article’s title, and also apply a `class` to it:

```
<a class="link--next" href="<txp:link_to_next />" title="<txp:next_title />">
    Next article →
</a>
```

Other tags used: next_title.

Link to prev

On this page:

Syntax

```
<txp:link_to_prev>
```

The **link_to_prev** tag can be used as a *single* tag or a *container* tag to return the permanent URL of the previous article by posting date.

If used as a container tag, the HTML required to output a hyperlink is returned; if used as a single tag, only the URL itself is returned.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `showalways="boolean"`
  Show the wrapped value even when no previous article exists.
  Values: `0` (no) or `1` (yes).
  Default: `0`.

Examples

Example 1: Link to previous article using its title

```
<txp:link_to_prev>
    <txp:prev_title />
</txp:link_to_prev>
```

Other tags used: prev_title.

Example 2: Link to previous article using static text

```
<txp:link_to_prev showalways="1">
    Previous
</txp:link_to_prev>
```

This will always display the text ‘Previous’, even when there is no previous article.

Note: While `showalways` will enable this tag to display what is wrapped inside it, prev_title returns nothing if there is no previous title, so nothing is displayed. Use text, or the returned value, that you need displayed.

Example 3: Customising links

The container tag returns only a very basic link, which doesn’t allow for customising the link title, or adding a CSS class, etc. Using the tag in its single tag capacity opens up a lot more possibilities.

For example, to give the link an HTML `title` attribute of the previous article’s title, and also apply a `class` to it:

```
<a class="link--prev" href="<txp:link_to_prev />" title="<txp:prev_title />">
    ← Previous article
</a>
```

Other tags used: prev_title.

Link url

On this page:

Syntax

```
<txp:link_url />
```

The **link_url** tag is a *single* tag which returns the URL of the link as text. This tag is used in a ‘links’ type form or inside the linklist container tag to show information about the current link in the list.

Attributes

This tag has no attributes.

Examples

Example 1: Display a link with class attribute

```
<a class="awesome-links" href="<txp:link_url />">
    <txp:link_name />
</a>
```

Other tags used: link_name.

Link

On this page:

Syntax

```
<txp:link />
```

The **link** tag is a *single* tag which is used to return an HTML hyperlink defined in the Links administration panel. It uses the ‘Title’ field as the link’s text.

This tag is used in ‘links’ type forms or inside the linklist container tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `id="integer"`
  Specifies the `id`, assigned at creation of the link, to display. Can be found on the Links administration panel. If both `name` and `id` are specified, `name` is used while `id` is ignored.
- `name="link name"`
  Specifies which link to display by its link `name` as shown on the Links administration panel.
- `rel="relation"`
  HTML rel attribute to be applied to link.
  Default: unset.

Examples

Example 1: Display a link and its description

```
<p>
    <txp:link />:
    <txp:link_description />
</p>
```

Other tags used: link_description.

Genealogy

Version 4.6.0

`id` and `name` attributes added.

Linkdesctitle

On this page:

Syntax

```
<txp:linkdesctitle />
```

The **linkdesctitle** tag is a *single* tag which is used to return an HTML hyperlink, defined within the Links administration panel.

It uses the ‘Title’ field as the link’s text; the ‘Description’ field contents will be displayed as an anchor `title` attribute. This tag is used in a link form or inside the linklist container tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `rel="relation"`
  HTML rel attribute to be applied to link.
  Default: unset.

Examples

Example 1: Display a link and its ‘Title’ field contents

```
<p>
    <txp:linkdesctitle />
</p>
```

Linklist

On this page:

Syntax

```
<txp:linklist />
```

The **linklist** tag is a *single* or a *container* tag which is used to produce a list of links from the predefined list created on the Links administration panel.

If used as a container, it must be specified as an opening and closing pair of tags, like this:

```
<txp:linklist>
    ...contained statements...
</txp:linklist>
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `author="author login name"`
  Restrict to links with the specified author.
  Default: unset.
- `auto_detect="string context"`
  List of Textpattern contexts to consider when automatically searching for links. If you wish to turn off the automatic check, set this to `auto_detect=""`.
  Values: `category` (to look in the URL for a category list) and/or `author` (to look in the URL for an author list).
  Default: `category, author`.
- `category="category name(s)"`
  Restrict to links from specified categories.
  Values: (comma separated list of) category name(s). **Note:** category names may be different to the ‘Title’ you typed when you created the category, as the names are sanitized for URL use. Check the Categories administration panel to ensure you are using the correct names.
  Default: unset.
- `form="form name"`
  Use specified form.
  Default: `plainlinks`.
- `id="integer"`
  Filter the links by this list of @id@s assigned at link creation time. The IDs can be found on the Links administration panel.
  Default: unset.
- `limit="integer"`
  Number of links to display.
  Default: `0` (no limit).
- `offset="integer"`
  The number of links to skip.
  Default: `0`.
- `pageby="integer or limit"`
  Number of links to jump each page. Without this attribute, you cannot navigate using the newer and older tags. Usually you will want to track the `limit` attribute. Use `pageby="limit"` to do this, which means you will not have to amend two values if you subsequently decide to alter the `limit`.
  Default: unset
- `realname="author real name"`
  Restrict to links with the specified author name.
  Default: unset.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `category`.

`date`.
`description`.
`id` (link id#).
`linkname`.
`linksort`.
`rand()` (random).
`url`.
Default: `linksort asc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: List of links from specified category

```
<txp:linklist form="Links" category="general" limit="10" sort="linksort" wraptag="p" />
```

Example 2: An ordered list of links

```
<txp:if_category name="100">
    <txp:linklist label="First Floor" category="First" wraptag="ol" break="li" />
</txp:if_category>
<txp:if_category name="200">
    <txp:linklist label="Second Floor" category="Second" wraptag="ol" break="li" />
</txp:if_category>
```

This example uses the displayed page's category as the criterion for choosing the linklist's category.

Other tags used: if_category.

Example 3: Used as a container tag

```
<txp:linklist wraptrag="ol" break="li">
    <txp:link /><br />
    <txp:link_description /><br />
    <txp:linkdesctitle />
</txp:linklist>
```

The tags within the container are repeated for each link provided by the linklist tag.

Other tags used: link, linkdesctitle, link_description.

Genealogy

Version 4.5.0

`id` attribute added.

Version 4.3.0

`pageby` attribute added to enable paging via newer and older tags.
`author` and `realname` attributes added.
`auto_detect` attribute added to allow automatic (URL-based) contextual listings.

Meta author

On this page:

Syntax

```
<txp:meta_author />
```

The **meta_author** tag is a *single* tag, used in the head of an individual article page template. Textpattern will replace this tag with an HTML meta tag as follows:

```
<meta name="author" content="Article author's name" />
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`;
- `format="boolean"`
  Display as `<meta>` tag or as raw value.
  Values: `meta` or unset.
  Default: `meta` (display as a `<meta>` tag).
- `title="boolean"`
  Whether to display the author's login name or real name.
  Values: `0` (login name) or `1` (real name).
  Default: `0`.

Examples

Example 1: Use article author for meta tag content

Tag in the head of an individual article page (article's author name is 'Biff Tannen'):

```
<txp:meta_author />
```

This results in the following:

```
<meta name="author" content="Biff Tannen" />
```

Genealogy

Version 4.6.0

`escape` and `format` attributes added.

Version 4.3.0

`title` attribute added.

Meta description

On this page:

Syntax

```
<txp:meta_description />
```

The **meta_description** tag is a *single* tag used in a variety of ways to display meta description content. One is to output description text as the content of a standard HTML `<meta>` tag in the `<head>` of a page template:

```
<meta name="description" content="{your description}" />
```

The other is to output raw description text anywhere else:

```
{your description}
```

The tag is context aware, in that it will use article descriptions (if available) on individual article pages, and use section and category descriptions (again, if available) on listing pages. The tag can be added directly in a [[Page template]] or as part of an article [[Form template]]. Either way, you may also use it in a given article tag.

Note: The corresponding description field has a 255 character limit by default, which includes spaces and punctuation. This is the MySQL database default. You can edit the default using phpMyAdmin, for example.

Attributes

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`;
- `format="value"`
  Display as `<meta>` tag or as raw value.
  Values: `meta` or unset.
  Default: `meta` (display as a `<meta>` tag).
- `type="value"`
  The context from which to display meta information.
  Values: `article`, `section`, or `category`. The `category` may also be augmented to include its type, e.g. `category.image` or `category.file` to display description information from the nominated category type.
  Default: empty (i.e.,automatic, depending on where the tag is used).

