

Tag Types

There are 3 different types of tags; Basic Tags, Hypertext tags, and Date tags. The latter two types can be optionally configured with attributes.

Basic Tags

Basic tags take no attributes, and are simply replaced with plain text weblog data.

Date Tags

Date Tags are replaced with date related data and can take the following optional attributes:

- **format** - Specifies how the date/time should be formatted.
- **lang** - Specifies the date's language locale. For instance, "en" for April or "fr" for Avril.
- **country** - Specifies the date's country locale (US, FR, DE, etc)

Example: <\$CurrentDate\$ format="MM/dd/yy"\$>

The tag above will be replaced with the current date formatted to display the month, day, and year. E.g 05/15/04.

The format attribute can take any valid Java date format pattern.

In addition you can specify the value "RFC822" for the format attribute. This is useful if you need to, for instance, supply RFC822 date formats for the RSS feed. If the value RFC822 is given, the the locale attributes, if specified, are ignored.

The following examples show how date and time format patterns are interpreted in the U.S. locale. The given date and time are 2001-07-04 12:08:56 local time in the U.S. Pacific time zone.

Date and Time Pattern	Result
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 AD at 12:08:56 PDT
"EEE, MMM d, 'yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o'clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700

The default, no attribute, format is: dd/MM/yy h:mm

Hypertext Tags

Hypertext Tags are replaced with data that likely contains HTML. Hypertext Tags can take the following attributes...

- `encode_html` – Converts html tag parts so they are printable. E.g the character ‘<’ becomes ‘<’
- `strip_html` – If the value is not 0, this attribute removes html from the chunk of text the tag represents.
- `limit_length` – If the value is not 0, this attribute limits the chunk of text that the tag represents to a specified amount of words.
- `words` – The number of words to limit to. This works in conjunction with the attribute above.

Let’s say we didn’t want any HTML in the weblog’s description, and that we want to limit the maximum number of words in the description to 10. We’d use the following tag and attributes...

```
<${BlogDescription strip_html="1" limit_lenth="1" words="10"}>
```

A default, no attribute, hypertext tag does not encode HTML, does not strip html, and does not limit the length.

Global Tags

Tags that work everywhere

The tags below can be used anywhere in any of the templates

Tag	Replaced with:	Type
<\$BlogTitle\$>	The title of the weblog	Basic
<\$BlogDescription\$>	The description of the weblog	Hypertext
<\$PageTitle\$>	The title of the page – i.e. Category, archive date, weblog title	Basic
<\$FrontPageLink\$>	The URL of the front page	Basic
<\$IndexPageLink\$>	The URL of the archive index page	Basic
<\$RssLink\$>	The URL of the syndication feed	Basic
<\$Charset\$>	The Charset of the weblog	Basic
<\$AppName\$>	The name of the generator. i.e. Thingamablog	Basic
<\$AppVersion\$>	The version of Thingamablog	Basic
<\$AppLink\$>	The URL to the Thingamablog website	Basic
<\$CurrentDate\$>	The current date/time	Date
<\$BaseURL\$>	The top most URL of the weblog	Basic
<\$Lang\$>	The language code of the blog's locale. (en, es, de, etc)	Basic
<\$Country\$>	The country code of the blog's locale. (US, FR, DE, etc)	Basic

Archive Lists

The `<ArchiveList>` container is used to generate a list of the weblog's archive pages. The `ArchiveList` container takes the form...

```
<ArchiveList>
<$ArchiveName$>
<$ArchiveLink$>
</ArchiveList>
```

This container works like a loop in that the inner value of the container is printed for each archive page.

The `ArchiveList` container can take the following attributes...

- **sort_order** – Sorts the list either ascending or descending depending on the attribute value (ascend or descend)
- **glue** – specifies the "glue" that separates each archive page.
- **format** – specifies the date format of the archive. This works the same as the format attribute of a Date Tag
- **span** – specifies whether or not to display the start AND end dates of the archive. 1 = both dates, 0 = start date only.

The `ArchiveList` container has two inner tags. These tags only have meaning between `<ArchiveList>` and `</ArchiveList>`

`<$ArchiveName$>` - Replaced with the name of the archive (e.g June 2004)

`<$ArchiveLink$>` - Replaced with the URL of the archive page.

