LoggingLevel

This is the project guideline for when the various log levels are appropriate/useful in Velocity classes. The aim here is keep Velocity log messages from becoming annoying, from hurting runtime performance, and especially to keep them as useful as possible.

Base Rules

- If you are logging below ERROR level, then you should wrap the log call and creation of the relevant message in an if(log.is[Level]Enabled()) block.
- If logging at ERROR level but not throwing an exception, consider wrapping things in an is[Level]Enabled() check.
- Try to keep messages and informative and non-generalized as possible

When Each Level Is Appropriate

TRACE

This is the level for things that may be useful to know when debugging *Velocity itself*. These are unlikely to be useful to anyone but those working to improve Velocity.

DEBUG

This level is appropriate for information that would be used during development (aka debugging) of an application *using Velocity*. This includes all configuration settings (whether defaults or not) at runtime initialization, engine state changes such as the addition of a new !Velocimacro, unhandled invalid references, resource loader activity, etc. This is not a level for things that would just be noise to a Velocity user.

INFO

This is the least useful level for a component library like Velocity. It is best limited to things of interest to a Velocity user/app during production, which is almost nothing. If you really want Velocity to make any production runtime info available, consider using something like JMX, not the logging API.

WARN

Again, not very useful for us. This is legitimate for when the Velocity user is doing something that is likely unwise but not a clear error. Don't give in to the temptation to use it when Velocity recovers from an minor error, when the user tries to do something forbidden by the configuration, or when there has been a misconfiguration, all of those things are errors.

ERROR

This is for when something goes wrong (whether Velocity recovers from it or not), when the user tries to do something not allowed by the configuration, and when there is a mis-configuration. It generally isn't be Velocity's job to clean up such messes, but work hard to to point out what went wrong and where as clearly as you can.

FATAL

Not to be used. Anything fatal to Velocity itself should just be logged as an error and passed on for the app, component or framework using Velocity to handle. They can decide whether it's fatal to the app or not.