# WebAnno Administrator Guide

The WebAnno Team

Version 3.6.2

# Table of Contents

This guide covers handling WebAnno from an administrator’s perspective.

# Installation

You can run WebAnno on any major platform supporting Java, i.e. Linux, macOS or Windows. However, we do not provide explicit for setting up a production-ready instance of each of these platforms.

This guide assumes Debian 9.1 (Stretch). It may also work on Ubuntu with some modifications, but we do not test this. Instructions for other Linux distributions and other platforms (i.e. macOS and Windows) 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 the application.

All commands assume that you are logged in as the **root** user.

If you cannot log in as root but have to use `sudo` to become root, then the recommended way to do that is using the command `sudo su -`.

# System Requirements

*Table 1. Requirements for users*

| | |
|---|---|
| Browser | Chrome or Safari |

*Table 2. Requirements to run the standalone version*

| | |
|---|---|
| Java Runtime Environment | version 8 or higher |

*Table 3. Requirements run a WebAnno server*

| | |
|---|---|
| Java Runtime Environment | version 8 or higher |
| Apache Tomcat | version 8.5 or higher (Servlet API 3.1.0) |
| MySQL Server | version 5 or higher |

# Install Java

You can install an Oracle Java 8 JDK using the following commands.

```
$ apt-get update
$ apt-get install dirmngr
$ echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee
/etc/apt/sources.list.d/webupd8team-java.list
$ echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a
/etc/apt/sources.list.d/webupd8team-java.list
$ apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
$ apt-get update
$ apt-get install oracle-java8-installer
$ apt-get install oracle-java8-set-default
```

# Application home folder

The WebAnno home folder is the place where WebAnno's configuration file `settings.properties` resides and where WebAnno stores its data. Mind that if you are using a MySQL database server (recommended), then WebAnno also stores some data in the MySQL database. This is important when you plan to perform a backup, as both the home folder and the database content need to be included in the backup.

Now, let's go through the steps of setting up a home folder for WebAnno and creating a configuration file instructing WebAnno to access the previously prepared MySQL database.

- Create WebAnno home folder. This is the directory where WebAnno settings files and projects (documents, annotations, etc.) are stored

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

- Edit `/srv/webanno/settings.properties` to define the database connection as well as internal backup properties:

```
database.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
database.driver=com.mysql.jdbc.Driver
database.url=jdbc:mysql://localhost:3306/webanno?useSSL=false&serverTimezone=UTC
database.username=webanno
database.password=t0t4llYSecreT

# 60 * 60 * 24 * 30 = 30 days
backup.keep.time=2592000

# 60 * 5 = 5 minutes
backup.interval=300

backup.keep.number=10
```

- Fix permissions in WebAnno home folder

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

# Database

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

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

- some users have reported that HSQLDB databases may become corrupt when the computer crashes (note that this could probably also happen with MySQL, but we did so far not have any reports about this);
- most WebAnno developers use MySQL when running WebAnno on their servers;
- 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.

# MySQL

For production use of WebAnno, it is highly recommended to use a MySQL database. In this section, we briefly describe how to install a MySQL server and how to prepare it for use with the application.

## Prepare database

- Install MySQL

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

- make sure your MySQL server is configured for UTF-8. Check the following line is present in `/etc/mysql/mariadb.conf.d/50-server.cnf` (this is specific to Debian 9; on other systems the relevant file may be `/etc/mysql/my.cnf`):

```
character-set-server = utf8
collation-server     = utf8_bin
```

- also ensure the default settings for client connections to are UTF-8 in `/etc/mysql/mariadb.conf.d/50-server.cnf` (again Debian 9; likely in `/etc/mysql/my.cnf` on other systems)

```
default-character-set = utf8
```

- login to MySQL

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

- create a database

```
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';
mysql> FLUSH PRIVILEGES;
```

For production use, make sure you choose a different, secret, and secure password.

