introduction

Creating custom downloaders is only for advanced users who understand HTML or JSON. Beware! If you are simply looking for how to add new downloaders, please head over here.

this system

The first versions of hydrus's downloaders were all hardcoded and static--I wrote everything into the program itself and nothing was user-creatable or -fixable. After the maintenance burden of the entire messy system proved too large for me to keep up with and a semi-editable booru system proved successful, I decided to overhaul the entire thing to allow user creation and sharing of every component. It is designed to be very simple to the front-end user--they will typically handle a couple of png files and then select a new downloader from a list--but very flexible (and hence potentially complicated) on the back-end. These help pages describe the different compontents with the intention of making an HTML- or JSON- fluent user able to create and share a full new downloader on their own.

As always, this is all under active development. Your feedback on the system would be appreciated, and if something is confusing or you discover something in here that is out of date, please let me know.

what is a downloader?

In hydrus, a downloader is one of:

- ### Gallery Downloader
- This takes a string like 'blue_eyes' to produce a series of thumbnail gallery page URLs that can be parsed for image page URLs which can ultimately be parsed for file URLs and metadata like tags. Boorus fall into this category.
- ### URL Downloader
- This does just the Gallery Downloader's back-end--instead of taking a string query, it takes the gallery or post URLs directly from the user, whether that is one from a drag-and-drop event or hundreds pasted from clipboard. For our purposes here, the URL Downloader is a subset of the Gallery Downloader.

- ### Watcher
- This takes a URL that it will check in timed intervals, parsing it for new URLs that it then queues up to be downloaded. It typically stops checking after the 'file velocity' (such as '1 new file per day') drops below a certain level. It is mostly for watching imageboard threads.
- ### Simple Downloader
- This takes a URL one-time and parses it for direct file URLs. This is a miscellaneous system for certain simple gallery types and some testing/'I just need the third <img> tag's *src* on this one page' jobs.

The system currently supports HTML and JSON parsing. XML should be fine under the HTML parser--it isn't strict about checking types and all that.

what does a downloader do?

The Gallery Downloader is the most complicated downloader and uses all the possible components. In order for hydrus to convert our example 'blue_eyes' query into a bunch of files with tags, it needs to:

- Present some user interface named 'safebooru tag search' to the user that will convert their input of 'blue_eyes' into

  https://safebooru.org/index.php?page=post&s=list&tags=blue_eyes&pid=0.
- Recognise https://safebooru.org/index.php?page=post&s=list&tags=blue_eyes&pid=0 as a Safebooru Gallery URL.
- Convert the HTML of a Safebooru Gallery URL into a list URLs like

  https://safebooru.org/index.php?page=post&s=view&id=2437965 and possibly a 'next page' URL (e.g.

  https://safebooru.org/index.php?page=post&s=list&tags=blue_eyes&pid=40) that points to the next page of thumbnails.
- Recognise the https://safebooru.org/index.php?page=post&s=view&id=2437965 URLs as Safebooru Post URLs.
- Convert the HTML of a Safebooru Post URL into a file URL like

  https://safebooru.org//images/2329/b6e8c263d691d1c39a2eeba5e00709849d8f864d.jpg and some tags like: 1girl, bangs, black gloves, blonde hair, blue eyes, braid, closed mouth, day, fingerless gloves, fingernails, gloves, grass, hair ornament, hairclip, hands clasped, creator:hankuri, interlocked fingers, long hair, long sleeves, outdoors, own hands together, parted bangs, pointy ears, character:princess zelda, smile, solo, series:the legend of zelda, underbust.

So we have three components:

- **Gallery URL Generator (GUG):** faces the user and converts text input into initialising Gallery URLs.
- **URL Class:** identifies URLs and informs the client how to deal with them.
- **Parser:** converts data from URLs into hydrus-understandable metadata.

URL downloaders and watchers do not need the Gallery URL Generator, as their input *is* an URL. And simple downloaders also have an explicit 'just download it and parse it with this simple rule' action, so they do not use URL Classes (or even full-fledged Page Parsers) either.

Revision #1
Created 12 March 2021 16:50:53 by CuddleBear
Updated 12 March 2021 18:24:07 by CuddleBear