Installation Guide

Table of Contents

1. Overview

This document describes how to install the Apache Roller Weblogger software. It explains what you need to install first, how to download Roller, how to configure Roller and how to install it to your existing Java application server and relational database.

1.1. Copyright and trademark information

1.2. Feedback

Please direct any comments or suggestions about this document to the Roller User Mailing list. For more information on the Roller mailing lists please refer to the following page:

Roller Mailing Lists - *https://cwiki.apache.org/confluence/x/ZYk*

1.3. Acknowledgments

The original version of this document was written by Dave Johnson. The document is currently written and updated by the Apache Roller project of the Apache Software Foundation.

2. Securing Roller

Security is crucial when setting up any website, even on a private network. Here are some recommendations to keep your Roller installation secure:

- **Install Roller on a secure network**. During installation, other users could interfere or access the Roller database or files. To prevent this, install Roller on a secure network or when the server is not in use by others.
- **Disable new user registrations**. By default, Roller allows self-registration, which means anyone can create an account. To prevent this, disable the **Allow New Users** option on the Server Administration page.
- **Use SSL for Roller**. Running Roller over HTTP can expose your password to snooping, especially on open WIFI networks. To configure SSL (https:// URLs), modify the web.xml in the Roller WAR (WEB-INF folder) by uncommenting the <security-constraint/> element and following the instructions. Then, follow your servlet container's SSL setup documentation (e.g., http://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html for Tomcat). Redeploy Roller and ensure secure pages like the login and registration pages are accessible only via https:// URLs.

Following these recommendations will help secure your Roller installation against common web vulnerabilities.

2.1. Safer defaults

As of Roller 6.1.4, several default settings have been updated to enhance security for multi-user weblog sites:

- **HTML content sanitization**: Roller now sanitizes all HTML content by default to prevent malicious content. This is controlled by the *weblogAdminsUntrusted=true* property in your

roller-custom.properties file.

- **Custom themes disabled**: By default, users cannot create custom themes. This can be enabled via the Server Admin page if you trust your users, as custom themes can pose security risks.
- **File uploads disabled**: By default, file uploads are not allowed. If you trust your users, you can enable this feature via the Server Admin page.

If you are a solo blogger, you can safely enable un-sanitized HTML, file uploads, and custom themes by adjusting the above settings.

3. Ready to roll?

First, let's make sure you have everything you need to install and run Roller.

Roller is a database-driven Java web application. To run it you need Java, a Java Servlet container such as Tomcat, a connection to a database such as MySQL and optionally a connect to a mail server. More specifically, here's what you need to install and run Roller:

- **Java Development Kit**, specifically the Java 2 SE 1.6 JDK or more recent. The computer on which you install Roller should already have the JDK installed and configured.
- **Java EE 6 Application Server**, or more specifically a Servlet container that supports at least the Servlet 2.4 API. Hereinafter, we'll just call this your *server*. Roller has traditionally worked best on Tomcat and Tomcat only, but Roller 6.0.0 is known to run on:
  - Tomcat 8
  - Jetty 9
- **Relational database** such as MySQL or Apache Derby. Roller stores blog entries, comments, bookmarks and almost all other data in a relational database accessed via the Java Persistence API 2.0. PostgresSQL, MySQL and Derby are best supported but Roller. Roller also includes database creation scripts for DB2, HSQL-DB, Microsoft SQL Server and Oracle but these have not been tested recently.
- **(Optional) An SMTP mail server**. Roller can send email notifications for comments and other events via the JavaMail and Activation APIs.
- **Roller installation file**. The Roller installation file contains the Roller WAR file, ready to deploy to your server, plus Roller license files, README and documentation. Unpack using either file based on a compression method your operating system supports:
  - roller-release-6.0.0-standard.zip
  - roller-release-6.0.0-standard.tar.gz

This step can also be done after deployment and determination that you would like themes other than the defaults available available, just modify the WAR and redeploy on your servlet container.

- **(Optional) Additional blog themes**. Roller comes pre-packaged with several blog themes (templates) for you to choose and optionally customize for each blog you create. You may wish to add additional themes to the Roller WAR file so they will be available to choose from when you deploy the application. To do this, just open the Roller WAR and add the theme to the

"themes" folder located at the top level of the WAR. Google <Apache Roller Themes> and/or check the non-Apache resources section of the Roller wiki page (https://cwiki.apache.org/confluence/display/ROLLER/Roller+Wiki) for any externally available themes—external themes are not supported by the Roller team, however.

4. Download and un-package Roller

Download the Apache Roller release file from http://roller.apache.org. If you're a Windows user download the .zip file and use your favorite ZIP program to unzip the release into a directory on your computer's disk. Unix users can download the .tar.gz file and use GNU tar to un-package.

4.1. Installation directory layout

Once you've unpackaged the files you'll find a directory structure like this:

```
README.txt

LICENSE.txt

NOTICE.txt

docs/

    roller-install-guide.pdf

    roller-user-guide.pdf

    roller-template-guide.pdf

webapp/

    roller.war
```

The *LICENCE.txt* and *NOTICE.txt* files contain the Apache Software License and other legal notices related to the release. The *README.txt* file just points to the documentation in the *docs* directory.

https://cwiki.apache.org/confluence/display/ROLLER/Roller+Wiki

5. Prepare your database for Roller

Before you can install Roller you'll probably need to some work to prepare your database. You'll need to create a place to put the Roller tables; some call this a table-space and we refer to it as a *database* in this installation guide. You'll need to create a database for Roller, or get your database administrator to do it for you. You also need to have a JDBC driver for your database of choice, but we'll cover that later.

