

Git pour Eclipse

Préliminaires

Avant toute chose, il est bon de se pencher sur le fonctionnement de GIT en soi.

Deux petits tutoriaux rapides :

<http://rogerdudler.github.io/git-guide/index.fr.html>

<https://www.hostinger.fr/tutoriels/tuto-git/#gref>

Eclipse

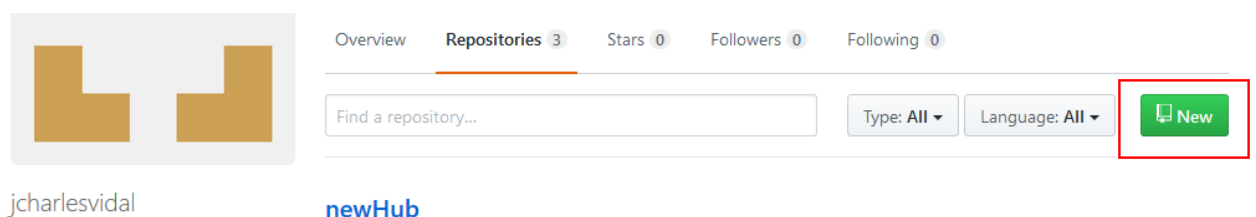
Le problème de GIT dans Eclipse, c'est qu'il est intégré d'une façon pas vraiment limpide, même si on connaît les bases de GIT.

En particulier, l'installation/configuration est tout sauf simple. On va donc essayer d'y voir plus clair.

Github

La première étape est de créer un compte Github.

Ce faisant, vous aurez un login et un mot de passe pour y accéder. Notez les bien, parce que vous en aurez besoin aussi dans Eclipse.



La première étape est de créer un repository, c.a.d un endroit où entreposer un ou des projets java.


Pour simplifier on va faire en sorte qu'un repository soit égal à un projet Eclipse. Mais je répète qu'on peut y mettre plusieurs projets java/Eclipse.

Créons donc un repository en appuyant sur *new*.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 jcharlesvidal ▾

Repository name

MonGitHub ✓

Great repository names are short and memorable. Need inspiration? How about **miniature-succotash**.

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾




Create repository


- On met le nom du repository (ce que vous voulez)
- On sélectionne *public*
- On coche *initialize with a README*
- On crée le repository


On arrive sur le repository nouvellement créé.

Branch: master ▾ New pull request

Create new file Upload files Find file Clone or download ▾

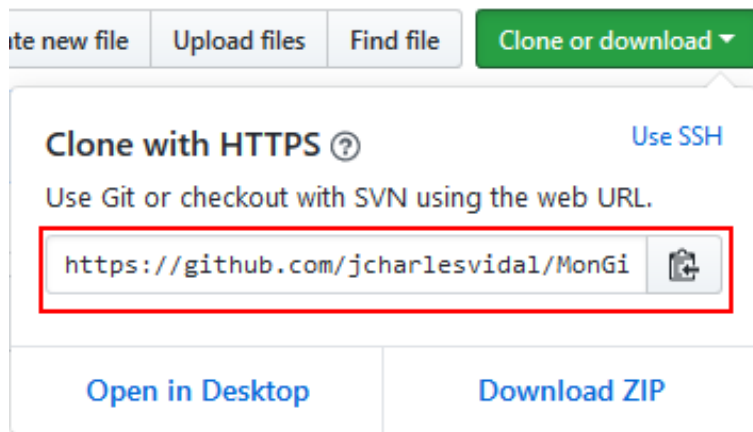
 jcharlesvidal Initial commit Latest commit 7dad930 a minute ago

 README.md Initial commit a minute ago

 README.md

MonGitHub

Et on click sur *clone or download*.

