WebAnno Administrator Guide

The WebAnno Team

Version 2.3.1

Table of Contents

System Requirements

Table 1. Requirements for users

Browser	Chrome or Safari

Table 2. Requirements to run the standalone version

Java Runtime Environment	version 7 or higher

Table 3. Requirements run a WebAnno server

Java Runtime Environment	version 7 or higher
Apache Tomcat	version 6 or higher
MySQL Server	version 5 or higher

Installation

Prerequisites

- This guide assumes Debian 6.0.0 (Squeeze). It may also work on Ubuntu with some modifications, but we do not test this. Instructions for other Linux distributions likely deviate significantly.
- It is further assumed that the user "www-data" already exists on the system and that it shall be used to run WebAnno. Finally, it is expected that you have set up a MySQL database that WebAnno can use.
- All commands assume that you are logged in as the **root** user.

Prepare database

- Install MySQL

```
$ sudo apt-get install mysql-server
```

- login to MySQL

```
$ mysql -u root -p
```

- Create databases

```
mysql> CREATE DATABASE webanno DEFAULT CHARACTER SET utf8 COLLATE utf8_bin ;
```

- create a database user called `webanno` with the password `t0t4llYSecreT`which is later used by the application to access the database (instructions for `settings.properties`file below).

```
mysql> CREATE USER 'webanno'@'localhost' IDENTIFIED BY 't0t4llYSecreT';
mysql> GRANT ALL PRIVILEGES ON webanno.* TO 'webanno'@'localhost';
```

Dedicated Tomcat instance

- Install Tomcat.

```
$ aptitude install tomcat6
```

- Install package to install user-instances of Tomcat.

```
$ aptitude install tomcat6-user
```

- Create new instance

```
$ cd /opt
$ tomcat6-instance-create -p 18080 -c 18005 webanno
$ chown -R www-data /opt/webanno
```

Now we have the /opt/webanno/ apache tomcat installation. You will put the **webanno.war** file in the /opt/webanno/webapps/ folder. If you get the war file in different name, such as Webanno-1.1-beta-10.war, **PLEASE RE-NAME IT TO** *webanno.war* * Configure the startup script. Edit `/etc/init.d/webanno` and add the following contents:

```
#!/bin/sh
export JAVA_OPTS="-Djava.awt.headless=true -Xmx750m -XX:+UseConcMarkSweepGC
-Dwebanno.home=/srv/webanno"

case "$1" in
start)
su -c "sh /opt/webanno/bin/startup.sh" www-data
;;

stop)
su -c "sh /opt/webanno/bin/shutdown.sh" www-data
;;

restart)
su -c "sh /opt/webanno/bin/shutdown.sh" www-data
su -c "sh /opt/webanno/bin/startup.sh" www-data
;;
esac

exit 0
```

- Make the script executable and register it to run during system start:

```
$ chmod +x /etc/init.d/webanno
$ update-rc.d webanno defaults
```

Deploy WAR file

- Place the WebAnno WAR into the Tomcat `webapps` folder:

```
$ cp webanno.war /opt/webanno/webapps/webanno.war
```

- Create WebAnno home folder. This is the directory wher webanno settings files and projects (Serialized Cases, Source documents...) are stored

```
$ mkdir /srv/webanno
```

- **Optional** If you want to test WebAnno with some sample data, skip the *Users and permissions* section and follow the instruction at [Sampledata], then come back here.
- Edit `/srv/webanno/settings.properties` to define the database connection as well as internal backup properties and enable/disable crowd sourcing component:

```
database.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
database.driver=com.mysql.jdbc.Driver
database.url=jdbc:mysql://localhost:3306/webanno
database.username=webanno
database.password=t0t4llYSecreT

# 60 * 60 * 24 * 30 = 30 days
backup.keep.time=1000000

# 60 * 5 = 5 minutes
backup.interval=1000

backup.keep.number=10
```

- Fix permissions in WebAnno home folder

```
$ chown -R www-data /srv/webanno
```

- Start WebAnno

```
$ service webanno start
```

- Open it with your browser at `http://localhost:18080/webanno`. If you chose to run WebAnno behind the Apache web-server use `http://localhost/webanno`. The first time, it will create a username

admin with password admin. login with this username and proceed.

You can start with the SampleProjects to explore some of the functionalities.

