



easy template system

version 3.05b

Franck Marcia
Copyright © 2002, 2003

Table of content

Introduction	3
PHP functions	3
Quick reference	4
Tutorial.....	5
1. Scalar variables.....	5
2. Objects.....	6
3. Arrays.....	7
4. Outer templates.....	8
5. Constant elements	9
6. Alternate tags	10
7. PHP elements.....	11
8. Missing values	12
9. Simple conditional elements	13
10. Choose-variable elements.....	14
11. Other conditional elements.....	15
12. Call elements	16
13. Path (part 1)	17
14. Path (part 2)	19
Reference	20
1. Simple tag element.....	20
2. Alternate tag element.....	22
3. Template element	24
4. Set element	26
5. Set-value element	27
6. Missing element	28
7. Missing-value element	29
8. PHP element.....	30
9. Constant element.....	31
10. If element.....	33
11. Choose element.....	34
12. When-test element.....	35
13. Else element	36
14. Choose-variable element.....	37
15. When-value element	38
16. Call element.....	39
17. Argument element.....	41
18. Repeat element.....	42
19. Include element.....	43
20. Insert element.....	44
21. Eval element.....	45
22. Safe element	46
23. Reduce element.....	47
24. Comment.....	48
25. Cdata	49
26. System variables	50
27. Path.....	51
28. Error handling	52
29. User-level templates storage	53
30. Cache system	54
31. Miscellaneous.....	57
Version history	58
License	62
Contact	66

Introduction

ETS is a template system written with PHP that enables you to transform a set of data to any type of document.

For example, ETS can transform a list of product descriptions retrieved from a database to a HTML page. It can also construct SQL statements, ASCII data, XML documents...

ETS provides 2 functions to match a set of data with templates:

- `sprintt` which returns the built template as a string,
- `printt` which prints it out.

ETS supplies:

- array management,
- various conditional elements,
- access to any level in the data tree from any level in the template,
- data formatting,
- size reducing,
- integrated debug messages.

ETS works with 2 elements: the data tree and templates. The data tree contains every data that will be available. Templates define the way the data tree will be presented. You can compare it to a XML document transformed with a XSLT template: it's exactly the same concept.

ETS can manage recursive templates, allows a complete reshuffle of the template with exactly the same data tree, is extremely valuable when working with database because of the implicit use of templates...

... It's a powerful tool that will help you efficiently to build documents.

PHP functions

ETS provides 2 functions. Here are their prototypes:

```
void printt(mixed datatree, mixed containers [, string entry])  
    Output a built template
```

```
string sprintt(mixed datatree, mixed containers [, string entry])  
    Return a built template
```

`datatree` can be an object, an array of objects or a NULL value. The object must be organized to store scalar values in object properties and lists in arrays. An object can contain children objects. NULL value is used to build a template without any data.

`containers` can be an array or a string. If it's an array, each element will be considered as a container name. If it's a string, it may contain one container name or a list of container names separated by commas.

`entry` defines the main template defined in containers. It's "main" by default. It's useful when you want to define several independent sets of templates in the same container.

Quick reference

The following table presents elements provided by ETS.

<i>Element</i>	<i>Description</i>
{name}	Places an outer template or the value of a variable
{variable:template}	Places an outer template with a different name
{mask:name} ... {/mask}	Defines a template and, if inner, places it
{set:variable} ... {/set}	Places a content when a variable is set (not missing)
{set:variable:value} ... {/set}	Places a content when a variable has a specific value
{mis:variable} ... {/mis}	Places a content when a variable is missing
{mis:variable:value} ... {/mis}	Places a content when a variable has not a specific value or is missing
{php} ... {/php}	Places a content which will be parsed as PHP code
{const:template}	Places an outer template
{if:test} ... {/if}	Places a content if a test is TRUE
{choose} ... {/choose}	Provides multiple conditions in conjunction with when-test and else elements
{when:test} ... {/when}	Places a content if a test is TRUE so disables other when-test or else siblings
{else} ... {/else}	Places a content if all when-test or when-value siblings are FALSE
{choose:variable} ... {/choose}	Provides multiple conditions in conjunction with when-value and else elements
{when:value} ... {/when}	Places a content if the variable of the choose parent has the value of this element so disables other when-value or else siblings
{call:template} ... {/call}	Places an outer template with arguments
{arg:name} ... {/arg}	Defines an argument of a call element
{repeat:n} ... {/repeat}	Places a content several times
{include:container}	Includes outer templates of a container
{insert:container}	Places the content of a container as text
{eval:container}	Places the content of a container as a template part
{safe:container}	Places the content of a container as a template part with restrictions
{reduce:value}	Defines a size reducing behaviour
{* ... *}	Defines a comment section
{# ... #}	Defines a section where the text is not parsed

Tutorial

You will discover in this tutorial many examples that gradually introduce you to essentials of ETS.

1. *Scalar variables*

Consider this data tree which contains general elements of a HTML page:

```
$page->title = 'Home page';  
$page->lastmodified = 'Thursday, January 23, 2003';
```

This template file, `ets.tpl`, contains the presentation of these data:

```
{mask:main}  
<html>  
  <head>  
    <title>{title}</title>  
  </head>  
  <body>  
    <h1>{title}</h1>  
    <hr>  
    <div align="center">Last modified: {lastmodified}</div>  
  </body>  
</html>  
{/mask}
```

If you run `printt($page, 'ets.tpl');` you will get:

```
<html>  
  <head>  
    <title>Home page</title>  
  </head>  
  <body>  
    <h1>Home page</h1>  
    <hr>  
    <div align="center">Last modified: Thursday, January 23, 2003</div>  
  </body>  
</html>
```

What happened?

ETS found the main template defined between tags `{mask:main}` and `{/mask}`,
ETS found the tag `{title}` twice and the tag `{lastmodified}` in main,
ETS replaced `{title}` with the value of `$page->title` and `{lastmodified}` with the value of `$page->lastmodified`,
ETS printed out the result.

Why did it happen?

The first level of the data tree, the root, corresponds to the first level of the template tree, the main template. In this example, the object `$page` corresponds to the template `main`. Then, every property of `$page` can be placed in the template `main`. In this example, the property `title` corresponds to the tag `{title}` and `lastmodified` to `{lastmodified}`.

2. Objects

We now add partner information to the previous data tree and group them in an object:

```
$page->title = 'Home page';  
$page->lastmodified = 'Thursday, January 23, 2003';  
  
$page->partner->name = 'foobar.com';  
$page->partner->id = 123;
```

The template file, `ets.tpl`, is modified to show these new data:

```
{mask:main}  
<html>  
  <head>  
    <title>{title}</title>  
  </head>  
  <body>  
    <h1>{title}</h1>  
    <hr>  
    <div align="center">Last modified: {lastmodified}</div>  
    {mask:partner}<div>with our partner {name} ({id})</div>{/mask}  
  </body>  
</html>  
{/mask}
```

If you run `printt($page, 'ets.tpl');` again, you will get now:

```
<html>  
  <head>  
    <title>Home page</title>  
  </head>  
  <body>  
    <h1>Home page</h1>  
    <hr>  
    <div align="center">Last modified: Thursday, January 23, 2003</div>  
    <div>with our partner foobar.com (123)</div>  
  </body>  
</html>
```

What happened?

...

ETS found the inner template `partner` between tags `{mask:partner}` and `{/mask}`.

ETS replaced `{name}` with the value of `$page->partner->name` and `{id}` with the value of `$page->partner->id`.

...

Why did it happen?

When ETS detects a template in another one, it changes the current level then begins again the process for the new level.

In this example, it changes from `$page` to `$page->partner` for data and from `main` to `partner` for templates. Then, every property of `$page->partner` are placed in the template `partner`. In this example, the property `id` corresponds to the tag `{id}` and `name` to `{name}`.

3. Arrays

We now want to add menu data to the previous data tree. If we apply the principle of our previous example, we can add these data like this:

```
$page->menu1->url = "download.php";  
$page->menu1->label = "Downloads";  
$page->menu2->url = "links.php";  
$page->menu2->label = "Links";
```

and so, create two new templates in the main template.

There's a better way to do it.

Instead of two new objects, we create an array of objects:

```
$page->title = 'Home page';  
$page->lastmodified = 'Thursday, January 23, 2003';  
  
$page->partner->name = 'foobar.com';  
$page->partner->id = 123;  
  
$page->menu[1]->url = "download.php";  
$page->menu[1]->label = "Downloads";  
$page->menu[2]->url = "links.php";  
$page->menu[2]->label = "Links";
```

Then we create one new template:

```
{mask:main}  
<html>  
  <head><title>{title}</title></head>  
  <body>  
    <h1>{title}</h1><hr>  
    {mask:menu}<a href="{url}">{label}</a> {/mask}<hr>  
    <div align="center">Last modified: {lastmodified}</div>  
    {mask:partner}<div>with our partner {name} ({id})</div>{/mask}  
  </body>  
</html>  
{/mask}
```

ETS will produce:

```
<html>  
  <head><title>Home page</title></head>  
  <body>  
    <h1>Home page</h1><hr>  
    <a href="download.php">Downloads</a> <a href="links.php">Links</a> <hr>  
    <div align="center">Last modified: Thursday, January 23, 2003</div>  
    <div>with our partner foobar.com (123)</div>  
  </body>  
</html>
```

What happened?