## Configuration options

This section explains some settings that can be added to the `database.url` in the `settings.properties` file when using MySQL. Settings are separated from the host name and database name with a `?` character and multiple settings are separated using the `&` character, e.g.:

```
database.url=jdbc:mysql://localhost:3306/webanno?useSSL=false&serverTimezone=UTC
```

To suppress the warning about non-SSL database connections with recent MySQL databases, append the following setting to the `database.url`:

```
useSSL=false
```

Recent MySQL drivers may refuse to work unless a database server timezone has been specified. The easiest way to do this is to add the following setting to the `database.url`:

```
serverTimezone=UTC
```

If you plan to use UTF-8 encoding for project name and tagset/tag name, make sure either of the following settings for MySQL database

- in the `settings.properties` file, make sure that `database.url` includes

```
useUnicode=true&characterEncoding=UTF-8
```

- change the `my.conf` MySQL databse configuration file to include the following line

```
character-set-server = utf8
```

# HSQLDB (embedded)

WebAnno displays a warning in the user interface when an embedded database is being used. It is not recommended to used an embedded database for various reasons:

- HSQLDB databases are known to run a risk of becoming corrupt in case of power failures which may render the application inaccessible and your data difficult to recover.
- In very rare cases it may be necessary to fix the database content which is more inconvenient for embedded databases.

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
```

# Running via embedded Tomcat (JAR)

The WebAnno standalone JAR with an embedded Tomcat server and can be easily set up as a UNIX service. This is the recommended way of running WebAnno on a server.

## Installing as a service

To set it up as a service, you can do the following steps. For the following example, I assume that you install WebAnno in `/srv/webanno`:

- Copy the standalone JAR file `webanno-standalone-3.6.2.jar` to `/srv/webanno/webanno.jar`. Note the change of the filename to `webanno.jar`.
- Create the file `/srv/webanno/webanno.conf` with the following content

  ```
  JAVA_OPTS="-Djava.awt.headless=true -Dwebanno.home=/srv/webanno"
  ```

- In the previous step, you have already created the `/srv/webanno/settings.properties` file. You **may optionally** configure the Tomcat port using the following line

  ```
  server.port=18080
  ```

  If you need to do additional configurations of the embedded Tomcat, best refer to the documentation of Spring Boot itself.

- Make sure that the file `/srv/webanno/webanno.conf` is owned by the root user. If this is not the case, WebAnno will ignore it and any settings made there will not have any effect. If you start WebAnno and intead of using the MySQL database, it is using an embedded database, then you should double-check that `/srv/webanno/webanno.conf` is owned by the root user.

  ```
  $ chown root:root /srv/webanno/webanno.conf
  ```

- Change the owner/group of `/srv/webanno/webanno.jar` to `www-data`. When the service is started, it will run with the privileges of the user that owns the JAR file, i.e. in this case WebAnno will run as under the `www-data` user. **Do NOT run WebAnno as root.**

  ```
  $ chown www-data:www-data /srv/webanno/webanno.jar
  ```

- Make the JAR file executable:

  ```
  $ chmod +x /srv/webanno/webanno.jar
  ```

- Create a symlink from `/etc/init.d` to the `/srv/webanno/webanno.jar`:

```
$ ln -s /srv/webanno/webanno.jar /etc/init.d/webanno
```

- Enable the WebAnno service using

```
$ systemctl enable webanno
```

- Start WebAnno using

```
$ service webanno start
```

- Check the log output

```
$ cat /var/log/webanno.log
```

- Stop WebAnno using

```
$ service webanno stop
```

# Running the standalone behind 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
- Add the following lines to `/srv/webanno/settings.properties`:

```
tomcat.ajp.port=18009
server.contextPath=/webanno
server.use-forward-headers=true
```

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

```
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

```
$ service apache2 restart
```

# Running via separate Tomcat (WAR)

You can also deploy the WebAnno WAR archive into a separately installed Tomcat instance.

## Installing Tomcat

- Install package to install user-instances of Tomcat.

```
$ apt-get install tomcat8-user authbind
```

- Create new instance

```
$ cd /opt
$ tomcat8-instance-create -p 18080 -c 18005 webanno
$ chown -R www-data /opt/webanno
```

If WebAnno is the only application you install on your server, then you can also have WebAnno running on port 80 or port 443. In that case, substitute all instances of port `18080` in these guidelines with the respective port. Mind that running via SSL on port 443 requires additional steps that we have not yet documented. Ports lower than 1024 are privileged and the WebAnno init script will automatically use a tool called **authbind** to allow WebAnno to operate on these ports as the unprivileged www-data user.

