

# Extrakcia dát z webu

[WebExtraction]

Metodika pre manažment verzií

<b>Tím:</b>	č. 16, WebX
<b>Vedúci tímu:</b>	Ing. Ivan Srba, PhD.
<b>Členovia tímu:</b>	Ján Brechtel, Tomáš Juhaniak, Martin Kalužník, Rastislav Krchňavý, Michal Kren, Martin Lacek, Andrej Vaculčíak
<b>Akademický rok:</b>	2016/2017
<b>Autor:</b>	Michal Kren
<b>Verzia číslo:</b>	1.3
<b>Dátum poslednej zmeny:</b>	12.12.2016

# 1 Úvod

Cieľom tejto metodiky je definovať základné postupy odovzdávania kódu a udržiavania verzií. Metodika opisuje prácu s nástrojom Git.

Touto metodikou sa riadi tím č. 16 WebX v rámci predmetu tímový projekt v akademickom roku 2016/2017. Metodikou sa riadia všetci členovia tímu.

## 2 Skratky a značky

Odovzdanie	(angl. commit) je množina zmien v súboroch projektu v určitom čase podpísaná autorom
Vetva	(angl. branch) je postupnosť odovzdaní. Vetvenie umožňuje spravovať súbory v jednej vetvy repozitáru nezávisle od iných vetiev
Feature	Funkcionalita, ktorá je podporená používateľským príbehom
Git	Distribučný systém na správu revízií zdrojového kódu

## 3 Postupy

### 3.1 Vetvy

- master - produkčné prostredie (spustenie v letnom semestri)
- staging -
- development - spoločná vetva pre vývoj
- ďalšie
  - názov: feature\_<nova funkcionalita>, fix\_<chyba>
  - napr: feature\_registration

Nový kód sa odovzdáva len cez *pull request* (ďalej PR) z vlastnej vetvy. Iný člen tímu merge PR do development vetvy po tom ako prebehne *code-review*.

Do development vetvy sa smie priamo *pushnúť* len v prípade opravy jedného riadku - musí schváliť aspoň jeden ďalší člen tímu.

Prísny zákaz *force push* do inej ako vlastnej vetvy - v prípade porušenia kupuje vinník pivo celému tímu.

Postup nasadzovania zmien

feature\_branch --- PR ---> development --- CI ---> staging (neskôr master)

Odovzdávanie kódu

- commit message
  - v angličtine
  - začína sa slovesom s veľkým začiatočným písmenom
  - jedna správa obsahuje najviac dve slovesá oddelené čiarkou
  - napr.: Fix email confirmation, add error messages to login form
- správa nekončí bodkou
- každý PR sa merge až po code-review

## 3.2 Pomôcka k základným príkazom

Vytvorenie vlastnej vetvy:

(aktuálna vetva je development)  
git pull  
git checkout -b feature\_branch

Commitnutie zmien

git commit -am "Add new feature"

Zobrazenie zmien oproti poslednému commitu

git diff

Prvý push vlastnej vetvy

git push -u origin feature\_branch

Udržiavanie aktuálneho kódu z development vetvy vo vlastnej vetve

### 1. merge

git checkout development  
git pull origin development  
git checkout feature\_branch  
git merge development

(pozn.) git merge --squash

- umožní vyhnúť sa zbytočným merge commitom
- merge commit sa pripojí k ďalšiemu "obyčajnému" commitu
- 

### 2. rebase

git checkout feature\_branch  
git rebase development / git rebase -i development (interactive)  
podľa pokynov vyriešiť konflikty a následne  
git rebase --continue

1. proces možno treba opakovať niekoľko krát, podľa počtu konfliktov

git rebase --abort

2. zruší zmeny spôsobené rebasom a vráti sa na pôvodný stav

Vrátenie sa na predošlý commit (**opatrne**)

`git reset --hard <označenie commitu>`

- označenie commitu získame cez git log
- stratia sa všetky zmeny od commitu, na ktorý sa vrátíme