SuperX-Entwicklerhandbuch

www.MemText.de

•Daniel Quathamer
danielq@memtext.de
•Meikel Bisping
mbisping@memtext.de
•

http://www.superx-projekt.de

Version 4.5
Stand 18.5.2015

Inhaltsverzeichnis

1 Einrichten der Entwicklungsumgebung

1.1 SuperX

Für die Entwicklung unter SuperX müssen Sie ein Unix/Linux System nutzen. Für die Entwicklung unter Windows können Sie eduSTORE nutzen.

Je nach Werkzeug gibt es unterschiedliche Vorgaben. Wichtig ist aber immer, dass die Shell-Umgebungsvariablen in der Datei SQL_ENV korrekt sind. Wenn Sie sich ein Kernmodul lokal installieren, ist die auf jeden Fall gegeben.

1.2 eduSTORE

Unter Windows können Sie z.B. Eclipse nutzen, um in eduSTORE zu entwickeln. Siehe auch http://wiki.his.de/mediawiki/index.php/EduStore:_Installation_und_Einrichtung_der_Entwicklungsumgebung

1.2.1 Installation und Konfiguration von Eclipse

Eclipse ist die Standard-Entwicklungsumgebung für eduSTORE. Ein Teil von eduSTORE wird im HIS-CVS gepflegt, und ein Teil in diversen `git` Repositories.

1.2.1.1 Starten von Eclipse

Die Installation von Eclipse ist denkbar einfach: Wenn Java installiert ist, muss man Eclipse nur noch herunterladen und Entpacken, und danach die Anwendung "eclipse" starten. Achten Sie darauf dass Eclipse mit genügen RAM gestartet wird. Aufrufbeispiel:

```
eclipse -vm /home/superx/tools/java/jdk1.6.0_21/bin -vmargs
-Dfile.encoding=UTF-8 -Xmx1500m -XX:MaxPermSize=200m
```

1.2.1.2 Tomcat Plugin

Um Tomcat aus Eclipse zu starten, muss man das Sysdeo Tomcat Plugin installieren.

- Laden Sie das Plugin herunter von der Eclipse-Site, und entpacken Sie es in <<Eclipse-Pfad>>/dropin.
- Starten Sie dann Eclipse einmal mit dem Parameter "-clean".
- Wenn das Tomcat Symbol für Start, Stopp und Restart erscheint ist alles ok. Danach starten Sie erneut mit den normalen Parametern.
- In dem Dialog Window | Preferences | Tomcat geben Sie die Tomcat Version an, die Sie nutzen, und den Pfad zu Tomcat.
- Das Projekt (SuperX, qisserver, oder ICE) deklarieren Sie dann in den Projekteingeschaften unter Tomcat als "Ist ein Tomcat Projekt". Die Pfade zu den Java-Quellen sind anzugeben.
- Machen Sie dann einen "Clean build".

•Wenn der Tomcat Start aus Eclipse dann klappt, können Sie damit arbeiten. Wenn die Quellen neu kompiliert sind, können Sie auch Servlet Debugging betreiben.

1.2.1.3 Umgebung für Eclipse

Darüber hinaus sollten Sie die Compilerwarnungen gem. Vorgabe HIS einrichten:

http://wiki.his.de/mediawiki/index.php/Eclipse_Konfiguration_f%C3%BCr_QIS

Derzeit ist nur die CVS-Entwicklung in HISinOne auf diese Compilerwarnungen umgestellt, aber die Edustore-Entwicklung noch nicht. Wenn Sie aus Eclipse auch ANT nutzen, müssen Sie unter {Window | Preferences | Ant | Runtime | Classpath} im Button "Ant Home" auf webapps/qisserver/WEB-INF/internal/ant umstellen.

1.2.1.3.1 Ergänzung Path-Properties für SuperX-Projekt.

Beim Aufruf von einem Jasperbericht kam die Fehlermeldung:

org.eclipse.jdt.internal.compiler.ICompilerRequestor not found.

Man muss in Project / Properties / Java Build Path / Libraries

Add external library

org.eclipse.jdt.core*.jar hinzufügen.

Sonst kommt org.*eclipse.jdt*.internal. compiler.*ICompilerRequest*

Alternative: ggfs. googlen, System Variable auf JDK_HOME setzen o.ä.

Denn:

Achtung Unterschiede zum Compiler ohne Eclipse

Primärinfo Datenblatt aufrufen und dann "Bericht als Kreuztabelle", kommt

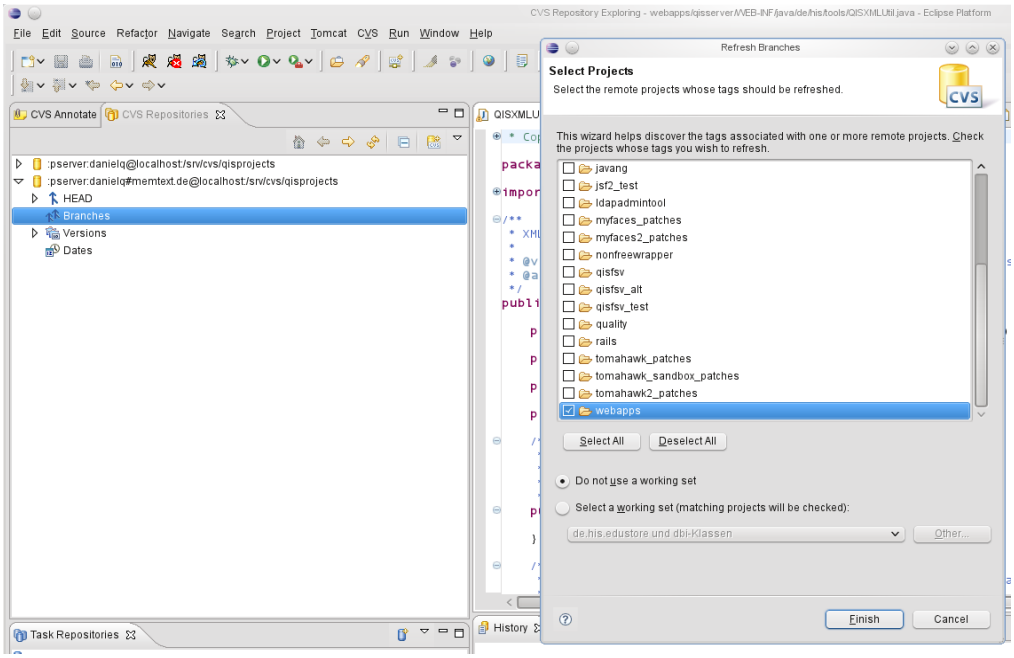
```
net.sf.jasperreports.engine.JRException: org.xml.sax.SAXParseException: cvc-
complex-type.2.4.d: Invalid content was found starting with element 'text-
Field'. No child element is expected at this point. at net.sf.jasperre-
ports.engine.xml.JRXmlLoader.loadXML(JRXmlLoader.java:247) at net.sf.jasper-
reports.engine.xml.JRXmlLoader.loadXML(JRXmlLoader.java:230) at
```

Das Problem existiert nicht bei Tomcat ohne Eclipse

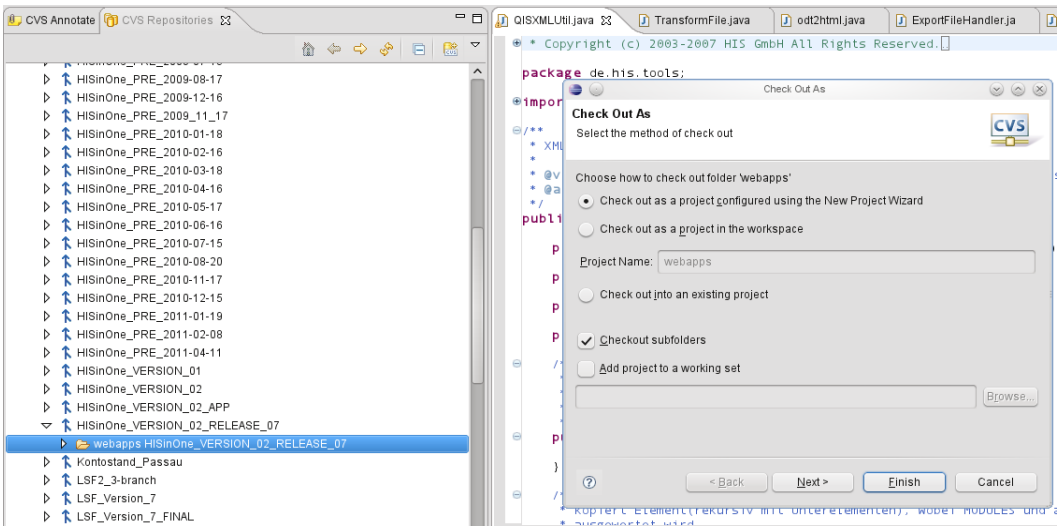
CVS

In Eclipse können Sie das CVS-Plugin nutzen, um Java Quellen zu verwalten und zu kompilieren.

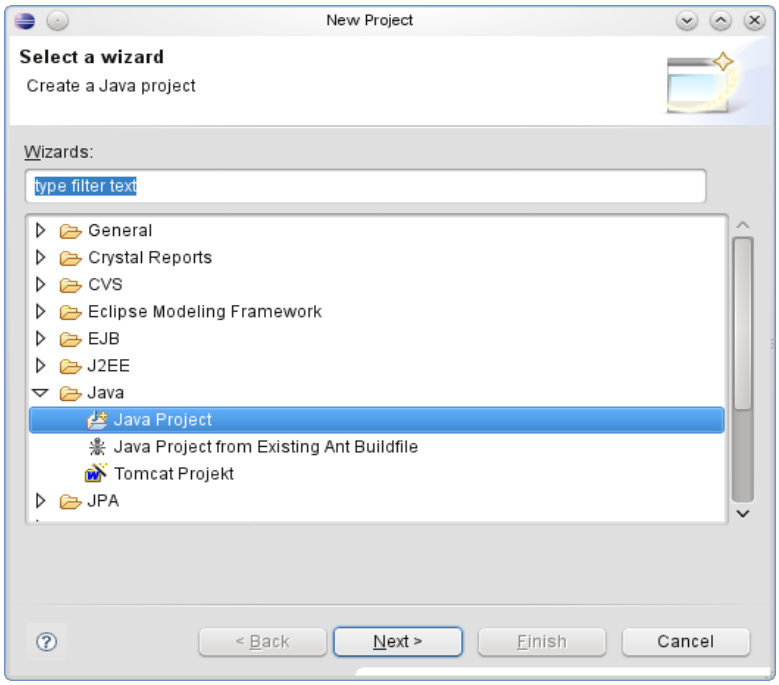
Gehen Sie dazu in die CVS Repository Perspective, und suchen zunächst den jew. Branch bzw. den HEAD. Hier das Beispiel für einen Branch:



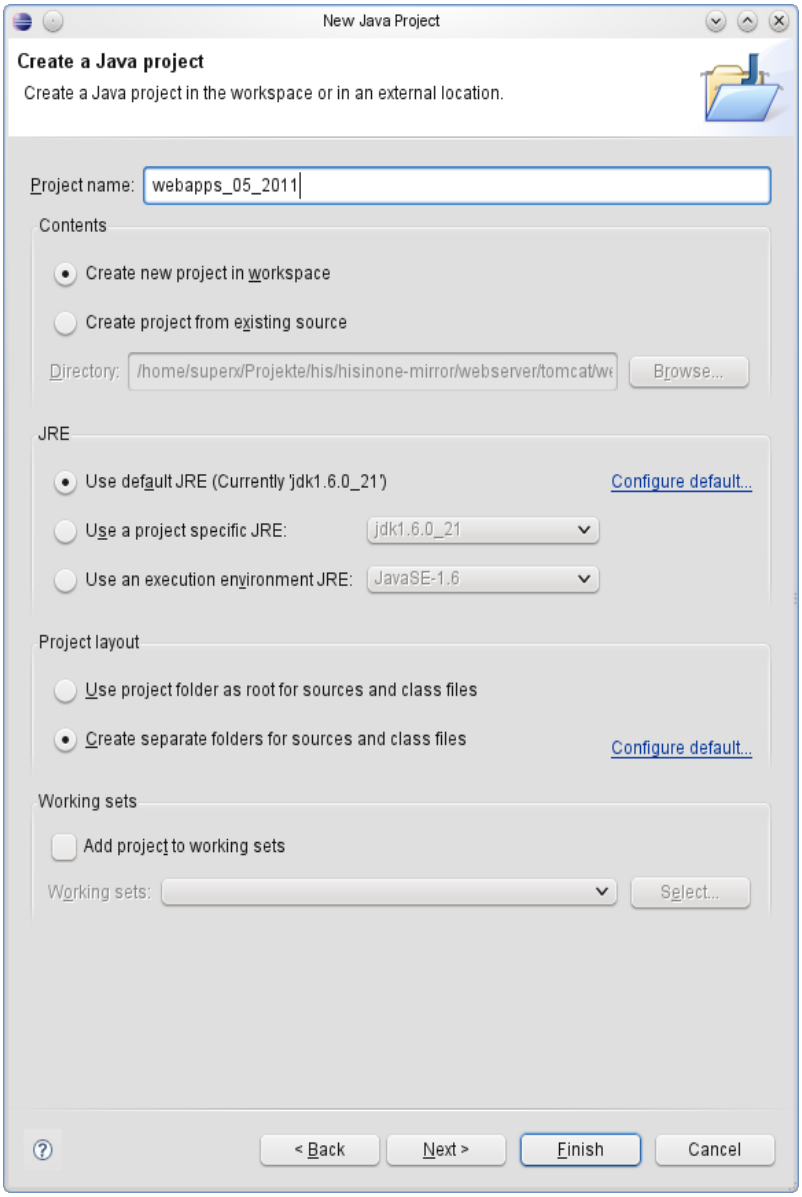
Markieren Sie "Branches", mit der rechten Maustaste können Sie "Refresh Branches" angeben. Dann suchen Sie alle Branches vom "webapps"-Projekt.