...

ETS found the inner template menu between tags {mask:menu} and {/mask}.

For each element of the array menu, ETS replaced {url} with the value of the property url and {label} with label.

...

Why did it happen?

When ETS detects that a template corresponds to an array instead of an object, it does the same than for an object but for each element of this array then concatenates built templates.

4. Outer templates

We want now to print information about our partner at the bottom and at the top of the page. We don't add new data but just rearrange the template file like this:

```
{mask:main}
<html>
  <head>
    <title>{title}</title>
  </head>
  <body>
    <h1>{title}</h1>
    <hr>
    {partner}
    <hr>
    {mask:menu}<a href="{url}">{label}</a> {/mask}
    <hr>
    <div align="center">Last modified: {lastmodified}</div>
    {partner}
  </body>
</html>
{/mask}

{mask:partner}
<div>with our partner {name} ({id})</div>
{/mask}
```

ETS will produce:

```
<html>
  <head>
    <title>Home page</title>
  </head>
  <body>
    <h1>Home page</h1>
    <hr>
    <div>with our partner foobar.com (123)</div>
    <hr>
    <a href="download.php">Downloads</a> <a href="links.php">Links</a>
    <hr>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
    <div>with our partner foobar.com (123)</div>
  </body>
</html>
```

What happened?

...

ETS found the tag {partner} twice.

ETS replaced {name} with the value of \$page->partner->name and {id} with the value of \$page->partner->id in the outer template partner.

ETS replaced {partner} by the built template partner.

...

Why did it happen?

When ETS detects that a tag corresponds to an object instead of a scalar variable, it looks for an outer template with the same name then does placements for each property of the object exactly like a inner template.

It works the same way when the corresponding variable is an array but the template is duplicated for each element of the array.

5. Constant elements

We want now to change the basic `<hr>` tag with a wonderful image we found on the web. We don't add new data but just rearrange again the template file like this:

```
{mask:main}
<html>
  <head>
    <title>{title}</title>
  </head>
  <body>
    <h1>{title}</h1>
    {const:separator}
    {partner}
    {const:separator}
    {mask:menu}<a href="{url}">{label}</a> {/mask}
    {const:separator}
    <div align="center">Last modified: {lastmodified}</div>
    {partner}
  </body>
</html>
{/mask}

{mask:partner}
<div>with our partner {name} ({id})</div>
{/mask}

{mask:separator}
<br>
{/mask}
```

ETS will produce:

```
<html>
  <head>
    <title>Home page</title>
  </head>
  <body>
    <h1>Home page</h1>
    <br>
    <div>with our partner foobar.com (123)</div>
    <br>
    <a href="download.php">Downloads</a> <a href="links.php">Links</a>
    <br>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
    <div>with our partner foobar.com (123)</div>
  </body>
</html>
```

What happened?

...

ETS found the tag `{const:separator}` three times.

ETS replaced each `{const:separator}` with the content of the outer template separator.

...

Why did it happen?

When ETS detects a constant element, it looks for an outer template with this name then replaces the constant element with the content of the template.

If the outer template contains tags, they are parsed at the same level and not at the next one. In this example, `{title}` would be replaced by "Home page" like others `{title}` tags of the template main.

6. Alternate tags

We want now to get two different templates with partner's data.

The template file is now:

```
{mask:main}
<html>
  <head><title>{title}</title></head>
  <body>
    <h1>{title}</h1>
    <hr>
    {partner:partner1}
    <hr>
    {mask:menu}<a href="{url}">{label}</a> {/mask}
    <hr>
    <div align="center">Last modified: {lastmodified}</div>
    {partner:partner2}
  </body>
</html>
{/mask}

{mask:partner1}
<p>with our partner {name}</p>
{/mask}

{mask:partner2}
<div>with our partner {name} ({id})</div>
{/mask}
```

ETS will produce:

```
<html>
  <head><title>Home page</title></head>
  <body>
    <h1>Home page</h1>
    <hr>
    <p>with our partner foobar.com</p>
    <hr>
    <a href="download.php">Downloads</a> <a href="links.php">Links</a>
    <hr>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
    <div>with our partner foobar.com (123)</div>
  </body>
</html>
```

What happened?

...

ETS found the alternate tag {partner:partner1}.

ETS replaced {name} with the value of \$page->partner->name in the template partner1.

ETS replaced {partner:partner1} with the built template partner1.

...

Why did it happen?

When ETS finds an alternate tag (two names separated by a colon), it changes the active level using the first name then uses the outer template defined by the second name. Next, every property of the new level are placed. In this example, the property name corresponds to the tag {name}.

7. PHP elements

Imagine now that we want in the second template of partner, the name to be in capitals.

Here is our new again template file:

```
{mask:main}
<html>
  <head><title>{title}</title></head>
  <body>
    <h1>{title}</h1>
    <hr>
    {partner:partner1}
    <hr>
    {mask:menu}<a href="{url}">{label}</a> {/mask}
    <hr>
    <div align="center">Last modified: {lastmodified}</div>
    {partner:partner2}
  </body>
</html>
{/mask}

{mask:partner1}
<p>with our partner {php}strtoupper({name}){/php}</p>
{/mask}

{mask:partner2}
<div>with our partner {name} ({id})</div>
{/mask}
```

ETS will produce:

```
<html>
  <head><title>Home page</title></head>
  <body>
    <h1>Home page</h1>
    <hr>
    <p>with our partner FOOBAR.COM</p>
    <hr>
    <a href="download.php">Downloads</a> <a href="links.php">Links</a>
    <hr>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
    <div>with our partner foobar.com (123)</div>
  </body>
</html>
```

What happened?

...

ETS found the PHP element defined between tags {php} and {/php}.

After placements in it, ETS places the result of the PHP code contained in the element.

...

Why did it happen?

When ETS finds a PHP element, it calls PHP to parse its content and places the result. The PHP code must be written to stand for a scalar value. It is not recommended to use PHP elements to do something else than formatting data to preserve the separation logic.

8. Missing values

We want the template to be valid even if, for some specific contexts, there is no partner information to print out.

Here is the data tree, without partner's data.

```
$page->title = 'Home page';
$page->lastmodified = 'Thursday, January 23, 2003';

$page->menu[1]->url = "download.php";
$page->menu[1]->label = "Downloads";
$page->menu[2]->url = "links.php";
$page->menu[2]->label = "Links";
```

We don't modify the template file:

```
{mask:main}
<html>
  <head><title>{title}</title></head>
  <body>
    <h1>{title}</h1>
    <hr>
    {partner:partner1}
    <hr>
    {mask:menu}<a href="{url}">{label}</a> {/mask}
    <hr>
    <div align="center">Last modified: {lastmodified}</div>
    {partner:partner2}
  </body>
</html>
{/mask}

{mask:partner1}
<p>with our partner {php}strtoupper({name}){/php}</p>
{/mask}

{mask:partner2}
<div>with our partner {name} ({id})</div>
{/mask}
```

ETS will produce:

```
<html>
  <head><title>Home page</title></head>
  <body>
    <h1>Home page</h1>
    <hr>
    <hr>
    <a href="download.php">Downloads</a> <a href="links.php">Links</a>
    <hr>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
  </body>
</html>
```

What happened?

...

ETS found the tag {partner:partner1}.

ETS didn't find partner in the data tree to match with the template so deleted the tag.

...

Why did it happen?

When ETS finds a simple tag, an alternate tag or an inner template which doesn't match with any data of the data tree, it deletes the element.

9. Simple conditional elements

We want now to print out a message when partner is missing.

We keep the same data tree, but modify the template file:

```
{mask:main}
<html>
  <head><title>{title}</title></head>
  <body>
    <h1>{title}</h1>
    <hr>
    {partner:partner1}{mis:partner}<b>No partner</b>{/mis}
    <hr>
    {mask:menu}<a href="{url}">{label}</a> {/mask}
    <hr>
    <div align="center">Last modified: {lastmodified}</div>
    {partner:partner2}
  </body>
</html>
{/mask}

{mask:partner1}
<p>with our partner {php}strtoupper({name}){/php}</p>
{/mask}

{mask:partner2}
<div>with our partner {name} ({id})</div>
{/mask}
```

ETS will print out:

```
<html>
  <head><title>Home page</title></head>
  <body>
    <h1>Home page</h1>
    <hr>
    <b>No partner</b>
    <hr>
    <a href="download.php">Downloads</a> <a href="links.php">Links</a>
    <hr>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
  </body>
</html>
```

What happened?