5.1. Create a database for Roller

If you're luck enough to have your own database administrator, ask them to setup a database for Roller. When they are done, ask them to provide you with this information, you'll need it later:

- Username and password for connecting to database
- JDBC connection URL for database
- JDBC driver class name

If you don't have a database administrator then you'll have to refer to the documentation for your database and do it yourself. You need to create a database for Roller, protected by username and password. For example, if you're using MySQL you might do something like this (be sure to use a different username and password from the scott/tiger below):

```
% sudo service mysql start

% mysql -u root -p

password: *****

mysql> create database rollerdb DEFAULT CHARACTER SET utf8 DEFAULT
COLLATE utf8_general_ci;

mysql> grant all on rollerdb.* to scott@'%' identified by 'tiger';

mysql> grant all on rollerdb.* to scott@localhost identified by 'tiger';

mysql> exit;
```

If you're using Derby:

```
% ij

ij> connect 'jdbc:derby:/path/to/new/MYROLLERDB;create=true';

ij> quit;
```

For PostgreSQL:

```
%sudo -u postgres psql postgres

postgres=# create user scott createdb;

postgres=# \du (see list of users and roles)

postgres=# \password scott

Enter new password: ?????

postgres-# \q

%createdb -h localhost -U scott -W pgsqlroller -E UTF8
```

5.2. MySQL and Oracle considerations

Based on our experience supporting MySQL, we have the following recommendations:

- For MySQL, make sure that TCP/IP networking is enabled.
- For MySQL, UTF-8 must be enabled for your database, as done in the "create database rollerdb" command above or server-wide (http://dev.mysql.com/doc/refman/5.6/en/charset-applications.html).

  If a non-UTF8 database has already been created you can switch the database to UTF-8 as follows providing the tables have **not** already been created:

  ```
  ALTER DATABASE roller DEFAULT CHARACTER SET utf8 COLLATE
  utf8_general_ci;
  ```

- For Oracle users, use the 10g (10.1.0.2 higher) drivers which should be packaged as ojdbc14.jar, even if operating on Oracle 9 server.
- See the server specific sections to information on where to place the JDBC driver jars.

6. Deploy Roller to Tomcat

Deploying Roller to the Tomcat servlet container involves creating a Roller configuration file, adding some jars to Tomcat and then deploying the Roller WAR file.

You are expected to install and configure Apache Tomcat before you attempt to install Roller, and be aware of how to deploy a WAR archive on Tomcat. Refer to the Tomcat documentation linked from this page for more information: *http://tomcat.apache.org*

6.1. Tomcat: Create Roller Configuration File

There are a variety of ways to configure Roller and Tomcat and here we'll explain the easiest route:

providing database and mail connection information directly to Roller via the Roller configuration file.

Create the Configuration File

For most settings, Roller can be configured from its own web console. But for some startup-properties and advanced configuration options you must set properties in an override file called:

roller-custom.properties

That is a simple Java properties file, a text file that overrides settings defined in Roller's internal *roller.properties* file. To configure Roller you look at Roller's internal properties file, decide which properties you need to override and then set those in your *roller-custom.properties* file.

The precise roller.properties file your distribution is using is located in /WEB-INF/classes/org/apache/roller/ weblogger/config/ within the roller.war file. It is also viewable online at http://svn.apache.org/viewvc/roller/trunk/app/src/main/resources/org/apache/roller/weblogger/config/roller.properties, click the "(view)" button at a revision just prior to the Roller release you're using. We encourage you to look through this file to determine other properties you may wish to override, but we'll get you started right here and now with a simple example that shows you the minimum startup, database, and mail configuration settings that you need to run Roller. You'll need to alter this information using settings appropriate for your filesystem, database, and mail server. (Also note the database and mail configuration shown below will be done differently if you're using JNDI, which will be discussed in the next section. JNDI, in particular, is presently required if your mail SMTP server requires authentication.)

Example: *roller-custom.properties* file

```
installation.type=auto

mediafiles.storage.dir=/usr/local/rollerdata/mediafiles

search.index.dir=/usr/local/rollerdata/searchindex

log4j.appender.roller.File=/usr/local/rollerdata/roller.log

database.configurationType=jdbc

database.jdbc.driverClass=com.mysql.jdbc.Driver

database.jdbc.connectionURL=jdbc:mysql://localhost:3306/rollerdb?autoReconnect=true&us
eUnicode=true&characterEncoding=utf-8&mysqlEncoding=utf8

database.jdbc.username=scott

database.jdbc.password=tiger

mail.configurationType=properties

mail.hostname=smtp-server.example.com

mail.username=scott

mail.password=tiger
```

The *installation.type=auto* property tells Roller to operate in automatic installation mode. In this mode Roller will provide very detailed error output to help you debug database connection problems. If Roller finds that the database exists but its tables are not, it will offer to run the database creation scripts. If find finds that the tables are there, but they are not up-to-date Roller will offer to upgrade them for you. Once your Roller installation is complete and you are ready to go "live" then you should set *installation.type=manual*.

