Teedy API Scripts / database queries

Gastzugang und spezielle Anpassungen (serverseitiger Code/Scripts)

Wir wollen unsere Inventarplattform so aufbereiten, dass sie auch für Gäste etwas zum Stöbern bietet und die Funktionen von Teedy als praktibale Open Source Software zeigt. Teedy hat einen Gastmodus, der von Haus aus leider neben der regulären Verwendung auch ein gewisses Spamming per Web-Browser und API erlaubt. Zur Reduktion haben wir Scripts geschrieben, die dies unterbinden.

Wir möchten Kommentare und wild angelegte Tags und Dokumente damit vermeiden, da sie die Struktur zu leicht unlesbar machen. Gäste können Kommentare, Tags und Dokumente von anderen Gastsitzungen generell einfach modifizieren oder löschen. Deshalb eignet sich der Gastmodus nur als reiner Betrachtermodus. Selbiges trifft auch geteilte/generische) ReadOnly-Benutzer zu.

Die Speicher-Quota für den Gast und für sonstige ReadOnly-Benutzer beträgt 0 MB.

Neben Gast-Tags werden Tags von anderen Nutzern, die keine Admins oder Editoren sind, ebenso vom System erkannt und automatisch gelöscht. Wir behalten uns vor unser System für Mitglieder zu kapseln, sodass das Tool nur noch vereinsintern und nicht durch Gäste genutzt werden kann, falls zu viele Schabernackversuche auftreten.

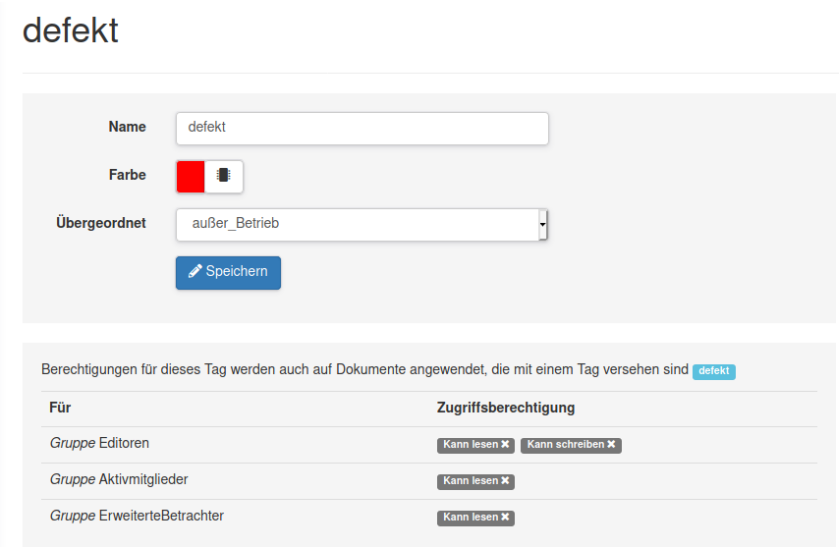
Scripts (diese laufen jeweils als cron job alle 10 Minuten)

- **Kommentare** (erzeugt von "guest" user) werden automatisch gescannt und gelöscht
  - Siehe Auto-delete guest comments.
  - Ein Wiederherstellen ist mit dem Zurücksetzen der **com_deletedate_d** Spalte möglich
  - Der Haupt-Admin erhält eine Mail über alle gelöschten Kommentare ins Webmaster-Postfach
- **Dokumente** (erzeugt von "guest" user oder anderen Benutzern, die nicht den Gruppen "Editoren" oder "Administratoren" angehören) werden automatisch gescannt und ebenso gelöscht
  - Siehe Auto-delete guest documents.
  - Ein Wiederherstellen ist mit dem Zurücksetzen der **doc_deletedate_d** Spalte möglich. Ein Wiederherstellen der Dateien ist allerdings nicht möglich
  - Der Haupt-Admin erhält eine Mail über alle gelöschten Dokumente ins Webmaster-Postfach

- **Tags** (erzeugt von "guest" user oder anderen Benutzern, die nicht den Gruppen "Editoren" oder "Administratoren" angehören) werden automatisch gescannt und gelöscht
  - Siehe Auto-delete guest tags
  - Ein Wiederherstellen ist mit dem Zurücksetzen der **tag_deletedate_d** Spalte möglich
  - Der Haupt-Admin erhält eine Mail über alle gelöschten Tags ins Webmaster-Postfach

Manuelle Pflege der Tag-Besitzer

Wer Tags sehen und bearbeiten kann, kann vom Admin entweder über die Datenbank oder das User Interface gesteuert werden.