...

ETS found the template partner between tags {mis:partner} and {/mis}.

ETS didn't find partner in the current level of the data tree so placed the content of this element.

...

Why did it happen?

When ETS finds a missing element, it places it if the corresponding data is not set, is null or doesn't exist.

The set element {set:partner} ... {/set} would be placed if partner was not missing.

ETS also provides {mis:variable:value} ... {/mis} which is placed when variable is different to value or variable is missing and {set:variable:value} ... {/set} which is placed when variable has a value of value.

10. Choose-variable elements

We want now to specify a style for each known partner.

Partner's data are back:

```
$page->title = 'Home page';
$page->lastmodified = 'Thursday, January 23, 2003';

$page->partner->name = 'foobar.com';
$page->partner->id = 123;
```

Here is the new template file:

```
{mask:main}
<html>
  <head><title>{title}</title></head>
  <style>
    {mask:partner}
    {choose:id}
      {when:120} body { color:red; } {/when}
      {when:123} body { color:blue; } {/when}
      {else} body { color:black; } {/else}
    {/choose}
  {/mask}
</style>
<body>
  <h1>{title}</h1>
  <hr>
  {mask:partner}<p>with our partner {name}</p>{/mask}
  <hr>
  <div align="center">Last modified: {lastmodified}</div>
</body>
</html>
{/mask}
```

ETS will produce:

```
<html>
  <head><title>Home page</title></head>
  <style>
    body { color:blue; }
  </style>
  <body>
    <h1>Home page</h1>
    <hr>
    <p>with our partner foobar.com</p>
    <hr>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
  </body>
</html>
```

What happened?

...

ETS found the choose-variable element between tags {choose:id} and {/choose}.

ETS placed the content of the when-value element between tags {when:123} and {/when}.

...

Why did it happen?

When ETS finds a choose-variable element, it places the content of the when-value element which corresponds to the value of the variable. If no when-value element corresponds, ETS places the else element.

11. Other conditional elements

ETS provides two other conditional elements.

If element

The if element `{if:test} ... {/if}` is used when you want to do more complex conditions. The part `test` of the tag `{if:test}` contains any valid PHP condition using simple tags.

Here is a valid if element:

```
{if: {id} > 99 and {name} != 'foobar.com'} ... {/if}
```

Simple choose element

The simple choose element `{choose} ... {choose}` is used when you've got to construct a condition sequence if-elseif-else.

Here is a valid simple choose element:

```
{choose}
  {when: {id} < 100} ... {/when}
  {when: {id} == 100} ... {/when}
  {else} ... {/else}
{/choose}
```

Note: these two element types are very useful in complex situations but there are twice slower than others conditional elements. So they should be used if simpler ones can't.

12. Call elements

Here is the extraction of a template file:

```
...
<td width="{const:a_width}"><a href="index.php?sort=a&{const:param}">{aa}</a></td>
<td width="{const:b_width}"><a href="index.php?sort=b&{const:param}">{ab}</a></td>
<td width="{const:c_width}"><a href="index.php?sort=c&{const:param}">{ac}</a></td>
...
{/mask}
```

We want to use the same template with different data because the template is heavily used and supposed to often change.

And here is the new one, once the template is modified to used a template:

```
...
{call:td}{arg:ta}{const:a_width}{/arg}{arg:tb}a{/arg}{arg:tc}{aa}{/arg}{/call}
{call:td}{arg:ta}{const:b_width}{/arg}{arg:tb}b{/arg}{arg:tc}{ab}{/arg}{/call}
{call:td}{arg:ta}{const:c_width}{/arg}{arg:tb}c{/arg}{arg:tc}{ac}{/arg}{/call}
...
{/mask}

{mask:td}
<td width="{ta}"><a href="index.php?sort={tb}&{const:param}">{tc}</a></td>
{/mask}
```

The two constructs will produce the same output. For example:

```
...
<td width="30"><a href="index.php?sort=a&par1=1&par2=2">One</a></td>
<td width="40"><a href="index.php?sort=b&par1=1&par2=2">Two</a></td>
<td width="30"><a href="index.php?sort=c&par1=1&par2=2">Three</a></td>
...
{/mask}
```

What happened?

...

ETS found the call element between tags {call:td} and {/call}.

ETS built the outer template td using built arguments of the call element.

ETS replaced the call element by the built template td.

...

Why did it happen?

When ETS detects a call element, it places the called outer template then places argument elements of the call element.

If the call element hasn't any argument, you can use a constant element instead.

13. Path (part 1)

Menu data are now back and we want to trace the active partner when a visitor clicks our links.

Here is the data:

```
$page->title = 'Home page';
$page->lastmodified = 'Thursday, January 23, 2003';

$page->partner->name = 'foobar.com';
$page->partner->id = 123;

$page->menu[1]->url = "download.php";
$page->menu[1]->label = "Downloads";
$page->menu[2]->url = "links.php";
$page->menu[2]->label = "Links";
```

Here is the new template file:

```
{mask:main}
<html>
  <head><title>{title}</title></head>
  <body>
    <h1>{title}</h1><hr>
    {menu}<hr>
    <div align="center">Last modified: {lastmodified}</div>
  </body>
</html>
{/mask}

{mask:menu}
<a href="{url}?p={//partner/id}">{label}</a>
{/mask}
```

ETS will produce:

```
<html>
  <head><title>Home page</title></head>
  <body>
    <h1>Home page</h1><hr>
    <a href="download.php?p=123">Downloads</a> <a href="links.php?p=123">Links</a><hr>
    <div align="center">Last modified: Thursday, January 23, 2003</div>
  </body>
</html>
```

What happened?

...

ETS found the tag { //partner/id } in the outer template menu.

ETS replaced it with the value of \$page->partner->id.

...

Why did it happen?

When ETS finds an absolute path (a path that begins with and double slash) in a valid element, it starts again from the root of the data tree to find the corresponding property. In this example, ETS looks for \$page->partner->id.

ETS also manages relative paths. In part 10, you could use {choose:partner/id} ... {/choose} instead of {mask:partner}{choose:{id}} ... {/choose}{/mask} in the main template to jump to the property id of the level partner of the current level.

Absolute and relative paths accept also index to specify a particular element of an array. For example, { / /nameA[3] /nameB } places nameB property of the fourth element of nameA.

You can also use system indexes `_first` and `_last` which represent first and last elements of the array.

Relative and absolute paths are available for simple tags, alternate tags, template and simple conditional elements, but also for the test part of complex conditional elements.

14. Path (part 2)

Imagine finally you want to display label's partners based on an indexed array...

Here is the data:

```
$page->title = 'Path using ..';

$page->mypartners[0]->id = 0;
$page->mypartners[1]->id = 3;

$page->partners[1]->id = 1;
$page->partners[1]->label = 'foobar.com';
$page->partners[2]->id = 2;
$page->partners[2]->label = 'foo.bar';
$page->partners[3]->id = 3;
$page->partners[3]->label = 'foobar 1';
```

Here is the new template file:

```
{mask:main}
<html>
  <head><title>{title}</title></head>
  <body>
    <h1>{title}</h1><hr>
    {mask:mypartners}
      {mask://partners}
        {if: {id} == {../../id} }<p>{label}</p>{/if}
      {/mask}
    {/mask}
  </body>
</html>
{/mask}
```

ETS will produce:

```
<html>
  <head><title>Path using ..</title></head>
  <body>
    <h1>Path using ..</h1><hr>
    <p>foobar.com</p>
    <p>foobar 1</p>
  </body>
</html>
```

What happened?

...

ETS found the tag {../../id} in the test part of the if element.

ETS replaced the tag with the value of the variable id of the current element of the array mypartners.

...

Why did it happen?

When ETS finds a parent element in a path, it goes back to the parent of the current template and retrieved associated level to define the new active level.

To bypass a template starting with //, you must use .. twice.

ETS creates numbers of elements when working with arrays: you can use `_start`, `_previous`, `_next` and `_end` to access respectively to the first, the previous, the next or the last element of the array which the current element belongs to. There's also `_parent` which is a alias for ..

Reference

You will find here the complete description of each element of ETS.

1. Simple tag element

Places an outer template or the value of a variable.

Syntax

{name}

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Test of if and when-test elements Any element aside from call, choose and choose-variable elements
<i>Child elements</i>	None

Rules

if the parent element is a test of a conditional element
 if the variable `name` is a string
 place the escaped string between double quotes
 else if the variable `name` is boolean
 place `TRUE` or `FALSE`
 else if the variable `name` is numeric
 place the value
 else if the variable is missing
 place `NULL`
 else
 disappear
else
 if the outer template `name` exists
 if the variable `name` is scalar
 place the content of the template
 else if the variable `name` is an object
 place the content of the template
 change the current level to `name`
 else if the variable `name` is an array
 for each element of the array
 place the content of the template
 change the current level to `name`
 else
 disappear
 else
 if the variable `name` is scalar
 place the value of this variable
 else
 disappear

Remarks

FALSE values and empty strings stand for scalar value in a test of a conditional element, PHP element and evaluated elements and for missing elsewhere. NULL values, non-existing variables, empty arrays and empty objects are always missing.