Examples

Example 1: Use article or section’s description for meta tag content

```
<head>
  <title>Example 3 page</title>
  <txp:meta_description />
</head>
```

The **meta_description** tag returns an HTML `meta` tag, populated with an article’s description if the visitor is viewing an individual article. If viewing an article list (landing) page, the tag will return the description from the Section instead.

The tag should normally be placed in your template’s ‘head’ section, between the opening and closing HTML `<head>` tags.

The above will output description metadata with `content=""` populated with the text set in the article or section’s ‘Description’ field (the tag returns nothing if no description is set for either).

Example 2: Use article’s description for Twitter Card description (without meta tag)

```
<txp:if_individual_article>
    <meta name="twitter:description" content="<txp:meta_description format="" />" />
```

```
</txp:if_individual_article>
```

The above will output the description as a raw value (not enclosed in a `<meta>` tag) populated with the text set in the article's 'Description' field.

Other tags used: if_individual_article.

Example 3: Display section meta description as tooltip in section list

```
<txp:section_list>
    <a href="/<txp:section />" title="<txp:meta_description format="" />">
        <txp:section title="1" />
    </a>
</txp:section_list>
```

Other tags used: section_list, section.

Example 4: Override automatic type

```
<head>
    <txp:meta_description />
</head>
...
<h1>
    Section:
    <txp:section title="1" />
</h1>
<txp:if_article_list>
    <h2>Introduction</h2>
    <p>
        <txp:meta_description type="section" format="" />
    </p>
</txp:if_article_list>
<txp:article />
```

The first time the tag is used in the `<head>`, the section meta description will be output as a `<meta>` tag if viewing the section landing page, or the article meta description if viewing an individual article.

The second instance is only shown if on a landing page, and will force the display of the section meta description as an introductory paragraph.

Other tags used: section, if_article_list, article.

Genealogy

Version 4.6.0

Tag support added.

Meta keywords

On this page:

Syntax

```
<txp:meta_keywords />
```

The **meta_keywords** tag is a *single* tag used in one of two ways. One is to output keyword data as the content of a standard HTML `<meta>` tag in the `<head>` of a page template:

```
<meta name="keywords" content="{your article's Keywords}" />
```

The other is to output raw keyword text anywhere else:

```
{your article keywords}
```

Context of use must be within a single article (as opposed to an article list). The tag can be added directly in a [[Page template]] or as part of an article [[Form template]]. Either way, you may also place it in a given article tag.

Note: The corresponding keywords field has a 255 character limit by default, which includes spaces and commas. This is the MySQL database default. You can edit the default using phpMyAdmin, for example.

Attributes

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`;
- `format="value"`
  Display as `<meta>` tag or as raw value.
  Values: `meta` or unset.
  Default: `meta` (display as a `<meta>` tag).
- `separator="value"`
  Character to be used as the keywords separator.
  Default: unset.

Examples

Example 1: Use article’s keywords for meta tag content

```
<txp:meta_keywords />
```

The **meta_keywords** tag returns an HTML `meta` tag, populated with an article’s keywords. The tag should always be placed in your template’s ‘head’ section, between the opening and closing HTML `<head>` tags.

The above will output keywords metadata with `content=""` populated with the list of keywords set in the article’s ‘Keywords’ field (the tag returns nothing if no keywords are set for an article). For example, if your article’s ‘Keywords’ field contains `sauce, caramel, sugar`, the tag will output the following:

```
<meta name="keywords" content="sauce,caramel,sugar" />
```

Example 2: Display keywords in context of an article form

```
<h1>
    <txp:permlink>
        <txp:title />
    </txp:permlink>
</h1>
<p>
    Date:
```

```
    <txp:posted />
</p>
<p>
    Keywords:
    <txp:meta_keywords format="" separator=", " />
</p>
<txp:excerpt />
<txp:body />
```

In this example, keywords are used in a Textpattern ‘article’ type [[Form template]] along with other article components. The keywords themselves are used like a list of topical ‘tags’, e.g. like you would use for more granular searching. The keywords would be presented above the article’s excerpt.

Other tags used: permlink, title, posted, excerpt, body.

Genealogy

Version 4.6.0

`escape`, `format` and `separator` attributes added.

Modified

On this page:

Syntax

```
<txp:modified />
```

The **modified** tag is a *single* tag which is used to return the modification date of the article being displayed. The format is determined by the settings specified in the Date Format, or Archive Date Format, fields in the Preferences administration panel.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values.
  Default: the ‘Archive date format’ set in preferences.
- `gmt="boolean"`
  Return either local time (according to the set time zone preferences) or GMT.
  Values: `0` (local time) or `1` (GMT).
  Default: `0`.
- `lang="ISO language code"`
  Format time string suitable for the specified language (locale).
  Values: locales adhere to ISO-639.
  Default: unset (time format set in the Preferences administration panel.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `wraptag="element"`
  HTML tag to wrap around output, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display ‘since’ format date setting

```
<p>
    Modified:
    <txp:modified format="since" />
</p>
```

This would result in the following HTML output:

```
<p>Modified: 29 days ago</p>
```

Example 2: Display custom date format

```
<p>
    Modified:
    <txp:modified format="%b %d, %Y" />
</p>
```

This would result in the following HTML output:

`<p>modified: May 28, 2005</p>`

Genealogy

Version 4.5.0

`wraptag` and `class` attributes added.

Version 4.0.7

Tag support added.

Newer

On this page:

Syntax

```
<txp:newer />
```

The **newer** tag is both a *single* tag and a *container* tag. The tag should be used in a page after an article tag.

Textpattern will replace this tag with a link to the next list of articles in the sort order. The container tags wrap the text or tag assigned to the link. As a single tag it outputs the URL for the next list page.

An article list consists of the assigned number of articles set by the article tag. If there are no articles available having ‘Newer’ status (articles ranked higher, or newer, in the present sort criteria than the present top of page article) `<txp:newer />` will not display unless the `showalways` attribute is set to `1`. It is normally seen used in tandem with older.

Given a `<txp:article limit="5" />` tag on the page in question, `<txp:newer />` will page down five articles at a time from the most oldest post forward in time to the most recently posted article.

Note: This tag is context-sensitive, meaning it will only fetch content from the section or category being viewed.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`.
- `showalways="boolean"`
  Show wrapped value even when no older page exists.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `title="text"`
  HTML title attribute to be applied to link.
  Default: unset.

Examples

Example 1: Container tag – link with text

```
<txp:newer>Newer</txp:newer>
```

Example 2: Single tag – link with image

```
<a href="<txp:newer />">
    <txp:image name="right-arrow.gif" />
</a>
```

Other tags used: image.

Example 3: Container tag – link with image

```
<txp:newer>
    <txp:image name="right-arrow.gif" />
</txp:newer>
```

The difference between examples 2 and 3 is that the tags in example 2 will display the image even if there are no newer articles, those used in example 3 won’t.

Other tags used: image.

Genealogy

Version 4.3.0

`title` attribute reintroduced after being accidentally removed.
`escape` attribute added.

Next title

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:next_title />
```

The **next_title** tag is a *single* tag which Textpattern will replace with the title of the next article in the sort order.

The container tag link_to_next wraps the text or tag and assigns the link.

Attributes

This tag has no attributes.

Examples

Example 1: Link to next article by its title

```
<txp:link_to_next>
    <txp:next_title />
</txp:link_to_next>
```

Other tags used: link_to_next.

Older

On this page:

Syntax

```
<txp:older />
```

The **older** tag is both a *single* tag and a *container* tag. The tag should be used in a page after an article tag.

Textpattern will replace this tag with a link to the next list of articles in the sort order. The container tags wrap the text or tag assigned to the link. As a single tag it outputs the URL for the previous list page.

An article list consists of the assigned number of articles set by the article tag. If there are no articles available having 'Older' status (articles ranked lower, or later, in the present sort criteria than the present bottom of page article) `<txp:older />` will not display unless the `showalways` attribute is set to `1`. It is normally seen used in tandem with newer.

Given a `<txp:article limit="5" />` tag on the page in question, `<txp:older />` will page down five articles at a time from the most recent post back in time to the oldest.

Note: This tag is context-sensitive, meaning it will only fetch content from the section or category being viewed.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&`.
  Values: `html` or unset.
  Default: `html`.
- `showalways="boolean"`
  Show wrapped value even when no older page exists.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `title="text"`
  HTML title attribute to be applied to link.
  Default: unset.

Examples

Example 1: Container tag – link with text

```
<txp:older>Older</txp:older>
```

Example 2: Single tag – link with image

```
<a href="<txp:older />">
    <txp:image name="left-arrow.gif" />
</a>
```

Other tags used: image.

Example 3: Container tag – link with image