Aus Gründen der Gesamtübersicht (Grafana ACL Tabelle) entfernen wir Einzelbenutzer und definieren **nur** Gruppen. Leider können in Teedy Tags nicht von ihrem Einzelbenutzer getrennt werden. Zumindest können die Tags nicht gelöscht werden, die vom Haupt-Admin erstellt worden sind. Letzteres ist nur über Datenbank Update SQL Scripts änderbar:

```
/*Finde alle Tags, die nicht dem Haupt-Admin gehören. Gast-Tags tauchen hier generell nicht
auf*/
SELECT
    T.tag_name_c AS "Tag",
    U.use_username_c AS "Ersteller",
    T.tag_createdate_d AS "Erstelldatum"
FROM
    t_tag AS T
```

```
JOIN t_user AS U ON U.use_id_c = T.tag_iduser_c
WHERE
    T.tag_deletedate_d IS NULL AND
    T.tag_iduser_c IN (
        SELECT
            use_id_c
        FROM t_user AS U
        JOIN t_user_group AS UG ON UG.ugp_iduser_c = U.use_id_c
        JOIN t_group AS G ON G.grp_id_c = UG.ugp_idgroup_c
        WHERE
            U.use_deletedate_d IS NULL AND
            UG.ugp_deletedate_d IS NULL AND
            G.grp_deletedate_d IS NULL
    ) AND
    U.use_id_c != 'admin'
;

/*
Übertrage Ersteller/Eigentümer des Tags von anderen Editor/Administratoren auf den Haupt-Administrator.
Dazu müssen nachträglich ebenso die ACLs angepasst werden. Die Spalte "tag_iduser_c" spielt für die Berechtigung keine direkte Rolle sondern enthält nur die Information, wer den Tag initial erstellt hat.
Aber da wir in den ACLs den User gegen admin tauschen, machen wir es somit konsistent!
*/
/*UPDATE t_tag SET tag_iduser_c = 'admin';*/ /*dieses Statement ist überflüssig und sollte nicht ausgeführt werden*/

/*
Finde alles, was nicht "Editoren", "ErweiterteBetrachter" oder "Aktivmitglieder" ist.
*/
SELECT
    A.acl_id_c,
    T.tag_name_c "Tag",
    A.acl_perm_c AS "Permission",
    U.use_username_c
FROM t_tag AS T
JOIN t_acl AS A ON A.acl_sourceid_c = T.tag_id_c
LEFT JOIN t_user AS U ON U.use_id_c = A.acl_targetid_c
```

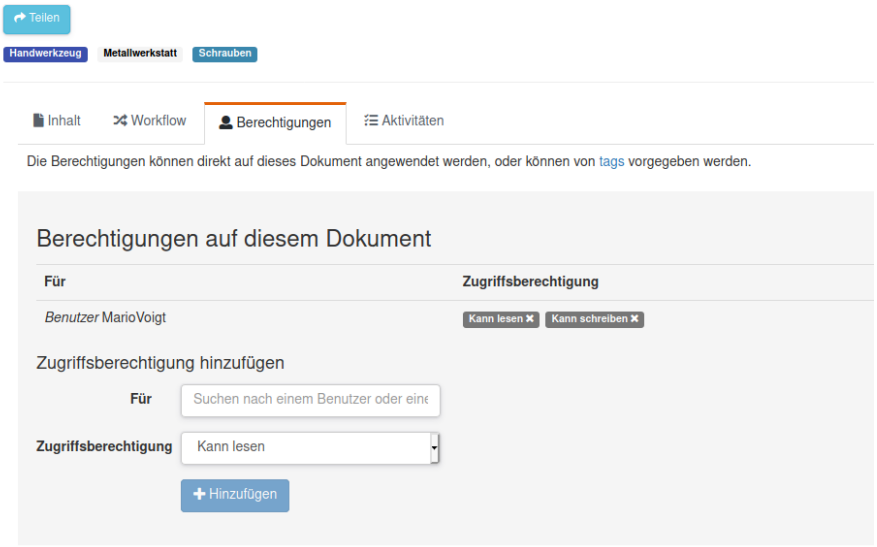
```
WHERE
    T.tag_deletedate_d IS NULL AND
    --A.acl_deletedate_d IS NULL AND
    U.use_disabledate_d IS NULL AND
    U.use_id_c = 'admin'
ORDER BY T.tag_name_c
;

/*
Wir wollen die ACLs nur auf Gruppen festlegen. Wir nehmen auch dem Haupt-Admin die
Benutzerberechtigungen, da dieser Einzelbenutzer die Übersichtlichkeit (in Grafana) künstlich
aufbläht.
Berechtigungen für Einzelnutzer soll es generell nicht geben (Ausnahme Tag "public" für den
User "guest")
*/
UPDATE t_acl SET acl_deletedate_d = NOW() WHERE acl_id_c IN (
    SELECT
        A.acl_id_c
    FROM t_tag AS T
    JOIN t_acl AS A ON A.acl_sourceid_c = T.tag_id_c
    LEFT JOIN t_user AS U ON U.use_id_c = A.acl_targetid_c
    WHERE
        T.tag_deletedate_d IS NULL AND
        U.use_disabledate_d IS NULL AND
        U.use_id_c = 'admin'
    )
;
```

Manuelle Pflege der Dokumenten-Besitzer

Wer Dokumente sehen und bearbeiten kann, kann vom Admin entweder über die Datenbank oder das User Interface gesteuert werden:

Feinmechanikerwerkzeug 08.11.2020 von ujoZzkKw2g




```
/*Finde alle Dokumente, die nicht dem Haupt-Admin gehören. Gast-Tags tauchen hier generell
nicht auf*/
SELECT
    D.doc_title_c AS "Gegenstand",
    U.use_username_c AS "Ersteller",
    D.doc_createdate_d AS "Erstelldatum"
FROM
    t_document AS D
JOIN t_user AS U ON U.use_id_c = D.doc_iduser_c
WHERE
    D.doc_deletedate_d IS NULL AND
    D.doc_iduser_c IN (
        SELECT
            use_id_c
        FROM t_user AS U
        JOIN t_user_group AS UG ON UG.ugp_iduser_c = U.use_id_c
        JOIN t_group AS G ON G.grp_id_c = UG.ugp_idgroup_c
        WHERE
            U.use_deletedate_d IS NULL AND
            UG.ugp_deletedate_d IS NULL AND
            G.grp_deletedate_d IS NULL
    ) AND
    U.use_id_c != 'admin'
;
```

```
/*
Übertrage Ersteller/Eigentümer des Tags von anderen Editor/Administratoren auf den Haupt-Administrator.
Dazu müssen nachträglich ebenso die ACLs angepasst werden. Die Spalte "doc_iduser_c" spielt für die Berechtigung keine direkte Rolle sondern enthält nur die Information, wer das Dokument initial erstellt hat.
Aber da wir in den ACLs den User gegen admin tauschen, machen wir es somit konsistent!
*/
/*UPDATE t_document SET doc_iduser_c = 'admin';*/ /*dieses Statement ist überflüssig und sollte nicht ausgeführt werden*/

/*
Finde alles, was nicht "Editoren", "ErweiterteBetrachter" oder "Aktivmitglieder" ist.
*/
SELECT
    A.acl_id_c,
    D.doc_title_c "Gegenstand",
    A.acl_perm_c AS "Permission",
    U.use_username_c
FROM t_document AS D
JOIN t_acl AS A ON A.acl_sourceid_c = D.doc_id_c
LEFT JOIN t_user AS U ON U.use_id_c = A.acl_targetid_c
WHERE
    D.doc_deletedate_d IS NULL AND
    --A.acl_deletedate_d IS NULL AND
    U.use_disabledate_d IS NULL AND
    U.use_id_c = 'admin'
ORDER BY D.doc_title_c
;
```

Auto-delete guest comments

In case you have a guest login enabled and don't want to accept guest spamming you can prevent it using the following bash script with cron trigger (running every 10 minutes). Guest comments are even useless because each guest can delete the guest comments from another guest session. So nobody can guarantee that those will exist a longer time. Deleting such stuff helps to keep clean useful documents which were **not created by guests** but regular users who wanted to put them to **public**.

Comment deletion

> This script is looking within i time fence of 10 minutes. If the script skipped in the meantime, it's possible that comments were overlooked. They have to be cleaned manually then.

```
vim /opt/teedy-clean-comments.sh
```

```
#!/bin/bash
#check for commments which have been created the last 10 minutes. if result is not empty we
send a new email
DB_USER="db_user"
DB_NAME="db_name"
OUT=$(psql -t -U$DB_USER $DB_NAME --no-align --command="
    SELECT
        t_document.doc_title_c,
        t_comment.com_content_c,
        t_comment.com_createdate_d||'\n'
    FROM t_comment
    JOIN t_user ON t_comment.com_iduser_c = t_user.use_id_c
    JOIN t_document ON t_comment.com_iddoc_c = t_document.doc_id_C
    WHERE
        t_document.doc_deletedate_d IS NULL AND
        t_comment.com_deletedate_d IS NULL AND
        t_user.use_username_c = 'guest' AND
        t_comment.com_createdate_d + interval '10 minute' >= now()
    ;
")
```

```
if [[ ! -z $OUT ]]; then
    #echo -e -n _${OUT}_
    #first inform about the comment via mail
    echo -e -n " "$OUT | mail -s "dms.yourdomain.de guest comments" post@fix.de

    OUT=$(psql -t -U$DB_USER $DB_NAME --no-align --command="
        SELECT
            t_comment.com_id_c
        FROM t_comment
        JOIN t_user ON t_comment.com_iduser_c = t_user.use_id_c
        JOIN t_document ON t_comment.com_iddoc_c = t_document.doc_id_C
        WHERE
            t_document.doc_deletedate_d IS NULL AND
            t_comment.com_deletedate_d IS NULL AND
            t_user.use_username_c = 'guest' AND
            t_comment.com_createdate_d + interval '10 minute' >= now()
        ;
    ")

    #echo $OUT

    BASE_URL="https://dms.yourdomain.de"
    BASE_URL="http://localhost:8080/dms"
    TEEDY_USER="teedy"
    AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
    if [ -z "$AUTH_TOKEN" ]
    then
        echo "NO AUTHTOKEN. Please create a session for the user first to automate things!"
>&2 #print to stderr to trigger cron.d mail on error
        exit 1
    else
        for VAR in $OUT; do
            curl --silent -X DELETE -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/comment/$VAR" -k
        done
    fi
else
    echo "Nothing to send and nothing to fix ..."
```

```
fi
```

cron.d script in `/etc/cron.d/teedy-clean-comments`

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
*/10 * * * * root /opt/teedy-clean-comments.sh > /dev/null
```

You can also directly perform a drop statement by SQL instead (but that might hurt the audit log)

```
DELETE FROM t_comment WHERE com_id_c IN (
SELECT
com_id_c
FROM t_comment
JOIN t_user ON t_comment.com_iduser_c = t_user.use_id_c
JOIN t_document ON t_comment.com_iddoc_c = t_document.doc_id_c
WHERE
t_document.doc_deletedate_d IS NULL AND
t_comment.com_deletedate_d IS NULL AND
t_user.use_username_c = 'guest' AND
t_comment.com_createdate_d + interval '1 minute' >= now())
;
```

Auto-delete guest documents

In case you have a guest login enabled and don't want to accept guest spamming you can prevent it using the following bash script with cron trigger (running every 10 minutes). Guest documents are even useless because each guest can delete the guest documents from another guest session. So nobody can guarantee that those will exist a longer time. Deleting such stuff helps to keep clean useful documents which were **not created by guests** but regular users who wanted to put them to **public**.

Document deletion

> This script is looking within i time fence of 10 minutes. If the script skipped in the meantime, it's possible that documents were overlooked. They have to be cleaned manually then.