Strings are placed with double quotes and are escaped to accept carriage returns and tabulations in PHP elements and evaluated part of elements. There are not otherwise.

This element accepts absolute and relative paths for `name`.

Examples with scalar variables

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<code>{mask:main}{a}{/mask}</code>	<code>\$ets->a = 4;</code>	4
<code>{mask:main}{b}{/mask}</code>	<code>\$ets->b = NULL;</code>	(none)

Examples with outer templates

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<code>{mask:main} {c} {/mask}</code> <code>{mask:c}c with {d}{/mask}</code>	<code>\$ets->c = 1; \$ets->d = 2;</code>	c with 2
<code>{mask:main} {e} {/mask}</code> <code>{mask:e}e with {f}{/mask}</code>	<code>\$ets->e->f = 2;</code>	e with 2
<code>{mask:main} {e} {/mask}</code> <code>{mask:e}Content of e{/mask}</code>	<code>\$ets->a = 1;</code>	(none)

Examples in tests

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<code>{if: {a} == 4 }a == 4{/if}</code>	<code>\$ets->a = 4;</code>	a == 4
<code>{if: {a} }a is TRUE{/if}</code>	<code>\$ets->a = TRUE;</code>	a is TRUE
<code>{if: {a} == 'ets' }ETS{/if}</code>	<code>\$ets->a = 'ets';</code>	ETS
<code>{if: {a} == 1 }a == 1{/if}</code>	<code>\$ets->not_a = 1;</code>	(none)
<code>{if: {a[1]/b} == 2}b == 2{/if}</code>	<code>\$ets->a[0]->b = 1; \$ets->a[1]->b = 2;</code>	b == 2

Examples in evaluated part of elements

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<code>{repeat: {a} / 10}*{/repeat}</code>	<code>\$ets->a = 30;</code>	***
<code>{mask:main} {b:{c}} {/mask}</code> <code>{mask:d}-{e}-{/mask}</code>	<code>\$ets->b->e = ' '; \$ets->c = 'd';</code>	- -

2. Alternate tag element

Places an outer template with a different name.

Syntax

```
{variable:template}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	In <code>template</code> , PHP code, simple tags

Rules

place simple tags in `template`

if `template` is a valid PHP code that stands for a string value

 if the outer template `template` exists

 if the variable `variable` is scalar

 place the content of the template

 else if the variable `variable` is an object

 place the content of the template

 change the current level to `variable`

 else if the variable `name` is an array

 for each element of the array

 place the content of the template

 change the current level to `name`

 else

 disappear

 else `variable` is scalar

 place the value of this variable

 else

 disappear

else if `template` produces a fatal error

 halt the script

else

 disappear

Remarks

FALSE values, empty strings, NULL values, non-existing variables, empty arrays and empty objects stand for missing in `variable`.

This element accepts absolute and relative paths for `variable`.

About simple tags in `template`:

- FALSE values and empty strings stand for scalar value,
- strings are placed with double quotes and are escaped to accept carriage returns and tabulations.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{mask:main} {a:b} {/mask} {mask:b}b with {a}{/mask}</pre>	<pre>\$ets->a = 1;</pre>	b with 1
<pre>{mask:main} {c:d} {/mask} {mask:d}c with {e}{/mask}</pre>	<pre>\$ets->c->e = 1;</pre>	c with 1
<pre>{mask:main} f = {f:xx} {/mask}</pre>	<pre>\$ets->f = 1;</pre>	f = 1

3. Template element

Defines a template and, if inner, places it.

Syntax

```
{mask:name} ... {/mask}
```

Template tree

<i>Number of occurrences</i>	Unique name if outer Unlimited if inner
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements if inner None if outer
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

```
if the variable name is scalar
    place the content of the template
else if the variable name is an object
    place the content of the template
    change the current level to name
else if the variable name is an array
    for each element of the array
        place the content of the template
        change the current level to name
else
    disappear
```

Remarks

This element accepts absolute and relative paths for `name` for inner templates. They don't in outer templates.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<code>{mask:main} {mask:a}Content of a{/mask} {/mask}</code>	<code>\$ets->a = 1;</code>	Content of a
<code>{mask:main} {mask:b}Content of b{/mask} {/mask}</code>	<code>\$ets->b->a = 1;</code>	Content of b
<code>{mask:main} {mask:c}Content of c{/mask} {/mask}</code>	<code>\$ets->c[0]->a = 1; \$ets->c[1]->a = 1; \$ets->c[2]->a = 1;</code>	Content of c Content of c Content of c
<code>{mask:main} {mask:d}Content of d{/mask} {/mask}</code>	<code>\$ets->d = '';</code>	(none)
<code>{mask:main} {mask:e}Content of e{/mask} {/mask}</code>	<code>\$ets->e[0][0]->a = 1; \$ets->e[0][1]->a = 1; \$ets->e[1][0]->a = 1; \$ets->e[1][1]->a = 1;</code>	Content of e Content of e Content of e Content of e

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre> {mask:main} {mask:f} 1st f {mask://f}2nd f{/mask} {/mask} {/mask> </pre>	<pre> \$ets->f[0]->a = 1; </pre>	<pre> 1st f 2nd f </pre>

4. Set element

Places a content when a variable is set (not missing).

Syntax

```
{set:variable} ... {/set}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

```
if variable is not missing
    place the content of the element
else
    disappear
```

Remarks

FALSE values, empty strings, NULL values, non-existing variables, empty arrays and empty objects stand for missing.

This element accepts absolute and relative paths for `variable`.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
{set:a}a is not missing{/set}	<code>\$ets->a = TRUE;</code>	a is not missing
{set:b}b is not missing{/set}	<code>\$ets->b = NULL;</code>	(none)
{set:c}c is not missing{/set}	<code>\$ets->c = FALSE;</code>	(none)
{set:d}d is not missing{/set}	<code>\$ets->d = '';</code>	(none)
{set:e}e is not missing{/set}	<code>\$ets->e->a = 1;</code> <code>\$ets->e->b = 2;</code>	e is not missing

5. Set-value element

Places a content when a variable has a specific value.

Syntax

```
{set:variable:value} ... {/set}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

```
if variable has a value of value
    place the content of the element
else
    disappear
```

Remarks

{set:variable:NULL} or {set:variable:} don't work. Use {mis:variable} instead.

This element accepts absolute and relative paths for `variable`.

`value` accepts simple strings without spaces, single and double quotes strings.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
{set:a:3}a == 3{/set}	<code>\$ets->a = 3;</code>	a == 3
{set:b:value}b == 'value'{/set}	<code>\$ets->b = 'value';</code>	b == 'value'
{set:c:"val ue"}c == "val ue"{/set}	<code>\$ets->c = 'value';</code>	(none)
{set:d:4}d == 4{/set}	<code>\$ets->d = NULL;</code>	(none)

6. Missing element

Places a content when a variable is missing.

Syntax

```
{mis:variable} ... {/mis}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

```
if foo is missing
    place the content of the element
else
    disappear
```

Remarks

FALSE values, empty strings, NULL values, non-existing variables, empty arrays and empty objects stand for missing.

This element accepts absolute and relative paths for `variable`.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
{mis:a}a is missing{/mis}	<code>\$ets->a = TRUE;</code>	(none)
{mis:b}b is missing{/mis}	<code>\$ets->b = NULL;</code>	b is missing
{mis:c}c is missing{/mis}	<code>\$ets->c = FALSE;</code>	c is missing
{mis:d}d is missing{/mis}	<code>\$ets->d = '';</code>	d is missing

7. Missing-value element

Places a content when a variable has not a specific value or is missing.

Syntax

```
{mis:variable:value} ... {/mis}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

```
if variable has not a value of value or is missing
    place the content of the element
else
    disappear
```

Remarks

{mis:variable:NULL} or {mis:variable:} don't work. Use {set:variable} instead.

This element accepts absolute and relative paths for *variable*.

value accepts simple strings without spaces, single and double quotes strings.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
{mis:a:3}a != 3{/mis}	<code>\$ets->a = 3;</code>	(none)
{mis:b:value}b != 'value'{/mis}	<code>\$ets->b = 'value';</code>	(none)
{mis:c:"val ue"}c != "val ue"{/mis}	<code>\$ets->c = 'value';</code>	c != "val ue"
{mis:d:4}d != 4{/mis}	<code>\$ets->d = NULL;</code>	d != 4

8. PHP element

Places a content which will be parsed as PHP code.

Syntax

```
{php} ... {/php}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Simple tags, PHP code

Rules

if the content is a valid PHP code and stands for a scalar value
 place the representation of this value
else if the content is produces a fatal error
 halt the script
else
 disappear

Remarks

PHP element should be used to format values only.

