Jameica / JVerein / Hibiscus Setup

Aus it.jungeverlagsmenschen.de

Inhaltsverzeichnis

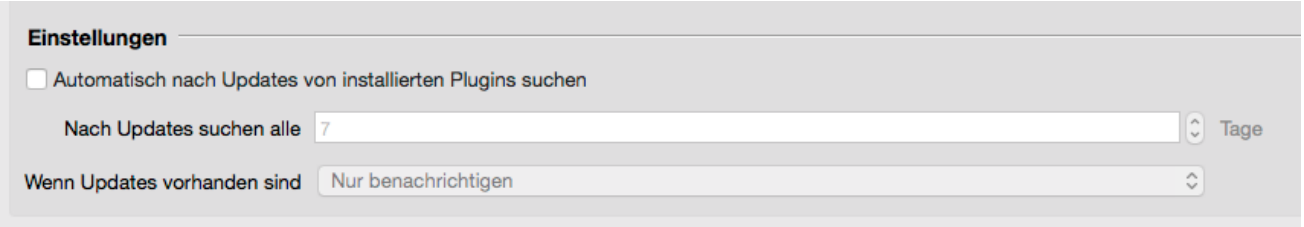
Jameica/JVerein/Hibiscus generell

JVerein besteht aus drei wesentlichen Teilen:

- Das Framework Jameica (http://www.willuhn.de/products/jameica/); dies stellt Datenbankschnittstellen, GUI-Komponenten, Messaging etc.
- Dem Homebanking-HBCI-Plugin Hibiscus (http://www.willuhn.de/products/hibiscus/), das genutzt werden kann, um die Bankgeschäfte des Vereins abzuwickeln. Wichtig: Auch wenn Hibiscus bspw. bei der Mitgliederverwaltung nicht aktiv gebraucht wird, muss es trotzdem mit installiert sein um JVerein nutzen zu können.
- Dem Vereinsverwaltung-Plugin JVerein selbst (http://www.jverein.de).

JVerein läuft ab der Java Version 1.8. Diese muss auf den Client-Rechnern installiert sein. Überprüfbar/installierbar mit: http://java.com/de/download/installed.jsp

Wichtig ist, dass bei allen Verwendern von JVerein die selbe Version von jameica und JVerein laufen muss. Daher wird zum einen die Installation von JVerein über vorkonfigurierte Installationspakete realisiert. Außerdem muss die automatische Online-Update-Funktion von JVerein und Hibiscus deaktiviert sein.



Sollte ein Update auf eine neu Jameica, Hibiscus, JVerein-Version gewünscht sein, ist sicherzustellen, dass das Update bei allen Client-Rechnern zu gleichen Zeit ausgerollt wird, da der Hersteller darauf hinweist, dass neue Versionen z.B. Datenbankstrukturen verändern können und dann der Zugriff mit alten Versionen nicht mehr funktioniert und den Datenhaushalt beschädigt.

Paketieren der Jameica-Versionen

- Download der neuen jameica-Version für das eigene Betriebssystem von der Herstellerseite in ein separates Verzeichnis z.B. Downloads. Das Plugin-Verzeichnis innerhalb des jameica-Ordners(Windows) bzw. der Jameica.app(Mac) ist noch leer.
- *Hinweis: In macOS High Sierra hat Apple mal wieder an den JRE-Pfaden gespielt. Daher muss man übergangsweise in jameica-macos64.sh den /System/Library/Frameworks/-If-Block auskommentieren*
- Ausführen von jameica und Installation von Hibiscus über die Plugin-Verwalten-Funktion. Wichtig ist die Installation in den Programme-Ordner (in diesem Beispiel das Download-Verzeichnis) und NICHT in den Benutzerordner.
- Neustart und Installation von JVerein.
- Nun sollten im Plugin-Verzeichnis beide Plugins (JVerein, Hibiscus) existieren.
- Download der neuen jameica-Versionen für die anderen Betriebssysteme
- Kopieren des Plugin-Verzeichnisses von der Version für das eigene Betriebssystem in alle anderen Betriebssystem-Versionen im Explorer (Windows) / Finder (Mac).
- Zippen aller jameica-Versionen und Upload als neue Version in Wiki
  - Datei:Jverein bundle mac.zip
  - Datei:Jverein bundle win32.zip
  - Datei:Jverein bundle win64.zip

