# #if, #elseif, and #else statement

The **#if** directive allows you to include text when generating a document, on condition that the if-statement is true, for example:

```
#if ($condition)
 Hello World
 #end
```

The variable: ***$condition*** is evaluated to determine whether it is true, which will happen under certain circumstances:

1. ***$condition*** is a Boolean (true/false) that has a true value
2. ***$condition*** is not 'null'.
3. ***$condition*** is a comparison that is evaluated and returns 'true'.
4. ***$condition*** is a compound logical statement that is evaluated and returns 'true'.

If **$condition** returns 'true', the content between the ***#if*** and ***#end*** statements becomes the output. In this case, if ***$condition*** is true, the output will be: "Hello World". Conversely, if ***$condition*** returns 'false', there will be no output.

The ***#elseif*** or ***#else*** element can be used with the ***#if*** element. Note that Velocity Engine will stop at the first expression that is found to be true. The following example shows you how to add **#elseif** and **#else** to the **#if** statement:

```
#if( $condition )
 Content 1
 #elseif( $condition2 )
 Content 2
 #elseif( $condition3 )
 Content 3
 #else
 Content 4
 #end
```

From the above example, let us assume that ***$condition1*** is false, ***$condition2*** is true, and ***$condition3*** is true. The output for this conditional block will be ***Content 2*** because ***$condition2*** comes before ***$condition3*** even though both of them are true.
**Comparing Values and Logical Operators**

So far, *$condition* in the #if directive is assumed to be a Boolean value, however just like Java, Velocity supports the syntax to compare (greater than (>), less than (<), and is equal to (==)) two variables and the Logical Operators logical AND (&&), logical OR (||), and logical NOT (!) which will return a Boolean value.

In Velocity, the equivalent operator ("==") can be used to compare string, value, and objects. Note that the semantics of the equivalent operator ("==") are slightly different than those of Java where the equivalent operator ("==") can only be used to test object equality. In Velocity the equivalent operator ("==") can be used to directly compare numbers, strings, or objects. When comparing two objects with different classes, the string that represents the objects is compared. The following example compares two variables to see whether or not they are equal:

```
#set ($var1 = "cat")
#set($var2 = "dog")
#if ($var1 == $var2)
Var1 equals Var2.
#else
Var1 does not equal Var2
#end
```

The comparison operators ("<" and ">") is the same as the ones used in Java. The following example shows how to use the comparison operators to compare two values in which the **#if** statement will be evaluated 'true' and "**Var1 is less than Var2**" will be printed out in the generated report:

```
#set ( $var1 = 6 )
#set ( $var2 = 7 )
#if ( $var1 < $var2 )
Var 1 is less than Var2
#end
```

The logical AND, in Velocity, is represented by && and it must have at least two arguments. To write a conditional statement with the logical AND type, for example:

```
#if( $var1 < $var3 && $var3 > $var2 )
 Content 1
 #end
```

To better understand the above example, let us assume that **$var1 < $var3** is **argument 1** and **$var3 > $var2** is **argument 2** (**argument 1** and **argument 2** will be evaluated separately and will return true/false). The ***#if()*** directive will be evaluated 'true' if both **argument 1** and **argument 2** are true. If **argument 1** is false, the expression will be evaluated false and **argument 2** will not be evaluated.

If **argument 1** is true, Velocity Engine will then check the value of **argument 2**. If the value is true, the entire expression is true and *Content 1* becomes the output. If the value is false then there will be no output as the entire expression is false.

Logical OR operators work the same way as Logical AND, except for one of the references that needs to be evaluated 'true' for the entire expression to be considered true, for example:

| | |
|---|---|
| #set ($var1 = 1)<br>#set ($var2 = 2)<br>#set ($var3 = 3)<br>#if( $var1 < $var3 | **$var3 < $var2<br>)<br>Content 1<br>#end** |

To better understand the above example, let us assume that $var1 < $var3 is argument 1 and $var3 < $var2 is argument 2. According to the example, argument 2 is false because $var2 is less than $var3 not the other way around. Since argument 1 is true, Velocity Engine does not need to look at argument 2, whether argument 2 is true or false, the expression will be true, and Content 1 will be the output. Basically, a Logical OR operator requires only one argument to be true (either argument 1 or argument 2) to make an expression true. In the case that both argument 1 and argument 2 are false, there will be no output because the expression is false.

With Logical NOT operators, there is only one argument:

```
#set ( $bool = false )
 #if( !$bool )
 Content 1
 #end
```

As shown in the above example, ***!$bool*** is evaluated true and Content 1 is the output. But if ***$bool*** is true, then ***!*$bool** is false and there will be no output.