```
vim /opt/teedy-clean-documents.sh
```

```
#!/bin/bash
#check for documents which have been created the last 10 minutes. if result is not empty we
send a new email
DB_USER="user"
DB_NAME="db"
OUT=$(psql -t -U$DB_USER $DB_NAME --no-align --command="
     SELECT
        D.doc_title_c,
        U.use_username_c,
        D.doc_createdate_d||'\n'
    FROM
        t_document AS D
    JOIN t_user AS U ON U.use_id_c = D.doc_iduser_c
    WHERE
        D.doc_deletedate_d IS NULL AND (
        D.doc_iduser_c IN (
        SELECT
            use_id_c
        FROM t_user AS U
        JOIN t_user_group AS UG ON UG.ugp_iduser_c = U.use_id_c
        JOIN t_group AS G ON G.grp_id_c = UG.ugp_idgroup_c
```

```
        WHERE
            U.use_deletedate_d IS NULL AND
            UG.ugp_deletedate_d IS NULL AND
            G.grp_deletedate_d IS NULL AND
            G.grp_name_c NOT IN ('Editoren','Administratoren') AND
            D.doc_createdate_d + interval '10 minute' >= now()
        ) OR
        D.doc_iduser_c = 'guest') AND /*guest ist in keiner Gruppe, deshalb muss er gesondert
aufgeführt werden*/
        D.doc_createdate_d + interval '10 minute' >= now()
    ;
    ")

if [[ ! -z $OUT ]]; then
    #echo -e -n _${OUT}_
    #first inform about the document via mail
    echo -e -n " "$OUT | mail -s "your.dms.de guest documents" webmaster@stadtfabrikanten.org

    OUT=$(psql -t -U$DB_USER $DB_NAME --no-align --command="
      SELECT
        D.doc_id_c
    FROM
        t_document AS D
    JOIN t_user AS U ON U.use_id_c = D.doc_iduser_c
    WHERE
        D.doc_deletedate_d IS NULL AND (
        D.doc_iduser_c IN (
        SELECT
            use_id_c
        FROM t_user AS U
        JOIN t_user_group AS UG ON UG.ugp_iduser_c = U.use_id_c
        JOIN t_group AS G ON G.grp_id_c = UG.ugp_idgroup_c
        WHERE
            U.use_deletedate_d IS NULL AND
            UG.ugp_deletedate_d IS NULL AND
            G.grp_deletedate_d IS NULL AND
            G.grp_name_c NOT IN ('Editoren','Administratoren') AND
            D.doc_createdate_d + interval '10 minute' >= now()
        ) OR
        D.doc_iduser_c = 'guest') AND /*guest ist in keiner Gruppe, deshalb muss er gesondert
```

```
aufgeführt werden*/
        D.doc_createdate_d + interval '10 minute' >= now()
    ;
   ")

    #echo $OUT

    BASE_URL="https://your.dms.de"
    BASE_URL="http://localhost:8080/dms"
    TEEDY_USER="pass"
    AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
    if [ -z "$AUTH_TOKEN" ]
    then
        echo "NO AUTHTOKEN. Please create a session for the user first to automate things!"
>&2 #print to stderr to trigger cron.d mail on error
        exit 1
    else
        for VAR in $OUT; do
            curl --silent -X DELETE -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/document/$VAR" -k
        done
    fi
else
    echo "Nothing to send and nothing to fix ..."
fi
```

cron.d script in `/etc/cron.d/teedy-clean-documents`

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
*/10 * * * * root /opt/teedy-clean-documents.sh > /dev/null
```

Auto-delete guest tags

In case you have a guest login enabled and don't want to accept guest spamming you can prevent it using the following bash script with cron trigger (running every 10 minutes). Guest tags are even useless because each guest can delete the guest tags from another guest session. So nobody can guarantee that those will exist a longer time. Deleting such stuff helps to keep clean useful tags which were **not created by guests** but regular users who wanted to put them to **public**.