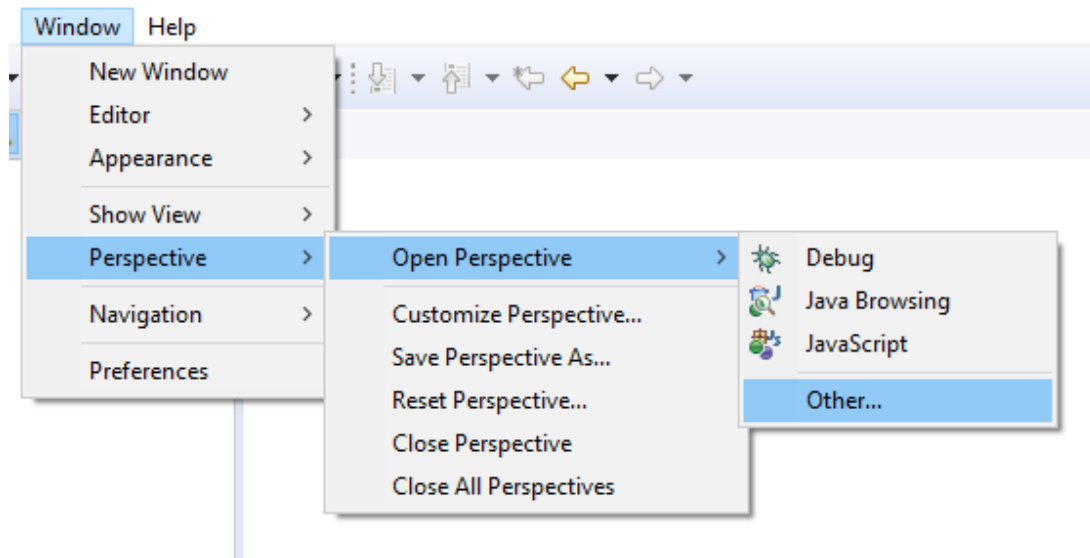


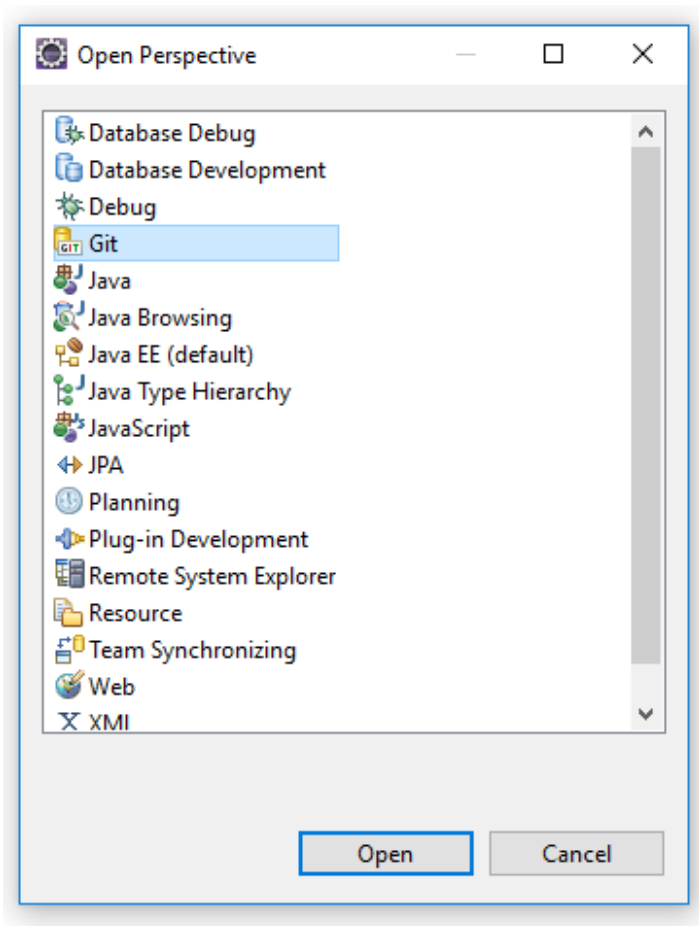
On récupère l'URL du repository et on la stocke quelque part

Eclipse

La première étape consiste à créer un repository local, si ce n'est pas déjà fait.

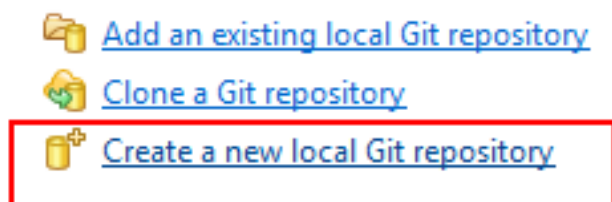
Il faut donc passer dans la perspective GIT de Eclipse.





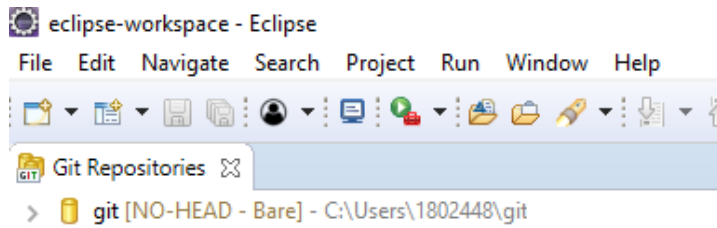
On arrive dans la perspective GIT. S'il n'y a pas de repository local on a quelque chose comme ça et on clique sur *create a new local Git repository*

Select one of the following to add a repository to this view:



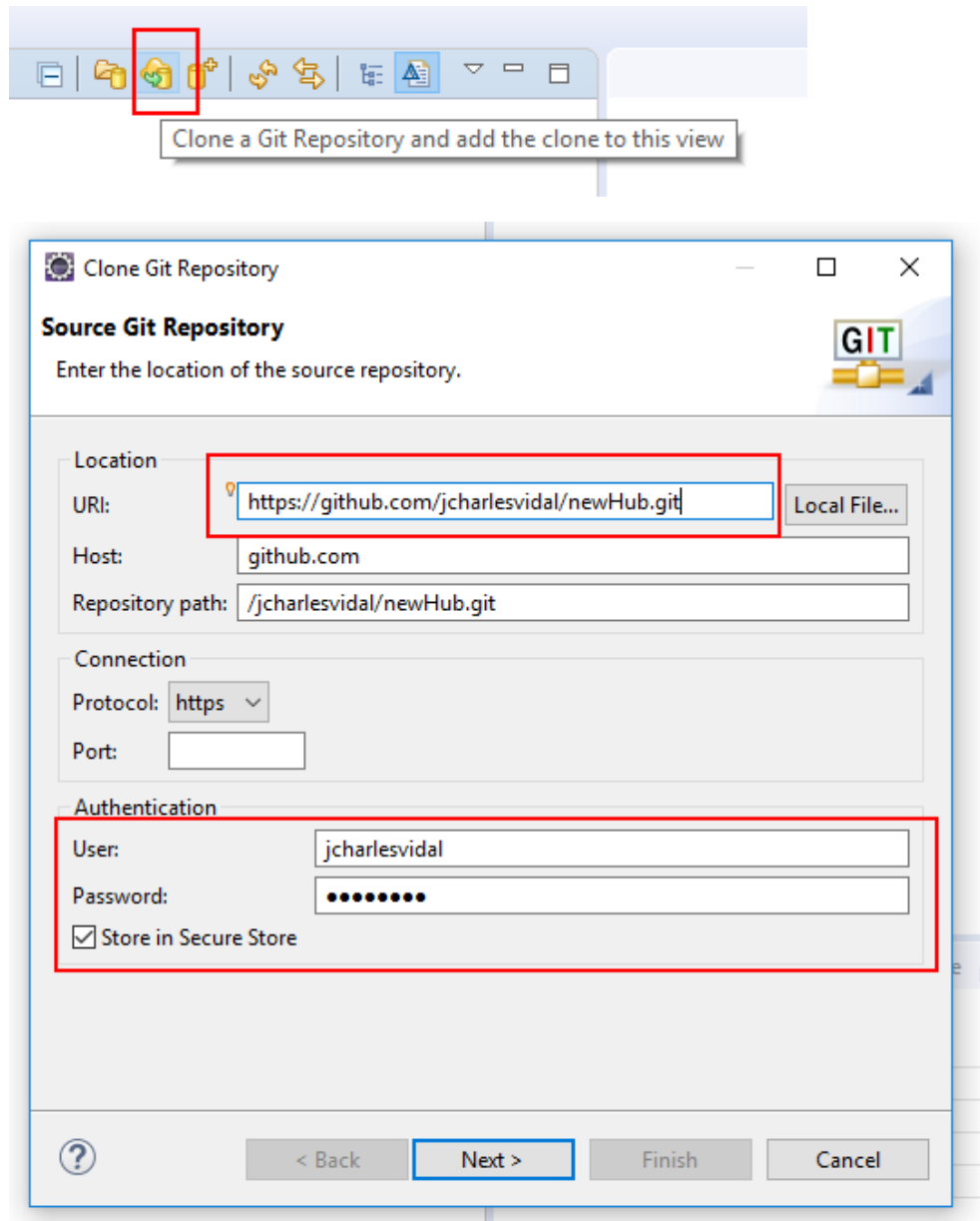
Eclipse va vous donner un emplacement par défaut. N'y touchez pas et sélectionner *create as bare repository*.

On obtient quelque chose comme ça :



A ce stade, il ne reste plus qu'à cloner le repository Github.

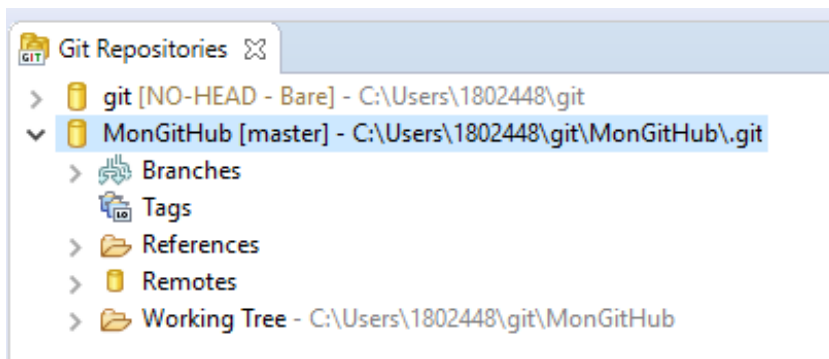
Récupérez donc l'url de tout à l'heure et cliquez sur le bouton ad hoc