```
<txp:older>
    <txp:image name="left-arrow.gif" />
</txp:older>
```

The difference between examples 2 and 3 is that the tags in example 2 will display the image even if there are no older articles, those used in example 3 won't.

Other tags used: image.

Genealogy

Version 4.3.0

`escape` attribute added.

Output form

On this page:

Syntax

```
<txp:output_form />
```

The **output_form** tag can be used as a *single* or a *container* tag. Textpattern will replace this tag with the content resulting from the form template called by the tag.

For the container tag usage, see the yield tag.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `form="form name"`
  Use specified form.
  Default: unset (no output).

Examples

Example 1: Manage small pieces of static text

```
<txp:output_form form="copyright" />
```

You can use this tag in combination with a form to create small pieces of text that would not otherwise be managed as a regular article. For example you might define the copyright conditions of content on your site in a form and add that to one or more places via the output_form tag. Name the form `copyright`, save it as type `misc` and call the form using the tag structure.

Note that Staff Writers and Freelancers can not edit the contents of forms.

Example 2: Manage header for all pages

Suppose you want to manage the `<head>` section of your page template as a single-sourced block of content. You can create a form called `head` and save it as type `misc`. The content of the form might look like this for example:

```
<head>
    <meta charset="utf-8">
    <title>
        <txp:page_title />
    </title>
    <txp:css format="link" media="" />
    <meta name="generator" content="Textpattern CMS">
    <meta name="robots" content="index, follow, noodp, noydir">
    <txp:feed_link flavor="rss" format="link" label="RSS" />
</head>
```

Then in each of your pages, you insert the header using…

```
<txp:output_form form="head" />
```

…which will add this `<head>` to all the pages automatically.

The advantage of this is that when you edit your page header, you can do so once in the form template and it will update all instances of use in your different pages at the same time.

Genealogy

Version 4.2.0

Can be used as a container tag.

Page title

On this page:

Syntax

```
<txp:page_title />
```

The **page_title** tag is a *single* tag that displays text depending on the context it is used. Its primary purpose is for outputting information suitable for the HTML `<title>` tag.

Results appear as follows:

1. Article list: `Your site name`.
2. Articles by category: `Your site name : Category title`.
3. Search results page: `Your site name : Search results: Search term`.
4. Single article page: `Your site name : Article name`.
5. Comments display: `Comments on: Article name`.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `separator="character(s)"`
  The character sequence you want between each piece of information.
  Default: `:` (a colon).

Examples

Example 1: Show page titles with custom separator

Page url

On this page:

Syntax

```
<txp:page_url />
```

The **page_url** tag is a *single* tag. It is used to return a particular component of the URL from the current page being displayed.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `type="type"`
  Specifies which component of the current page’s URL will be returned.
  Values:
  `request_uri`: current article’s URL-title including any query string.
  `id`: current article’s id on a single article page.
  `s`: current page’s section.
  `c`: current page’s category.
  `q`: search query string.
  `pg`: current page number in article list mode.
  `month`: current page’s month on time based article lists.
  `author`: current page’s author on article lists filtered by author.
  `status`: HTTP error response (200, 404).
  `css`: current style sheet name.
  `page`: current page template name.
  Default: `request_uri`.

Examples

Example 1: Show the current article’s ID, HTTP status and section

```
<p>
    Article ID:
    <txp:page_url type="id" />,
    From section:
    <txp:page_url type="s" />
    (Result: <txp:page_url type="status" />)
</p>
```

This would result in the following, for example:

```
<p>
    Article ID:
    88,
    From section:
    Tasty pea recipes
    (Result: 200)
</p>
```

Password protect

On this page:

Syntax

```
<txp:password_protect />
```

The **password_protect** tag can be used as either a *single* tag or *container* tag. When Textpattern encounters the password protect tag it causes the user to be prompted for a username and password, if these match the attributes set in the tag, the user is allowed access to the site/content. The tag can go anywhere, from page templates, to articles and forms.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `login="text"`
  The username the user has to enter.
  Default: unset.
- `pass="text"`
  The password the user has to enter.
  Default: unset.

Examples

Example 1: Cause Textpattern to prompt the user for a login

```
<txp:password_protect login="theusername" pass="thepassword" />
```

Note: It is not adequate to protect a single section. This is not due to the tag itself, but rather because of how Textpattern handles URLs. By changing the URL an article can be rendered with a different section template, which would mean that the tag in the protected section would not be rendered and could not protect the article – only page requests that would be rendered in that section would be protected.

Example 2: Container tag

```
<txp:password_protect>
    <p>
        This content is only visible to authenticated users.
        <a>Click here</a> for free diamonds.
    </p>
</txp:password_protect>
```

When used as a container, the tag hides the wrapped content if the user doesn’t have access to it. When using authentication, it prompts for login too, but doesn’t kill the page.

Genealogy

Version 4.6.0

Can be used as a container tag.

Permlink

On this page:

Syntax

The **permlink** can be used as a *single* tag or a *container* tag to return the permanent url of the article being displayed.

If used as a container tag, the HTML required to output a hyperlink is returned; if used as a single tag, only the URL itself is returned.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `id="integer"`
  Specifies the article `id`, assigned at creation of the article, to link. Can be found on the Articles administration panel.
  Default: unset (current article).
- `title="text"`
  HTML `title` attribute.
  Default: unset.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `a` tag when used as a container tag.
  Default: tag name or unset (see [[class cross-reference]]).
- `style="style rule"`
  Inline CSS `style` rule. It's recommended that you assign CSS rules via `class` attribute instead.
  Default: unset.

Examples

Example 1: Single tag

```
<txp:permlink />
```

This would result in something like:

```
http://example.com/index.php?id=2
```

Example 2: Container tag

```
<txp:permlink>
    <txp:title />
</txp:permlink>
```

This would result in the following:

```
<a rel="bookmark" href="http://example.com/index.php?id=2">Article title</a>
```

Other tags used: title.

Example 3: Customising permanent links

```
<txp:permlink class="awesome-article">
    <txp:title />
</txp:permlink>
```

By default `<txp:permlink />` returns only a very basic link, which doesn't allow for customising the link title, or adding a CSS class, etc. Using the tag in its single tag capacity opens up a lot more possibilities.

For example, to give the permanent link an HTML `title` attribute of the article's title, and also apply a `class` to it:

```
<a class="awesome-article" href="<txp:permlink />">
    <txp:title />
</a>
```

Other tags used: title.

Php

On this page:

Syntax

`<txp:php>`

Textpattern’s **php** tag is a *container* tag that provides the same output abilities as `<?php //Code goes here... ?>`. Textpattern’s tag version is used like this:

```
<txp:php>
    // Code goes here...
</txp:php>
```

Control over where this tag is allowed to appear (i.e. the privileges required to allow it to appear in pages and forms) are governed by settings in the Preferences administration panel.

Attributes

This tag has no attributes.

Notes on use

Using PHP inside of an article

When inserting markup or PHP into the content boxes of a Textpattern article:

1. Don’t include the usual PHP delineations: i.e. `<?php ... ?>`.
2. Use PHP as you would use normal PHP, not interspersed with markup. For example, inside the PHP tags, use PHP’s echo command to output HTML, rather than writing HTML directly.
3. Surround the code with both the special `<notextile>` tag and Textpattern `<txp:php>` tag to **disable Textile** parsing:

```
<notextile>
    <txp:php>
        ...code goes here...
    </txp:php>
</notextile>
```

Equivalent programmatic names

All Textpattern tags have equivalent programmatic names which are *exactly* the same as the tag names. For example, `<txp:recent_articles />` is `recent_articles()`.

Arrays must be passed to all functions

You must pass an array to all tag functions, even if there are no attributes to set. For tags that require no attributes or those that you do not wish to modify the defaults, pass an empty array, e.g. `category1(array());`.

Examples

Example 1: Display PHP server library information

```
<txp:php>
    phpinfo();
</txp:php>
```

Example 2: Show the current linked category title

```
<txp:php>
    echo "The current Textpattern category is: "
        .category(array(
            'title'   => '1',
            'link'    => '1',
            'wraptag' => 'div'
        ));
</txp:php>
```

Popup

On this page:

Syntax

`<txp:popup />`

The **popup** tag is a *single* tag. Textpattern will replace this tag with a popup selector for browsing by section or category.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `section="section_name"`
  Jump to the selected category for the named section.
  Default: unset.
- `this_section="boolean"`
  Jump to the selected category for the currently active section.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `type="type"`
  Values: `s` (section), `c` (category).
  Default: `c`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Browse by category popup selector

```
<txp:popup type="c" wraptag="p" />
```

Example 2: Popup selector with custom label

```
<txp:popup label="Browse this site" type="c" wraptag="p" />
```

Genealogy

Version 4.3.0

`class` attribute added.

Posted

On this page:

Syntax

