Programowanie - Java

Jakub Mazurek
kuba@fenix.club

21 grudnia 2016

Spis treści

1 git - system kontroli wersji

System `git` jest rozproszonym systemem kontroli wersji. W skrócie oznacza to, że wiele osób może pracować niezależnie nad tym samym projektem mając jego kopie na swoich własnych maszynach. Kiedy jest to potrzebne, ich zmiany mogą zostać dodane do centralnej kopii projektu na serwerze.
System `git` tworzy i operuje na tzw. **repozytoriach**.

Główne zalety korzystania z `git`a:

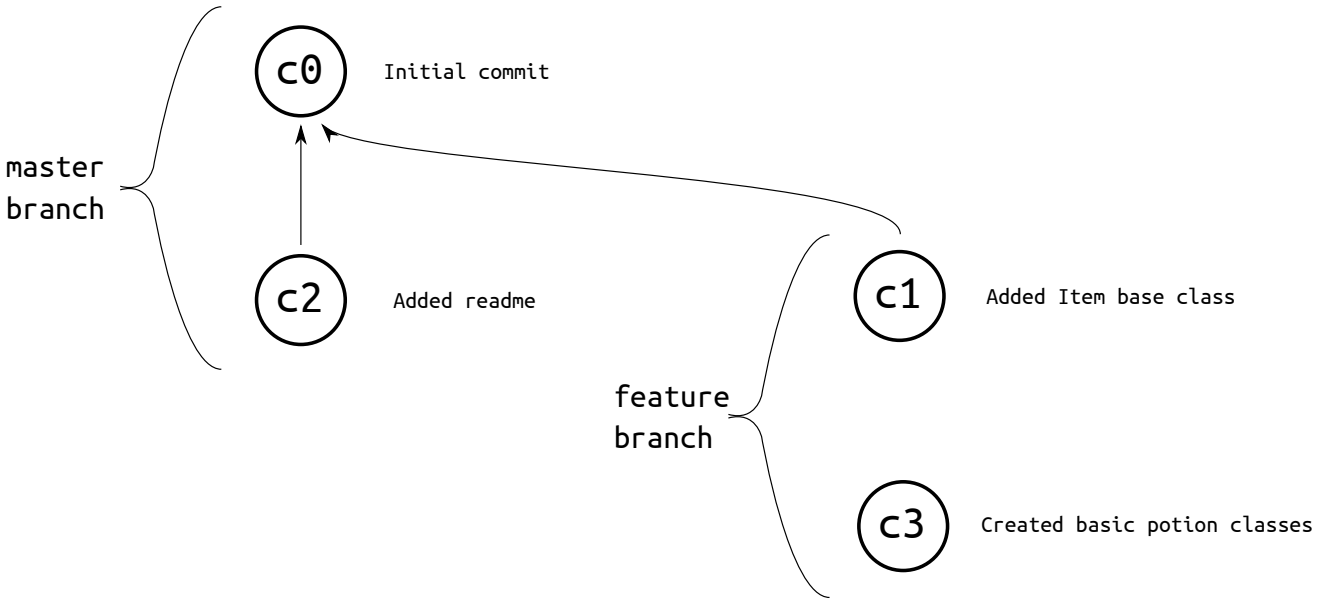
- Znacząco ułatwia jednoczesną pracę wielu osób nad jednym projektem,
- Operuje na zmianach, przez co utrzymuje historię projektu,
- Umożliwia cofanie się do poprzednich stanów repozytorium,
- W przypadku wykorzystania serwera `git`a - utrzymuje kopię zapasową naszego projektu,
- Pozwala na sprawdzanie kodu zanim zostanie on dodany do projektu (pull request).

2 Działanie gita

Stan naszego projektu w systemie `git` jest odzwierciedlany za pomocą **drzewa zmian**. Podstawowymi pojęcami, o których warto wspomnieć na tym etapie są **gałęzie** (ang. *branch*) oraz **commit'y**.

