Inhaltsverzeichnis

Memtext Hochschul-git

Um eine gute Zusammenarbeit zu gewährleisten müssen Änderungen nachvollziehbar sein und alle Beteiligten am aktuellen Stand arbeiten können. Und genau dafür gibt es GIT. Wir setzen GIT ein um mit Hochschulen gemeinsam in unseren Projekten voran zu kommen und die Übersicht zu behalten.

Das git-Repository enthält **nicht** die komplette Installation, sondern nur die hochschulspezifischen Dateien. Das ist ein wichtiger Unterschied zur 3-Säulen-Architektur bei HIS. Unser Ziel ist es, dass das Repository mehrere Releases "überleben" kann. Außerdem ist die Zahl der Dateien überschaubar.

Registrieren

imgsrcgit_reg1.jpgwidth400pxcaptionRegistrierung

Um an unseren und hochschulinternen Projekten teilzunehmen muss man sich lediglich in unserem GIT registrieren.

Dazu sind folgende Schritte notwendig:

1. Auf Seite gehen: https://superx-rocks.de/git/
2. Oben rechts auf "Registrieren" klicken
3. Benutzername und Passwort wählen und E-Mail Adresse angeben
4. Fertig

imgsrcgit_reg2.jpgwidth400pxcaptionRegistrierung

Danach ist man sofort angemeldet.

imgsrcgit_ang.jpgwidth400pxcaptionAngemeldet

Bitte teilen Sie uns nach der Registrierung Ihre Kennung mit und wir richten für Sie dann die Organisation und das Repository ein.

Repository verwenden

Das Repository kann man im Browser, lokal auf dem PC oder auch direkt auf dem Server verwenden.

Daten bearbeiten

Um Daten zu bearbeiten oder erst mal einen bestimmten Stand in das Git zu bekommen kann der Browser oder auch Eclipse oder ähnliches verwendet werden. Die Ordnerstruktur sollte der auf dem Server ähneln, um später diese auch direkt auf dem Server Synchronisieren zu können.

Daten im Browser bearbeiten

Um die Daten im Browser zu bearbeiten, geht man zunächst in das gewünschte Repository. Dazu klickt man in der Übersicht unter Repositories auf das gewünschte. Hier im Beispiel steht nur eins zur Verfügung.

imgsrcgit_edit1.jpgwidth400pxcaptionBearbeiten

Man landet im root Verzeichnis des Repositories. Mit Klick auf eine Datei kann diese nun bearbeitet werden oder mit klick auf einen Ordner, gelangt man in diesen. Wenn es mehrere Unterordner gibt und dazwischen keine weiteren Verzweigungen oder Dateien, werden diese gebündelt dargestellt. D.h. bei klick auf diesen Link gelangt man direkt zu dem Punkt an dem es Dateien oder weitere Verzweigungen gibt.

imgsrcgit_edit2.jpgwidth400pxcaptionBearbeiten

Möchte man eine neue Datei irgendwo dazwischen anlegen klickt man zuerst auf den gebündelten Ordner und dann oben drüber in der Pfadangabe auf den Ordner wo man hin möchte.

imgsrcgit_edit3.jpgwidth400pxcaptionBearbeiten

Um eine Datei nun zu bearbeiten geht man in den entsprechenden Ordner und klickt diese an. Dadurch kann man diese erst mal betrachten.

imgsrcgit_edit4.jpgwidth400pxcaptionBearbeiten

Wenn man nun auf den Stift klickt, kann man diese auch bearbeiten.

imgsrcgit_edit5.jpgwidth400pxcaptionBearbeiten

Wenn die Arbeiten abgeschlossen sind, scrollt man nach unten. Dort gibt es 2 Textfelder unterhalb von "Anderungen committen". In dem ersten gibt man eine kurze Beschreibung ein, was geändert wurde und in dem unteren Textfeld eine lange Beschreibung.

