Roller Installation Guide

Roller Weblogger
Version 2.1-incubating

Table of Contents

Introduction

This document describes how to install Roller in the following environment:

- Operating System: UNIX or Windows based operating system
- Java development kit: Java 2 SE 1.4 SDK (or later)
- Application server: Tomcat 5.X (or later)
- Relational Database: MySQL 4.X (or later)

What do you need to know to install Roller? You need to know how to use the UNIX or Windows command-line, how to set environment variables, how to create a database in MySQL and how to start and stop Tomcat.

What about other servlet containers? These instructions target Tomcat, but you should be able to make Roller work with just about any standard Servlet 2.3 compatible application server. For full support of Roller's internationalization features, we recommend Servlet 2.4. If you deploy Roller to a non-Tomcat server, please contribute your install notes to help others who might want to do the same.

What about other databases? These instructions target MySQL, but Roller also includes database setup scripts for PostgreSQL 7.X later, Apache Derby, IBM DB2, Oracle and HSQL-DB.

What platform combinations are known to work? For information on which platforms we can vouch for, see the the *Platforms* page on the Roller wiki.

STEP 1: Before you install Roller

Before you install Roller software you should install and configure the Java development kit, your application server and your database.

As part of the Tomcat install you should have set the environment variable CATALINA_HOME to point to your Tomcat installation directory. If not, you might want to set it now because we will refer to it in this installation guide. Below are some examples that show how to set this variable. Make sure you substitute the right path to your Tomcat installation.

For UNIX with bash shell:

```
% export CATALINA_HOME=/opt/jakarta-tomcat-5.0.29
```

For UNIX with c-shell:

```
% setenv CATALINA_HOME /opt/jakarta-tomcat-5.0.29
```

For Windows with DOS shell

```
C> set CATALINA_HOME d:\jakarta-tomcat-5.0.29
```

NOTES

- For MySQL, make sure you enable UTF-8 support. See the page Setting Up UTF-8 on MySQL page on the Roller wiki for details.
- For MySQL, make sure that TCP/IP networking is enabled. In some versions of MySQL, this option is off by default. See the page Debian MySQL for details. The Connector/J JDBC driver can only access MySQL via TCP/IP.

STEP 2: Unpack the Roller distribution

Pick a directory on your computer and unpack the Roller distribution using either GNU tar on UNIX or WinZip on Windows. Hereinafter, we'll refer to that directory as your Roller installation directory, `$ROLLER` in the UNIX examples or `%ROLLER%` in the Windows examples.

Here are some examples to show you how you might unpack Roller on your computer.

UNIX example

Assuming you download the distribution into your home directory and you'd like to install Roller into `/usr/local` you might do something like this:

```
% cd /usr/local
% tar xzvf ~/roller-2.1.tar.gz
```

That would create the Roller installation directory `/usr/local/roller`.

Windows example

Use WinZip to extract the Roller distribution file into the directory of your choice for example, no example necessary.

STEP 3: Create Roller tables in your database

Create a new database within your MySQL installation, create a user with all privileges within that database and run the Roller database creation script to create tables within that new database.

Roller includes database creation scripts for a variety of database, but MySQL is the most widely used and best supported option. You can find the database creation scripts in the Roller installation directory under `WEB-INF/dbscripts/<dbname>`. Here's the list of scripts currently in Roller:

- `WEB-INF/dbscripts/mysql/creatdb.sql`
- `WEB-INF/dbscripts/postgresql/creatdb.sql`
- `WEB-INF/dbscripts/hsqldb/creatdb.sql`
- `WEB-INF/dbscripts/derby/creatdb.sql`
- `WEB-INF/dbscripts/db2/creatdb.sql`
- `WEB-INF/dbscripts/oracle/creatdb.sql`

Here are some examples to show you how you might create the Roller tables in MySQL:

UNIX example:

```
% cd $ROLLER/WEB-INF/dbscripts/mysql
% mysql -u root -p
password: *****
mysql> create database roller;
mysql> grant all on roller.* to scott@'%' identified by 'tiger';
mysql> grant all on roller.* to scott@localhost identified by 'tiger';
mysql> use roller;
mysql> source createdb.sql
mysql> quit
```