```
<txp:posted />
```

The **posted** tag is a *single* tag which is used to return the publish date of the article being displayed. The format is determined by the settings specified in the ‘Date format’ or ‘Archive date format’ fields in the Preferences administration panel.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values, `since`, `iso8601` (ISO 8601 reference), `w3cdtf` (W3CDTF reference), or `rfc822` (RFC 822 reference).
  Default: the ‘Date format’ set in preferences.
- `gmt="boolean"`
  Return either local time (according to the set time zone preferences) or GMT.
  Values: `0` (local time) or `1` (GMT).
  Default: `0`.
- `lang="ISO language code"`
  Format time string suitable for the specified language (locale).
  Values: locales adhere to ISO-639.
  Default: unset (time format set in the Preferences administration panel.

Common presentational attributes

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: unset (see [[class cross-reference]]).
- `wraptag="element"`
  HTML element to wrap (markup) the posted date, specified without brackets (e.g. `wraptag="p"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: ‘since’ format date setting

```
<p>
    Posted:
    <txp:posted format="since" />
</p>
```

This would result in the following HTML output:

```
<p>Posted: 29 days ago</p>
```

Example 2: Custom format date setting

```
<p>
    Posted:
    <txp:posted format="%b %d, %Y" />
</p>
```

This would result in the following HTML output:

```
<p>Posted: Feb 03, 2014</p>
```

Example 3: Extended custom format date setting

```
<p>
    Posted:
    <time datetime="<txp:posted format="iso8601" />">
        <txp:posted class="time-day" wraptag="span" format="%d" />
        <txp:posted class="time-month" wraptag="span" format="%b" />
        <txp:posted class="time-year" wraptag="span" format="%Y" />
    </time>
</p>
```

This would result in the following HTML output:

```
<p>
    Posted:
    <time datetime="2014-02-03T10:43:39Z">
        <span class="time-day">03</span>
        <span class="time-month">Feb</span>
        <span class="time-year">2014</span>
    </time>
</p>
```

This provides styling hooks for each date part.

Genealogy

Version 4.0.4

`class` and `wraptag` attributes added.

Prev title

On this page:

Syntax

```
<txp:prev_title />
```

The **prev_title** tag is a *single* tag which Textpattern will replace with the name (text) of the previous article in the sort order.

The container tag link_to_prev wraps the text or tag and assigns the link.

Attributes

This tag takes no attributes.

Examples

Example 1: Display a link to the previous article when displaying individual articles

```
<txp:link_to_prev>
    <txp:prev_title />
</txp:link_to_prev>
```

Other tags used: link_to_prev.

Recent articles

On this page:

Syntax

`<txp:recent_articles />`

The **recent_articles** tag is a *single* tag which is used to produce a list of permanent links to recent articles by title.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `category="category name"`
  Restrict to articles from specified category/categories. Note: the category names may be different to the title you typed when you created the category, as the names are sanitized for URL use. Check the Categories administration panel to ensure you are using the correct names.
  Values: (comma separated list of) category name(s).
  Default: unset, retrieves from all categories.
- `limit="integer"`
  The number of articles to display.
  Default: `10`.
- `no_widow="boolean"`
  Control widows and overrule 'widows' setting in the Preferences administration panel.
  Values: `0` allow the last word in the title to appear on its own line, i.e.,the title content is rendered unchanged, `1` ensure the last word is not left on its own line – Textpattern inserts an invisible code (a non-breaking space) between the last two words.
  Default: as set in the Preferences administration panel.
- `offset="integer"`
  The number of articles to skip.
  Default: `0`.
- `section="section name"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: unset, retrieves from all sections.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `authorid` (author name).
  `category1`.
  `category2`.
  `comments_count`.
  `custom_1` through `custom_10` (from Textpattern 4.2.0 onwards: `custom_n` where 'n' is the number of your custom field – for numeric values use `(custom_n+0)`).
  `id` (article id#).
  `image` (article image id#).
  `keywords`.
  `lastmod` (date last modified).
  `lastmodid` (author name of last modification).
  `Posted` (date posted).
  `rand()` (random).
  `section`.
  `status`.
  `title`.
  `url_title`.
  Default: `posted desc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).

- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Labelled list of recent articles

```
<txp:recent_articles label="Latest and greatest" limit="5" />
```

Produces a list of the 5 most recent articles, along with the label 'Latest and greatest'.

Example 2: List of recent articles by category, with a heading

```
<txp:recent_articles label="Latest" labeltag="h3" break="li" wraptag="ol" category="code" sort="Section desc" />
```

Produces a numerical list of the 5 most recent articles categorized with `code`, along with the heading 'Latest'.

Example 3: Offsetting a recent article list

```
<txp:recent_articles label="Other recent articles" limit="5" offset="1" />
```

Produces a list of the 5 most recent articles *apart* from the most recent one, along with the label 'Other recent articles'.

Genealogy

Version 4.6.0

`offset` attribute added.

Version 4.0.6

Added support for comma separated lists for `section` and `category` attributes.

Recent comments

On this page:

Syntax

```
<txp:recent_comments />
```

The **recent_comments** tag is a *single* or a *container* tag. Textpattern will replace this tag with a list of permanent links to recent comments. This list will be displayed with the format:

```
User's Name (Article Name)
```

If used as a container, the tag must be specified as an opening and closing pair, like this:

```
<txp:recent_comments>
    ...contained statements...
</txp:recent_comments>
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `form="form name"`
  Use specified form template.
  Default: unset (if left empty, the commenter's name and article title in which the comment was made will be permlinked).
- `limit="integer"`
  Number of comments to display.
  Default: `10`.
- `offset="integer"`
  Number of comments to skip.
  Default: `0`.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `discussid` (comment ID#)
  `email`
  `ip` (IP address).
  `message`
  `name`
  `parentid` (article ID#)
  `posted`
  `rand()` (random).
  `web`
  Default: `posted asc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`

HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Labelled list of recent comments

```
<txp:recent_comments label="Recent Comments" wraptag="p" break="br" />
```

Example 2: Recent comments as an ordered list

```
<txp:recent_comments label="Recent Comments" wraptag="ol" break="li" limit="25" />
```

This example also increases the results to a maximum of 25 list items (instead of the default 10).

Genealogy

Version 4.0.7

Can be used as a container tag.
`offset` attribute added.

Related articles

On this page:

Syntax

`<txp:related_articles>`

The **related_articles** tag can be used as either a *single* tag or a *container* tag, and is used to produce a list of related (by category) articles.

If used as a container, it must be specified as an opening and closing pair of tags, like this:

```
<txp:related_articles>
    ...contained statements...
</txp:related_articles>
```

This is equivalent to putting the contained statements into a form named ‘my_form’ and using `<txp:related_articles form="my_form" />`.

Related matches are selected as follows:

If a match to category 1 (category1) or category 2 (category2) of the individual article being displayed is found in either category 1 or category 2 of any other article in the database, it will cause that article to be listed as related.

If category 1 of the individual article being displayed is left blank and category 2 is not blank, then all other articles are selected as being related. If both categories are left blank, then no articles are selected.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `form="form name"`
  Use specified form.
  Default: unset. If left empty, the permlinked article title(s) will be displayed.
- `limit="integer"`
  Number of articles to display.
  Default: `10`.
- `match="category number(s)"`
  Restrict to articles related by specified category.
  Values: `Category1`, `Category2`, `Category1,Category2`.
  Default: `Category1,Category2`.
- `no_widow="boolean"`
  Control widows and overrule ‘widows’ setting in the Preferences administration panel.
  Values: `0` allow the last word in the title to appear on its own line, i.e.,the title content is rendered unchanged, `1` ensure the last word is not left on its own line – Textpattern inserts an invisible code (a non-breaking space) between the last two words.
  Default: as set in the Preferences administration panel.
- `section="section name(s)"`
  Restrict to articles from specified section(s).
  Values: (comma separated list of) section name(s).
  Default: unset, retrieves from all sections.
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `authorid` (author name).
  `category1`.
  `category2`.
  `comments_count`.
  `custom_1` through `custom_10` (from Textpattern 4.2.0 onwards: `custom_n` where ‘n’ is the number of your custom field – for numeric values use `(custom_n+0)`).

`id` (article id#).
`image` (article image id#).
`keywords`.
`lastmod` (date last modified).
`lastmodid` (author name of last modification).
`Posted` (date posted).
`rand()` (random).
`section`.
`status`.
`title`.
`url_title`.
Default: `posted desc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Labelled list of related articles

```
<txp:related_articles label="Related to this:" limit="5" />
```

Example 2: Related articles as an unordered list

```
<txp:related_articles label="Related" limit="10" break="li" wraptag="ul" />
```

Example 3: Used as a container tag

```
<txp:related_articles label="Related" labeltag="h3" limit="10" break="li" wraptag="ul">
    <txp:permlink>
        <txp:title />
    </txp:permlink>
    by
    <txp:author />
</txp:related_articles>
```

Other tags used: permlink, title, author.

