

# LogChuteCommonsLog

This is a replacement of the [LogSystemCommonsLog](#) that is part of [VelocityTools](#) 1.1 and 1.2. It will be added to the trunk of [VelocityTools](#) after version 1.2 is released. This is dependent on Velocity 1.5 and provides additional logging facilities to match improvements in that version.

```
/*
 * Copyright 2005 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.generic.log;

import org.apache.velocity.app.Velocity;
import org.apache.velocity.app.VelocityEngine;
import org.apache.velocity.runtime.log.Log;
import org.apache.velocity.runtime.log.LogChute;

/**
 * Redirects commons-logging messages to Velocity's configured LogChute.
 *
 * <p>To use, specify this class in your commons-logging.properties:
 * <code>
 * org.apache.commons.logging.Log=org.apache.velocity.tools.log.LogChuteCommonsLog
 * </code>
 * </p>
 * @since VelocityTools 1.3
 * @version $Id: LogChuteCommonsLog.java 72115 2004-11-11 07:00:54Z nbubna $
 */
public class LogChuteCommonsLog implements org.apache.commons.logging.Log
{
    protected static Log target = null;

    /**
     * Set a VelocityEngine to handle all the log messages.
     */
    public static void setVelocityLog(Log target)
    {
        LogChuteCommonsLog.target = target;
    }

    // ***** begin non-static stuff *****

    private String name;

    public LogChuteCommonsLog()
    {
        this("");
    }

    public LogChuteCommonsLog(String name)
    {
        if (name == null)
        {
            throw new NullPointerException("Log name cannot be null");
        }
    }
}
```

```

protected Log getTarget()
{
    if (target == null)
    {
        return Velocity.getLog();
    }
    else
    {
        return target;
    }
}

/***** Commons Log Interface *****/

/**
 * Passes messages to Velocity's LogChute at "DEBUG" level.
 * (it's the lowest available. sorry.)
 */
public void trace(Object message)
{
    getTarget().trace(message);
}

/**
 * Passes messages to Velocity's LogChute at "DEBUG" level.
 * (it's the lowest available. sorry.)
 */
public void trace(Object message, Throwable t)
{
    getTarget().trace(message, t);
}

/**
 * Passes messages to Velocity's LogChute at "DEBUG" level.
 */
public void debug(Object message)
{
    getTarget().debug(message);
}

/**
 * Passes messages to Velocity's LogChute at "DEBUG" level.
 */
public void debug(Object message, Throwable t)
{
    getTarget().debug(message, t);
}

/**
 * Passes messages to Velocity's LogChute at "INFO" level.
 */
public void info(Object message)
{
    getTarget().info(message);
}

/**
 * Passes messages to Velocity's LogChute at "INFO" level.
 */
public void info(Object message, Throwable t)
{
    getTarget().info(message, t);
}

/**
 * Passes messages to Velocity's LogChute at "WARN" level.
 */
public void warn(Object message)
{
    getTarget().warn(message);
}

```

```

}

/**
 * Passes messages to Velocity's LogChute at "WARN" level.
 */
public void warn(Object message, Throwable t)
{
    getTarget().warn(message, t);
}

/**
 * Passes messages to Velocity's LogChute at "ERROR" level.
 */
public void error(Object message)
{
    getTarget().error(message);
}

/**
 * Passes messages to Velocity's LogChute at "ERROR" level.
 */
public void error(Object message, Throwable t)
{
    getTarget().error(message, t);
}

/**
 * Passes messages to Velocity's LogChute at "ERROR" level.
 * (it's the highest available. sorry.)
 */
public void fatal(Object message)
{
    getTarget().error(message);
}

/**
 * Passes messages to Velocity's LogChute at "ERROR" level.
 * (it's the highest available. sorry.)
 */
public void fatal(Object message, Throwable t)
{
    getTarget().error(message, t);
}

/**
 * Returns true if Velocity's LogChute returns true
 * for isTraceEnabled().
 */
public boolean isTraceEnabled()
{
    return getTarget().isTraceEnabled();
}

/**
 * Returns true if Velocity's LogChute returns true
 * for isDebugEnabled().
 */
public boolean isDebugEnabled()
{
    return getTarget().isDebugEnabled();
}

/**
 * Returns true if Velocity's LogChute returns true
 * for isInfoEnabled().
 */
public boolean isInfoEnabled()
{
    return getTarget().isInfoEnabled();
}

```

```
/**
 * Returns true if Velocity's LogChute returns true
 * for isWarnEnabled().
 */
public boolean isWarnEnabled()
{
    return getTarget().isWarnEnabled();
}

/**
 * Returns true if Velocity's LogChute returns true
 * for isErrorEnabled().
 */
public boolean isErrorEnabled()
{
    return getTarget().isErrorEnabled();
}

/**
 * Returns true if isErrorEnabled() returns true, since
 * Velocity's LogChute doesn't support this level.
 */
public boolean isFatalEnabled()
{
    return isErrorEnabled();
}
}
```