

Administration

- Bookstack Dokumentation
- Backup
- Suchmaschinenoptimierung
- Optimierung von Bilddateien
- überflüssige Bilder löschen

Bookstack Dokumentation

<https://www.bookstackapp.com/docs/admin/>

Fehlersuche im System:

<https://www.bookstackapp.com/docs/admin/debugging/>

Backup

Dieses Skript sichert die Datenbank und alle Benutzerdateien. Siehe auch

<https://www.bookstackapp.com/docs/admin/backup-restore>

Skript

```
#!/bin/bash
#set -x

BACKUPDIR=/backup/bookstack
MYSQL_DB=bookstack
WWW_DIR=/var/www/bookstack
ITEMS=".env public/uploads/ storage/uploads/ themes/"

MINFILESIZE=200 # in kb
KEEP=28 # in days
EMAIL="[Absender]"

DATE=$(date "+%Y-%m-%dT%H%M%S%z")
FILENAME=${BACKUPDIR}/${MYSQL_DB}-${DATE}.sql

mysqldump --triggers --routines $MYSQL_DB -r $FILENAME
DB_DUMPSIZE=$(du -k $FILENAME|cut -f1)
if [[ -r $FILENAME && $DB_DUMPSIZE -ge $MINFILESIZE ]]
then
    xz -9e --lzma2=dict=1024MiB,nice=273 $FILENAME
else
    echo -e "Host:\t\t$(hostname -f)\nDB:\t\t${MYSQL_DB}\nDatei:\t\t${FILENAME}\nGröße (min):\t\t${MINFILESIZE} kb\nGröße (tatsächlich):\t\t${DB_DUMPSIZE} kb"|mailx -s "Backup gescheitert: $MYSQL_DB" $EMAIL
fi

FILENAME_ARCHIVE=${BACKUPDIR}/${(basename $WWW_DIR)-}${DATE}.tar.xz
tar --exclude=storage/framework -cf $FILENAME_ARCHIVE --use-compress-program='xz -9 --lzma2=dict=1024MiB,nice=273' -C /${WWW_DIR} $ITEMS
```

```
find $BACKUPDIR/* -type f -mtime +$KEEP -exec rm {} \; 2>&1
```

Cronjob

```
15 3 * * 2-6 /usr/local/bin/bs-backup-db.sh
```

Suchmaschinenoptimierung

Sitemap

Um Suchmaschinen die Indizierung Inhalte von Bookstack zu erleichtern, wird mittels eines Skriptes eine Sitemap generiert.

Installation

- Skript herunterladen

```
curl https://raw.githubusercontent.com/BookStackApp/api-scripts/main/php-generate-sitemap/generate-sitemap.php > /user/local/bin/bs-gen-sitemap.php
```

- Benutzer *Sitemap* anlegen
- Rolle für öffentliche Sichtbarkeit zuweisen (Guest), Rolle für API zuweisen; Durch die Rolle *API* erhält der Nutzer Zugriff auf die API. Durch die Rolle *Guest* werden nur öffentliche Seiten in die Sitemap aufgenommen.
- API-Token erzeugen und Variablen im PHP-Skript setzen

```
$baseUrl = 'https://kb.el.uni-leipzig.de';  
// sitemap  
$clientId = '[geheim]';  
$clientSecret = '[geheim]';
```

- Cronjob anlegen bzw. Sitemap manuell erstellen

```
15 19 * * 1-5 php /usr/local/bin/bs-gen-sitemap.php /var/www/bookstack/public/sitemap.xml
```

- URL-Liste für aus sitemap.xml erstellen

```
sed -n 's:.*<loc>\(.*\)</loc>.*:1:p' /var/www/bookstack/public/sitemap.xml
```

- veröffentlichte Seite zählen

```
sed -n 's:.*<loc>\(.*\)</loc>.*:1:p' /var/www/bookstack/public/sitemap.xml | wc -l
```