JVerein im Multi-User-Setup

Jverein ist eine Opensource-Mitgliederverwaltungs-Applikation auf Basis von Java. Neben allen Vorzügen, die die Software bietet, ist ein großer Nachteil, dass sie nicht nativ auf Multi-User-Fähigkeiten ausgelegt ist. D.h. bedeutet, dass die Verteilung/Aktualisierung der JVerein-Datenbank bei allen Verwendern nicht nativ unterstützt wird. Dabei ist auch zu beachten, dass die JVerein-Datenbank wie eine Binärdatei zu behandeln ist, d.h. ein schreibender Zugriff darf immer nur durch einen Verwender erfolgen.

Um dies sicherzustellen, sind generell mehrere Setups denkbar. Der Hersteller bzw. die Community um JVerein schlagen dazu mehrere Varianten vor:

- http://www.jverein.de/wiki/index.php?title=Multiuser
- http://www.jverein.de/forum/viewtopic.php?f=4&t=1085

Um ein passende Lösung auswählen zu können, sollte man sich die Anforderungen der JVM an ein Multi-User-Setup bewusst machen:

- Es gibt mehrere Personen, die schreibenden Zugriff benötigen, da die Aufgaben der Mitgliederverwaltung, der Kasse/Schatzmeister auf mehrere Personen verteilt sind
- Darüber hinaus gibt es mehrere Personen, die Nur-Lese-Zugriff z.B. für umfangreiche Auswertungen benötigen
- Der Zugriff auf die Daten muss personalisiert erfolgen und für einzelne Personen leicht entzogen werden können (Ausrollen eines Master-Passworts sollte vermieden werden)
- Der Zugriff auf die Daten muss aus mehreren Städten in Deutschland (und außerhalb) funktionieren
- Die Daten müssen mehrmals die Woche verteilt werden, da Änderungen mehrmals die Woche vorgenommen werden
- Es muss sicher gestellt sein, dass immer nur eine Person zu einem Zeitpunkt schreibenden Zugriff auf die JVerein-Datenbank hat ([[1] (https://de.wikipedia.org/wiki/Versionsverwaltung#Lock_Modify_Write%7CLock-Modiy-Write-Workflow)])
- Es soll einfach möglich sein alte Stände der Datenbank wiederherzustellen. Dabei soll nachvollziehbar sein, welcher Anwender welchen Stand der Datenbank erzeugt hat
- Die Daten sollen vor allem aus Gründen des Datenschutzes (personenbezogene Daten wie Adressen, Geburtstage, Teilnahme an Veranstaltungen etc) aber auch zum Schutz der Daten des Vereins (Buchführung wird genutzt) nicht irgendwo (Cloud) gespeichert sein, sondern bei einem Hoster mit Serverstandort in Deutschland

Aktuelles Setup: Versionsverwaltung mit SVN

Zum besseren Verständnis hier eine kurze Exkurs zu SVN-Begriffen.

Die Versionsverwaltung der JVerein-Datenbank inkl. der sequentiellen Zugriffskontrolle ist mit Hilfe von Subversion(SVN) gelöst. Die JVerein-Datenbank (jverein.h2.db) liegt in einem zentralen Repository mit dem Namen "jverein".

Um sicherzustellen, dass eine Bearbeitung der jverein.h2.db immer nur durch einen Anwender erfolgen kann, ist auf der jverein.h2.db die Eigenschaft(Property) svn:needs-lock gesetzt.

Nachteile dieser Lösung sind:

- Die Anwender müssen sich mit dem SVN Lock-Modify-Update-Workflow vertraut machen und einen SVN-Client entsprechend bedienen können
- Es gibt keine Workflow-Integration, d.h. die Anwender müssen jameica und den SVN-Client unabhängig von einander bedienen und z.B. das vollständige Beenden von jameica abwarten
- Die jverein.h2.db mit allen Daten (Mitgliederdaten, Buchführung) liegen in einer ungesicherten Datenbank im Netz. Der Zugriff ist nur durch die Zugriffsteuerung des SVN geschützt.