Running behind Apache HTTPD

These are **optional** instructions if you want to run WebAnno behind an Apache web-server instead of accessing it directly. This assumes that you already have the following packages installed:

- Apache Web Server
- mod_proxy
- mod_proxy_ajp
- Edit `/opt/webanno/conf/server.xml`
- Enable AJP Connector on localhost (comment in, add address, and change port)

```
<Connector port="18009" protocol="AJP/1.3" redirectPort="8443" address="127.0.0.1"/>
```

- Disable HTTP Connector (just comment it out)

```
<!--Connector port="8080" protocol="HTTP/1.1".
           connectionTimeout="20000".
           URIEncoding="UTF-8"
           redirectPort="8443" /-->
```

- Edit `/etc/apache2/conf.d/webanno.local`

```
ProxyPreserveHost On

<Proxy ajp://localhost/webanno >
  Order Deny,Allow
  Deny from none
  Allow from all
</Proxy>

<Location /webanno >
  ProxyPass ajp://localhost:18009/webanno timeout=1200
  ProxyPassReverse http://localhost/webanno
</Location>
```

- Restart Apache web server

```
$ sudo /etc/init.d/apache2 restart
```

Database

WebAnno uses an SQL database to store project and user data.

We test MySQL using a MySQL server. WebAnno uses by default an embedded HSQLDB database. However, we recommend using the embedded database only for testing purposes. For production use, we recomment using a MySQL server. The reason for this is, that:

- we do more testing on the MySQL server and
- in the past, we had cases where we described in-place upgrade procedures that required performing SQL commands to change the data model as part of the upgrade. We promise to try avoiding this in the future. However, in case we offer advice on fixing anything directly in the database, this advice will refer to a MySQL database.

We try to keep the data model simple, so there should be no significant requirements to the database being used. Theoretically, it should be possible to use any JDBC-compatible database after adding a corresponding driver to the classpath and configuring WebAnno to use the driver in the `settings.properties` file.

Using HSQLDB in production

WebAnno displays a warning in the user interface when an embedded database is being used. In case that you really want to run WebAnno with an embedded database in production, you probably want to disable this warning. To do so, please add the following entry to the `settings.properties` file:

```
warnings.embeddedDatabase=false
```

Upgrade

Exporting/importing

- Log into WebAnno and export all the projects that you wish to migrate using the **Export** pane in the project settings
- Move your WebAnno home folder to a safe location so that WebAnno and create a new home folder in the old location
- Copy the **settings.properties** and **formats.properties** (if present) back from your moved folder
- Start the new WebAnno version to initialize the database
- Recreate the users
  - If you are using MySQL
    - create a new database for the new WebAnno version and update the **settings.properties** accordingly
    - use mysqldump to dump the tables `users` and `authorities` from the old database and load it back into the new database
  - If you are not using MySQL, you have to recreate the users manually
- When upgrading to WebAnno 2.x from a pre 2.x version, remove the **format.properties** file from the WebAnno home folder
- Restart WebAnno and import the previously exported projects

In-place update

This method should work when updating only a bugfix version, e.g. from 2.0.9 to 2.0.10. When performing a minor or major update, better use the exporting/importing method above.

- **Make a backup of your data before upgrading to a new version:**
  - Make a copy of your WebAnno home folder
  - If you are using MySQL, make a backup of your WebAnno database, e.g. using the mysqldump command.
- Standalone version
  - clean up the temporary installation before running the new version:
    - On OS X: `rm -R "$TMPDIR/winstoneEmbeddedWAR"`
    - On Linux: `rm -R /tmp/winstoneEmbeddedWAR`
    - On Windows: remove the `winstoneEmbeddedWAR` that should be somewhere under `C:\Users\<username>\AppData\Local\Temp`

start the new version

- WAR version
  - while Tomcat is running, delete the old WAR from your `webapps` folder
  - wait until Tomcat has automatically deleted the WebAnno folder
  - stop Tomcat
  - place the new WAR file into your `webapps` folder
  - start Tomcat

System Properties

Setting	Description	Default	Example
webanno.home	WebAnno home folder	~/.webanno	/srv/webanno
javamelody.disabled	Disable JavaMelody	true	false