Windows example:

```
C> cd %ROLLER%\dbscripts\mysql
C> mysql -u root -p
password: *****
mysql> create database roller;
mysql> grant all on roller.* to scott@'%' identified by 'tiger';
mysql> grant all on roller.* to scott@'localhost' identified by 'tiger';
mysql> use roller;
mysql> source createdb.sql
mysql> quit
```

NOTES

- For MySQL, don't forget to call *flush privileges* to commit your changes to MySQL.
- To check whether your MySQL is setup properly, use the command line mysql program to connect using the user name and password you created. For example (we use 127.0.0.1 here instead of localhost to ensure that TCP/IP networking is enabled):

```
mysql roller -h 127.0.0.1 -u scott -ptiger
```

STEP 4: Install required third party jars

You also need to download and install some third-party jars, jars that we can't include in Roller due to licensing restrictions. These are the JDBC driver, JavaMail and Activation jars.

STEP 4.1: Install JDBC driver jar(s)

Download the JDBC driver jar for your database and put it in the classpath of your application server. For example, assuming Tomcat and MySQL, you'd download the J/Connector JDBC driver from mysql.com and you'd place it in the Tomcat `common/lib` directory.

UNIX example

```
% cp mysql-connector.jar $CATALINA_HOME/common/lib
```

Windows example

```
C> copy mysql-connector.jar %CATALINA_HOME%\common\lib
```

NOTES

- For MySQL 4.1.X users, we recommend that you use the J/Connector 3.0.X JDBC drivers instead of the newer 3.1.X series. If you *must* use J/Connector/J 3.1.X then please read *Installation FAQ* page item #13 on the Roller wiki.

STEP 4.2: Install JavaMail and Activation jars

If you like to use Roller's e-mail notification features, you'll need to add the JavaMail and Activation jars to your application server's classpath. Currently, we ship these jars with Roller, so **move** them from the Roller `WEB-INF/lib` directory and into the Tomcat `common/lib` directory.

UNIX example

```
% cp $ROLLER/WEB-INF/lib/mail.jar $CATALINA_HOME/common/lib
% cp $ROLLER/WEB-INF/lib/activation.jar $CATALINA_HOME/common/lib
```

Windows example

```
c> cp %ROLLER%\WEB-INF\lib\mail.jar %CATALINA_HOME%\common\lib
C> cp %ROLLER%\WEB-INF\lib\activation.jar %CATALINA_HOME%\common\lib
```

NOTES

- To enable the Roller's e-mail notification features, you'll also need to setup a mail session resource in your application server configuration file (see the next section) and you'll need to configure e-mail notification in the Roller UI.

STEP 5: Deploy Roller to your application server

To deploy Roller to your application server you need to inform your application server:

- Where to find the Roller installation directory
- How to configure the Roller datasource under the JNDI name `jdbc/rollerdb`

For Tomcat you can do this by creating what's known as a context configuration file named `roller.xml` and placing that file in the Tomcat `conf/Catalina/localhost` directory.

Example context configuration file for Tomcat 5.0.X users

The portions shown in bold are the ones that you'll probably have to change. Make sure you set the docBase to point to your Roller installation directory. Make sure you set the JDBC connection string to point to your database and the database username and password too.

```
<Context path="/roller" docBase="/usr/local/roller" debug="0">
  <Resource name="jdbc/rollerdb" auth="Container" type="javax.sql.DataSource" />
  <ResourceParams name="jdbc/rollerdb">
    <parameter>
      <name>factory</name>
      <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>com.mysql.jdbc.Driver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>
      jdbc:mysql://localhost:3306/roller?autoReconnect=true&amp;useUnicode=true&amp;characterEncoding=utf-8&amp;mysqlEncoding=utf8
      </value>
    </parameter>
    <parameter><name>username</name><value>scott</value></parameter>
    <parameter><name>password</name><value>tiger</value></parameter>
    <parameter><name>maxActive</name><value>20</value></parameter>
    <parameter><name>maxIdle</name><value>3</value></parameter>
    <parameter><name>removeAbandoned</name><value>true</value></parameter>
    <parameter><name>maxWait</name><value>3000</value></parameter>
  </ResourceParams>
  <!-- If you want e-mail features, un-comment the section below -->
  <!--
  <Resource name="mail/Session" auth="Container" type="javax.mail.Session"/>
    <ResourceParams name="mail/Session">
      <parameter>
        <name>mail.smtp.host</name>
        <value>mailhost.example.com</value>
      </parameter>
  </ResourceParams>
  -->
</Context>
```

