Installation Guide

Apache Roller
(incubating)
Version 3.1

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)

NOTE: If you're upgrading from an earlier release of Roller, read Appendix A first

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-5.5.9
```

For UNIX with c-shell:

```
% setenv CATALINA_HOME /opt/jakarta-tomcat-5.5.9
```

For Windows with DOS shell

```
C> set CATALINA_HOME d:\jakarta-tomcat-5.5.9
```

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. 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 in the bash shell:

```
% cd /usr/local
% tar xzvf ~/apache-roller-3.0-incubating.tar.gz
```

That would create the Roller installation directory `/usr/local/apache-roller-3.0-incubating`.

Windows example

You can Use WinZip to extract the Roller distribution file into the directory of your choice.

Roller distribution layout

Once you've extracted the files you'll see that the Roller release contains two directories and a couple of text files:

docs	Directory containing Roller documentation
webapp/roller	Directory containing rhe Roller web application in WAR directory layout
README.txt	Explains what Roller is
CHANGES.txt	Lists changes made in each release
NOTICE.txt	Copyright notices and credits
LICENSE.txt	The Apache Software License

The ROLLER environment variable

In this guide, we'll refer to the Roller web application directory using the ROLLER environment variable. In UNIX this will be $ROLLER. In Windows, it will be %ROLLER%. You don't *have* to set the ROLLER environment variable, we just use it to simplify the installation guide, but here's how you'd do it.

UNIX example (assuming you installed into `/usr/local`):

```
% set $ROLLER = /usr/local/apache-roller-3.0-incubating/webapp/roller
```

Windows example (assuming you installed into `c:\`)

```
% set %ROLLER% = c:\apache-roller-3.0-incubating/webapp/roller
```

STEP 3: 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 3.1: Download and install Hibernate

Roller *requires* the Hibernate persistence library, which you must download separately from Roller.

> Download Hibernate 3.1.2 from SourceForge
>
> http://prdownloads.sourceforge.net/hibernate/hibernate-3.1.2.tar.gz

Copy the following files from Hibernate into the Roller WEB-INF/lib directory:

- hibernate3.jar
- asm-attrs.jar
- asm.jar
- cglib-2.1.3.jar
- dom4j.1.6.1.jar
- ehcache-1.1.jar
- jdbc2_0-stdext.jar
- jta.jar

The Roller Support project at Java.Net offers some bundles that might make this part of the installation easier. Visit http://roller.dev.java.net for more information.

STEP 3.2: 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.
- For MySQL 5.X users, we recommend that you use the J/Connector 3.1.X JDBC drivers instead of the newer 3.1.X series, you'll also need to change the Hibernate configuration file to use the MySQL5 dialect (see Section 8.3 for details on that).
- For Oracle users, we recommend that you use the 10g (10.1.0.2 or higher) drivers which should be packaged as ojdbc14.jar -- even if operating on Oracle 9 server.

STEP 3.3: 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. You can find them here: http://java.sun.com/products/javamail/. Add them to Tomcat `common/lib` directory, or your server's equivalent location.

UNIX example

```
% cp mail.jar $CATALINA_HOME/common/lib
% cp activation.jar $CATALINA_HOME/common/lib
```

Windows example

```
c> copy mail.jar %CATALINA_HOME%\common\lib
C> copy 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.
- The Roller Support project at Java.Net offers some bundles that might make this part of the installation easier. It includes both mail.jar and activation.jar. Visit http://roller.dev.java.net for more information.

STEP 4: 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 webapp directory `$ROLLER/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`
- `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%\WEB-INF\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 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/apache-roller-3.0-
incubating/webapp/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/apache-roller-3.0-
incubating/webapp/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;useUnicod
e=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"
    maxThreads="150"
    minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false"
    redirectPort="8443"
    acceptCount="100"
    debug="0"
    connectionTimeout="20000"
    disableUploadTimeout="true"
    URIEncoding="UTF-8"  />
```

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
```

Where `${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/search-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 configuration override 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`). See the Roller User Guide for more information on configuring Roller's built-in aggregator.

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 B.

