Velocity Template Language
Office of Operational Service (OOS)
04/24/2008

This document is only valid for TO-8 Release of AWIPSII. It will have to be updated as new information comes available and with subsequent releases. It should be considered for informational purposes only.
Velocity Template Language
About this Guide
References
Variables
Properties
Methods
Directives

#set - Establishes the value of a reference
#elseif/#else - Output conditional on truth of statements
#foreach - Loops through a list of objects
#include - Renders local file(s) that are not parsed by Velocity
#parse - Renders a local template that is parsed by Velocity
#stop - Stops the template engine
#evaluate - Dynamically evaluates a string or reference
#macro - Allows users to define a Velocimacro (VM), a repeated segment of a VTL template, as required

Comments
Single Line
Multi Line
About this Guide

This guide is the reference for the Velocity Template Language (VTL). For more information, please also refer to the Velocity User Guide.

References

Variables

Notation:

$ [ ! ][ { }[ a..z, A..Z ][ a..z, A..Z, 0..9, -, _ ][ } ]$

Examples:

- Normal notation: $testMessage$
- Silent notation: $!testMessage$
- Formal notation: ${testMessage}$

NOTE: When using normal notation, if the variable is NULL it will print the variable name. When using Silent notation, if the variable is NULL it will print an empty string: "".

Properties

Notation:

$ [ { }[ a..z, A..Z ][ a..z, A..Z, 0..9, -, _ ]*.[a..z, A..Z ][ a..z, A-Z, 0..9, -, _ ]*[ [ ] ]$

Examples:

- Regular Notation: $city.name$
- Formal Notation: ${city.name}$

Methods

Notation:

$ [ { }[ a..z, A..Z ][ a..z, A..Z, 0..9, -, _ ]*.[ a..z, A..Z ][ a..z, A..Z, 0..9, -, _ ]*( [ optional parameter list... ] ) [ } ]$
Examples:

- Regular Notation: `$dateUtil.formatLocal(${now},${timeFormat.header})`
- Formal Notation: 
  `$dateUtil.formatLocal(${now},({timeFormat.header}))`
- Regular Notation with Parameter List: `$page.setTitle("My Home Page")`

VTL Properties can be used as a shorthand notation for VTL Methods that take `get` and `set`. Either `$object.getMethod()` or `$object.setMethod()` can be abbreviated as `$object.Method`. It is generally preferable to use a Property when available. The main difference between Properties and Methods is that you can specify a parameter list to a Method.

**Directives**

### #set - Establishes the value of a reference

**Format:**

```
# [ {} set [ ] ] ( $ref = [ "", ]arg[ "," ] )
```

**Usage:**

- `$ref` - The LHS of the assignment must be a variable reference or a property reference.
- `arg` - The RHS of the assignment, `arg` is parsed if enclosed in double quotes, and not parsed if enclosed in single quotes. If the RHS evaluates to `null`, it is **not** assigned to the LHS.

**Examples:**

- Variable reference: `#set( $officeName = $officeshort )`
- String literal: `#set( $area.partOfArea= 'Jefferson County' )`
- Property reference: `#set($area.partOfArea = $.myArea.name )`
- Method reference: `#set( $area.partOfArea = $yourArea.getName($otherArea) )`
- Number literal: `#set( $city.count = 123 )`
- Range operator: `#set( $city.count = [1..3] )`
- Object array: `#set( $city.list = ["Washington", $myCity, "New York"] )`

The RHS can also be a simple arithmetic expression, such as:

- Addition: `#set( $value = $foo + 1 )`
- Subtraction: `#set( $value = $bar - 1 )`
- Multiplication: `#set( $value = $foo * $bar )`
- Division: `#set( $value = $foo / $bar )`
- Remainder: `#set( $value = $foo % $bar )`
#if/#elseif/#else - Output conditional on truth of statements

Format:

```
#else [ ] [ ] ( [condition] ) [output] [ # [ ] elseif [ ] ] ( [condition] ) [output] * [ # [ ] else [ ] ] [output] ] [ # [ ] end [ ] ]
```

Usage:

- **condition** - If a boolean, considered true if it has a true false; if not a boolean, considered true if not null.
- **output** - May contain VTL.

Examples (showing different operators):

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Symbol</th>
<th>Alternative Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals Number</td>
<td>==</td>
<td>eq</td>
<td>#if( $foo == 42 )</td>
</tr>
<tr>
<td>Equals String</td>
<td>==</td>
<td>eq</td>
<td>#if( $foo == &quot;bar&quot; )</td>
</tr>
<tr>
<td>Object Equivalence</td>
<td>!=</td>
<td>eq</td>
<td>#if( $foo == $bar )</td>
</tr>
<tr>
<td>Not Equals</td>
<td>!=</td>
<td>ne</td>
<td>#if( $foo != $bar )</td>
</tr>
<tr>
<td>Greater Than</td>
<td>&gt;</td>
<td>gt</td>
<td>#if( $foo &gt; 42 )</td>
</tr>
<tr>
<td>Less Than</td>
<td>&lt;</td>
<td>lt</td>
<td>#if( $foo &lt; 42 )</td>
</tr>
<tr>
<td>Greater Than or Equal To</td>
<td>&gt;=</td>
<td>ge</td>
<td>#if( $foo &gt;= 42 )</td>
</tr>
<tr>
<td>Less Than or Equal To</td>
<td>&lt;=</td>
<td>le</td>
<td>#if( $foo &lt;= 42 )</td>
</tr>
<tr>
<td>Boolean NOT</td>
<td>!</td>
<td>not</td>
<td>#if( !$foo )</td>
</tr>
</tbody>
</table>

