

ERML-0.0.1

(as released with erlweb-0.0.1)

Erlang Markup Language is a tag based scripting language that is designed to reduce the learning curve and development time of websites based in Erlang while maintaining the power and flexibility of Erlang.

Each .erml file needs to be written in XHTML with the addition of ERML tags that will be discussed later in this document. The reason for this is to allow for the user of xmerl, the XML processor that is built into Erlang. The .erml files are parsed into an XML tree of Erlang data structures and the tree is then traversed to collect information and modify the structure of the document based on the specific ERML commands.

ERML uses the built in Erlang expression evaluation functions to evaluate variables and expressions. This allows for a great amount of flexibility in any expression of a conditional nature or one that sets variables in any way. While the majority of Erlang syntax is left in place there are a few characters that are not valid in an XHTML document. Specifically the ones that are handles different in ERML are current the greater then (>) and less then (<) signs. In order to make comparisons using these signs four reserved words have been created, they are:

lt which replaces less then (<)

lte which replaces less then or equal to (= <)

gt which replaces greater then (>)

gte which replaces greater then or (= >)

To display a variable anywhere in an ERML page you need only enclose it with a percent sign on either end. %test% would display the value of the variable **test**, where as %Test% would display as it is written.

ERML Tags

ermlset

The first and most basic ERML command is the ermlset command. This command lets you set variables that can be displayed in your ERML pages or used by later commands. ERML variable names are assumed to be all lowercase alpha-numeric characters and the underscore. ([a-z][0-9]_) All Erlang terms will be supported as variable data, but in the current version only integers, floats and strings (a list of characters and numbers) are known to work without side effects.

Example:

```
<ermlset test="This is a test variable"/>
```

This example would set a variable named **test** equal to the string "**This is a test variable**".

ermlif (ermlesleif) (ermlelse)

The ermlif, ermlelseif and ermlelse statements are the bulk of the decision making tags in the current version of ERML. The ermlif and ermlelseif statements take a required parameter called **condition** which will be evaluated for its value. If the value is either true or a number greater than zero the condition is processed, otherwise the next statement is evaluated until one is found to be true. If there are no true condition and an ermlelse statement is included the ermlesle statement will be processed, without an ermlelse statement a false condition will display nothing.

Example:

```
<ermlif condition="i == 2">
```

This proves that i equals 2


```
<ermlesleif condition="j == 3"/>
```

This proves that j equals 3


```
<ermlesleif condition="h lte 10"/>
```

This proves that h less than or equal to 10

<ermlse/>

Now we know that i is not 2, j is not 3 and h is greater then 10

</ermlif>

ermlloop

There are going to be at least four separate types of loops built into ERML, but at the current time only two of them are working. The index loop and the conditional loop.

Index Loop

The index loop is a basic counter, it require and index variable, a number to start from and number to end at and an optional step.

Example:

```
<ermlloop index="i" from="1" to="10" step="1">
```

```
We are on step number %i%<br/>
```

```
</ermlloop>
```

Conditional Loop

The conditional loop will continue to run until the condition no longer returns true.

Example:

```
<ermlset index="0"/>
```

```
<ermlloop condition="index lte 25">
```

Index is currently %index%


```
<ermlset index=index+2>
```

```
</ermlloop>
```

ermlininclude

The ermlininclude command take another file and inserts into content at the point where the command is placed. This is very handy when you are creating large files and want to reduce the amount of working code in each file or want to have reusable code.

```
<ermlininclude src="anotherfilename.erml"/>
```

Other Commands

There are a few other commands that are in the source code, but are not part of the language at this point. For the most part they are being used as place holders for other commands of a simple way to clean up the HTML file after it has been processed. Currently those comments are **ermlcommand** and **ermloutput**.