Tag deletion

This script is looking within i time fence of 10 minutes. If the script skipped in the meantime, it's possible that comments were overlooked. They have to be cleaned manually then.

```
vim /opt/teedy-clean-tags.sh
```

```
#!/bin/bash
#check for tags which have been created the last 10 minutes. if result is not empty we send a
new email
DB_USER="user"
DB_NAME="db"
OUT=$(psql -t -U$DB_USER $DB_NAME --no-align --command="
    /*werden Tags gelöscht, wenn der Benutzer gelöscht wird?*/
    SELECT
        T.tag_name_c,
        U.use_username_c,
        T.tag_createdate_d||'\n'
    FROM
        t_tag AS T
    JOIN t_user AS U ON U.use_id_c = T.tag_iduser_c
    WHERE
        T.tag_deletedate_d IS NULL AND (
        T.tag_iduser_c IN (
        SELECT
            use_id_c
        FROM t_user AS U
        JOIN t_user_group AS UG ON UG.ugp_iduser_c = U.use_id_c
        JOIN t_group AS G ON G.grp_id_c = UG.ugp_idgroup_c
```

```
        WHERE
            U.use_deletedate_d IS NULL AND
            UG.ugp_deletedate_d IS NULL AND
            G.grp_deletedate_d IS NULL AND
            G.grp_name_c NOT IN ('Editoren','Administratoren') AND
            T.tag_createdate_d + interval '10 minute' >= now()
        ) OR
        T.tag_iduser_c = 'guest') AND /*guest ist in keiner Gruppe, deshalb muss er gesondert
aufgeführt werden*/
        T.tag_createdate_d + interval '10 minute' >= now()
    ;

")

if [[ ! -z $OUT ]]; then
    #echo -e -n _${OUT}_
    #first inform about the document via mail
    echo -e -n " "$OUT | mail -s "your.dms.de guest tags" post@fix.org

    OUT=$(psql -t -U$DB_USER $DB_NAME --no-align --command="
        SELECT
            T.tag_id_c
        FROM
            t_tag AS T
        JOIN t_user AS U ON U.use_id_c = T.tag_iduser_c
        WHERE
            T.tag_deletedate_d IS NULL AND (
            T.tag_iduser_c IN (
            SELECT
                use_id_c
            FROM t_user AS U
            JOIN t_user_group AS UG ON UG.ugp_iduser_c = U.use_id_c
            JOIN t_group AS G ON G.grp_id_c = UG.ugp_idgroup_c
            WHERE
                U.use_deletedate_d IS NULL AND
                UG.ugp_deletedate_d IS NULL AND
                G.grp_deletedate_d IS NULL AND
                G.grp_name_c NOT IN ('Editoren','Administratoren') AND
                T.tag_createdate_d + interval '10 minute' >= now()
            ) OR
```

```
            T.tag_iduser_c = 'guest') AND
            T.tag_createdate_d + interval '10 minute' >= now()
        ;
    ")

    #echo $OUT

    BASE_URL="https://your.dms.de"
    BASE_URL="http://localhost:8080/dms"
    TEEDY_USER="password"
    AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
    if [ -z "$AUTH_TOKEN" ]
    then
        echo "NO AUTHTOKEN. Please create a session for the user first to automate things!"
>&2 #print to stderr to trigger cron.d mail on error
        exit 1
    else
        for VAR in $OUT; do
            echo
            curl --silent -X DELETE -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/tag/$VAR" -k
        done
    fi
else
    echo "Nothing to send and nothing to fix ..."
fi
```

cron.d script in `/etc/cron.d/teedy-clean-comments`

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
*/10 * * * * root /opt/teedy-clean-tags.sh > /dev/null
```

Change owner of tags/files/documents

Within Teedy, there is no function to change the owner of a document. So we cannot transfer docs from one user to another. In case we want to keep documents in the system, we may not delete the author of the document. We can only deactivate the user. Under some circumstances this is going to make the system untidy.

> The following steps should be only be done in convenience with DSGVO.

Warning: all files in Teedy are encrypted by the users `use_privatekey_c` value from `t_user` table. In case we change the owner, we have to decrypt and encrypt the file again. We need to know the correct mapping between file and user each. So please do not try to migrate multiple users at once. At the moment we have no routine to do the re-encryption by console commands.

The belonging files are:

- https://github.com/sismics/docs/blob/master/docs-core/src/main/java/com/sismics/docs/core/listener/async/FileProcessingAsyncListener.java
- https://github.com/sismics/docs/blob/master/docs-core/src/main/java/com/sismics/docs/core/util/EncryptionUtil.java

So do the following steps at your own risk!

Stop Jetty

```
sudo systemctl stop jetty11
```

Take care to make a backup of `/var/docs` and the database before doing the following steps:

```
--get some info
select * from t_document where doc_title_c =  '<some title of the users OLD_ID to determine
his/her OLD_ID';
select * from t_document where doc_iduser_c = 'OLD_ID'; --we check the docs of the OLD_ID user


--now update the tables
```

```
update t_document set doc_iduser_c =  'NEW_ID' WHERE doc_iduser_c = 'OLD_ID';
update t_file set fil_iduser_c =  'NEW_ID' WHERE fil_iduser_c = 'OLD_ID';
update t_tag set tag_iduser_c =  'NEW_ID' WHERE tag_iduser_c = 'OLD_ID';
update t_comment set com_iduser_c =  'NEW_ID' WHERE com_iduser_c = 'OLD_ID';
update t_contributor set ctr_iduser_c =  'NEW_ID' WHERE ctr_iduser_c = 'OLD_ID';
update t_audit_log set log_iduser_c =  'NEW_ID' WHERE log_iduser_c = 'OLD_ID';
update t_acl set acl_targetid_c =  'NEW_ID' WHERE acl_targetid_c = 'OLD_ID';
```

```
sudo systemctl start jetty11
```

After the change, you should deactivate the old user in the UI backend.

Undelete document

Once a document was deleted, we can restore most information except document files:

```
SELECT * FROM  t_document WHERE doc_id_c = 'b9a538d2-906f-4566-b00d-ee4aa70d8ff8';
UPDATE t_document SET doc_deletedate_d = NULL  WHERE doc_id_c = 'b9a538d2-906f-4566-b00d-
ee4aa70d8ff8';
```

Scan for users without groups

To have better control about users, which logged in first time by LDAP and did not get mapped to default group, we use some SQL / bash script to inform the webmaster about that:

```
vim /opt/teedy-unmapped-users.sh
```

```
#!/bin/bash
#check for new users which have been created the last 10 minutes, but have no propery group
membership. if result is not empty we send a new email
DB_USER="db_user"
DB_NAME="db_name"
OUT=$(psql -t -U$DB_USER $DB_NAME --no-align --command="
SELECT
     use_id_c,
    use_username_c
FROM t_user AS U
JOIN t_user_group AS UG ON UG.ugp_iduser_c = U.use_id_c
JOIN t_group AS G ON G.grp_id_c = UG.ugp_idgroup_c
WHERE
    U.use_deletedate_d IS NULL AND
    U.use_disabledate_d IS NULL AND
    UG.ugp_deletedate_d IS NULL AND
    G.grp_deletedate_d IS NULL AND
    G.grp_name_c NOT IN ('Administratoren', 'Group2', 'Editors', 'Viewers') AND
    U.use_username_c NOT IN ('admin', 'anotherUser', 'anotherUser2')
;
")

if [[ ! -z $OUT ]]; then
    #echo -e -n _${OUT}_
    #first inform about the document via mail
    echo -e -n " "$OUT | mail -s "dms.domain.org new user(s) have to be mapped"
webmaster@domain.org

    BASE_URL="https://dms.domain.org"
    BASE_URL="http://localhost:8080/dms"
    TEEDY_USER="user"
```

```
    AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
    if [ -z "$AUTH_TOKEN" ]
    then
        echo "NO AUTHTOKEN. Please create a session for the user first to automate things!"
>&2 #print to stderr to trigger cron.d mail on error
        exit 1
    else
        for VAR in $OUT; do
            echo
            curl --silent -X DELETE -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/tag/$VAR" -k
        done
    fi
else
    echo "No users to map to groups ..."
fi
```

Create new documents which act as collectors

This is an example for document names with german titles. We call this script once per month to automatically create new documents. We can use https://crontab.guru/ to generate a time schedule.

```
vim /etc/cron.d/teedy-prepare-monthly.sh
```

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
0 1 1 * *     root   /opt/teedy-prepare-monthly.sh > /dev/null
# “At 01:00 on day-of-month 1.”
```

```
vim /opt/teedy-prepare-monthly.sh
```

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy"
DB_NAME="teedy_db"
TEEDY_USER="admin"
AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
if [ -z "$AUTH_TOKEN" ]
then
    echo "NO AUTHTOKEN. Please create a session for the user first to automate things!" >&2
#print to stderr to trigger cron.d mail on error
    exit 1
else
    THIS_MONTH=`date +'%m' -d 'now'` #return the recent month in format 01 ... 12
    THIS_YEAR=`date +'%Y' -d 'now'` #return the recent year
    #echo $THIS_MONTH
    #echo $THIS_YEAR

    #generate the date of the last day of the recent month
    TARGET_DATESTRING=$(date --date="$(date +$THIS_YEAR'-'$THIS_MONTH'-'01) + 1 month - 1 day
00:00" +"%s")000
    #echo $TARGET_DATESTRING
```

```
    #list of desired tags (clear name). Get the ID from database
    TAGID_SAMMELDOKUMENT=$(    psql -t -U$DB_USER $DB_NAME --command="SELECT tag_id_c FROM
t_tag WHERE tag_name_c = 'Sammeldokument' AND tag_deletedate_d IS NULL;")
    TAGID_RECHNUNG=$(          psql -t -U$DB_USER $DB_NAME --command="SELECT tag_id_c FROM
t_tag WHERE tag_name_c = 'Rechnung' AND tag_deletedate_d IS NULL;")
    TAGID_RECHNUNGSKORREKTUR=$(psql -t -U$DB_USER $DB_NAME --command="SELECT tag_id_c FROM
t_tag WHERE tag_name_c = 'Rechnungskorrektur' AND tag_deletedate_d IS NULL;")
    TAGID_AUFTRAG=$(           psql -t -U$DB_USER $DB_NAME --command="SELECT tag_id_c FROM
t_tag WHERE tag_name_c = 'Auftrag' AND tag_deletedate_d IS NULL;")
    TAGID_LIEFERSCHEIN=$(      psql -t -U$DB_USER $DB_NAME --command="SELECT tag_id_c FROM
t_tag WHERE tag_name_c = 'Lieferschein' AND tag_deletedate_d IS NULL;")
    TAGID_ANGEBOT=$(           psql -t -U$DB_USER $DB_NAME --command="SELECT tag_id_c FROM
t_tag WHERE tag_name_c = 'Angebot' AND tag_deletedate_d IS NULL;")

        TAGID_SAMMELDOKUMENT=${TAGID_SAMMELDOKUMENT:1}
             TAGID_RECHNUNG=${TAGID_RECHNUNG:1}
    TAGID_RECHNUNGSKORREKTUR=${TAGID_RECHNUNGSKORREKTUR:1}
              TAGID_AUFTRAG=${TAGID_AUFTRAG:1}
         TAGID_LIEFERSCHEIN=${TAGID_LIEFERSCHEIN:1}
              TAGID_ANGEBOT=${TAGID_ANGEBOT:1}

