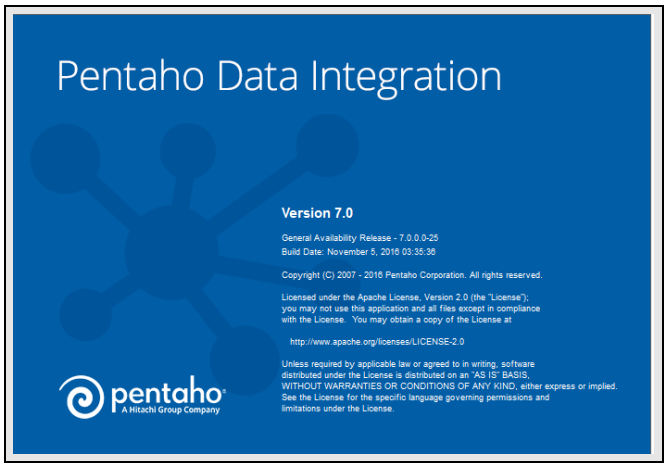
Inhaltsverzeichnis

Inhaltsverzeichnis 1
Kettle für Fortgeschrittene - Monitoring, Embedding 2
Kursbeschreibung 2
Monitoring/Logging von Kettle Jobs 3
Mailversand in Kettle 3
Mailversand Beispiel 1 3
Mailversand mit Variablen 5
Mailversand mit ausgelagerten Variablen 6
Loglevel und -dateien 7
Versionierung von Jobs 9
Entfernen der Connection-Elemente 9
Automatisierung mit Groovy 10
Jobmanagement 12
Eigenes BI-Modul 12
Ladejob-Maske erstellen und ausführen 13
Tomcat Konfiguration 14
Job registrieren 14
Ladejob-Masken erstellen 15
Ladejob-Masken nutzen 17
Kitchen 17
Home-Verzeichnis 17
Umgebungsvariablen 18
Kettle.properties 18
Shared.xml 18
Shared.xml in SuperX bzw. HISinOne-BI 19
Ausführung, Logging und Fehlermeldung 19
KETTLE_ENV einrichten 19
Ausführungsskript inkl. Mailversand 19
Parameter übergeben 20

Kettle für Fortgeschrittene - Monitoring, Embedding



Kettle für Fortgeschrittene - Monitoring, Embedding

Universität Wuppertal, 8.12.2022

Kettle / Pentaho ist ein Produkt von Hitachi Vantara

Kursbeschreibung

Der Online-Kurs führt Inhalte aus der Kettle-Grundlagenschulung fort und adressiert folgende Themen

- Monitoring von Kettle Jobs
  - Mailversand
  - Logging und Fehlerbehandlung
- Embedding Kettle
  - in HISinone-BI
  - Standalone unter Linux
- Versionierung von Quellcodes

Download Kursmaterial:

- Per Browser: https://superx-rocks.de/git/Memtext/Kettle-Schulung
- Per git:

```
git clone https://superx-rocks.de/git/Memtext/Kettle-Schulung.git
```

Nach dem Download Klonen laden Sie Ihre SQL_ENV, und führen aus:

```
./rsync_to_h1.x
```

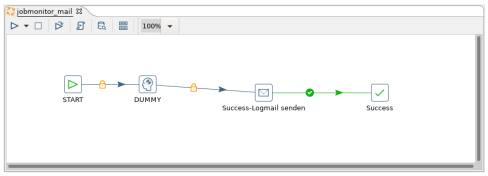
Damit wird das Modul in Ihrer BI eingespielt.

Monitoring/Logging von Kettle Jobs

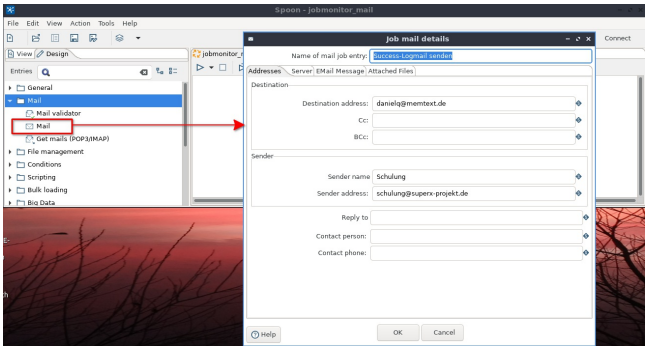
Mailversand in Kettle

Mailversand Beispiel 1

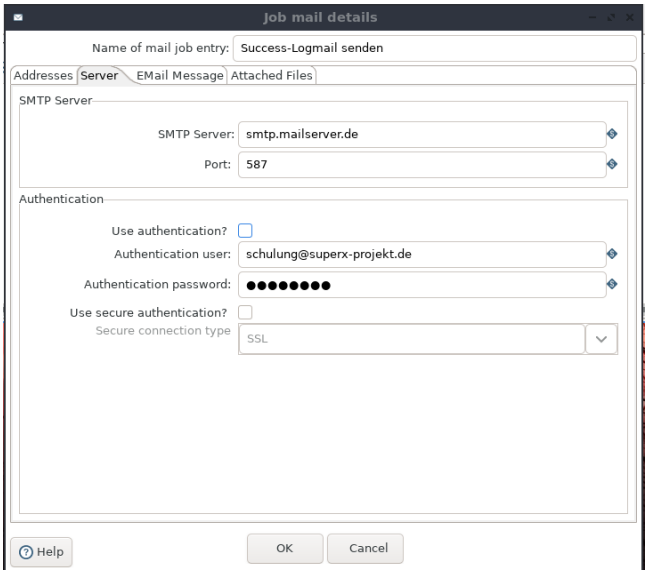
Unser erster Beispieljob versendet einfach nur eine Mail. Hier eine Gesamtübersicht der Schritte im Job:



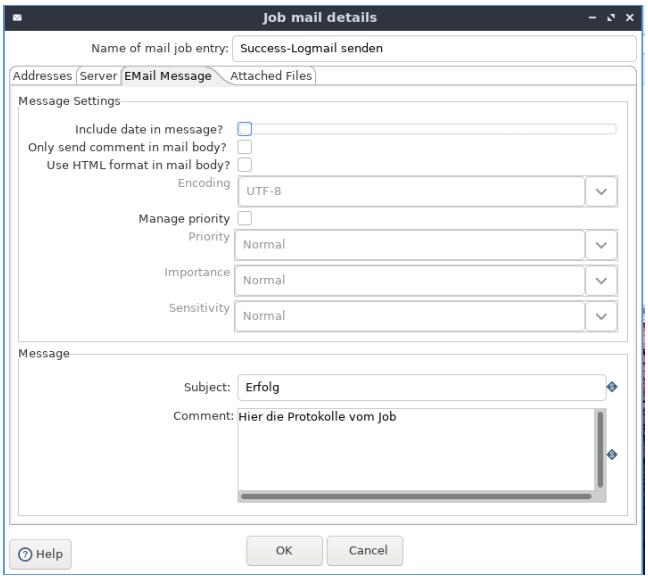
Der Job führt nur einen Dummy-Step aus, und verschickt dann eine Mail. Dazu gibt es im Reiter "Design" einen Step:



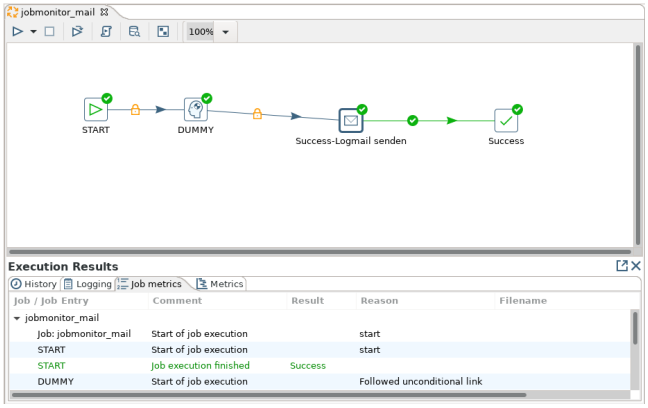
In dem Dialog im Reiter "Adresses" spezifizieren Sie die Absender- und Empfängeradresse. Danach geben Sie Daten zum Mailserver an:



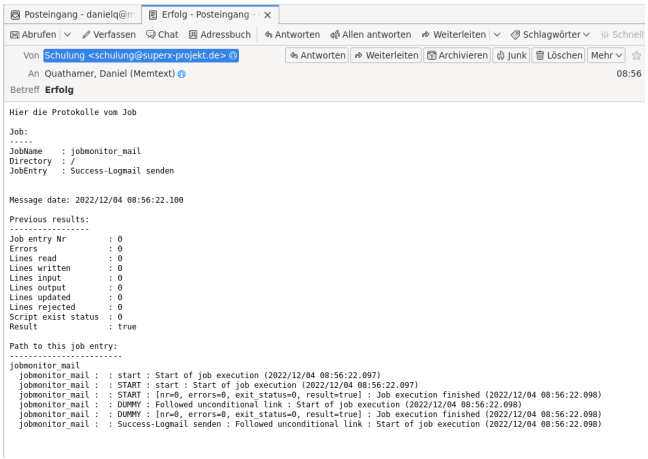
Unser Beispiel-Mailserver verlangt Zugangsdaten zum Mailversand, diese tippen wir einfach ein. Danach wird Betreff und Inhalt der Mail definiert:



Wir versenden eine einfache Testmail mit voreingestellten Kettle-Protokolldaten. Dann starten wir den Job:



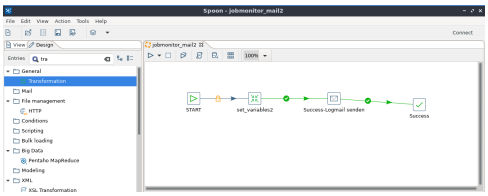
Der Job war erfolgreich, und es müßte tatsächlich eine Mail ankommen:



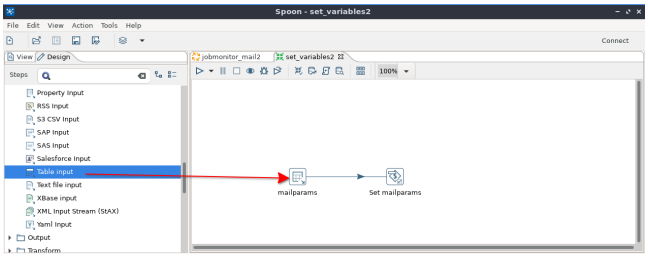
Damit haben wir ein einfaches Beispiel zum Mailversand abgeschlossen.

Mailversand mit Variablen

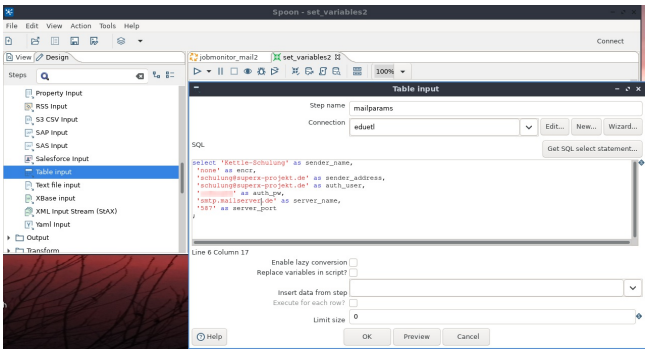
Im obigen Beispiel haben wir die Angaben zum Mailversand "hartcodiert" in den Dialog eingetragen. Da es vermutlich mehrere Jobs mit Mailversand gibt, ist es sinnvoll die immer wiederkehrenden Angaben zum Mailserver etc. in Variablen auszulagern. Wir verfeinern also obiges Beispiel, indem wir dem Mailversand eine Transformation vorschalten, die die Variablen definiert und setzt. Hier eine Gesamtübersicht des Jobs:



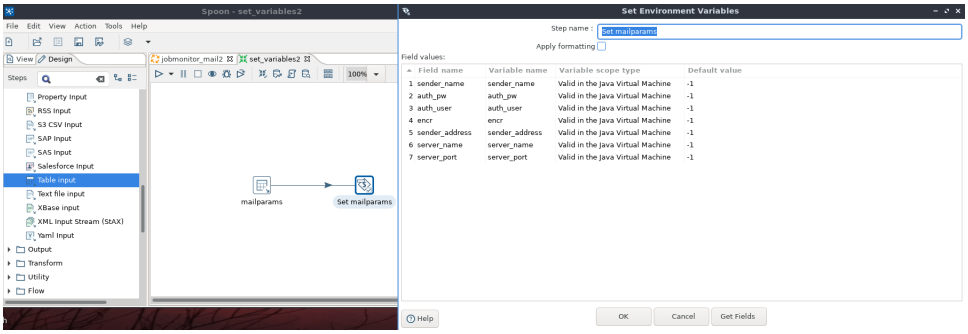
Da in der Regel solche Angaben in einer Datenbank vorgehalten werden, wählen wir in der Transformation einen Table-Input Step:



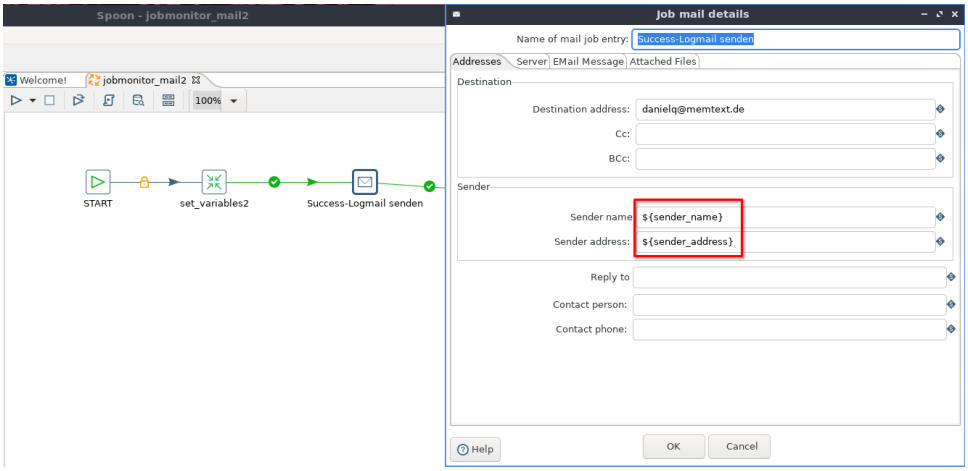
Der Step selektiert die jew. Parameter, das Beispiel funktioniert unter Postgres und kann so leicht auf die eigene DB angepaßt werden.



Die Parameter werden dann über den "Set Variables"-Step zur Laufzeit im gesamten Job zur Verfügung gestellt:



Der folgende Dialog zum Mailversand wir dann statt "hartcodiert" mit Verweisen auf die Variablen versehen:

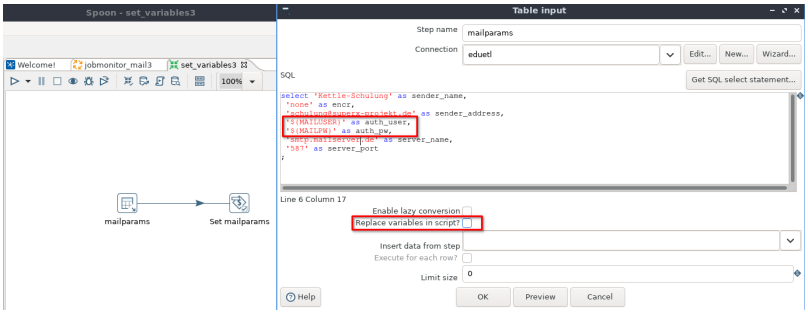


Im folgenden Reiter zum Mailserver sind die Angaben dann analog vorzunehmen. Der Job sollte danach genauso funktionieren wie das erste Beispiel.