About simple tags in a PHP element:

- FALSE values and empty strings stand for scalar value,
- strings are placed with double quotes and are escaped to accept carriage returns and tabulations.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function. When the code produces a parse error or something else, the script doesn't halt but the PHP element disappears.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
{php}sprintf('%.2f', {price}){/php}	\$ets->price = 4.3;	4.30
{php}date('Y', {date}){/php}	\$ets->date = time();	2003
{php}strtolower({string}){/php}	\$ets->string = 'LOWER';	lower
{php}sprintf('%.2f', {price}){/php}	\$ets->price = NULL;	(none)
Before {php}missing_function({label}){/php} After	\$ets->label = 'abc';	(none)

9. Constant element

Places an outer template.

Syntax

```
{const:template}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	In template, PHP code, simple tags

Rules

```
place simple tags in template
if template is a valid PHP code that stands for a string value
    if the outer template template exists
        place its content
    else
        disappear
else if template produces a fatal error
    halt the script
else
    disappear
```

Remarks

This element doesn't change the current level.

This element is independent from the data tree.

About simple tags in template:

- FALSE values and empty strings stand for scalar value,
- strings are placed with double quotes and are escaped to accept carriage returns and tabulations.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{mask:main} {const:a} {/mask} {mask:a}Content of a{/mask}</pre>	<pre>\$ets->not_a = 1;</pre>	Content of a
<pre>{mask:main} {const:b} {/mask} {mask:not_b}Content{/mask}</pre>	<pre>\$ets->price = 3;</pre>	(none)

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{mask:main} {const:c} {/mask} {mask:c}c with {d}{/mask}</pre>	<pre>\$ets->d = 1;</pre>	<pre>c with 1</pre>

10. If element

Places a content if a test is TRUE.

Syntax

```
{if:test} ... {/if}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	In <code>test</code> , PHP code, simple tags In content, text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

place simple tags in `test`
if `test` is a valid PHP code that stands for the boolean value `TRUE`
 place its content
else if `test` produces a fatal error
 halt the script
else
 disappear

Remarks

About simple tags in an if element:

- `FALSE` values and empty strings stand for scalar value,
- strings are placed with double quotes and are escaped to accept carriage returns and tabulations.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<code>{if: {a} == 4 }a == 4{/if}</code>	<code>\$ets->a = 4;</code>	<code>a == 4</code>
<code>{if: {a} }a is TRUE{/if}</code>	<code>\$ets->a = TRUE;</code>	<code>a is TRUE</code>
<code>{if: {a} == 'ets' }ETS{/if}</code>	<code>\$ets->a = 'ets';</code>	<code>ETS</code>
<code>{if: {a} == 1 }a == 1{/if}</code>	<code>\$ets->not_a = 1;</code>	<code>(none)</code>
<code>{if: {a[1]/b} == 2}b == 2{/if}</code>	<code>\$ets->a[0]->b = 1;</code> <code>\$ets->a[1]->b = 2;</code>	<code>b == 2</code>

11. Choose element

Provides multiple conditions in conjunction with when-test and else elements.

Syntax

```
{choose} ... {/choose}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	when-test element, else element and comment

Rules

```
for each when-test child
    place simple tags in test
    if test is a valid PHP code that stands for the boolean value TRUE
        place its content
        exit
    else if test produces a fatal error
        halt the script
if an else child exists
    place its content
    exit
else
    disappear
```

Remarks

Choose element defines a container to organize a sequence of multiple tests parsed like if/elseif/.../else.

When-test children are evaluated in order from the top to the bottom until a test is TRUE.

Else element must be unique in each choose element but can be defined anywhere in the choose element.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{choose} {when:{a} < 2}singular{/when} {else}plural{/else} {/choose}</pre>	<pre>\$ets->a = 2;</pre>	plural
<pre>{choose} {when:{a} < 0}negative{/when} {when:{a} < 9}a < 9{/when} {else}a >= 9{/else} {/choose}</pre>	<pre>\$ets->a = 2;</pre>	a < 9

12. When-test element

Places a content if a test is TRUE so disables other when-test or else siblings.

Syntax

```
{when:test} ... {/when}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Choose element
<i>Child elements</i>	In test, PHP code, simple tags In content, text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

See choose element.

Remarks

About simple tags in an when-test element:

- FALSE values and empty strings stand for scalar value,
- strings are placed with double quotes and are escaped to accept carriage returns and tabulations.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{choose} {when:{a} < 0}negative{/when} {when:{a} < 9}a < 9{/when} {when:{a} < 99}9 < a < 99{/when} {else}a >= 99{/else} {/choose}</pre>	<pre>\$ets->a = 2;</pre>	a < 9
<pre>{choose} {when:{a} < 0}negative{/when} {else}nul{/else} {when:{a} > 0}positive{/when} {/choose}</pre>	<pre>\$ets->a = 2;</pre>	positive

13. Else element

Places a content if all when-test or when-value siblings are FALSE.

Syntax

```
{else} ... {/else}
```

Template tree

<i>Number of occurrences</i>	One per choose or choose-variable element
<i>Parent elements</i>	Choose or choose-variable element
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

See choose and choose-variable elements.

Remarks

Else element provides a default content for choose and choose-variable elements.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{choose} {when:{a} < 0}negative{/when} {when:{a} < 9}a < 9{/when} {when:{a} < 99}9 < a < 99{/when} {else}a >= 99{/else} {/choose}</pre>	<pre>\$ets->a = 2;</pre>	a < 9
<pre>{choose} {else}useless tags{/else} {/choose}</pre>	<pre>\$ets->a = 1;</pre>	useless tags

14. Choose-variable element

Provides multiple conditions in conjunction with when-value and else elements.

Syntax

```
{choose:variable} ... {/choose}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	when-value element, else element and comment

Rules

```
if the variable variable is scalar
  for each when-value child
    if the variable variable has the value of this element
      place its content
      exit
    if there an else child exists
      place its content
      exit
    else
      disappear
  else
    disappear
```

Remarks

Choose-variable element defines a container to organize a sequence of multiple tests parsed like switch/case/default.

When-value children are evaluated in order from the top to the bottom until a test is **TRUE**.

Else element must be unique in each choose-variable element but can be defined anywhere in the choose element.

This element accepts absolute and relative paths for `variable`.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{choose:partner/id} {when:1}partner #1{/when} {when:2}partner #2{/when} {when:3}partner #3{/when} {else}no partner{/else} {/choose}</pre>	<pre>\$ets->partner->id = 2;</pre>	partner #2
<pre>{choose:label} {when:'the fox'}1{/when} {when:'the dog'}2{/when} {else}0{/else} {/choose}</pre>	<pre>\$ets->label = 'the dog';</pre>	2

15. When-value element

Places a content if the variable of the choose parent has the value of this element so disables other when-value or else siblings.

Syntax

```
{when:value} ... {/when}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Choose-variable element
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

See choose-variable element.

Remarks

{when:NULL} or {when:} don't work. Use {mis:variable} out of choose-variable parent instead.

This element accepts absolute and relative paths for `variable`.

`value` accepts simple strings without spaces, single and double quotes strings.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{choose:partner/id} {when:1}partner #1{/when} {when:2}partner #2{/when} {when:3}partner #3{/when} {else}no partner{/else} {/choose}</pre>	<pre>\$ets->partner->id = 2;</pre>	partner #2
<pre>{choose:label} {when:'the fox'}1{/when} {when:'the dog'}2{/when} {else}0{/else} {/choose}</pre>	<pre>\$ets->label = 'the dog';</pre>	2

16. Call element

Places an outer template with arguments.

Syntax

```
{call:template} ... {/call}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	In <code>template</code> , PHP code, simple tags In content, argument element, comment

Rules

```
place simple tags in template
if template is a valid PHP code that stands for a string value
    if the outer template template exists
        place its content
        for each argument child of the call element
            add the built argument to the data tree
    else
        disappear
else if template produces a fatal error
    halt the script
else
    disappear
```

Remarks

Call element is parsed like a function call and works like constant element plus defines a container where to define arguments.

Call element doesn't change the current level.

Be careful when choosing the name of an argument to not overwrite existing data. Choose for example a prefix like `a_`.

About simple tags in `template`:

- `FALSE` values and empty strings stand for scalar value,
- strings are placed with double quotes and are escaped to accept carriage returns and tabulations.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Example

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>... {call:color}{arg:a_id}1{/arg}/{/call} {call:color}{arg:a_id}2{/arg}/{/call} {const:color} ... {/mask} {mask:color} {set:a_id:1}red{/set} {set:a_id:2}blue{/set} {mis:a_id}black{/mis} {/mask}</pre>	<pre>\$ets->a = 1;</pre>	<pre>red blue</pre>

17. Argument element

Defines an argument of a call element.

Syntax

```
{arg:name} ... {/arg}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Call element
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

See call element.