imgsrcgit_edit6.jpgwidth400pxcaptionBearbeiten

Die kurze Beschreibung erscheint immer direkt neben den Dateien in der Übersicht. Hier am besten nur ganz wenige Stichworte verwenden.

imgsrcgit_edit7.jpgwidth400pxcaptionBearbeiten

Die lange Beschreibung kann man sich mit klick auf die Kurzbeschreibung anzeigen lassen und dazu auch direkt die Änderungen. Somit sieht man die Änderungen und hat auch direkt die lange Erklärung dazu.

imgsrcgit_edit8.jpgwidth400pxcaptionBearbeiten

Klonen und Synchronisieren des Repository

Klonen des Repository

Auf dem Ziel-Server kann man sich dann entscheiden, ob hier die Daten per https oder per ssh geladen werden sollen.

Die Git Anbindung per SSH war bisher immer etwas stabiler, außerdem kann man die Passworteingabe leicht ausschalten.

Dazu muss der SSH Schlüssel des Servers im GIT Profil hinterlegt werden. Loggen Sie sich dazu auhttps://superx-rocks.de/git mit der Kennung, die auf dem Server verwendet werden soll ein und wechseln oben rechts bei dem Profil auf Einstellungen.

imgsrcgit_ssh_key.jpgwidth400pxcaptionSSH-Key

Hier sind folgende Schritte nötig:

1. Klick auf auf "SSH- / GPG-Schlüssel".
2. Dann bei "SSH-Schlüssel verwalten" auf "Schlüssel hinzufügen" klicken.
3. Für den SSH Schlüssel einen Namen wählen. Der Name dient nur zur Identifizierung für einen selbst. Kann also frei gewählt werden.
4. Bei Inhalt kommt dann der Schlüssel aus "~/.ssh/id_rsa.pub" aus dem Home Verzeichnis des Servers rein. (Falls es die Datei auf dem Server noch nicht gibt kann diese mit "ssh-keygen" in der Shell angelegt werden)
5. Zum Abschluss noch auf "Schlüssel hinzufügen" klicken. Fertig.

Wenn das eingerichtet ist sollte mit folgendem Befehl das Repository verwendet werden können.

```
git clone ssh://git@superx-rocks.de:22222/</>.git
```

Alternativ, falls Ihre Firewall den Port 22222 sperrt, kann auch https für die Übertragung verwendet werden, allerdings wird nach unserer Erfahrung immer nach einem Passwort gefragt und kann daher nicht automatisiert ablaufen.

```
git clone https://<@superx-rocks.de/git/</>.git
```

Synchronisieren des Repository

Wir befinden uns in unserem Schaubild

imgsrcgit_und_rsync2.pngwidth800pxcaptionRepository und Installation

in dem rot umrandeten Bereich, d.h. wir haben das Repository lokal geklont, und synchronisieren dies nun mit der Installation, d.h. dem Ziel-Tomcat-Server. Dazu liefern wir ein Script aus:

- Für SuperX-Installationen:

```
rsync_to_superx.x
```

- Für HISinOne-BI-Installationen:

```
rsync_to_h1.x
```

Die Scripte nutzen die folgenden Umgebungsvariablen:

```
LOCAL_DIR -> Quellverzeichnis
REMOTE_HOST ->Ziel-Hostname
REMOTE_USER ->Ziel-Benutzerkennung
REMOTE_DIR -> Zielpfad
```

Die ersten beiden REMOTE-Variablen werden nur benötigt, wenn die Installation auf einem anderen physischen Rechner liegt. Der Zielpfad ist das Verzeichnis, bei SuperX Installationen /home/superx, bei HISinOne-BI .../webapps/superx

Entwicklung mit dem Repository

Manche Repositories liefern auch ANT-Scripte mit aus, mit denen SuperX-Module aus dem Quellcode erstellt werden können. Wenn dies der Fall ist, liegt im Wurzelverzeichnis eine Datei "build.xml".

