Working around the temporary blobs fault in Deskpro <2022.0.1

Christopher Nadeau - 2022-03-02 - Comments (0) - Deskpro Legacy

In 2022.0.1 an update was rolled out to address an issue to do with *temporary blobs* remaining public even after a user may have abandoned them. This was due to a fault in automatic cleanup that would fail to remove temporary blobs properly.

Question
What is a temporary blob? A temp blob is a file attachment or upload that is accepted by the server but is not retained until the user submits their form in full. If the user leaves before their form is submitted, then the temp blob should be cleaned up after some time.

The update addresses two issues:

1) Temporary blobs should now be cleaned up as expected when a user abandons them.

2) Mitigations were put in place to prevent *old* temporary blobs from being visible.

Mitigations for old temporary blobs
Your database may contain many blobs marked as temporary. However, due to the bug described above, we cannot necessarily trust that flag in the database, and therefore can't automatically delete every blob that is marked so. There are a few mitigations that help with this.

Note: If you are already using database storage for blobs, you don't necessarily need to do anything - the mitigation to prevent serving temp blobs (the first point below) will already be in effect.

i) Move temp blobs to database storage

When a blob is served from the database, the system can detect when it is too old, and prevent it from being served (i.e. return a 404 to the user). This has the advantage in that old temp blobs won't be visible, but they won't be deleted either - which means you may restore them in the event a blob was erroneously marked as temp.

To move all temp blobs to the database, execute this MySQL query:

```
UPDATE blobs
SET storage_loc_pref = 'db'
WHERE storage_loc != 'db' AND is_temp = 1
```

This will make the system start to slowly move blobs back into the database on cron. This may take a long time depending on the number of blobs affected.

Pros:

- This is the safest option because it does not delete any data. In the event an old blob needs to be restored, you may simply toggle off the *is_temp* flag on the record.

Cons:

- This will increase the size of your database (possibly even significantly)
- This will not actually remove abandoned blobs.

Correcting an erroneously tagged temp blob

If you find a case where a blob appears to have gone "missing", then you may make it visible again by removing the temp flag on it. Given a URL to a blob like `file.php/xxx/21601380XXXXXXXXXXXXXXX/...` you'll note a long string of characters in the middle; it begins with a number, then a string of characters. This *authcode* identifies the blob and you can toggle the flag with a query like:

```
UPDATE blobs
SET is_temp = 0
WHERE authcode = '21601380XXXXXXXXXXXXXXX';
```

ii) Detect dangling blobs and delete them

There is a utility that will attempt to scan the database for all uses of blobs to detect "dangling blobs" - that is, blobs that appear to have no recorded use elsewhere. This tool is run from the command line:

```
cd path/to/deskpro

# get a summary of findings
php bin/console dp:utility:dangling-blobs --where "blobs.is_temp=1"

# actually delete the found records
# ALWAYS backup before performing deletes
php bin/console dp:utility:dangling-blobs --where "blobs.is_temp=1" --delete
```

Pros:

- This will actually *remove* the affected blobs

Cons:

- Deleting blobs this way is permanent.
- There are certain cases where a temp blob might not have an explicit database relationship, but may be used in such a way as to appear normal to end-users. E.g. if you are hotlinking to a blob from somewhere, the blob itself may not have an explicit relationship but that hotlink may still be used/visible by end-users.
- For larger databases, this may be a very long and intensive process.

Enable Attachment Authorisation

You can also enable attachment auth from Admin > Tickets > Settings to force a logged-in session to view any ticket attachment. This removes the possibility for a user to view an attachment, even if they have the full authcode, without being explicitly logged-in.