

BookStack

das ist das, worauf dieses Wiki hier überhaupt läuft

- [Browsercache verhindert Editor nach Update auf v22.07](#)
- [BookStack-Autoupdate per Shellsript](#)
- [Editor Shortcuts](#)

Browsercache verhindert Editor nach Update auf v22.07

Kleine Notiz:

Nach Update auf 22.07 war der Editor grau, wurde einfach nicht geladen.

Lag nicht an den Browser Extensions, einfaches erzwungenes laden per Strg+F5 half. Damit wird der Browsercache umgangen und immer vom Server abgerufen.

Ja auch nach

```
php artisan cache:clear  
php artisan view:clear
```

Das tut ja nur was auf dem Server, nicht auf dem Client.

War bei vorherigen Updates eigentlich nie das Problem...

Nicht zu voreilig updaten, oftmals hauen die nach einem Update gleich zwei bis drei Minor Updates hinterher.

BookStack-Autoupdate per Shellscript

BookStack hat ein Problem. Es gibt kein Autoupdate. Autoupdate ist essenziell um einigermaßen Sicherheit gewährleisten zu können.

Also müssen wir das Autoupdate selbst machen.

Die Grundlage hierzu ist natürlich ein Userspace. Dort können wir kein sudo nutzen, also muss alles im Userspace laufen:

Script

Das Script geht davon aus dass die [Anleitung aus dem Lab](#) zur Installation verwendet wurde und dass eine globale \$USER Variable existiert. Also die Datenbank euren Username mit dem Präfix _bookstack trägt.

```
#!/bin/bash

# Path to the last known version file
VERSION_FILE="/home/$USER/bookstack-latest-version.txt"
LOG_FILE="/home/$USER/bookstack-update.log"

# Backup directory
BACKUP_DIR="/home/$USER/bookstack-backups"
# Create backup directory if it doesn't exist
mkdir -p "$BACKUP_DIR"

# using github API
LATEST_RELEASE=$(curl -s https://api.github.com/repos/BookStackApp/BookStack/releases/latest |
grep -oP '"tag_name": "\K[^\"]*')

echo "Debug: Latest release detected: $LATEST_RELEASE" >> "$LOG_FILE"

# Check if version file exists and has content
```

```

if [ ! -s "$VERSION_FILE" ]; then
    echo "Debug: Version file is empty or doesn't exist" >> "$LOG_FILE"
    echo "$LATEST_RELEASE" > "$VERSION_FILE"
    echo "Debug: Created version file with: $LATEST_RELEASE" >> "$LOG_FILE"
    exit 0
fi

# Compare current stored version with latest release
STORED_VERSION=$(cat "$VERSION_FILE")

echo "Debug: Stored version: $STORED_VERSION" >> "$LOG_FILE"
echo "Debug: Latest release: $LATEST_RELEASE" >> "$LOG_FILE"

if [ "$LATEST_RELEASE" != "$STORED_VERSION" ]; then
    echo "New version detected: $LATEST_RELEASE" >> "$LOG_FILE"

    # Database backup before update
    BACKUP_FILENAME="bookstack_backup_$(date +"%Y%m%d_%H%M%S").sql.gz"
    BACKUP_PATH="${BACKUP_DIR}/${BACKUP_FILENAME}"

    # Perform MySQL dump
    mysqldump --defaults-group-suffix=client -u "$USER" "$USER"_bookstack | gzip >
"$BACKUP_PATH"

    # Check backup success
    if [ $? -eq 0 ]; then
        echo "Database backup successful: $BACKUP_PATH" >> "$LOG_FILE"

        # Cleanup old backups (keep last 5)
        ls -t "$BACKUP_DIR"/bookstack_backup_*.sql.gz | tail -n +6 | xargs -d '\n' rm -f
    else
        echo "Database backup failed!" >> "$LOG_FILE"
        exit 1
    fi

    # Perform update steps
    cd /var/www/virtual/$USER/BookStack
    git fetch origin
    git pull origin release
    composer install --no-dev

```

```
php artisan migrate --force
php artisan cache:clear
php artisan view:clear

# Update the version file
echo "$LATEST_RELEASE" > "$VERSION_FILE"
echo "$(date): Updated to $LATEST_RELEASE" >> "$LOG_FILE"
else
    echo "$(date): No new version available" >> "$LOG_FILE"
fi
```

php artisan migrate --force ist hier der kritischste Punkt. Die Abfrage für eine Migration ist nicht ohne Grund manuell. Wenn etwas irgendwann Nacharbeit erfordert, dann das...

