Upload in Nuxeo

Requirements

When uploading files to Nuxeo, we have the following requirements :

- upload progress must be trackable client side
  - server side tracking is false because of reverse proxies
- upload should be done outside of transaction
  - otherwise transaction will timeout
- upload should be resumable
  - otherwise uploading large files over a broken connection could take days
- upload should check for duplicate
  - to avoid many users to upload the same file
- upload should avoid temporary storages
  - to avoid copying the data multiple times

The first 2 requirements are handled by the `BatchManager` and the `BatchUpload` Endpoint.

This code was initially introduced for Android SDK and is now used by D&D, Nuxeo Drive and Automation API.

The 3 other requirements needs to be addressed.

BatchUpload improvements

Upload Resume

Currently the `BatchManager` does not provide any API for resuming an upload, however this would be very simple to add.

We probably need to add 2 APIs on the REST EndPoint:

- `HEAD /upload` (with all headers) : return the length of the uploaded stream
- `PUT /upload` with all headers + `X-File-Resume` : continue upload

If we think that Drive implementation is easy and a priority : server side implementation can be very fast and should be back portable.

Check for existing entry

We need to add again 2 APIs :

- `HEAD /upload/digest` : to check if binary exists
- `POST /upload` with a X-File-Digest header to "upload the file"

This second API is useful because we may actually want to create a document using a Blob that is already on the server, so we need to create an entry on the BatchManager.

This implementation should be simple.

Direct upload

How it works today The batch Manager stores the stream in temporary files.

These tmp files will be used to create Blobs that will be associated to a Document and that finally will be moved to the BinaryManager.

The final step will involve a copy of the file to write it inside the binary manager : the `storeAndDigest` will read / compute digest / write the stream to a file.

With S3 BinaryManager this read/digest/write will occur on a temporary store and then the file will be copied over S3.

What we may want to improve Client side upload is supposed to be limited by client side connectivity : S3/NAS accesss is likely to be faster than the http channel used by the client to upload the file.

So, we could leverage this to automatically have the batch manager write the content into the BinaryManager :

- no more file duplication
- no more slow write on S3 during end of transaction

Induced problems Doing so would create several issues

- BinaryManager would contain temporary streams
  - *this is actually not really an isssue : we have GC for that*
- Resume download is a problem
  - BinaryManager resolve stream according to their digest : you can not find it without a digest
  - you don't have the Digest if you did not finish the upload

Changing the BinaryManager API To be able to achieve what we want we need to change the BinaryManager API.

Current APIs :

```
Binary getBinary(Blob blob) throws IOException;

Binary getBinary(String digest);
```

We would need to add 3 new APIs :

```
void storeTemporaryStream(InputStream is, String uuid, long offset);

InputStream getTemporaryStream(String uuid);

Binary propoteToBinary(String uuid);
```

Having this API we could : store temporary stream using a uuid and complete the stream as needed, then move it to the digest file when validated.