As an example, let's say you wanted to generate a comma-separated list of links to the archive pages. You'd use the following code...

```
<ArchiveList sort_order="ascend" glue="," format="dd/mm/yy" span="1">
<a href="<$ArchiveLink$>"><$ArchiveName$></a>
</ArchiveList>
```

`<ArchiveYears>`

The `<ArchiveYears>` container generates a list of years and archive pages. The syntax is:

```
<ArchiveYears sort_order="descend">
  <$Year$>
  <ArchiveYear>
    <$ArchiveLink$><$ArchiveName$>
  </ArchiveYear>
</ArchiveYears>
```

Example: generating a yearly list of weblog archives

```
<ul>
<ArchiveYears sort_order="descend">
<li><$Year$></li>
  <ul>
    <ArchiveYear>
      <li><a href="<$ArchiveLink$>"><$ArchiveName$></a></li>
    </ArchiveYear>
  </ul>
</ArchiveYears>
</ul>
```

Note: Archive lists have no meaning on the Feed Template

Category Lists

The <CategoryList> container is used to generate a list of the weblog's category pages. The CategoryList container takes the form...

```
<CategoryList>
<${CategoryName$}>
<${CategoryLink$}>
</CategoryList>
```

This container works like a loop in that the inner value of the container is printed for each category page.

The CategoryList container can take the following attributes...

- **sort_order** – Sorts the list either ascending or descending depending on the attribute value (ascend or descend)
- **glue** – specifies the "glue" that separates each category page.

The CategoryList container has two inner tags.

- <\${CategoryLink\$}> – replaced with the URL of the category page
- <\${CategoryName\$}> – replaced with the name of the category

As an example, let's say you wanted to generate a comma-separated list of links to the category pages. You'd use the following code...

```
<CategoryList sort_order="ascend" glue=",">
<a href="<${CategoryLink$}>"><${CategoryName$}></a>
</CategoryList>
```

Note: CategoryLists have no meaning on the Feed Template

Calendar Tags

The <Calendar> container is useful for generating date-organized links to days with posts. The Calendar takes the form...

```
<Calendar>
<$MonthLabel$>
<WeekDays><$WeekDay$></WeekDays> <CalendarWeek>
<CalendarDay>
<IfCurrentDay> </IfCurrentDay>
<IfDayHasNoEntries><$DayOfMonth$><$DateOfDay$></IfDayHasNoEntries>
<IfDayHasEntries><$EntryArchivePage$><$EntryID$><$DayOfMonth$><$DateOfDay$>
</IfDayHasEntries>
<IfEmptySpace></IfEmptySpace>
</CalendarDay>
</CalendarWeek>
</Calendar>
```

The Calendar container is comprised of the following template elements...

- <\$MonthLabel\$> - Replaced with the name of the current month. e.g July
- <CalendarWeek> - Describes the layout of a week in the calendar.
- <WeekDays> - Describes the localized week days (mon, tue, etc) of the calendar

The CalendarWeek container is comprised of the following optional conditional tags...

- <IfCurrentDay> - The text inside this container is used if the day is equal to the current day
- <IfDayHasNoEntries> - The text inside this container is used if the day has no entries
- <IfDayHasEntries> - The text inside this container is used if the day has entries
- <IfEmptySpace> - The text inside this container is used if this is a blank space in the calendar

Tags which have meaning between the above "IfDay..." tags:

- <\$DayOfMonth\$> - The calendar day of the month (1 through 31)
- <\$DateOfDay\$> - The date of the calendar day. E.g (June 15th, 2004). This is a date tag and can take a format attribute to format the date.

Tags which have meaning inside the <IfDayHasEntries> container:

- <\$EntryArchivePage\$> - The URL of the archive page which contains the entries for the calendar day.
- <\$EntryID\$> - The unique ID of the first entry of the calendar day.

