VelocityVersusWebMacro

Differences: Velocity vs. WebMacro

jon@latchkey.com

geirm@optonline.net

On the mailing lists, we often get asked to explain the differences between Velocity and WebMacro. Instead of answering the same question over and over, we have started this document to describe some of the differences in our approaches. In reality, both are excellent tools and we feel that it is up to the user base to decide which one they would like to use.

The language syntax is very close but not exactly the same. The syntaxes were kept close for a few reasons:

1) The syntactical approach of $<foo> for references and
#<bar> for directives is a good one! It's easy for designers to work with, people understand it, and there was no reason to reinvent the wheel.

2) Since so many people were using the syntax, why invent another one? Also, it makes the WebMacro to Velocity conversion much easier 🙂

Velocity was built new from the ground up. There isn't a snippet of shared code. This was originally done because WebMacro was released under a license (GPL) that is not compatible with the BSD/ASF license and we (the original authors) needed a solution that was not under a GPL license. Since then, WebMacro has been released under a dual GPL/ASF license, however, that was too late, this project was already well underway and we also felt that we have a better technical solution by going with a generated parser instead of a hand coded one.

By starting new and fresh, we are less encumbered with some of the baggage that WebMacro must deal with due to backwards compatibility issues. We have fixed a few of the template language inconsistencies that we saw creeping into WebMacro such as the fact that we do not support using {} for code blocks (because we do not need to) and that we use () around all #directives on the right hand side. Examples:

```
#set ($bar = "foo")

#if ($bar.length() > 0)
stuff here
#end
```

Architecturally, the biggest difference until recently was the parser. The Velocity parser is written using a tool called javacc, a very popular Java-based parser generator. It was originally developed for Sun by Metamata and is now maintained and managed by Metamata as well : http://www.metamata.com/.

The WebMacro parser was written by hand until recently and was the cause of much pain when using WebMacro (due to many bugs in it) - their new parser, for their 0.95 release is also written using javacc and promises to be better (however, support for a lot of the baggage still exists).

Velocity is very simple in its architecture, as it is meant to be nothing more than a template engine. The idea is to do one thing only and do it well. Much in the spirit of the Unix philosophy - have a good set of simple tools, and create bigger things using them. This has enabled Velocity to be used as the basis for several different tools, such as Anakia and Texen.

Velocity is recognized that it isn't going to try to be a 'development framework' - there are plenty around, such as Turbine

Therefore, all facilities like 'context tool' and 'resource brokers' are purposely left out of Velocity, acknowledging that not only will frameworks do it, they will do it better anyway. Further, by leaving out of the Velocity core much that should be done at the application / user
/ framework level, the Velocity core remains smaller and therefore, in our opinion, more maintainable and extensible.

Internally, Velocity is pretty straightforward. There are no threads created or managed for you. Templates are parsed and caches, kept in their internal form of an 'Abstract Syntax Tree', a tree structure that is then rapidly walked when your template is merged. Each node corresponds to on of the syntactical elements in your template.

Internally within WebMacro, many of the methods and classes are marked as private and/or final in order to achieve better speed. However, this has prevented people from easily extending or overriding some of the core funtionality and re-using some of the code. We would rather sacrifice a few ms in favor of allowing more people to use our software easily.

The biggest feature difference between the two are the Velocimacros. Velocimacros can be thought of as snippets of re-usable data that can contain anything. They are parsed just like the data in a #parse() statement would be and you can pass in variables that can optionally have scope only within the Velocimacro. Velocimacro's help keep the core #directive set small and clean : you can extend the designers toolkit with libraries of Velocimacros rather than adding elements to the syntax or having two write tools that generate HTML (or other specific) markup.

We hope this helps. The best way to understand Velocity is to use it, and of course nothing beats looking at the source code.