    #Create new documents - WARNING: NO CHECK FOR DUPLICATE DOCUMENTS RIGHT NOW
    curl --silent -X PUT -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/document" -d
"title=Ausgangsrechnungen "$THIS_YEAR"\\"$THIS_MONTH -d "create_date="$TARGET_DATESTRING -d
"language=deu" -d "tags="$TAGID_RECHNUNG -d "tags="$TAGID_SAMMELDOKUMENT
    curl --silent  -X PUT -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/document" -d
"title=Ausgangsrechnungskorrekturen "$THIS_YEAR"\\"$THIS_MONTH -d
"create_date="$TARGET_DATESTRING -d "language=deu" -d "tags="$TAGID_RECHNUNGSKORREKTUR -d
"tags="$TAGID_SAMMELDOKUMENT
    curl --silent  -X PUT -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/document" -d
"title=Ausgangsaufträge "$THIS_YEAR"\\"$THIS_MONTH -d "create_date="$TARGET_DATESTRING -d
"language=deu" -d "tags="$TAGID_AUFTRAG -d "tags="$TAGID_SAMMELDOKUMENT
    curl --silent  -X PUT -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/document" -d
"title=Ausgangslieferscheine "$THIS_YEAR"\\"$THIS_MONTH -d "create_date="$TARGET_DATESTRING -d
"language=deu" -d "tags="$TAGID_LIEFERSCHEIN -d "tags="$TAGID_SAMMELDOKUMENT
    curl --silent  -X PUT -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/document" -d
"title=Ausgangsangebote "$THIS_YEAR"\\"$THIS_MONTH -d "create_date="$TARGET_DATESTRING -d
"language=deu" -d "tags="$TAGID_ANGEBOT -d "tags="$TAGID_SAMMELDOKUMENT
```

```
fi
```

Find ugly document titles

This statement looks for titles with unrequired whitespace duplicates

```
/*title which contain doubled whitespaces*/
SELECT
    doc_title_c
FROM t_document
WHERE
    LENGTH(RTRIM(LTRIM(doc_title_c))) <> LENGTH(doc_title_c) OR
    doc_title_c LIKE '%  %' AND
    doc_deletedate_d IS NULL
;
```

Reindex / index repairing script

This script is for use as cron for example.

By teedy username and password

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
AUTH_TOKEN=$(curl -i -X POST -d username="username" -d password="password"
"$BASE_URL/api/user/login" -k|grep "auth_token"|cut -c24-59)
if [ -z "$AUTH_TOKEN" ]
then
    echo "NO AUTHTOKEN. Please create a session for the user first to automate things!" >&2
#print to stderr to trigger cron.d mail on error
    exit 1
else
    curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/app/batch/reindex" -k
    curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/user/logout" -k
fi
```