Genealogy

Version 4.0.7

Can be used as a container tag.
`form` and `no-widow` attributes added.

Version 4.0.6

Support added for comma separated list for `section` attribute.

Rsd [todo:pw]

DEPRECATED 4.6.0

On this page:

Syntax

```
<txp:rsd />
```

The **rsd** tag is a *single* tag which is used to insert a Really Simple Discoverability link element, helping XML-RPC client programs to configure themselves.

Genealogy

Version 4.6.0

Tag support removed.

Version 4.0.7

Tag support added.

Search input

On this page:

Syntax

```
<txp:search_input />
```

The **search_input** tag is a *single* tag. This tag will provide a text entry field for search parameters and an optional button to initiate the search.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `button="text"`
  Creates and labels a button to initiate the search.
  Default: unset (no button is created).
- `form="form name"`
  Use specified form template to build a customized HTML form.
  Default: `search_input`.
- `html_id="id"`
  The HTML `id` attribute assigned to the search form.
  Default: unset.
- `match="match type"`
  Set the search mode. Choose from:
  `exact`: search terms must exactly match the words in the article in the order given. This mode is also automatically selected if the search term is surrounded with double quotes.
  `any`: any of the search terms in an article will cause it to show up in the results.
  `all`: all of the search terms in the article must exist (in any order) for the article to be included in the results.
  Default `exact`.
- `section="section name"`
  Use the specified section as the destination page that will display the search results.
  Default: unset (use the front page).
- `size="integer"`
  Sets the `size` attribute of the search `input` field.
  Default: `15`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `wraptag="element"`
  HTML element to wrap markup, specified without brackets (e.g. `wraptag="div"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display a search input form

```
<txp:search_input label="Search" button="Search" size="20" wraptag="div" />
```

Example 2: Elements required for building a customized HTML search form

You can build your own custom search form by specifying `form="form-name"` inside the `<txp:search_input />` tag:

```
<txp:search_input form="form-name" />
```

You would then need to build your Form (i.e. `form-name`), and the absolute minimum Textpattern tags and attributes required would be:

```
<form action="<txp:site_url />">
    <input name="q" type="search" />
</form>
```

When using a customized form template, Textpattern doesn't automatically wrap the HTML form output with `<form>` tags, thus you need the opening and closing `<form>` tag pair. The `name="q"` attribute and value is **required** to initiate a search query.

Other tags used: site_url.

Textpattern will use a user defined form named `search_results`, or an internally defined default form if no search result form is defined by you.

Genealogy

Version 4.3.0

`match` attribute added.

Version 4.0.7

`html_id` attribute added.

Search result count

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:search_result_count />
```

The **search_result_count** tag is a *single* tag that returns the number of articles returned by an article tag. Use if_search to count search results or use in regular page after the article tag. If you need the results' count *before* the list of results, use the article tag in conjunction with `pgonly="1"` (see example 3).

Note: The if_search conditional tag is required to recognize actual search results, without them the number of articles is returned by default.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `text="text"`
  Text to display with the number of matches.
  Default: `articles found`.

Examples

Example 1: Display a number of matches

```
<txp:if_search>
    <txp:article />
    <txp:search_result_count />
</txp:if_search>
```

If the visitor is searching for articles, it will show the number of articles that matched the search term (e.g. 5) as follows: `5 articles found`.

Other tags used: article, if_search.

Example 2: Number of matches with custom text

```
<txp:if_search>
    <txp:search_result_count text="hits" />
</txp:if_search />
```

Displays the number of articles returned (e.g. 5) as follows: `5 hits`.

Other tags used: if_search.

Example 3: Search result count above results

```
<txp:if_search>
    <txp:article pgonly="1" limit="20" />
    <txp:search_result_count text="hits" />
    <txp:article limit="20" />
</txp:if_search />
```

Displays the number of articles returned (e.g. 5) as follows: `5 hits` – then displays results as an article listing.

Note: The `pgonly` attribute sets the article tag to return pagination statistics without rendering the article list. Care must be taken to remain consistent with article tag attributes to keep statistics accurate.

Other tags used: article, if_search.

Search result date

On this page:

Syntax

```
<txp:search_result_date />
```

The **search_result_date** tag is a *single* tag. This tag will provide the article posted date as returned by the search function.

Attributes

- `format="format string"`
  Override the default date format set in the Preferences administration panel.
  Values: any valid strftime string values, `since`, `iso8601` (ISO 8601 reference), `w3cdtf` (W3CDTF reference), or `rfc822` (RFC 822 reference).
  Default: the ‘Archive date format’ set in preferences.

Examples

Example 1: Displays the posting date of an article

```
<h3>
    <txp:permlink>
        <txp:title />
    </txp:permlink>
</h3>
<p>
    <txp:search_result_date /> | <txp:posted />
</p>
```

Used in a search results form, this offers a search result entry comprising a hyperlinked article title, and the date that article was posted.

Other tags used: title, permlink, posted.

Search result excerpt

On this page:

Syntax

```
<txp:search_result_excerpt />
```

The **search_result_excerpt** tag is a *single* tag. The tag will show the occurrence of the search term with some surrounding context.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `break="tag"`
  Trailing string.
  Default: … (ellipsis).
- `hilight="tag"`
  HTML tag to be used for search term matches in excerpt text, without brackets.
  Default: `strong`.
- `limit="integer"`
  Maximum number of search match excerpts per search result.
  Default: `5`.

Examples

Example 1: Display up to 15 search excerpts with a search results form

```
<h3>
    <txp:permlink>
        <txp:title />
    </txp:permlink>
</h3>
<p>
    <txp:search_result_excerpt hilight="p" limit="15" /> |
    <txp:permlink>
        <txp:permlink />
    </txp:permlink>
    | <txp:posted />
</p>
```

Other tags used: permlink, posted, title.

Genealogy

Version 4.0.6

`break` attribute added.

Search result title

On this page:

Syntax

```
<txp:search_result_title />
```

The **search_result_title** tag is a *single* tag. This tag will provide a hyperlinked title to an article as returned by the search function.

Attributes

This tag has no attributes.

Examples

Example 1: Display a hyperlinked title to an article

```
<h3>
    <txp:search_result_title />
    <txp:search_result_date />
</h3>
```

In a search results form, this shows the title of an article that matched the visitor’s search results, and its posted date.

Other tags used: search_result_date.

Search result url

On this page:

Syntax

```
<txp:search_result_url />
```

The **search_result_url** tag is a *single* tag. This tag will provide a hyperlinked URL to an article as returned by the search function.

Attributes

This tag has no attributes.

Examples

Example 1: Display a hyperlinked URL to an article

```
<txp:search_result_url />
```

Used within a search results form to allow visitors to click on the link and be taken to the article that matched their search results.

Search term

On this page:

Syntax

```
<txp:search_term />
```

The **search_term** tag is a *single* tag which returns the expression the user searched for through the full text search form.

Attributes

This tag has no attributes.

Examples

Example 1: Display the search term on the search results page

```
<txp:if_search>
    <h1>Search results</h1>
    <h3>
        You searched for:
        <txp:search_term />
    </h3>
    <txp:article />
</txp:if_search>
```

Genealogy

Version 4.5.0

`escape` attribute deprecated.

Version 4.0.6

Tag support added.

Section list

On this page:

Syntax

```
<txp:section_list />
```

The **section_list** tag is a *single* or *container* tag which is used to produce a list of linked sections. When used as a container tag, it is used as an opening and closing pair, like this:

```
<txp:section_list>
    ...contained statements...
</txp:section_list>
```

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `active_class="class name"` (only works in the *single* tag without the `form` attribute)
  HTML `class` to apply to the ‘active’ or current link in a list.
  Default: unset
- `default_title="text"`
  Text used as a title for the ‘default’ section when `include_default` is set to `1`.
  Default: site name.
- `exclude="section name(s)"`
  Comma-separated list of section names to exclude from the list. `sections` takes precedence over `exclude`.
  Default: unset (none).
- `form="form name"`
  Use specified form template to process each included section.
  Default: unset.
- `html_id="id"`
  The HTML `id` attribute applied to the `wraptag`, if set.
- `include_default="boolean" Whether to include 'default' section in section list. Values: @0`
  (no) or `1` (yes).
  Default: `0`.
- `limit="integer"`
  The number of articles to display.
  Default: `0` (no limit).
- `offset="integer"`
  The number of articles to skip.
  Default: `0`.
- `sections="section name(s)"`
  Comma-separated list of section names to include in the list, displayed in specified order (unless overridden by the `sort` attribute).
  Default: unset (all sections).
- `sort="sort value(s)"`
  How to sort resulting list.
  Values:
  `css`.
  `in_rss`.
  `is_default`.
  `name`.
  `on_frontpage`.
  `page`.
  `rand()` (random).
  `searchable`.
  `Title`.
  Each field in the `textpattern` database table can be used as a sort key.
  When viewing a search results list, `score` (how well the search terms match the article) is available as an additional

value.
Default: `name asc`.

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `break="value"`
  Where value is an HTML element, specified without brackets (e.g. `break="li"`) or some string to separate list items.
  Default: `br` (but see [[break cross-reference]] for exceptions).
- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value.
  Default: tag name or unset (see [[class cross-reference]]).
- `label="text"`
  Label prepended to item.
  Default: unset (but see [[label cross-reference]] for exceptions).
- `labeltag="element"`
  HTML element to wrap (markup) label, specified without brackets (e.g. labeltag="h3").
  Default: unset.
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display a linked section list

```
<txp:section_list label="Sections" wraptag="p" break="br" />
```

Adds the label ‘Sections’ and wraps the output in a paragraph with each section on its own line.

Example 2: Display a styled section list

```
<txp:section_list wraptag="ul" break="li" />
```

Example 3: Set active class using the container tag

```
<txp:section_list wraptag="ul" break="">
    <li<txp:if_section name='<txp:section />'> class="active"></txp:if_section>>
        <txp:section title="1" link="1" />
    </li>
</txp:section_list>
```

This code will add `class="active"` to the `<li>` element around the currently viewed section in the list.

Other tags used: if_section, section.

Genealogy

Version 4.6.0

`html_id`, `limit` and `offset` attributes added.

Version 4.0.7

Can be used as a container tag.
`form` attribute added.

Section

On this page:

Syntax

`<txp:section>`

The **section** tag can be used as either a *single* tag or *container* tag. It will display information about the section as defined by either the `name` attribute, the section currently being viewed, or the section of the article being displayed (if used within a Textpattern 'article' type [[Form template]], or an if_individual_article conditional tag).

When used as a containing tag, it will turn the contents into a link to that section. Otherwise, it will return plain text.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `link="boolean"` (works only in the *single* tag)
  Display as plain text or a link.
  Values: `0` (plain text) or `1` (link).
  Default: `0`.
- `name="section name"`
  Display the named section.
  Default: unset (display the current section).
- `title="boolean"`
  Display either the section name or its title.
  Values: `0` (name) or `1` (title).
  Default: `0`.
- `url="boolean"`
  Display plain URL or full link.
  Values: `0` or `1`.
  Default: `0` (display title or full link, depending on `link`).

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value. If no wraptag is supplied (and `link="1"`), the class is applied to the anchor instead.
  Default: unset (see [[class cross-reference]]).
- `wraptag="element"`
  HTML element to wrap (markup) list block, specified without brackets (e.g. `wraptag="ul"`).
  Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display the current section name

```
<txp:section />
```

Example 2: Display hyperlinked section title

```
<txp:section link="1" title="1" />
```

In an article form, it displays the article's section title as a hyperlink to the section home page. Otherwise, it displays the title of the section currently being viewed as a hyperlink to the section home page.

Example 3: Display a link to a specified section

```
<txp:section link="1" title="1" wraptag="p" name="archive" />
```

Displays a hyperlink to the ‘archive’ section home page, wrapped in `<p>` tags, using the section’s title as link text.

Example 4: Container tag example

```
<txp:section name="archive">
    My Archive
</txp:section>
```

Displays the text ‘My Archive’ as a hyperlink to the ‘archive’ section home page. HTML output for clean URLs:

```
<a href="http://example.com/archive/">My Archive</a>
```

And for messy URLs:

```
<a href="http://example.com/index.php?s=archive">My Archive</a>
```

Example 5: Single tag example

```
<a href="<txp:section name="about" url="1" />">
    <txp:section name="about" title="1" />
</a>
```

Displays the section title ‘About’ as a hyperlink to the ‘about’ section home page. HTML output for clean URLs:

```
<a href="http://example.com/about/">About</a>
```

And for messy URLs:

```
<a href="http://example.com/index.php?s=about">About</a>
```

Genealogy

Version 4.0.7

Applies `class` attribute to the `<a>` element when `wraptag` is empty.
`url` attribute added.

Site name

On this page:

Syntax

```
<txp:site_name />
```

The **site_name** tag is a *single* tag that returns the site’s name as defined within the Preferences administration panel.

Attributes

This tag has no attributes

Examples

Example 1: Display the site’s name

```
<h1>
    <txp:site_name />
</h1>
```

Site slogan

On this page:

Syntax

```
<txp:site_slogan />
```

The **site_slogan** is a *single* tag which is used to output the site's tagline (labeled as 'Site slogan' in the Preferences administration panel).

The slogan is a brief (255 characters maximum) tagline or description of your site which can be used, for example, in XML feeds.

Attributes

This tag has no attributes.

Examples

Example 1: General display of slogan

```
<h2>
    <txp:site_slogan />
</h2>
```

Example 2: Slogan as content filler

The slogan could be used for the content attribute of the `description` metadata element. Either whole…

```
<meta name="description" content="<txp:site_slogan />">
```

or partial…

```
<meta name="description" content="<txp:site_slogan />. And the rest of your pithy description would go here.">
```

Site url

On this page:

- Syntax
- Attributes
- Examples

Syntax

```
<txp:site_url />
```

The **site_url** tag is a *single* tag which returns the full URL of the site (as defined in the Preferences administration panel) as text. If you maintain local development versions of your live sites and import databases between them, then this tag is extremely valuable for ensuring your domain links are never confused (and thus broken) between the two locations (see example 1, below).

Attributes

This tag has no attributes.

Examples

Example 1: Maintain accurate domain paths

```
<nav>
    <ul>
        <li>
            <a href="<txp:site_url />articles">Articles</a>
        </li>
        <li>
            <a href="<txp:site_url />photos">Photographs</a>
        </li>
    </ul>
</nav>
```

A classic example is with navigation links – the idea is that you don't break URL paths after importing a database from local development to live, or vice versa. By using this tag it will automatically be relative to a given site and you'll never have to manually edit broken domain paths again.

Example 2: HTML header paths

```
<link rel="stylesheet" href="<txp:site_url />assets/css/main.css">
```

In the `<head>` section of your HTML pages you might have a variety of links to locations relative to the local server, such as CSS files, JavaScript files and so forth. The relevance is similar to example 1 – you want to ensure the paths are accurate relative to the server if a database has been imported from another location.

Example 3: Display a hyperlink to download a .ZIP file

```
<a href="<txp:site_url />download.zip">Download</a>
```

Tag attributes cross-reference [todo:pw]

This cross-reference makes clear the different Textpattern tags associated with a given attribute. For each attribute, the associated tags are listed underneath it.

Authors: DO NOT add new tags/attributes here from development versions of Textpattern, add them to tags_in_development to avoid confusion. Only the latest stable version of Textpattern is officially supported, and this list should only reflect the latest stable version.

active_class

HTML class attribute to be applied to the “active” or current link in a list.

- category_list
- section_list

align

As of version 4.2.0 this attribute is deprecated.

HTML img tag align attribute. Recommended that you use CSS via **class** or **id** attribute instead.

- article_image
- image
- thumbnail

allowoverride

Used to disable assigned override forms. Works within article lists and single article display. Both article tags use

```
1
```

(yes) as default.

- article
- article_custom

anchor

- comment_permlink

author

- article_custom
- images
- linklist
- file_download_list

auto_detect

- images
- linklist
- file_download_list

break

HTML tag (without brackets) or string used to separate list items. Suggested values include

```
br
```

and

```
hr
```

for presentational markup, or

```
li
```

if semantic markup is preferred. Textpattern cares for the correct nesting of tags in either case.

Default is