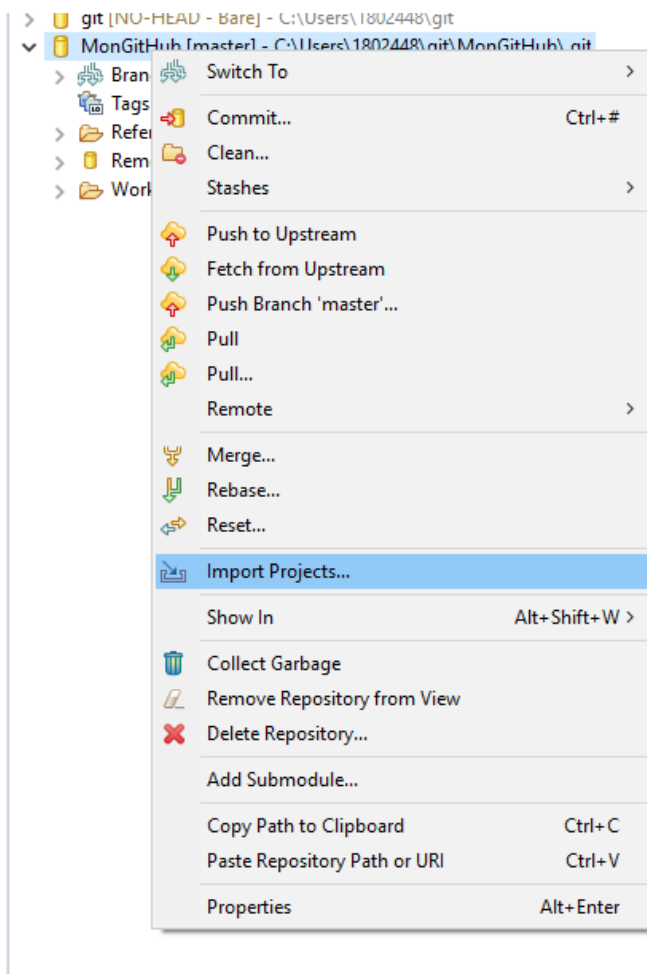
Remplissez la ligne du haut avec l'url (en copier/coller) ; le reste se remplira automatiquement.

Remplissez la zone login/mot de passe en cochant *Store in Secure Store* (sans quoi le login/pwd sera demandé en permanence).

On obtient quelque chose comme ça :



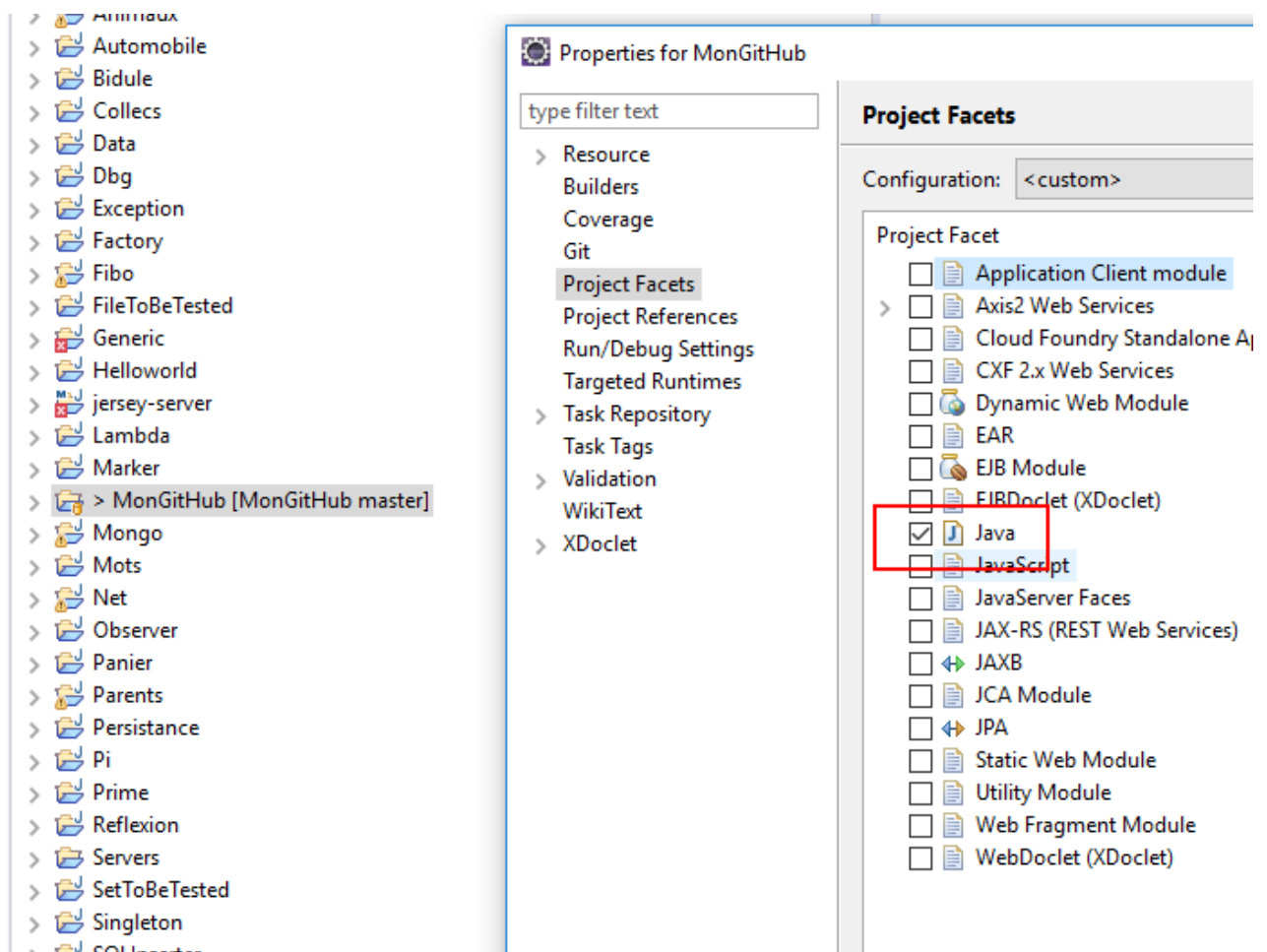
Il ne reste plus qu'à créer le projet Eclipse à partir de cela.



