Stash

Sort order/Filter: The default sort order of items of the tabs is stored with the default filter. Overwriting an existing filter does not overwrite the sort order though. The filter needs to be deleted, then saved and set to default again. The filters are stored in the database, not the configuration file.

Plugins: https://github.com/stashapp/CommunityScripts/

User script for local performer face detection: https://github.com/cc1234475/visage

https://github.com/cc1234475/visage-ml

https://github.com/stashapp/CommunityScripts/pull/180/files

https://github.com/stashapp/CommunityScripts/issues/186

Custom CSS/JS

Custom CSS/Javascript needs to be enabled in Settings→Interface→Custom CSS→Enable then Edit

Note: Firefox before v120/121 needs: about:config value of layout.css.has-selector.enabled set to true

```
/* [Performer tab] Show more item per row */
@media only screen and (min-device-width: 768px){
    :not(.recommendation-row .performer-card).performer-card {
        width: 15%;
    }
    :not(.recommendation-row .performer-card-image).performer-card-image {
        width: 100%;
    }
    .performer-card h5 {
        text-align: center !important;
        display: block;
    }
    .performer-card-image {
        height: 18rem;
    }
}

/* [Studios tab] Show more item per row */
:not(.recommendation-row .studio-card).studio-card {
    width: 15%
}
:not(.recommendation-row .studio-card-image).studio-card-image {
    width: 100%
}
.studio-card h5 {
    text-align: center !important;
```

```
    display: block;
}


/* [Global changes] Hide the Donate button */
.btn-primary.btn.donate.minimal {
  display: none;
}
button.minimal.donate.btn.btn-primary {
  display: none;
}


/* [Global changes] Modify card when checkbox is selected
Note: Firefox before v120/121 needs: about:config value of layout.css.has-selector.enabled set to true
*/
.grid-card.card:has(input:checked) {
   box-shadow: 0 0 0 1px var(--primary,rgba(255, 255, 255, 0.30));
}

/* Fix face AI plugin being popup being behind the content */
.face-tabs.svelte-p95y28 {
    // height: 60%;
    z-index: 1;
}
```

Scene Search at top

This moves the scene search from the sidebar to the top bar

```
// Move search from sidebar next to filter at top
(function() {
  const sourceSelector = 'body div div.main.container-fluid div.item-list-container.scene-list div.sidebar-pane div.sidebar div.sidebar-search-container';
  const targetSelector = 'body div div.main.container-fluid div.item-list-container.scene-list.hide-sidebar div.sidebar-pane.hide-sidebar div div.scene-list-toolbar.btn-toolbar';

  const observer = new MutationObserver(() => {
    const sourceElement = document.querySelector(sourceSelector);
    const targetContainer = document.querySelector(targetSelector);

    if (sourceElement && targetContainer) {
      targetContainer.insertBefore(sourceElement, targetContainer.lastChild);
      observer.disconnect(); // Stop observing once done
      console.log('Moved element to target container.');
    }
```

```
  });

  // Observe the entire document for added/removed elements
  observer.observe(document.body, {
    childList: true,
    subtree: true
  });
})();
```

Studio as Text

This replaces unset studio logos with a clickable text link.

CSS

```
/* Studio as text */
a[data-processed] {
    font-size: 1.5rem;
    color: inherit;
    text-decoration: none;
    display: inline-block;
}

a[data-processed]:hover {
    text-decoration: underline;
}
```

Javascript

```
/* Studio as text */
function processStudioLogos() {
    document.querySelectorAll('.studio-logo').forEach(img => {
        const link = img.closest('a');
        if (link && img.alt && !link.hasAttribute('data-processed') &&
img.src.includes('default=true')) {
            const cleanName = img.alt.replace(/\s+logo$/i, '');
            link.textContent = cleanName;
            link.setAttribute('data-processed', 'true');
        }
    });
}

const observer = new MutationObserver(function(mutations) {
    mutations.forEach(function(mutation) {
        if (mutation.addedNodes.length) {
            processStudioLogos();
        }
    });
});
```

```
observer.observe(document.body, {
    childList: true,
    subtree: true
});
```