Remarks

Argument element is built then used as a scalar variable in the called outer template.

Argument element doesn't change the current level.

`{arg:name} {/arg}` will produce the same result than if it is not defined because it defines a empty string which stands for a missing variable.

Example

See call element.

18. Repeat element

Places a content several times.

Syntax

```
{repeat:n} ... {/repeat}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Text, comment, cdata and any element aside from argument, when-value, when-test and else elements

Rules

if *n* is a valid PHP code that stands for an integer value
 loop *n* times
 place the content
else if *n* produces a fatal error
 halt the script
else
 disappear

Remarks

When possible, ETS creates a system variable, `_count`, which contains the current loop.

If the number of loops is negative or zero, repeat element disappears.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Example

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{mask:graph} {repeat:{val} / 100}*{/repeat} {/mask}</pre>	<pre>\$ets->graph[0]->val = 300; \$ets->graph[1]->val = 100; \$ets->graph[2]->val = 600; \$ets->graph[3]->val = 150;</pre>	<pre>*** * ***** *</pre>
<pre>{repeat: {pages} } {_count}{if: {_count} < {pages} }, {/if} {/repeat}</pre>	<pre>\$ets->pages = 4;</pre>	<pre>1, 2, 3, 4</pre>

19. Include element

Includes outer templates from a container.

Syntax

```
{include:container}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements if inner None if outer
<i>Child elements</i>	In container, PHP code, simple tags

Rules

place simple tags in container

if `container` is a valid PHP code that stands for a string value

 include the content of `container`

else if `container` produces a fatal error

 halt the script

else

 disappear

Remarks

If inner, the content of the container is parsed only when needed; if outer, it's parsed immediately.

Include element doesn't place the content of the container where it's declared. It adds outer templates defined in the container to available outer templates list. Thus, it's an alternative management of container names as argument of `printt` or `sprintt`.

If the container is a file name, a relative path starts from the location of the PHP script that calls ETS.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Example

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{include:library.tpl} {mask:main} {set:page:search} {include:search.tpl} {/set} {body} {/mask}</pre>	<pre>\$ets->page = 'search';</pre>	<pre>... <h1>Search</h1>
 5 results found
 ...</pre>

20. Insert element

Places the content of a container as text.

Syntax

```
{insert:container}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	In container, PHP code, simple tags

Rules

```
place simple tags in container
if container is a valid PHP code that stands for a string value
    insert the content of container as text
else if container produces a fatal error
    halt the script
else
    disappear
```

Remarks

Insert element inserts the content of a container only when needed and where it's declared.

If the container is a file name, a relative path starts from the location of the PHP script that calls ETS.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{set:intro} {insert:'file.txt'} {/set}</pre>	<pre>\$ets->intro = FALSE;</pre>	(none)
<pre>{set:license} {insert:'lgpl.txt'} {/set}</pre>	<pre>\$ets->license = TRUE;</pre>	GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999 Copyright (C) 1991, 1999 Free Software Foundation, Inc. ...

21. Eval element

Places the content of a container as a template part.

Syntax

```
{eval:container}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	In container, PHP code, simple tags

Rules

```
place simple tags in container
if container is a valid PHP code that stands for a string value
    insert the content of container as template part
else if container produces a fatal error
    halt the script
else
    disappear
```

Remarks

Eval element works like include element because its content is parsed and works like insert element because its content is placed where it's declared.

Eval element inserts the content of a container only when needed.

If the container is a file name, a relative path starts from the location of the PHP script that calls ETS.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

Examples

<i>Template</i>	<i>PHP code</i>	<i>Result</i>
<pre>{set:intro} {eval:'file.ets'} {/set}</pre>	<pre>\$ets->intro = FALSE;</pre>	(none)
<pre>{set:license} {eval:'lgpl.ets'} {/set}</pre>	<pre>\$ets->license = TRUE; \$ets->title = 'ETS';</pre>	<pre>ETS ----- GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999 Copyright (C) 1991, 1999 Free Software Foundation, Inc. ...</pre>
lgpl.ets		
<pre> {title} ----- GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February</pre>		

22. Safe element

Places the content of a container as a template part with restrictions.

Syntax

```
{safe:container}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Text, comment, cdata, simple tags, set, set-value, missing, missing-value and template elements

Rules

```
place simple tags in container
if container is a valid PHP code that stands for a string value
    insert the content of container as template part
else if container produces a fatal error
    halt the script
else
    disappear
```

Remarks

Safe element works like eval element but only accepts, in the called container, elements which haven't a part that can be PHP code.

It can be used when a site contains public and private containers or when you can't make sure that some containers are well-formed.

ETS produces a notice message when safed containers contains unauthorized elements.

Warning: as indicated in rules, when the evaluated code produces a fatal error, the script halts. This means that you will get an empty result. It happens generally when you're calling a missing function.

23. Reduce element

Defines a size reducing behaviour.

Syntax

```
{reduce:value}
```

Template tree

<i>Number of occurrences</i>	One
<i>Parent elements</i>	None
<i>Child elements</i>	None

Rules

if value is OFF or NOTHING

 don't change the built document

else if value is SPACE or SPACES

 remove whitespaces from the beginning and end of each line of the built document

else if value is CRLF or ON or ALL

 remove whitespaces from the beginning and end of each line of the built document

 remove carriage returns of the built document

Remarks

This directive must be placed out of any template. If more than one reduce directive is specified, the last one will be used.

The default behaviour for size reducing is to do nothing.

Whitespaces stand for spaces and tabulations.

You can also use comments to suppress specific white spaces.

Whitespaces and carriage returns of cdata sections are not affected.

Examples

<i>Template</i>	<i>Result</i>
<pre>{reduce:nothing} {mask:main}{* *}x y {/mask}</pre>	<pre>x y</pre>
<pre>{reduce:spaces} {mask:main} x y {/mask}</pre>	<pre>x y</pre>
<pre>{reduce:all} {mask:main} x y {/mask}</pre>	<pre>xy</pre>

24. Comment

Defines a comment section.

Syntax

```
{ * ... * }
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from PHP element
<i>Child elements</i>	Ignored

Remarks

Comments are not part of the built template.

Comments can also be specified between outer templates: these parts of containers are ignored by ETS.

Comments can be nested.

Apart from a debug process, don't leave useless big comments in a container; it slows down ETS because all comments are read.

Examples

<i>Template</i>	<i>Result</i>
The root template: {mask:main} {const:ets} { * the main template { * nested but works too! * } * } {/mask} Another template: {mask:ets} ABC {/mask}	ABC
Active: {mask:main} A {/mask} Backup: { * {mask:main} B {/mask} * }	A

25. Cdata

Defines a section where the text is not parsed.

Syntax

```
{# ... #}
```

Template tree

<i>Number of occurrences</i>	Unlimited
<i>Parent elements</i>	Any element aside from call, choose, choose-variable and PHP elements
<i>Child elements</i>	Anything becomes text

Remarks

In some situations, for example, in a CSS section or a javascript section of a HTML document, you can safely use characters { and } if these sections are written between {# and #}. These characters define a cdata section. In other words, these sections will be considered by ETS as text.

Cdata can be nested.

To avoid parsing, you can also add a space, tab or carriage return after { . In this case, ETS doesn't generate an error and considers this content as text.

Spaces, tabs and end of lines are not affected by reduce directive in cdata.

Examples

<i>Template</i>	<i>Result</i>
{# div{color:blue;} #}	div{color:blue;}
div { color:blue; }	div { color:blue; }
div{color:blue;}	(parse error)

26. System variables

When a level is an element of an array, ETS creates system variables for this level which can be used as other ones.