- Configure the startup script. Edit `/etc/init.d/webanno` and add the following contents or just download the file from here and place it in `/etc/init.d`.

```
#!/bin/sh

# Licensed under the Apache License, Version 2.0:
http://www.apache.org/licenses/LICENSE-2.0

# kFreeBSD do not accept scripts as interpreters, using #!/bin/sh and sourcing.
if [ true != "$INIT_D_SCRIPT_SOURCED" ] ; then
    set "$0" "$@"; INIT_D_SCRIPT_SOURCED=true . /lib/init/init-d-script
fi
### BEGIN INIT INFO
# Provides:          webanno
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: WebAnno init script
# Description:       This file should be placed in /etc/init.d. It
#                    allows starting/stopping WebAnno using the
```

```
#                    "service" command and ensures that WebAnno starts
#                    when the system is booted.
### END INIT INFO

# Author: Richard Eckart de Castilho

NAME="WebAnno"
DAEMON=none
WEBANNO_HOME="/srv/webanno"
WEBANNO_PORT="18080"
WEBANNO_USER="www-data"
CATALINA_BASE="/opt/webanno"
AUTHBIND=""
JAVA_OPTS="-Djava.awt.headless=true -Xmx750m -XX:+UseConcMarkSweepGC
-Dwebanno.home=$WEBANNO_HOME"

setup_authbind() {
  # log_action_msg "Setting up authbind configuration for $DESC on port
$WEBANNO_PORT"
  touch /etc/authbind/byport/$WEBANNO_PORT
  chmod 500 /etc/authbind/byport/$WEBANNO_PORT
  chown $WEBANNO_USER /etc/authbind/byport/$WEBANNO_PORT
  AUTHBIND="authbind --deep"
}

tomcat_pid() {
  echo `ps -fe | grep -- "-Dcatalina.base=$CATALINA_BASE" | grep -v grep | tr -s "
"|cut -d" " -f2`
}

do_start_cmd_override() {
  if [ $WEBANNO_PORT -lt 1024 ]
  then
    setup_authbind
  fi

  su - www-data -s "/bin/bash" -c "JAVA_OPTS=\"$JAVA_OPTS\" $AUTHBIND
$CATALINA_BASE/bin/startup.sh" > /dev/null 2>&1
}

do_stop_cmd_override() {
  su - www-data -s "/bin/bash" -c "$CATALINA_BASE/bin/shutdown.sh" > /dev/null 2>&1
}

do_status() {
  local pid
  pid=$(tomcat_pid)
  if [ -n "$pid" ]
  then
    log_action_msg "Status $DESC: running"
    return 0
```

```
  else
    log_action_msg "Status $DESC: stopped"
    return 1
  fi
}
```

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

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

If you deploy WebAnno on a Linux machine that is short on entropy, you can significantly decrease startup time by adding `-Djava.security.egd=file:/dev/urandom` to the `JAVA_OPTS` variable in the init script.

Now we have a dedicated Apache Tomcat instance for WebAnno installed at `/opt/webanno/` that automatically starts when the system boots and that can be managed through the usual `service` commands.

# Deploying the WAR file

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

```
$ cp webanno-webapp-3.6.2.war /opt/webanno/webapps/webanno.war
```

Mind that the copy command above renames the WAR file to `webanno.war`! This is important so that WebAnno is accessible at the URL noted later in the present guidelines.

- 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`. Log in with this username and proceed.

# Running the WAR 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` and 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.conf`

```
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

```
$ service apache2 restart
```

# Securing with SSL

This section assumes Debian 9.1 (Stretch) as the operating system using NGINX as a web server with WebAnno running through a separate Tomcat instance.

It further assumes that you want to use Let's Encrypt as a CA for obtaining valid SSL certificates.

- In addition, you will need a fully registered domain name. This tutorial uses `example.com`. Replace it accordingly.

We strongly encourage securing your production system with a firewall like UFW.

## Obtaining a Let's Encrypt certificate

The Certification Authority (CA) *Let's Encrypt* provides free TLS/SSL certificates. These certificates allow for secure HTTPS connections on web servers. *Let's Encrypt* provides the software Certbot which automates the obtaining process for NGINX.

- Enable the Stretch backports repo if needed
- Install Certbot preconfigured for NGINX

```
$ apt-get install python-certbot-nginx -t stretch-backports
```

- Obtain the certificates for your domain `example.com`

```
$ certbot --nginx certonly -d example.com
```

- You will be prompted to enter your e-mail address and asked to agree to the terms of service. Certificate renewal information will be sent to this e-mail. If the certification process is successful it will yield the information where your certificates can be found.