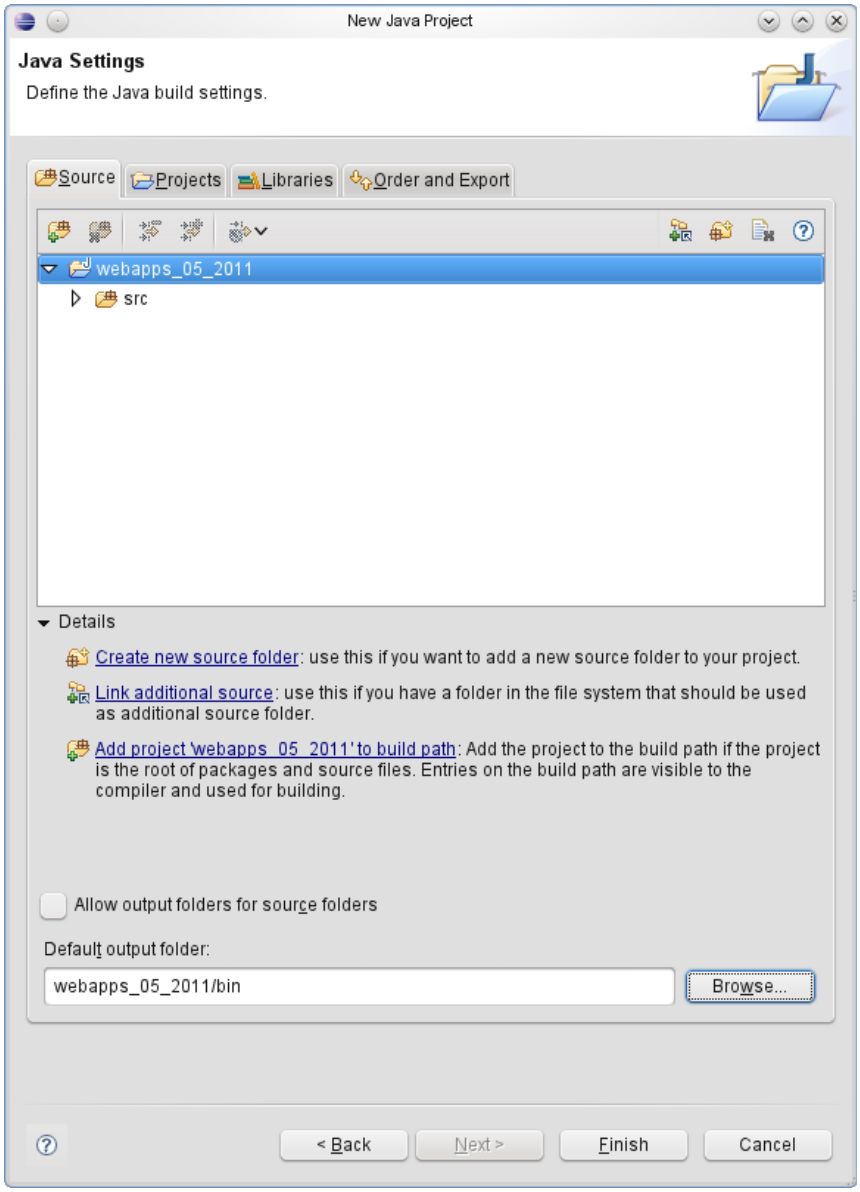
Checken Sie das jew. Projekt als neues Projekt mit dem Project Wizard aus. Die Art des Projektes ist ein Java Projekt.



Dann geben Sie den Speicherort an:



Den Pfad für Quellen und kompilierte Dateien gibt Eclipse zunächst vor, Sie übernehmen die Einstellungen zunächst:



In der `.project` des CVS wird das später automatisch korrigiert. Mit Klick auf "Finish" wird das Projekt geladen.

Danach können Sie direkt den Build des Projektes starten.

1.2.1.4 Git

Ab August 2011 wird git genutzt, vergl.

https://wiki.his.de/mediawiki/index.php/Arbeiten_mit_dem_verteilten_Versionskontrollsystem_Git

Für die Webanwendung superx sind keine weiteren Tätigkeiten mehr nötig. Bei den Webanwendungen **iceproject** und **cocoon** müssen Sie zunächst einen Build ausführen. Gehen Sie dazu in der Shell in das Verzeichnis `ice_ii_build`

und starten Sie das Script

```
ant -f build_his1.xml dist_his1
```

Die Webanwendungen `cocoon` und `iceproject` werden im Unterverzeichnis `dist` erzeugt, und können dort direkt genutzt / in einem Tomcat gemounted werden. Das Vorgehen wird im Folgenden beschrieben.

1.2.1.5 Integration in einen Tomcat

Sie können alle Webanwendungen nicht in einem Eclipse Projekt betreiben, weil die Einstellungen von CVS und git kollidieren. Aber in einem Tomcat lassen sich alle Webanwendungen bereitstellen, indem man wie folgt vorgeht:

•Stellen Sie in der `server.xml` Ihres Tomcat die `appBase` auf das Verzeichnis, wo Ihr qisserver liegt (genauer gesagt eines darüber). Damit sind der qisserver und die ROOT-Webanwendung aus HISinOne direkt verfügbar.

•Erstellen Sie unterhalb dieses appBase-Verzeichnisses symbolische Links zu den Webanwendungen superx, iceproject und cocoon

•Nun können Sie entweder den Tomcat-Installer von Edustore nutzen, oder ein vorgefertigtes ANT-Script.

Beim Tomcat Installer:

•Erzeugen Sie die Datenbanken edudata, edugeta, und eduetl (und für eduetl noch die Prozedursprache plpgsql), und konfigurieren Sie Ihre databases.xml sowie Ihr Spezialmodul. Anleitung siehe hier: http://wiki.his.de/mediawiki/index.php/Edustore_Installation_und_Administration#Datenbanken_einrichten

•Danach brauchen Sie nur noch Tomcat zu starten, im ersten Start wird Edustore automatisch konfiguriert.