The above sample *roller-custom.properties* uses a MySQL connection. It shows the MySQL JDBC driver class name, an example MySQL connection URL and username/password settings for the mail connection:

If you're using Derby, database configuration properties similar to the following will be more appropriate. Note authentication is not used by default with Derby (any username and password provided below will be accepted), see http://db.apache.org/derby/docs/10.2/tuning/rtunproper27467.html on how to require authentication with Derby. The username configured below will be the table owner used when the Roller installation process later creates the database tables.

```
database.configurationType=jdbc

database.jdbc.driverClass=org.apache.derby.jdbc.EmbeddedDriver

database.jdbc.connectionURL=jdbc:derby:/path/to/new/MYROLLERDB

database.jdbc.username=app

database.jdbc.password=app
```

For PostgreSQL:

```
database.configurationType=jdbc

database.jdbc.driverClass=org.postgresql.Driver

database.jdbc.connectionURL=jdbc:postgresql://localhost:5432/pgsqlroller

database.jdbc.username=scott

database.jdbc.password=tiger
```

Alternative Authentication Options

The above instructions rely on Roller's default user authentication mechanism, i.e., using a Roller-provided database table (roller_user) to store usernames and encrypted passwords. Roller provides other authentication options defined under the "authentication.method" setting in the roller.properties file: OpenID, OpenID/DB combination, and LDAP (https://cwiki.apache.org/confluence/display/ROLLER/Roller+5.1+with+LDAP). These authentication methods are used less frequently so should be tested more thoroughly with your particular setup if you wish to use them. Check the roller.properties file included in your WAR for available options and configuration information, and consult the Roller User's Mailing List should you need assistance.

Add Configuration file to Tomcat

Place the configuration file into the Tomcat lib directory so that it is on the Tomcat classpath and therefore available to Roller.

6.2. Using Server-provided database & mail resources (optional)

It's easiest to setup your Roller for Tomcat database connection using the 'jdbc' approach and the mail connection using 'properties' but in some cases you might want to use the datasource and/or mail session resources provided by your application server instead. For authentication-requiring mail connections like Google's Gmail service, JNDI is presently required. For databases, you might use JNDI to take advantage of the database connection pool management that is built into your server. Or, your boss might want everything to be managed via your server's Admin Console. No

matter the reason, it's easy to do in Roller.

Here, you omit the *roller-custom.properties* database and mail configuration given in the previous section and replace it with just:

```
installation.type=auto
mediafiles.storage.dir=/usr/local/rollerdata/mediafiles
search.index.dir=/usr/local/rollerdata/searchindex
log4j.appender.roller.File=/usr/local/rollerdata/roller.log

database.configurationType=jndi
database.jndi.name=jdbc/rollerdb
mail.configurationType=jndi
mail.jndi.name=mail/Session
```

The *database.configurationType=jndi* setting tells Roller to look up its datasource via Java Naming and Directory Interface (JNDI). Roller will look for a datasource with the JNDI name *jdbc/rollerdb*. You must set that datasource up in your server.

The *mail.configurationType=jndi* setting tells Roller to look up it's mail sessions via JNDI. Roller will look for a mail session provider with the JNDI name *mail/Session*. You must set that provider up in your server. Let's discuss how to do that on Tomcat.

Setting up database and mail resources on Tomcat

There are a couple of different ways to setup database and mail resources on Tomcat. One way is to provide a Context Configuration file. Here's how to do this on Tomcat.

Before you deploy Roller to Tomcat, create a new Context Configuration file in the installation directory *webapp/roller/META-INF*. You'll find an example configuration file there, shown below. Rename it from *context.xml-example* to *context.xml* and substitute the correct values for driverClassName, url, username, password in 'jdbc/rollerdb' Resource and mail.smtp.user password in 'mail/Session' Resource.

```
<Context path="/roller" debug="0">
    <Resource name="jdbc/rollerdb" auth="Container" type="javax.sql.DataSource"
            driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/rollerdb?autoReconnect=true&amp;useUnicode=true&amp;c
haracterEncoding=utf-8&amp;mysqlEncoding=utf8"
            username="scott"
            password="tiger"
            maxActive="20" maxIdle="3" removeAbandoned="true" maxWait="3000" />
    <Resource name="mail/Session" auth="Container" type="javax.mail.Session"
            mail.transport.protocol="smtp"
            mail.smtp.host="smtp.gmail.com"
            mail.smtp.port="465"
            mail.smtp.auth="true"
            mail.smtp.user="blah.blah@gmail.com"
            password="yourgmailpassword"
            mail.smtp.starttls.enable="true"
            mail.smtp.socketFactory.class="javax.net.ssl.SSLSocketFactory"
            mail.smtp.socketFactory.port="465"
            mail.smtp.socketFactory.fallback="false"
            mail.debug="false"/>
</Context>
```

The Java mail properties listed above are defined here: https://javamail.java.net/nonav/docs/api/com/sun/mail/smtp/package-summary.html. Note the email account defined above will appear in the "From:" line of notification email messages sent to blog owners (and, if they select "Notify me of further comments", blog commenters) so take care not to use an email account you wish to keep private.