Mailversand mit ausgelagerten Variablen

Im obigen Beispiel haben wir "sensitive" Angaben zum Mailversand (Benutzername, Passwort) "hartcodiert" in die Datenbank geschrieben. Hier ist es sinnvoll, diese Angaben aus Job und Datenbank herauszunehmen. Kettle bietet die Möglichkeit, zentrale bzw. systemspezifische Variablen in eine Datei "kettle.properties" auszulagern, Details dazu siehe unten.

Das verschlüsselte Passwort und die Zugangskennung wird dann als Variable übergeben.



Das Passwort-Feld ist maskiert, die Eingabe ist

${MAILPW}

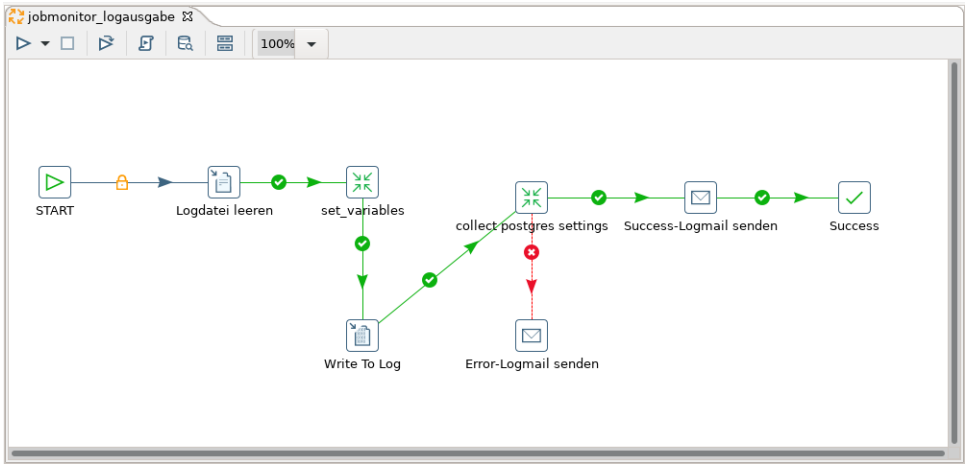
Bei einer Änderung der kettle.properties müssen Sie Spoon neu starten.

Loglevel und -dateien

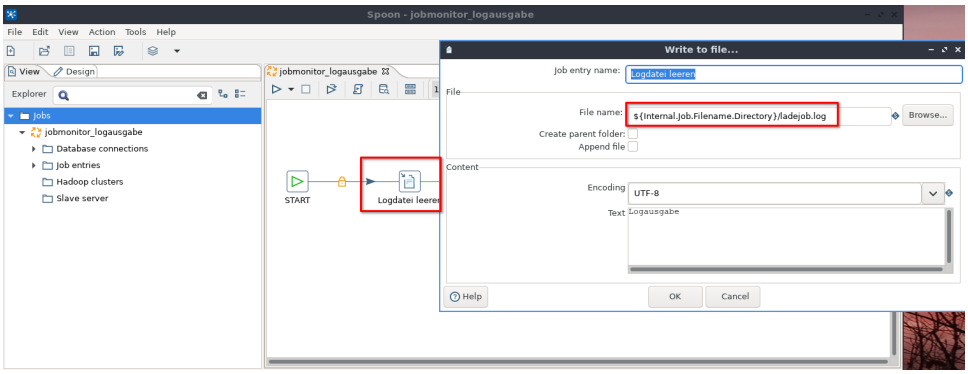
Kettle bietet diverse Möglichkeiten zum Logging und zur Fehlerbehandlung:

- Logausgabe von Spoon / Kitchen
  - In Jobs kann man diese verfeinern mit dem "Write to log"-Step
- Eine Transformation kann
  - die Logausgabe in einer dezidierten Logdatei speichern bzw. erweitern
  - im Aufruf eines Jobs in eine Fehlerbehandlung laufen

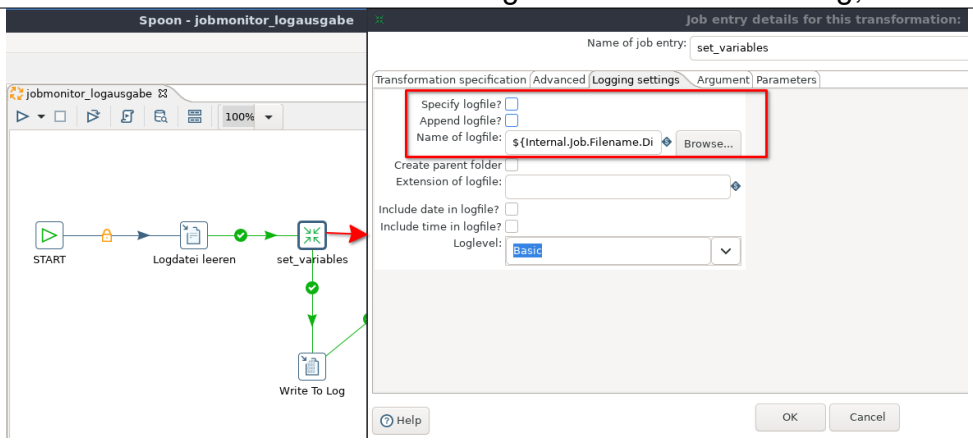
Das folgende Beispiel zeigt die Nutzung, im Job wird eine Logdatei angelegt, und dann schreiben die einzelnen Transformationen in diese Logdatei. Diese wird dann am Ende per Mail als Anhang verschickt - egal ob es mit Fehler oder erfolgreich war:



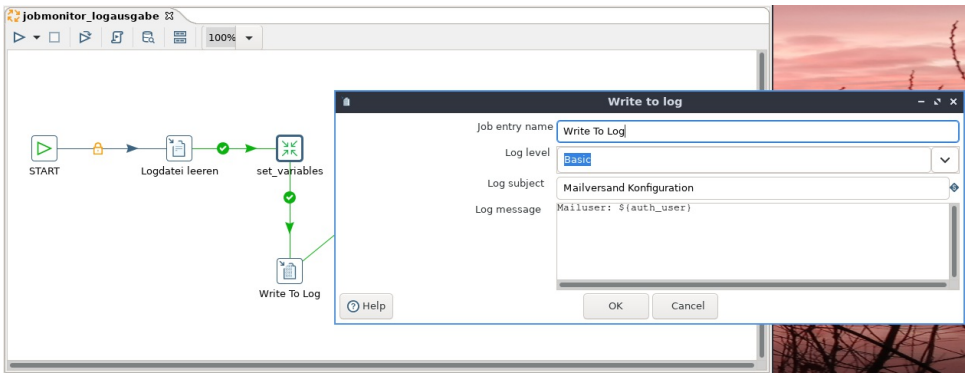
Im ersten Schritt wird die Logdatei "ladejob.log" angelegt bzw. geleert, wenn sie schon existiert:



Die erste "Set variables"-Transformation wird im Reiter "Logging" so konfiguriert, dass die Ausgabe in die Logdatei "ladejob.log" geloggt wird.