Tags that have meaning inside the <WeekDays> container:

- <\$WeekDay\$> - The day of the week

The <WeekDays> container can take the optional attribute "long" which, if true, prints the week days in long format (Sunday, Monday, Tuesday, etc)

Syntax:

```
<WeekDays><$WeekDay$></WeekDays>
```

Syntax (Long weekday names):

```
<WeekDays long="1"><$WeekDay$></WeekDays>
```

The Calendar container is flexible enough to generate various types of calendars. The following example demonstrates how to generate a typical calendar month using an HTML table.

```
<Calendar>
<table border="0" cellspacing="4" cellpadding="0">
<caption><$MonthLabel$></caption>
<tr>
<th align="center">Sun</th>
<th align="center">Mon</th>
<th align="center">Tue</th>
<th align="center">Wed</th>
<th align="center">Thu</th>
<th align="center">Fri</th>
<th align="center">Sat</th>
</tr>
<CalendarWeek>
<tr>
<CalendarDay>
<td align="center" <IfCurrentDay>bgcolor="silver"</IfCurrentDay>>
<IfDayHasNoEntries><$DayOfMonth$></IfDayHasNoEntries>
<IfDayHasEntries><a
href="<$EntryArchivePage$>#<$EntryID$>"><$DayOfMonth$></a></IfDayHasEntries
>
<IfEmptySpace>&nbsp;</IfEmptySpace></td>
</CalendarDay>
</tr>
</CalendarWeek>
</table>
</Calendar>
```

Note: The <Calendar> containers have no meaning on the Feed Template

Entry Tags

The <BlogEntry> container describes the layout of a single entry. Think of <BlogEntry>...</BlogEntry> as a sub-template within the main template for entries.

When the weblog is constructed, everything between these tags will be replaced with all of the entries for that page.

The <BlogEntry> container takes the following optional attributes...

- **sort_order** – Sorts the entries chronologically depending on the attribute value (ascend or descend)
- **limit** – Indicates whether or not the number of entries should be limited.
- **limit_by** – specifies the number of entries that should be on the page. Works in conjunction with the "limit" attribute above.

Tags that only work between <BlogEntry> and </BlogEntry>:

Tag	Replaced With:	Type
<\$EntryID\$>	The unique ID of the entry	Basic
<\$EntryDate\$>	The post date of the entry	Date
<\$EntryTime\$>	The post time of the entry	Date
<\$EntryDateTime\$>	The post date/time of the entry	Date
<\$EntryBody\$>	The body text of the entry	Hypertext
<\$EntryArchivePage\$>	The URL of the archive page of the entry	Basic
<\$EntryAuthor\$>	The name of the author of the entry	Basic
<\$EntryAuthorEmail\$>	The entry author's email address Can take a "mung" attribute which hides the address from spam robots. Example: <\$EntryAuthorEmail mung="1"\$>	Basic
<\$EntryAuthorURL\$>	The entry author's URL	Basic

<code><\$EntryPermalink\$></code>	The URL of the entry page of the entry.	Basic
---	---	-------

In addition, the `<BlogEntry>` container is comprised of the following optional sub containers...

<EntryTitle>

The `<EntryTitle>` and `</EntryTitle>` specify how to lay out the entry's title if it has one. The `<$EntryTitle$>` tag only has meaning between these two tags.

Example: `<EntryTitle> <h2><$EntryTitle$></h2></EntryTitle>`

Day Headers and Footers

Blog entries can be grouped by days on generated pages through the use of `<DayHeader>` and `<DayFooter>`. `<$DayHeaderDate$>` is replaced with the date of the day and only has meaning between `<DayHeader>` and `</DayHeader>`. The Day Header Date will appear at the start of each new day of posts.

Example: `<DayHeader>Posts for <$DayHeaderDate$></DayHeader>`

Any HTML that you want to appear at the end of a day of posts should be contained within `<DayFooter>` and `</DayFooter>`.

Example: `<DayFooter> <hr> </DayFooter>`

<EntryModifiedDate>

If an entry has been modified, you can specify the HTML that signifies the modification through the use of the `<EntryModifiedDate>` and `</EntryModifiedDate>` tags. The `<$EntryModifiedDate$>` is replaced with the date and time of the modification and only works between `<EntryModifiedDate>` and `</EntryModifiedDate>`.

Example: `<EntryModifiedDate>Edited on: <$EntryModifiedDate$></EntryModifiedDate>`

<EntryCategories>

You can display which categories and entry belongs to through the use of `<EntryCategories>` and `</EntryCategories>`.

The `EntryCategories` container works the same as a `<CategoryList>` containers and takes the following attributes...

- **sort_order** – Sorts the list either ascending or descending depending on the attribute value (ascend or descend)
- **glue** – specifies the "glue" that separates each category page.

