Aktualisierungsprozess von fn2Web

Versionierung, Releases und Updates für fn2Web

fn2Web ist nach dem Schema <fn2Web-Version>.<Release>.<Update> versioniert. Die Versionsnummer

```
2.04.05.032
```

beschreibt zum Beispiel die fn2Web-Version 2.04, das Release 5 und das Update 32.

fn2Web-Version
Die fn2Web-Version wird nur bei grundlegenden, tiefgreifenden Änderungen im System fortgeführt, was nur sehr selten der Fall ist.

Releases
Releases werden von uns grob zwei Mal im Jahr geplant, zum Jahres-Anfang und im Herbst. Optional können bei Bedarf zusätzliche Releases veröffentlicht werden, z.B. wenn ein schwerer Fehler im System eine zeitnahe Aktualisierung bedingt. Releases beinhalten aufwändigere Erweiterungen und Anforderungen und verlangen häufig das Anpassen von hochschulspezifischen Konfigurationsdateien.

Obwohl ein neues Release so schnell wie möglich eingespielt werden sollte, ist uns bewusst, dass das aufgrund des hohen Test-Aufands nicht immer möglich ist. Daher versorgen wir sowohl das aktuelle, als auch das vorhergehende Release mit Updates.



Die Termine für Releases und Updates sind absichtlich etwas vage gehalten. Die dadurch erlangte Freiheit ermöglicht es uns zeitnah und agil auf die Bedürfnisse unserer Kunden eingehen zu können.

Updates
Updates sind meist kleinere Erweiterungen und dringende Fehlerbehebungen, die von uns zeitnah veröffentlicht werden. Es ist nicht zwingend notwendig, fn2Web immer auf dem neuesten Update-Stand zu halten. Dennoch sollten die Release Notes aktiv gelesen werden, da mit Updates auch kritische allgemeine Fehler behoben werden. Eine viel genutzte best practice ist auch im Ticketing-

System die neuesten erledigten Tickets im Auge zu behalten oder, bei Problemen, dort erst ein mal stichpunktartig zu suchen.

Releasezyklus für fn2Module

Der Releaseprozess für die fn2Module ändert sich nicht. Neue Versionen, die von Kunden angeforderte Features enthalten, werden nach Fertigstellung bereitgestellt. Die in den neuen Versionen enthaltenen Änderungen sind in den dazugehörigen Release Notes dokumentiert.

Gegenseitig abhängige Updates von fn2Module und fn2Web

Wenn es zwingend erforderlich ist, dass beide Bereiche (fn2Module und fn2Web) voneinander abhängig sind und folglich gleichzeitig aktualisiert werden müssen, gibt es dazu von uns vorab entsprechende Informationen. Ein simultanes Update von fn2Web und fn2Module ist bspw. erforderlich, wenn neue Modellierungsmöglichkeiten oder neue Druckkomponenten hinzugefügt worden sind. Bei kleineren Änderungen hingegen, z.B. Bedienverbesserungen in fn2Module, kann das Update von fn2Module unabhängig von fn2Web eingespielt werden. Das gleiche gilt für fn2Web.

(SQL-)Skripte zur Aktualisierung der Datenbankstrukturen

Sowohl für fn2Module als auch für die Komponenten von fn2Web gibt es SQL-Skripte. Diese **müssen** eingespielt werden.

Dateien mit den Skripten für die zwei von uns unterstützten Datenbanken (Oracle DB und PostgreSQL) finden sich auf dem ftp-Server. Bitte suchen Sie selbst die passende Variante heraus. Achten Sie darauf, dass Sie Skripte lückenlos einspielen. Dabei hilft die eindeutige Skript-Nummer. Achten Sie auch darauf, jedes Skript mit „commit;“ abzuschließen, sofern kein Autocommit eingestellt ist. Sollte ein Skript eine längere Laufzeit haben, brechen Sie es **nicht** ab, auch wenn damit „nur“ ein Default-Wert gesetzt wird.

Es kann vorkommen, dass Sie selbst Werte im Skript eintragen oder ergänzen müssen. Beachten Sie dazu die Hinweise in den Kommentaren.

Bei Rückfragen wenden Sie sich zunächst an die im Skript angegebene Ansprechperson. Sollte ein Problem auftreten, das mehrere User betreffen könnte, eröffnen Sie bitte ein Ticket in dem entsprechenden Bereich SQL-Skripte unseres Ticketing-Systems

git zum Download fn2Web-Versionen

git ist eine freie Software zur verteilten Versionsverwaltung und kann hier für alle gängigen Plattformen heruntergeladen.

git verwaltet Änderungen (**Commits**) und gliedert diese in Zweige (**Branches**), die von einander abgezweigt und zusammen geführt werden können. Alle Branches zusammen genommen werden als ein **Repository** bezeichnet. Ein git-Repository existiert immer erst einmal für sich lokal, wobei einzelne Branches mit den Zweigen eines anderen Repositories verknüpft werden können. Andere

Repositories können bei einem lokalen Repository als sog. **Remotes** hinterlegt werden.