Wer gut aufgepasst hat: mysqldump benötigt hier einen write-permission user. Warum? Mysqldump muss die Tabelle für den Prozess des Backups vor Änderungen schützen und nutzt dafür LOCK TABLES. Das ist in MySQL als write klassifiziert.

chmod +x ~/bookstack-update.sh nicht vergessen

PHP

Manche Bookstack Updates setzen das Requirement für PHP höher. Läuft Bookstack nach Autoupdate auf einer kleineren PHP Version kann es sein dass Dinge wie der Login oder auch nur Teile davon (z.B. 2FA) nicht laufen bis das Requirement gegeben ist.

Dieses Script hier ist spezifisch für Uberspace, da deren Lösung für PHP Umstellungen im Userspace laufen muss.

```
#!/bin/bash

LOGFILE=~/.php-update.log
MAILTO="MAIL" # <== Bitte anpassen!
URL_TO_TEST="SITE"

NOW=$(date +"%Y-%m-%d %H:%M:%S")

# Alte Logeinträge (älter als 6 Monate) bereinigen
```

```

TMP_LOG=$(mktemp)
while IFS= read -r line; do
    ENTRY_DATE=$(echo "$line" | awk '{print $1}')
    if [[ "$ENTRY_DATE" =~ ^[0-9]{4}-[0-9]{2}-[0-9]{2}$ ]]; then
        if [[ $(date -d "$ENTRY_DATE +6 months" +%s) -ge $(date +%s) ]]; then
            echo "$line" >> "$TMP_LOG"
        fi
    fi
done < "$LOGFILE"
mv "$TMP_LOG" "$LOGFILE"

# PHP-Versionen ermitteln
CURRENT_VERSION=$(uberspace tools version show php | awk -F'"' '/Using/{print $4}')
LATEST_VERSION=$(uberspace tools version list php | sed 's/^\[- ]*//' | sort -V | tail -n 1)

# Funktion zum HTTP-Test
function check_site() {
    curl -s -o /dev/null -w "%{http_code}" "$1"
}

# Funktion zur Mailbenachrichtigung
function send_error_mail() {
    echo -e "$2" | mail -s "$1" "$MAILTO"
}

# Wenn aktuelle Version bereits die neueste ist, abbrechen
if [ "$CURRENT_VERSION" == "$LATEST_VERSION" ]; then
    echo "[$NOW] PHP ist bereits aktuell: $CURRENT_VERSION"
    exit 0
fi

# Vor dem Update testen
STATUS_BEFORE=$(check_site "$URL_TO_TEST")
if [ "$STATUS_BEFORE" != "200" ]; then
    send_error_mail "[PHP-Update] Wiki nicht erreichbar vor Update" \
        "Das Wiki unter $URL_TO_TEST ist vor dem PHP-Update nicht mit Status 200 erreichbar
(Status: $STATUS_BEFORE). Kein Update durchgeführt."
    echo "[$NOW] Fehler: Seite vor Update nicht erreichbar ($STATUS_BEFORE)" >> "$LOGFILE"
    exit 1
fi