api

https://docs.stashapp.cc/networking/api/

https://github.com/stashapp/stash/issues/3918

```
#If API authentication is required, add the following to the curl commands:
-H "ApiKey: YOUR_API_KEY_HERE" \
```

```
# Metadata Scan of a particular directory
curl -X POST http://192.168.1.2:9998/graphql \
  -H "Content-Type: application/json" \
  -d '{
    "query": "mutation MetadataScan($paths: [String!]!) {
metadataScan(input: { paths: $paths }) }",
    "variables": {
      "paths": ["/data/stash-temp"]
    }
  }'
```

```
# Full Metadata Scan
curl -X POST http://localhost:9998/graphql \
  -H "Content-Type: application/json" \
  -d '{"query": "mutation { metadataScan(input: {}) }"}'
```

Video Compare Userscript

This needs violentmonkey or similar browser plugin.

Base script from: https://gist.github.com/DogmaDragon/fb3ed033c0d1f0a6811137dfea0c4ce8

video_compare_userscript.user.js

```
// ==UserScript==
// @name         Stash Video Compare Userscript
// @description Userscript for Stash
// @match        http://192.168.1.2:9999/*
// @match        http://192.168.1.2:9998/*
// @grant        GM_openInTab
// @require      https://code.jquery.com/jquery-1.12.4.min.js
// @require
https://gist.github.com/raw/2625891/waitForKeyElements.js
```

```
// @version     1.1
// @author      Scruffynerf / AnonTester
// ==/UserScript==

(function() {
  'use strict';
  console.log('Script Stash initialize');
  waitForKeyElements(".navbar-brand", addbutton);

  function addbutton() {
    const navBar = document.querySelector(".navbar-nav");
    const buttonClass = navBar.firstChild.attributes.class.value;
    const linkClass =
navBar.firstChild.firstChild.attributes.class.value;
    const newButton = document.createElement("div");
    newButton.setAttribute("class", buttonClass);
    newButton.onclick = compare
    const innerLink = document.createElement("a");
    innerLink.setAttribute("class", linkClass);
    const buttonLabel = document.createElement("span");
    buttonLabel.innerText = "Video Compare";
    innerLink.appendChild(buttonLabel);
    newButton.appendChild(innerLink);
    navBar.appendChild(newButton);
  }

  function compare() {
    if (window.location.pathname == "/sceneDuplicateChecker") {
      var numberOfChecked = document.querySelectorAll('input.position-
static[type="checkbox"]:checked').length;
      if (numberOfChecked == 2) {
        var lr = []
        const list =
document.querySelectorAll('input[type="checkbox"].position-
static:checked')
        for (let item of list) {
          const row = item.closest('tr');
          const nextAnchor = row.querySelector('a');
          const url =
nextAnchor.getAttribute('href').replace(/\/scenes\//g, '/scene/')
          lr.push(window.location.origin.concat(url))
        }
        var site = "http://scruffynerf.stashapp.cc"
        var url = site.concat("?leftVideoUrl=", lr[0],
"/stream&rightVideoUrl=", lr[1], "/stream&hideHelp=1")
        GM_openInTab(url, true);
      }
    } else {
      var numberOfChecked =
document.querySelectorAll('input[type="checkbox"]:checked').length;
      if (numberOfChecked == 2) {
```

```
        const r = /[^"]+\/scene\/\d+\/s/ms;
        var lr = []
        const list =
document.querySelectorAll('input[type=checkbox]:checked')
        for (let item of list) {
          var urlstuff = item.nextElementSibling.innerHTML
          var m = r.exec(urlstuff)
          lr.push(m[0])
        }
        var site = "http://scruffynerf.stashapp.cc"
        var url = site.concat("?leftVideoUrl=", lr[0],
"tream&rightVideoUrl=", lr[1], "tream&hideHelp=1")
        GM_openInTab(url, true);
      }
    }
  }
})();
```

Scene merge script after video conversions

After converting video files already included in stash and doing a rescan, new scenes will be created for the new files. This leaves duplicate scenes in the database.

This script creates a backup of the stash database first, then finds all scenes in stash that have the same filenames but different extensions, merges the scenes into the older one and optionally delete the non-mp4 file from the merged scene.