Eine Übersicht zahlreicher grafischer Oberflächen für git gibt es hier. Die Verwendung einer grafischen Oberfläche ist allerdings nicht zwingend erforderlich, da alle benötigten Befehle in diesem Wiki beschrieben werden. Ein Grundlagenbuch ist hier frei verfügbar. Ein tieferer Einblick in die Syntax von git-Befehlen kann in der offiziellen Dokumention erlangt werden.

fn2Web kann von unserem GitLab-Server bezogen werden. Grundsätzlich gibt es zwei Möglichkeiten das fn2Web-Repository herunter zu laden, die im Folgenden beschrieben werden:

Direkter Download von fn2Web (nicht empfohlen)

Das Repository und auch einzelne Dateien können dirkt über die GitLab-Oberfläche in verschiedenen Formaten gepackt herunter geladen werden.



Da hierbei sämtliche Vorteile, die git bietet, ignoriert werden, sollte so nur in Ausnahmefällen vorgegangen werden.

Verknüpfung mit einem lokalen git-Repository (dringend empfohlen)

Der wesentlich elegantere und ganzheitlichere Weg fn2Web zu beziehen ist ein lokales git-Repository mit dem fn2webapps-Repository auf dem GitLab-Server zu verknüpfen.

Im Folgenden werden die jeweiligen Konsolen-Befehle benutzt. Es gibt zahlreiche grafische Tools für git, die im Grunde alle das gleiche können, da sie auf eben exakt jene Befehle zurückgreifen. Hier auf einzelne Tools ein zu gehen würde den Rahmen dieses Wiki-Eintrages sprengen.

Ersteinrichtung
Initialisierung (Erstellen) eines lokalen git-Repositories:

```
git init
```

erstellt einen Ordner .git/ im aktuellen Verszeichnis (für fn2Web-Zwecke sollte dieses Verzeichnis leer sein) in dem sich die Verwaltungsdateien für das Repository befinden.

Verknüpfen des Repositories mit dem fn2webapps-Repository über eine Remote:

```
git remote add fn2web https://gitlab.ihb-eg.de/fn2/fn2webapps.git
```

erstellt eine Remote mit dem Namen „fn2web“ und verknüpft diese mit dem fn2webapps-Repository.

```
git fetch fn2web
```

läd die Informationen zur Remote „fn2web“ herunter. Es werden noch keine Dateien erstellt.

```
git checkout -b 2.04.06 fn2web/2.04.06
```

erstellt einen neuen lokalen Branch „2.04.06“, verknüpft diesen mit dem Branch 2.04.06 im fn2webapps-Repository und checkt ihn aus, d.h. es werden die tatsächlichen Dateien, wie sie im Branch hinterlegt sind, erstellt. Die Ersteinrichtung ist damit abgeschlossen.

Mit

```
git log
```

kann überprüft werden, auf welchem Stand sich der Branch befindet (genauer gesagt können so die Commits des Branches betrachtet werden).

Aktualisierung
Das Aktualisieren des momentan ausgecheckten Branches ist denkbar einfach:

```
git pull
```

Soll auf ein neues Release aktualisiert werden, ist dieses im fn2webapps-Repository als ein eigener Branch vorhanden (Branch „2.04.05“ ist Release 5, Branch „2.04.06“ ist Release 6, etc.). Es muss lediglich ein

```
git fetch
```

ausgeführt werden, um das lokale Repository mit der remote abzugleichen und anschließend ein

```
git checkout -b [neuer Branch] fn2web/[Release]
```

gemacht werden um den neuen Branch lokal an zu legen. Falls der Name des gewünschten Branches mit einem '#' beginnt, muss er in Anführungszeichen gesetzt werden.

Danach müssen nur noch die aktualisierten Dateien in das Webapplikationsverzeichnis des Servers kopiert werden. Dies kann über unsere mitgelieferten Kopierskripts (Achtung, müssen angepasst werden!), oder auch gerne durch andere Mechanismen geschehen.



