GIT Spickzettel

Kompetenzbereich Webtechnologie

Titel Git Basics

Ziel Die Schüler sollen nach Durcharbeiten dieses Moduls

1. die Grundlagen und Notwendigkeit einer Versionierungssoftware an Hand von Git erklären und anwenden können
2. einen GitHub-Account und ein Repository anlegen können
3. Username und Email konfigurieren können
4. Git installieren und die wesentlichen Befehle anwenden können

Voraussetzungen Betriebssystem-Grundlagen, Konsolen-Kenntnisse, Bash-Befehle…

Fachbegriffe Git, Bash, add, init, commit, push, pull, merge, branch, status, tag, local repository, repository, clone, diff, master, origin

Links Git tutorials  
<https://www.youtube.com/watch?v=cEGIFZDyszA&list=PL6gx4Cwl9DGAKWClAD_iKpNC0bGHxGhcx>

<http://t3n.de/news/eigentlich-github-472886/>

https://book.git-scm.com/book/en/v2

http://marklodato.github.io/visual-git-guide/index-de.html

GitHub-Account anlegen / Git-Hub-Tutorial  
https://guides.github.com/activities/hello-world/

http://www.vogella.com/tutorials/Git/article.html

Git installieren  
<https://git-scm.com/>

Cheatsheet  
https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet

Ein Repository erstellen  
https://www.atlassian.com/git/tutorials/setting-up-a-repository

Quellen Netz

Git-Spickzettel

Start Warum GIT? Wie gehen wir mit verschiedenen Versionen um? Wie gehen wir mit Updates um?

Erste Schritte Download, Installieren, Windows- und Git-Bash aktivieren

Git-Befehle Konfigurieren  
$ git config --global user.name "Roman Jerabek"  
$ git config --global user.email "jrb@htl.rennweg.at"

Anzeigen aller Einstellungen  
$ git config –list

Anzeigen einzelner Einstellungen  
$ git config user.name

Hilfe allgemein  
$ git help

Hilfe speziell  
$ git help commit

Wo bin ich gerade?  
$ git pwd

Wie komme ich in mein PC-Root?  
$ cd ~

In mein Arbeitsverzeichnis wechseln  
Ordner anlegen und dorthin wechseln  
$ cd c:/Schule/Git/Arbeitsordner

Ein Projekt (Repository) vorbereiten und initialisieren  
$ git init

Was befindet sich in meinem Ordner?  
$ ls

Und wie sehe ich jetzt die versteckten Files?  
Entweder Ansicht im Windows Explorer oder  
$ ls -la

Nun erstellen wir ein einfaches Textfile  
first.txt

Wir möchten, dass GIT alle Änderungen in unterschiedlichen Files im angegebenen Ordner speichert  
$ git add . (der Punkt steht für den aktuellen Ordner)

Wir möchten jetzt „alle“ informieren, was mit der soeben erstellen Datei passiert ist  
$ git commit -m „a message“  
Immer in der Gegenwart schreiben, auf Englisch, genaue Angabe, was geschehen ist.

Welche Änderungen / commits wurden gemacht?  
$ git log

Welche commits gibt es von einem bestimmten Author (es könnten ja auch andere commits gemacht haben in der Zwischenzeit)?  
$ git log –author=“Roman“ −> der Vorname reicht in diesem Fall.

Jetzt fügen wir zwei Files hinzu second.txt und third.txt  
$ git status sagt uns jetzt aber, dass diese Files nicht getrackt werden, da sie noch nicht committed wurden.

Das holen wir jetzt nach  
$ git add second.txt  
$ git status zeigt uns jetzt die Änderung an.

Nun wollen wir die Änderung (Hinzufügen von second.txt zur staging area) auch comitten.  
$ git commit -m „second file ‚second.txt‘ was added“

Änderungen am dritten File dokumentieren  
$ git add third.txt

Drittes Commitment  
$ git commit -m „third file ready for repository“

Erstes File ändern und abspeichern  
$ git status −> jetzt zeigt uns GIT an, dass wir Änderungen durchgeführt haben, was soll geschehen?

Änderungen durchführen und local working area mit dem repository vergleichen  
$ git diff −> nach add und commit sind keine Differenzen mehr zu sehen

Ist das File aber schon mit git add in die stage area „verschoben“ wurde, so wird keine Differenz erkannt. git diff erkennt nur die Unterschiede zwischen dem local workspace und dem repository

Änderungen auch zwischen stage area und repository erkennen  
$ git diff --staged

Sicherste Art zu löschen  
$ git rm filename  
Dabei wird das File s0wohl vom local workspace als auch vom repository gelöscht. Commiten wir das Ganze zusätzlich, können wir später auf diesen Snapshot zugreifen. Ist eine gute Idee, falls wir das File zufällig gelöscht haben sollten.

Beste Art, Dateien umzunennen  
$ git mv oldfile.txt newfile.txt

Beim committen die staging area umgehen  
$ git commit -am „message“ −> ist aber nur beding zu empfehlen, da git dies gleich für alle Files des Git-Ordners durchführt

Eine fehlerhafte Datei des local workspaces durch die „richtige“ Datei im repository ersetzen  
$ git checkout richtigeDatei.txt −> checkout holt die Datei aus dem Repository und überschreibt die „fehlerhafte“ Datei desselben Namens im local workspace.

Git pushen  
$ git push origin master