This script uses stashapp-tools module to interface with stash. Installation of this:

```
pip install stashapp-tools
```

This script performs a dry run unless the -n option is provided to avoid data loss. It can take a filename or part of a path as option or the -a option to scan all stash scenes. The -d parameter deletes the non-mp4 files after merging scenes.

```
usage: stash-merge.py [-h] [-a] [-d] [-n] [filename]

Merge stash scenes with the same filenames in different formats, optionally
delete non-mp4 files.

positional arguments:
  filename           The filename to process

options:
  -h, --help         show this help message and exit
  -a, --all          Process all scenes with duplicate filenames
  -d, --delete       Delete non mp4 files after merging scenes
  -n, --non-dry-run  Actually perform actions, runs in dry-run mode by
```

```
default
```

Adjust the scheme/IP/port at the end of the script to your requirements if not http, localhost and port 9999.

stash-merge.py

```
import stashapi.log as log
from stashapi.stashapp import StashInterface
import argparse
import os, sys

def find_scenes_by_path_regex(self, f:dict={}, filter:dict={"per_page":
-1}, q:str="", fragment=None, get_count=False, callback=None):

    query = """
        query FindScenes($filter: FindFilterType) {
            findScenesByPathRegex(filter: $filter) {
                count
                scenes {
                    ...Scene
                }
            }
        }
        """
    if fragment:
        query = re.sub(r'\.\.\.Scene', fragment, query)

    filter["q"] = q
    variables = {
        "filter": filter,
        "scene_filter": f
    }

    result = self.call_GQL(query, variables, callback=callback)
    if get_count:
        return result['findScenesByPathRegex']['count'],
result['findScenes']['scenes']
    else:
        return result['findScenesByPathRegex']['scenes']

def merge_scenes_with_same_filename(stash_scheme, stash_host,
stash_port, delete_non_mp4=False, dryrun=True, file=None):
    stash = StashInterface({
        "scheme": stash_scheme,
        "host": stash_host,
        "port": stash_port,
        "logger": log
    })

    if dryrun:
```

```
        print("Dry-run:\n")
    else:
        print("Live-run:\n")

        print("Backing up database ...")
        query = """
        mutation backupDatabase{
            backupDatabase(input: {download: false})
        }
        """
        variables = {}
        stash.call_GQL(query, variables)

    if file is None: # Fetch all scenes
        scenes = stash.find_scenes()
    else:
        # note rsplit fails for files without extension in directories
starting with dot, but path in stash db is absolute
        file = file.rsplit('.', maxsplit=1)[0]
        scenes = find_scenes_by_path_regex(stash,q=file)
        #print(scenes)

    # print(len(scenes))

    # Group scenes by path without extension
    scenes_by_path_ex_ext = {}
    for scene in scenes:
        #print(scene)
        for files in scene['files']:
            # note rsplit fails for files without extension in
directories starting with dot, but path in stash db is absolute
            path = files['path'].rsplit('.', maxsplit=1)[0]
            if path not in scenes_by_path_ex_ext:
                scenes_by_path_ex_ext[path] = []
            scenes_by_path_ex_ext[path].append(scene)
    #print(len(scenes_by_path_ex_ext))

    for path, scene_group in scenes_by_path_ex_ext.items():
        if len(scene_group) > 1:
            if dryrun:
                print(f"Scene group:\n{scene_group}")
            # Identify the target scene to merge into
            #target_scene = max(scene_group, key=lambda s:
(bool(s['performers']), bool(s['stash_ids']), s['organized']==True))
            #always merge into the lowest scene id
            target_scene = sorted(scene_group, key=lambda s:
s['id'])[0]
            if dryrun:
                print(f"Target scene:\n{target_scene}")
            target_scene_id = target_scene['id']
            mp4_file_id = None
```

```
            print(f"Target scene id: {target_scene_id} Path: {path}")

            for scene in scene_group:
                if scene['id'] != target_scene_id:
                    # Merge scene into target