Der FTP-Server ist für git nicht mehr verfügbar und wurde von einem GitLab-Server abgelöst. Deshalb muss in der Datei `/.git/config` für die fn2web-Remote die URL `https://gitlab.ihb-eg.de/fn2/fn2webapps.git` hinterlegt sein (ggf. mit Nutzer und Passwort). Zum Beispiel:

```
[remote "fn2web"]
    url = https://gitlab.ihb-eg.de/fn2/fn2webapps.git
```

für die Remote „fn2web“.



Es ist leider eine gängige Praxis bei Linux-Systemen (und in geringerem Umfang auch Windows) standartmäßig mit dem Superuser zu arbeiten. In Zeiten von Emotet und ähnlicher Schadsoftware, die auch schon einige Hochschulen befallen haben, sollte es überflüssig sein zu erläutern, wieso das generell eine schlechte Idee ist. Hier ein weiterer Grund:
Wenn mit dem root-User git-Operationen ausgeführt werden, dann werden die Änderungen im .git-Ordner auch mit dem root-User als Inhaber hinterlegt. Ein normaler Nutzer kann dann diese nicht mehr ändern ohne die git-Befehle selbst unter Superuser-Kontext auszuführen. Je nach Art der Änderungen wird das dem Nutzer aber von git nicht mit geteilt. In anderen Worten: es kann dadurch passieren, dass ein 'git pull' ohne Fehlermeldung durchläuft, jedoch keine oder nicht alle Dateien aktualisiert werden.
Daher noch einmal:

Bitte, wiederstehen Sie der 'sudo -s'-Versuchung.