Example:

```
<EntryCategories glue=", ">
<a href="<$CategoryLink$"><$CategoryName$></a>
</EntryCategories >
```

A typical <BlogEntry> container might look like this:

```
<BlogEntry>
<DayHeader><h2><$DayHeaderDate$></h2></DayHeader>
<a name="<$EntryID$"></a>
<EntryTitle>
<h3><$EntryTitle$></h3>
</EntryTitle>
<$EntryBody$>
<br>
Posted by <a href="mailto:<$EntryAuthorEmail$"><$EntryAuthor$></a> at
<a href="<$EntryArchivePage$>#<$EntryID$">
title="permalink"><$EntryTime$></a><br>
<EntryModifiedDate>
<i>Edited on: <$EntryModifiedDate$></i><br>
</EntryModifiedDate>
Categories: <EntryCategories glue=", "><a
href="<$CategoryLink$"><$CategoryName$></a></EntryCategories><br>
</div>
</div>
</BlogEntry>
```

NextPage/PreviousPage Containers

These containers are replaced with link to the next/previous pages on the archive, category, and entry pages.

The `<NextPage>` container syntax is:

```
<NextPage>
  <IfPageExists>
    <$PageLink$><$PageName$>
  </IfPageExists>
  <IfNoPageExists></IfNoPageExists>
</NextPage>
```

The `<PreviousPage>` container syntax is:

```
<PreviousPage>
  <IfPageExists>
    <$PageLink$><$PageName$>
  </IfPageExists>
  <IfNoPageExists></IfNoPageExists>
</PreviousPage>
```

Example: Next and previous links on the archive pages

```
<PreviousPage>
  <IfPageExists>
    <a href="<$PageLink$>">&laquo; <$PageName$></a> |
  </IfPageExists>
  <IfNoPageExists>
    No More Archives
  </IfNoPageExists>
</PreviousPage>
<a href="<$FrontPageLink$>">Main</a>
<NextPage>

  <IfPageExists>
    | <a href="<$PageLink$>"><$PageName$> &raquo;</a>
  </IfPageExists>
  <IfNoPageExists>
    No More Archives
  </IfNoPageExists>
</NextPage>
```

Output:

« December 2004 | Main | February 2005 »

Include Container

The `<Include>` container can be used on any template and prints the contents of a file into the template. The syntax is:

```
<Include file="filepath">  
    <${IncludeText}>  
</Include>
```

The tag `<${IncludeText}>` is replaced with the contents of the file IF the file exists and can be read. The filepath attribute might be, for instance, `C:\myfolder\myfile.txt`.

What's new in 1.5

Code:

```
<$EntryExtra1$>
<$EntryExtra2$>
<$EntryKeywords$>
<$EntryDescription$>
```

Containers:

Code:

```
<AuthorList>
  <$AuthorName$>
  <$AuthorURL$>
  <$AuthorEmail$>
  <$AuthorDetails$>
</AuthorList>
```

Code:

```
<Ignore>
Anything inside this is ignored and will not be output.
</Ignore>
```

Code:

```
<Entries>
</Entries>
```

Same as `<BlogEntry>` except that it doesn't care which template it is on. By default it will always return a list of the most recent entries. This is different from `<BlogEntry>` which by default knows about the context of the template and will return the entries appropriate for that page.

Both `<Entries>` and `<BlogEntry>` can now take a few arguments to provide more control about what entries get output. Here are some examples:

Code:

```
<Entries category="What Ever"></Entries>
```

This will output the the entries only belonging to a category "What Ever" on any page.

Code:

```
<Entries category="What Ever" entries_between="2008-12-31 AND 2010-12-31">
</Entries>
```

THIS will output the the entries in "What Ever" between 2008-12-31 and 2010-12-31

Code:

```
<Entries id="10">
</Entries>
```

This will output the entry with ID of 10 (if it exists)

Also new are "Labels." Labels are similar to categories except that no page is generated for a Label. Labels are use in conjunction with the arguments above to help layout a page. For example, suppose you wanted to have "Sticky posts" at the top of the front page of your blog. You'd create a label called "sticky" and do something like this on the front page template..

Code:

```
<h1>Sticky Posts</h1>
<Entries label="sticky">
Only "sticky" posts will be here
...
</Entries>

<h2>Regular Blog posts</h2>
<BlogEntry>
The usual blog entry stuff goes here
...
</BlogEntry>
```