```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at
   /etc/letsencrypt/live/example.com/fullchain.pem. Your cert will
   expire on 2019-04-22. To obtain a new or tweaked version of this
   certificate in the future, simply run certbot again with the
   "certonly" option. To non-interactively renew *all* of your
   certificates, run "certbot renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Certificates issued by *Let's Encrypt* are valid for 90 days. You will receive an expiry notification to the e-mail address you provided during the certification process.

- Run Certbot with the command `renew` to renew all certificates that are due. You can also create a cron job for this purpose. The command for renewal is

```
$ certbot --nginx renew
```

- You can simulate the certificate renewal process with the command

```
$ certbot --nginx renew --dry-run
```

- The directory `/etc/letsencrypt/live/example.com/` now contains the necessary certificates to proceed

```
$ ls /etc/letsencrypt/live/example.com
Output:
cert.pem  chain.pem  fullchain.pem  privkey.pem
```

# Installing NGINX

This section assumes Debian 9.1 (Stretch) as the operating system using NGINX as a web server. It further assumes that you want to use Let's Encrypt as a CA for obtaining valid SSL certificates.

- You can install NGINX by typing

```
$ apt-get update
$ apt-get install nginx
```

- Verify the installation with

```
$ systemctl status nginx
Output:
● nginx.service - A high-performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-01-21 14:42:01 CET; 20h ago
     Docs: man:nginx(8)
  Process: 7947 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5
--pidfile /run/nginx.pid (code=exited, status=0/SUCCESS)
  Process: 7953 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
(code=exited, status=0/SUCCESS)
  Process: 7950 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on;
(code=exited, status=0/SUCCESS)
 Main PID: 7955 (nginx)
    Tasks: 9 (limit: 4915)
   CGroup: /system.slice/nginx.service
           ├─7955 nginx: master process /usr/sbin/nginx -g daemon on; master_process
on;
           ├─7956 nginx: worker process
```

- You can stop, start or restart NGINX with

```
$ systemctl stop nginx

$ systemctl start nginx

$ systemctl restart nginx
```

# Putting it all together

By now you should have

- WebAnno running on port 18080
- NGINX running with default configurations on port 80
- your issued SSL certificates

We will now configure NGINX to proxy pass all traffic received at `example.com/webanno` to our WebAnno instance.

Create a new virtual host for your domain. Inside of `/etc/nginx-sites-available/` create a new file for your domain (e.g. `example.com`). Paste the following contents:

```
# Server block for insecure http connections on port 80. Redirect to https on port 443
server {
        listen          80;
        listen          [::]:80;
        server_name     example.com;
        return          301 https://$server_name$request_uri;
}

# Server block for secure https connections
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name webanno.lingterm.net;

        ssl on;

        # Replace certificate paths
        ssl_certificate         /etc/letsencrypt/live/example.com/fullchain.pem;
        ssl_certificate_key     /etc/letsencrypt/live/example.com/privkey.pem;
        ssl_trusted_certificate /etc/letsencrypt/live/example.com/fullchain.pem;

        # Modern SSL Config from
        # https://mozilla.github.io/server-side-tls/ssl-config-generator/
        ssl_protocols TLSv1.2;
        ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-
```

```
SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-
SHA256:ECDHE-RSA-AES128-SHA256';
        ssl_prefer_server_ciphers on;
        ssl_session_timeout 1d;
        ssl_session_tickets off;
        add_header Strict-Transport-Security max-age=15768000;
        ssl_stapling on;
        ssl_stapling_verify on;

        ignore_invalid_headers off; #pass through headers from WebAnno which are
considered invalid by NGINX server.

        # Change body size if needed. This defines the maximum upload size for files.
        client_max_body_size    10M;

        # Uncommend this for a redirect from example.com to example.com/webanno
        #location / {
        #    return 301 https://$host/webanno;
        #}

        location ^~ /webanno/ {
            proxy_pass http://127.0.0.1:18080/webanno/;
            proxy_redirect default;
            proxy_http_version 1.1;

            proxy_set_header   Host             $host;
            proxy_set_header   X-Real-IP        $remote_addr;
            proxy_set_header   X-Forwarded-For  $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Proto $scheme;
            proxy_max_temp_file_size 0;

            proxy_connect_timeout      180;
            proxy_send_timeout         180;
            proxy_read_timeout         180;

            proxy_temp_file_write_size 64k;

            # Required for new HTTP-based CLI
            proxy_request_buffering off;
            proxy_buffering off; # Required for HTTP-based CLI to work over SSL
            proxy_set_header Connection ""; # Clear for keepalive
    }

    # Deny access to Apache .htaccess files. They have no special meaning for NGINX
and might leak sensitive information
    location ~ /\.ht {
            deny all;
    }
}
```