Example context configuration file for Tomcat 5.5.X users

The portions shown in bold are the ones that you'll probably have to change. Make sure you set the docBase to point to your Roller installation directory. Make sure you set the JDBC connection string to point to your database and the database username and password too.

```
<Context path="/roller" docBase="/usr/local/roller" debug="0">
  <Resource name="jdbc/rollerdb" auth="Container" type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/roller?autoReconnect=true&amp;useUnicode=true&amp;characterEncoding=utf-8&amp;mysqlEncoding=utf8"
    username="scott"
    password="tiger"
    maxActive="20"
    maxIdle="3"
    removeAbandoned="true"
    maxWait="3000" />
  <!-- If you want e-mail features, un-comment the section below -->
  <!--
  <Resource name="mail/Session" auth="Container" type="javax.mail.Session"
    mail.smtp.host="mailhost.example.com" />
  -->
</Context>
```

NOTES

- If Roller starts up fine but later fails and you find an error like the one below in your roller.log file then try dropping your maxActive, maxIdle, and removeAbandoned values. Depending on your database configuration you may have to go pretty low, such as setting maxActive to 6, maxIdle to 3 and removeAbandonedTimeout to 60.

  User scott@localhost has more than 'max_user_connections' active connections

STEP 6: Check your internationalization settings

Roller's approach to internationalization (I18N) is to do everything in UTF-8. So, if you want I18N to work properly, you'll need to configure your application server and your web server to use UTF-8 encoding.

Check your application server's URI encoding setting!

Make sure that your web application server uses UTF-8 to encode URI's. This allows you to use diacritical characters like 'ç' in your URLs. This is important for Roller because weblog entry titles are used in URLs.

For example, in Tomcat the URI encoding is specified in the connectors that are configured in the Tomcat configuration file `conf/server.xml`. Here's a connector with the URI encoding attribute set properly:

```
<Connector port="8080"
      enableLookups="false"
      URIEncoding="UTF-8"
      redirectPort="8443"
      debug="0"
      protocol="AJP/1.3" />
```

And make sure you do this for *every* connector through which you use Roller. For example, if you use the AJP connector or HTTPS connector you need to add the `URIEncoding="UTF-8"` attribute to those connectors as well.

STEP 7: Setup Roller data directories

Roller stores file uploads, search index files, cache files and log files on disk. So before you start, check to make sure the directories that Roller expects exist and are writable by the Tomcat process.

STEP 7.1: Create `uploads` directory

By default, Roller saves uploaded files under the directory:

```
${user.home}/roller_data/uploads
```

Here `${user.home}` is the Java system property that normally evaluates to the home directory of the user identity executing the server's JVM process.

In most cases, this default will probably work fine for you. However, for security reasons some application servers are set up to run as a server user identity whose home directory does not exist or is not writable by the server user itself. If this is the case for your server, override the property `uploads.dir` in the `roller.properties` file. See step 8 for more information on the `roller.properties` file.

STEP 7.2: Create `search-index` directory

By default, Roller creates and maintains its text search index data in files under the directory

```
${user.home}/roller_data/roller-index
```

Again, `${user.home}` is the Java system property that normally evaluates to the home directory of the user identity executing the server's JVM process. You can specify a different directory by overriding the property `search.index.dir` in `roller.properties`. See step 8 for more information on the `roller.properties` file.

STEP 7.3: Create `planet-cache` directory (optional)

You only need to do this if you are planning on using Roller's integrated planet aggregator: create a directory for the planet cache (e.g. /var/roller/planet-cache).

The planet aggregator is off by default. To enable it, you'll have to override the `planet.aggregator.enabled` property (see the [ConfigurationGuide](#)). Once you've got Roller up and running, login (as an adminstrator), go to the Planet:Configuration page and set full-path to the directory you choose for the planet-cache.

Currently, you must specify the location of the planet-cache directory through the Roller web UI (but we planning to move it to roller.properties because it is really a startup property).

STEP 7.4: Make sure that the `logs` subdirectory exists

The `roller.log` file is written to the location:

```
${catalina.base}/logs/roller.log
```

Make sure that that this directory exists. Tomcat 5.0.X users will normally have this directory by default. Tomcat 5.5.x users may need to create this subdirectory under their base directory.

