5 mai 2021

université PARIS-SACLAY

TER Projet
M1 Miage CFA

Git et squelette de code

Objectif

Le but de cette feuille est de :

1. se familiariser avec `git` et ses concepts, en ligne de commande
2. organiser le dépot du projet
3. importer le squelette de code dans le dépot `git`
4. y apporter des modifications (renommer) et découvrir la structure d'un projet Web Java/JSP
5. voire comment utiliser `git` depuis IntelliJ

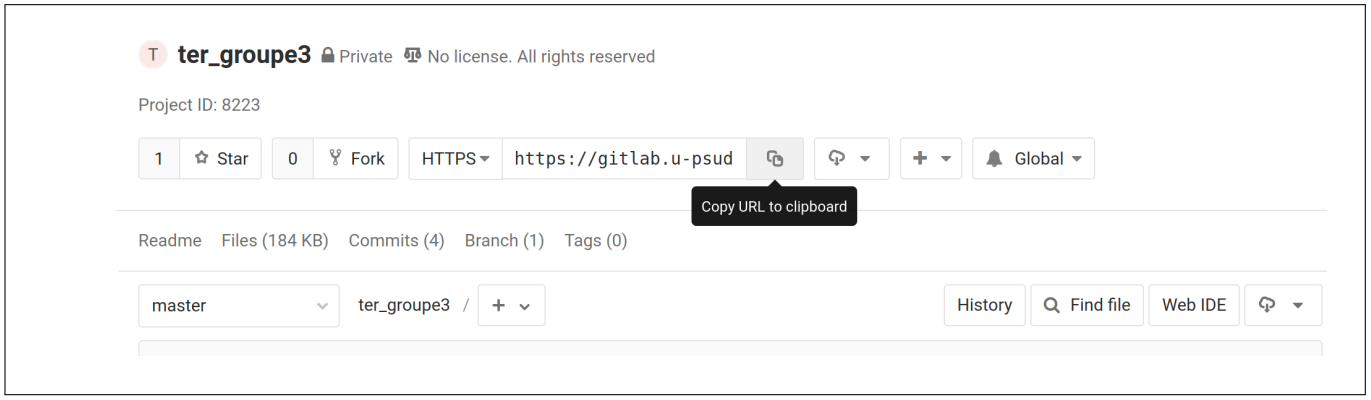
1 Introduction à `Git`

Dans cette section, on effectue toutes les opérations `git` en ligne de commande.

1. Cloner le dépôt `git` de votre groupe :

   ```
   git clone 'https://prenom.nom@gitlab.u-psud.fr/groupe/projet.git'
   ```

   Vous devez **modifier** `prenom.nom` par votre login adonis (le même que vous utilisez sur le site Web `gitlab.u-psud.fr`), `groupe` par le nom de votre groupe et `projet` par le nom du projet.
   Cette URL peut être récupérée directement depuis l'interface Web :



   Il ne faut cependant pas oublier d'ajouter `prenom.nom@` après `http`.
   Normalement, le système d'exploitation mémorise le mot de passe. Si ce n'est pas le cas, (c'est à dire si dans les questions suivantes, lors d'un `push` ou `pull` le mot de passe est demandé, merci de vous manifester).

2. Lors de la première séance, chaque groupe a créé des documents de travail et les a déposé sous git. Nous allons les réorganiser. **Un des membre du groupe** (désigné par le prof) effectue les actions suivantes, depuis le répertoire du projet :

   ```
   mkdir -p doc/
   git mv f1 f2 f3 ... doc/
   ```

   Dans la seconde ligne, il faut remplacer `f1`, `f2`, ... par les fichiers se trouvant dans le dépôt (il est suggéré de ne pas mettre d'accent dans les noms de fichiers). À la fin de cette manipulation votre dépôt ne doit contenir qu'un seule répertoire `doc` contenant tous vos fichiers.
   Effectuer ensuite un `git commit`. Entrer un message de commit lisible (par exemple « Réorganisation du dépôt »).
   Puis faire `git push`.