Cronjob

Anmeldung zur Indizierung

Die Website inkl. Sitemap bei Suchmaschinen anmelden

- <https://www.bing.com/webmasters/sitemaps?siteUrl=https://kb.el.uni-leipzig.de/>
- https://search.google.com/search-console?resource_id=https%3A%2F%2Fkb.el.uni-leipzig.de%2F
- <https://webmaster.yandex.com/>

Optimierung von Bilddateien

Im Wesentlichen werden zwei Bildformate verwendet, wobei jedes Format für bestimmte Inhalte optimiert ist:

- PNG: Geeignet für Screenshots, Grafiken und Motive mit wenigen, flächigen Farben
- JPEG: Geeignet für natürliche Inhalte wie Fotos und Bilder mit komplexen Strukturen, Farben und Farbverläufen

Meist werden diese Bilder nicht im für Webseiten optimalen Format gespeichert, d. h.

- in PNGs wird die komplette Farbpalette (RGB + Transparenz, mit 24 bzw. 32 bit) verwendet, obwohl für Darstellung deutlich weniger Farben ausreichen
- JPEGs werden qualitativ sehr hochwertig erstellt, erstellt Dateien sind recht groß
- JPEGs werden im Format *Baseline* erstellt, können also nicht progressiv geladen werden
- Bilder enthalten Metadaten

Hier einige Referenzen zur Optimierung von Bildern für die Darstellung auf Webseiten:

- Analyse der Bildqualität von JPEGs in Abhängigkeit von der Kompression
- Warum sollte man PNGs optimieren?
- Baseline oder Progressive?
- Performance mit progressiven JPEGs verbessern

Bilder in Bookstack automatisch optimieren

Um Nutzer nicht unnötig mit der Optimierung zu belasten, optimiert nachfolgendes Skript Bilddateien automatisch. Dabei kommen `pngquant` und `jpegoptim` zum Einsatz, welche für gängige Linux-Distributionen angeboten werden. Täglich 1x werden die Bilder aller Unterverzeichnisse (cover_book, cover_bookshelf, drawio, gallery, system, user) des aktuellen Monats im Image-Ordner bearbeitet.

Verzeichnis: `/var/www/bookstack/public/uploads/images/[asset]/[YYYY-MM]/`

Mit jedem Durchlauf werden Bilder weiter optimiert, wobei die Optimierung beim ersten Durchlauf am größten ist. `Pngquant` und `jpegoptim` optimieren die Bilder nur, wenn eine Mindestqualität nicht

unterschriften wird. Die mehrfache Anwendung des Skriptes verschlechtert die Bildqualität daher nicht. Nach 2 bis 3 Durchläufen ist das angestrebte Optimum erreicht. In Tests mit mehreren hundert Bildern wurden zwischen 66 und 80 % der Dateigröße ohne sichtbaren Qualitätsverlust eingespart. Bei Dateien die mit in Bookstack integriertem **draw.io** erstellt/bearbeitet wurden und welche im Ordner `drawio` liegen, werden die Metadaten beibehalten, da diese die Vektorgrafiken für spätere Bearbeitung enthalten. Würden sie entfernt, könnten sie nicht mehr mit draw.io bearbeitet werden.

Benötigte Komponenten installieren:

```
apt-get update
apt-get install pngquant
apt-get install jpegoptim
```