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
```

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
```

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
passwds.encryption.enabled=true

# etc, etc, etc ... any other properties you want to override
```

NOTE: Setting password encryption to true is a very good idea for new Roller sites, but if you are upgrading an existing Roller site you'll need to come up with a strategy for converting your old unencrypted passwords before you turn encryption on.

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 8.3: Verify the database dialet setting in the Hibernate configuration file

If you're using MySQL 4.X for Roller, then you can skip this step.

If you're using some other database with Roller, then you must check the Hibernate dialect setting in the file `WEB-INF/classes/hibernate.cfg.xml`. In that file, you'll see the section of code shown below. The line shown in bold is the current dialect setting. If you want to switch to another database, move the MySQLDialect line inside the `<!-- -->` comments and replace it with the line for your database of choice.

```
<!-- select SQL dialect, MySQL 3.X or 4.X by default -->
<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<!--
If you want to use HSQLDB, PostgreSQL, DB2 or Derby, Oracle, SQLServer then use
the right dialect
<property name="dialect">org.hibernate.dialect.HQLDBDialect</property>
<property name="dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="dialect">org.hibernate.dialect.DB2Dialect</property>
<property name="dialect">org.hibernate.dialect.DerbyDialect</property>
<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
<property name="dialect">org.hibernate.dialect.SQLServerDialect</property>

For MySQL 5.X, use the MySQL5 dialect and J/Connector 3.1.X
<property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
-->
```

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.

UNIX example

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

Windows example

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

If you installed Roller on Tomcat then your Roller start page URL is probably:

```
http://localhost:8080/roller
```

When Roller starts up, it will display a startup page that gives you instructions on how to complete the installation by creating a Roller user account, creating a weblog and designating the front-page weblog for the site. Follow those instructions and . . .

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: Upgrading an existing Roller installation

This document describes how to upgrade an existing installation to the latest release of Roller by upgrading the Roller database and replacing the old Roller files (which are typically found in `tomcat/webapps/roller`) with the new release. The steps are:

- STEP 1: Shutdown and backup your old Roller
- STEP 2: Install the new Roller
- STEP 3: Copy old configuration
- STEP 4: Upgrade the database
- STEP 5: Startup Tomcat

Important notes about Roller 3.0

WARNING! Roller 3.0 is a major release and makes some big changes to the way that Roller works. If you're upgrading you need to be aware of the new URL structure and the new template system.

The new URL structure

The most significant change is the new URL structure – we've completely change all of the Roller weblog URLs. We continue to support old Roller URLs, but they are redirected (using HTTP 301) redirects to the new system. That ensures that nobody will get a 404 when accessing your weblog using an old URL, but you'll still want to encourage people to change links that point to your weblog's old URL, which was of the format:

```
http://<hostname>/roller/page/<weblog-handle>
```

To use the new format, which is:

```
http://<hostname>/roller/<weblog-handle>
```

Unfortunately, some custom templates that use relative URLs will have problem with this new URL structure. So, before you go live you should set-up a test server and allow your users to take a look at their weblogs.

The new template system

We've developed a completely new and greatly improved template system for Roller, which includes new models and new macros. We want to encourage people to start using this new system for all new template and theme development, so we have made it the default. And, by default, we've turned off the old system.

So, if you are upgrading and you want your weblogs to work, you MUST override this Roller property to enable the old “legacy” template system to work:

```
rendering.legacyModels.enabled=true
```

See section 8 for information on overriding Roller startup properties.

Non-core themes removed

The Roller project is establishing a community site for sharing and maintaining of themes and plugins.

From now only, Roller will ship with only a core set of four themes (Basic, BrushedMetal, Sotto and a new Frontpage themes, which is just for site-wide front-page weblogs) . All other themes have been removed from Roller and moved to the Roller Support project at Java.net (http://roller.dev.java.net). If you or your users use any other themes, then you'll need to download and install them according to the instructions on that site.

Non-core plugins removed

The same applies to plugins. The JSPWiki, Textile and Read More plugins have also been moved to the Roller Support project at Java.net (http://roller.dev.java.net) to become part of a community maintained repository. If you or your users use any other themes, then you'll need to download and install them according to the instructions on that site.

UPGRADE STEP 1: Shutdown and backup your old Roller

Before you get started with your upgrade, you should shutdown your existing Roller install, make a backup of your data, and move the old Roller files out of the way. Here is an example of how you'd do this with a Tomcat/MySQL setup:

Run shutdown.sh to stop Tomcat, for example:

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

Backup your database to somewhere safe on your system or to a remote file-system, for example if you use MySQL you might do something like this:

```
% mysqldump -u scott -p rollerdb > \
/somewhere/safe/roller-backup-20050420.dmp
```

Here's an example for PostgreSQL users:

```
pg_dump -h 127.0.0.1 -W -U scott rollerdb > roller.db
```

Move your Roller files to somewhere safe, for example:

```
% cd $CATALINA_HOME/webapps
% mkdir /somewhere/safe/roller-old
% mv roller /somewhere/safe/roller-old
```

UPGRADE STEP 2: Install the new Roller

Follow the normal installation instructions to install Roller, except:

- Don't create a new database for Roller, instead point the new Roller to your old Roller database
- DO NOT start Tomcat when you are done with the installation, we'll do that later

UPGRADE STEP 3: Copy resources and update configs

3.1 Copy your old resources and other files you've added

User uploaded files are, by default, stored in the `/resources` sub-directory of the Roller context directory. You should copy your old resources directory into your new Roller installation.

For example, on UNIX you can use `cp -r` to copy the whole directory:

```
% cd %CATALINA_HOME/webapps/roller
% cp -r /somewhere/safe/roller-old/roller/resources .
```

NOTE: If you have any new themes under roller/themes, make sure to copy those as well.

3.2 Review configuration properties

Review properties as described in STEP #8 of the installation guide.

- **WARNING: In Roller 2.3 we changed the Roller package names from org.roller to org.apache.roller. Because of this you MUST review your roller-custom.properties file, search for the string "org.roller"and replace all occurances of it with "org.apache.roller".**

- **WARNING: The files required for Roller's JSPWiki plugin, Ekit editor and Javascript enhanced editor have been removed from the Roller distribution. If you'd like to continue to use these plugins, please visit the Roller Support project on Java.Net (http://roller.dev.java.net).**

UPGRADE STEP 4: Upgrade the database

Use the appropriate database upgrade script to upgrade your database. To do this, login to your database and run one (or more) of the Roller upgrade scripts located in Roller's `WEB-INF/dbscripts` directory that corresponds to your database. There's a directory for MySQL, PostgreSQL, HSQLDB and more.

The database script directories

- `WEB-INF/dbscripts/mysql`
- `WEB-INF/dbscripts/postgresql`
- `WEB-INF/dbscripts/hsql`
- and more...

There is an upgrade script for each release of Roller. If you're upgrading from Roller 2.3, which was the last release before 3.0 you'll need to run three scripts: `230-to-240-migration.sql`, `240-to-300-migration.sql` (Roller 2.4 was never officially released) and finally `300-to-310-migration.sql`. For example, here's how you'd do that for a MySQL database running on UNIX:

```
% cd $CATALINA_HOME/webapps/roller/WEB-INF/dbscripts/mysql
% mysql -u root -p
password: *****
mysql> use roller;
mysql> source 230-to-240-migration.sql
mysql> source 240-to-300-migration.sql
mysql> source 300-to-310-migration.sql
mysql> quit
```

If you're upgrading from an earlier release you'll have to run each of the older scripts in order to upgrade your database.

UPGRADE STEP 5: Startup your app server

Use the standard Tomcat `startup.sh` (or `startup.bat` on Windows) script to start Tomcat. As Roller starts up, it will perform some final steps to upgrade your database, this may take few seconds longer than your average Roller startup.

And you're done!

If Roller doesn't come up, check the logs for exceptions and error messages. You should see these messages in the `tomcat/logs/catalina.out` file and in `tomcat/logs/roller.log`. If you still can't diagnose and fix your startup program, then subscribe to the Roller user mailing list for help. If there are any interesting

messages in the log files, send those along too.

Appendix B: 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.

```
# Licensed to the Apache Software Foundation (ASF) under one or more
#  contributor license agreements.  The ASF licenses this file to You
# 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.  For additional information regarding
# copyright in this work, please see the NOTICE file in the top level
# directory of this distribution.

# roller.properties
#
# This file is for meant for Roller deployment properties
# Any properties that don't change at runtime are defined here
#
# You can override the values in this file in a couple ways ..
#   1. define a roller-custom.properties file and place it somewhere
#       at the root of one of your classpath locations.
#       for example:
#           $TOMCAT_HOME/common/classes
#           $ROLLER_LOCATION/WEB-INF/classes
#
#   2. specify a custom properties file via jvm option
#       example:
#           roller.custom.config=/path/to/properties/file
#
#   **NOTE: if you run multiple instances of roller on the same server
#           then you should beware of the fact that the override options above
#           may affect *all* of you running instances.  if you want to do
#           custom overrides for each instance then you must do it by
#           placing a roller-custom.properties file at WEB-INF/classes/ in
#           each of you roller installations.
#
# properties in this file are accessed like this ...
#    RollerConfig.getProperty("propname");

#---------------------------------
# Database configuration settings

# Hibernate dialect: You must override this to use a database other than MySQL4
hibernate.dialect=org.hibernate.dialect.MySQLDialect

#---------------------------------
# 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 resoures will be made available
uploads.url=/resources

#----------------------------------
# Themes settings

# The directory in which Roller will look for themes
themes.dir=${webapp.context}

#----------------------------------
# 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

#----------------------------------
# Rendering system settings.