Zunächst installieren Sie das BuildTool ANT. Wir empfehlen die Versionen 1.10 oder höher. Hier die Schritte für die Shell am Beispiel der Version 1.10.12:

```
wget https://dlcdn.apache.org//ant/binaries/apache-ant-1.10.12-bin.tar.gz
tar -xzvf apache-ant-1.10.12-bin.tar.gz
ANT_HOME=./apache-ant-1.10.12
export ANT_HOME
PATH=$PATH:$ANT_HOME/bin
export PATH
```

Danach können Sie die Build-Befehle auflisten mit

```
ant -p
```

Die wichtigsten git Funktionen

Woher kommt git

Linus Torvalds, der "Vater" von Linux, nutzte anfangs eine kommerzielle Versionierungsssoftware. Als die Lizenz restriktiver wurde, suchte er nach Alternativen. Die damaligen "Platzhirsche" CVS und SVN gefielen ihm nicht, u.a. weil sie (zum Committen) eine permanente Verbindung zum Server voraussetzte. Daher entwickelte er kurzerhand eine neue Software, die alle Features bot, die er brauchte. Da er wegen "Linux" im Ruf stand, seine Softwareprodukte mit seinem eigenen Namen zu verbinden, taufte er seit Software "git" (zu deutsch "Trottel") - ein ironischer Seitenhieb.

Nützliche Werkeuge

Wir haben oben beschrieben wie Sie mit git im Browser arbeiten können. Es gibt aber mit der Kommandozeile noch wesentlich mächtigere Werkzeuge.

- erstes Tool der Wahl ist natürlich das Kommandozeilen-Werkzeugt "git". Details dazu siehe unten.

git gui

- git gui ist ein (etwas altbacken wirkendes) graphisches Frontend

imgsrcgitgui.pngwidth800pxcaptiongit gui

Vorsicht: Sie sollten das tool nur in der englischen Lokalisierung nutzen, bei der deutschen stiften Sie nur Verwirrung. Auch bei Konflikten arbeitet es nicht sehr gut.

gitk

Auch das Tool "gitk" wirkt zwar auf den ersten Blick recht "altbacken", aber es eignet sich sehr gut zum Durchsuchen einer Commit Historie bzw. zum Prüfen von Änderungen.

imgsrcgitk.pngwidth800pxcaptiongitk Beispiel

Eclipse Plugin

- Auch in Eclipse gibt es ein git-Plugin, das bei der Java oder J2EE-Edition nicht einmal mehr nachinstalliert werden muss. Dies fügt sich nahtlos in die Oberfläche ein:

imgsrcgit_eclipse.pngwidth800pxcaptionEclipse und git

Sie finden die git-bezogenen Menüs mit der rechten Maustaste unter "Team".

Nützliche Kommandozeilen-Befehle

Umgebung

Die Kommandozeile funktioniert, wenn man sich in einem Verzeichnis befindet, unterhalb dessen das Repository "geklont" wurde. Mit

```
git status
```

zeigt man den aktuellen Status an.
Unterhalb es Stamm-Verzeichnisses gibt es lokal immer einen Ordner ".git", der unsichtbar ist. Und dort gibt es eine Textdatei "config", die wichtige Konfigurationen enthält.

Das lokale Verzeichnis ist ein Klon des Remote Repository. Mit

```
git pull
```

bekommen Sie immer den aktuellen Stand. Dies sollten Sie regelmäßig machen, damit Ihre lokale Installation nicht zu sehr divergiert.

Arbeit mit Branches

Lokale Branches

Bei git entwickelt man typischerweise in sog. "Branches", d.h. man zweigt vom ausgelieferten Code (der "Hauptbranch", früher hieß der "master", neuerdings nennt man es "Default-Branch") ab, entwickelt dort in Ruhe, und wenn alles funktioniert "merged" man seinen Branch in den Hauptbranch.