Benutzerverwaltung

Jeder Verwender von JVerein wird als Benutzer im SVN-Repository angelegt. Dabei wird die Konvention ein oder zwei Buchstaben des Vornamens + alle Buchstaben des Nachnamens verwendet. Umlaute müssen ersetzt werden. Bei der Vergabe der Passwörter sollte auf sichere Passwörter geachtet werden. Am besten sollte dazu ein Passwort-Generator wie in KeePassX oder ähnlichem enthalten verwendet werden. Wurde das Passwort vom Benutzer vergessen, kann dies in der Benutzerverwaltung geändert werden.

Der Zugriff auf das Repository kann für jeden Benutzer einzeln als Lesen+Schreiben (Mitgliederverwaltung, Schatzmeisterin) oder Nur-Lesen (Vorstand) eingerichtet werden. Dafür wurden dem entsprechend zwei Gruppen angelegt: jverein_write und jverein_read.

Die Benutzer müssen einer diesen beiden Gruppen zugewiesen werden.

Soll einem Benutzer der Zugriff auf die JVerein-Datenbank entzogen werden, wird sein Benutzer aus den Zugriffsrechten für das Repository entfernt.

Nicht verwendet

Andere Versionsverwaltungen

Am besten wäre es gewesen auf ein zusätzlichen Hoster verzichten zu können. Problem dabei war aber, dass ein Lock-Modify-Write-Workflow (https://de.wikipedia.org/wiki/Versionsverwaltung#Lock_Modify_Write) nur von zentralen Version-Control-Systems (VCS) unterstützt werden. Dezentrale VCS ohne zentralen Server wie GIT, Mercurial oder ähnliche sind dafür nicht ausgelegt. Andere VCS (Boar, Bazaar) wiederrum unterstützen entweder den Locking-Mechanimus nicht oder brauchen wie SVN einen zentral laufenden Server-Prozess. Dieser wird aber von unserem Provider trotz SSH-Zugang verständlicher Weise nicht unterstützt.

Daher blieb am Ende nur die Variante ein Repository auf einem eigenständigen Subversion-Server anzumieten.

Eigenentwickelte Versionsverwaltung JH2MU

Um eine Versionsverwaltung der JVerein-Datenbank zu realisieren und sicherstellen zu können, dass immer nur ein Anwender schreibend auf die zentrale JVerein-Datenbank zugreift, wurde das Programm JH2MU (JVerein H2DB Multi-User) als Eigenentwicklung in Java (Java-Fx GUI) realisiert. Vorteil dieser Lösung war, dass keine eigenständiger Provider für das Versionskontrollsystem benötigt wurde, sondern die Daten auf dem bereits bestehenden Server des JVM-Webhosting-Providers gespeichert wurden.

Weitere Features von JH2MU waren:

- Download der jeweils aktuellen Version der jverein.h2.db über SSH/SFTP in den Benutzerordner des Anwenders und Upload der geänderten jverein.h2.db nach dem Beenden von JVerein
- Integration in den Workflow: JH2MU hat automatisch nach dem Download der JVerein-Datenbank jameica gestartet und gewartet bis jameica beendet wurde. Danach folgte vollautomatisch ein Upload.
- Sicherstellung, dass jeweils nur ein schreibender Zugriff durch einen Anwender erfolgt durch Umsetzung eines Lock-Modify-Update-Workflows
- Speicherung der Daten als verschlüsseltes ZIP-Archiv auf dem Server
- Vorhalten von alten Versionen der zentralen Version der jverein.h2.db (Backup-Zyklus: die letzten 5 + jeweils eine Version eines Anwenders).

Aufgrund des Wartungsaufwandes (Anpassen an neue Java-Versionen) und häufig auftretenden Datenbanksperrungen (z.B. bei Programm-/Rechnerabstürzen) wurde diese Lösung zu Gunsten der SVN-Lösung aufgegeben.