Commit możemy rozumieć jako "jednostkę zmiany"w naszym projekcie (pojedynczy węzeł w drzewie). Poprzez commit wyrażamy zbiór zmian w plikach naszego projektu.

Gałąź jest nazwanym ciągiem następujących po sobie commitów. Wedle konwencji, główna gałąź repozytorium nosi nazwę **`master`**. Gałęzie są niezbędne przy wspólnej pracy nad jednym projektem, ponieważ umożliwiają równoległą pracę wielu osób.



Rysunek 1: Gałęzie

Na powyższej grafice mamy przedstawione repozytorium w którym istnieją dwie różne gałęzie. Jedna z nich to **master**, druga to **feature**.
W przedstawionej sytuacji, gałęzie te są rozłączne (ponieważ każda z nich ma pewne zmiany, których nie ma druga). Pozostaną one w takim stanie dopóki zmiany z jednej nie zostaną **scalone** (ang. *merge*) z drugą.

3 Podstawy pracy z git'em

Z `git`a możemy korzystać z poziomu konsoli lub z wykorzystaniem programów z graficznym interfesjem. Istnieje wiele programów, SourceTree...

Aby sprawdzić czy git jest zainstalowany na naszym systemie: git version Po wywołaniu tej komendy powinniśmy zobaczyć w konsoli:

W przypadku systemu Windows istnieje też wersja linii komend stworzona przez GitHub'a (link).

3.1 Pobieranie istniejących repozytoriów

Jednym z najbardziej podstawowych i najczęstszych zastosowań `git`a jest pobieranie istniejących już repozytoriów na lokalną maszynę. Instrukcją, którą wykorzystamy w tym przypadku jest **`git clone`**. Przykładowo:

`git clone https://github.com/Fenix-Club/java-roguelike`

Powyższa komenda skopiuje zawartość repozytorium znajdującego się pod podanym przez nas adresem na nasz komputer.
Projekt będzie się znajdował w katalogu z którego wywołaliśmy komendę `git clone` w osobnym folderze o nazwie takiej, jak nazwa samego repozytorium (w tym wypadku będzie to **`java-2016-2017`**).

Co w przypadku, gdy repozytorium które chcemy pobrać nie jest widoczne publicznie (tzw. prywatne repozytorium)?

W takiej sytuacji musimy dokonać **uwierzytelnienia** (ang. *authentication*) na serwerze, czyli przedstawienia odpowiednich danych do logowania. W przypadku `git`a, dwoma najpopularniejszymi sposobami uwierzytelniania są:

- **`https`** - w tym przypadku będziemy musieli podać login oraz hasło do odpowiedniej usługi (np. GitHub'a),
- **`ssh`** - w tym wypadku `git` skorzysta z pary kluczy `ssh` na naszej maszynie.

W różnych wypadkach możemy korzystać z różnych sposobów uwierzytelniania. Aby wskazać którego ze sposobów chcemy użyć, możemy odpowiednio zmienić link do repozytorium:

`https://github.com/Fenix-Club/java-roguelike`

`git@github.com:Fenix-Club/java-roguelike.git`

3.2 Zarządzanie lokalnymi zmianami

3.2.1 git status

Mając kopię istniejącego repozytorium na swojej maszynie, zobaczmy teraz jak się nim posługiwać.
Jedną z najczęściej wykorzystywanych komend `git`a jest **`git status`**. Służy nam ona do wyświetlania statusu repozytorium w którym aktualnie się znajdujemy:

```
> git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
```

Powyższa wiadomość jest przykładowym wynikiem tej komendy. Jest tutaj zawartych kilka informacji:

- Podana jest gałąź repozytorium na której obecnie pracujemy (w przykładzie jest to gałąź `master`).
- `git` mówi nam, czy mamy lokalne commity, które nie znajdują się w **ostatnim znanym** stanie repozytorium na serwerze (o ile takie istnieje).

- Ostatnią informacją jest lista plików, które zmieniły się od ostatniego commita (w powyższym przykładzie nie ma takich plików, co jest sygnalizowane wiadomością: `nothing to commit, working directory clean`).