Create a symlink for the new configuration file to the folder for accessible websites:

```
$ ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/example.com
```

Test if the NGINX configuration file works without restarting (and possibly breaking) the webserver:

```
$ nginx -t
Output:
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

If the config works restart the webserver to enable the new site

```
$ service nginx restart
```

# Tell WebAnno that it is running behind a proxy

Edit the WebAnno `server.xml` file at `/opt/webanno/conf/server.xml` and insert the following `<Valve>` property into the `<Host>` block:

```
      <Host>
        ...
        <!-- Announce NGINX proxy header fields to WebAnno
        <Valve className="org.apache.catalina.valves.RemoteIpValve"
               internalProxies="127\.0\.[0-1]\.1"
               remoteIpHeader="x-forwarded-for"
               requestAttributesEnabled="true"
               protocolHeader="x-forwarded-proto"
               protocolHeaderHttpsValue="https"/>
        <!-- -->

        ...
      </Host>
    </Engine>
  </Service>
</Server>
```

Restart WebAnno

```
$ service webanno restart
```

WebAnno now knows how to interpret the proxy header fields from NGINX. With this step, everything is now set up to access WebAnno trough a secure https connection.

# Running via Docker

## Quick start

If you have Docker installed, you can run WebAnno using

```
docker run -it --name webanno -p8080:8080 webanno/webanno:3.6.2
```

The command download WebAnno from Dockerhub and starts it on port 8080. If this port is not available on your machine, you should provide another port to the `-p` parameter.

The logs will be printed to the console. To stop the container, press `CTRL-C`.

To run the WebAnno docker in the background use

```
docker run -d --name webanno -p8080:8080 webanno/webanno:3.6.2
```

Logs are accessible by typing

```
docker logs webanno
```

Use `docker run` only the first time that you run WebAnno. If you try it a second time, Docker will complain about the name `webanno` already being in use. If you follow Docker`s suggestion to delete the container, you will loose all your WebAnno data. Further below, we explain how you can store your data outside the container in a folder on your host.

When you want to run WebAnno again later, use the command

```
docker start -ai webanno
```

or for the background mode

```
docker start webanno
```

## Storing data on the host

If you follow the quick start instructions above, WebAnno will store all its data inside the docker container. This is normally not what you want because as soon as you delete the container, all data is gone. That means for example that you cannot easily upgrade to a new version of the WebAnno docker image when one is released.

To store your data on your host computer, first create a folder where you want to store your data. For example, if you are on Linux, you could create a folder `/srv/webanno`:

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

When you run WebAnno via Docker, you then mount this folder into the container:

```
docker run -it --name webanno -v /srv/webanno:/export -p8080:8080
webanno/webanno:3.6.2
```

## Settings file

The dockerized WebAnno expects the `settings.properties` file in the `/export` folder. Instead of injecting a custom `settings.properties` file into the container, it is strongly recommender to use the instructions above (Storing data on the host) to mount a folder from the host system to `/export` then to place the into the mounted folder `settings.properties`. Thus, if you follow the instructions above, the settings file would go to `/srv/webanno/settings.properties` on the host system.

## Connecting to a MySQL database

By default, WebAnno uses an embedded SQL database to store its metadata (not the texts and annotations, these are stored in files on disk). For production use, it is highly recommended to use a separate MySQL database instead of the embedded SQL database.

## Docker Compose

Using Docker Compose, you can manage multiple related containers. This section illustrates how to use Docker Compose to jointly set up a WebAnno container as well as a database container (i.e. this one).

The following Compose script sets these containers up.

The script contains example usernames and passwords used by the WebAnno container to connect to the database container, namely `DBUSER` and `DBPASSWORD`. It is highly recommended that you change these!

*Docker Compose script*