In welchem Branch man sich befindet, zeigt folgendes Kommando an:

```
git branch
```

Ergebnis z.B.:

```
* master
```

Dabei wird der Branch, in dem man sich befindet mit einem "*" gekennzeichnet. Einen neuen lokalen Branch erstellt man einfach mit

```
git checkout -b meine_coole_entwicklung
git branch
master
* meine_coole_entwicklung
```

Wenn ein erstellter lokaler Branch nicht mehr benötigt wird, kann dieser mit

```
git branch -D meine_coole_entwicklung
```

gelöscht werden.

Remote Branches

Der neue Branch ist zunächst nur lokal verfügbar. Dies kann man so beibehalten, falls man nur alleine in diesem arbeitet. Er lässt sich jedoch auch mit

```
git push  --set-upstream origin meine_coole_entwicklung
```

auf ein entferntes Repository veröffentlichen um mit anderen zusammen zu arbeiten.

Wenn man fertig ist und einen stabilen Stand hat, kann man diesen in den "master" mergen:

```
git checkout master
git pull
git merge meine_coole_entwicklung
git push
```

Alle verfügbaren romote-Branches kann man sich anzeigen mit

```
git branch -r
```

die entfernten Branches anzeigen lassen (-a würde lokale und entfernte anzeigen) und den entsprechenden Branch mit

```
git checkout -b jemandes_coole_entwicklung origin/jemandes_coole_entwicklung
```

lokal verfügbar machen.

Auch den Remote Branch können Sie löschen mit

```
git push origin :meine_coole_entwicklung
```

Wenn Branches zu grob sind

Manchmal will man in der Commit Historie zu einem speziellen Commit zurückkehren (z.B. um zu prüfen ob damals noch alles funktionierte). Dafür sind Branches nicht geeignet, weil man die im Nachhinein nicht anlegen kann. Sie können über die Commit Historie den Hash ausfindig machen und so ganz leicht zum damaligen Stand zurückkehren, z.B.:

```
git checkout 4735d97b8f44e6d809783019f6ce0fe515d621c2
```

Hier sollten Sie nicht committen oder branchen, es dient nur der Diagnose.

Committen

Neue Commits erstellen

Bei git committen Sie in zwei (oder drei) Schritten:

1. Sie ändern den Code, und fügen dann die Änderung mit

```
git add Dateipfad
```

auf eine Art "Bühne", also Dateien, die für einen Commit vorgemerkt sind. Dies wiederholen Sie für beliebig viele Dateien, die Sie in einem Commit bündeln wollen, oder Sie geben direkt ein

```
git add -A -v
```

Damit werden alle geänderten Dateien auf die Bühne geschoben.
Vorher sollten Sie mit *git status* anzeigen, was alles "geadded" wird.

2. Sie committen mit einer sprechenden Nachricht:

```
git commit -m "Bugfix meiner coolen Entwicklung #Ticketnr"
```

3. Wenn Sie mit remote branches arbeiten publizieren Sie die Änderung mit

```
git push
```

Commit-Hashes

Git speichert jeden einzelnen Commit mit einem sog. "Hash"-Wert, also einer SHA-1-Zufalls-Zeichenkette, die eindeutig ist und im Sinne einer digitalen Signatur auch nicht änderbar ist.

Die Hashes können Sie z.B. nutzen, um Zustände einer Datei

- aus der Vergangenheit wieder herzustellen (checkout)
- Einzelne Commits zurückzunehmen (revert)
- Differenzen zwischen Commits anzuzeigen (diff

Eine Konvention bei git ist es, dass man die Hashes abkürzen kann, also statt des gesamten Hashes z.B.

```
4735d97b8f44e6d809783019f6ce0fe515d621c2
```

kann man auch nur die ersten 8 Stellen nehmen:

```
4735d97b
```

Commit-Historie einsehen

Mit dem Kommando

```
git log --pretty=format:"%h - %an, %ar : %s" Dateipfad
```

kann man sich die Historie von Dateien ansehen. Noch komfortabler geht das mitgitk

imgsrcgitk.pngwidth800pxcaptiongitk

Den Hash kann man sich im Feld "SHA1-ID" mit STRG-C rauskopieren.

