Правильное использование систем контроля версий

Дунаева Ольга Александровна

2018

Создание репозитория

Интерфейс

- Название управляющей программы: git / **hg**
- Существуют графические интерфейсы: SmatrGit, SourceTree, TortoiseGit, **TortoiseHg**

Создание репозитория

Интерфейс

- Название управляющей программы: git / **hg**
- Существуют графические интерфейсы: SmatrGit, SourceTree, TortoiseGit, **TortoiseHg**

Создание репозитория

- Перейти в выбранный/созданный каталог
- Для создания нового репозитория выполнить команду

```
$ git init
```

- Если репозиторий на сервере, то клонировать

```
$ git clone <адрес>
```

Например, *https://bitbucket.org/olya/studlib*

Структура проекта

- Все файлы в директории остались на месте
- Репозиторий располагается в директории .git
- Все остальные файлы считаются рабочей копией

Для просмотра содержимого каталога используется команда

```
$ ls
```

Для просмотра всех файлов, включая скрытые, используется команда

```
$ ls -a
```

Добавление/Удаление файлов

Надо указать git, что файл должен быть добавлен/удален в систему контроля версий

Работа с файлами

- Добавить `$ git add <имя файла>`
- Удалить `$ git rm <имя файла>`
- Перенести `$ git mv <текущее имя> <новое имя>`

Просто переместить = удаление + добавление нового.
Не зыбываем, что необходимо

фиксировать изменения в репозиторий `$ commit -m` .
Не забываем писать грамотные комментарии!!!

Просмотр состояния рабочей копии

Состояние рабочей копии не синхронизируется с состоянием репозитория автоматически!

Команды для просмотра состояния

```
$ git status
```

Статусы файлов

- ?? - не отслеживается
- M - модифицирован
- A - добавлен
- D - удалён
- ...

Идентификация изменений

Каждое изменение снабжается метаинформацией

- Уникальный номер изменения
- Автор изменения
- e-mail автора изменения
- Время изменения
- Комментарий к версии

Просмотр внесённых изменений

Вы можете свободно манипулировать рабочей копией.
Единственное ограничение: Не удаляйте репозиторий проекта.
Для просмотра изменений используйте

```
$ git diff [<имя файла>]
```

Сохранение изменений

- Выберите файлы, содержащие изменения

  ```
  $ git add <имя файла>
  ```

- Сохраните (зафиксируйте) изменения

  ```
  $ git commit [<имя файла(-ов)>] -m <комментарий>
  ```

Отмена изменений

Любые изменения, внесённые в отслеживаемые файлы, можно отменить, если не удалена директория репозитория.
Для восстановления состояния файла/файлов команда

```
$ git checkout <имя файла>
```

или

```
$ hg revert <имя файла>
```

Восстановление всех файлов до последней версии:

```
$ git checkout
```

или

```
$ hg revert -all
```

Возвращение к прошлой версии

Последовательность действий

- Посмотрите список изменений с помощью
  ```
  $ git log
  ```
- Выберите нужную версию (id commit)
- Выполните команду

```
$ git checkout id commit
```

или

```
$ hg update id commit
```

Если есть локальные изменения, то их можно забыть, накатить на данную версию, отложить в карман/на полку.

Позиционирование в репозитории

Дополнительные элементы: "указатели"на конкретные версии

Известные нам указатели

- HEAD/tip - указывает на текущую версию кода в рабочей копии
- master/default - название ветки по умолчанию

Два типа указателей: ветки (branches) и теги (tags)

Теги

Для любой версии исходных кодов приложения можно добавить читаемое название, называемое тегом.

Просмотр списка тегов

```
$ git tag -l
```

```
$ hg tags
```

Создание нового тега

- Найдите версию, для которой хотите создать тег

```
$ git tag <tag name> -m <message>
```

```
$ hg tag -f [-r version] <tag name> -m <message>
```

Использование тегов

Перемещение к нужной версии исходных кодов

```
$ git checkout <tag name>
```

```
$ hg update <tag name>
```

Удаление тега

```
$ git tag -d <tag name>
```

```
$ hg tag -remove <tag name>
```

Ветки

Ветка (branch) - линия развития изменений.

Просмотр существующих веток

- `$ git branch`

`$ hg branchs`

Создание новой ветки

- `$ git branch <branch name> [<start point>]`

Переключение к другой ветке

`$ git checkout <branch name>`

`$ git update <branch name>`

Слияние веток

Основная задача при создании ветки - разработка функционала, не мешая разработке основного функционала.

Хорошая практика: сохранить все существующие изменения перед слиянием

Перенос изменений из одной ветки в другую

- Выбрать ветку, в которую надо занести изменения и перейти в неё
- Выбрать ветку, разработки которой надо перенести в текущую
- Слить изменения

```
$ git merge <branch>
```

```
$ hg merge -r <id commit>
```

Разрешение конфликтов

При слиянии веток git постарается совместить изменения. Если невозможно автоматически соединить изменения, то git добавит информационные маркеры в места конфликта.

Что делать?

- Узнать: в каких файлах есть проблемы

```
$ git status
```

- Отредактировать проблемные файлы
- Сделать комит, содержащий все изменения

```
$ git commit -a -m
```

Разрешение конфликтов. Продолжение

Для облегчения процесса разрешения конфликта git содержит в себе 3 копии спорных файлов

- `$ git show :1:file.txt # Версия файла от общего предка`
- `$ git show :2:file.txt # Версия файла из HEAD`
- `$ git show :3:file.txt # Версия файла из MERGE_HEAD`

Сравнение файла из рабочей копии с другими

- `$ git diff -base file.txt`
- `$ git diff -ours file.txt`
- `$ git diff -theirs file.txt`

Разрешение конфликтов. Заключение

Специальный инструмент

```
$ git mergetool
```

Отмена слияния до сохранения в репозиторий

```
$ git reset -hard HEAD
```

Отмена слияния после сохранения

```
$ git reset -hard ORIG_HEAD
```

Не отменяйте слияния или другие комиты, если они стали доступны публично, т.е. попали в другие репозитории.

Игнорирование файлов

По умолчанию git попытается отслеживать все файлы в директории

Лучше не отслеживать:

- Временные файлы
- Генерируемые файлы

Для "сокрытия"файлов от git используется файл `.gitignore`

Формат файла .gitignore

- Пустые линии игнорируются
- Линии, начинающиеся со знака # игнорируются
- Линии, заканчивающиеся знаком / считаются директориями
- Знаки * считаются за любую последовательность символов
- Линии, начинающие со знака ! отменяют предыдущие сокрытия

Получение информации о работе с Git

Локальные ресурсы Git

- `$ git`
- `$ git help <имя команды>`

Ресурсы в сети Интернет

- Официальный сайт git `http://git-scm.com/`
- Git Tutorial by Lars Vogel `http://www.vogella.de/articles/Git/article.html`
- duckduckgo.com or google it