```
##
# docker-compose up [-d]
# docker-compose down
##
version: '2.1'

networks:
  webanno-net:
```

```
services:
  mysqlserver:
    image: "mysql:5"
    environment:
      - MYSQL_RANDOM_ROOT_PASSWORD=yes
      - MYSQL_DATABASE=webanno
      - MYSQL_USER=DBUSER
      - MYSQL_PORT=3306
      - MYSQL_PASSWORD=DBPASSWORD
    volumes:
      - ${WEBANNO_HOME}/mysql-data:/var/lib/mysql
    command: ["--character-set-server=utf8", "--collation-server=utf8_bin"]
    healthcheck:
      test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost", "-pDBPASSWORD", "-
uDBUSER"]
      interval: 20s
      timeout: 10s
      retries: 10
    networks:
      webanno-net:

  webserver:
    image: "webanno/webanno-snapshots:3.6.2"
    ports:
      - "${WEBANNO_PORT}:8080"
    environment:
      - WEBANNO_DB_DIALECT=org.hibernate.dialect.MySQL5InnoDBDialect
      - WEBANNO_DB_DRIVER=com.mysql.jdbc.Driver
      -
WEBANNO_DB_URL=jdbc:mysql://mysqlserver:3306/webanno?useSSL=false&useUnicode=true&char
acterEncoding=UTF-8
      - WEBANNO_DB_USERNAME=DBUSER
      - WEBANNO_DB_PASSWORD=DBPASSWORD
    volumes:
      - ${WEBANNO_HOME}/server-data:/export
    depends_on:
      mysqlserver:
        condition: service_healthy
    mem_limit: 1g
    memswap_limit: 1g
    restart: unless-stopped
    networks:
      webanno-net:
```

Place the script into any folder, change to that folder, and issue the commands

```
export WEBANNO_HOME=/srv/webanno
export WEBANNO_PORT=8080
docker-compose -p webanno up -d
```

This will start two docker containers: `webanno_mysqlserver_1`, and `webanno_webserver_1`. You can check the logs of each by running

```
docker logs webanno_mysqlserver_1
docker logs webanno_webserver_1
```

Two directories in your WebAnno home folder will be created: `mysql-data` and `webserver-data`. No data is stored in the containers themselves, you are safe to delete them with

```
docker-compose -p webanno down
```

You can also just stop or pause them, please see the docker-compose reference for details.

The settings within the docker-compose.yml file are just examples. Adjust the database URL, username, and password accordingly.

# Upgrading

In general, it is possible to perform an in-place upgrade of the application. However, before doing an upgrade, it is recommender to create a backup of the application and data to allow coming back to a working system if case of a problem during the upgrade. Mind that the upgrade is only completed once the new version has successfully started because during startup, the application may make changes to the database schema or to the data on disk.

# Backup your data

- 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.

# Upgrading with embedded Tomcat

- Stop the WebAnno service
- Replace the `webanno.jar` file with the new version
- Ensure that the file has the right owner/group (usually `www-data`)
- Start the WebAnno service again

# Upgrading with separate Tomcat

- 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

## Upgrading Tomcat 7 to Tomcat 8

If you have been using our installation instructions to install WebAnno on Linux, you are probably running an instance of Tomcat 6. WebAnno 3.3.0 is no longer compatible with Tomcat 6 and requires at least Tomcat 8.

To upgrade your existing instance, you can try the following procedure (adapt the procedure as necessary if you have deviated from our installation instructions):

- Stop the current WebAnno Tomcat 7 instance

```
$ service webanno stop
```

- Move your old Tomcat instance out of the way

```
$ mv /opt/webanno /opt/webanno-tomcat7
```

- Install `tomcat8-user` package (this will automatically uninstall Tomcat 7)

```
$ apt-get install tomcat8-user
```

- Create new instance

```
$ cd /opt
$ tomcat8-instance-create -p 18080 -c 18005 webanno
$ chown -R www-data /opt/webanno
```

- Copy the WAR file over to the new instance