Notes:

1. The `==` operator can be used to compare numbers, strings, objects of the same class, or objects of different classes. In the last case (when objects are of different classes), the `toString()` method is called on each object and the resulting Strings are compared.

2. You can also use brackets to delimit directives. This is especially useful when text immediately follows an `#else` directive.

```
#if( $foo == $bar)it's true!#{else}it's not!#{end</li>
```

#foreach - Loops through a list of objects

Format:

```
# [ ] foreach [ ] ( $ref in arg ) statement # [ ] end [ ]
```

Usage:
• $ref - The first variable reference is the item.
• arg - May be one of the following: a reference to a list (i.e. object array, collection, or map), an array list, or the range operator.
• statement - What is output each time Velocity finds a valid item in the list denoted above as arg. This output is any valid VTL and is rendered each iteration of the loop.

Examples of the #foreach(), omitting the statement block:

- Reference: #foreach ( $item in $items )
- Array list: #foreach ( $item in ["Not", $my, "fault"] )
- Range operator: #foreach ( $item in [1..3] )

Velocity provides an easy way to get the loop counter so that you can do something like the following:

For the following cities: ...
#foreach( $city in $cityList )
  CityList.name
#end

The default name for the loop counter variable reference, which is specified in the velocity.properties file, is $velocityCount. By default the counter starts at 1, but this can be set to either 0 or 1 in the velocity.properties file. Here's what the loop counter properties section of the velocity.properties file appears:

# Default name of the loop counter variable reference.
directive.foreach.counter.name = velocityCount

# Default starting value of the loop counter variable reference.
directive.foreach.counter.initial.value = 1

Additionally, the maximum allowed number of loop iterations can be controlled engine-wide (an ability introduced in Velocity 1.5). By default, there is no limit:

# The maximum allowed number of loops.
directive.foreach.maxloops = -1

#include - Renders local file(s) that are not parsed by Velocity

Format:

# [ {} ] include [ ] ( arg[ arg2 ... argn] )

• arg - Refers to a valid file under TEMPLATE_ROOT.
Examples:

- String: `#include( "disclaimer.txt" "wfoInfo.txt" )`
- Variable: `#include( $foo $bar )`

#parse - Renders a local template that is parsed by Velocity

Format:

```
# [] parse [ ] ( arg )
```

- `arg` - Refers to a template under TEMPLATE_ROOT.

Examples:

- String: `#parse( "tornado.vm" )`
- Variable: `#parse( $foo )`

Recursion permitted. See `parse_directive.maxdepth` in `velocity.properties` to change from parse depth. (The default parse depth is 10.)

#stop - Stops the template engine

Format:

```
# [] stop [ ]
```

Usage:

This will stop execution of the current template. This is good for debugging a template.

#evaluate - Dynamically evaluates a string or reference

Format:

```
# [] evaluate [ ] ( arg )
```

- `arg` - String literal or reference to be dynamically evaluated.

Examples:

- String: `#evaluate( 'string with VTL #if(true)will be displayed#end' )`
- Variable: `#include( $foo )`

#macro - Allows users to define a Velocimacro (VM), a repeated segment of a VTL template, as required
Format:

```velocimacro
# [ { ] macro [ } ] ( vmname $arg1 [ $arg2 $arg3 ... $argn ] ) [ VM VTL code... ] # [ { ]
#end [ } ]
```

- `vmname` - Name used to call the VM (`#vmname`)
- `$arg1 $arg2 [ ... ]` - Arguments to the VM. There can be any number of arguments, but the number used at invocation must match the number specified in the definition.
- `[ VM VTL code... ]` - Any valid VTL code, anything you can put into a template, can be put into a VM.

Once defined, the VM is used like any other VTL directive in a template.

```velocimacro
#vmname( $arg1 $arg2 )
```

VMs can be defined in one of two places:

1. **Template library:** can be either VMs pre-packaged with Velocity or custom-made, user-defined, site-specific VMs; available from any template
2. **Inline:** found in regular templates, only usable when `velocimacro.permissions.allowInline=true` in `velocity.properties`.

**Comments**

Comments are not rendered at runtime.

**Single Line**

Example:

```velocimacro
## This is a comment.
```

**Multi Line**

Example:

```velocimacro
#*
This is a multiline comment.
This is the second line
*#
```

The original / updated version of this file is located at: 