Settings

Setting	Description	Default	Example
auth.mode	Authentication mode	database	preauth
auth.preauth.header.principal	Principal header	remote_user	*some other header*
auth.preauth.newuser.roles	Default roles for new users (comma separated)	*<none>*	ROLE_PROJECT_CREATOR
auth.user.*<username>*.roles	Extra roles for user (comma separated)	*<none>*	ROLE_ADMIN
database.dialect	Database dialect	org.hibernate.dialect.HSQLDialect	org.hibernate.dialect.MySQL5InnoDBDialect
database.driver	Database driver	org.hsqldb.jdbc.JDBCDriver	com.mysql.jdbc.Driver
database.url	JDBC connection string	*location in WebAnno home*	jdbc:mysql://localhost:3306/weblab
database.username	Database username	sa	user
database.password	Database password	sa	pass
database.initial-pool-size	Initial database connection pool size	4	
database.min-pool-size	Minimum database connection pool size	4	
database.max-pool-size	Maximum database connection pool size	10	
backup.interval	Time between backups (seconds)	0	300 *(60 * 5 = 5 minutes)*
backup.keep.number	Maximum number of backups to keep	0	5
backup.keep.time	Maximum age of backups to keep (seconds)	0	2592000 *(60 * 60 * 24 * 30 = 30 days)*
crowdsource.enabled	Enable crowdsourcing	0	1
style.logo	Logo image displayed in the upper-right corner	*unset*	*path to an image file*

Setting	Description	Default	Example
warnings.embeddedDatabase	Warn about using an embedded database	true	false
warnings.unsupportedBrowser	Warn about unsupported browser	true	false

Internal backups

WebAnno stores its annotations internally in files. Whenever a user performs an action on a document, the file is updated. It is possible to configure WebAnno to keep internal backups of these files, e.g. to safeguard against crashes or bugs.

The internal backups are controlled through three properties:

Setting	Description	Default
backup.interval	Time between backups (seconds)	`0` *(disabled)*
backup.keep.number	Maximum number of backups to keep	`0` *(unlimited)*
backup.keep.time	Maximum age of backups to keep (seconds)	`0` *(unlimited)*

By default, backups are disabled (**backup.interval** is set to `0`). Changing this properties to any positive number enables internal backups. The interval controls the minimum time between changes to a document that needs to have elapsed in order for a new backup to be created.

When backups are enabled, either or both of the properties **backup.keep.number** and **backup.keep.time** should be changed as well, because their default values will cause the backups to be stored indefinitely and they will eventually fill up the disk.

The properties **backup.keep.number** and **backup.keep.time** control how long backups are keep and the maximal number of backups to keep. These settings are effective simultaneously.

Example: Make backups every 5 minutes and keep 10 backups irrespective of age

```
backup.interval    = 300
backup.keep.number = 10
backup.keep.time   = 0
```

*Example: Make backups every 5 minutes and all not older than 7 days (60 * 60 * 24 * 7 seconds)*

```
backup.interval    = 300
backup.keep.number = 0
backup.keep.time   = 604800
```

Example: Make backups every 5 minutes and keep at most 10 backups that are not older than 7 days

```
backup.interval    = 300
backup.keep.number = 10
backup.keep.time   = 604800
```

External pre-authentication

WebAnno can be used in conjunction with header-based external per-authentication. In this mode, WebAnno looks for a special HTTP header (by default `remote_user`) and if that header exists, it is taken for granted that this user has been authenticated. WebAnno will check its internal database if a user by the given name exists, otherwise it will create the user.

Pre-authentication can be enabled by setting the property `auth.mode` to `preauth`. When enabling pre-authentication mode, the default roles for new users can be controlled using the `auth.preauth.newuser.roles` property. The `ROLE_USER` is always added, even if not specified explicitly. Adding also the role `ROLE_PROEJCT_CREATOR` allows all auto-created users also to create their own projects.

Since the default administrator user is not created in pre-authentication, it is useful to also declare at least one user as an administrator. This is done through the property `auth.user.<username>.roles` where `<username>` must be replaced with the name of the user. The example below shows how the user **Franz** is given administrator permissions.

Example: Authenticate using the `remote_user` header, new users can create projects, user **Franz** *is always admin.*

```
auth.mode                      = preauth
auth.preauth.header.principal = remote_user
auth.preauth.newuser.roles    = ROLE_PROJECT_CREATOR
auth.user.Franz.roles         = ROLE_ADMIN
```

The roles specified through `auth.preauth.newuser.roles` are saved in the database when a user logs in for the first time and can be changed after creation through the user interface.

The roles added through `auth.user.<username>.roles` properties are **not** saved in the database and **cannot** be edited through the user interface.