

Kompilering

Der er ikke ændret krav til kompilering i forhold til den oprindelige JAMWiki. For at kompilere kræves følgende:

JDK 5.0 eller nyere (6.0 har været brugt under udvikling).

Maven2

Internet adgang

Udpak source koden, gå til roden af koden (hvor pom.xml ligger) og kørs kommandoen:

```
mvn package
```

Så vil Maven hente alle de krævede pakker, kompilere source koden og pakke resultatet i en war-fil. Resultatet (war-filen) bliver lagt i jamwiki-war/target/. NetBeans har et Maven2 plugin, så man kan kompilere direkte fra NetBeans. Maven er blevet standard i NetBeans 6.7 hvor Java EE er installeret, men i NetBeans 6.5, som er blevet brugt under udvikling af softwaren, skal det installeres separat.

Installation

For at installere JAMWiki med integritets udvidelsen, så skal man have en Java Applikation Server, som f.eks. Tomcat eller GlassFish. Serveren skal som minimum bruge Java 5.0. Softwaren skal køres i exploded mode, hvilket er standard i den officielle udgave af Tomcat 6. I forbindelse med Tomcat skal man være opmærksom på at war filer, som bliver aktiveret via "Auto Deploy" funktionen, som udgangspunkt bliver belagt med stærke restriktioner, mens war filer, som aktiveres gennem interfacet bliver kørt i exploited mode.

Windows udgaven af Tomcat kan findes på: <http://tomcat.apache.org/download-60.cgi> og hedder "Windows Service Installer", efter installation kan JAMWiki inkl. Integritets udvidelserne aktiveres på <http://localhost:8080/manager/html/> ved at vælge war filen fra CD-rommen.

Når war-filen er blevet aktiveret, gå til siden og installations mulighederne vil blive vist. Der skal bruges standard databasen, de øvrige databaser er ikke fuldt ud implementeret endnu.

Der skal angives et sted på harddisken, hvor Wikien har lov til at skrive til. Husk at softwaren køres som Tomcat brugeren.

Angiv en administrator konto: brugernavn + password + password + gentaget.

Hvis der ikke sker fejl efter tryk på udfør, så er wikien klar til brug.

Det er ikke muligt at stige i niveau før der er nogle brugere på højere niveauer. Derfor vil det være nødvendigt for administratorene at opgradere de første brugere manuelt. Dette gøres ved at gå til UserClassification siden (Klik på User classification på brugerens side) og i URL'en tilføje: &changeLevel=X, hvor X er det niveau, man vil ændre brugeren til. For at undgå at aktivere

changeLevel ved en fejl anbefales det ikke at bruge administrator kontoen under dagligt brug. Når først systemet kører forventes det at brugerne selv stemmer hinanden op i systemet og changeLevel bør ikke længere bruges.

Parametre til Integritets modulet skal skrives i WEB-INF/classes/integrity.properties. Det kan være nødvendigt at køre Reload fra tomcat manager for at aktivere ændringerne.

En ændring man typisk vil lave er f.eks. at sætte votesForUserInc=2, så der kun skal to stemmer til at sende en bruger op i hierarkiet. Standardværdien er 20 og for høj indtil der er kommet tilstrækkeligt mange brugere. Bemærk at integrity.properties bliver slettet hvis man laver en Undeploy/Redeploy, så gem en kopi af integrity.properties for en sikkerheds skyld.

Eksempel på et alternativt interface

Det har hele tiden været meningen, at man skulle kunne ændre afstemningen. Og her er så en alternativ implementering. Det fungerer således:

IntegritySettingsAlternative tilføjes til projektet og modulet skrives. Projektet kompileres, nu har vi war-filen, men den ville kræve at vi skulle tage hele systemet ned og sætte det op igen, plus alle properties-filer nulstilles, hvilket vi vil undgå.

I stedet placeres IntegritySettingsAlternative.class i /org/jamwiki/model/ i en jar-fil, denne jar-fil placeres i jamwiki-war/WEB-INF/lib

I jamwiki-war/WEB-INF/classes/integrity.properties tilføjes linjen:
INTEGRITYSETTINGS=org.jamwiki.model.IntegritySettingsAlternative

Fra manageren vælges "reload" for at få JAMWiki til at læse properties filerne igen.

Nu bruges de alternative settings. Uden at vi har skullet sætte softwaren op igen.