# Are the old pre-3.0 models and macros enabled?
rendering.legacyModels.enabled=false

# The set of default Roller renderer factories.
rendering.rollerRendererFactories=\
org.apache.roller.ui.rendering.velocity.VelocityRendererFactory

# The set of user defined renderer factories.  These are prepended to the list above.
rendering.userRendererFactories=

# The set of default Roller request mappers
rendering.rollerRequestMappers=\
org.apache.roller.ui.rendering.WeblogRequestMapper

# The set of user defined request mappers.  These are prepended to the list above.
rendering.userRequestMappers=

# Url path elements which can NEVER be considered a weblog url
# each represents a url at the application root, i.e. /<elem>/*
rendering.weblogMapper.rollerProtectedUrls=\
roller-ui,images,theme,themes,CommentAuthenticatorServlet,\
index.jsp,favicon.ico,robots.txt,taglibs.jsp,\
page,flavor,rss,atom,language,search,comments,rsd,resource,xmlrpc,planetrss

# Set of user defined protected urls.  These are added to the set above.
```

```
rendering.weblogMapper.userProtectedUrls=

# Set of models to be made available for weblog page rendering
rendering.pageModels=\
org.apache.roller.ui.rendering.model.PageModel,\
org.apache.roller.ui.rendering.model.ConfigModel,\
org.apache.roller.ui.rendering.model.UtilitiesModel,\
org.apache.roller.ui.rendering.model.URLModel,\
org.apache.roller.ui.rendering.model.MessageModel,\
org.apache.roller.ui.rendering.model.CalendarModel,\
org.apache.roller.ui.rendering.model.MenuModel

# Set of models to be made available for weblog feed rendering
rendering.feedModels=\
org.apache.roller.ui.rendering.model.FeedModel,\
org.apache.roller.ui.rendering.model.ConfigModel,\
org.apache.roller.ui.rendering.model.UtilitiesModel,\
org.apache.roller.ui.rendering.model.URLModel,\
org.apache.roller.ui.rendering.model.MessageModel

# Set of models to be made available for weblog search rendering
rendering.searchModels=\
org.apache.roller.ui.rendering.model.SearchResultsModel,\
org.apache.roller.ui.rendering.model.ConfigModel,\
org.apache.roller.ui.rendering.model.UtilitiesModel,\
org.apache.roller.ui.rendering.model.URLModel,\
org.apache.roller.ui.rendering.model.MessageModel,\
org.apache.roller.ui.rendering.model.CalendarModel,\
org.apache.roller.ui.rendering.model.MenuModel

# Set of models to be made available for weblog page *preview* rendering
# NOTE: this *does* have some differences between the pageModels
rendering.previewModels=\
org.apache.roller.ui.rendering.model.PreviewPageModel,\
org.apache.roller.ui.rendering.model.ConfigModel,\
org.apache.roller.ui.rendering.model.UtilitiesModel,\
org.apache.roller.ui.rendering.model.PreviewURLModel,\
org.apache.roller.ui.rendering.model.MessageModel,\
org.apache.roller.ui.rendering.model.CalendarModel,\
org.apache.roller.ui.rendering.model.MenuModel

# Set of page models specifically for site-wide rendering
rendering.siteModels=\
org.apache.roller.ui.rendering.model.SiteModel

# Velocity settings
velocity.properties=/WEB-INF/velocity.properties

# Old velocity macro libraries
velocity.oldMacroLibraries=\
deprecated/roller.vm,deprecated/bookmark.vm,deprecated/comments.vm,\
deprecated/navbar.vm,deprecated/newsfeed.vm,deprecated/referer.vm,\
deprecated/atommacros.vm,deprecated/rssmacros.vm,deprecated/user.vm,\
deprecated/weblog.vm,deprecated/website.vm

#-----------------------------------
# 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.apache.roller.util.cache.ExpiringLRUCacheFactoryImpl
cache.customHandlers=