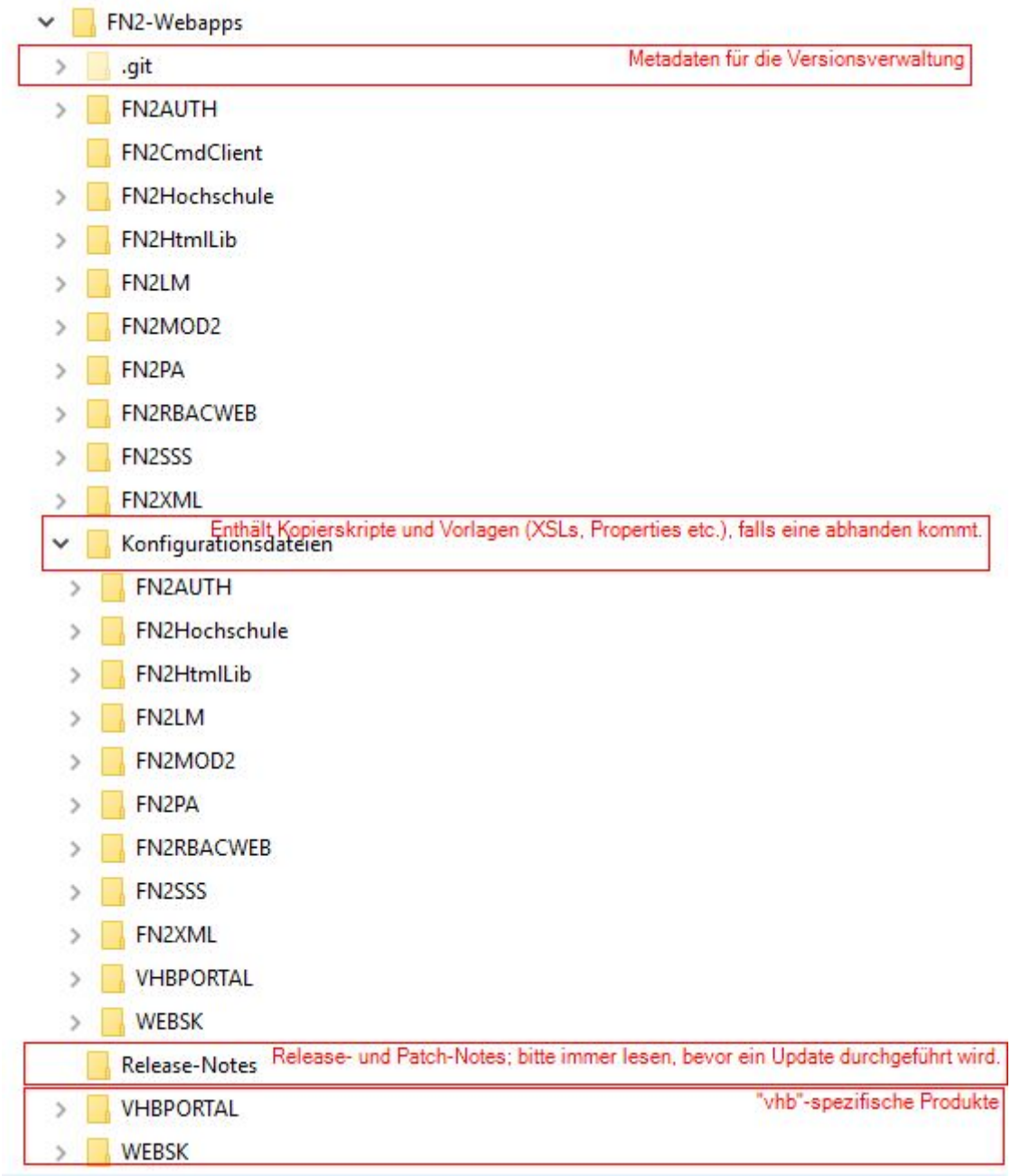
Aktualisieren der FN2-Installation

Im Ordner Konfigurationsdateien gibt es Skripte für Windows (*CopyFN2-Webapps.cmd*) und Linux (*CopyFN2-Webapps.sh* und *fn2excl.txt*). Bitte diese als Kopiervorlage verwenden und in den Skripten die gewünschten Variablen setzen bzw. die auszuschließenden Ordner als solche markieren.

Anschließend müssen alle Einträge und Änderungen in den Hochschul-spezifischen Dateien vorgenommen werden, die seit dem letzten Update hinzugekommen sind. Daher, die Release Notes müssen vom alten bis zum aktuellen Stand durchgegangen und die Änderungen übernommen werden. Da das bei vielen Änderungen ein sehr aufwändiger und zeitraubender Prozess sein kann, empfehlen wir Updates möglichst häufig aufzuspielen um Fehler zu vermeiden.

fn2Web-Releases Verzeichnisstruktur

Mit dem nachfolgenden Screenshot wird die Verzeichnisstruktur, die man beim Download von fn2Web-Releases erhält, erklärt.



Wichtige GIT-Befehle

```
git fetch --all -v
```

aktualisiert die lokalen Informationen zu den bekannten Remotes und gibt Änderungen detailiert an. Im Endeffekt kann so überprüft werden ob neue Releases und Updates veröffentlich wurden.

```
git pull
```

führt eigentlich zwei separate Befehle aus: 'git fetch' und 'git merge'.
Fetch wurde bereits beschrieben, Merge führt lokal eingepflegte (commitete) Änderungen mit denen des Remote-Branches zusammen. Wir möchten Sie bitten, darauf zu verzichten lokal Änderungen ein zu pflegen. Wir übernehmen weder Garantie, noch Support, für derartige Änderungen. Bitte sprechen Sie solche Eingriffe mit uns ab!

```
git branch
```

zeigt die lokalen Zweige/Branches an und markiert den aktuell ausgecheckten.

```
git remote -v
```

zeigt die lokal definierten Remotes und zugehörige URLs an.

```
git log
git show
```

sind beides Befehle, mit denen eine Übersicht über den Verlauf von Commits im Repository gewonnen werden kann.

Die meisten git Befehle leisten viel mehr, als hier aufgeführt werden kann. Daher sei auch hier nocheinmal auf die git-Dokumentation und ihre (sehr gute und hilfreiche) Suchfunktion verwießen.

Troubleshooting

Wenn vorhandene Dateien geändert wurden, ist folgendes durchzuführen:

Ein Update wird dann mit dem Hinweis abbrechen, dass lokale Änderungen durch das Update überschrieben würden und man diese entweder committen (was von FlexNow-Seite aus nicht erwünscht bzw möglich ist) oder diese „Stashen“ soll. In diesem Fall bitte ein Stash durchzuführen, d.h die veränderten Daten in einen Zwischenspeicher abzulegen.

```
git stash
```

Danach ist es nötig mit dem bereits erwähnten Befehl

```
git pull
```

ein Update zu holen und abschließend den Stash zu bereinigen.

```
git stash list
```

```
git stash drop stash@{0}
```

Hinweise gibt es im GIT-Buch.

Alternativ kann, vor allem bei schwereren Beschädigungen am Repository, mit dem reset-Befehl gearbeitet werden:

```
git reset --hard <commit>
```