STEP 8: Review Roller configuration

Before you start Roller for the first time, review your configuration.

STEP 8.1: Review the `WEB-INF/classes/roller.properties` file

Roller tries to pick a good set of configuration defaults which should let anyone start up the application without much work, but here are a few properties which are custom to each install and should be set before you start up Roller. We'll first talk about what ways there are for defining your custom Roller configuration, then show which properties we think you should set. **NOTE**: the default `roller.properties` file is shown in Appendix A.

Where to define custom properties

There are three ways you can alter the default configuration for Roller.

1) Define a `roller-custom.properties` file and place it somewhere at the root of one of your classpath locations. This is the recommended way to override Roller configuration properties. For example:

```
$TOMCAT_HOME/common/classes/roller-custom.properties
```
Or...
```
$TOMCAT_HOME/webapps/$ROLLER_WEBAPP/WEB-INF/classes/roller-custom.properties
```

2) Specify a custom properties file via JVM option. This is another good option but is more dependant on what servlet container you use. For example:

```
# this is how you might do it for tomcat
JAVA_OPTS="-Droller.custom.config=/path/to/properties/file"
export $JAVA_OPTS
$TOMCAT_HOME/bin/startup.sh
```

3) If you like, you can open up the default roller config file and change the values there. This option should not be necessary and is **NOT** recommended. For example:

```
$TOMCAT_HOME/webapps/$ROLLER_WEBAPP/WEB-INF/classes/roller.properties
```

What properties you should set

We are going to assume you have defined a `roller-custom.properties` file and placed it in your classpath somewhere. To override any of the default Roller properties you simply add a line with the proper key and the new value you wish to use.

Here is a sample `roller-custom.properties` with the few properties that should be overridden

```
uploads.dir=/app/roller/roller_data/uploads
search.index.dir=/app/roller/roller_data/search-index
# etc, etc, etc ... any other properties you want to override
```

STEP 8.2: Change keys in the `WEB-INF/security.xml` file

Starting with version 2.1, Roller uses the Acegi security infrastructure. Several of the security features rely on keys that are intended to be site-specific. These keys are used to compute HMAC (hash-based message authentication code) values for *Remember Me* cookies. Knowledge of these keys could allow an attacker to forge invalid cookies, and thereby gain unauthorized access to your Roller installation (at the application level).

Roller ships with default values, and these should assumed to be widely known. You should change your keys to be secret values specific to your own site.

Here is how to change the keys.

1. Find your `WEB-INF/security.xml` file and open it in a text editor.
2. For the beans with ids "anonymousAuthenticationProvider" and "anonymousProcessingFilter" change the `value` field of the property with `name="key"` to be different from the default value of "anonymous". You can use any string value of your choosing. It should be a secret specific to your site. Use the same key value in these two beans; they must match.
3. For the beans with ids "rememberMeServices" and "rememberMeAuthenticationProvider" change the `value` field of the property with `name="key"` to be different from the default value of "rollerlovesacegi". You can use any string value of your choosing. It should be a secret specific to your site. Use the same key value in these two beans; they must match

NOTES
- The reason one should change the anonymous provider key is that a granted authorities list is embedded within the anonymous authentication token.

STEP 9: Start Tomcat and start using Roller

Start your Servlet Container, open your web browser, browse to the Roller start page and start using Roller. If you are installing Roller on Tomcat then your Roller start page URL is probably `http://localhost:8080/roller`.

UNIX example

```
% cd $CATALINA_HOME/bin
% ./startup.sh
```

Windows example

```
C> cd %CATALINA_HOME%\bin
C> startup
```

NOTE: the first user you create will have administrator privileges, so make sure you create the first user yourself. An admin user can grant and revoke admin rights from other users.

You're done!

Roller should be working perfectly now. If not, then please consult the Installation FAQ page on the Roller wiki and then check with the experts on the Roller mailing lists. Somebody has probably encountered the very same problems that you are encountering.