```
$ mv /opt/webanno-tomcat7/webapps/webanno.war /opt/webanno/webapps/webanno.war
```

- Stop the new WebAnno Tomcat 8 instance

```
$ service webanno start
```

If you have made additional changes to the Tomcat 7 configuration files, e.g. changed `conf/server.xml`, please make sure to redo them in the new Tomcat 8 instance.

# Upgrading via export/import

This option can be used when performing an upgrade by exporting all data from once instance of the application into another instance which may potentially reside on a different machine. It is a very tedious approach because there is no option to bulk-export all projects.

- 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** 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

# Migration notes

## Version 3.2.x to 3.3.0

- When upgrading from 3.2.x or earlier to 3.3.0 or later, Automation projects break.

## Version 2.3.1 to 3.0.0

- The access permissions of administrators have changed. Administrators can no longer access annotation, curation, and monitoring pages for all projects. They can only access them if they are annotators, managers, or curators in the respective projects. However, they still have full access to the project settings of all projects and can simply give themselves the missing permissions. **After an upgrade to 3.0.0, all administrators who require project permissions on existing projects should assign these permissions to themselves. This also applies when importing old projects.** For new projects, the creator of the project always starts with annotator, curator, and manager permissions. If these permissions are not required by the project creator, they should be removed after project creation.

# Remote API

In order to programmatically manage annotation project, a REST-like remote API is offered. This API is disabled by default. In order to enable it, add the setting `remote-api.enabled=true` to the `settings.properties` file.

Once the remote API is enabled, it becomes possible to assign the role `ROLE_REMOTE` to a user. Create a new user, e.g. `remote-api` via the user management page and assign at least the roles `ROLE_USER` and `ROLE_REMOTE`. Most of the actions accessible through the remote API require administrator access, so adding the `ROLE_ADMIN` is usually necessary as well.

Once the remote API has been enabled, it offers a convenient and self-explanatory web-based user interface under `<APPLICATION_URL>/swagger-ui.html` which can be accessed by any user with the role `ROLE_REMOTE`. Here, you can browse the different operations, their parameters, and even try them out directly via a web browser. The actual AERO remote API uses `<APPLICATION_URL/api/aero/v1` as the base URL for its operations.

The API follows the Annotation Editor Remote Operations (AERO) protocol.

*Table 4. Remote API settings*

| Setting | Description | Default | Example |
|---|---|---|---|
| remote-api.enabled | Enable remote API | false | true |

# Webhooks

Webhooks allow WebAnno to notify external services about certain events. For example, an external service can be triggered when an annotator marks a document as finished or when all documents in a project have been completely curated.

Webhooks are declared in the `settings.properties` file. For every webhook, it is necessary to specify an URL (`url`) and a set of topics (`topics`) about with the remote service listening at the given URL is notified. If the remote service is accessible via https and the certificate is not known to the JVM running WebAnno, the certificate verification can be disabled (`verify-certificates`).

The following topics are supported:

- `DOCUMENT_STATE` - events related to the change of a document state such as when any user starts annotating or curating the document.
- `ANNOTATION_STATE` - events related to the change of an annotation state such as when a user starts or completes the annotation of a document.
- `PROJECT_STATE` - events related to the change of an entire project such as when all documents have been curated.

*Example webhook configuration*