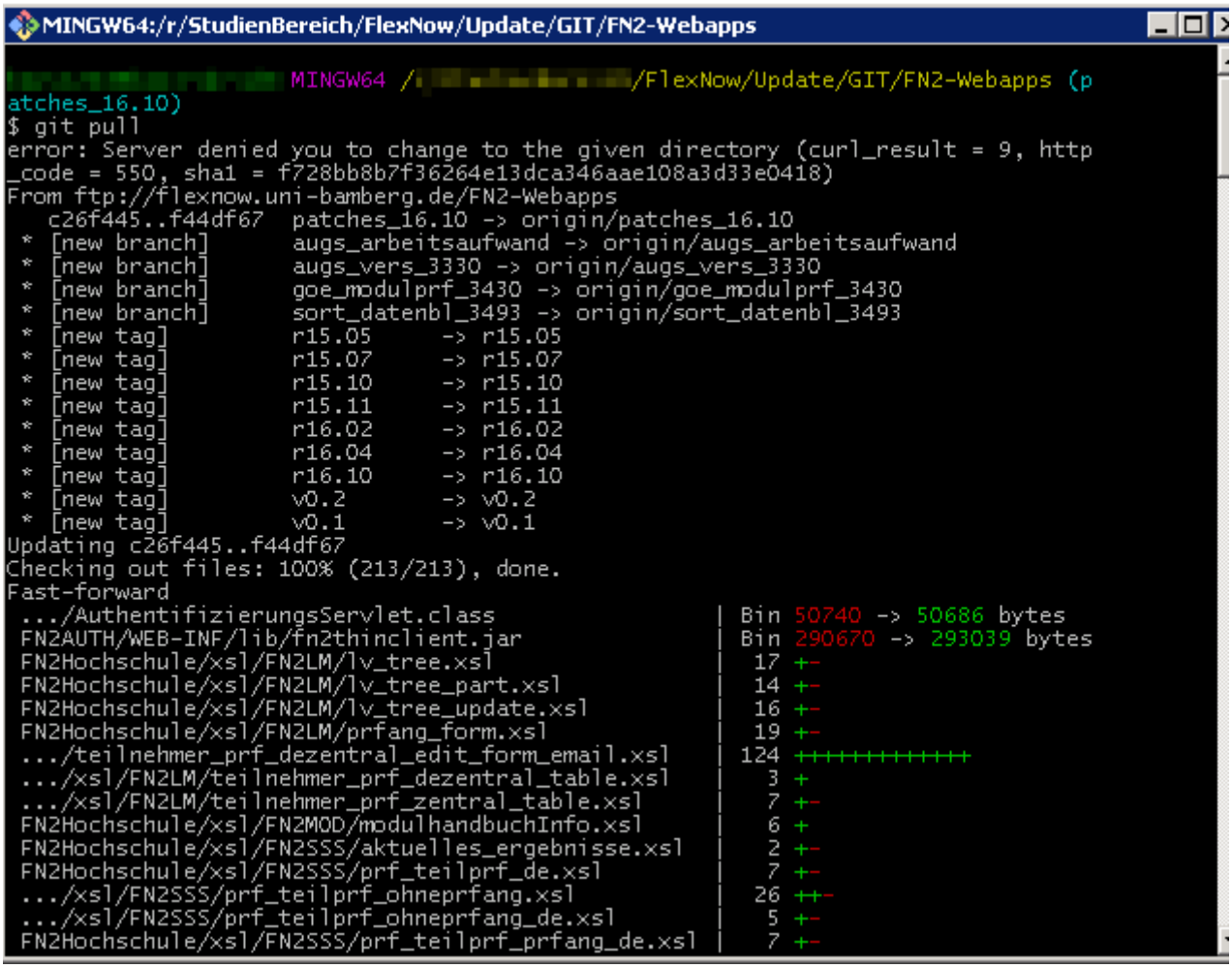
bringt sowohl den HEAD als auch den Working-Tree (also die verwalteten Dateien) auf den Stand von <commit>.

Probleme mit Sonderzeichen im Passwort

Falls ein Sonderzeichen im Passwort ist und der Zugang zum GIT-Server verweigert wird, muss das Passwort im lokalen Kunden-Repository in der Datei .../.git/config in Hochkommata gesetzt werden (Quelle)

Fehlermeldungen beim Update

Fehlermeldungen, wie der beim nachfolgenden Screenshot (*error: Server denied you to change to the given directory*) oder wie dieser dokumentiert im Ticket #2922 (*error: RETR respons: 550 ...*), können ignoriert werden. Das Update wird trotzdem ausgeführt. Überprüft werden kann das, indem entweder der Befehl *git pull* erneut eingegeben wird, diesmal mit dem Ergebnis *Already up-to-date*, oder, indem im Unterordner *Release-Notes* die aktuelle *Hinweis-Datei* geöffnet wird und geprüft wird, ob ein neuer Eintrag hinzugefügt wurde.



From:
https://wiki.ihb-eg.de/ - **FlexWiki**

Permanent link:
https://wiki.ihb-eg.de/doku.php/fn2/aktualisierung

Last update: **2024/03/04 14:54**