```

```

# Update durchführen
echo "[$NOW] Aktualisiere PHP von $CURRENT_VERSION auf $LATEST_VERSION"
if uberspace tools version use php "$LATEST_VERSION"; then
    sleep 5 # Kurze Pause nach Umstellung
    STATUS_AFTER=$(check_site "$URL_TO_TEST")
    if [ "$STATUS_AFTER" == "200" ]; then
        echo "$(date +"%Y-%m-%d") PHP-Version auf $LATEST_VERSION gesetzt" >> "$LOGFILE"
        echo "[$NOW] Update erfolgreich und Seite weiterhin erreichbar."
        exit 0
    else
        # Rückgängig machen
        uberspace tools version use php "$CURRENT_VERSION"
        send_error_mail "[PHP-Update] Wiki nicht erreichbar nach Update" \
            "Das Wiki unter $URL_TO_TEST war nach dem Update auf PHP $LATEST_VERSION nicht mit
Status 200 erreichbar (Status: $STATUS_AFTER).\n\nDie Version wurde$
        echo "[$NOW] Fehler: Seite nach Update nicht erreichbar ($STATUS_AFTER). Rollback
durchgeführt." >> "$LOGFILE"
        exit 1
    fi
else
    send_error_mail "[PHP-Update] Fehler beim Wechsel" \
        "Beim Versuch, die PHP-Version von $CURRENT_VERSION auf $LATEST_VERSION umzustellen, ist
ein Fehler aufgetreten. Kein Wechsel durchgeführt."
    echo "[$NOW] Fehler beim Wechsel auf $LATEST_VERSION" >> "$LOGFILE"
    exit 1
fi

```

Scriptanalyse

So, jetzt müssen wir das Ganze verstehen können, immerhin ist's durch eine LLM-KI deutlich schneller entstanden als ohne diese.

Was nutzen wir dafür? Klar, auch LLM-KI's...

Macht das bitte selbst, würde ich meinen KI Output hier pasten ist der so dass ich es verstehe, aber vielleicht nicht ihr...

Test-run

```
/bin/bash ~/bookstack-update.sh  
tail ~/bookstack-update.log
```

Dies wird beim ersten Ausführen eure bestehende Installation einmal mit der gleichen Version überschreiben, erst dann existiert das version-tracking file `/home/$USER/bookstack-latest-version.txt`

Cronjob-Automatismus

Nun ab in den Cronjob damit.

```
crontab -e  
MAILTO=""  
0 */2 * * * /bin/bash /home/$USER/bookstack-update.sh
```

Das Script erzeugt Output von den `curl` Befehlen, den würden wir im Falle von Überspaces per Mail gesammelt bekommen, wenn das MAILTO dort nicht wäre.

Denkt dran: `:wq` falls ihr in VI oder VIM gelandet seid. [Crontab-Guru](#) für andere Zeitintervalle.

Editor Shortcuts

Ganz ungeil kopiert von <https://www.bookstackapp.com/docs/user/wysiwyg-editor/>

Editor Shortcuts

The following shortcuts are available in the WYSIWYG Editor:

Shortcut (Windows & Linux/Mac)	Description
Ctrl+S / Cmd+S	Save Draft
Ctrl+Enter / Cmd+Enter	Save Page & Continue
Ctrl+B / Cmd+B	Bold
Ctrl+I / Cmd+I	Italic
Ctrl+U / Cmd+U	Underline
Ctrl+1 / Cmd+1 Ctrl+2 / Cmd+2 Ctrl+3 / Cmd+3 Ctrl+4 / Cmd+4	Header Large Header Medium Header Small Header Tiny
Ctrl+5 / Cmd+5 Ctrl+D / Cmd+D	Normal Paragraph
Ctrl+6 / Cmd+6 Ctrl+Q / Cmd+Q	Blockquote
Ctrl+7 / Cmd+7 Ctrl+E / Cmd+E	Code Block
Ctrl+8 / Cmd+8 Ctrl+Shift+E / Cmd+Shift+E	Inline Code
Ctrl+9 / Cmd+9	Callout (Keep pressing to toggle through styles)
Ctrl+0 / Cmd+0 Ctrl+P / Cmd+P	Ordered List Bullet List
Ctrl+Shift+K / Cmd+Shift+K	Link to BookStack content