```
br
```

(except for tags marked by an asterisk, which default to unset).

- article *
- article_custom *
- category_list
- image_index
- comments
- comments_error
- file_download_list
- images
- linklist *
- recent_articles
- recent_comments
- related_articles
- search_result_excerpt
- section_list

breakclass

HTML class attribute to be applied to **break** (when value supplied is a tag).

- comments

button

- search_input

category

- article_custom
- feed_link
- image_index
- link_feed_link
- linklist
- recent_articles
- file_download_list

categories

- category_list

children

- category_list

class

HTML class attribute to be applied to the specified wraptag. Tags marked with an asterisk use the tag name as the class name by default. Otherwise the default is empty.

- article
- article_custom
- article_image
- breadcrumb
- category
- category1
- category2
- category_list *
- comments *
- comments_error *
- comments_invite *
- comments_form *
- comments_preview *
- expires

- image
- image_index
- feed_link
- file_download_category
- file_download_description
- file_download_list *
- images *
- link_category
- link_description
- link_feed_link *
- link_to_home
- linklist *
- modified
- permlink
- popup
- posted
- recent_articles *
- related_articles *
- recent_comments *
- search_input *
- section
- section_list *
- thumbnail

//customfieldname//

In your tag, replace "_customfieldname_" with the actual name of the desired custom field (see also ""Important notes on creating custom field names":/home/www/zendstudio/dokuwiki/bin/doku.php?id=advanced_preferences#custom_fields").

- article
- article_custom

decimals

- file_download_size

default

- custom_field

default_title

- section_list

email

- email

escape

Escape HTML entities.

- article_image
- author_email
- custom_field
- file_download_description
- image
- image_info
- link_description
- link_name
- newer
- older
- thumbnail

excerpted

- article_custom

exclude

- category_list
- section_list

expired

- article_custom

extension

- images

filename

- file_download
- file_download_link
- file_download_name

flavor

- feed_link
- link_feed_link

forgetlabel

- comments_form

form

Used to format content for display. See forms_explained.

- article
- article_custom
- category_list
- comments
- comments_form
- comments_preview
- file_download
- file_download_list
- linklist
- output_form
- recent_comments
- related_articles
- search_input
- section_list

format

- comment_time
- css
- expires
- feed_link
- file_download_created
- file_download_modified
- file_download_size
- link_date
- link_feed_link
- modified
- posted

frontpage

- article_custom

gmt

- comment_time

- expires
- link_date
- modified
- posted

height

- article_image
- image

hilight

- search_result_excerpt

html_id

HTML id attribute to be applied to the **wraptag**.

- article_image
- image
- search_input
- thumbnail

id

- article_custom
- file_download
- file_download_link
- file_download_list
- if_article_id
- image
- images
- permlink
- thumbnail

include_default

- section_list

item

- text

isize

HTML size attribute to be applied to HTML form input output.

- comments_form

keywords

- article
- article_custom
- if_keywords

label

This string will be prepended to the output. When using a wraptag value of either

`ol`

or

`ul`

, the label will be the first list item.

Default is unset (except where shown in parentheses).

- article
- article_custom
- breadcrumb (*Site name*)
- category_list
- feed_link
- file_download_list
- image_index
- images
- link_category
- link_description
- link_feed_link
- linklist
- popup (

  ```
  Browse
  ```

)
- recent_articles (

  ```
  Recent Articles
  ```

)
- recent_comments
- related_articles
- search_input (

  ```
  Search
  ```

)
- section_list

labeltag

HTML tag (without brackets) to wrap around label. Default is unset.

- article
- article_custom
- category_list
- file_download_list
- image_index
- images
- link_category
- link_description
- linklist
- recent_articles
- recent_comments
- related_articles
- section_list

lang

- comment_time
- expires
- link_date
- modified
- posted

limit

- article
- article_custom
- comments
- feed_link
- file_download_list
- image_index

- linklist
- recent_articles
- recent_comments
- related_articles
- search_result_excerpt

link

- author
- breadcrumb
- category
- category1
- category2
- comment_name
- section
- thumbnail

linkclass

HTML class attribute to be applied to links.

- breadcrumb

linktext

- email

listform

See forms_explained.

- article

login

- password_protect

match

- related_articles
- search_input

media

- css

msgcols

HTML cols attribute to be applied to HTML form textarea output.

- comments_form

msgrows

HTML rows attribute to be applied to HTML form textarea output.

- comments_form

msgstyle

HTML style attribute to be applied to HTML form textarea output. Recommended that you use CSS via textarea's class or id attribute instead.

- comments_form

month

- article_custom

no_widow

- recent_articles
- related_articles
- title

name

- category
- css
- custom_field
- if_article_author
- if_article_category
- if_article_section
- if_author
- if_custom_field
- if_category
- if_plugin
- if_section
- if_variable
- image
- thumbnail
- section
- variable

number

- if_article_category

offset

- article
- article_custom
- comments
- file_download_list
- image_index
- linklist
- recent_comments

pageby

- article
- file_download_list
- images
- linklist

parent

- category_list

pass

- password_protect

pgonly

Used to return pagination statistics without rendering the article list. Care must be taken to remain consistent with article tag attributes to keep statistics accurate.

- article
- article_custom

poplink

- thumbnail

previewlabel

- comments_form

realname

- images
- linklist
- file_download_list

rel

HTML rel attribute to be applied to links.

- css
- link
- linkdesctitle

rememberlabel

- comments_form

searchall

- article

searchform

See forms_explained.

- article

searchsticky

- article

section

- article_custom
- author
- category
- category1
- category2
- category_list
- feed_link
- popup
- related_articles
- recent_articles
- search_input

sections

- section_list

separator

- breadcrumb
- page_title

showalways

- comments_invite
- link_to_next
- link_to_prev
- newer
- older

showcount

- comments_invite

show_preview

- comments_form

size

HTML size attribute to be applied to HTML form input output.

- search_input

sort

How to sort the resulting list. Available and default values vary by tag, but multiple, comma-separated values can be used. Each value can be specified as either

```
asc
```

(ascending) or

```
desc
```

(descending) order.

- article
- article_custom
- category_list
- comments
- file_download_list
- image_index
- linklist
- recent_articles
- recent_comments
- related_articles
- section_list

status

- article
- article_custom
- file_download_list
- if_status

style

- article_image
- image
- permlink
- thumbnail

submitlabel

- comments_form

text

- search_result_count

textonly

- comments_invite

this_section

- author
- category_list
- category
- category1
- category2
- popup

thumbnail

- article_image

time

- article
- article_custom

title

- author
- breadcrumb
- category
- category1
- category2
- css
- email
- feed_link
- file_download_category
- file_download_author
- file_download_name
- image_author
- link_author
- link_category
- link_feed_link
- newer
- older
- permlink
- section

type

- category
- category_list
- if_author
- if_category
- page_url
- popup

url

- txp_die

value

- if_custom_field
- if_variable
- variable

version

- if_plugin

width

- article_image
- image

wraptag

HTML tag (without brackets) to wrap around output. Default value is unset, except where shown in parentheses.

- article
- article_custom
- article_image

- breadcrumb (

  `p`

)

- category
- category1
- category2
- category_list
- comments

- comments_error (

  `div`

)

- comments_form
- comments_invite
- comments_preview
- expires
- feed_link
- image
- image_index
- images
- file_download_category
- file_download_description
- file_download_list
- link_category
- link_description
- link_feed_link
- linklist
- modified
- popup
- posted
- recent_articles
- recent_comments
- related_articles

- search_input (

  `p`

)

- section
- section_list
- thumbnail

Tags in development [todo:pw]

This page represents tags and attributes added in development versions of Textpattern since the the latest, stable release (4.6.0), whether completely new tags or existing tags that are undergoing modifications.

Because we are talking about development, the information here is subject to change.

If you wish to write the documentation for new tags in preparation for the next release, please restrict them to the ‘Future tags’ category only until the development version becomes stable whereby they can be recategorized in the correct location(s). At that point, remember to also check any example syntax in modified tags to ensure they still follow recommended use and will still work as before.

New tags

- authors
- if_first_file
- if_first_image
- if_first_link
- if_last_file
- if_last_image
- if_last_link
- if_yield
- meta_description

Text

On this page:

Syntax

```
<txp:text />
```

The **text** tag is a *single* tag which is primarily used to return localized language strings from the `txp_lang` database table.

Note: only language strings designated with a type of `common` or `public` are available to use. All other language string types are reserved for use within the Textpattern system itself.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&` within text.
  Values: `html` or unset.
  Default: `html`.
- `item="text"`
  Piece of text to display, preferably an item from the `name` column of the `txp_lang` database table. If the given item matches a key there, the contents of the respective item in the `data` column will be returned. Otherwise, whatever you supply as the `item` value is returned verbatim.

Examples

Example 1: Display some localized text