1.2.1.6 Installation / Updates via ANT

Beim ANT-Script werden die vorhandenen Datenbanken geleert, und dann wird jedes Modul nacheinander installiert und (beim Target "reinstall_demo") mit Demodaten geladen. Zur Einrichtung müssen Sie das Edustore-Repository (Head oder einen Branch) herunterladen. Danach können Sie ANT installieren und das Script benutzen:

http://wiki.his.de/mediawiki/index.php/Edustore_Installation_und_Administration#Apache_ANT_einrichten

Sie können dann sämtliche Datenbanken leeren und neu installieren sowie mit Demodaten füllen, indem Sie das ANT-Script mit

```
ant -f install_his1.xml reinstall_demo
```

ausführen.

Auch andere Tätigkeiten sind möglich, z.B Module zu deinstallieren und zu installieren. Rufen Sie

```
ant -f install_his1.xml -p
```

auf, um eine Übersicht zu bekommen.

Achtung: die ANT-targets, die mit dem Kürzel "update_" beginnen, laden standardmäßig immer die Demodaten aus dem Edustore-Repository. Wenn Sie eige-

ne Testdaten verwenden wollen, sollten Sie direkt über Eclipse bzw. die Java-Kommandozeile arbeiten.

1.2.2 DOS

Für den Start von Scripten aus der DOS Box müssen Sie ein paar Umgebungsvariablen einstellen, die sich auf Java und auf Pfade zum Edustore Server beziehen:

```
set JAVA_OPTS="-Xmx256M -Dfile.encoding=UTF-8"
set QISSERVER_PFAD="<<Ihr Pfad zu Tomcat>>\webapps\qisserver"
rem die folgenden Variablen braucht man nicht ändern
rem ggf. prüfen ob Dateiname der freemarker.jar existiert:
set QISSERVER_LIB_PFAD=%QISSERVER_PFAD%\WEB-INF\lib
set QIS_CLASSPATH="%QISSERVER_PFAD%\WEB-INF\classes:%QISSERVER_LIB_PFAD
%\quartz-1.5.2.jar:%QISSERVER_LIB_PFAD%\log4j-1.2.16.jar:%QISSERVER_LIB_PFAD
%\mail-1.4.jar:%QISSERVER_LIB_PFAD%\commons-io-1.4.jar:%QISSERVER_LIB_PFAD
%\freemarker.jar:%QISSERVER_LIB_PFAD%\ifxjdbc.jar:%QISSERVER_LIB_PFAD%\ant-
1.8.1.jar:%QISSERVER_LIB_PFAD%\hosu.jar:%QISSERVER_LIB_PFAD%\postgresql-8.3-
604.jdbc4.jar:%QISSERVER_LIB_PFAD%\xercesImpl-2.8.1.jar:%QISSERVER_PFAD
%\..\..\lib\servlet-api.jar:%QISSERVER_LIB_PFAD%\velocity-1.6.1-patched.jar:
%QISSERVER_LIB_PFAD%\commons-lang-2.4.jar:%QISSERVER_LIB_PFAD%\jdom-patche-
d.jar:%QISSERVER_LIB_PFAD%\mail-1.4.2.jar"
```

2 Erzeugung und Änderung von Masken

Die Abfragemasken liefern die Daten aus den Basissystemen an das SuperX-Frontend aus. Einige Abfragen zur Administration sind im Kernmodul enthalten, die Abfragen zu den Basissystemen sind in den jeweiligen Modulen enthalten. Die Abfragen in der Administration erlauben es, neue Masken anzulegen, zu kopieren und zu löschen.

Um den Austausch von Abfragen innerhalb der Hochschulen zu erleichtern ("Abfragen-Pooling" über die SuperX-Website), sollten die Masken immer im Nummernkreis xxxx0000 bis xxxx9990 liegen, wobei xxxx der von der HIS verwandten Hochschulnummer entspricht. Die Zehnerschritte ergeben sich daraus, dass die dazwischen liegenden Nummern für die Maskenfelder (Tabelle `felderinfo`) reserviert sind[1].

Im Folgenden finden Sie allgemeine Hinweise für die Verwaltung der Masken.

Die Masken lassen sich browserbasiert, über UNIX-Shellscripte, und über Access administrieren.

[1] Aus historischen Gründen liegen die Nummern aus Karlsruhe im Bereich 0-9990, aus Duisburg im Bereich 10000-19990.

Weitergehende Möglichkeiten bietet aber das XML-Frontend (Möglichkeit der Editierung von großen `text`-Feldern bei Postgres als Datenbanksystem). Nach der Anmeldung haben Administratoren das Recht, Masken zu löschen, zu kopieren und erzeugen. Die einzelnen Felder der Masken lassen sich direkt in der Datenbank oder z.B. mit MS Access verändern. Im Applet sind nur grundlegende Verwaltungsoperationen möglich. Sie sind als Ersatz für die UNIX-Scripte gedacht.

