C2R1

Tutorat Git

Version du 30 septembre 2014

Table des matières

1 Introduction **1**
1.1 Histoire . . . 1
1.1.1 Versionnage de code ? . . . 1
1.1.2 Outils de versionnage . . . 1
1.2 Installation . . . 2
1.2.1 Windows . . . 2
1.2.2 Linux . . . 2
1.2.3 Mac OS . . . 2
1.3 Github et Gitlab . . . 2

2 Manipulation basique de git **3**
2.1 Premier commit . . . 3
2.1.1 créer le dépot . . . 3
2.1.2 pull . . . 3
2.1.3 readme . . . 3
2.1.4 add/diff/ignore/commit . . . 3
2.1.5 push . . . 4
2.2 Historique . . . 4

3 Manipulation avancée **4**
3.1 Méthode de développement . . . 4
3.2 branch . . . 4
3.2.1 créer sa branche . . . 4
3.2.2 fusionner . . . 4
3.2.3 résoudre un conflit . . . 4
3.2.4 savoir qui a fait quoi . . . 4
3.3 Se positionner sur un commit . . . 4
3.4 Contribuer à un autre repository . . . 4

1 Introduction

1.1 Histoire

1.1.1 Versionnage de code ?

Avez vous déjà travaillé sur un code en groupe ? Avez vous déjà tenté de programmer un logiciel sur plusieurs versions ? Si oui, vous vous êtes sans doute rendu compte qu'il était nécessaire d'avoir un outil pour pouvoir facilement travailler sur le même code au même moment. Ou alors pouvoir avoir un historique des modifications en fonction des versions. Vous avez donc besoin d'un outil de versionnage de code.

1.1.2 Outils de versionnage

Il existe de nombreux outils de versionnage de code. Les 3 plus connus sont sans doute Git, Mercurial et SVN.

SVN (subversion) Ce fut le plus utilisé pendant longtemps. Développé par la fondation Apache, il s'agit d'une amélioration d'un programme nommé CVS (très peu utilisé aujourd'hui). Le plus gros problème de SVN est qu'il s'agit d'un système centralisé. Un serveur contient donc le code, des clients travaillent dessus. Il n'y a qu'un seul versionning.

Mercurial Contrairement à SVN, il s'agit d'un système décentralisé. Chacun possède son propre repository et publie son code sur le repository public. Une autre différence avec SVN est qu'il utilise la notion de changeset. C'est à dire qu'il préfère garder en mémoire les changements appliqués que les versions des fichiers.

Git Git possède très peu de différences avec Mercurial, mais l'histoire a fait qu'il c'est plus imposé que mercurial (qui est quand meme utilisé dans pas mal de projets dont ceux de Mozilla, même si git est possible). C'est en partie grâce à Linus Torvalds qui en a fait la pub et des projets comme github que nous allons utiliser dans le reste du tutorat. Pour plus d'informations sur les différences vous pouvez vous référer à ces articles : `http://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/`, `http://www.rockstarprogrammer.org/post/2008/apr/06/differences-between-mercurial-and-git/`.

1.2 Installation

1.2.1 Windows

Lancez le programme que vous pouvez trouver sur cette page : `http://msysgit.github.io` ou `http://www.git-scm.com/`

1.2.2 Linux

`apt-get install git` ou `yum install git` selon la distribution

1.2.3 Mac OS

```
sudo port install git-core +svn +doc +bash_completion +gitweb
```

1.3 Github et Gitlab

Pour la suite de ce tutorat, il vous faut créer un compte github : `https://github.com/`. Vous pouvez aussi vous intéresser à gitlab : `https://about.gitlab.com/`

2 Manipulation basique de git

2.1 Premier commit

2.1.1 créer le dépot

2.1.2 pull

2.1.3 readme

Le README est un fichier de présentation du projet. Généralement, on y décrit son installation, les fonctionnalités à venir, comment contribuer, la licence, ...

2.1.4 add/diff/ignore/commit

git diff : montre les changements effectués depuis le dernier commit. Voici la forme d'un fichier diff :

```
     diff --git a/apps/system/js/sound_manager.js b/apps/system/js/sound_manager.js
index b796d13..3847b93 100644
--- a/apps/system/js/sound_manager.js
+++ b/apps/system/js/sound_manager.js
@@ -686,7 +686,11 @@
     };
   }

-  function setVibrationEnabled(enabled) {
+  function setVibrationEnabled(enabled) {
+    //vibrate
+    if ('vibrate' in navigator && enabled) {
+      navigator.vibrate([200, 100, 200]);
+    }
     setVibrationEnabledCount++;
     SettingsListener.getSettingsLock().set({
       'vibration.enabled': enabled
```

On y trouve donc le numéro du commit, les fichiers modifiés, les lignes supprimées ainsi que les lignes ajoutées.

2.1.5 push

2.2 Historique

3 Manipulation avancée

3.1 Méthode de développement

3.2 branch

3.2.1 créer sa branche

3.2.2 fusionner

3.2.3 résoudre un conflit

3.2.4 savoir qui a fait quoi

3.3 Se positionner sur un commit

3.4 Contribuer à un autre repository

Généralement, il y a deux possibilités : * Contribuer sans coder, (graphisme, traduction, communication, ouverture de bugs) * Coder L'ouverture d'issues nécessite de regarder un minimum si l'issue n'a pas déjà été ouverte, de détailler son probleme et de bien regarder la version qu'on utilise. Pour programmer, il faut forker le dépot et le récupérer en local. Puis il suffit de créer sa branche, de réaliser les modifications nécessaires. Quelques fois, il est demandé de créer des tests pour son bout de code. Avant de proposer il faut vérifier son code (norme, lisibilité, ...) Enfin il faut push votre travail sur votre fork et effectuer une pull request. La suite dépend du fonctionnement du projet.