Another method is to the add the configuration to the Tomcat server.xml file under the correct host value already present in the file. (The Tomcat project advises against this method as it requires restarting the server whenever changes are made to this file, see http://tomcat.apache.org/tomcat-7.0-doc/config/context.html#Defining_a_context.) For example, with the same mail connection as above:

```
<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

  <Context
    path="/roller"
    docBase="roller"
    antiResourceLocking="false">

    <Resource name="mail/Session"
      auth="Container"
      type="javax.mail.Session"
      mail.transport.protocol="smtp"
      mail.smtp.host="smtp.gmail.com"
      ⋯rest of properties same as above⋯
    />
  </Context>
</Host>
```

6.3. Tomcat: Add JDBC Driver and JavaMail API Files

You will also need to place some additional jars in the Tomcat *lib* directory:

- **JDBC Driver Jars.** Add the appropriate JDBC driver jars to the Tomcat classpath. Once they are in your classpath, Roller's database subsystem will be able to find and use them. Download them from your database vendor/provider and place them in Tomcat's *lib* directory.
- **Java Mail and Activation.** Tomcat does not include the Java Mail and Activation jars. Even if you do not plan to use email features, you must download those jars and place them in Tomcat's classpath. Download them from Oracle (https://java.net/projects/javamail/pages/Home) and place them in Tomcat's *lib* directory.

6.4. Tomcat: Set URI Encoding

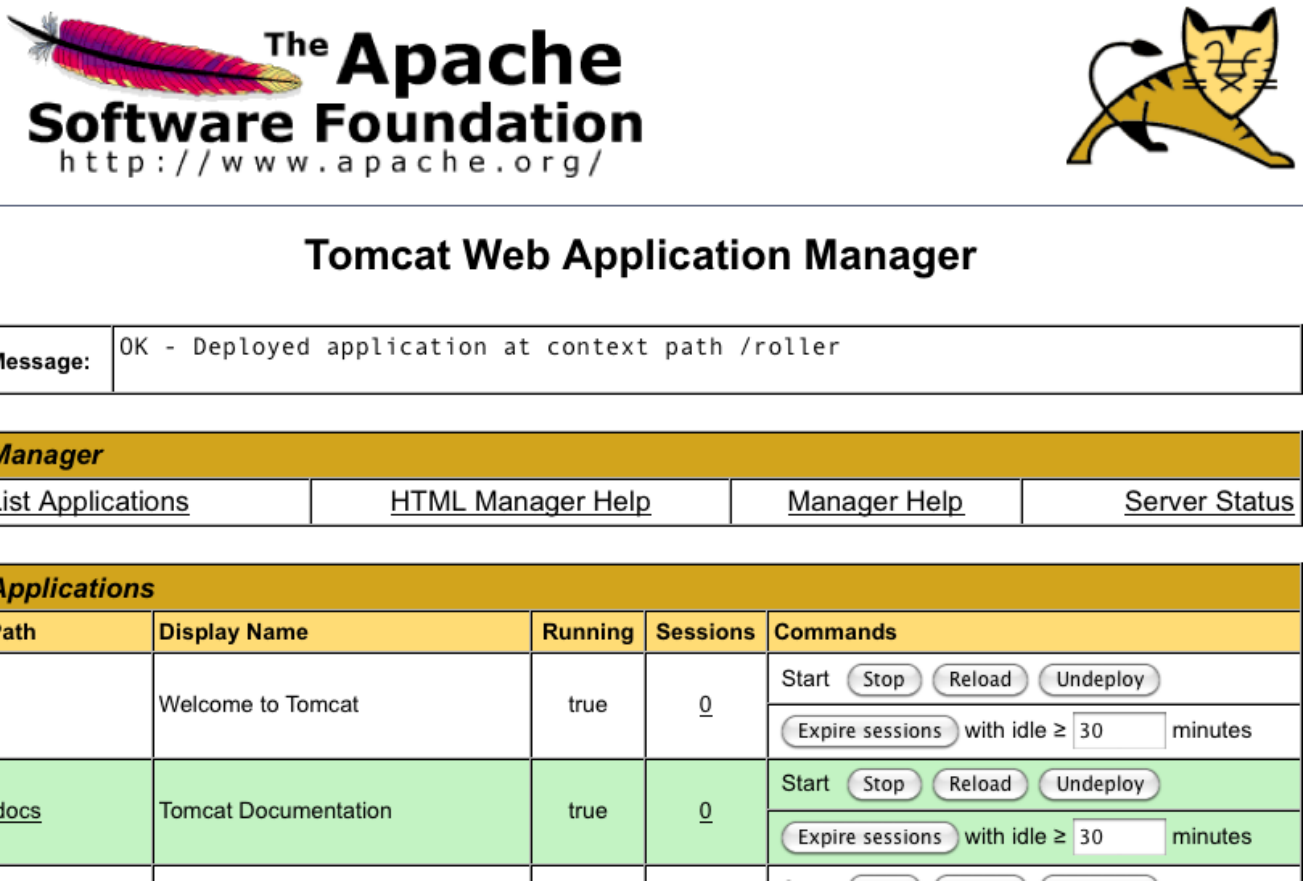
Roller supports internationalization (I18N), but on Tomcat some additional configuration is necessary. You must ensure that Tomcat's URI encoding is set to UTF-8. You can do this by editing the Tomcat configuration file conf/server.xml and adding URIEncoding="UTF-8" to each connector element, as shown below:

```
<Connector port="8080" maxThreads="150" minSpareThreads="25"
maxSpareThreads="75" enableLookups="false" redirectPort="8443" debug="0"
acceptCount="100" connectionTimeout="20000" disableUploadTimeout="true"
URIEncoding="UTF-8"/>
```

