Création de plug-ins - GALACTIC CLI

Alexy Lafosse

2024

Laboratoire L3i

créé en 1993

dirigé par M. Yacine GHAMRI-DOUDANE

composé de 3 équipes :

- Modèles et Connaissances
- Images et Contenus
- Dynamique des systèmes et Adaptativité

Plateforme GALACTIC

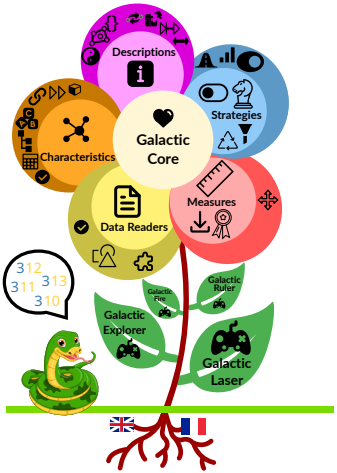

Figure 1: Structure GALACTIC

GALACTIC est l'acronyme de GAlois LAttices Concept Theory Implicational system and Closures

Outils permettant de travailler sur :

- l'Analyse Formelle des Concepts
- Théorie des Treillis

Codée en python

Plateforme GALACTIC

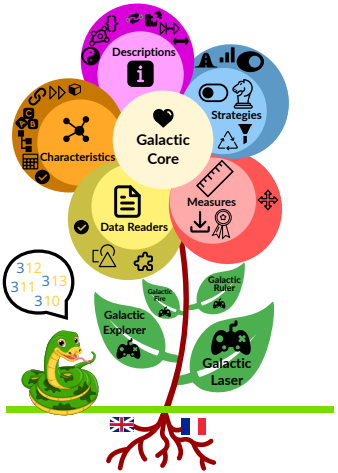

Figure 2: Structure GALACTIC

Structure :

- noyau : coeur de la fleur
- plug-ins : pétales de la fleur
- applications : feuilles de la fleur

Versions de python compatibles : 3.10 à 3.13

Langues disponibles : anglais et français

Commande *galactic*

Interface en ligne de commande

Coeur : Commande *galactic*

- ne peut rien faire seule
- est extensible

Figure 3: Commande *galactic* - Coeur

Commande *galactic*

Application GALACTIC Laser

Elle peut être une extension de cette commande

La commande n'existe pas encore mais la structure permet ce développement. Exemple : *galactic laser run*

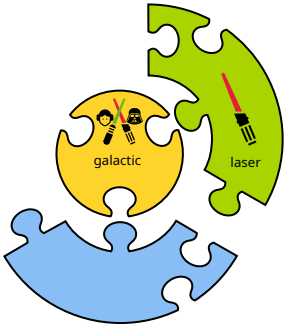

Figure 4: Commande *galactic* - Extension laser

Commande *galactic*

Framework

Il est extensible

La commande de base est héritée par les extensions

Chacune de ses extension a pour but de créer des bases de plug-ins pour GALACTIC

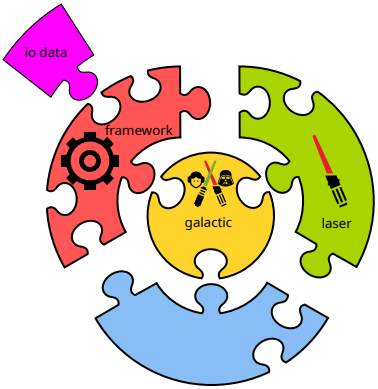

Figure 5: Commande *galactic* - Framework & extension *io data*

Commande *galactic*

Extension *io data*

Commande : *galactic framework io data create*

Elle génère des squelettes de plug-ins *Data Readers* personalisés

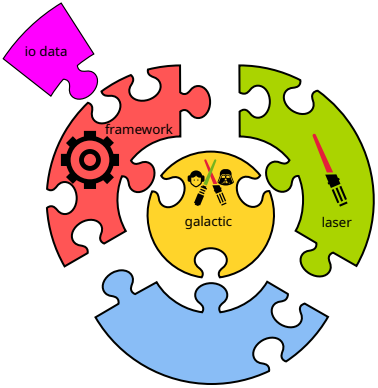

Figure 6: Commande *galactic* - Framework & extension *io data*

Commande *galactic*

Extension *meta*

Elle génère un squelette d'extension pour le framework

Le framework peut donc créer des extensions pour lui-même.