Folgende "Abfragen" zur Maskenverwaltung gibt es im Sachgebiet Administration:	•Maske kopieren •Maske löschen
Darunter im Ast "Felder" gibt es noch folgende Abfragen:	•Feld kopieren •Feld löschen
Darüber hinaus gibt es (nur unter Postgres) die Masken zur Pflege von Masken bzw. Feldern	•Maske suchen •Feld suchen

Maske kopieren. Wie im UNIX - Script wir eine Maske in eine neue Maske kopiert, und alle zugehörigen Tabellen werden aktualisiert. Zusätzlich wird auch der Eintrag im Themenbaum gemacht. Bei der Nummer der Maske (tid) sollten Sie das Nummernschema von SuperX einhalten, um in Zukunft Abfragen-Pooling zu ermöglichen.

Maske löschen. Wie im UNIX-Script werden Masken aus allen dazugehörigen Tabellen entfernt. Zusätzlich wird auch der Eintrag im Themenbaum gelöscht. Zur Sicherheit muss die Nummer der Maske manuell eingegeben werden.

Maske suchen . Sie können Masken suchen und im XML-Frontend komfortabel editieren. Schränken Sie Ihre Auswahl auf ein Sachgebiet ein, und drücken Sie "Abschicken". Sie erhalten eine Liste mit "Treffern", und rechts befinden sich jeweils Buttons zum ansehen bzw. editieren einer Maske. Die Maske läuft nur unter Postgres, weil Informix kein direktes Bearbeiten von Blob-Feldern mit sql unterstützt.

Feld suchen. Sie können analog zu "Maske suchen" auch Felder suchen und bearbeiten.

Die Abfragen sind selbsterklärend; das Erzeugen neuer Masken, Löschen vorhandener Masken und Kopieren vorhandener Masken ist nur für Userkennungen möglich, die in der Tabelle `userinfo` im Feld `administration` den Wert 1 haben. Natürlich sollten die Abfragen sehr vorsichtig benutzt werden, sie sind die einzigen Abfragen in SuperX, die tatsächlich Änderungen an der Datenbank vornehmen können.

2.1 Ein Tutorial

Im Folgenden wollen wir zeigen, wie Abfragemasken in SuperX arbeiten und wie man die Ergebnisdarstellung von Abfragen verändern kann. Wir zeigen dies am Beispiel der Abfrage **Studierende (Zeitreihe)**.

Erforderliche Kenntnisse:

•SQL und Datenbankbedienung
•Grundkenntnisse zu SuperX

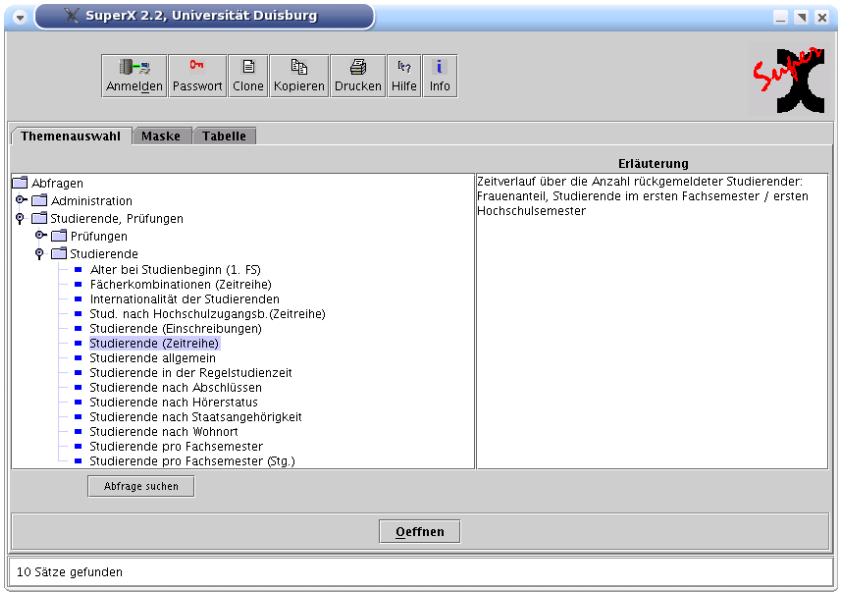
2.1.1 Ausgangspunkt

2.1.1.1 Das Beispiel

Der Ausgangspunkt ist ein Beispiel aus dem SOS-Modul.

