

Les applications et serveurs web pour git : l'exemple Github

Thibault Clérice

Octobre X, 2017

1. Github, gitlab, etc. : pourquoi ?

- ▶ Serveur distant
- ▶ Gestion d'équipe et de projets
 - ▶ Comptes utilisateurs
 - ▶ Comptes organisations
 - ▶ Equipes aux niveaux repository et organisations
- ▶ Gestion de bug et de tickets (Issues)
- ▶ Gestion de fusion de branche et de "review"

2. Github : les issues

- ▶ un bug : on donne alors un titre au bug
 - ▶ si c'est dans le cadre d'un logiciel/d'une application/d'un site, on donne la démarche pour le reproduire
 - ▶ on pense à donner les fichiers concernés par le bug
 - ▶ on essaye de décrire au mieux le problème. Même si c'est sur notre propre dépôt
 - ▶ Sinon, on oublie le bug, on revient deux mois plus tard, on sait plus ce que c'est.

3. Github : les issues

- ▶ une nouveauté : on veut une nouvelle fonctionnalité, un nouvel ensemble de données
 - ▶ on décrit correctement ce que l'on veut, de manière à pouvoir transférer la charge ou s'en occuper soi même plus tard
- ▶ une discussion : on essaye de comprendre un morceau de code, etc.

4. Github : les issues

- ▶ Les issues peuvent être **attribuées** à des membres du dépôt : si je travaille avec @vincentjolivet, et qu'un bug le concerne, je peux lui attribuer en tant que membre. Sinon, en voyant un bug qui me concerne, je peux me l'attribuer.
- ▶ Les issues peuvent avoir des **labels**. Une série de label par défaut existe, mais on peut tout à fait en ajouter à ses propres dépôts.
- ▶ Les issues peuvent faire l'objet de regroupement dans des **projects** (sans deadline), des **milestones** (objectifs de sortie d'une nouvelle version).

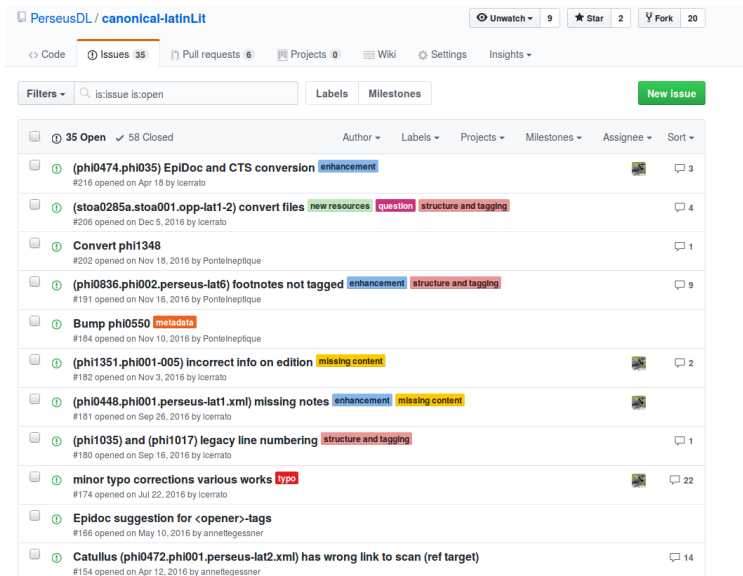
5. Github : les issues d'un repo logiciel

The screenshot displays the Github interface for the repository `EHRI / ehri-frontend`. At the top, navigation links include `<> Code`, `Issues 123`, `Pull requests 2`, `Projects 0`, `Wiki`, and `Insights`. On the right, there are buttons for `Watch 6`, `Unstar 8`, and `Fork 4`. Below the navigation bar, a search filter is set to `is:issue is:open`, and there are buttons for `Labels`, `Milestones`, and a green `New issue` button. The main content area shows a list of 123 open issues, with a summary of 441 closed issues. The issues are sorted by default, and the list includes the following details:

Issue ID	Title	Labels	Opened	By
#951	Display aggregated events in the admin...	admin, enhancement	4 days ago	mikesname
#940	Devise a better system for long-running tasks...	admin, bug, enhancement	20 days ago	mikesname
#911	Solr Search w/ parenthesis?	bug, search	May 22	mikesname
#904	Write some integration tests that leveraged dockerized Solr config...	enhancement	Apr 24	mikesname
#896	When adding user to admin, verify their account as well...	admin, enhancement	Mar 17	mikesname
#894	Add a page for public canned queries...	enhancement, feature, portal	Mar 16	mikesname
#893	Highlighting only works for one search term if non-adjacent...	bug, search	Mar 16	mikesname
#888	Long-awaited prod/stage sync script...	admin, bug, question, tech-debt	Mar 13	mikesname
#884	Issue with notes list metadata...	bug, portal	Feb 16	mikesname
#879	Vocabulary syncing	admin, bug, enhancement, high priority	Feb 1	mikesname

Figure 1: Issues de `github.com/EHRI/ehri-frontend`

6. Github : les issues d'un repo data



The screenshot displays the GitHub interface for the repository **PerseusDL / canonical-LatinLit**. At the top, navigation links include Code, Issues (35), Pull requests (6), Projects (0), Wiki, Settings, and Insights. The repository has 9 Unwatch, 2 Stars, and 20 Forks. Below the navigation bar, there are filters for 'is:issue is:open' and buttons for 'Labels', 'Milestones', and 'New issue'.

The issues are listed in a table with the following columns: Author, Labels, Projects, Milestones, Assignee, and Sort. The issues are sorted by 'Open' status, showing 35 Open and 58 Closed issues.

Issue Title	Labels	Comments
(phi0474.phi035) EpiDoc and CTS conversion	enhancement	3
(stoa0285a.stoa001.opp-lat1-2) convert files	new resources, question, structure and tagging	4
Convert phi1348		1
(phi0836.phi002.perseus-lat6) footnotes not tagged	enhancement, structure and tagging	9
Bump phi0550	metadata	
(phi1351.phi001-005) incorrect info on edition	missing content	2
(phi0448.phi001.perseus-lat1.xml) missing notes	enhancement, missing content	
(phi1035) and (phi1017) legacy line numbering	structure and tagging	1
minor typo corrections various works	typo	22
Epidoc suggestion for <opener>-tags		
Catullus (phi0472.phi001.perseus-lat2.xml) has wrong link to scan (ref target)		14

Figure 2: Issues de github.com/PerseusDL/canonical-latinLit

7. Github : un exemple de bonne issue

Language subtags #1548

 **Open** annettegessner opened this issue on Jun 29 · 3 comments

Edit

New Issue



annettegessner commented on Jun 29

Owner



I spotted some incoherence in our use of language tags. Do we have our own list of abbreviations that we use? If yes, please direct me to it! If not:

[Epidoc](#) states, that the [IANA Language Subtag Registry](#) should be used and I would follow this advise.

Here are some examples of language subtags we need on a regular basis according to this recommended list:

en --> English

de --> German

(el --> Modern Greek)

grc --> Ancient Greek

la --> Latin

mul --> multiple languages

Some of the deviations I found in First1kGreek:

"eng" instead of "en" for English

"lat" instead of "la" for Latin

"ger" or "deu" instead of "de" for German

"el" which is Modern Greek, when it should be Ancient Greek "grc"

There are two different instances, where we used those subtags:

1. language id="" --> to define the languages used in the body of the XML (i.e. the languages the text itself is written in)
2. xml:lang="" --> to define language used in (a section in) the header

Please let me know, if you agree with these abbreviations, so we can start fixing these!

Assignees



No one—assign yourself

Labels



general problem

Projects



None yet

Milestone



No milestone

Notifications

 Subscribe

You're not receiving notifications from this thread.