Appendix A: The `WEB-INF/roller.properties` file

This file defines the default start-up properties for Roller. See step 8 for instructions on how to override the properties in this file.

```
# roller.properties
#
# This file is for meant for Roller deployment properties
# Any properties that don't change at runtime are defined here

# User management settings

# True to enable group blogging. False to prevent users from creating more
# than one weblog and from joining other weblogs.
groupblogging.enabled=true

#--------------------------------
# Property expansion settings

# Values of the properties in this list get system property expansion
# applied to them when loaded.
config.expandedProperties=uploads.dir,search.index.dir

#--------------------------------
# Upload settings

# The directory in which Roller will upload files
uploads.dir=${user.home}/roller_data/uploads

# The context path under which resures will be made available
uploads.url=/resources

#--------------------------------
# Search index settings

# Enables indexing of weblog entries and comments and enables search servlet
search.enabled=true
# Directory in which search index is to be created (delete this directory to
# force Roller to recreate the entire search index)
search.index.dir=${user.home}/roller_data/search-index
# Whether or not to include comments in the search index.  If this
# is false, comments are not included in the index.
search.index.comments=true

#--------------------------------
# Cache settings.
# Remember ... times are in seconds
# Default settings suitable for 100 user system

# Cache properties all follow the given format ...
#     cache.<cache_id>.<prop>=<value>
# we then pass all <prop>=<value> pairs into the cache manager when the cache
# is being constructed.  this makes it easy to add cache properties that can
# be used by the specified CacheFactory you are using.
#
# NOTE: it is expected that property validation happens in the CacheFactory
```

```
#----------------------------------

# The default cache implementation we want to use
cache.defaultFactory=org.roller.presentation.cache.ExpiringLRUCacheFactoryImpl
cache.customHandlers=

# Main/Planet page cache (this is low on purpose)
cache.mainpage.size=10
cache.mainpage.timeout=1800
# set "true" to NOT cache the custom pages for users who are logged in
cache.mainpage.excludeOwnerEditPages=false

# Weblog page cache (all the weblog content)
cache.weblogpage.size=400
cache.weblogpage.timeout=3600
# set "true" to NOT cache the custom pages for users who are logged in
cache.weblogpage.excludeOwnerEditPages=false

# Weblog page last-modified-date cache
# you want this fairly high, like weblogs * 10, with long timeouts
cache.ifmodified.weblogpage.size=1000
cache.ifmodified.weblogpage.timeout=14400

# Feed cache (xml feeds like rss, atom, etc)
cache.feed.size=200
cache.feed.timeout=3600

# Feed last-modified-date cache
# you want a reasonable size, like weblogs * 2, with long timeouts
cache.ifmodified.feed.size=200
cache.ifmodified.feed.timeout=14400

# Planet cache (planet page and rss feed)
cache.planet.size=10
cache.planet.timeout=1800
# set "true" to NOT cache the custom pages for users who are logged in
cache.planet.excludeOwnerEditPages=false

#--------------------------------
# Secure login configs

# Enables HTTPS for login page only
securelogin.enabled=false

# Enable scheme enforcement?
# Scheme enforcement ensures that specific URLs are viewed only via HTTPS
schemeenforcement.enabled=false
# URL patterns that require HTTPS
schemeenforcement.https.urls=/editor/yourProfile.do*,/admin/user.do*

# Password security settings
passwds.encryption.enabled=false
passwds.encryption.algorithm=SHA

#--------------------------------
# Enabled plugins ... remember, order does matter!!

# Weblog entry editor plugins
plugins.page=\
```

```
org.roller.presentation.velocity.plugins.convertbreaks.ConvertLineBreaksPlugin \
,org.roller.presentation.velocity.plugins.topictag.TopicTagPlugin \
,org.roller.presentation.velocity.plugins.search.WikipediaLinkPlugin \
,org.roller.presentation.velocity.plugins.search.GoogleLinkPlugin \
,org.roller.presentation.velocity.plugins.textile.TextilePlugin \
,org.roller.presentation.velocity.plugins.acronyms.AcronymsPlugin \
,org.roller.presentation.velocity.plugins.bookmarks.BookmarkPlugin \
,org.roller.presentation.velocity.plugins.email.ObfuscateEmailPlugin \
,org.roller.presentation.velocity.plugins.smileys.SmileysPlugin \