Wenn man z.B. einen Patch erstellen will, kann man die Dateien erstmal auflisten:

```
git log --name-only Dateipfad
```

Diese Dateiliste kann man in eine Textdatei kopieren, und mit folgenden Kommando sortieren und alle Duplikate entfernen, z.B. die Datei files.txt:

```
cat files.txt | sort -u >files_sorted.txt
```

Daraus kann man dann eine zip-Datei erzeugen, mit allen Dateien in files_sorted.txt:

```
zip mein_patch.zip -@ < files_sorted.txt
```

Wenn Sie in der Shell Differenzen einer Datei zwischen zwei Commits einsehen wollen, können Sie die SHA-Ids aus obigem git log Kommando direkt vergleichen, mit

```
git diff <
```

also z.B.

```
git diff 3e9919e7c 231b35927 -- superx/WEB-INF/conf/edustore/db/module/sos/hilfstabellen/sos_stg_aggr_fuellen.sql
```

Wenn Sie einen Commit nach dem Hash z.B. 3e9919e7c suchen, schreiben Sie

```
git show 3e9919e7c
```

Konflikte

Wenn es Konflikte gibt, z.B. weil die gleiche Datei von verschiedenen Parteien geändert und "gepushed" wurde, können Sie ein sog. "mergetool" nutzen, um die Änderungen einzeln zu prüfen:

```
git mergetool --tool=meld
```

Resetten

Falls Sie Ihre Änderungen verwerfen möchten, können Sie wie folgt vorgehen:

Wenn einzelne Dateien/Verzeichnisse zurückgesetzt werden sollen, und noch nicht geadded/commited wurde:

```
git checkout -- Dateipfad
```

Bei Angabe eines Verzeichnisses werden rekursiv alle Dateien in allen Unterverzeichnissen zurückgesetzt.

Wenn man z. B. alle Änderungen im aktuellen Verzeichnis und dessen Unterverzeichnissen verwerfen will, gibt man ein:

```
git checkout -- .
```

(wichtig ist der Punkt am Ende)

Wenn Dateien schon committed wurden, kann man die remote-Datei erzwingen mit

```
git checkout --theirs -- Dateipfad
```

Wenn Commits schon gepusht wurden, holt man sich aus der Commit Historie den Hash des Commits, um dann z.B. den Commit rückgängig zu machen:

```
git revert 4735d97b8f44e6d809783019f6ce0fe515d621c2
git push
```

Pakete erzeugen und installieren

Wenn das genutzte Repository ein "Modul-Repository" ist, können sie aus den Quellen einen Build erzeugen, und daraus entweder eine Distribution erzeugen (zip-Datei), oder direkt in einem Ziel-Su

Beispiel:

https://dlcdn.apache.org//ant/binaries/apache-ant-1.10.12-bin.tar.gz

Nach dem Entpacken setzen Sie ANT_HOME, sowie übernehmen die Binary von ANT in den PATH:

```
ANT_HOME=/home/superx/tools/apache-ant-1.10.12
export ANT_HOME
PATH=$PATH:$ANT_HOME/bin
export PATH
```

Danach können Sie die Modulscripte generieren mit (Beispiel COStage):

```
. SQL_ENV
cd git/COSTAGE/
ant  -DMODULE_PATH=$COSTAGE_PFAD -DBASE_DIR=. -DWEBAPP=$WEBAPP -DMODULE=costage all
```

So erzeugen Sie dann ein SuperX-Paket (Beispiel COStage):

```
ant  -DMODULE_PATH=$COSTAGE_PFAD -DWEBAPP_DIR=$WEBAPP -DMODULE=costage dist
```