3.2.2 git add

Spróbujmy teraz dokonać jakichś zmian w naszych lokalnych plikach. Stworzymy w naszym katalogu nowy plik o nazwie `temp.txt`.
Oto co będzie wynikiem komendy `git status` po dodaniu naszego pliku:

```
> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

  temp.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Plik o nazwie `temp.txt` został podany jako **untracked**. Oznacza to, że jest nowym plikiem i nie jest jeszcze śledzony przez `git`a.
Możemy dodać go do plików śledzonych przez `git`a za pomocą komendy **`git add`**:

```
> git add temp.txt
> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

  new file:   temp.txt
```

3.2.3 git commit

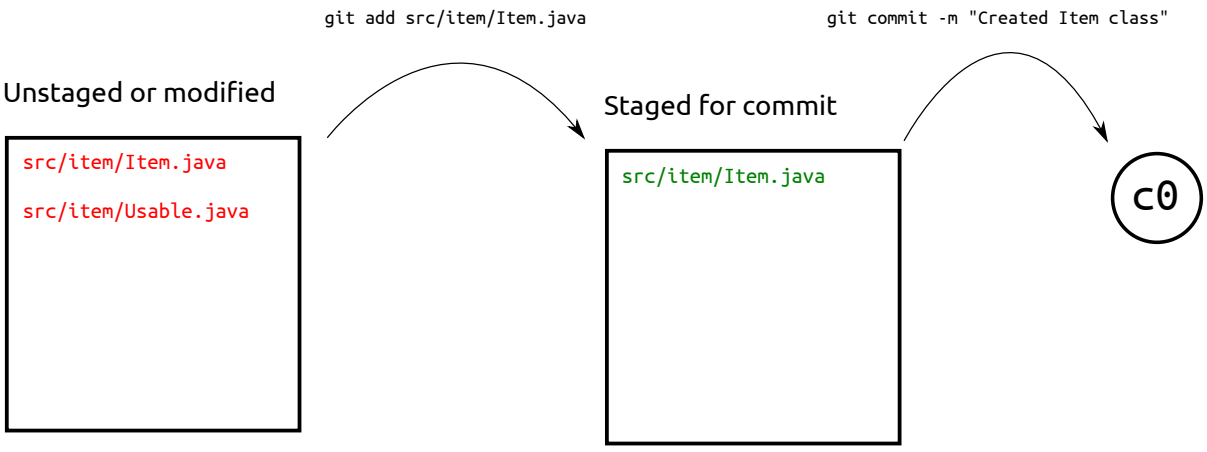
Nasz plik znalazł się teraz w sekcji `changes to be committed` zwanej też `staging`. Oznacza to, że przy utworzeniu przez nas następnego commit'a ten plik wejdzie w jego skład.

Na tym etapie możemy zdecydować się na stworzenie commit'a który zawrze w sobie zmiany dodane przez nas przy pomocy `git add`.
Aby go stworzyć, użyjemy komendy **`git commit`**:

```
> git commit -m "Added temp file"
[master af66d5a] Added temp file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 temp.txt
```

Za pomocą flagi `-m` możemy podać treść wiadomości dla naszego commita. Taka wiadomość powinna być krótkim opisem wprowadzonych przez nas zmian.

Poniżej przedstawiona jest droga pliku od jego dodania do utworzenia commita:



Rysunek 2: Commitowanie zmian

3.3 Przesyłanie zmian

Podstawowe dwie komendy służące do interakcji z repozytoriami na serwerze to:

- **`git pull`** - pobiera nowe zmiany z serwera.
- **`git push`** - wysyła nasze lokalne commity na serwer.

Obydwie z tych operacji odnoszą się tylko do gałęzi na której się obecnie znajdujemy.

4 Zadanie

Wejdź na stronę: learngitbranching.js.org/ i zrób zadania z pierwszej sekcji w zakładce **Main** oraz z pierwszej sekcji zakładki **Remote**. Możliwe jest również swobodne eksperymentowanie z gitem w trybie sandbox.