# set "true" to NOT cache the custom pages for users who are logged in
cache.excludeOwnerEditPages=false

# This sets how many minutes into the future we look to prepare
# entries posted into the future which need to be invalidated from the cache.
# It is very unlikely that this should ever need to be changed
cache.futureInvalidations.peerTime=3

# Site-wide cache (all content for site-wide frontpage weblog)
cache.sitewide.enabled=true
cache.sitewide.size=50
cache.sitewide.timeout=1800

# Weblog page cache (all the weblog content)
cache.weblogpage.enabled=true
cache.weblogpage.size=400
cache.weblogpage.timeout=3600

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

# Planet cache (planet page and rss feed)
cache.planet.enabled=true
cache.planet.size=10
cache.planet.timeout=1800

#----------------------------------
# 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=/j_security_check,/roller-ui/login-redirect.jsp,\
/roller-ui/login.do,/roller-ui/user.do,/roller-ui/yourProfile.do,\
/roller-ui/admin/user.do,/roller-ui/authoring/userdata

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

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

# Weblog entry plugins
plugins.page=\
org.apache.roller.ui.rendering.plugins.ConvertLineBreaksPlugin \
,org.apache.roller.ui.rendering.plugins.TopicTagPlugin \
```

```
,org.apache.roller.ui.rendering.plugins.ObfuscateEmailPlugin \
,org.apache.roller.ui.rendering.plugins.SmileysPlugin
#,org.apache.roller.ui.rendering.plugins.WikipediaLinkPlugin \
#,org.apache.roller.ui.rendering.plugins.GoogleLinkPlugin \
#,org.apache.roller.ui.rendering.plugins.AcronymsPlugin \
#,org.apache.roller.ui.rendering.plugins.BookmarkPlugin

# The list of configured WeblogEntryEditors available to users
plugins.weblogEntryEditors=\
org.apache.roller.ui.core.plugins.TextEditor,\
org.apache.roller.ui.core.plugins.XinhaEditor

# The "id" of the default editor to use.  NOT the class name
plugins.defaultEditor=editor-text.jsp

#----------------------------------
# Scheduled Background Tasks ... all times are in minutes.
#
# Task properties should follow the given format ...
#     tasks.<taskname>.<prop>=<value>
#
# The *enabled* tasks are defined by tasks.enabled=<taskname>[,<taskname>]
#----------------------------------

# Tasks which are enabled.  Only tasks listed here will be run.
tasks.enabled=ResetHitCountsTask,TurnoverReferersTask,PingQueueTask

# Reset hit counts
tasks.ResetHitCountsTask.class=org.apache.roller.business.runnable.ResetHitCountsTask
tasks.ResetHitCountsTask.startTime=startOfDay
tasks.ResetHitCountsTask.interval=1440
tasks.ResetHitCountsTask.leaseTime=30

# Reset referer counts
tasks.TurnoverReferersTask.class=org.apache.roller.business.runnable.TurnoverReferersTask
tasks.TurnoverReferersTask.startTime=startOfDay
tasks.TurnoverReferersTask.interval=1440
tasks.TurnoverReferersTask.leaseTime=30

# Ping processor, does sending of pings
tasks.PingQueueTask.class=org.apache.roller.business.pings.PingQueueTask
tasks.PingQueueTask.startTime=immediate
tasks.PingQueueTask.interval=5
tasks.PingQueueTask.leaseTime=30

# Sync Roller weblogs with planet
tasks.SyncWebsitesTask.class=org.apache.roller.planet.tasks.SyncWebsitesTask
tasks.SyncWebsitesTask.startTime=startOfDay
tasks.SyncWebsitesTask.interval=1440
tasks.SyncWebsitesTask.leaseTime=30

# Refresh entries for planet feeds
tasks.RefreshEntriesTask.class=org.apache.roller.planet.tasks.RefreshEntriesTask
tasks.RefreshEntriesTask.startTime=startOfHour
tasks.RefreshEntriesTask.interval=60
tasks.RefreshEntriesTask.leaseTime=30