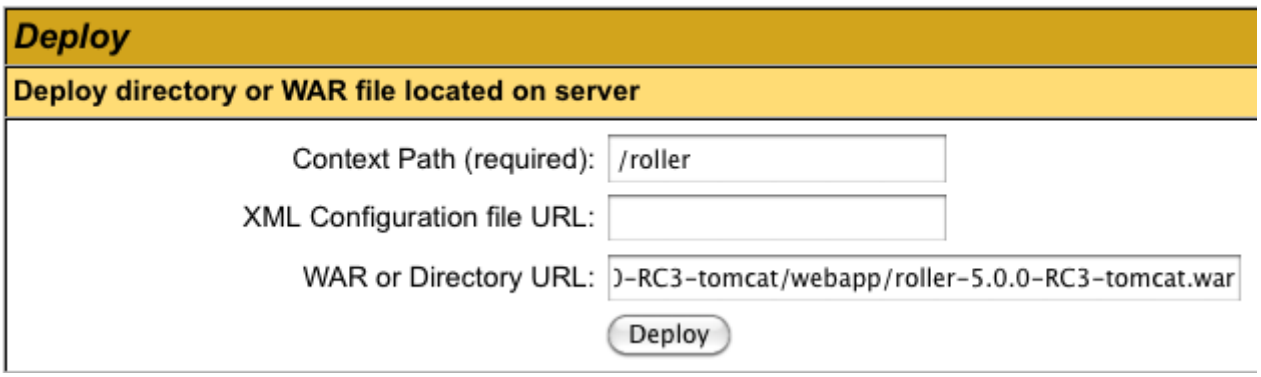
And make sure you do this for *every* connector through which you use Roller. For example, if you use the AJP connector or HTTPS connector you need to add the URIEncoding="UTF-8" attribute to those connectors as well.

6.5. Tomcat: Deploy Roller

Refer to the Tomcat documentation for information on the various ways to deploy a WAR. By renaming the Roller WAR to roller.war and placing it in the webapps directory of a running Tomcat instance, you should be able to access Roller at http://localhost:8080/roller (the /roller portion comes from the name of the WAR.) Another way to do this is to use the Tomcat Manager application, which you can reach at the following URL http://localhost:8080/manager. Once you are there, you'll see something like this:



On the manager screen above, scroll down until you see the **Deploy** section, see below:



Enter the context path at which you would like to see Roller, above we use */roller*. Enter the full path to the Roller WAR file, in the webapps directory of the Roller installation and click **Deploy** to deploy Roller.

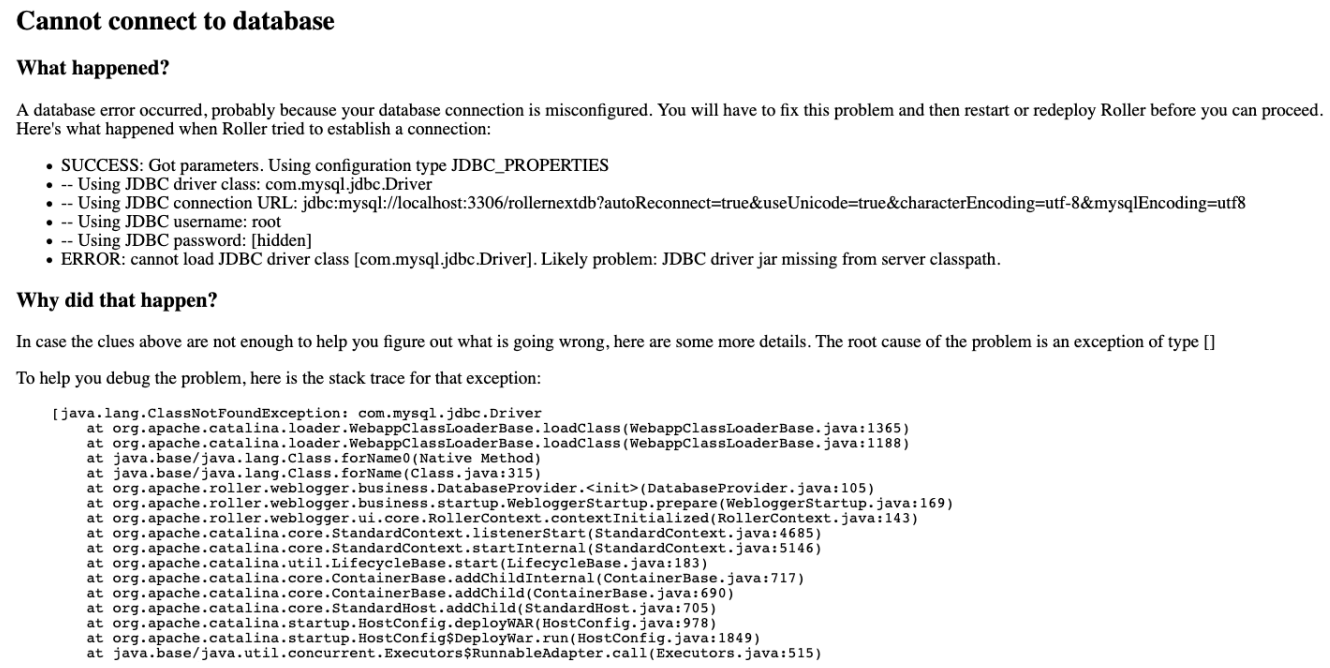
Finally, navigate to http://localhost:8080/roller to complete the installation.