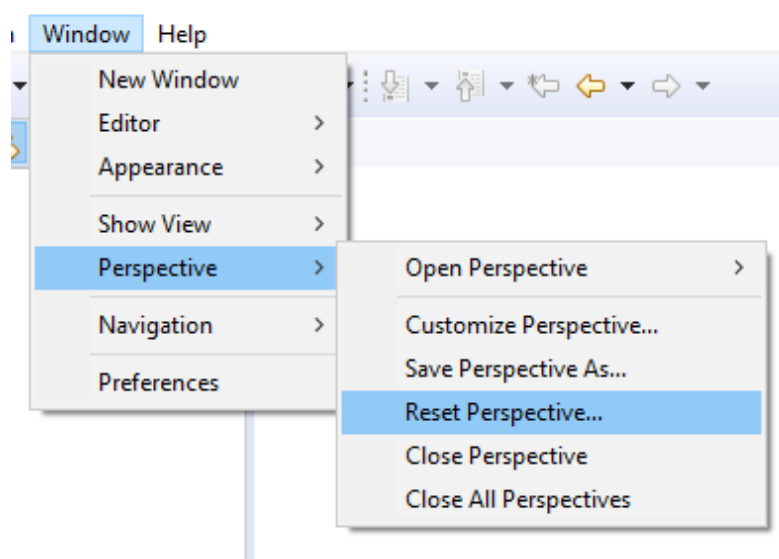
Le projet va apparaître dans Eclipse. On repasse donc en perspective java.

Petit souci, Eclipse ne sait pas que le projet est de type java.

Donc click droit sur le projet, puis *properties*, puis *facets* et on choisit java



Du fait d'un petit bug Eclipse il faut réinitialiser la perspective java.



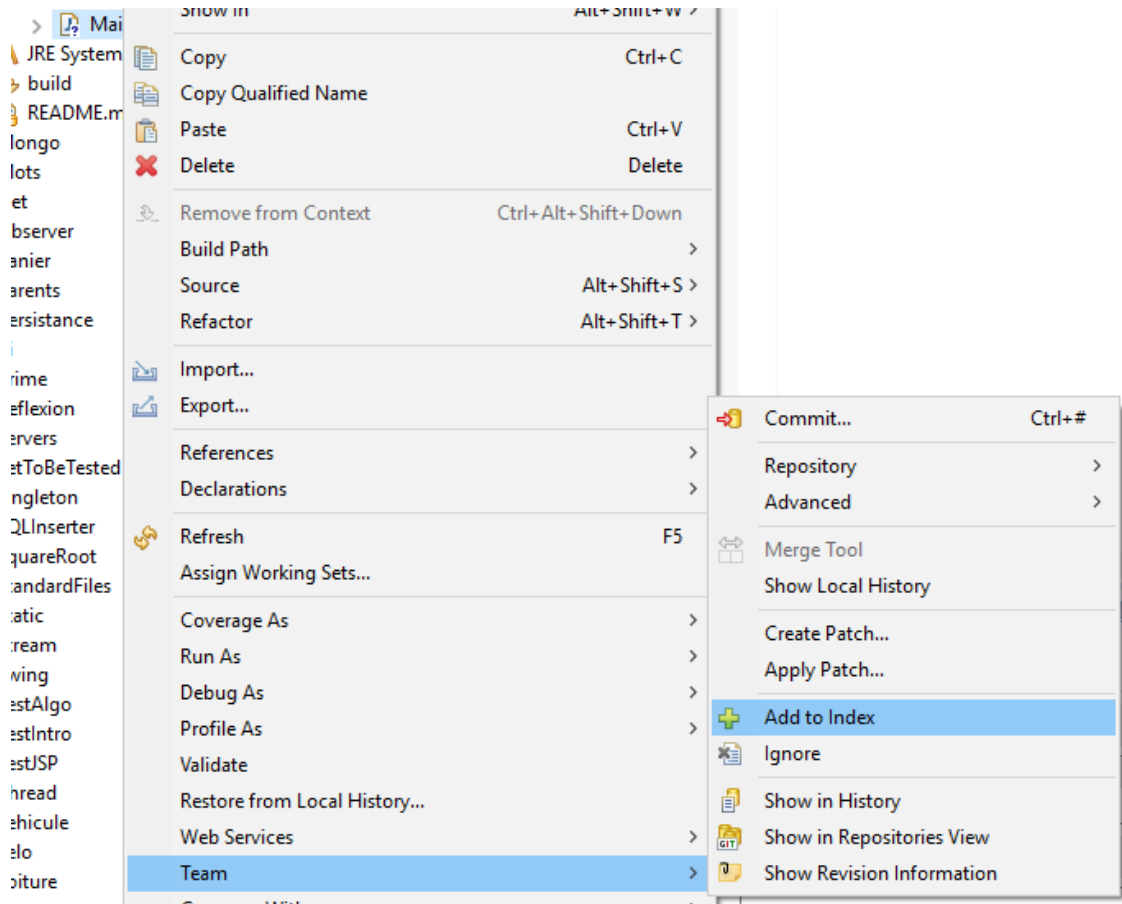
C'est bon on peut commencer.

