VelocityWhitespaceGobbleSetindentDirective

```
March 9, 2007 - revised description of proposed #setindent directive.
Based on comments in mailing list, the directive would specify an amount of indentation.
```

For the most part, I want my templates to be readable and I use indenting to accomplish that. The indenting of the template has nothing to do with the desired output. For example, a simple template to generate a Java class might look like this:

```
import com.myco.MyClass;
public Class AnotherClass {
#foreach ($t in $types)
    #if ($t.name.contains("_")
        Integer $t.name = 0;
        #if ($t.type == "static")
            // ignore static type $t.name
        #end
    #end
#end
}
```

This template should produce something like:

```
import com.myco.MyClass;
public Class AnotherClass {
    Integer v_1 = 0;
    Integer v_2 = 0;
    // ignore static type s_0
}
```

Note that the indentation of the Integer statements and the // ignore .. comment are the same even though they have different amounts of leading whitespace in the template.

Given that each line of the nested blocks within template are indented further to improve readability, there is no good way for Velocity to know what I really want it to do. There would need to be some marker in the template to indicate where indentation should be set for subsequent lines.

The same template can be modified slightly to indicate how to indent lines.

```
import com.myco.MyClass;
public Class AnotherClass {
    #set ($indent = 4)
    #setindent($indent)              ## a new setindent directive
    #foreach ($t in $types)
        #if ($t.name.contains("_")
            Integer $t.name = 0;
            #if ($t.type == "static")
                // ignore static type $t.name
            #end
            void set${t.name}(Integer v) {
            #set ($indent = $indent + 4)
            #setindent($indent)
                // some code within the method block
            #set ($indent = $indent - 4)
            #setindent($indent)
            }
        #end
    #end
}
```

In this example, a #setindent directive specifies the amount of whitespace to inject at the start of each line of generated text and the template is managing a variable ($indent) to keep track of the indentation amount. This strategy assumes that all leading whitespace would be ignored, and indentation would be controlled exclusively by the #setindent() directives.

This approach requires the template to manage an indentation variable ($indent) and execute #setindent directives. *The template is pretty messy if there is not much text being generated.* It would be nice if #setindent managed an internal variable and allowed the template to adjust the variable using absolute, increment and decrement values:

```
#setindent(4)  ## set indentation at 4 spaces
#setindent(+4) ## increment indentation by 4 spaces
#setindent(-4) ## decrement indentation by 4 spaces
```

This would result in a template like this:

```
import com.myco.MyClass;
public Class AnotherClass {
    #setindent(4)              ## set indentation at 4 spaces
    #foreach ($t in $types)
        #if ($t.name.contains("_")
            Integer $t.name = 0;
            #if ($t.type == "static")
                // ignore static type $t.name
            #end
            void set${t.name}(Integer v) {
                #setindent(+4)   ## add 4 spaces to current indentation
                // some code within the method block
                #setindent(-4)   ## remove 4 spaces from current indentation
            }
        #end
    #end
}
```

I realize that whitespace processing is much more complicated that indentation processing alone. This proposal only addresses how velocity could be enhanced to control indentation of generated text. Other approaches can be reviewed at VelocityWhitespaceGobbling.