These variables are:

<i>Name</i>	<i>Data type</i>	<i>Description</i>
<code>_key</code>	Mixed	Actual key of the element
<code>_index</code>	Integer	Numbers the element from 0
<code>_rank</code>	Integer	Numbers the element from 1
<code>_odd</code>	Boolean	TRUE if <code>_rank</code> is odd
<code>_even</code>	Boolean	TRUE if <code>_rank</code> is even
<code>_first</code>	Boolean	TRUE if the element is the first of the array
<code>_last</code>	Boolean	TRUE if the element is the last of the array
<code>_middle</code>	Boolean	TRUE if <code>_first</code> is FALSE and <code>_last</code> is FALSE
<code>_not_first</code>	Boolean	TRUE if <code>_first</code> is FALSE
<code>_not_middle</code>	Boolean	TRUE if <code>_middle</code> is FALSE
<code>_not_last</code>	Boolean	TRUE if <code>_last</code> is FALSE
<code>_not_even</code>	Boolean	TRUE if <code>_even</code> is FALSE
<code>_not_odd</code>	Boolean	TRUE if <code>_odd</code> is FALSE

Examples

This example shows how to alternate color of lines in a table and use a different color for the first line.

```
{mask:main}
  {set:line}
  <table>
    {mask:line}
    <tr>
      <td bgcolor="{const:color}">{id}</td>
      <td bgcolor="{const:color}">{label}</td>
    </tr>
    {/mask}
  </table>
  {/set}

  {mis:line}<b>No data</b>{/mis}
{/mask}

{mask:color}{choose}
  {when:{_first}}#00FF00{/when}
  {when:{_odd}}#FFFFFF{/when}
  {when:{_even}}#C0C0C0{/when}
{/choose}{/mask}
```

This example shows how to calculate the number of elements in an array and add a "s" if there is more than 1 element.

```
{mask:main}
  {mask:array}{set:_last}{_rank} element{const:s} in array{/set}{/mask}
{/mask}

{mask:s}{if: {_rank} > 1}s{/if}{/mask}
```

27. Path

In a path, ETS accepts different operators and names shown in the following table:

<i>Name</i>	<i>Description</i>
/	Child level in the data tree
//	Root level of the data tree
..	Level of the data tree used in the parent template
_parent	Alias of ..
_start	First element of the current array in the data tree
_previous	Previous element of the current array in the data tree
_next	Next element of the current array in the data tree
_end	Last element of the current array in the data tree

Example

Template	Data tree	Result view																																								
<pre>{mask:main} <table border="1"> {mask:b} <tr> <td>{_start/c}</td> <td>{_previous/_previous/c}</td> <td>{_previous/c}</td> <td>{c}</td> <td>{_next/c}</td> <td>{_next/_next/c}</td> <td>{_end/c}</td> <td>{../a}</td> </tr> </mask> </mask></pre>	<pre>\$ets->a = 'a'; \$ets->b[0]->c = 1; \$ets->b[1]->c = 2; \$ets->b[2]->c = 3; \$ets->b[3]->c = 4; \$ets->b[4]->c = 5;</pre>	<table><tr><td>1</td><td></td><td></td><td>1</td><td>2</td><td>3</td><td>5</td><td>a</td></tr><tr><td>1</td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>a</td></tr><tr><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>5</td><td>a</td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td><td>5</td><td>a</td></tr><tr><td>1</td><td>3</td><td>4</td><td>5</td><td></td><td></td><td>5</td><td>a</td></tr></table>	1			1	2	3	5	a	1		1	2	3	4	5	a	1	1	2	3	4	5	5	a	1	2	3	4	5		5	a	1	3	4	5			5	a
1			1	2	3	5	a																																			
1		1	2	3	4	5	a																																			
1	1	2	3	4	5	5	a																																			
1	2	3	4	5		5	a																																			
1	3	4	5			5	a																																			

28. Error handling

ETS checks PHP error level reporting before displaying a message.
ETS follows the 3 error levels of PHP.

Notice

ETS displays a message and continues to parse containers or match data and templates.

Notice messages are:

- container not found
- duplicated reduce element
- invalid value for reduce element
- datatree is not an array, an object or null (given as argument)
- wrong element in safe container

Warning

ETS displays a message and stops to parse the outer template which causes the message.

Warning message are:

- wrong element in another or at root
- unexpected closing tag
- unexpected character or space in tag
- end of comment or cdata not found
- closing tag not found
- duplicated templates
- else element duplicated

Error

ETS displays a message and halts the script

Error message are:

- entry mask not found
- containers are not array or string (given as argument)

29. User-level templates storage

If you don't want to use the standard way to retrieve template contents provided by ETS, you can set your own.

In this case, you must define a function named `ets_source_read_handler` which needs one argument, the id of the template container, and returns the content of the container or `FALSE` on error.

For example, if you store template containers in a MySQL database, this function could be:

```
function ets_source_read_handler($id)
{
    $content = FALSE;
    mysql_connect('myhost', 'myuser', 'mypassword');
    mysql_select_db('mydatabase');
    $id = mysql_escape_string($id);
    $h = mysql_query("select content from ets_source where id = '$id'");
    if ($o = mysql_fetch_object($h)) {
        $content = $o->content;
    }
    mysql_free_result($h);
    return $content;
}
```

When this function is defined, ETS uses it each time it needs to retrieve the content of a container; those given as argument of `print` and `sprintt` and those used by `include` or `insert` elements as well.

30. Cache system

In complement to the user-level templates storage, you can optionally store and re-use compiled template containers so ETS will not have to parse containers each time it uses them before playing with a data tree.

To define a complete cache system, you must define 2 functions in addition to `ets_source_read_handler`: `ets_cache_read_handler` and `ets_cache_write_handler`.

`ets_cache_read_handler` needs one argument, the id of the container, and returns the compiled container or `FALSE` when the compiled container is obsolete or on error.

`ets_cache_write_handler` needs two arguments, the id of the template container and the compiled content of the container, and doesn't return anything.

The cache system provided by ETS will automatically refresh compiled containers each time a source container is modified.

ETS never compiles containers which are used by insert elements because there are not template containers but only data containers.

Here is a complete set of functions which can be used to implement a file-based cache system:

```
// read a compiled container and check if it's not obsolete
function ets_cache_read_handler($id)
{
    $content = FALSE;
    if (@filemtime("ets/$id.ets") > @filemtime("tpl/$id.html")) {
        if ($handle = @fopen("ets/$id.ets", 'rb')) {
            $size = @filesize("ets/$id.ets");
            $content = @fread($handle, $size);
            fclose($handle);
        }
    }
    return $content;
}

// write a compiled container
function ets_cache_write_handler($id, $content)
{
    if ($handle = @fopen("ets/$id.ets", 'wb')) {
        @fwrite($handle, $content);
        fclose($handle);
    }
}

// read a source container
function ets_source_read_handler($id)
{
    $content = FALSE;
    if ($handle = @fopen("tpl/$id.html", 'rb')) {
        $size = @filesize("tpl/$id.html");
        $content = @fread($handle, $size);
        fclose($handle);
    }
    return $content;
}
```

In this example, source templates containers are stored in the directory `tpl` and compiled ones in `ets`.

Before using this cache system, you must fill the directory `tpl` with original templates files.

Here is a MySQL-based cache system:

```
// read a compiled container and check if it's not obsolete
function ets_cache_read_handler($id)
{
    $content = FALSE;
    mysql_connect('myhost', 'myuser', 'mypassword');
    mysql_select_db('mydatabase');
    $id = mysql_escape_string($id);
    $h = mysql_query("select c.content from ets_cache c inner join " .
        "ets_source s on c.id = s.id and s.date < c.date where c.id = '$id'");
    if ($o = mysql_fetch_object($h)) {
        $content = $o->content;
    }
    mysql_free_result($h);
    return $content;
}

// write a compiled container
function ets_cache_write_handler($id, $content)
{
    mysql_connect('myhost', 'myuser', 'mypassword');
    mysql_select_db('mydatabase');
    $id = mysql_escape_string($id);
    $content = mysql_escape_string($content);
    $h = mysql_query("select id from ets_cache where id = '$id'");
    $o = mysql_fetch_object($h);
    mysql_free_result($h);
    if ($o) {
        mysql_query("update ets_cache set content = '$content' where id = '$id'");
    } else {
        mysql_query("insert into ets_cache (id, content) values ('$id', '$content')");
    }
}

// read a source container
function ets_source_read_handler($id)
{
    $content = FALSE;
    mysql_connect('myhost', 'myuser', 'mypassword');
    mysql_select_db('mydatabase');
    $id = mysql_escape_string($id);
    $h = mysql_query("select content from ets_source where id = '$id'");
    if ($o = mysql_fetch_object($h)) {
        $content = $o->content;
    }
    mysql_free_result($h);
    return $content;
}
```

In this example, the table `ets_source` contains source containers and `ets_cache` compiled ones and have this structure:

```
CREATE TABLE ets_cache (
  id varchar(100) NOT NULL default '',
  date timestamp(14) NOT NULL,
  content text NOT NULL,
  PRIMARY KEY (id)
) TYPE=MyISAM;

CREATE TABLE ets_source (
  id varchar(100) NOT NULL default '',
  date timestamp(14) NOT NULL,
  content text NOT NULL,
  PRIMARY KEY (id)
) TYPE=MyISAM;
```

Before using this cache system, you must fill the table `ets_source` with original templates containers.

Here is a Turck-MMCache-based cache system.

According to Turck MMCache site (http://www.turcksoft.com/en/e_mmc.htm):

"Turck MMCache is a free open source PHP accelerator, optimizer, encoder and dynamic content cache for PHP."