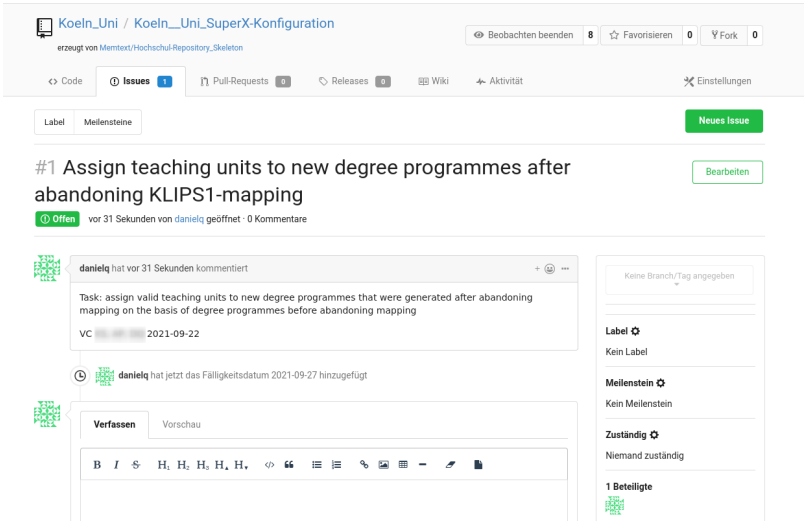
Dokumentation

Zur Dokumentation können Sie das interne Wiki und Ticket-System nutzen.

Ticket System

Tickets anlegen und bearbeiten

Im Ticket System können Sie ein Ticket anlegen, im Menü "Issue":



Hier können Sie ein Thema, eine Beschreibung, und Ziel-Meilensteine bzw. Fälligkeiten definieren. In der rechten Seitenleiste können Sie auch steuern, ob Sie bei Änderungen Emails "abonnieren" o

Es gibt in der Bearbeitung ein paar Unterschiede zu HISZILLA:

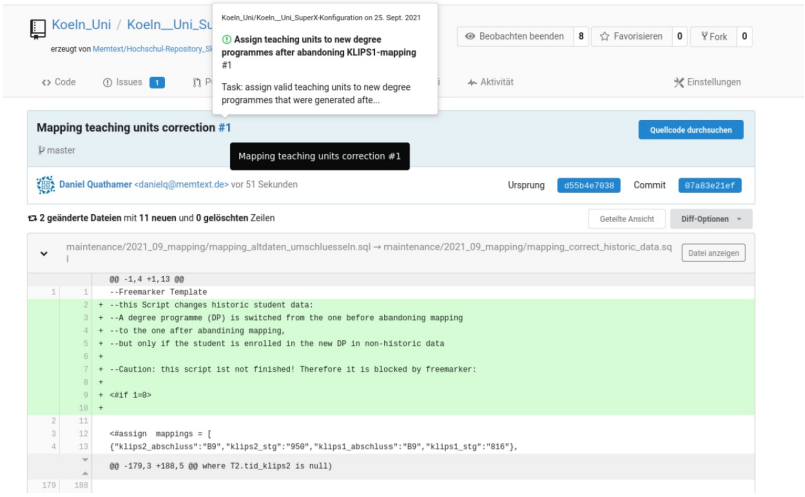
- Ticket Kommentare lassen sich bearbeiten / löschen
- Sie können nur bestimmte Dateitypen anhängen (Text). Binärdateien oder große Dateien bitte über git selbst hochladen.
- Zur Fomatierung wird Markdown genutzt
- Man kann in seinem Profil einen Avatar wählen oder anlegen

Kombination mit Commits

Jedes Ticket bekommt eine fortlaufende Nummer (hier im Beispiel die #1), auf die Sie in Commits referenzieren können. Wenn Sie z.B. einen Commit mit der Message

```
git commit -m "Mapping teaching units correction #1"
```

anlegen, wird dieser später in der Browser Oberfläche automatisch mit den Ticket verlinkt:



Sie können auch auf Issues in andere Repositories verlinken, nach dem Schema

```
owner/repository#1234
```