TP 1 – Prise en main Git et Github

Les TP s'effectuent en environnement Windows.
Créer un dossier de travail (« DCLL-TP1-Git » par exemple) sur votre espace disque réseau (lecteur « z »).
Toutes les opérations en ligne de commande doivent être effectuée en utilisant le programme Git-Bash qui permet de lancer un terminal adapté.

Exercice 1 – Création d'un compte Github

1. Rendez vous sur le site de Github : https://github.com et créez un compte « gratuit ».
2. Suivez les instructions indiquées en **annexe** pour mettre à jour vos clés SSH sur votre environnement de développement et sur votre compte Github.
3. Rendez vous sur le projet Github https://github.com/FranckSilvestre/ExemplesMoodleXML et effectuez un fork du projet.
4. Dans votre dossier de travail, en utilisant Git-Bash, clonez **votre** projet ExemplesMoodleXML résultant du fork exécuté précédemment. Le clone doit être effectué en utilisant le protocole SSH (préfixe de l'URL du projet: git@...). Vérifiez à l'aide de quelques commandes Git (git status, git branch -a, git remote -v) que votre projet est bien disponible localement.

Exercice 2 – Travail en solo

1. Configurer votre identité Git en utilisant les commandes suivantes :
   git config --global user.name "mon_user_name"
   git config --global user.email "mon_adresse"
   où « mon_user_name » est à remplacer par votre user name Github et « mon_adresse » est à remplacer par votre adresse mail utilisée pour la création de votre compte Github.
2. Le projet contient deux fichiers XML. Ces fichiers sont mal formés
3. Editez les fichiers disponibles dans le projet. Corrigez les fichiers de manière à ce que les documents XML contenus dans ces fichiers soient bien formés. Sauvegardez les fichiers corrigés.
4. Reportez les modifications dans votre repository local.
   Rappel commandes : « git add . » suivi de « git commit -m“commentaire...“ »
5. Créez un nouveau fichier XML intitulé « quiz.xml » qui aggrège dans un élément racine « quiz » les deux documents XML corrigés précédemment. Sauvegardez le fichier.
6. Reportez les modifications dans votre repository local.
7. Créer une branche intitulée « dev/exempleTrueFalse » et basculez sur la branche en question. Dans cette branche créez un nouveau fichier « TrueFalseQuestion.xml » et ajouter un exemple de question de type true/false en vous appuyant sur la documentation Moodle (http://docs.moodle.org/22/en/Moodle_XML_format#True.2Ffalse). Sauvegardez votre fichier et ajoutez le dans le repository dans votre branche

« dev/exempleTrueFalse »
8. « Envoyez » votre branche « dev/exempleTrueFalse » sur votre repository Github. Rendez vous sur le site Github pour vérifier que vos modifications sont bien prises en compte sur le repositoty distant.
9. Rebasculez sur la branche master et effectuez un merge de la branche « dev/exempleTrueFalse ».
10. « Envoyez » les modifications sur la branche master sur votre repository Github.

Exercice 3 – Travail en duo

1. Choisissez un binôme. Un des deux binômes doit être désigné comme titulaire du projet partagé sur Github. Le titulaire du projet partagé doit, sur Github, ajouter son binôme à la liste des contributeurs de son projet. L'autre binôme doit alors effectuer dans son dossier de travail un clone du projet correspondant au projet partagé. Une fois les manipulations réalisées, exécutez la commande « git remote -v ». vérifiez que vous et votre binôme pointez sur le même repository distant.
2. Créez une branche dans votre repository local. Chaque binôme choisit un nom de branche distinct. Basculez sur la nouvelle branche et ajoutez un fichier XML illustrant un nouvel exemple de question au format XML Moodle en s'appuyant sur la documentation Moodle (cf. lien ci-dessus). Sauvegardez et commitez le nouveau fichier sur votre branche.
3. « Envoyez » votre branche sur le repository Github partagé.
4. Basculez sur la branche master et effectuez un merge de la branche précédemment créé.
5. « Envoyez » les modifications sur la branche master sur le repository Github partagé.
6. Définissez avec votre binôme une séquence opératoire permettant d'aboutir à un conflit sur un fichier. Mettez vous en situation de conflit sur un fichier et résolvez le conflit.

Annexe : Création de clé SSH sur machine de TP

L'aide Github pour la création et la prise en compte de votre clé sur Github se trouve ici : https://help.github.com/articles/generating-ssh-keys#platform-windows
Dans l'environnement particulier des machines utilisées en TP, la clé SSH doit être créée dans un dossier sur votre lecteur réseau. Les instructions ci-dessous sont celles décrites dans l'aide Github avec quelques adaptations pour faire en sorte que l'agent SSH de votre machine prenne en compte la clé SSH enregistrée dans votre dossier sur le lecteur « z ». Remplacez « mon_compte » par votre nom de compte Windows et « mon_adresse » par votre adresse e-mail ayant servi à l'ouverture de votre compte Github:

```
$ mkdir /z/mon_compte/.ssh
$ ssh-keygen -t rsa -C "mon_adresse"
Generating public/private rsa key pair.
Enter file in which to save the key (/z/.ssh/id_rsa): /z/mon_compte/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /z/mon_compte/.ssh/id_rsa.
Your public key has been saved in /z/mon_compte/.ssh/id_rsa.pub.
The key fingerprint is:
8d:3e:19:19:ac:43:40:c0:80:6a:96:7a:18:d9:90:85 mon_adresse
$ clip < .ssh/id_rsa.pub
# puis rendez vous sur le site Github pour ajouter la clé dans
# Github. Aidez vous de l'étape 3 de l'aide Github :
# https://help.github.com/articles/generating-ssh-keys#platform-windows
# puis retourner dans Git-Bash pour les opérations suivantes
$ exec ssh-agent bash
bash-3.1$ ssh-add .ssh/id_rsa
Identity added: .ssh/id_rsa (.ssh/id_rsa)
bash-3.1$ ssh -T git@github.com
...Hi ...! You've successfully authenticated, but GitHub does not
provide shell access.
```