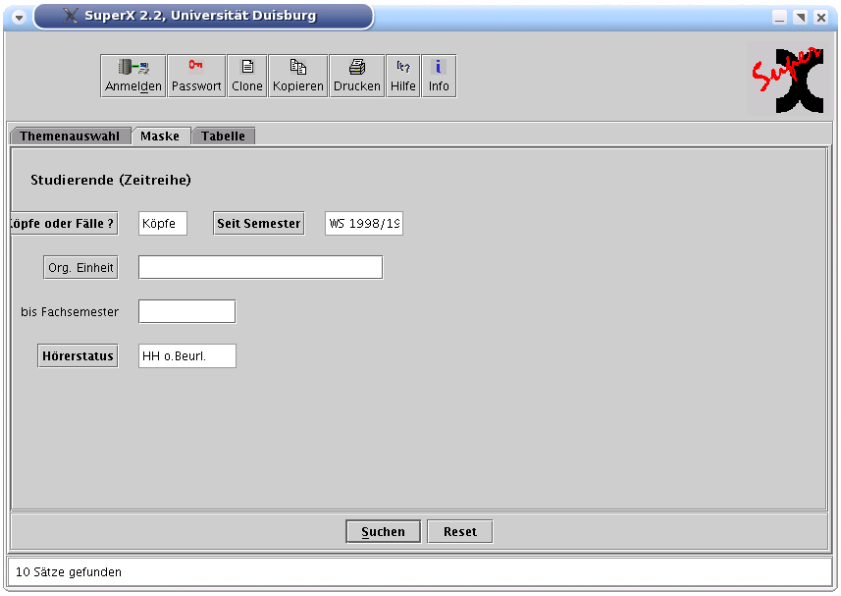
(Klicken Sie jeweils auf die Grafiken, um sie zu vergrößern).

Wir wählen aus dem Themenbaum im Bereich Studierende die Abfrage "Studierende (Zeitreihe)".



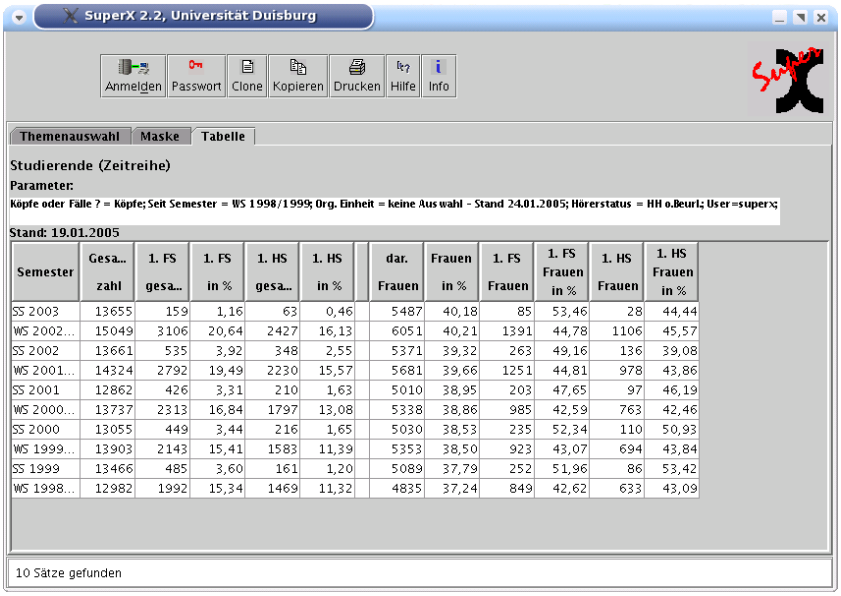
Die Abfrage liefert eine Statistik über Studierende im Laufe von mehreren Semestern, je nach 1. Fachsemester, 1. Hochschulsemester und Geschlecht. Die folgende Abbildung zeigt die Maske:

Die Abbildung zeigt die Auswahlfelder der Maske. Wir wählen für den Zeitraum im Feld "Seit Semester" das WS 1998/1999.



Wenn wir "Suchen" drücken, erscheint folgende Ergebnisdarstellung:

Die Tabelle zeigt in der ersten Spalte die Semester, dann die Gesamtzahl der Studierenden und die Studierenden im 1. Fachsemester.



Semester	Gesa... zahl	1. FS gesa...	1. FS in %	1. HS gesa...	1. HS in %	dar. Frauen	Frauen in %	1. FS Frauen	1. FS Frauen in %	1. HS Frauen	1. HS Frauen in %
SS 2003	13655	159	1,16	63	0,46	5487	40,18	85	53,46	28	44,44
WS 2002...	15049	3106	20,64	2427	16,13	6051	40,21	1391	44,78	1106	45,57
SS 2002	13661	535	3,92	348	2,55	5371	39,32	263	49,16	136	39,08
WS 2001...	14324	2792	19,49	2230	15,57	5681	39,66	1251	44,81	978	43,86
SS 2001	12862	426	3,31	210	1,63	5010	38,95	203	47,65	97	46,19
WS 2000...	13737	2313	16,84	1797	13,08	5338	38,86	985	42,59	763	42,46
SS 2000	13055	449	3,44	216	1,65	5030	38,53	235	52,34	110	50,93
WS 1999...	13903	2143	15,41	1583	11,39	5353	38,50	923	43,07	694	43,84
SS 1999	13466	485	3,60	161	1,20	5089	37,79	252	51,96	86	53,42
WS 1998...	12982	1992	15,34	1469	11,32	4835	37,24	849	42,62	633	43,09