A ce stade on peut donc utiliser les commandes de base de GIT :

- Commit
- Push
- Pull

Créons donc un package et dans celui-ci une classe.

Petit problème, on ne peut pas commiter tout de suite dans le repository local. La classe apparait avec un ?. le ? signifie qu'il faut d'abord ajouter le fichier dans le projet GIT.



Ok.

Donc on fait un commit en mettant un commentaire.

Puis on fait un push pour l'envoyer dans Github.

[Retour à Github](#)

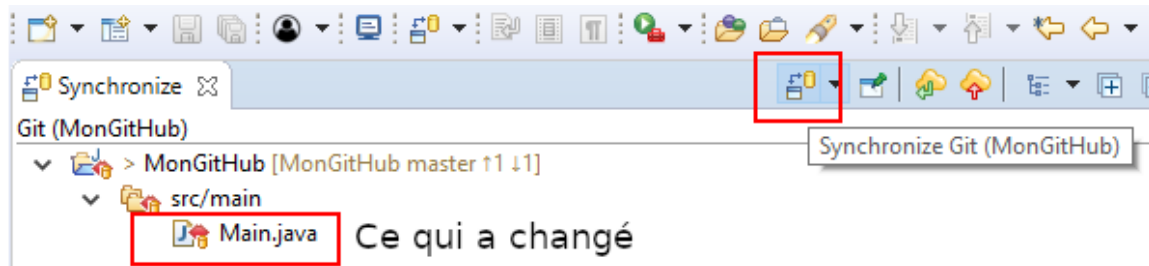
Si on retourne dans Github, on voit ce qui a été ajouté via Eclipse.

On peut par exemple modifier la classe (dans Github).

Retour à Eclipse

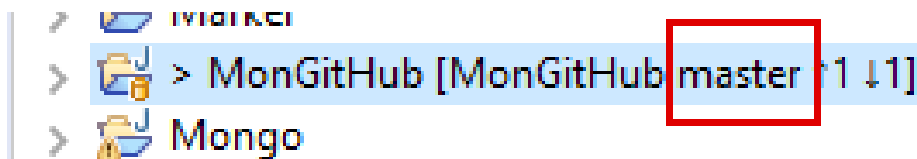
On revient dans Eclipse et on fait un pull. Et on peut voir les modifications qui ont été faites dans Github. C'est magique.

A noter qu'on peut voir les modifs distantes (sur github) via la view *Synchronise/History* et donc avant de faire le pull.