3. Tous les autres membre du projets peuvent alors faire `git pull` et constater que leur version locale du répertoire est modifiée.
4. Un membre du projet (désigné) crée un fichier texte `README.md` (ou le modifie s'il existe). Il y indique :

```
# Projet XXXX

## Membres
  Foo Bar

## Description
  Todo
```

   où `XXXX` est le nom donné à votre projet et `Foo Bar` sont les noms de la personne. Faire ensuite `git add README.md` puis `git commit` avec un message raisonnable. Il faut ensuite faire `git push` pour envoyer le fichier sur le serveur. Constater que dans l'interface Web, le contenu du fichier `README.md` s'affiche joliment (le format `md` est du Markdown, dans lequel `#, ##, ###` sont des titres de section, `*foo*` met du texte en `gras`, ...).
5. Les autres membres font un `git pull` et éditent le fichier pour ajouter leur nom. Chacun fait `git commit -a README.md` (qui ajoute et `commit` tous les fichiers modifiés) puis `git push`. Que se passe-t'il ?
6. Régler les conflits. Chaque personne qui a eu un conflit, fait `git pull`, édite le fichier `README.md` pour retirer le conflit puis `git commit -a` suivi de `git push`.

2 Import du projet

Dans cette section, un seul membre de chaque groupe fait les manipulation. En fin de section les autre membre du groupes pourront faire un `git pull` pour se mettre à jour.

1. télécharger l'archive `ter_m1_miage_tp01.zip` sur la page du cours et la décompresser dans le répertoire du projet (mais ne pas mettre le `.zip` directement dans ce répertoire).
2. renommer le répertoire `ter_m1_miage_tp01` en |code|
3. ouvrir `IntelliJ` et y importer le répertoire `projet_a_remplacer` se trouvant dans `code` (suivre les indications en visio et le faire en même temps).
4. suivre les indications pour changer toutes les occurrences de `projet_a_completer` en un nom pour votre projet. Ce nom doit être un nom de package Java valide, donc ne contenir que des lettres des chiffres et des `_`.
5. quitter l'éditeur. Renommer aussi le répertoire `projet_a_remplacer` en utilisant le nom choisi pour le package.
6. réouvrir l'éditeur et ouvrir le répertoire renommer. Effectuer l'action `run` (flèche verte). La première invocation peut être longue car `Maven` télécharge les dépendences Java nécessaires. Si un serveur Web est démarré sur le port `9090`. Visiter à l'aide de votre navigateur Web `http://localhost:9090/mon_projet` où `mon_projet` est le nom choisi dans la question 4. S'il y a un souci lors du run c'est sans doute lié au fait que votre environnement n'est pas fonctionnel (pas de Java ou mauvaise version, il faudra faire en sorte que cela fonctionne).
7. quiter `IntelliJ`. Faire `git add code` (dans le répertoire principal du projet) puis `git commit`. Indiquer dans un message de commit que le code initial est importé.
8. pour tous les membres du groupe, faire `git pull` puis ouvrir le projet sur `IntelliJ` et tester s'il fonctionne.

3 Découverte du code

Suivre la démo en ligne du code (la vidéo restera en ligne). Ce code constitue un sequlette fonctionnel permettant de faire votre site. Évidemment tout n'est pas compréhensible aujourd'hui mais cela permet de baliser le travail à faire pour la séance de vendredi :

— créer les tables SQL dans la base du PUIO

— imaginer toutes les étapes nécessaires pour effectuer la fonctionalité « consultation de la liste des produits en mode anonyme »
— quand toutes les étapes ont été identifiées : répartir le travail entre membres du groupe :
  — concevoir les requêtes SQL
  — penser au modèle de données (objets auxiliaires et méthodes de la classe `...DB`
  — penser aux contrôleurs, et donc à l'architecture du site (quelles URLs, quels paramètres)
  — imaginer la vue et la façon d'afficher les choses grossièrement en HTML d'abord (avant de se lancer dans des framework)

  Documenter un maximum de choses dans des fichiers de travail au (format `.md` par exemple) déposés dans `doc/` et commités.