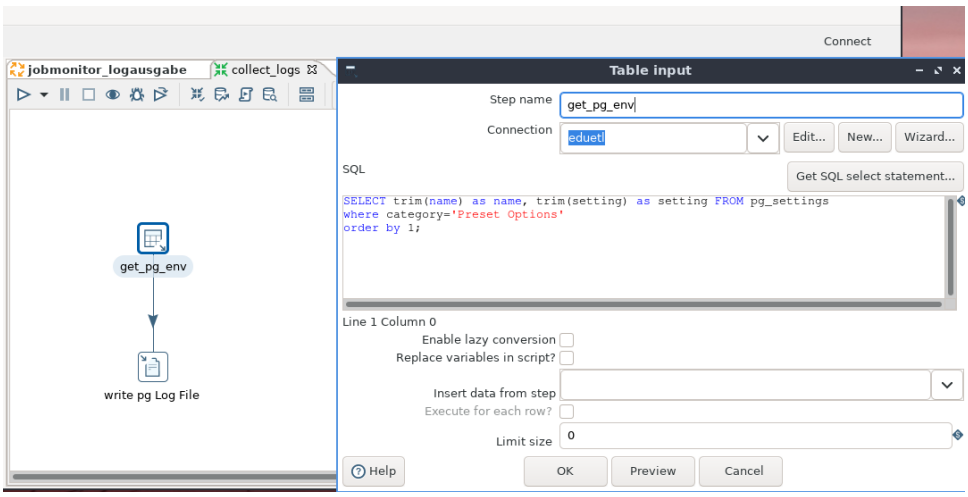


Ergänzend wollen wir auch die gesamt-Logausgabe des Jobs um eine Zeile erweitern, die die übergebene Mailkennung aus der kettle.properties ausgibt:

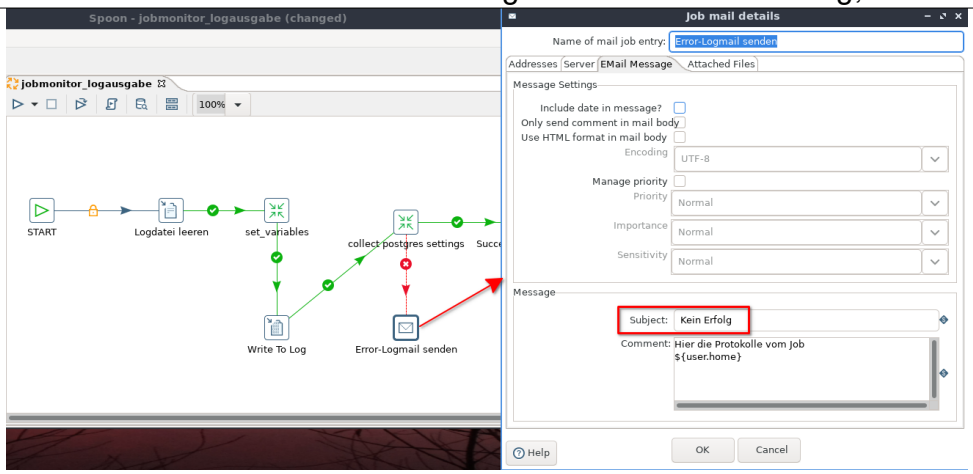


Danach wird eine Transformation erstellt, die Postgres-spezifische Systemvariablen

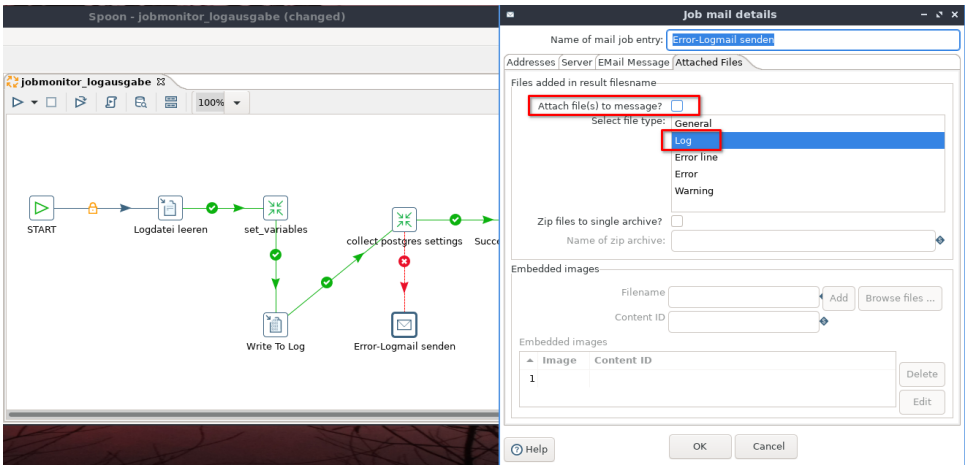
- selektiert und
- das Ergebnis in die Logdatei schreibt



Da diese Operation aus der Datenbank liest und ggf. Fehler bewirken kann, wird neben der normalen Erfolgs-Ausgabe ein "Fehlerkanal" eingerichtet, der eine entsprechende Fehler-Mail verschickt:



Im Reiter "Attached files" wird die Logausgabe des Jobs angehängt:



Unser Job arbeitet von nun an mit einer Fehler- und Logbehandlung.

Mit den "Logleveln" können Sie wählen wie detailliert geloggt wird. Möglich sind:

- Nothing
- Error
- Minimal
- Basic (dies ist der Default)
- Detailed
- Debug
- Row level (Achtung: erzeugt viele Logeinträge)

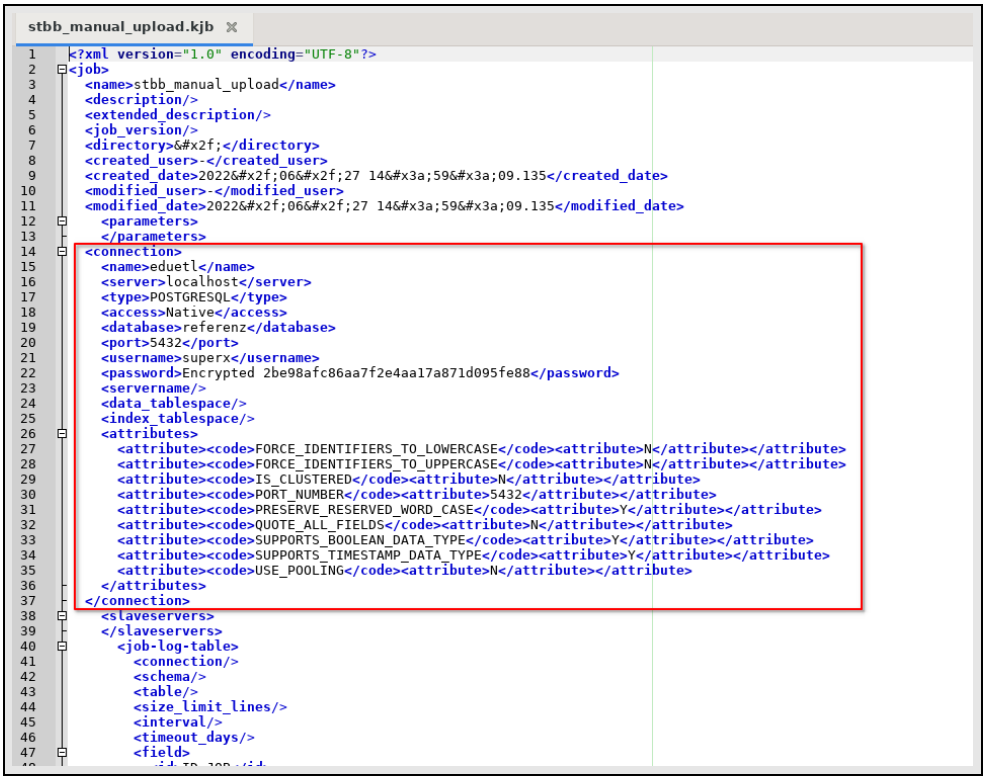
Versionierung von Jobs

Kettle Jobs und Transformationen sind XML-Dateien und lassen sich mit Versionskontrollsystemen (z.B. git) versionieren. Es gibt allerdings ein paar Fallstricke:

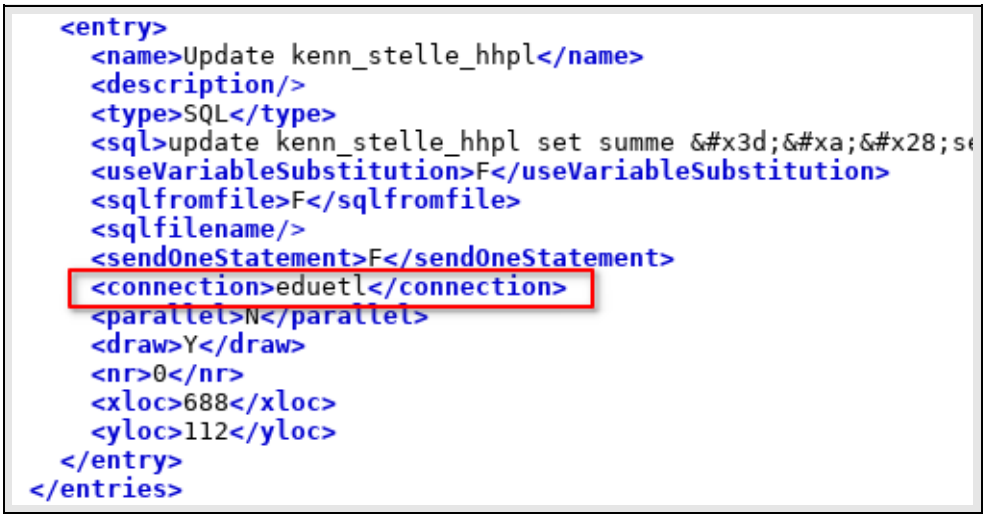
- Kleinste Layoutänderungen in Spoon führen mitunter zu großen Änderungen im Quellcode, so dass Sie solche Änderungen eher in separate Commits trennen sollten, und den Layout-Comment in der Commit Message z.B. als "reine Layout-Änderung" kennzeichnen.
- Die Connection-Angaben müssen entfernt werden. Wie das geht zeigen wir nun.