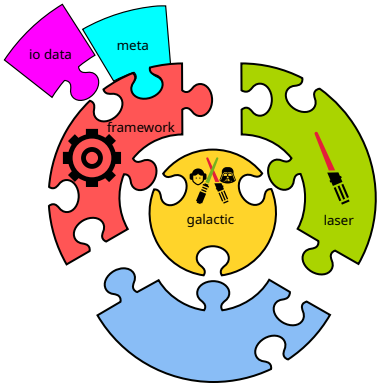

Figure 7: Commande *galactic* - Framework & extension *meta*

Objectif

Figure 8: Commande `galactic` - Coeur

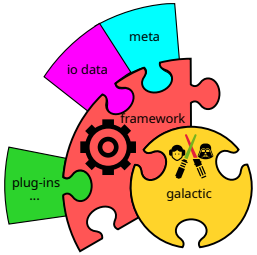

Figure 9: Commande `galactic` - Framework & extensions

Préparation

Outils Développement :

Outils Framework :

Poetry

Fichier de configuration : *pyproject.toml*

Commandes utiles :

- *poetry shell*
- *poetry install*
- *poetry build*
- *poetry run*

Deux extensions pour *poetry* :

- *poeblix*
- *poetry-dynamic-versioning*

Commande de base : *core*

Commande parente des plug-ins

Commande *galactic framework* ne crée rien mais propose d'autres commandes

Prend un argument obligatoire : chemin

Deux options :
- auteurs
- description

Si pas d'options précisées, alors des questions sont posées

```
alactic-apps-cli-framework-io-data % galactic framework

The command "framework" does not exist.

Did you mean one of these?
    framework self create
    framework io data create
```

Figure 10: Exemple - Commande *galactic framework*

Commande de base : *core*

Figure 11: Exemple - Auteurs

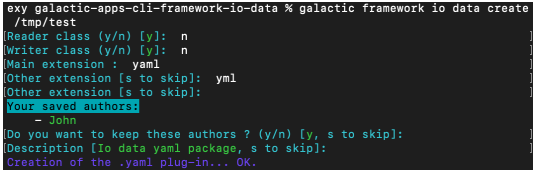

Figure 12: Exemple - Description

Extension `io-data`

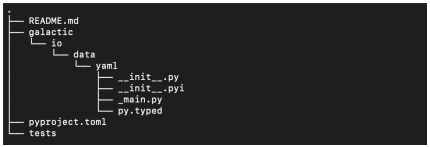

Figure 13: Exemple - Structure du plugin créé

Squelette de plug-in permettant de lire et/ou écrire dans un fichier

Plug-ins *Data Readers*

```
galactic framework io data create
```

Extension *io-data*

```
(galactic-apps-cli-framework-io-data-py3.12) lafossealexy@MacBook-Air-de-Alexy galactic-apps-cli-framework-io-data % galactic framework io data create /tmp/test --help

Description:
  Create a data-io plugin

Usage:
  framework io data create [options] [--] <path>

Arguments:
  path                          The path to create the project at.

Options:
      --author=AUTHOR           Name of an author of the package. (multiple values allowed)
      --description=DESCRIPTION  Description of the package.
  -r, --reader                  Create a data reader class.
  -w, --writer                  Create a data writer class.
  -e, --extension=EXTENSION     An extension allowed in the plug-in. (multiple values allowed)
      --readerClass=READERCLASS  Set the name of the reader class.
      --writerClass=WRITERCLASS  Set the name of the writer class.
  -h, --help                    Display help for the given command. When no command is given display help for the list command.
  -q, --quiet                   Do not output any message.
  -V, --version                 Display this application version.
      --ansi                    Force ANSI output.
      --no-ansi                 Disable ANSI output.
  -n, --no-interaction          Do not ask any interactive question.
  -v|vv|vvv, --verbose          Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug.
```

Figure 14: Exemple - Commande *io data* - Liste options

Extension *io-data*

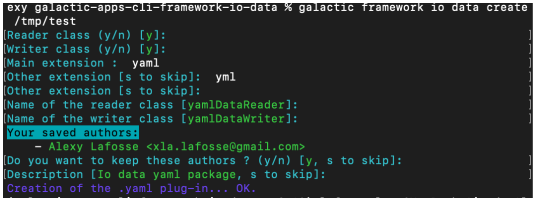

Figure 15: Exemple - Commande *io data* sans options

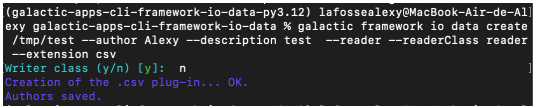

Figure 16: Exemple - Commande *io data* avec options

Autres Plug-ins & Améliorations

Autres plug-ins :

- Plug-in `meta`
- Générateur de plug-ins `characteristics`

Améliorations :

En fonction de l'utilisation

Tests

Vérifications à faire :

- code propre
- tests

Commandes utilisées :

- `poetry run tox`
- `poetry run tox -e style`
- `poetry run tox -e linter`
- `poetry run tox -e coverage`

Utilisation

```
pip install \
     --index-url https://gitlab.univ-lr.fr/api/v4/groups/galactic/-/packag
     galactic-apps-cli-framework-io-data
```

Pas nécessaire d'installer la commande de base

Principale difficulté

Compréhension de la structure

- Nombreuses questions et réflexions

Zones d'ombres et erreurs de compréhension

- Vision en constante évolution

Conclusion

Extensions du framework de la commande *galactic*

Extension *io data* fonctionnelle

Créer l'extension *meta* et d'autres

- Faciliter le développement de nouveaux plug-ins

Malgré les difficultés, j'ai beaucoup appris

Conclusion

Merci de votre attention