4D & Git デモンストレーションの手順

2020年7月6日

Step 1-0: 準備

git をインストールします。
URL: https://git-scm.com/book/ja/v2/使い始める-Gitのインストール

GitHub アカウントを作成します。
URL: https://github.com/

gitにGitHubアカウントのアカウント名とメールアドレスを設定します：
ターミナルにて：

```
$ git config --global user.name "username"
$ git config --global user.email "email@example.com"
```

Step 1-1:

プロジェクトデータベースを新規作成します。名前は: 4DProjectDemo

Step 1-2:

トムの Method1を追加します (メソッド追加ショートカット Cmd + Shift + K):

```
  //----------------------------------------------
  // メソッド作成者: トム
  //----------------------------------------------
  // メソッド: Method1
  // 説明
  //
  // パラメーター
  //
  //----------------------------------------------

C_LONGINT($vCounter;$r)
For ($vCounter;1;100)

    $r:=do_stuff ($vCounter)  // なんらかの処理
    do_something_else ($r)  // さらなる処理

End for
```

使用する git コマンド:

```
git init
git add .
git status
git commit -m "first commit by Tom"
git status
git log
```

Step 1-3:

プッシュする前に間違った内容を追加した場合は、ローカルで以前のバージョンに戻せます。このロールバック機能を試すため、アラートを表示する簡単なコードをMethod1に追加します。

使用する git コマンド:

```
git commit -a -m "Error commit"
git log (ここでhashをコピーします)
git reset --hard <hash> (ここでhashをペーストします)
git log
```

追加したコードが消えるのを確認します。

Step 1-4:

ローカルリポジトリをリモートリポジトリにプッシュしてみましょう。
まずはGitHubにてプライベートリポジトリを作成します。
httpsのURLをコピーします。

使用する git コマンド:

```
git remote add origin <httpsのURL>
git push origin master
```

GitHub上でリポジトリが更新されたのを確認します。
Method1を確認します。

Step 2-0: 準備
VisualStudioCode をインストールします。
URL: https://azure.microsoft.com/ja-jp/products/visual-studio-code/

日本語化することもできます：
メニューバーから View > Command Palette > Configure Display Language
Install additional language から日本語を選択してインストールします。

Step 2-1:
キムはVisualStudioCodeを使います。
GitHubからhttpsのURLを取得して、リモートリポジトリをクローンします。
一旦VSCodeを終了し、リポジトリの名称を4DProjectDemo-Kimに変更します。

Step 2-2:
4Dでプロジェクトを開き、Method1 に下のコードを追加します:

```
If ($r<0)  // 追加 by キム

    handle_error ($r)  // do_stuff がエラーを返した場合の処理
    do_something_else ($r/2)  // $rの半分の値で処理

End if
```

4Dを終了し、プロジェクトをVSCodeで開きます。
VSCodeで編集履歴を確認します。

Step 2-3:
タイムスタンプの変更履歴がgit管理下にあることを気づいたキムは
.gitignoreを追加して、一旦コミットします。

.gitignoreの内容:

```
Data/
Project/DerivedData/
Project/Trash/
userPreferences.*/
```

使用する git コマンド:

```
git commit -a -m “.gitignore追加”
```

一度キャッシュをクリアし、ファイルを改めて git add で追加します。
コミットおよびプッシュします。

使用する git コマンド:

```
git rm -r --cached .
git add .
git commit -m ".gitignore有効"
git push origin master
```

GitHub上で更新を確認します。
一旦すべて閉じます。

Step 3-0: 準備

まず、GitHub Desktopをインストールします。
URL: https://desktop.github.com/

次に、p4mergeをインストールします。
URL:
https://www.perforce.com/ja/zhipin/helix-core-apps/merge-diff-tool-p4merge
そのままの設定では、日本語のファイルを開くとレイアウトが崩れるため、
環境設定でフォントをOsakaなど日本語対応のものにします。

p4mergeをmergetoolとして登録します。

ターミナルにて、git をインストールしたディレクトリに移動し、
viエディタを使って、.gitconfigに下記内容を設定します：
$ vi .gitconfig

```
[merge]
	tool = p4merge
[mergetool]
	keepBackup = false
[mergetool "p4merge"]
	path = p4merge
	keepTemporaries = false
	trustExitCode = false
```

次に、/usr/local/binにp4mergeシェルを作ります。

ディレクトリを移動します：
$ cd /usr/local/bin

vi エディタを使ってp4mergeシェルを作ります:
$ vi p4merge

```
#!/bin/sh

P4MERGE=/Applications/p4merge.app/Contents/MacOS/p4merge
${P4MERGE} $*
```

vi エディタの使い方：
i で編集モードになります。
Esc で編集モードが終了します。
:wq で保存して終了します。

Step 3-1:
次は編集が競合した場合を再現します。
4DProjectDemo-Kimを複製して4DProjectDemo-Henryに名称変更します。

キムは4DでMethod1を編集します。

```
For ($vCounter;1;200) // 編集 by キム
```

VSCodeで開いてコミットし、プッシュします。

Step 3-2:

ヘンリーも4DでMethod1を編集します。

```
For ($vCounter;1;300) // 編集 by ヘンリー
```

ヘンリーはGitHub Desktopを使っていますので、これを開きます。
コミットし、プッシュしようとすると、エラーが表示されます。

Step3-3:
コンフリクトの解決

使用する git コマンド:

```
git mergetool
```

あらかじめ登録してある p4merge というマージツールが開きます。
採用したい変更を選択して保存し、ツールを閉じます。
GitHubDesktopでコンフリクトが解決されたのを確認し、
コミットし、プッシュします。
GitHubで確認します。

Step3-4:
トムは最新の状態をプルします。