Entfernen der Connection-Elemente

Beim Speichern eines Jobs wird immer die Connection mitgespeichert. Daher muss sie vor dem Commit ins git gelöscht werden. Hierzu ist der gesamte Tag sowohl im Job, als auch in den Transformationen zu löschen.



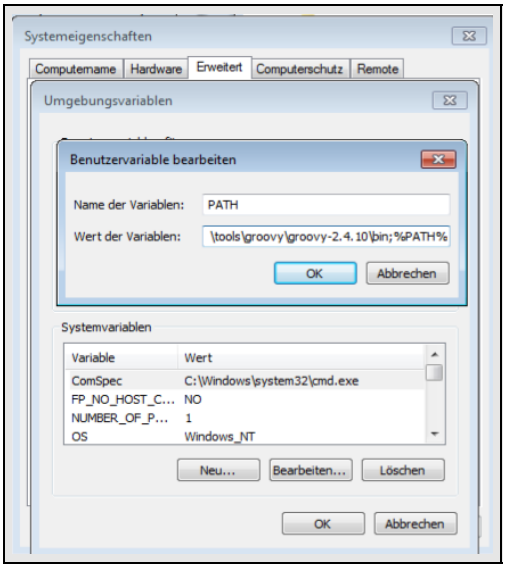
Die Connection-Information in den einzelnen Schritten bleibt jedoch enthalten.



Automatisierung mit Groovy

Sie können mit dem Java-Tool Groovy und einem von uns mitgelieferten Script die Connection-Elemente automatisch entfernen. Gehen sie dazu wie folgt vor:

- Stellen Sie sicher dass am Arbeitsplatz-Rechner Java installiert ist
- Laden Sie Groovy herunter, und entpacken Sie es in einem Ordner Ihrer Wahl (am besten ohne Leerzeichen im Pfadnamen).
- Fügen Sie den bin-Pfad der Groovy-Installation in Ihren PATH an,
  - unter Windows z.B.:
    - in der Systemsteuerung -> Erweiterterte Einstellungen -> Umgebungsvariablen -> PATH



- unter Linux z.B.:

```
PATH=$PATH:~/tools/groovy/groovy-2.4.10/bin
export  PATH
```

- Wechseln Sie in der Shell (DOS oder Linux) in das Verzeichnis, wo die ETL-Jobs liegen (hier also /superx/superx/WEB-INF/conf/edustore/db/module/myjobs/etl)
- Legen Sie dort einen Unterordner an mit dem Namen einer Kopie des ETL-Verzeichnisses , hier also z.B. "jobmonitor_git".
- Dann führen Sie das Script aus

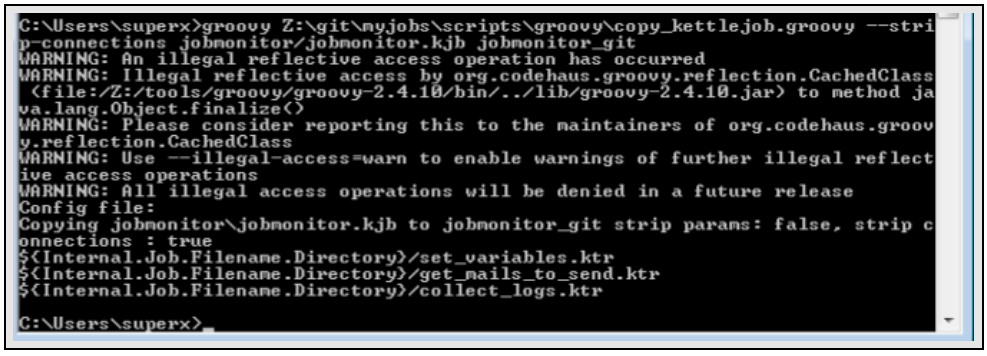
Unter Linux:

```
groovy  ~/git/myjobs/scripts/groovy/copy_kettlejob.groovy  --strip-connections  jobmonitor/jobmonitor.kjb  jobmonitor_git
```

Unter Windows:

```
groovy  Z:\\git\\myjobs\\scripts\\groovy\\copy_kettlejob.groovy  --strip-connections  jobmonitor\\jobmonitor.kjb  jobmonitor_git
```

Damit werden alle Connection Elemente in dem Job und in den darin aufgerufenen Transformationen entfernt:



Wenn das geklappt hat, können Sie die bereinigten Dateien in den "richtigen" Ordner "jobmonitor" kopieren und versionieren.

Jobmanagement

Im folgenden zeigen wir wie Sie Jobs "on demand" direkt aus HISinOne oder SuperX im Browser vom Nutzer starten lassen.

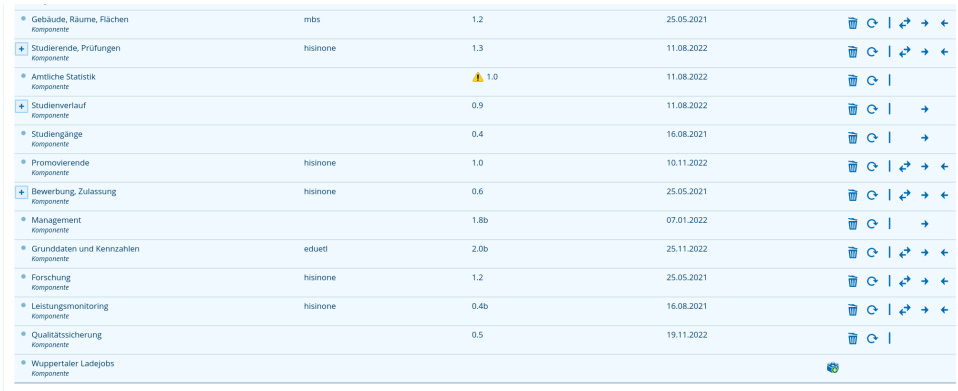
Achten Sie darauf, dass Ihre lokale genutzte Version zu der in HISinOne-/SuperX integrierten Version paßt. Sonst könnte es ggf. Probleme geben.

Software	Version	Kettle Version
HISinOne-BI	ab 2021.06	Kettle 8.3
SuperX	ab Kernmodul 4.9	Kettle 6.0

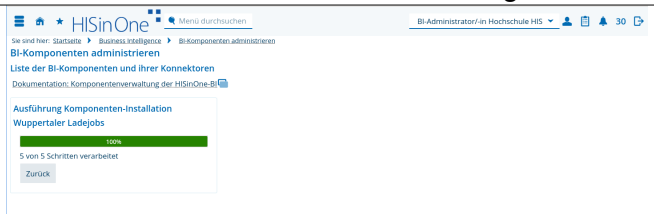
Eigenes BI-Modul

Ein Kettle Job ohne Eingabeparameter kann in einem eigenen "Mini"-Modul deklariert und über das HISinOne-Jobmanagement ausgeführt werden.