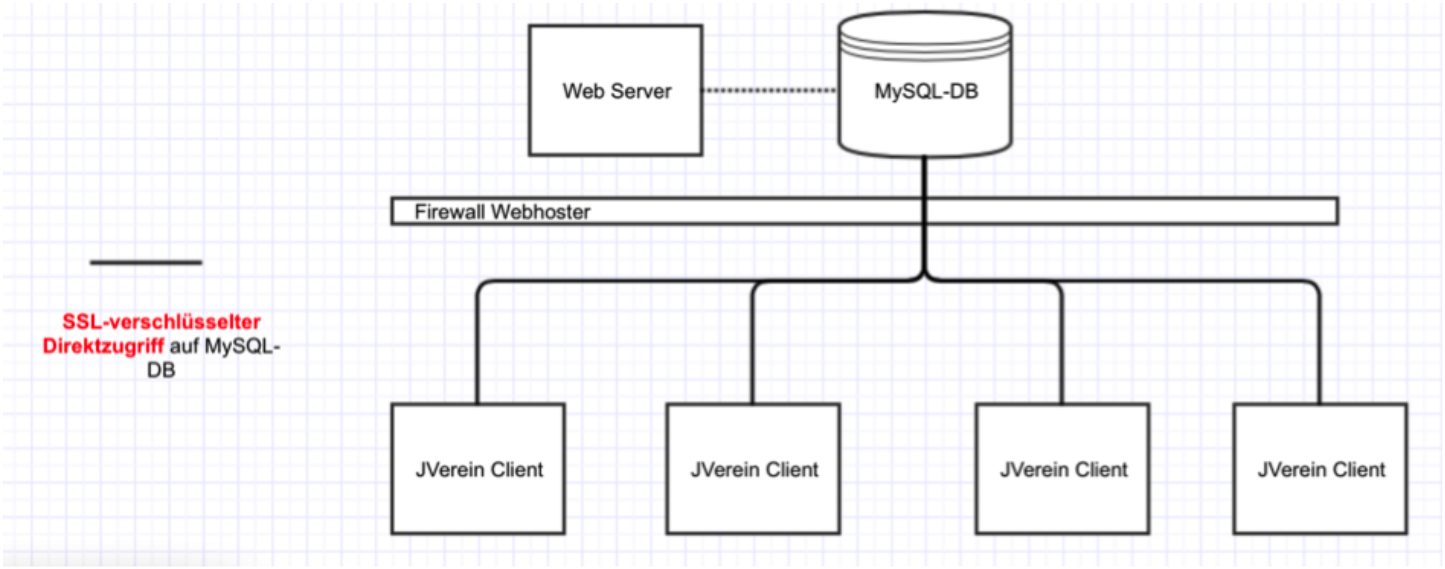
Austausch der JVerein-Datenbank über Dropbox & Co

Darauf wurde aus folgenden Gründen verzichtet:

- Damit kann nicht sichergestellt werden, dass nicht zwei Anwender parallel an der Datenbank arbeiten und das zu zwei konfligierenden Versionen kommt
- Es besteht die Gefahr, dass der Upload der geänderten Datenbank im Hintergrund durch Dropbox nicht abgeschlossen wird, weil z.B. der Rechner zu früh ausgeschaltet wird. Die geänderten Daten werden dann von den anderen Nutzern nicht bemerkt und die Änderungen gehen durch den entstehenden Konflikt verloren.
- Bedenken zum Datenschutz: Es ist nicht immer klar, wo genau die Datenbank von Dropbox & Co gespeichert werden

Zentrale MySQL-Datenbank

JVerein kann generell auch auf einer zentralen MySQL-Datenbank betrieben werden (http://www.jverein.de/wiki/index.php?title=MySQL-Support). Dieses Setup wurde auch getestet und würde bei Nutzung einer MySQL-Datenbank von all-inkl.com so aussehen:



Dabei treten aber folgende Probleme auf:

- Es kann bei gleichzeitigem Arbeiten zu intransparenten Datenkonstellationen kommen: Auf der JVerein-GUI augenscheinlich gespeicherte Eingaben sind noch nicht in die Datenbank geschrieben und daher für andere Nutzer nicht sichtbar. Teilweise erfolgt die Speicherung der Daten in der Datenbank erst beim Beenden von JVerein selbst. Änderungen von einem Nutzer überschreiben daher unter Umständen Änderungen eines anderen Nutzer. Daher sollte es möglichst vermieden werden gleichzeitig auf Datenbank zu arbeiten.
- Dabei ist zu beachten, dass die Datenbank direkt (d.h. ohne Umweg z.B. über einen Webserver) aus dem Internet erreichbar sein muss. Deswegen ist es zweitens unumgänglich, dass auf den MySQL-Server nur verschlüsselt zugegriffen werden darf. Dafür gibt es zwei Möglichkeiten:
  - Zugriff über eine direkte SSL-Verbindung zum MySQL-Server (siehe z.B. hier https://www.thomas-krenn.com/de/wiki/MySQL_Verbindungen_mit_SSL_verschl%C3%BCsseln). Diese Option bietet aber der MySQL-Server von all-inkl.com nicht an: "have_openssl -> DISABLED und have_ssl -> DISABLED".
  - Zugriff über eine verschlüsselte Verbindung mit MySQL-Boardmitteln über den Parameter use:encryption. Die Performance-Test waren dabei durchgehend negativ. Die Abfrage aller Mitglieder dauerte über 30 Sekunden.

Hinweise zur Konfiguration: Die Einstellung der Verbindungsparameter zur zentralen Datenbank pro Benutzer erfolgt über die Datei cfg/de.jost_net.JVerein.rmi.JVereinDBService.properties. Dort ist neben der Angabe des jeweiligen Benutzers auch die SSL-Verschlüsselung zu konfigurieren. Dabei ist die Angabe des Pfads zum Truststore notwendig, in dem das Server-Zertifikat des MySQL-Servers abgelegt sein muss. Wichtig der JDBC-MySQL-Treiber unterstützt die Angabe eines Truststores erst ab Version 5.1.0.

Migration von Alt-Daten nach JVerein

Hinweis: Bei Benutzung der Funktion „Jverein->Administration->Import“ werden ALLE bestehenden Mitglieder, deren zugehörige Daten und Eigenschaftsgruppen gelöscht.

Der Import erfolgt über eine csv-Datei. Die Dokumentation (teilweise veraltet; s. Pflichtfeld Mandat für SEPA) gibt es unter: http://www.jverein.de/wiki/index.php?title=Import#Anf.C3.A4ngerfehler_beim_Importieren

Wichtig: Damit alle (zusätzlichen, nicht im Standard von JVerein enthaltenen) Spalten übernommen werden, müssen diese zuerst in Administration->Felddefinitionen angelegt werden.

Die letzte funktionierende Version ist eine angepasste (Spaltennamen, erlaubte Spalten) Version der vorhergehenden JVM-Mitgliederliste. Dort sind auch alle Einträge gelöscht die nicht automatisiert importiert werden können, da Pflichtfelder wie Mitgliedsnummer und Eintrittsdatum nicht vorhanden sind.

Hier ist ein Beispiel dieser Liste.

Zusätzlich ergänzte Felder:

Name	Label	Datentyp	Länge
Verlag	Verlag	Zeichenfolge	255
Bereich	Bereich	Zeichenfolge	255
Position	Position	Zeichenfolge	255
Email2	Email2	Zeichenfolge	255
Ort2	Ort2	Zeichenfolge	255
Staedtegruppe	Städtegruppe	Zeichenfolge	255
Mitgliedsanfrage	Mitgliedsanfrage	Datum	10

Durchzuführende Schritte vor Import:

- Umwandlung der Excel-Datei in eine csv-Datei. Beachte dabei die richtige Kodierung (Stichwort UTF-8).
- Suchen und ersetzen aller Semikola durch anderes Zeichen z.B. Komma
- Export über Speichern unter ... als Windows-CSV.
- Ersetzen aller Leerzeichen vor Semikola -> Suchen/Ersetzen in Texteditor „ ;“ durch „;“ -> mehrmals durchführen, da mehrere Leerzeichen möglich!
- Ersetzen aller Leerzeichen nach Semikola -> Suchen/Ersetzen in Texteditor „; “ durch „;“ -> mehrmals durchführen, da mehrere Leerzeichen möglich!

Danach ausführen des Imports in JVerein. Die Felder aus der csv-Datei sollten automatisch richtig zugeordnet werden. Bei Fehlern Fehlermeldungen beachten und Fehlerlog überprüfen

Hinweis: Nach erfolgreichem Import sollte die Bezeichnung der Eigenschaften-Gruppe „Eigentschaft_Mitgliedart“ auf Mitgliedschaftsart geändert werden.
