

StructuredGlobbingResourceLoader

This is a structured whitespace globbing implementation that uses an early whitespace filtering while the resource is loaded.

This version inherits from `org.apache.velocity.tools.view.WebappResourceLoader`, but one could easily craft a version inheriting from `org.apache.velocity.runtime.resource.loader.FileResourceLoader`.

To use it, you would typically put the following lines in your `velocity.properties`:

```
resource.loader = globbing,string
globbing.resource.loader.class = StructuredGlobbingResourceLoader
string.resource.loader.class = org.apache.velocity.runtime.resource.loader.StringResourceLoader
```

StructuredGlobbingResourceLoader.java

```
/*
 * 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.
 */
package org.apache.velocity.tools.view;

import java.io.FilterInputStream;
import java.io.InputStream;
import java.io.IOException;

public class StructuredGlobbingResourceLoader extends WebappResourceLoader
{
    public static class VTLIndentationGlobber extends FilterInputStream
    {
        protected String buffer = "";
        protected int bufpos = 0;
        protected enum State
        {
            defstate, hash, comment, directive, schmoo, eol, eof
        }
        protected State state = State.defstate;

        public VTLIndentationGlobber(InputStream is)
        {
            super(is);
        }

        // TODO - multiline comments /* ... */ not taken into account for now in all cases
        public int read() throws IOException
        {
            while(true)
            {
                switch(state)
                {
                    case defstate:
                    {
                        int ch = in.read();
                        switch(ch)
                        {
                            case (int) '#':
                                state = State.hash;

```

```

        buffer = "";
        bufpos = 0;
        return ch;
    case (int)' ':
    case (int)'\t':
        buffer += (char)ch;
        break;
    case -1:
        state = State.eof;
        break;
    default:
        buffer += (char)ch;
        state = State.schmoo;
        break;
    }
    break;
}
case eol:
    if(bufpos < buffer.length()) return (int)buffer.charAt(bufpos++);
    else
    {
        state = State.defstate;
        buffer = "";
        bufpos = 0;
        return '\n';
    }
case eof:
    if(bufpos < buffer.length()) return (int)buffer.charAt(bufpos++);
    else return -1;
case hash:
{
    int ch = (int)in.read();
    switch(ch)
    {
        case (int)'#':
            state = State.directive;
            return ch;
        case -1:
            state = State.eof;
            return -1;
        default:
            state = State.directive;
            buffer = "##";
            return ch;
    }
}
case directive:
{
    int ch = (int)in.read();
    if(ch == (int)'\n')
    {
        state = State.eol;
        break;
    }
    else if (ch == -1)
    {
        state = State.eof;
        break;
    }
    else return ch;
}
case schmoo:
{
    int ch = (int)in.read();
    if(ch == (int)'\n')
    {
        state = State.eol;
        break;
    }
    else if (ch == -1)
    {

```

```

        state = State.eof;
        break;
    }
    else
    {
        buffer += (char)ch;
        return (int)buffer.charAt(bufpos++);
    }
}
}
}

public int read(byte[] b, int off, int len) throws IOException
{
    int i;
    int ok = 0;
    while (len-- > 0) {
        i = read();
        if (i == -1) return (ok == 0) ? -1 : ok;
        b[off++] = (byte) i;
        ok++;
    }
    return ok;
}

public int read(byte[] b) throws IOException
{
    return read(b,0,b.length);
}

public boolean markSupported()
{
    return false;
}
}

public synchronized InputStream getResourceStream(String name)
{
    return new VTIndentationGlobber(super.getResourceStream(name));
}

// test
public static void main(String args[])
{
    try
    {
        java.io.BufferedReader reader = new java.io.BufferedReader(new java.io.InputStreamReader(new
VTIndentationGlobber(new java.io.FileInputStream(args[0]))));
        String line;
        while( (line = reader.readLine() ) != null )
        {
            System.out.println(line);
        }
    }
    catch(IOException ioe)
    {
        ioe.printStackTrace();
    }
}
}

```