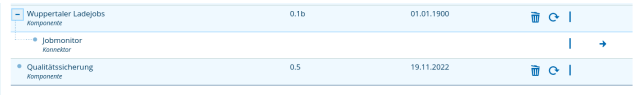
1. Erstellen Sie ein minimales Modul mit einer Modul-XML-Datei
   - Siehe z.B. https://superx-rocks.de/git/Memtext/Kettle-Schulung
2. In der Modul-XML-Datei im Bereich ETL können Sie Ladejobs deklarieren
3. Das Modul können Sie auf eine SuperX- oder BI-Installation synchronisieren oder die zip-Datei dort unter $SUPERX_DIR bzw. webapps/superx entpacken
4. Danach ist es in der BI-Komponentenverwaltung sichtbar:



Mit Klick auf das "Installations-"Icon wird es installiert



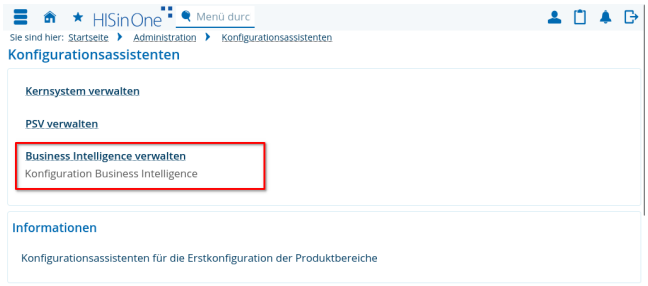
und danach ist es auch in der Konnektorenübersicht sichtbar, inkl. Unter-Ladejobs:



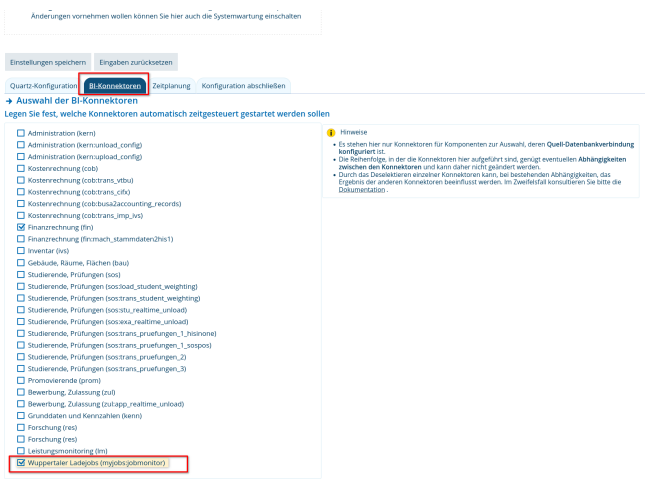
Die Jobs können dann über das BI Jobmanagement ausgeführt werden.

Auch ein zeitgesteuertes Ausführen ist möglich , per Kommandozeile.

Über die Browser-Oberfläche ist ebenfalls ein zeitgesteuerter Update möglich. Gehen Sie dazu in die Admin-Rolle, und wählen das Menü Administration -> Konfigurationsassistenten:



Ajuf dem Reiter "Business Intelligence" können Sie den jew. Job markieren und so in die nächtliche Laderoutine aufnehmen:



Weitere Details siehe die Beschreibung des Installationsassistenten bei HIS.

Ladejob-Maske erstellen und ausführen

Tomcat Konfiguration

Vorab: Sie müssen die Umgebungsvariable CATALINA_OPTS erweitern:

```
CATALINA_OPTS="...  -DMODULE_PFAD=/var/lib/tomcat9/webapps/superx/WEB-INF/conf/edustore/db/module  ..."
```

Unter Ubuntu Linux liegt das in der Datei

```
/etc/default/tomcat9
```

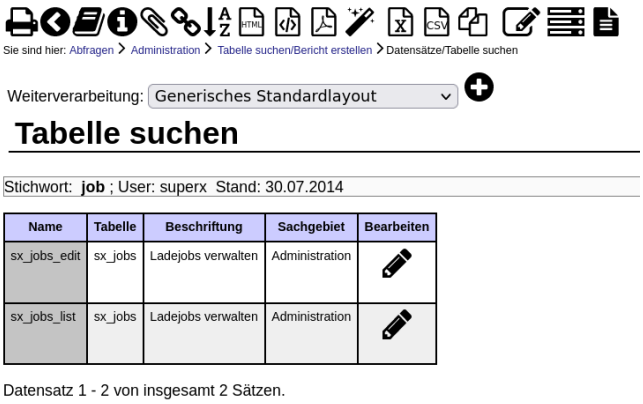
dort die Variable

```
JAVA_OPTS="...  -DMODULE_PFAD=/var/lib/tomcat9/webapps/superx/WEB-INF/conf/edustore/db/module  ..."
```

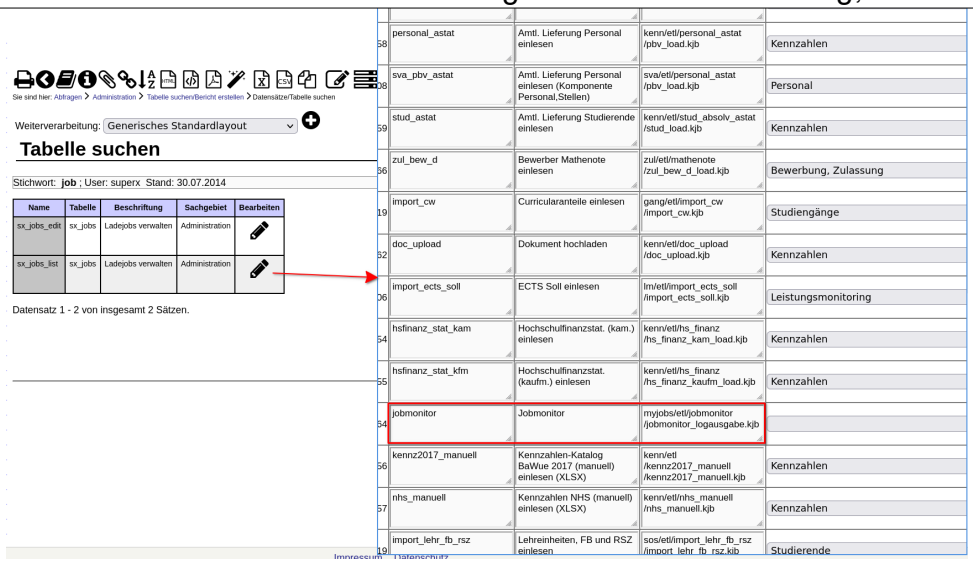
Danach muss man Tomcat neu starten.

Job registrieren

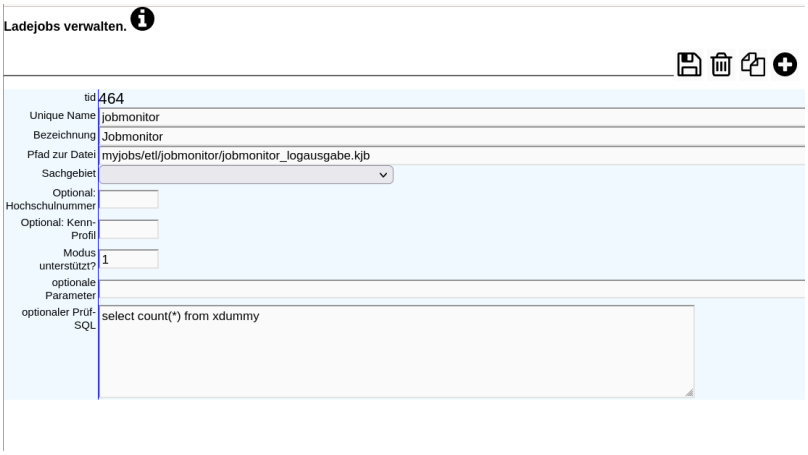
Um den Job im System bekannt zu machen wird ein Eintrag in der Tabelle sx_jobs angelegt. Sie erreichen die Bearbeitung über das Menü Administration -> Tabelle suchen.



Im Listenformular können Sie neue Jobs anlegen, oder vorhandene ändern.