"[Turck MMCache] has been tested under PHP 4.1.0-4.3.2 under Linux and Windows with Apache 1.3 and 2.0"

To use MMCache, you must install and configure it as explained on its site.

```
// read a compiled container and check if it's not obsolete
function ets_cache_read_handler($id)
{
    $content = FALSE;
    if (mmcache_get("T_$id") > @filemtime("tpl/$id.html")) {
        $content = mmcache_get("C_$id");
    }
    return $content;
}

// write a compiled container
function ets_cache_write_handler($id, $content)
{
    mmcache_put("T_$id", time());
    mmcache_put("C_$id", $content);
}

// read a source container
function ets_source_read_handler($id)
{
    $content = FALSE;
    if ($handle = @fopen("tpl/$id.html", 'rb')) {
        $size = @filesize("tpl/$id.html");
        $content = @fread($handle, $size);
        fclose($handle);
    }
    return $content;
}
```

In this example, source templates containers are stored in the directory `tpl` and compiled ones in memory.

31. Miscellaneous

Simplified closing tags

Every closing tag can be simplified with `{/}`:

`{/mask}` is equivalent to `{/}`,
`{/set}` is equivalent to `{/}`,
`{/when}` is equivalent to `{/}`,
and so on.

External presentation logic

In some situations, the presentation logic can be stored outside templates containers. In this case, ETS provides several ways to exploit it:

- you can use variable template names in alternate tag, constant and call elements,
- you can use variable container names in include, insert and eval elements.

This example shows a way to use the template name stored in the variable `tpl1` to present the level data:

```
{data:{tpl1}}
```

This example shows a way to use a directory and a file defined in the data tree to include a file:

```
{include: {/directory} . '/' . {filename} }
```

Functions mis and set

To check if data are missing or not, you can use two functions provided by ETS which mimic the behaviour of missing and set elements: `mis(value)` and `set(value)`.

These two examples are equivalents:

```
{mis:var1}{mis:var2}var1 and var2 are missing{/mis}{/mis}
```

```
{if: mis({var1}) && mis({var2})}var1 and var2 are missing{/if}
```

These functions are useful to check the state of two or more variables with the `||` operator of PHP. This example is harder to define with only `mis` elements:

```
{if: mis({var1}) or mis({var2})}var1 or var2 is missing{/if}
```

Escapes

For a value of `set-value`, `mis-value` or `when-value`:

- in a no-quote string, use `\}` to escape `}`
- in a single-quoted string, use `\'` to escape `'`
- in a double-quoted string, use `\"` to escape `"`

In the evaluated part of any element, use `\}` to escape `}`.

Version history

Version 3.05b (2003-12-12)

Spaces, tabs and ends of line are not affected by reduce directive in cdata sections.

Version 3.05a (2003-08-08)

Fixes and clean up

Version 3.05 (2003-07-23)

Accept empty containers (0 byte or without template)

Accept null containers (which can be several)

Check PHP error level before displaying an error

Go on parsing when a notice or a warning is found

Accept empty string as container name

Introduce safe element which works like eval but with restrictions (basically, no PHP code is evaluated in templates)

Version 3.04a (2003-04-25)

Fix a tiny bug (display of a notice message)

Version 3.04 (2003-03-31)

Modify alternate tag, const and call elements to accept string expressions in template name

Modify include, insert, eval to accept string expressions in container name

fix a warning message for stored reduce directive

introduce eval element which works like include but for mask parts

fix text stored in choose and call elements when using simplified closing tag

Version 3.03 (2003-03-12)

introduce complete but optional cache of parsed templates with user-level handler functions

introduce user-level handler function used to delegate reads of templates

introduce _start, _next, _previous, _end and _parent (../id is the same than _parent/id)

fix behaviour of parent element of a path (..)

introduce mis(variable) and set(variable) which works like respectively missing and set element and can be used in if elements

Version 3.02a (2003-03-03)

fix: blank page when no reduce directive

Version 3.02 (2003-02-28)

update documentation

introduce {insert:file_name} which places a file content but doesn't parse it

introduce {include:file_name} which adds outer templates of included file

fix colon character in value

introduce {repeat:n} which repeats its content n times

introduce simplified closing tag {/}

fix line number of error messages when there's more than 1 file

error message when more than 1 use of reduce element

error message when a file is missing

more strict syntax when parsing templates (so more error messages)

great cleaning of code

Version 3.01 (2003-02-19)

Introduce access to data of the parent template with .. in path
Introduce management of missing outer templates with alternate tags to avoid calls to a template with the same name (i.e. {variable:template} places the value of variable if template doesn't exist)
Suppress infinite loops of const elements detection because it's not conclusive in some contexts.

Version 3.00 (2003-02-01)

Introduce {call:foo} ... {/call} which works like {const:foo} with arguments
introduce {choose} ... {/choose} which works like if/elseif/.../else (cf. doc)
introduce {choose:foo} ... {/choose} which works like switch/case/default (cf. doc)
introduce {if: test } ... {/if} where test may contain tags (with simple, relative or absolute paths)
enhance debugging information on parsing error
introduce cdata blocks {# ... #} to avoid their parsing (can be nested too)
accept now nested comments
{var:foo} and {var:foo:bar} don't work any more, use {set:foo} and {set:foo:bar} instead
{val:foo} doesn't work any more, use {foo} instead
remove the ability to propagate the value of a variable (scope) (it's a construct limit) but introduce access to a specific element from the root (ex. {/foo/bar})
accept now outer template from variable (depending on the type of the variable)
introduce access to a specific element of an array (ex. {set:foo/bar[3]/foobar}) then introduce system index [_first] and [_last]
introduce call to an outer template with a different name than the corresponding data {variable:template}
really detect infinite loop in circular const
MASK and STRING properties of a data level doesn't work any more (this concept doesn't allow several inner templates corresponding to the same level in the datatree and puts presentation logic in the data tree)
introduce reduce directive but remove reduce parameter of printt (reduce behavior belongs to presentation logic)
introduce new system variable _middle, _not_first, not_middle, not_last
accept now non printable characters (spaces, tabs, carriage returns...) before } of every tag
accept now single or double quoted strings in the value part of elements like {set:foo:...} {mis:foo:...} and {when:foo}
accept now relative and absolute path (every data can be accessed from everywhere)
accept now multiple definition of inner templates matching with the same level in the datatree
Closing tags are simplified ({/mask} instead of {/mask:main})
ETS becomes "template driven" instead of "data driven" (ETS was completely rewritten)

Version 2.04a (2002-11-28)

fix: manage values which contain preg special character '/'

Version 2.04 (2002-11-25)

introduce mask for specific missing values of a variable
introduce mask for specific values of a variable
introduce scope of variables
introduce new system tags for arrays : _first and _last
accept now NULL as data tree
introduce new system tag for arrays : _key
change license (LGPL instead of GPL)
fix: when you reuse the same data tree with several templates, ETS kept old masks
introduce size reducing parameter
accept now variables in constant masks
reintroduce nested constant masks
introduce the prefix 'const:' for constant tags then remove verbose flag for all functions
clear out left tags because they correspond to missing variables
clear out missing level tags when the level is not missing
accept now constant mask nested in variable masks
accept now nested variable masks

Version 2.03 (private release only)

introduce system tags for arrays : _index, _odd, _even and _rank
php stuff at level step

Version 2.02 (2002-06-17)

introduce mask of missing variable {mis:varname} ... {/mis:varname}
introduce php mask {php} ... {/php}
introduce comment {* ... *}
store unfound masks (faster)
introduce mask of variable {var:varname} ... {/var:varname}
use single quote instead of double in preg expressions
the second parameter of printt can now be an array instead of a list in a string
change conditions to be format like "constant-operator-variable" (e.g. if (FALSE === \$test))
prevent infinite loops in constant masks process
modify error handler to produce errors according to PHP error settings
modify error handler output to be HTML'd

Version 2.01 (private release only)

add error handler
cache of masks becomes a global variable instead of a static variable of the function _ets_mask

Version 2.00 (2002-06-04)

masks and variables names follow the same rules than PHP variables
constant masks
use implicit masks assuming they had the same name than the corresponding object or array
main mask name is 'main'
sprintt now has only two parameters: the data tree and an optional comma separated files list
tags can't contain spaces
give up array notation, give up file masks
blocks become masks

Version 1.28 (private release only)

rearrange examples (cosmetic)
modify example templates to use new block tags
rearrange documentation
modify block tags because olders could produce non-well-formed templates (i.e, <ets:foo> becomes {block:foo})
complete header printt function

Version 1.27 (2002-05-17)

modify header
modify prefix of private functions
fix detection of nested blocks
fix unused placeholder deletion
add test.php and test.html in package

Version 1.26 (2002-05-01)

authorize non-printable characters before the closing character (>) of ets tags

License

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program

that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place

satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6.

Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing

the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License

incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Contact

Web site: <http://ets.sourceforge.net>

Email: phpets at hotmail dot com (indicate ETS: at the beginning of the subject to not be filtered)