7. Getting started with Roller

You're not quite done with the installation process, but now you're ready to start using Roller, so we'll walk you through getting started, registering a user and setting up a blog. We'll also discuss briefly what happens when there is an error.

7.1. Navigate to Roller and finish the install

Navigate to Roller, if you are using a default Tomcat or Glassfish installation then then URL of Roller is probably http://localhost:8080/roller. You will see either a web page of error messages, a web page offering to create database tables for you or web page asking you to complete the installation by registering an admin user and creating a front-page blog. First, let's talk about what happens when things go wrong.



Cannot connect to database

What happened?

A database error occurred, probably because your database connection is misconfigured. You will have to fix this problem and then restart or redeploy Roller before you can proceed. Here's what happened when Roller tried to establish a connection:

- SUCCESS: Got parameters. Using configuration type JDBC_PROPERTIES
- -- Using JDBC driver class: com.mysql.jdbc.Driver
- -- Using JDBC connection URL: jdbc:mysql://localhost:3306/rollernextdb?autoReconnect=true&useUnicode=true&characterEncoding=utf-8&mysqlEncoding=utf8
- -- Using JDBC username: root
- -- Using JDBC password: [hidden]
- ERROR: cannot load JDBC driver class [com.mysql.jdbc.Driver]. Likely problem: JDBC driver jar missing from server classpath.

Why did that happen?

In case the clues above are not enough to help you figure out what is going wrong, here are some more details. The root cause of the problem is an exception of type []

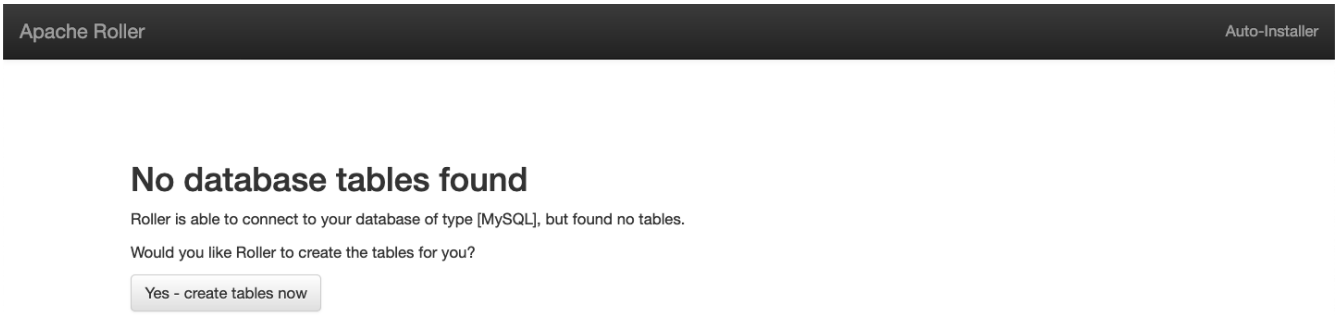
To help you debug the problem, here is the stack trace for that exception:

```
[java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
    at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1365)
    at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1188)
    at java.base/java.lang.Class.forName0(Native Method)
    at java.base/java.lang.Class.forName(Class.java:315)
    at org.apache.roller.weblogger.business.DatabaseProvider.<init>(DatabaseProvider.java:105)
    at org.apache.roller.weblogger.business.startup.WebloggerStartup.prepare(WebloggerStartup.java:169)
    at org.apache.roller.weblogger.ui.core.RollerContext.contextInitialized(RollerContext.java:143)
    at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:4685)
    at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5146)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:717)
    at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:690)
    at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:705)
    at org.apache.catalina.startup.HostConfig.deployWAR(HostConfig.java:978)
    at org.apache.catalina.startup.HostConfig$DeployWar.run(HostConfig.java:1849)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
```

If there's a problem with your database configuration, Roller will display a page or error messages to help you diagnose the problem. It's possible that you entered the wrong JDBC driver class name, connection URL, username or password. Or perhaps your database is not running. Use the information provided to determine what is wrong, fix it and then redeploy Roller.

