

ReloadingTextFile

2008-03-18 21:30:44

Häufig gewollt: Wenn sich ein Template oder eine Konfigurationsdatei ändert, dann soll sie auch neu geladen werden, ohne die Anwendung neu zu starten, das ganze manuell zu machen oder bei jeder Anfrage die Datei neu zu lesen auf dem aktuellen Stand bleiben:

```
import java.io.File; import java.io.FileReader; import java.io.IOException; import java.io.Reader; import
java.io.StringWriter; import java.io.Writer; /** * Created 18.03.2008 * * @author M. Serhat Cinar */
public class ReloadingTextFile { private long checkEvery = 10000; private long lastCheck; private long
fileLastModified; private File file; private String content; /** * Checks every 10 second * * @param
file */ public ReloadingTextFile(File file) { this(file, 10000); } /** * @param file * @param
checkEvery */ public ReloadingTextFile(File file, long checkEvery) { super(); this.file = file;
this.checkEvery = checkEvery; } public synchronized String getContent() throws IOException { final
long now = System.currentTimeMillis(); if (now - lastCheck > checkEvery) { // recheck file timestamp
long actualFileLastModified = file.lastModified(); if (actualFileLastModified != fileLastModified) { //
change occurred, reload content = loadTextFile(file); fileLastModified = actualFileLastModified; }
lastCheck = now; } return content; } private final String loadTextFile() throws IOException { final
StringWriter sw = new StringWriter(((int) file.length())); final FileReader fr = new FileReader(file); try {
copy(fr, sw); } finally { if (fr != null) try { fr.close(); } catch (IOException e) { // TODO LOG ME
} try { if (sw != null) sw.close(); } catch (IOException e) { // TODO LOG ME } } return
sw.toString(); } private final void copy(final Reader fis, final Writer fos) throws IOException { final char
buffer[] = new char[0xffff]; int readChars; while ((readChars = fis.read(buffer)) != -1) {
fos.write(buffer, 0, readChars); } fos.flush(); } }
```