Skript installieren

`/usr/local/bin/bs-opt-img.sh`

```
# !/bin/bash
# set -x
#
# Recompresses uploaded images of the current month and strips meta tags.
# Images are only processed if there are potential savings. Script can be run
# multiple times without recompressing already compressed images - processed
# images are skipped. Needs pngquant and jpegoptim to be installed. Run script
# atleast once a week better daily.
# Logs at /var/log/[script-name].log
# Metadata of png files in drawio are not stripped because they are containing
# vector data which are necessary to edit these drawings
#
# Version 2023-10-25
# daniel.obst@uni-leipzig.de

FOLDER_BASE=/var/www/bookstack/public/uploads/images

LOG=/var/log/`echo $(basename "$0") | rev | cut -d. -f2- | rev`.log
#FOLDERS="cover_book cover_bookshelf drawio gallery system user"
echo "$(date '+%Y-%m-%d %H:%M:%S') === START ===" >> $LOG
FOLDERS=""`ls -q $FOLDER_BASE`
```



```

for FOLDER in $FOLDERS; do
    FOLDER_LONG=$FOLDER_BASE/$FOLDER/`date +"%Y-%m"`/
# FOLDER_LONG=$FOLDER_BASE/$i/2023-07/
    if [ -d "$FOLDER_LONG" ]; then
        DU_BEFORE=`du -s --block-size=1 $FOLDER_LONG | cut -f1`
        DU_BEFORE_K=$((DU_BEFORE/1024))
        FILES=`ls -lq $FOLDER_LONG | grep -iE "\.?(jpg|jpeg|png)$" | wc -l`
        if [ $FILES -gt 0 ]; then
            if [ "$FOLDER" = "drawio" ]; then
                find $FOLDER_LONG -name "*.png" -exec pngquant --force --skip-if-larger --speed=1 --ext .png --
quality=60-80 {} \;
                find $FOLDER_LONG -name "*.PNG" -exec pngquant --force --skip-if-larger --speed=1 --ext .PNG --
quality=60-80 {} \;
            else
                find $FOLDER_LONG -name "*.png" -exec pngquant --strip --force --skip-if-larger --speed=1 --ext .png --
quality=60-80 {} \;
                find $FOLDER_LONG -name "*.PNG" -exec pngquant --strip --force --skip-if-larger --speed=1 --ext .PNG --
quality=60-80 {} \;
            fi
            find $FOLDER_LONG -iname "*.png" -exec chown -f --reference=$FOLDER_LONG {} \;
            find $FOLDER_LONG -iregex "\.?(jpe?g)" -exec jpegoptim -q -p --all-progressive -m75 --strip-com --strip-exif --
strip-iptc --strip-icc --strip-xmp {} \;
            DU_AFTER=`du -s --block-size=1 $FOLDER_LONG | cut -f1`
            DU_AFTER_K=$((DU_AFTER/1024))
            if [ $DU_AFTER -lt $DU_BEFORE ]; then
                SAVED_K=$((DU_BEFORE_K-$DU_AFTER_K))
                echo "$(date '+%Y-%m-%d %H:%M:%S') $FOLDER_LONG $FILES images, folder shrunked from
${DU_BEFORE_K}k to ${DU_AFTER_K}k, ${SAVED_K}k saved" >> $LOG
            else
                echo "$(date '+%Y-%m-%d %H:%M:%S') $FOLDER_LONG nothing to do, $FILES images, ${DU_AFTER_K}k"
>> $LOG
            fi
        else
            echo "$(date '+%Y-%m-%d %H:%M:%S') $FOLDER_LONG no images found" >> $LOG
        fi
    else
        echo "$(date '+%Y-%m-%d %H:%M:%S') $FOLDER_LONG does not exists" >> $LOG
    fi
done
echo "$(date '+%Y-%m-%d %H:%M:%S') $FOLDER_BASE/ `find $FOLDER_BASE/. -type f | wc -l` files, `du -sh

```

```
$FOLDER_BASE | cut -f1`" >> $LOG  
echo "$(date '+%Y-%m-%d %H:%M:%S') === STOP ===" >> $LOG
```

Cronjob

mit `crontab -e`

```
0 23 * * * /usr/local/bin/bs-opt-img.sh
```

Logs

Das Skript schreibt Logs nach `/var/log/bs-opt-img.log`. Um zu verhindern, dass die Logs die Festplatte vollschreiben, sollten die Logs rotiert werden. Dazu eine Datei `/etc/logrotate.d/bs-opt-img` mit folgendem Inhalt anlegen:

```
/var/log/bs-opt-img.log  
{  
    rotate 4  
    weekly  
    missingok  
    notifempty  
    compress  
    delaycompress  
    sharedscripts  
}
```