Her er koden til modulet. Resten af kildekoden ligger på CD'en

```
/**
 * Licensed under the GNU LESSER GENERAL PUBLIC LICENSE, version 2.1, dated
 * February 1999.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the latest version of the GNU Lesser General
 * Public License as published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with this program (LICENSE.txt); if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
 */
package org.jamwiki.model;

import java.util.Properties;
import org.jamwiki.Environment;
import org.jamwiki.utils.WikiLogger;

/**
 * This is the handler that is called. This handler will load the correct
 * IntegritySettings module and make all calls to it. All calls to the
```

```

IntegritySettings
* must go through this Handler.
* This is implemented as a singleton
* @author poul
*/
public class IntegritySettingsHandler implements IntegritySettingsI {
    private static final WikiLogger logger =
WikiLogger.getLogger(IntegritySettingsHandler.class.getName());
    static private IntegritySettingsHandler singleton;
    //The default settings to load:
    static public String className = "org.jamwiki.model.IntegritySettings";

    /**
     * The properties file that we should look in to see if another Settings
     * file should be loaded
     */
    private static final String RESOURCE_NAME = "/integrity.properties";

    /**
     * The IntegritySettings must implment the interface
     */
    private IntegritySettingsI is;

    /**
     * private constructor for singleton
     */
    private IntegritySettingsHandler(){
        logger.fine("Constructing singleton");
        try{
            Properties p = Environment.loadProperties(RESOURCE_NAME);
            //Try looking in the properties file to see if
INTEGRITYSETTINGS is defined
            className = p.getProperty("INTEGRITYSETTINGS");
            //If not then fill default classname
            if(className == null || className.length()<2) {
                logger.fine("INTEGRITYSETTINGS not defined - using
default");
                className = "org.jamwiki.model.IntegritySettings";
            }
            //Load the class:
            ClassLoader classLoader =
IntegritySettingsHandler.class.getClassLoader();
            Class theClassToUse = classLoader.loadClass(className);
            is = (IntegritySettingsI)theClassToUse.newInstance();
            logger.info("IntegriSettings \""+className+"\" loaded");
        } catch (Exception e) {
            //If something wrong happens then load the failsafe settings.
            is = new IntegritySettingsFailsafe();
            logger.warning("Failed to load class \""+className+"\" using
failsafe");
        }
    }

    /**
     * Gets the instance of the class
     * @return the singletonclass
     */
    static public IntegritySettingsHandler getInstance() {
        logger.finest("getInstance called");
        if(singleton==null)
        {
            singleton = new IntegritySettingsHandler();

```

```

        }
        return singleton;
    }

    /**
     * Gives a description of the integrity settings implemented.
     * @return the string that can be put on the homepage as clean text
     */
    public String getDescription() {
        logger.finest("getDescription called");
        return is.getDescription();
    }

    /**
     * updates the logic. The actual logic is implementation dependent.
     */
    public void runLogic() {
        logger.finer("runLogic called");
        is.updateLogic();
    }

    /**
     * Just to fully implement the interface
     */
    public void updateLogic() {
        runLogic();
    }

    /**
     * Determens if a user is allowed to vote
     * @param wu The wikiuser that wants to vote
     * @param wv The vote the user wants to vote for
     * @return true if the user is allowed to vote.
     */
    public boolean userCanVote(WikiUser wu, WikiVote wv) {
        return is.userCanVote(wu, wv);
    }

    /**
     * Max level the topic may be moved up to by vote.
     * @param caller the user that wants to increase the level
     * @param id The topic that needs to get its level increased
     * @return the level it may be moved to.
     */
    public int maxLevelForTopic(WikiUser caller, Topic topic) {
        return is.maxLevelForTopic(caller, topic);
    }

    public int minLevelForTopic(WikiUser caller, Topic topic) {
        return is.minLevelForTopic(caller, topic);
    }

    /**
     * The max level the user may be moved to by a single vote.
     * @param user the user in question
     * @return the level that the user may be moved to by vote.
     */
    public int maxLevelForUser(WikiUser user) {
        return is.maxLevelForUser(user);
    }

    /**

```

```

    * The min level the user may be moved to by vote (degraded)
    * @param user the user in question
    * @return the level the user may be moved to by vote.
    */
    public int minLevelForUser(WikiUser user) {
        return is.minLevelForUser(user);
    }

    /**
     * Can the users votes affect the user in question:
     * @param caller The user that wants to vote
     * @param target The user that will be voted for
     * @return If the vote will count
     */
    public boolean userCanVote(WikiUser caller, WikiUser target) {
        return is.userCanVote(caller, target);
    }

    /**
     * Can the user degrade another user by calling a vote?
     * @param caller The user that wants to call a vote
     * @param target The user that will be degraded should the vote pass
     * @return If the vote can be started
     */
    public boolean userCanVoteDown(WikiUser caller, WikiUser target) {
        return is.userCanVoteDown(caller, target);
    }

    /**
     * Fills end and expire dates in a vote based on starttime
     * @param vote The vote to fill
     */
    public void fillDates(WikiVote vote) {
        is.fillDates(vote);
    }

    /**
     * Checks a vote for ended expired etc.
     * @param v the vote to be handled
     */
    public void handleVote(WikiVote v) {
        is.handleVote(v);
    }
}

```