,org.roller.presentation.velocity.plugins.readmore.ReadMorePlugin

#---------------------------------
# scheduled tasks, each is comma separated list of classes

# Comma separated list of task classnames to be executed once per day
tasks.daily=org.roller.presentation.TurnoverReferersTask\

# Needed to enable nightly fetching of blacklist
#,org.roller.presentation.BlacklistUpdateTask\

# Daily Planet task: syncs weblog list with Roller, refreshes Technorati stats
#,org.roller.presentation.planet.SyncWebsitesTask

# Comma separated list of task classnames to be executed hourly
tasks.hourly=\

# Hourly Planet task: refresh latest entry list from all weblogs in list
#org.roller.presentation.planet.RefreshEntriesTask

#---------------------------------
# Velocity settings

velocity.properties=/WEB-INF/velocity.properties
velocity.toolbox.file=/WEB-INF/toolbox.xml

# Page model implementation
velocity.pagemodel.classname=org.roller.presentation.velocity.PageModel
# Experimental page model that allows user's access to Planet aggregations
#velocity.pagemodel.classname=org.roller.presentation.velocity.planet.PlanetPageMod
el

#---------------------------------
# Persistence settings

persistence.roller.classname=org.roller.business.hibernate.HibernateRollerImpl
persistence.filemanager.classname=org.roller.business.FileManagerImpl

# authenticator settings (experimental)
authenticator.classname=org.roller.presentation.DefaultAuthenticator

#---------------------------------
# comment, referrer and trackback settings

comment.authenticator.classname=org.roller.presentation.velocity.MathCommentAuthent
icator
comment.notification.separateOwnerMessage=false
comment.notification.hideCommenterAddresses=false

# enables site full blacklist check on comment posts (default: true)
```

```
site.blacklist.enable.comments=true

# enables site full blacklist check at time of trackback post (default: true)
site.blacklist.enable.trackbacks=true

# enables partial blacklist check (not including blacklist.txt) for each incoming
referrer
site.blacklist.enable.referrers=true

# Trackback protection. Set this only if you need to limit the URLs to
# which users may send trackbacks. Regex expressions are allowed, for example:
# trackback.allowedURLs=http://w3.ibm.com/.*||http://another.example.com/.*
trackback.allowedURLs=

#Robot check in referral processing.  If this pattern is set and the User-Agent in
the
#request matches this pattern, all referral processing is skipped; this means that
#the referral spam check is skipped, the request is allowed to proceed, but the
#referrer is not recorded and hit count is not incremented.  Recommended for large
sites
#that get a lot of legitimate crawler bot traffic.  The pattern here is a
suggestion that
#has been reported to work well.
#referrer.robotCheck.userAgentPattern=.*(slurp|bot|java).*

# Enable built-in referrer processing?
referrers.processing.enabled=true

# Change to true if you want to process referrers asynchronously.
# You can choose how many threads to use and sleep time (in seconds)
referrers.asyncProcessing.enabled=false
referrers.queue.numWorkers=3
referrers.queue.sleepTime=10

#-------------------------------
# ping settings

# The number of attempts to try to reach a ping target before refusing to
# requeue it for further retrials. If absent, this defaults to 3.
pings.maxPingAttempts=3

# The interval between ping queue processing runs in minutes.  Must be between
# 0 and 120. If set to 0, ping queue processing is disabled on this server;
# this is for clustered environments. Make sure it is nonzero on one host in
# a cluster.  Don't use the value 0 here to disable ping functionality, you
# will instead get an infinitely growing ping queue.  See the documentation on
# the properties below to disable ping functionality if you need to.
# If absent, this defaults to 5.
pings.queueProcessingIntervalMins=5

# The set of initial common ping targets.  This is used to initialize the
# database if there are no common ping targets at startup.  Ping targets are
# specified as a comma-separated list, each target in the form {{name}{url}}.
# To disable initialization of common ping targets, comment this out, or set it
# to the empty value.  Common targets can be edited in the UI; this is just
# used to set up some typical ones.
pings.initialCommonTargets=\
{{Technorati}{http://rpc.technorati.com/rpc/ping}}\
,{{Weblogs.com}{http://rpc.weblogs.com/RPC2}}\
```

```
,{{blo.gs}{http://ping.blo.gs/}}\