Syslog Daemon neustarten:

```
systemctl restart rsyslog.service
```

überflüssige Bilder löschen

Nachfolgendes Skript löscht Bilder, die nicht mehr referenziert sind:

Skript

/usr/local/bin/bs-cleanup-img.sh:

```
#!/bin/bash
# set -x
#
# Cleanup images and drawings
# Also delete images that are only used in old revisions
# depending on amount of images cleanup will take several minutes
#
# Version 2024-02-29

APP_FOLDER=/var/www/bookstack
IMG_FOLDER=${APP_FOLDER}/public/uploads/images
LOG=/var/log/`echo $(basename "$0") | rev | cut -d. -f2 | rev`.log

echo "$(date '+%Y-%m-%d %H:%M:%S') === START ===" >> $LOG
echo "working on ${APP_FOLDER}" >> $LOG
DU_BEFORE=`du -s --block-size=1 $IMG_FOLDER | cut -f1`
DU_BEFORE_K=$((DU_BEFORE/1024))
FILES_BEFORE=`ls -lqR ${IMG_FOLDER} | grep -iE "\.(\.jpg|jpeg|png)$" | wc -l`

php ${APP_FOLDER}/artisan bookstack:cleanup-images -f -n -v >> $LOG

DU_AFTER=`du -s --block-size=1 $IMG_FOLDER | cut -f1`
DU_AFTER_K=$((DU_AFTER/1024))
FILES_AFTER=`ls -lqR ${IMG_FOLDER} | grep -iE "\.(\.jpg|jpeg|png)$" | wc -l`

if [ $DU_AFTER -lt $DU_BEFORE ]; then
    DELETED_K=$((DU_BEFORE_K-$DU_AFTER_K))
    FILES_DELETED=$((FILES_BEFORE-$FILES_AFTER))
```

```
echo "${FILES_BEFORE} - ${FILES_AFTER} = ${FILES_DELETED} deleted" >> $LOG
echo "${DU_BEFORE_K} - ${DU_AFTER_K} = ${DELETED_K}k deleted" >> $LOG
else
echo "nothing to do, ${FILES_AFTER} files, ${DU_AFTER_K}k" >> $LOG
fi
echo "$(date '+%Y-%m-%d %H:%M:%S') === STOP ===" >> $LOG
```

Auszug aus dem Log:

```
2024-02-29 16:31:19 === START ===
working on /var/www/bookstack
This operation is destructive and is not guaranteed to be fully accurate.
Ensure you have a backup of your images.

Image(s) to delete:
/uploads/images/gallery/2023-07/img-20220630-132119.jpg
/uploads/images/gallery/2023-08/PGBimage.png
[..]
/uploads/images/gallery/2024-01/vRZgrafik.png
/uploads/images/drawio/2024-02/drawing-7-1706817605.png
/uploads/images/drawio/2024-02/drawing-21-1708611127.png
325 image(s) deleted
8934 - 8000 = 934 files deleted
315784k - 256552k = 59232k deleted
2024-02-29 16:32:37 === STOP ===
2024-02-29 17:09:58 === START ===
working on /var/www/bookstack
This operation is destructive and is not guaranteed to be fully accurate.
Ensure you have a backup of your images.

0 image(s) deleted
nothing to do, 8000 files, 256552k
2024-02-29 17:24:18 === STOP ===
```

Cronjob

Skript sonntags 3:15 Uhr ausführen:

```
15 3 * * 7 /usr/local/bin/bs-cleanup-img.sh
```

Logrotate

/etc/logrotate.d/bs-cleanup-img

```
/var/log/bs-cleanup-img.log
{
    rotate 4
    weekly
    missingok
    notifempty
    compress
    delaycompress
    sharedscripts
}
```

Service neustarten, um Konfig zu aktivieren

```
systemctl restart rsyslog.service
```