```
<txp:older>
    <txp:text item="older" />
</txp:older>
```

Outputs the text ‘older’ inside the `<txp:older />` tag, respecting the current Textpattern language, instead of using the tag like this: `<txp:older>older</txp:older>` which would always render the English text ‘older’, it replaces the contents with the value assigned to the name ‘older’ in the current language. So you would see a link with the word ‘älter’ if you were using German `de` as the Textpattern site language.

Other tags used: older.

Genealogy

Version 4.6.0

Accepts L10n replacement tags as attributes.
`escape` attribute added.

Thumbnail

On this page:

Syntax

`<txp:thumbnail />`

The **thumbnail** tag is a *single* tag that Textpattern will replace with the `<img src="...">` HTML tag matching the thumbnail image of the numeric `id` assigned by Textpattern when the parent image was uploaded via the Textpattern Images administration panel.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `escape="html"`
  Escape HTML entities such as `<`, `>` and `&` for the image's `alt` and `title` attributes.
  Values: `html` or unset.
  Default: `html`.
- `height="integer"`
  Specify an image `height` which overrides the value stored in the database. Use `height="0"` to turn off the output of a width attribute in the `<img>` tag (thus the browser will scale the height if a width is used).
- `html_id="id"`
  The HTML `id` attribute assigned to the image (or to the `wraptag`, if set).
  Default: unset.
- `id="integer"`
  Specifies the `id` assigned at upload of the image to display. Can be found on the Textpattern Images administration panel. If both `name` and `id` are specified, `name` is used while `id` is ignored.
- `link="boolean"`
  If set, the thumbnail will be rendered as a (non-Javascript) URL link to the full-size image.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `link_rel="relation"`
  Value for the HTML `rel` attribute.
  Default: unset.
- `name="image name"`
  Specifies which image thumbnail to display by its image name as shown on the Textpattern Images administration panel.
- `poplink="boolean"`
  If set, the image will be rendered in a popup window.
  Values: `0` (no) or `1` (yes).
  Default: `0`.
- `width="integer"`
  Specify an image `width` which overrides the value stored in the database. Use `width="0"` to turn off the output of a width attribute in the `<img>` tag (thus the browser will scale the width if a height is used).

Common presentational attributes

These attributes, which affect presentation, are shared by many tags. Note that default values can vary among tags.

- `class="class name"`
  HTML `class` to apply to the `wraptag` attribute value, if set, otherwise to the `<img>` tag.
  Default: unset (see [[class cross-reference]]).
- `style="style rule"`
  Inline CSS `style` rule. It's recommended that you assign CSS rules via `class` attribute instead.
  Default: unset.
- `wraptag="element"`
  HTML tag to be used to wrap the `<img>` tag, specified without brackets (e.g. `wraptag="p"`).

Default: unset (but see [[wraptag cross-reference]] for exceptions).

Examples

Example 1: Display the given thumbnail

```
<txp:thumbnail id="23" />
```

Displays the image thumbnail for the image uploaded as ID #23.

Genealogy

Version 4.2.0

`align` attribute deprecated.

Version 4.0.7

Default value for `escape` attribute changed from unset to `html`.

Version 4.0.6

`link` and `link_rel` attributes added.

Version 4.0.4

`html_id`, `escape` and `wraptag` attributes added.

Title

On this page:

Syntax

```
<txp:title />
```

The **title** tag is a *single* tag which is used to return the title of the article being displayed. It is usually used in an ‘article’ type form.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `no_widow="boolean"`
  Control widows and overrule ‘widows’ setting in the Preferences administration panel.
  Values: `0` allow the last word in the title to appear on its own line, i.e. the title content is rendered unchanged, `1` ensure the last word is not left on its own line – Textpattern inserts an invisible code (a non-breaking space) between the last two words.
  Default: as set in the Preferences administration panel.

Examples

Example 1: Display an article title

```
<h1>
    <txp:title />
</h1>
<p>
    <txp:author /> at <txp:posted />
</p>
<txp:body />
```

Shows the current article title as the page heading, a few other pieces of information such as the article’s author and posted date, then the article body itself.

Other tags used: author, posted, body.

Example 2: Display a hyperlinked title

```
<txp:permlink>
    <txp:title />
</txp:permlink>
```

Wraps a permanent link to the current article around its title.

Other tags used: permlink.

Txp die

On this page:

Syntax

```
<txp:txp_die />
```

The **txp_die** tag is a *single* tag that will terminate normal page rendition and return the given status to the user agent (browser, search engine crawler, feed aggregator). An error page will also be returned to the user agent.

The status can be displayed by the error_status tag. A textual message can be associated with the error status and retrieved with the error_message tag. See also: [[Custom Error Pages]].

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `msg="message"`
  Textual representation of the error condition.
- `status="number"`
  Numerical representation of the error condition.
  Typical values: `301`, `302`, `304`, `307`, `401`, `403`, `404`, `408`, `410`, `503`, or any other valid status code.
  Default: `503`.
- `url="url"`
  Redirects to the specified URL. Can be used with redirection statuses `301`, `302` and `307`.
  Default: unset.

Examples

Example 1: Force a 404 ‘not found’ error

```
<txp:txp_die status="404" />
```

Example 2: Issue a permanent redirect

```
<txp:txp_die status="301" url="http://example.com/new/location" />
```

Genealogy

Version 4.5.0

`url` attribute added.

Variable

On this page:

Syntax

```
<txp:variable />
```

The **variable** tag is both a *single* and a *container* tag which sets or returns a user-defined global variable.

If used as a *container* tag, the result of the contained statements are assigned to the given variable `name`, like this:

```
<txp:variable>
    ...contained statements...
</txp:variable>
```

Note: Avoid entering white space characters for better code readability between the opening and closing *variable* tags, they will lead to falsified results in the if_variable evaluation.

Attributes

Tag will accept the following attributes (**case-sensitive**):

- `name="text"`
  The variable name for which you wish to assign a value. Valid variable names must not contain any single or double quotes.
- `value="value"`
  (Optionally) define the value to which you wish to set the variable. Without this attribute, the tag returns the current value assigned to the named variable.

Examples

Example 1: Store site-wide constants

Allows you to define constants at a single location (e.g. a [[Form]] template, or even at the top of a [[Page]] template) and use them elsewhere later on.

Somewhere at the very beginning of a template you would define names and values, just like you do on your desktop calculatorâ€™s “memory” keys:

```
<txp:if_search>
    <title>My blog search results: <txp:search_term /></title>
    <meta name="description" content="Blog article search results.">
    <meta name="robots" content="none">
<txp:else />
    <txp:if_category>
        <title>Blog category: <txp:category title="1" /></title>
        <meta name="description" content="Blog article ‘<txp:category title="1" />’ category archive.">
        <meta name="robots" content="noindex, follow, noodp, noydir">
    <txp:else />
        <title>My blog homepage</title>
        <meta name="description" content="The great homepage of my great blog.">
        <meta name="robots" content="index, follow, noodp, noydir">
        <txp:variable name="homepage" value="1" />
    </txp:if_category>
</txp:if_search>
```

Later down the [[Page]] template or in a separate [[Form]] you can read the attribute values previously set conditionals come in handy at times:

```
<txp:if_variable name="homepage" value="1">
    ...homepage content...
</txp:if_variable>
```

Other tags used: else, if_category, if_search, if_variable, search_term.

Example 2: Use any tag’s value as a conditional expression

There are two parts to making this work. First a variable is created that stores the output of any tag as the `value` (the `name` is arbitrary)…

```
<txp:variable name="foo" value='<txp:permlink />' />
```

Note: A Textpattern tag, used as an attribute (a parsed attribute), must be surrounded with single quotes.

The variable ‘foo’ can then be used as a conditional later in the code.

```
<txp:if_variable name="foo" value="example.com/bar/baz">
    ...do this...
</txp:if_variable>
```

The conditional is saying if there is a variable named ‘foo’ having a specific value of ‘example.com/bar/baz’, then output what is defined, i.e. ‘do this’.

Other tags used: if_variable.

Genealogy

Version 4.0.7

Tag support added.

Yield

On this page:

Syntax

`<txp:yield />`

The **yield** tag is a *single* tag which is used to return the inner content of the enclosing `<txp:output_form />` tag.

The tag works in unison with the output_form tag; You place a `<txp:yield />` tag in a form and then wherever you use that form the content wrapped inside the **output_form** tag gets placed where the **yield** tag was. Essentially, the value of this is that you can use forms as flexible building blocks – each building block has a common structure, but individualized content.

Attributes

This tag has no attributes.

Examples

Example: Inner content

Given the following form named `example_form`:

```
<div>
    This content is static and will be the same every time this form is invoked.
    <txp:yield />
</div>
```

We can invoke it twice with different inner content each time:

```
<txp:output_form form="example_form">
    Invoking 'example_form' with some inner content.
</txp:output_form>
<txp:output_form form="example_form">
    Invoking 'example_form' again, this time with different inner content.
</txp:output_form>
```

And the result will be:

```
<div>
    This content is static and will be the same every time this form is invoked.
    Invoking 'example_form' with some inner content.
</div>
<div>
    This content is static and will be the same every time this form is invoked.
    Invoking 'example_form' again, this time with different inner content.
</div>
```

Other tags used: output_form.

Genealogy

Version 4.2.0

Tag support added.