,{{java.blogs}{http://javablogs.com/xmlrpc}}\
,{{blogrolling.com}{http://rpc.blogrolling.com/pinger/}}

# This controls whether users are allowed to add custom ping targets.
# Set this to false to disallow adding custom targets; if false, the
# Weblog:Custom Ping Targets menu item will not appear and associated actions
# will result in access denied messages.  Leave this false or commented for
# normal behavior.
# CAUTION: Setting this to true will cause the server to remove all users'
# existing custom targets on startup.
pings.disallowCustomTargets=false

# This controls whether the Weblog:Pings menu item and its associated actions
# are enabled.  Set this to false to disallow users from configuring autopings
# and doing manual pings.  If absent, this defaults to true.
# NOTE: There is a separate runtime property (configurable from the
# Admin:Configuration page, that can be used to suspend ping processing without
# disabling the UI.
# CAUTION: Setting this to true will cause the server to remove all users'
# existing autoping configurations on startup. Leave this false or commented
# for normal behavior.
pings.disablePingUsage=false

# Setting both pings.disallowCustomTarget=true and pings.disablePingUsage=true
# will effectively disable the ping functionality.

# This is used for debugging the ping mechanism in Roller.  If this is set
# to true, pings that would normally be sent will cause log messages to be sent
# but will NOT actually result in real pings being sent.  Leave this false or
# commented for normal behavior.
pings.logOnly=false

#-------------------------------
# Planet Aggregator settings

# Set to true to enable the Planet aggregator. This will cause:
# - The main page of Roller will become an aggregated view of all blogs in the
#   Roller database, plus those defined in the Planet group 'external'.
# - A new menu tab will appear for Roller admin users. This allows admins to
#   add/remove newsfeed subscriptions in the 'external' group.
# - Users can then subscribe to several newsfeeds:
#      - http://localhost:8080/roller/rss
#      - http://localhost:8080/roller/planetrss
#      - http://localhost:8080/roller/planetrss?group=external
planet.aggregator.enabled=false

# Planet cache must exist and must be writable by Roller process
planet.aggregator.cache.dir=/var/roller/planetcache

#-------------------------------
# user creation setttings: blogroll and cats for new users

# list of links to include in root bookmark folder of each new blog
# format is like so: linktitle2|linkurl2,linktitle2|linkurl2,linktitle3|linkurl3
newuser.blogroll=\
Dave Johnson|http://rollerweblogger.org/page/roller,\
Matt Raible|http://raibledesigns.com/page/rd,\
Lance Lavandowska|http://brainopolis.dnsalias.com/roller/page/lance,\
```

```
Henri Yandell|http://blog.generationjava.com/roller/page/bayard,\
blogs.sun.com|http://blogs.sun.com,\
jroller.com|http://jroller.com

# comma-separated list of top-level categories to be created in each new weblog
newuser.categories=\
General,Status,Java,Music,Politics


#-------------------------------
# misc settings

rememberme.enabled=true
breadcrumbs.stacksize=3
debug.memory.enabled=false

# editor theme to be used (corresponds to directory name under /theme)
editor.theme=tan

#-------------------------------
# settings for various plugins

# Optional site-wide customization settings for the TopicTag plugin.
# n.b. these default settings match the coded default values that would be
# applied if these were omitted.
org.roller.presentation.velocity.plugins.topictag.TopicTagPlugin.defaultTopicBookma
rkName=Default Topic Site
org.roller.presentation.velocity.plugins.topictag.TopicTagPlugin.defaultTopicSite=h
ttp://www.technorati.com/tag
org.roller.presentation.velocity.plugins.topictag.TopicTagPlugin.tagRegexWithBookma
rk=topic:\\{(.*?)\\}\\[(.*?)\\]
org.roller.presentation.velocity.plugins.topictag.TopicTagPlugin.tagRegexWithoutBoo
kmark=topic:\\[(.*?)\\]
org.roller.presentation.velocity.plugins.topictag.TopicTagPlugin.linkFormatString=<
a rel=\"tag\" href=\"{0}{1}\">{2}</a>

# Set to true to allow only default topic tag site (and avoid costly bookmark
queries)
org.roller.presentation.velocity.plugins.topictag.TopicTagPlugin.ignoreBookmarks=tr
ue


#-------------------------------
# legacy settings (thing that should be deprecated

# settings for old #showNewseeds macro (not related to Planet stuff)
aggregator.enabled=false
aggregator.cache.enabled=
aggregator.cache.timeout=14400
```