```
webhooks.globalHooks[0].url=http://localhost:3333/
webhooks.globalHooks[0].topics[0]=DOCUMENT_STATE
webhooks.globalHooks[0].topics[1]=ANNOTATION_STATE
webhooks.globalHooks[0].topics[2]=PROJECT_STATE
webhooks.globalHooks[0].verify-certificates=false
```

# Settings

Application settings are managed via a file called `settings.properties` which must reside in the application home folder. The file is optional. If it does not exist, default values are assumed.

# General Settings

*Table 5. General settings*

| Setting | Description | Default | Example |
|---|---|---|---|
| warnings.unsupportedBrowser | Warn about unsupported browser | true | false |
| debug.showExceptionPage | Show a page with a stack trace instead of an "Internal error" page. Do not use in production! | false | true |
| login.message | Custom message to appear on the login page, such as project web-site, annotation guideline link, ... The message can be an HTML content. | *unset* | `<span style="color:red; font-size: 200%;">Use are your own risk.</span>` |
| user.profile.accessible | Whether regular users can access their own profile to change their password and other profile information. This setting has no effect when running in pre-authentication mode. | false | true |
| userSelection.hideUsers | Whether the list of users show in the users tab of the project settings is restricted. If this setting is enable, the full name of a user has to be entered into the input field before the user can be added. If this setting is disabled, it is possible to see all enabled users and to add any of them to the project. | false | true |

# Database connection

*Table 6. Database settings in the `settings.properties` file*

| Setting | Description | Default | Example |
|---|---|---|---|
| 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 application home* | jdbc:mysql://localhost:3306/weblab?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC |
| database.username | Database username | sa | user |
| database.password | Database password | *unset* | 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 | |
| warnings.embeddedDatabase | Warn about using an embedded database | true | false |

The basic database connection details can also be configured via environment variables. When these environment variables are present, they are preferred over the `settings.properties` file. The following environment variables can be used:

*Table 7. Database configuration via environment variables*

| Setting | Description | Default | Example |
|---|---|---|---|
| `WEBANNO_DB_DIALECT` | Database dialect | org.hibernate.dialect.HSQLDialect | org.hibernate.dialect.MySQL5InnoDBDialect |
| `WEBANNO_DB_DRIVER` | Database driver | org.hsqldb.jdbc.JDBCDriver | com.mysql.jdbc.Driver |
| `WEBANNO_DB_URL` | JDBC connection string | *location in application home* | jdbc:mysql://localhost:3306/webanno?useUnicode=true&characterEncoding=UTF-8 |
| `WEBANNO_DB_USERNAME` | Database username | sa | user |
| `WEBANNO_DB_PASSWORD` | Database password | *unset* | pass |

# Internal backup

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:

*Table 8. Database settings in the `settings.properties` file*

| Setting | Description | Default | Example |
|---|---|---|---|
| backup.interval | Time between backups (seconds) | 0 *(disabled)* | 300 *(60 * 5 = 5 minutes)* |
| backup.keep.number | Maximum number of backups to keep | 0 *(unlimited)* | 5 |
| backup.keep.time | Maximum age of backups to keep (seconds) | 0 *(unlimited)* | 2592000 *(60 * 60 * 24 * 30 = 30 days)* |

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
```

# Custom header icons

WebAnno allows adding custom icons to the page header. You can declare such custom icons in the `settings.properties` file as shown in the example below. Each declaration begins with the prefix `style.header.icon.` followed by an identifier (here `myOrganization` and `mySupport`). The suffixes `.linkUrl` and `.imageUrl` indicate the URL of the target page and of the icon image respectively. Images are automatically resized via CSS. However, to keep loading times low, you should point to a reasonably small image.

The order of the icons is controlled by the ID, not by the order in the configuration file!

*Example: Custom header icon*

```
style.header.icon.myOrganization.linkUrl=http://my.org
style.header.icon.myOrganization.imageUrl=http://my.org/logo.png
style.header.icon.mySupport.linkUrl=http://my.org/support
style.header.icon.mySupport.imageUrl=http://my.org/help.png
```

| Setting | Description | Default | Example |
|---|---|---|---|
| style.logo | Logo image displayed in the upper-right corner | *unset* | *path to an image file* |
| style.header.icon… | Icons/links to display in the page header. For details, see below. | *unset* | |

# Annotation editor

| Setting | Description | Default | Example |
|---|---|---|---|
| ui.brat.autoScroll | Whether to scroll the annotation being edited into the center of the page | true | |
| ui.brat.pageSize | The number of sentences to display per page | 5 | |
| ui.brat.singleClickSelection | Whether to select annotations with a single click | false | |
| ui.brat.rememberLayer | Whether "remember layer" is activated by default | false | |
| annotation.feature-support.string.autoCompleteThreshold | If the tagset is larger than the threshold, an auto-complete field is used instead of a standard combobox. | 75 | 100 |
| annotation.feature-support.string.autoCompleteMaxResults | When an auto-complete field is used, this determines the maximum number of items shown in the dropdown menu. | 100 | 1000 |

# External pre-authentication

WebAnno can be used in conjunction with header-based external per-authentication. In this mode, the application 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. The application 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.

| 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 |