Adding sites to the application

This document will walk you through the necessary steps to add a site to the application.

It first describes the properties and methods necessary to implement a site and in a seconf time presents the important steps, using the `ffn_net.FFN` class as an example. Be sure to open the file `sites/ffn_net.py` besides this document to properly grasp it.

IMPORTANT

If you are not interested about the internal workings of the app, it may end up being quite a boring read. You have been warned.

Adding a site can only be done by coding a class in Python 3. If you don't know how to code, contact me (the author of the app) through GitHub or any other way you have. I will probably accept to add it for you.

The entire application is made to make adding a site as easy as possible. According to this idea, you should not need to modify any `.py` file besides `utilities/story_writer.py` and the one or two files needed for the site itself.

Basic documentation and a working example

All the basic documentation is available in the `sites/story.py` file. A working example is available in the `sites/fanfiction_net.py` and `sites/fanfiction_net_contants.py` files. It is tailored to the fanfiction.net website.

Necessary properties

To properly represent a story, many properties are needed. Following are (quite) long descriptions for each of them. *A shorter description is available for each of them in the* `story.Story` *class.*

Story.author [*str*]: the name of the author and nothing else. Will be used in the statistics and in the `Story.get_author()` method which is called when writing the informations file about the

story.

Story.chapters [*list*]: the titles of the different chapters. Some stories do not have any (like one-shots on fanfiction.net): in this case having it empty is recommended.

Story.chapter_count [*int*]: the number of chapters in the story. Must obviously be a strictly positive integer. Not always equal to the legnth of the `chapters` property.

Story.curated_tokens [*str*]: the tokens to use when building the statistics. Can be equal to `Story.tokens` but may also be shorter, like the `FFN` class for fanfiction.net does.

Story.language [*str*]: the main language in which the story is written.

Story.relative_path [str]: the name of the folder containing the story directory. It can be the same as `Story.site`. Will be transformed to *lower case* when used so be aware that 'A' and 'a' cannot be two different relative paths.

Story.site [*str*]: the site from which hails the story. It corresponds to the key in the `constants.SITES` dictionnary, which is the name of the site (something like fanfiction.net, AO3, ...).

Story.status [*str*]: Either "Complete" or "In Progress". Somes sites only host completed stories (often the ones which allow no updates): hardcoding `self.status = 'Complete'` is completely okay in this case. Note that having anything else than the two accepted values will set the status to "In Progress".

Story.story_dir [*str*]: the name of the folder containing the story itself. It **MUST** be unique to the story for the entire site. Will be transformed to *lower case* when used so be aware that 'B' and 'b' cannot be two different story directories.

Story.summary [*str*]: the summary of the story. If none is available, an empty string.

Story.title [*str*]: the title of the story. Should be a *.title()* string but it's not required and no formatting will be done externally (contrary to `Story.relative_path` for example).

Story.tokens [*str*]: miscellaneous informations about the story, like characters, rating, update/publication date. No specific formatting is required for them except to be consistent with the other stories from the same site.

Story.universe [*str*]: the universe of the story (Harry Potter, Star Wars, ...)

Story.url [*str*]: the url to the base web page for the story (which may not be the one entered by the user at first: the class has to be able to handle this case). It's not always the first chapter. For example, the registered url for a story from this site would be something like this URL, which is not the first chapter at all but allows access to the index and from there every chapter.

Story.word_count [*int*]: the number of words in the story. A strictly positive integer.

NOTES:

1. How you initialize all these variables will vary depending on the site but heavy usage of regex and constants works (see the `ffn_net.FFN` class)
2. Outside of `url`, none of the values are initialized to force you to do it when subclassing. Be sure to do so to avoid annoying errors.
3. Other informations are needed to describe a story but they are handled directly by the `data_handler.DataHandler` class since they cannot be found in a page more often that not and must be decided by the user. As such, they are not described here.

Necessary methods

There are a few methods that needs to be implemented to ensure the application do not crash spectacularly the second you test it for your site.

Story.get_informations_title*(self) -> str*: an informations file has a name that respect the following pattern: `{informations_title}_informations.html`. This method should return the `{informations_title}` part (will be *lower case* when used)

Story.get_author*(self) -> str*: return either the author's name or the author's name embedded in a url to their page.

Story.get_universe*(self) -> str*: return either the universe or the universe embedded in a url to their page.

Story.get_chapter*(self, chapter_num: int) -> str*: get the page corresponding to the asked for chapter and strip it of everything that is not the chapter itself then returns the text of the chapter. Be aware some sites have different HTML depending on the number of chapters in the story (single vs many) and you should handle that inside this method. The `chapter_num` parameter is the number that indicates which chapter to download. To see how the text will be used, see `constants.CHAPTER_TEMPLATE`, especially the `{chapter_text}` part.

Important steps

1 - The URL

If you take a look at the `ffn_net.FFN` class you will see it starts by declaring a private property, `__num_id`, and immediatly uses it to initialize the `Story` class from from which `FFN` inherits, which set up the `url` property with the base url for fanfiction.net. This is done because the url given by the user may not be the shortest one and may not be linking to the first chapter of the story (which, for fanfiction.net, is the first url to give access to a maximum of informations about the story) .

Another example can be found in the `uhp_ffn.UHP` class, where the base url is not the first chapter but the page containing the summary and tokens of the story. Again, parsing this page gives access to everything else.

Doing something similar for your site is strongly recommended.

2 - Other properties

All the other properties are still unset. To set them download the pages which contains them (for fanfiction.net only the first chapter is needed but for Ultimate Harry Potter Fanfiction the first chapter do not contain enough informations about the story to be truly useful) and then set them up, all the while liberally using regex to parse the HTML (which is usually not recommended but here you are supposed to know the internal look of the site's webpages precisely and as such it is not as big of a problem).

Creating private methods and properties is perfectly normal and may even be necessary.

For an example of this, see the `FFN.get_author` method, which makes use of a private property to build a link the author's page on fanfiction.net and also uses many private methods to get all the informations.

3 - Constants

You may need several constants for your site, and that is perfectly normal. If their number is becoming quite large as you progress, do not hesitate to put them in a second file.

If you do that, and assuming the file for the new site is called `new_site.py`, name the constants file `my_site_constants.py`. It will greatly help maintain clarity for everyone else.

4 - Final step

You have just finished coding the class for your site and want to add it to app proper ? Here is how to do it.

- Open the `utilities/story_writer.py` file.
- Import the file containing your site's class under the ones already here
- Add your site, its class and identifier under the ones already here, following the existing examples

6 - Testing, testing

Once all this steps are done, you should have a finished adding the site you wanted. Do not forget to try it with several stories of different types (for testing when changes have been made, read `checklist.pdf`).

Since you added a site yourself, you get to pick a list of stories coming from the newly-added site, respecting the following requirements:

- Two differents authors,
- A one chapter story,
- A story with three or more chapters,
- Two stories belonging to a series

Once done, add that list to `test_stories.txt`, following the pattern already in use.