# Technorati rankings for planet feeds
tasks.TechnoratiRankingsTask.class=org.apache.roller.planet.tasks.TechnoratiRankingsTask
tasks.TechnoratiRankingsTask.startTime=startOfDay
tasks.TechnoratiRankingsTask.interval=1440
tasks.TechnoratiRankingsTask.leaseTime=30
```

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

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

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

# comment authenticator settings (experimental)
authenticator.classname=org.apache.roller.ui.core.DefaultAuthenticator

comment.authenticator.classname=org.apache.roller.ui.rendering.util.MathCommentAuthentica
tor
comment.notification.separateOwnerMessage=false
comment.notification.hideCommenterAddresses=false
comment.throttle.enabled=false
comment.throttle.threshold=25
comment.throttle.interval=60
comment.throttle.maxentries=250

# 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/}}\
,{{IceRocket}{http://rpc.icerocket.com:10080/}}

# Specify variant options for known buggy ping targets.
pings.variantOptions=\
{{http://rpc.icerocket.com:10080/}{noname}}


# 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:
# - 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
# - You'll be able to add the $planet model to the list of page models available
#   to blogs (rendering.pageModels) or to only site-wide blogs (rendering.siteModels)
#   (classname org.apache.roller.ui.rendering.model.SiteModel)
#
planet.aggregator.enabled=false

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

# Number of queries allowed per day
planet.aggregator.technorati.limit=500

#----------------------------------
# defaults for new weblogs

# 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,\
Elias Torres|http://torrez.us/,\
Jeff Blattman|http://blogs.sun.com/jtb,\
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

# Default weblog editor
# The list of available editors is in rollerRuntimeConfigDefs.xml
newweblog.editor=editor-text.jsp


#----------------------------------
# Single-Sign-On

# Enables Roller to behave differently when registering new users
# in an SSO-enabled environment. You must configure security.xml appropriately.
users.sso.enabled=false

# Set these properties for a custom LDAP schema (optional)
#users.sso.registry.ldap.attributes.name=cn
#users.sso.registry.ldap.attributes.email=mail
#users.sso.registry.ldap.attributes.locale=locale
#users.sso.registry.ldap.attributes.timezone=timezone

# If you don't want user credentials from LDAP/etc to be stored in Roller
# (possibly in clear-text) leave this alone, otherwise set to true.
# i.e. you would like a backup auth mechanism in case LDAP is down.
users.sso.passwords.save=false

# if you don't want passwords stored in DB, set this to the default value.
users.sso.passwords.defaultValue=<unknown>

users.sso.autoProvision.enabled=false
users.sso.autoProvision.className=org.apache.roller.ui.core.security.BasicUserAutoProvisi
on

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

```
# Characters to be allowed in user names (change at your own risk)
username.allowedChars=A-Za-z0-9

rememberme.enabled=true
debug.memory.enabled=false
compression.gzipResponse.enabled=true

# specifies the max number of tags allowed in URL ( /feed?tags=foo+bar+baz )
tags.queries.maxIntersectionSize=3

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

# Hibernate config resource (a classpath-based path)
# NO NEED TO OVERRIDE this unless you are customizing Roller
hibernate.configResource=/hibernate.cfg.xml

# JDBC configuration parameters ONLY NEEDED FOR RUNNING STANDALONE TASKS
# Don't override these in the roller-custom.properties file you use with the
# Roller webapp, but for the standalone tasks that you run outside of Roller
# (e.g. refresh entries) you'll need to override these properties. Do it in a
# separate roller-custom.properties file.
jdbc.driverClass=
jdbc.connectionURL=
jdbc.username=
jdbc.password=

#----------------------------------
# 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.
plugins.topictag.defaultTopicBookmarkName=Default Topic Site
plugins.topictag.defaultTopicSite=http://www.technorati.com/tag
plugins.topictag.tagRegexWithBookmark=topic:\\{(.*?)\\}\\[(.*?)\\]
plugins.topictag.tagRegexWithoutBookmark=topic:\\[(.*?)\\]
plugins.topictag.linkFormatString=<a rel=\"tag\" href=\"{0}{1}\">{2}</a>

# Set to true to allow only default topic tag site (and avoid costly bookmark queries)
plugins.topictag.ignoreBookmarks=true

#----------------------------------
# Experimental settings

# Atom Publishing Protocol (APP) - this is an incomplete and untested
# implementation of an unfinished IETF specification.
# Intended only for interoperability testing. DO NOT ENABLE IN PRODUCTION!
webservices.atomprotocol.enabled=false

# Atom-like Admin Publishing Protocol (AAPP) - this is an experimental admin
# protocol based on ideas from the Atom protocol.
# Intended only for interoperability testing. DO NOT ENABLE IN PRODUCTION!
webservices.adminprotocol.enabled=false

#----------------------------------
# legacy settings (things that should be deprecated)

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