Hier unser Beispiel:

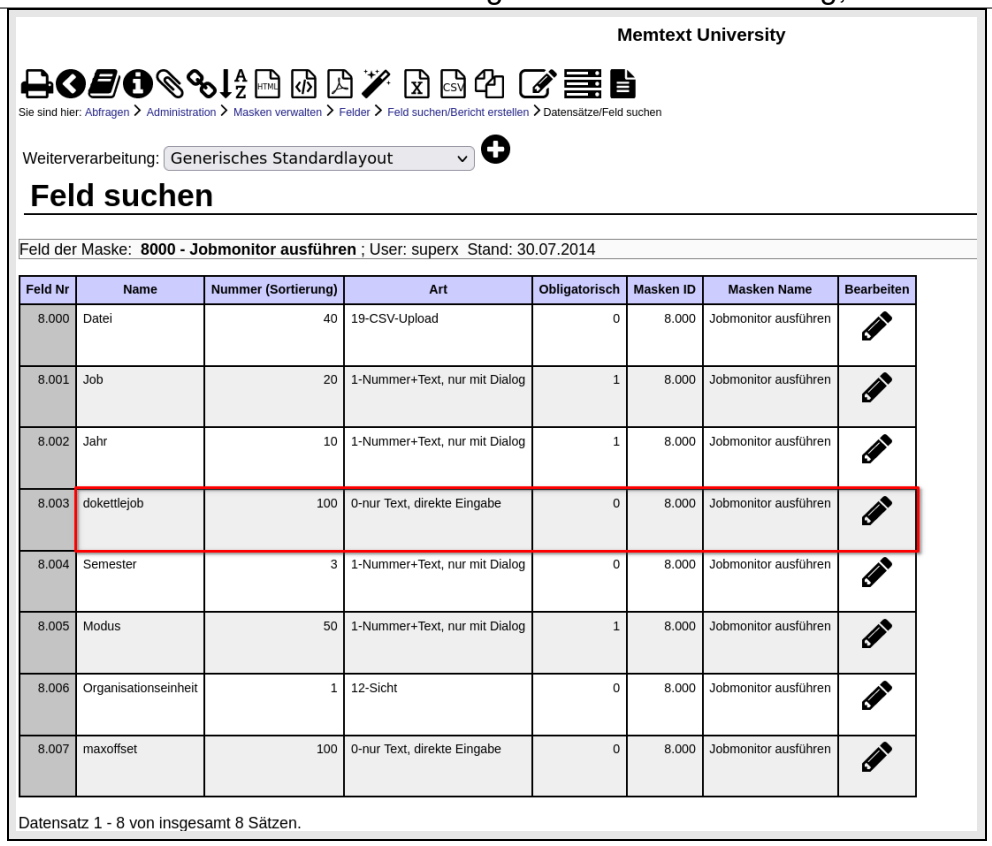


Das Ergebnis des Prüfprotokolls erscheint nach Ausführen des Jobs im Ladeprotokoll. Hier können Sie z.B. die Anzahl der Datensätze in einer Zieltabelle zählen.

Ladejob-Masken erstellen

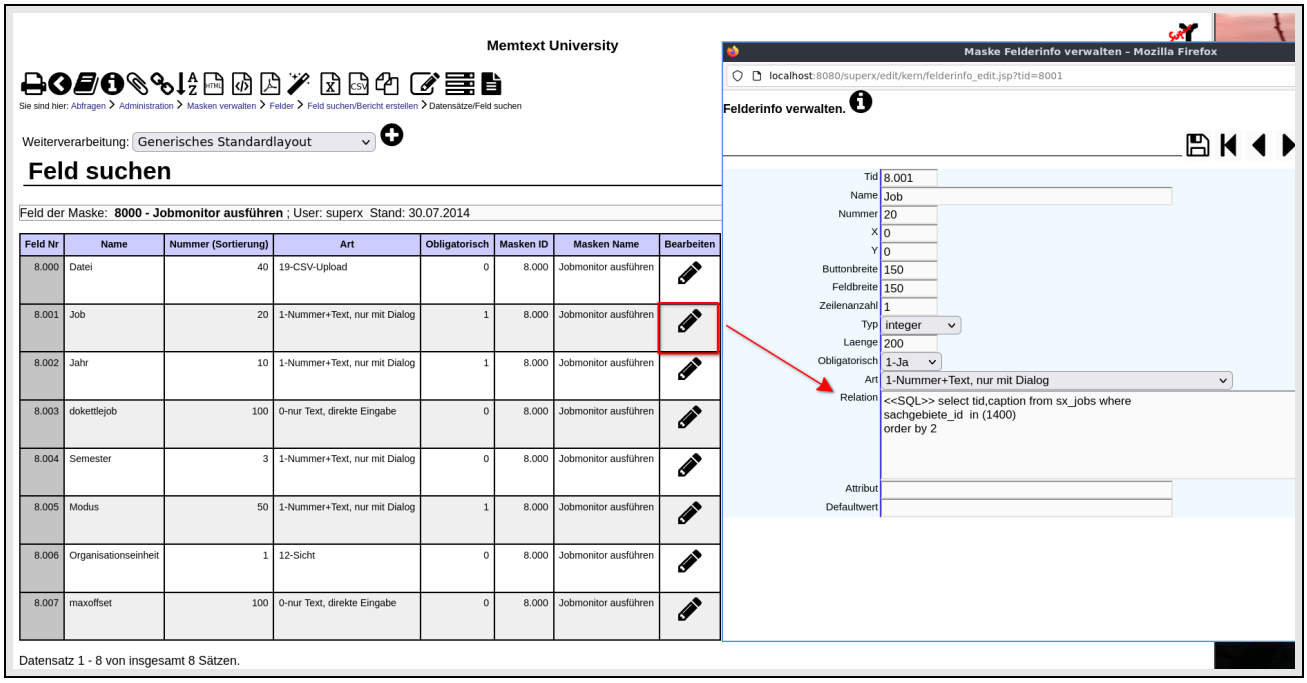
:

Die Masken, welche Ladejobs ausführen benötigen zwingend das Feld **dokettlejob**. Anhand dieses Feldes wird dem System mitgeteilt, dass ein Kettle-Job auszuführen ist. Das Feld darf versteckt werden.



:

Um die auswählbaren Kettle-Jobs zu definieren wird auf die Tabelle sx_jobs zugegriffen.



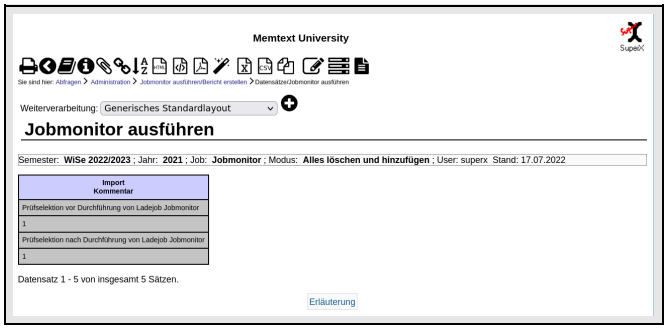
Die Maske kann nach den vorhandenen SuperX-BI-Techniken mit speziellen Benutzer- und Gruppenrechten versehen werden. Auch neue Maskenfelder sind möglich, diese werden automatisch als Parameter an den Job übergeben (sofern

der Kettle-Job den Parameter kennt).

Ladejob-Masken nutzen

Der Ladejob bietet eine einfache Browser-Oberfläche:

In der Maske wählen Sie den Ladejob, dies ist ein Pflichtfeld. Mit dem Abschicken wird der Job gestartet, und der oben definierte Test-SQL im Sinne einer "vorher-nachher"-Messung abgesetzt.



Kitchen

Kitchen dient der Ausführung von Kettle Jobs via Kommandozeile. Wichtig ist die Umgebung, in der Kitchen gestartet wird.