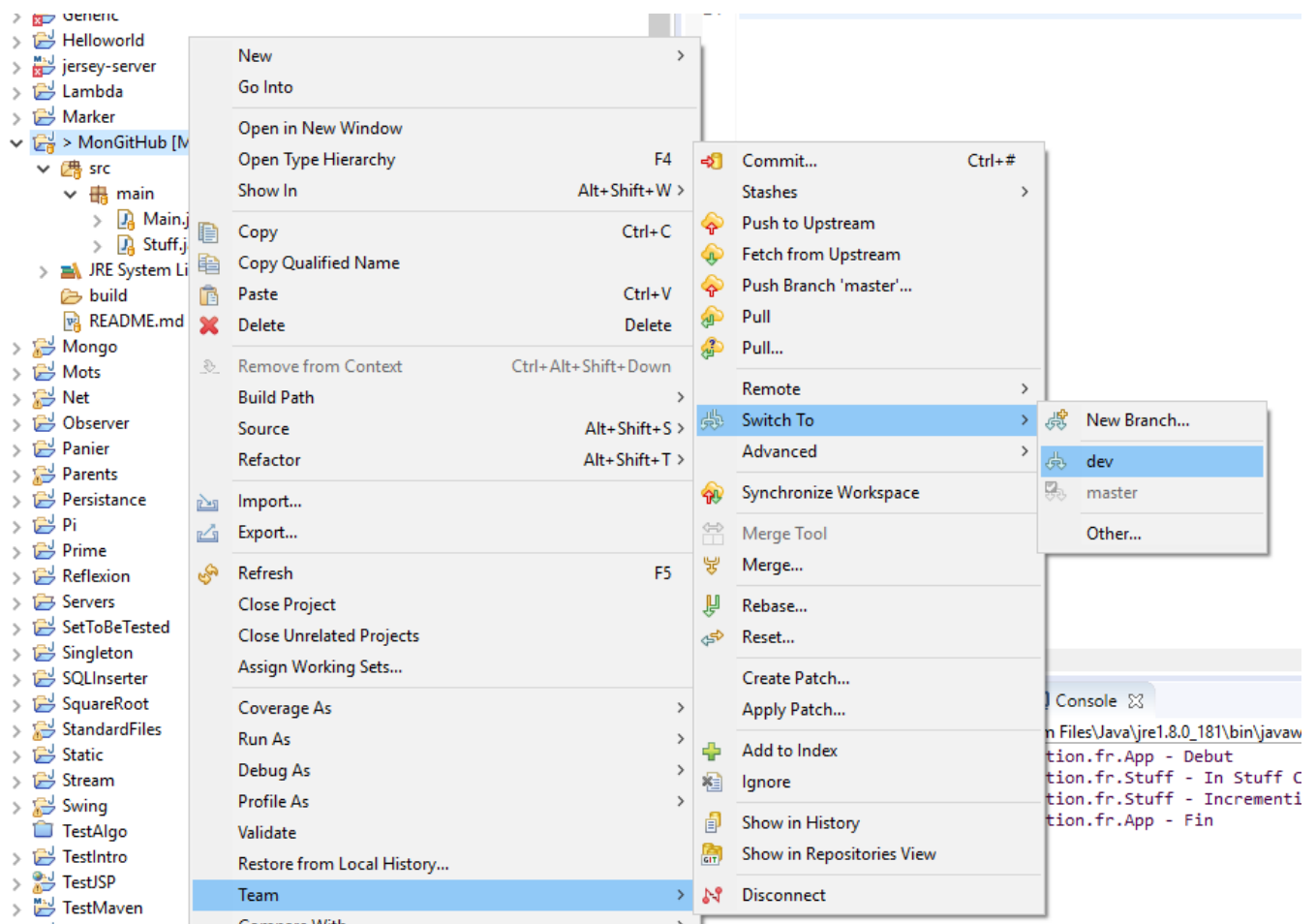


Branches

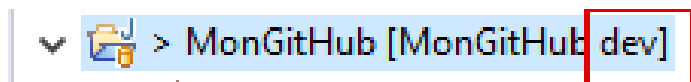
On voit dans Eclipse quelle est la branch courante



Pour changer/créer une branche, on fait *team/switch to*



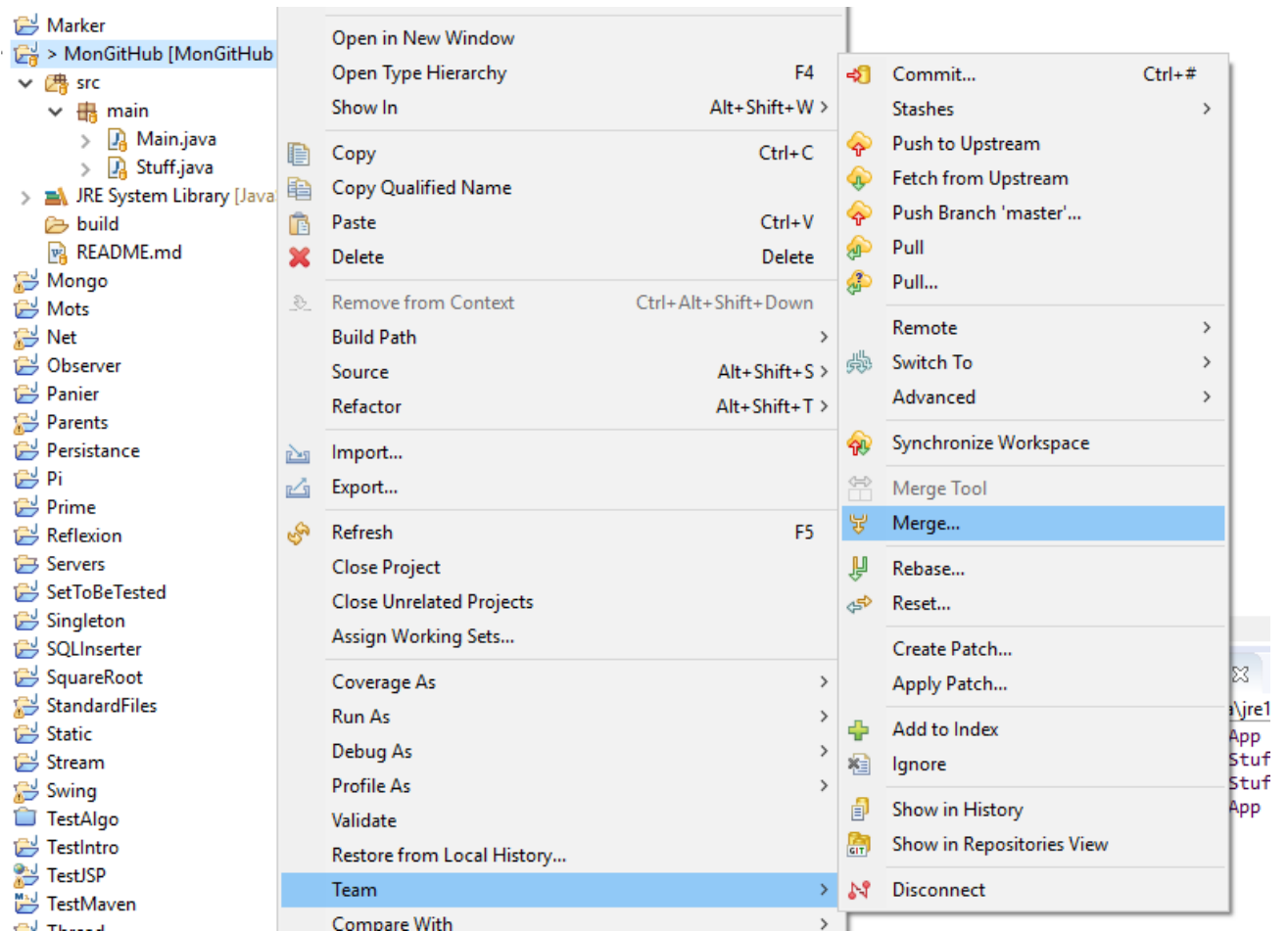
Et on voit de nouveau la branche courante



