VelocityWhitespaceGobbleStructuredTemplates

Arguments in favour of the Structured Whitespace Handling

The typical use scenario for velocity is to add VTL markup to text pages. Two forms can be differentiated:

- a velocity directive on a line by itself
- directives embedded within other text.

In the first case, it is desireable to allow indenting of the markup to better visualize the constructs; OTOH, the rendered output should not contain the whitespace indentation at the line beginning and the linefeed at the end. Example:

```
#foreach( $i in [1..3] )
  $i
  #if( $i > 1 )
  --
  #end
#end
```

For the second case, with embedded directives in a line, surrounding whitespaces should **not** be gobbled. A typical example of this from is:

```
You selected:#if( $value ) $value#else nothing#end/*
 * Copyright 2000-2004 The Apache Software Foundation.
 *
 * Licensed under the Apache License, Version 2.0 (the "License")
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
```

This proposal does not deal/affect VTL-references.

Very rare cases exist where VTL-markup is embedded in non-line based files (binary?). For such uses, it is desireable to allow switching off the whitespace gobbling as a whole. A nice solution would be to add a pragma directive:

```
#pragma( whitespace-gobbling off )
```

This should turn the gobbling off only within the declaring template. It should not affect any outer template parsing a template declaring the whitespace handling pragma.

The implementation of this proposal implies that the velocity parser must be aware of whitespaces before and after directives, and that during rendering these are handled according to the whitepace handling pragma switch.

Early filtering implementation

Here is an implementation that uses a java.io.FilterInputStream to filter unwanted whitespaces while loading the resource: StructuredGlobbingResourceLoader