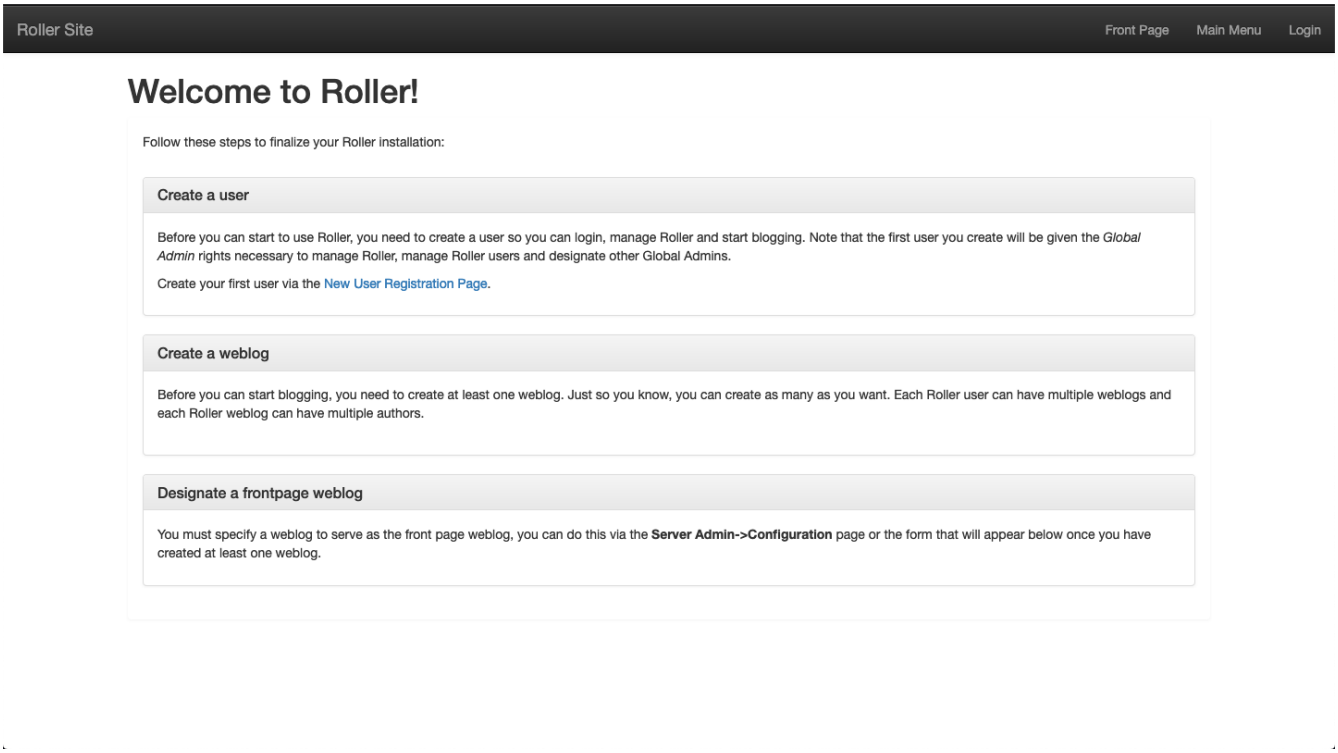
Automatic tables creation

If your database configuration is good but Roller cannot find its database tables, then Roller will offer to create those pages automatically for you. If you give the go-ahead, Roller will run the appropriate database creation script for your database and then show you the results. You can then proceed to the next step to setup your first user account and weblog.



7.2. Register a user and create a weblog

If Roller starts up fine but doesn't find a front-page weblog then it will display the Completing Your Installation below that explains how to register your first user, create your first weblog and setup your site's front page.



You have to decide what you want as the front-page of your Roller site. If you are using Roller to run your personal weblog, then you probably want your weblog to be the front-page of the site. In this case, create a weblog for yourself, *don't* choose the front-page theme but *do* set your weblog as the front-page weblog for the site.

If you are using Roller to run a community of multiple weblogs, then you'll probably want to display an aggregated front-page combining all weblogs on the site. In that case, create a weblog to serve as the front-page, set it as the front-page weblog and make sure you set the "aggregated front-page" setting on the Server Admin page.

Don't forget: Reset the *installation.type* flag

Now that you're done with the installation you should turn off Roller's auto-installation system. Edit your *roller-custom.properties* file and set *installation.type=manual*. Then restart your server or Roller so that it accepts the new setting.

What's next?

Once you've gotten Roller up and running refer to the Roller User Guide for more information on running your Roller system and your weblog. For information on customizing your weblog, refer to the Roller Template Guide. If you can't find what you want in the documentation then subscribe to the Roller user mailing list and ask your questions there:

https://cwiki.apache.org/confluence/display/ROLLER/Roller+Mailing+Lists

8. Configuration tips and tricks

This section covers some tips and tricks that can help you get the most out of Roller. It covers Roller's Planet feed aggregator and how to setup Roller to use server-provided resources.

8.1. Setting up Roller's Planet feed aggregator

Roller includes a RSS/Atom feed aggregator that makes it possible to run a site like https://blogs.oracle.com/ which provides weblogs for thousands of writers and an aggregated front-page that displays the most recent posts from those plus dozens of Sun bloggers from other sites such as blogger.com, typepad.com and other services. Here's what you need to do.

8.1.1. STEP 1: Create a Planet cache directory

Roller Planet needs a cache directory in which to store the feeds it fetches. By default, Roller Planet will put it's cache in your home directory under *roller_data/planetcache*. If you want to place the cache somewhere else, you must override the planet.aggregator.cache.dir property in your *roller-custom.properties* file. For example:

```
cache.dir=c:\\roller_data\planetcache
```

Whether you override that property or not, **you must create the cache directory**. Planet will not work unless the cache directory exists and is writable by Roller.

