git使い込み術

pochi_black

(manual: http://www.kernel.org/pub/software/scm/git/docs/)

今日のお話

- 説明する事
  - 社内のどんなとこで使われているか
  - gitの基本的なコマンド
  - 逆引きとかで利用するコマンド
- 説明しない事
  - Subversionとの違い
  - 中央集権型SCMと分散型SCMの違い

gitを知るためのご紹介
(コマンドではなくなにができるかを覚えておく)

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- git
+++ git

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- git
+++ git

社内git事情

- Railsアプリ
- Apacheの設定ファイル
- 黒田作成ツール群

--- git
+++

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

ソースコード管理とは

- 開発はRPGのようなもの
  - 技術力は武器
  - コミュニケーション能力は防具
  - 魔法は楽に戦うための特殊能力
- ソースコード管理は開発における冒険の書！
  - コミットする事はセーブすること
    - レベルが上がるとセーブするでしょ？
  - ゲームも仕事も冒険の書があると安心してプレイできる

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

基本コマンド

- レポジトリの初期化
  - git init
- 作業コピーの作成
  - git clone
- 作業コピーの変更
  - git add
  - git rm
  - git mv
- コミット
  - git commit

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

+++ git

リビジョン

- リビジョンの特定方法
  - リビジョン番号
  - キーワード
  - あいまい指定
  - 日付指定
- リビジョン指定方法
  - git log #{revision_number}
- キーワード
  - git log HEAD
  - git log HEAD^^ #=> 2つ前のコミット

git

リビジョン

- あいまい指定
  - 先頭文字から一意なとこまで指定すると自動補完
- 日付指定
  - git log --before=2011-4-12
  - git log --after=2011-4-2
  - git log --before=2011-4-12 --after=2011-4-1

--- git
+++ git

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- +++ git

レポジトリとの差異を確認したい

- マスター(リモート想定)を登録して見に行く
  - git remote add #{name} #{url_or_directory}
  - git fetch #{name}
  - git log #{name}
- 過去の差分を知りたい
  - git log -p #{revision_number}
- まだコミットしていない分の差異を知りたい
  - git diff
- コミット未済だがaddした分の差分を知りたい
  - git diff --cached

--- git
+++

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- git
+++

衝突発生時の対応

- 複数人で開発を行うと衝突が発生し得ます

```
$ git diff
diff --cc moge
index 19a5b4b,e21e265..0000000
--- a/moge
+++ b/moge
@@@ -1,1 -1,3 +1,7 @@@
- pochipochi
++<<<<<<< HEAD
++pochipochi
++=======
+ kurokuro
+
+ hoageohgaoi
++>>>>>>> 0cbc7c504b13ee691e0bd20441df5370cf408189
```

- 手動で修正してコミットしましょう

--- git
+++

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- git
+++ git

削除したファイルの復活

- リビジョン及びファイルを指定する
  - git checkout #{revision} #{file_name}

```
$ git checkout HEAD^ moge
```

--- git
+++ git

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

+++ git

ロック時の対応

- gitではロックという機能が存在しない
  - subversionではあるけど分散系SCMではない機能

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

管理したくないファイルの登録

- ログファイルや環境に依存する設定ファイルは管理したくないファイル
  - gitレポジトリ直下に.gitignoreファイルを作成して対応

```
*.dump

vendor/gems/*
assets
# sass generates below
public/stylesheets/application.css
# application data.
assets/uploaded_data/**/*
```

--- git
+++

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- +++ git

ブランチやタブを作りたい

- ブランチとは？
  - ファイルやディレクトリ・ツリーに対して別の流れを作成する事
  - 安定板と開発版に分ける場合に作成する
  - git checkout -b #{branch_name}
- タグとは？
  - プロジェクトのある時点でのスナップショット
  - リポジトリに対して名前を付ける(Version0.1.0等)
  - git tag #{tag_name}

--- +++ git

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- git
+++ git

ブランチの修正を取り込む

- アプリケーション共通のバグは開発版にも安定板にも取り込みたい
  - git (merge|rebase) #{branch_name} (ORIG_HEAD)
  - ORIG_HEADをつけると取り込んだ後UNDOできる

```
(master)    A-------------------------------------A'
                      |
(working)             ----------------B
```

[merge] => 失敗する。本体に既に更新が入ってしまっているため

[rebase] => 成功する。branchのcommitを取り込んでA'のcommitを加える

```
(master)    A---------                 ---------------A''
                      |               |
(working)             ----------------B
```

--- +++ git

ある地点にロードバックしたい

- パーティーが全滅した場合セーブポイントまで戻りたい
  - git checkout --hard #{revision}
  - git checkout --hard ORIG_HEAD(commitをORIG_HEAD設定していた場合)

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- git
+++ git

参照しているレポジトリを変更したい

- 名前は変えたくないけどURLは変更したい
  - git remote rm #{name}
  - git remote add #{name} #{new_url}
- 単純に参照先を追加したい
  - git remote add #{new_name} #{new_url}

アジェンダ

- 社内git事情
- ソースコード管理ツールとは
- 基本コマンド
- リビジョン
- リポジトリとの差異確認(履歴)
- 衝突時の対応
- 削除したファイルの復活したい
- ロック時の対応
- 管理したくないファイルを登録したい
- ブランチやタグを作りたい
- ブランチの修正を取り込む
- ある地点にロールバックする
- 参照しているリポジトリをスイッチしたい
- gitで設定できる項目が知りたい

--- +++ git

設定できる項目を知りたい

```
[user]
	name = pochi
	email = pochi.black@gmail.com
[core]
	editor = emacs
[color]
	status = auto
	diff = auto
	branch = auto
[alias]
	s = status
	ci = commit
	co = checkout
	cob = checkout -b
	br = branch
	sh = !git-sh
	a = add
	aa = add -A
	b = branch
	sa = stash
	sap = stash pop
	di = diff
	l = log
```

--- +++ git

まとめ

- gitを利用して安心して作業する事が可能
- subversionよりコマンド早いです
- gitサーバがないのが不便です
- test-productionでまだ接続できてないです
- 1commit1チケットでRedmineと連携するといいです！
  - commitが何がしたかったのかが一目瞭然
- これ以外の機能も一杯ありますが最低限今日のこと覚えておけば結構使えるはずです

--- git
+++ git

Safe Hacking!