2 participants



 Lock conversation

Figure 3:

<https://github.com/OpenGreekAndLatin/First1KGreek/issues/1548>

8. Github : un exemple de mauvaise issue

Fix duplicate reference issues #86



Pontelneptique opened this issue on Dec 10, 2015 · 0 comments

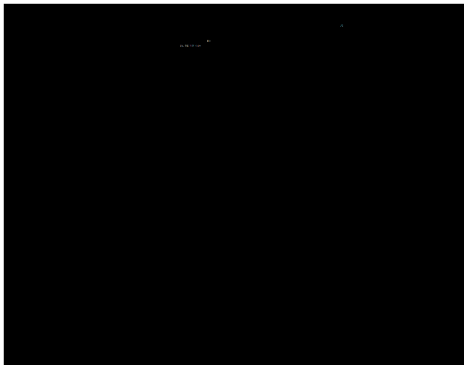
Edit

New Issue



Pontelneptique commented on Dec 10, 2015

Member



Assignees



No one—assign yourself

Labels



None yet

Projects



None yet

Milestone



No milestone

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

1 participant



Lock conversation

Figure 4: <https://github.com/PerseusDL/canonical-latinLit/issues/86>

9. Les dépôts dérivés (forks)

- ▶ Les données de <https://github.com/chartes/elec> vous intéresse. Mais vous aimeriez changer une partie des données : ne conserver que les testaments de poilu et supprimer les éléments qui ne correspondent pas à vos besoin.
- ▶ À ce moment là, on va réaliser un **fork** du repository sur son propre compte. On obtient ainsi les droits d'écritures sur le repository tout en gardant l'ensemble de son historique. On travail sur ce repository ensuite localement, comme on travaillerait avec n'importe quel repository.
- ▶ **Note:** Si l'on fait des modifications sur un dépôt d'une autre personne, il est plus respectueux de faire un fork que de télécharger et de créer un nouveau dépôt.

10. Collaborer : les *pull requests*

- ▶ Une *pull request* (Vocabulaire de *github*) est une demande d'autorisation de contribution à un dépôt.
- ▶ Une pull request se fait entre deux branches, comme un merge. Cependant, cela peut être deux branches de deux forks.
- ▶ Contribuer à un repository ?
 - ▶ Non seulement j'aide la personne et les utilisateurs du repository
 - ▶ Mais en plus j'apparaîs sur l'historique

11. Collaborer : les *pull requests* (Histoire exemple)

- ▶ Je remarque une erreur dans les métadonnées d'un document
 1. J'ouvre une issue qui explique le problème (Typo dans le fichier X, etc.)
 2. Parce que je veux être encore plus utile, je veux proposer la modification pour que le propriétaire n'ait pas à faire la correction lui-même
 3. Je fais un fork du dépôt du cours (
<https://github.com/pontineptique/cours-git>)
 4. Je crée une branche spécifique et je rentre dessus
 - ▶ Tips : quand on fait une branche pour un objectif particulier, on l'appelle du nom de l'issue pour se souvenir de pourquoi on a créé cette branche
 5. Je fais la modification et je la commit
 6. Je push
 7. Je fais une pull request sur le repository d'origine

12. Exercice

- ▶ Aller sur <https://github.com/Pontelneptique/canonical-latinLit>
- ▶ Prendre un dossier auteur de <https://github.com/Pontelneptique/canonical-greekLit/tree/master/data> (Identifié par `tlg****`)
- ▶ Dans ce dossier, ouvrir le fichier `cts.xml`
- ▶ Suivant les bonnes pratiques (issues, fork, pull request), proposer la traduction du nom de l'auteur
- ▶ **Bonus:** Ajouter l'identifiant Wikidata

...

```
<cpt:structured-metadata
```

```
  xmlns:cpt="http://purl.org/capitains/ns/1.0#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
```

```
<dc:identifier>https://www.wikidata.org/wiki/Q2098</dc:iden
```

```
</cpt:structured-metadata>
```

```
</ti:textgroup>
```