8.1.2. STEP 2: Enable Planet via Roller custom properties

Enable Planet by adding the following to your *roller-custom.properties* file:

```
planet.aggregator.enabled=true

# Tasks which are enabled.  Only tasks listed here will be run.
tasks.enabled=ScheduledEntriesTask,ResetHitCountsTask,\
PingQueueTask,RefreshRollerPlanetTask,SyncWebsitesTask

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

Those property settings enable Planet and enable the Planet tasks, both the *RefreshRollerPlanetTask*, which runs every hour and fetches all RSS/Atom feed subsciptions, and the *SyncWebsitesTask*, which runs every midnight and ensures that each weblog in the Roller system is represented by a subscription in the Planet aggregator. To enable usage of the PlanetModel in the front-page weblog, we also override the *rendering.siteModels* property.

8.1.3. STEP 3: Configure Planet via Planet custom properties

Create a new file called *planet-custom.properties* and place it in the same directory as your existing *roller-custom.properties* file. In this configuration file, add a property called *cache.dir* that points to the directory that you'd like Planet to use for caching it's RSS and Atom newsfeeds. The default setting is:

```
cache.dir=${user.home}/roller_data/planetcache
```

Once you've made those property settings restart Roller and proceed to the next step.

8.1.4. STEP 4: Display your Planet aggregations

You can use Roller's UI to add external RSS/Atom feeds to the Planet setup. To display these feeds you'll need to do a little template customization. The easier way to get started is to Roller's existing Front-Page theme. Here's how.

Create a weblog to server as the front-page of your Roller site. Start with the Front-Page theme and customize it. Edit the weblog template and look for the part that mentions PLANET-entries. Comment-out the SITE-WIDE part and un-comment the PLANET-entries part. The double hash "##" marks indicate a commented-out line. The code should look like this:

```
## 1) SITE-WIDE entries (the default)
##set($pager = $site.getWeblogEntriesPager($since, $maxResults))

## 2) PLANET-entries
#set($pager = $planet.getAggregationPager($since, $maxResults))
```

With that in place, your front-page will be display your Planet entries. You can find your Planet feeds at the following URLs:

- Main Planet feed http://localhost:8080/roller/planetrss
- Per group feed http://localhost:8080/roller/planetrss?group=<group-name>

8.2. Manual table creation and upgrade

If you would rather create your database tables yourself instead of letting Roller do it automatically, you can. Instead of enabling automatic installation you should disable it by putting this in your *roller-custom.properties* file:

installation.type=manual

Now you've got to run the database creation script. You can find the database creation scripts in the *webapp/roller/WEB-INF/classes/dbscripts* directory. You'll find a *createdb.sql* script for each of the databases we hope to support.

If you are upgrading Roller, you'll have to run the migration scripts instead of createdb.sql. You'll find those under the *dbscripts* directory too. However, the migration script should probably be run statement-by-statement checking the database responses as you go along, or alternatively by first removing any delete index or delete foreign key statement that you know doesn't exist in your database. Certain databases like MySQL throw errors when one attempts to delete objects such as foreign keys or indexes that don't already exist, a specific error type which the automated installation process is coded to ignore.

9. Upgrading Roller

This section describes how to upgrade an existing Roller installation to the latest release of Roller by shutting down, backing up and then following the installation instructions with a couple of key exceptions. But first, there is some required reading for those upgrading from ancient versions of Roller.

9.1. Backup your old Roller

Before you get started with your upgrade, shutdown your existing Roller install and make a backup of your Roller data.

Backup your database to somewhere safe on your system or to a remote file-system. Here are a couple of examples: of how to do that on various databases:

- On MySQL you create a dump file

```
mysqldump -u scott -p rollerdb > /somewhere/safe/roller.dmp
```

- With PostgreSQL you can do the same thing

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

And backup any other data. Make a copy of your Roller data directory, i.e. the one with your Roller resources and search-index files. If you added or modified any files within your old Roller web application directory, then you'll want to backup that whole directory.

Migrating your old file uploads to the new Media Blogging system

If upgrading from Roller 4.0 to 5.1 (5.0 already has this configuration done), when you first start Roller 5.1 it will migrate your old file uploads to the new Media Blogging system. If this is to work properly you **MUST** ensure that the three properties below are set correctly before you start Roller 5.0/5.1 for the first time.

```
# The directory in which Roller 5.x will upload files

mediafiles.storage.dir=$\{user.home}/roller_data/mediafiles

# The directory in which Roller 4.0 uploaded files

uploads.dir=$\{user.home}/roller_data/uploads

# Set to true to enable migration

uploads.migrate.auto=true
```

The *mediafiles.storage.dir* property should be set to the location where you would like to store uploaded files. The *uploads.dir* property should be set to the location where you stored uploaded files in Roller 4.0.

9.2. Install and startup the new Roller

Follow the normal installation instructions for the new version of Roller, but…

- When creating your *roller-custom.properties*, copy of your old one. Carefully review each property and compare it to the property settings in the Roller property file described in Section 6.1.
- Don't create a new database for Roller. Instead point Roller to your existing Roller database. **This is completely safe because you created a backup of your database, right?**

When you deploy and startup, Roller will detect that your database needs to be upgraded and it will offer to run each of the migrations scripts necessary to upgrade you from your old version to the new version of Roller.

NOTE: You can run the database scripts manually too, see Section 8.2.

NOTE: On Tomcat, before startup you should delete the contents of the Tomcat work directory (located under the webapps folder.)