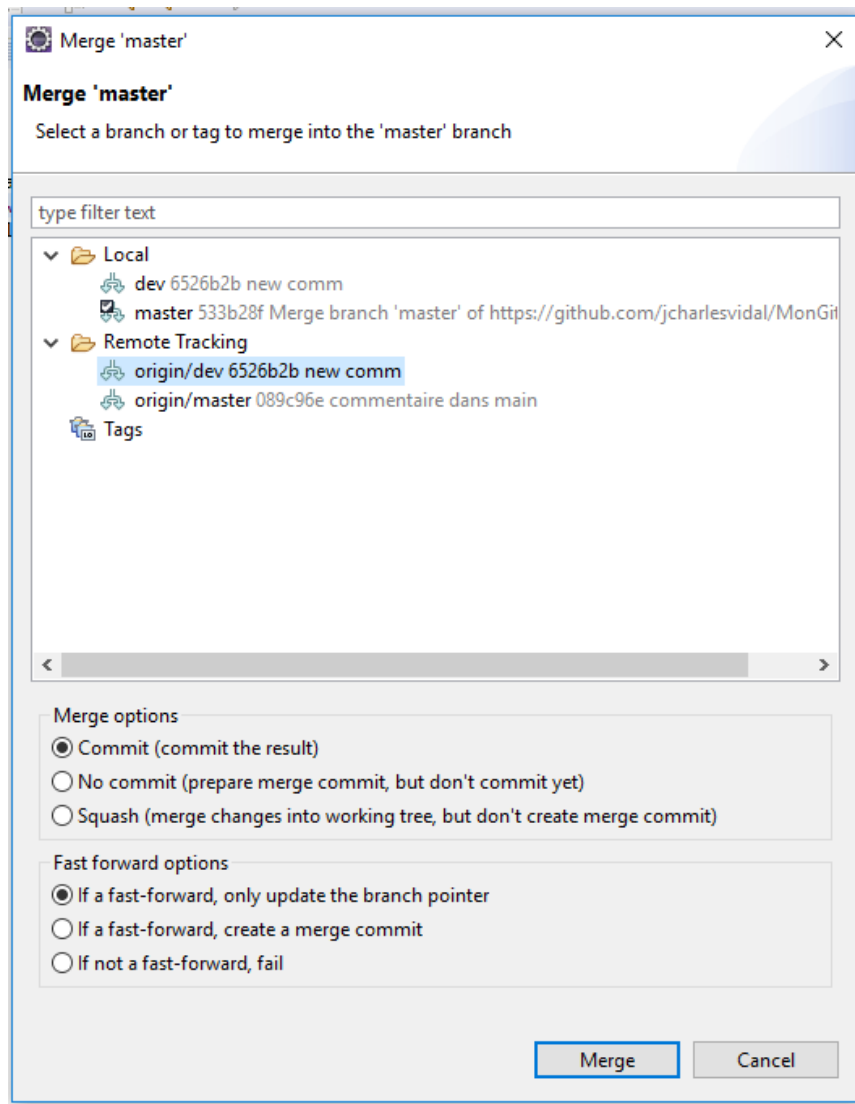
ATTENTION : avant de changer de branch (ou d'en créer une), il **FAUT** faire un commit de ce qui est actuellement modifié sans quoi les résultats sont imprévisibles.

Si on travaille sur une branche (ici, dev par ex), les ajouts/modifs devront donc être commités avant de revenir sur master.

Une fois revenus sur master, il est possible d'intégrer les modifs de dev via la commande merge.



Et on sélectionne la branche dev pour merger dans master.



Exercice inverse

Cette fois on crée un repository dans Github, puis on le clone dans Eclipse.

Puis on partage un projet java dans ce repository et on le push dans Github. On vérifie que tout est bien là (dans Github).

On peut ajouter un autre projet dans ce repository histoire de voir qu'un repository peut contenir plusieurs projets.