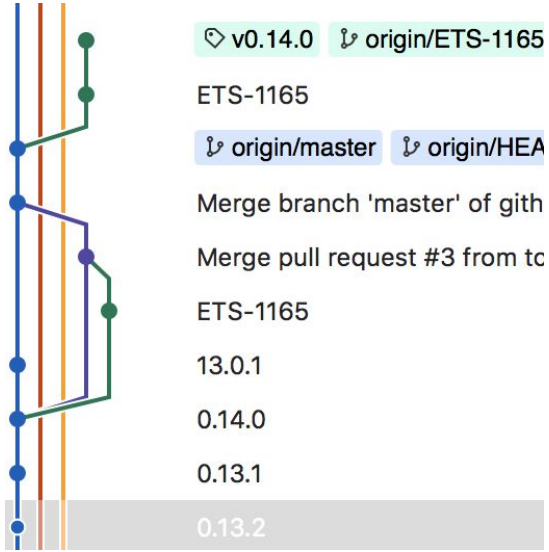
GitHub

Gestionnaire de version

Permet de créer un repository en ligne

Permet de partager du code

Intègre un système de branches et de pull request pour les contributions externes



Git

Interface entre la console et un gestionnaire de version en ligne

Permet aussi de gérer un dépôt local à travers des commandes

Git

- `git clone` `// clone un dépôt distant`
- `git add` `// indique à git de suivre le fichier`
- `git commit` `// crée un commit avec un commentaire`
- `git push` `// met en ligne les commits réalisés`
- `git pull` `// tire les modifications en ligne`
- `git status` `// information sur la copie locale du dépôt`

Introspector.getBeanInfo

Cette méthode permet de dévoiler l'intérieur d'une classe. Si celle-ci respecte la convention javaBean il est possible d'en extraire les getters/setters.

Introspector.getBeanInfo(Class<?>).getPropertyDescriptors() renvoie la liste des attributs de la classe

ClassValue<T>

Une classe abstraite qui permet de stocker des couples Class/T dans une Map.

Cette classe peut être utilisée comme cache puisque sa méthode *get(Class<?>)* renvoie une valeur calculée au préalable par la méthode *computeValue(Class<?>)* si celle-ci n'était pas déjà stockée dans le cache.

@Annotation

Les annotations sont des catégories spécifiques d'interface. Elles peuvent être utilisées à la compilation ou à l'exécution.

On peut préciser l'utilisation dans certains contextes avec *@Target* et récupérer la valeurs de ses attributs à Runtime avec l'annotation *@Retention(RUNTIME)*.

{“Json”:true}

Json est un format qui représente des données sous forme de texte formaté avec “clé” : “valeur”.

Il est utilisé pour représenter des objets, des listes ou des types primitifs.

- Objet -> {“key”: “value”}
- Liste -> [,]
- type primitifs : true/false, 123, “toto”, 3.141592, null

Un objet peut contenir un objet, de façon récursive.