VelocityWhitespaceTruncatedByLineComment

Whitespace is an issue that still needs some resolution.

The problem is that sometimes your template text will include extra whitespace, including newlines.

The is because Velocity will include in the output all the whitespace after your text up to the next directive. Whitespace between directives (excluding newlines) will be including in the output.

This forces your to make your macro or template compact to fix the whitespace layout. The problem with this is that it makes your template unreadable. For example:

```
#macro( testMacro ) #if ( <some condition> ) Compacted text#else But its too hard to read#end #end
```

What we would like to write is:

```
#macro( testMacro )
    #if ( <some condition> )
This text here will include the four space prior to the "if" directive, as well as the whitespace up to the
"else" which includes a newline character.
    #else
So while this text is more readable the template output isn't what we want.
    #end
#end
```

So while VelocityWhitespaceGobbling is being resolved you can use this small workaround.

We need a way to tell Velocity that our text to be included stops here and not to include any more whitespace from this point to the next directive.

But what directive could we use?

Well the line comment "##" is a recognized Velocity directive that starts a comment that lasts until the end of the line. So if we mark the end of our text with the line comment directive hopefully we can tell Velocity to stop including text in the output **AND** get readable templates. Luckily this appears to work.

We still have the problem that we can't indent our macro code, but honestly if you are writing complicated macro code you should have put it into the java code instead and accessed the results from the template.

For example:

```
#macro( testMacro )
#if ( <some condition> )
This output will not have any leading whitespace, nor trailing
whitespace because the line is terminated via the line comment##
#else
Without the line comment, this line has an extra new line which wrecks
your template.
#end
#end
```

While I don't have a work-around for the macro indentation you can make it more readable by avoiding nesting if/else and instead invoking another macro.

For example:

```
#macro ( testMacro )
#if ( <some condition> )
#avoidNestingInIfs
#else
#avoidNestingInElses
#end
#end

#macro ( avoidNestingInIfs )
#if ( <other condition> )
#end
#end

#macro ( avoidNestingInElses )
...
#end
```

Note: while you can't indent at the start of the line, you can indent between #if and "(" as well as after your #else/#end statements.

```
#macro( testMacro )
#if          ( <some condition> )
This output will not have any leading whitespace, nor trailing
whitespace because the line is terminated via the line comment##
#else##      ( <some condition> )
Without the line comment, this line has an extra new line which wrecks
your template.
#end##       ( <some condition> )
#end##  testMacro
```