                    print(f"Merging scene {scene['id']} into
{target_scene_id} Source title: {scene['title']}")
                    if not dryrun:
                        stash.merge_scenes(scene['id'],
target_scene_id)

                # Identify mp4 file
                for files in scene['files']:
                    if files['path'].endswith('.mp4'):
                        mp4_file_id = files['id']
                        print(f"scene with mp4 path: {scene['id']} file
id: {mp4_file_id} path: {files['path']}")

            # Set the primary file to be the mp4 file
            if mp4_file_id:
                print(f"setting primary id to {mp4_file_id}")
                if not dryrun:
                    # first set all files NOT future primary to not be
primary
                    stash.sql_commit("UPDATE `scenes_files` SET
`primary`=? WHERE ((`scenes_files`.`scene_id` = ?) AND
(`scenes_files`.`file_id` != ?))", ("0", target_scene_id, mp4_file_id)
)
                    # now set new primary file id
                    stash.sql_commit("UPDATE `scenes_files` SET
`primary`=? WHERE ((`scenes_files`.`scene_id` = ?) AND
(`scenes_files`.`file_id` = ?))", ("1", target_scene_id, mp4_file_id) )

            # Optionally delete non-mp4 files if more than 1 file is
assigned to the scene
            if delete_non_mp4 and len(scene_group) > 1 and mp4_file_id:
                for scene in scene_group:
                    for files in scene['files']:
                        if files['id'] != mp4_file_id:
                            print(f"Deleting file id {files['id']}
while keeping {mp4_file_id} Deleted file path: {files['path']}")
                            if not dryrun:
                                stash.destroy_files(files['id'])
            print("")

def main():
    parser = argparse.ArgumentParser(description="Merge stash scenes
with the same filenames in different formats, optionally delete non-mp4
files.")

    # Optional argument for filename
```

```
    parser.add_argument('filename', nargs='?', default=None, help='The
filename to process')

    # Non-positional flags with short forms
    parser.add_argument('-a', '--all', action='store_true',
help='Process all scenes with duplicate filenames')
    parser.add_argument('-d', '--delete', action='store_true',
help='Delete non mp4 files after merging scenes')
    parser.add_argument('-n', '--non-dry-run', action='store_true',
help='Actually perform actions, runs in dry-run mode by default')

    if len(sys.argv)==1:
        print("Error: No filename or arguments provided.\n")
        parser.print_help(sys.stderr)
        sys.exit(1)

    args = parser.parse_args()

    if args.filename:
        FILE=args.filename

    if args.filename and args.delete:
        print(f"Merge scenes and delete non mp4 files:
{args.filename}\n")
        merge_scenes_with_same_filename(STASH_SCHEME, STASH_HOST,
STASH_PORT, delete_non_mp4=True, dryrun=not args.non_dry_run,
file=args.filename)
    elif args.filename:
        print(f"Merge scenes and keeping files containing:
{args.filename}\n")
        merge_scenes_with_same_filename(STASH_SCHEME, STASH_HOST,
STASH_PORT, delete_non_mp4=False, dryrun=not args.non_dry_run,
file=args.filename)
    elif args.all and args.delete:
        print("Merge all scenes with duplicate files, deleting
duplicate files.\n")
        merge_scenes_with_same_filename(STASH_SCHEME, STASH_HOST,
STASH_PORT, delete_non_mp4=True, dryrun=not args.non_dry_run,
file=None)
    elif args.all:
        print("Merge all scenes with duplicate files, not deleting
duplicate files.\n")
        merge_scenes_with_same_filename(STASH_SCHEME, STASH_HOST,
STASH_PORT, delete_non_mp4=False, dryrun=not args.non_dry_run,
file=None)
    else:
        print("No filename or all argument provided, not doing
anything\n")

if __name__ == "__main__":
    # Replace with your Stash app URL
```

```
    STASH_SCHEME = "http"
    STASH_HOST = "localhost"
    STASH_PORT = "9999"

    main()
    #merge_scenes_with_same_filename(STASH_SCHEME, STASH_HOST,
STASH_PORT, DELETE, DRYRUN, FILE)
```

From:
http://wuff.dyndns.org/ - **Wulf's Various Things**

Permanent link:
http://wuff.dyndns.org/doku.php?id=config:stash&rev=1755866010

Last update: **2025/08/22 13:33**