Velocity15ReleaseNotes

Release Notes and Changes for the Velocity 1.5 release

This release is a drop-in replacement for earlier versions of Velocity. No changes to your application should be required. The only exception is that several of the dependent JAR libraries have been upgraded and a requirement for Commons Lang has been added. See the section "Dependency Notes" for more details.

You may also want to consult the Velocity RoadMap for details on future direction.

Bug Fixes

There have been many, many bugs fixed. Check the Change Log and JIRA for details Some of the most important:

- DatasourceResourceLoader contained a bug that could have been exploited by SQL injection.
- Velocity can now be compiled with JDK 1.5 due to removal of "enum" variable.
- Memory leaks in DatasourceResourceLoader have been fixed.
- #stop now works properly.
- Velocity singleton and engine will attempt to auto-init() when reasonable and possible instead of throwing NPEs when used without calling init() first.
- Anakia generates consistent line endings on Windows platforms.
- Fixed error in which Velocity would complain if the template ended with "##".
- Fixed race condition in template retrieval that caused macros to fail under simultaneous load.
- Methods were not being properly cached due to use of inconsistent keys.

VTL Enhancements

The Velocity Template Language (VTL) has been improved to be easier to use. Significant enhancements include new data types, new operators, improvements to the behavior of "==", and minor syntax improvements.

- VTL now has complete support for decimals and all number types. Decimals can be used in mathematical expressions, compared, and set with #set.
- VTL now has syntax for creating maps. This new syntax ~~{ "foo" : $bar, $key : 'value' }~~ may be used in the same places that the list creation syntax was valid.
- new VTL string concatenation operator "+".
- When comparing objects with "==" will compare String representations if objects are of different classes.
- Line breaks are allowed in directive calls, macro calls, method calls, and string literals.
- Macros now accept spaces or commas in separating arguments.
- Directives can now be delimited with curly braces, for example #if($condition)something#{else}otherthing#{end}.

Velocity Enhancements

There are many big and small improvements to the functionality of the Velocity engine. Many of these improvements are aimed at improving error reporting, simplifying integration into larger applications, and simplifying configuration.

- Much improved logging facilities, making Velocity a better citizen when integrated into other projects
- The log messages have been cleaned up.
- New support for JDK 1.4 logging is included.
- The logging facilities no longer panic and fail when no logger can be found. Instead they fall back to System.out and System.err for log output.
- The error messages now include template name, line and column number.
- New IncludeEventHandler allows modification of #parse and #include behavior.
- A number of specific event handler implementations are included, including several that escape HTML/XML entities.
- A URLResourceLoader is included for loading templates right off the web.
- Anakia can now be loaded with custom content from XML file.
- #set statements can be used to set a reference to null. Similarly, #foreach will properly iterate through a collection containing one or more nulls.
- You can now deploy the Velocity jar file in the container classpath while keeping your application in a webapp classpath.
- FileResourceLoader now accepts absolute paths (when configured to accept them).
- The default resource cache implementation now has an upper bound on cache size.
- New StringResourceLoader can retrieve templates from repository of in-memory Strings.
- MethodInvocationException now contains line, column, template name allowing application to produce more useful error messages.
- Change the meaning of localscope for macros to allow access to references from calling context.
- New event handler InvalidReferenceHandler allows application to catch invalid references. Sample implementation collects them in list and optionally throws exception.
- New, optional SecureIntrospector prohibits methods that involve manipulation of classes, classloaders or reflection objects. Use this introspector to secure Velocity against a risk of template writers using reflection to perform malicious acts.

Dependency Notes

If you are upgrading Velocity in an existing application, be sure to check that you have the copies of the correct versions of dependendent jar files.

- Requires JRE 1.3 to run, JDK 1.4 to compile.
- Upgraded to Commons Collection 3.1
- Upgraded to JDom 1.0

- Added Commons Lang 2.1 dependency
- Requires JavaCC 3.2+ to generate new syntax files

Known Issues

There are a few known bugs. See JIRA for details.

- In order to run "ant test" with ant 1.6.x, you will need to copy the junit.jar into <ANT_HOME>/lib. Otherwise, please upgrade to ant 1.7 (which accesses the junit jar from within the Velocity classpath).
- Macros cannot be defined in one file and included with #parse in another. See MacroIssues for more information.
- If you call VelocityEngine.evaluate() (or use the RenderTool) from within a macro and a #foreach loop, you will not be able to access the loop variable. See Velocity-285.
- If your references used shorthand notation (e.g. "$foo") and were followed immediately by a "(" character, they may have slipped through a hole in the parser which was closed in Velocity 1.5, causing them to throw parse exceptions. See Velocity-528. Using formal notation (e.g. "${foo}") will resolve this.
- There remain various subtle escaping bugs

Other Notes

- The documentation has been significantly improved, with new articles on getting started and on building webapps.
- VelocityServlet is now officially deprecated. Please use VelocityViewServlet from the VelocityTools subproject instead.
- VelocityFormatter is also deprecated. Please use the various formatting tools in the VelocityTools subproject.
- An optional Maven build is now available. However, ant remains the primary build tool.