Home-Verzeichnis

Zunächst ist es wichtig, das Home-Verzeichnis von Kettle beim Aufruf von Kitchen zu kennen. Standardmäßig liegt das Verzeichnis in HOME/.kettle, also unter Linux (3 Varianten):

$HOME/.kettle
~/.kettle
/home/Benutzername/.kettle

und unter Windows (2 Varianten):

C:\users\Benutzername
C:\Documents and Settings\{Benutzername}\.kettle

Im Home-Verzeichnis von Kettle werden Konfigurationsdateien gesucht. Sie können das Home-Verzeichnis auch variieren, indem Sie die System-Umgebungsvariable KETTLE_HOME setzen.

Umgebungsvariablen

Im Home-Verzeichnis wird nach zwei Dateien gesucht:

kettle.properties:
Allgemeine Umgebungsvariablen

shared.xml:
Datenbankverbindungen

Kettle.properties

Sie können in der Textdatei beliebige Konfigurationen setzen, die Sie nicht in den Jobs / Transformationen oder in der Datenbank speichern wollen, z.B. Zugangsdaten für Email-Server.

Beispielinhalt:

```
MAILUSER = schulung@superx-projekt.de
MAILPW = anfang12
```

Sie können Passworte sogar verschlüsseln, indem Sie das Kommandozeilen-Tool "encr" nutzen. Unter Linux:

```
encr.sh -kettle anfang12
```

Unter Windows:

```
encr.bat -kettle anfang12
```

Wenn Ihr Passwort Leerzeichen oder Sonderzeichen enthält, ist es sicherer das Passwort mit einfachem Hochkomma (Linux) oder doppelten Anführungsstrichen (Windows) zu umschließen.

Es wird ein verschlüsseltes Passwort ausgegeben:

```
Encrypted 2be98afc86aa7f2e4aa17a871d095fe88
```

Dieses Passwort können Sie dann in die kettle.properties eintragen:

```
MAILUSER = schulung@superx-projekt.de
MAILPW =Encrypted 2be98afc86aa7f2e4aa17a871d095fe88
```

Shared.xml

Speziell für Datenbankverbindungen gibt es eine eigene Datei **shared.xml**, diese liegt im Homeverzeichnis der Betriebssystem-Benutzerkennung, die den Job startet, im Unterverzeichnis .kettle . Die Datei wird von Spoon angelegt, wenn man eine Connection "teilt" (engl. "share"). Hier ein Beispiel:

```
<?xml version="1.0"
encoding="UTF-8"?>
```

Es werden zwei Datenbankverbindungen konfiguriert, die dann in Kettle Transformationen im Element "connection" genutzt werden können.

Bei der Entwicklung von Jobs und Transformationen werden die Connection-Elemente auch in den jew. Quelltext geschrieben und gespeichert. Und bei der Ausführung haben diese Elemente eine höherer Priorität als die shared.xml, so dass man diese Elemente vor dem Installieren im Zielsystem entfernen muss.

Shared.xml in SuperX bzw. HISinOne-BI

Die Datenbankverbindungen im jew. Job-Quellcode, bzw. in der Transformation, bzw. in der Kettle Kurs Teil 2 stehen in SuperX bzw. HISinOne-BI in "Konkurrenz" zur "databases.xml" (HISinOne-BI) bzw. zur "db.properties" (SuperX). Hier eine Erläuterung des Zusammenhangs:

1. in erster bzw. höchster Priorität wird die Connection im jew. Job-Quellcode, bzw. in der Transformation genutzt
2. in zweiter Priorität die Datei shared.xml im Homeverzeichnis der Benutzerkennung.
3. in dritter Priorität, also wenn es weder 1. noch 2. gibt, und nur beim Lauf eines Jobs in der SuperX-Webanwendung und nur bei der dbconnection "eduetl" wird die "databases.xml" (HISinOne-BI) bzw. zur "db.properties" (SuperX) genutzt.

Beim Betrieb in SuperX bzw. HISinOne-BI unter Debian/Ubuntu/Suse Linux liegt die shared.xml standardmäßig im Verzeichnis /var/lib/tomcat*/.kettle , und gehört dem User tomcat. Achten Sie auf Lese- und Schreibrechte.

Ausführung, Logging und Fehlermeldung

KETTLE_ENV einrichten

Für das im folgenden Abschnitt erläuterte Skript wird eine KETTLE_ENV benötigt, hier ein Beispiel für Ubuntu Linux.

Für Kettle-Version 7.0.0.0-25 wird Java 8 benötigt.

```
#!/bin/bash
KETTLE_PFAD=~/kettle/7.0.0.0-25
export  KETTLE_PFAD
PENTAHO_JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export  PENTAHO_JAVA_HOME
PENTAHO_JAVA=java
export  PENTAHO_JAVA
PATH=$PENTAHO_JAVA_HOME/bin:$PATH
export  PATH
PENTAHO_DI_JAVA_OPTIONS="-Xms1024m  -Xmx2048m  -XX:MaxPermSize=256m  "
export  PENTAHO_DI_JAVA_OPTIONS
```

Ausführungsskript inkl. Mailversand

Folgendes Skript führt Kitchen aus, schreibt die Ausgabe in eine Logdatei und sendet bei fehlerhafter Ausführung eine Mail mit der Logdatei im Anhang. Das Beispiel gilt für Ubuntu Linux mit dem Mailclient "s-nail".

```
#!/bin/bash
.  ./KETTLE_ENV
JOB_PFAD="$MYJOBS_PFAD/etl/jobmonitor"
JOB_FILE="jobmonitor_logausgabe"

$KETTLE_PFAD/kitchen.sh  /file=$JOB_PFAD/$JOB_FILE.kjb  /norep  >$JOB_PFAD/$JOB_FILE.log  2>&1

if  "0"  -ne  "$?"
```

```
then
# Mailversand
echo "Mail wird versandt."
echo "Logfile im Anhang." | s-nail -s "Kettle-Job $JOB_FILE.kjb ist fehlgeschlagen!" -a "$JOB_PFAD/$JOB_FILE.log" test@mailserver.de
else
echo "Job lief ohne Fehler."
fi
```

Zunächst wird KETTLE_PFAD gesetzt, also der Ort, wo Kettle installiert ist. Dann werden der Jobpfad sowie -name übergeben.

Zum Ausführen des Kettle-Jobs durch Kitchen wird im Kettle-Verzeichnis unter Linux kitchen.sh ausgeführt. Über die Option -file werden der Dateipfad und -name übergeben. Die Ausgabe wird in eine Logdatei geschrieben.

```
$KETTLE_PFAD/kitchen.sh /file=$JOB_PFAD/$JOB_FILE.kjb /norep >$JOB_PFAD/$JOB_FILE.log 2>&1
```

Abschließend erfolt eine Auswertung des Returncodes. Falls dieser ungleich 0 ist, also einen Fehler jeglicher Art kennzeichnet, wird eine Mail inklusive Logdatei versandt.

```
if "0" -ne "$?"
then
# Mailversand
echo "Mail wird versandt."
echo "Logfile im Anhang." | s-nail -s "Kettle-Job $JOB_FILE.kjb ist fehlgeschlagen!" -a "$JOB_PFAD/$JOB_FILE.log" test@mailserver.de
else
echo "Job lief ohne Fehler."
fi
```

Achtung:

Kitchen versteht als Parameter-Präfix sowohl "-" als auch "/", also z.B. "-file:..." und "/file:...". Man könnte auch "=" statt ":" als Trenner nehmen. Aber in einer Publikation von Matt Casters et al. (2010, S.323) werden "/" und ":" empfohlen, die machen unter Windows/DOS weniger Probleme.

Parameter übergeben

Neben der Übergabe eines Jobfiles lässt kitchen.sh weitere Optionen zu (s. Kitchen Optionen). Sehr nützlich ist die Übergabe von Parametern für den ausgeführten Job.

```
./kitchen.sh /file=test.kjb -param:Semester=20222
```

In diesem Beispiel wird dem Jobparameter Semester der Wert 20222 übergeben. Bei Parametern mit Leerzeichen müssen Sie Anführungszeichen drum herum setzen.

Sie können auch spezielle JVM-Parameter übergeben, java-typisch mit -D vorangestellt.

```
./kitchen.sh /file:test.kjb -param:Semester=20222 -Dlanguage=de
```