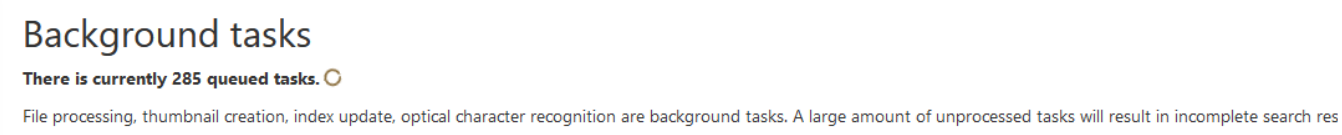
By database user and password

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy
DB_NAME="teedy_db"
TEEDY_USER="theuser"
AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
if [ -z "$AUTH_TOKEN" ]
then
    echo "NO AUTHTOKEN. Please create a session for the user first to automate things!" >&2
#print to stderr to trigger cron.d mail on error
    exit 1
else
    curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
```

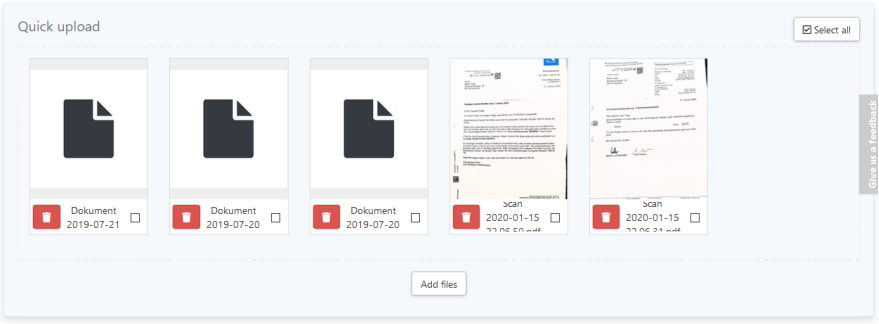
```
"$BASE_URL/api/app/batch/reindex" -k
fi
```

Reprocess all files for a given user

- Note that you can only re-process files which are your's! If you want to reprocess everything for everybody you will need to loop over each user while getting his username/passwort or auth_token (only way if user has 2FA secured account) from database. An auth_token can only stripped out from database if the user did not log off (otherwise it's null/empty).
- reprocessing makes sense in case of updating tesseract version which might improve the OCR text output quality. So for longterm storing of documents it might be worth to reprocess files.
- Reprocessing will raise a lot of background tasks into schedule:

- Note: Reprocessing of files does not work in quick upload mask (only works for files assigned to existing documents):

Way 1: pure API calls ? for a regular user (no 2FA secured)

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
TEEDY_USER="admin"
TEEDY_USER_PASS="password"
AUTH_TOKEN=$(curl -i -X POST -d username="$TEEDY_USER" -d password="$TEEDY_USER_PASS"
"$BASE_URL/api/user/login" -k|grep "auth_token"|cut -c24-59)
BACKUP_DIR="/backup/teedy"
TARGET_DOCLIST_JSON=$BACKUP_DIR"/documentlist_forfiles.json"
TARGET_FILELIST_JSON=$BACKUP_DIR"/filelist.json"
mkdir -p "$BACKUP_DIR"
rm $TARGET_DOCLIST_JSON
rm $TARGET_FILELIST_JSON
```

```
echo "Retrieving document list"
curl --silent -X GET -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/document/list?limit=0"
-k | jq . > "$TARGET_DOCLIST_JSON"
echo "Retrieving file list based on document list"
COUNT=0
jq -c '.|{documents}|.[]|.[]|{id}+{title}+{create_date}' "$TARGET_DOCLIST_JSON" | while read -
r i; do
    COUNT=$((COUNT + 1))
    DOC_ID=$(jq -c '.|{id}|.id' <<< $(printf '%s\n' "$i"))
    DOC_ID=${DOC_ID:1:-1}
    curl --silent -X GET -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/list?id=$DOC_ID" >> "$TARGET_FILELIST_JSON"
    echo Getting $COUNT : $DOC_ID
    #put some sleep time here if your server has less ressources. Otherwise you might overload
PostgreSQL as well as Jetty. It leads to unstability of the running instance
    #have a look at https://dms.yourdomain.de/#/settings/monitoring to check the average OCR
time per document. Usually 3 to 10 seconds should be normal
    #sleep 5
done
echo "Starting to process files"
COUNT=0
jq -c '.[]|.[]|{id}' "$TARGET_FILELIST_JSON" | while read -r i; do
        COUNT=$((COUNT + 1))
    FILE_ID=$(jq -c '.|{id}|.id' <<< $(printf '%s\n' "$i"))
        FILE_ID=${FILE_ID:1:-1}
        echo Processing $COUNT : $FILE_ID
        curl  --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/$FILE_ID/process"
done
curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/user/logout" -k
```

Way 2: API + psql calls ? for a regular user (no 2FA secured)

This script is much shorter and more elegant to reprocess. Note that this script logins in the user by a fresh created token. It will fail if the user login is secured with 2FA. In this case see below!

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy"
DB_NAME="teedy_db"
TEEDY_USER="admin"
TEEDY_USER_PASS="password"
AUTH_TOKEN=$(curl -i -X POST -d username="$TEEDY_USER" -d password="$TEEDY_USER_PASS"
"$BASE_URL/api/user/login" -k|grep "auth_token"|cut -c24-59)

for FILE_ID in $(psql -t -U$DB_USER $DB_NAME --command="SELECT fil_id_c FROM t_file AS F JOIN
t_user AS U ON U.use_id_c = F.fil_iduser_c WHERE F.fil_deletedate_d IS NULL AND
U.use_username_c = '$TEEDY_USER';"); do
    echo PROCESSING "$FILE_ID"
    curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/$FILE_ID/process"
    sleep 5
done

#logout
curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/user/logout" -k
```

Way 3: API + psql calls ? for a single 2FA secured user

The following script can be used if your user is secured with 2FA. Note that the user needs to be logged in once (if he logs off the recent token will be destroyed).

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy"
DB_NAME="teedy_db"
TEEDY_USER="admin"

#this reads exactly one token from the given user. If the user is not logged in it will be
null and the token will be null too!
AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
```

```
for FILE_ID in $(psql -t -U$DB_USER $DB_NAME --command="SELECT fil_id_c FROM t_file AS F JOIN t_user AS U ON U.use_id_c = F.fil_iduser_c WHERE F.fil_deletedate_d IS NULL AND U.use_username_c = '$TEEDY_USER';"); do
        echo PROCESSING "$FILE_ID"
        #curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/file/$FILE_ID/process"
    sleep 2
done

#do not logout to keep the token
```

Way 4: API + psql calls ? loop over all users (2FA secured users)

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy"
DB_NAME="teedy_db"
TEEDY_USERS=$(psql -t -U$DB_USER $DB_NAME --command="SELECT use_username_c FROM t_user;")

for TEEDY_USER in $TEEDY_USERS; do
  echo LOGGING IN AS $TEEDY_USER
  #this reads exactly one token from the given user. If the user is not logged in it will be null and the token will be null too!
  AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
  for FILE_ID in $(psql -t -U$DB_USER $DB_NAME --command="SELECT fil_id_c FROM t_file AS F JOIN t_user AS U ON U.use_id_c = F.fil_iduser_c WHERE F.fil_deletedate_d IS NULL AND U.use_username_c = '$TEEDY_USER';"); do
    echo PROCESSING "$FILE_ID"
    curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/file/$FILE_ID/process"
    sleep 2
  done
done
#do not logout after processing (to keep the token alive!)
```