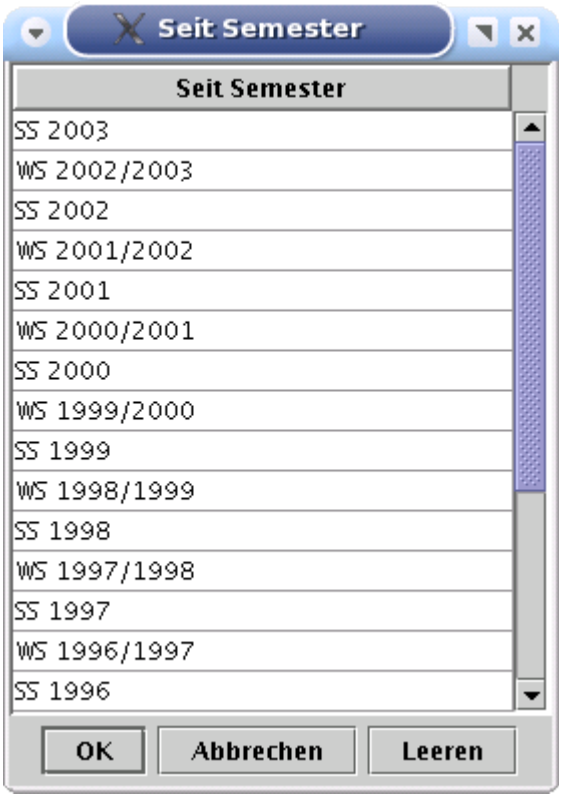
2.1.1.2 Hintergründe

Wie werden nun Felder in den Auswahlmasken gefüllt, und wie werden die Ergebnisse in SuperX ermittelt?

2.1.1.2.1 Die Felddefinitionen

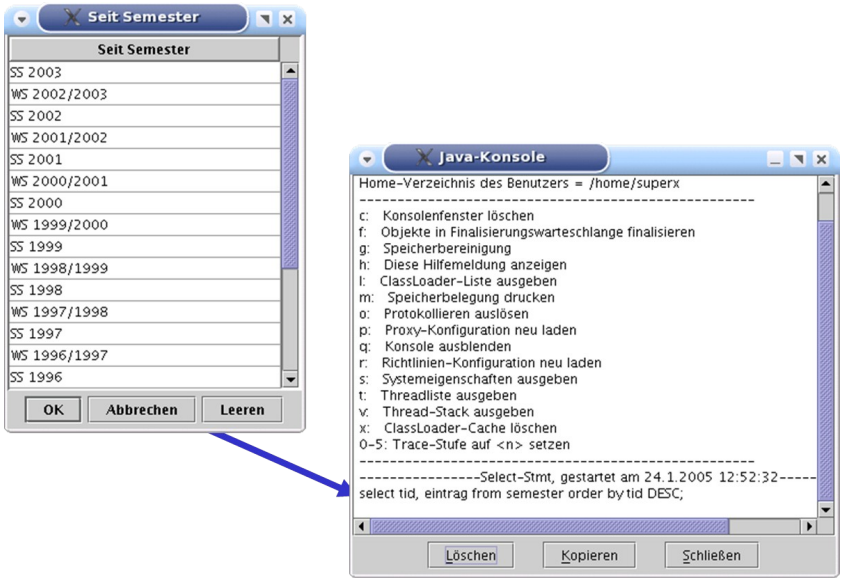
Gehen wir kurz zurück zur Auswahlmaske. Jedes Feld der Maske, z.B. "Seit Semester", ist ein Datensatz in der Tabelle `felderinfo`. Dort finden Sie Angaben zum Namen, Inhalt und Layout des Feldes. Gehen wir zunächst zum Inhalt des Feldes: Die Liste der Semester.

Beim Klick auf das Feld Semester erhalten wir eine Reiche von Semestern zur Auswahl. Die Liste ist absteigend sortiert.



Um den Inhalt des Feldes zu erläutern, wollen wir kurz auf eine nützliche Funktion bei der Abfragenentwicklung hinweisen, die Java-Konsole. Wenn Sie die Java-Konsole in der Systemsteuerung aktiviert haben, dann können Sie im Browser die Konsole anzeigen lassen. Im Mozilla z.B. gehen Sie in das Menü "Werkzeuge"->"Web-Entwicklung" und dort auf "Java Konsole". Im Internet Explorer machen Sie einen Doppelklick auf das Apfelmännchen bzw. eine Kaffeetasse (Java-Symbol je nach Java-Version) unten rechts in der Shortcut-Leiste des Betriebssystems.

Beim Klick auf den Button "Semester" sehen wir in der Konsole folgenden SQL-Befehl unten rechts.



Die SQL-Anweisung liefert aus der Tabelle `semester` die Felder `tid` ("Tupelidentifier") und `eintrag` (der Volltext des Semesters). Der Schlüssel des Feldes `tid` ist unsichtbar, sorgt aber dafür, dass die Sortierung richtig erfolgt.

Hier sehen Sie einen Screenshot der Tabelle `semester` (Auszug) direkt in der Datenbank. Die Nummerierung ist fünfstellig und besteht aus Jahr (vier Stellen) und 1 für Sommer- und 2 für Wintersemester.

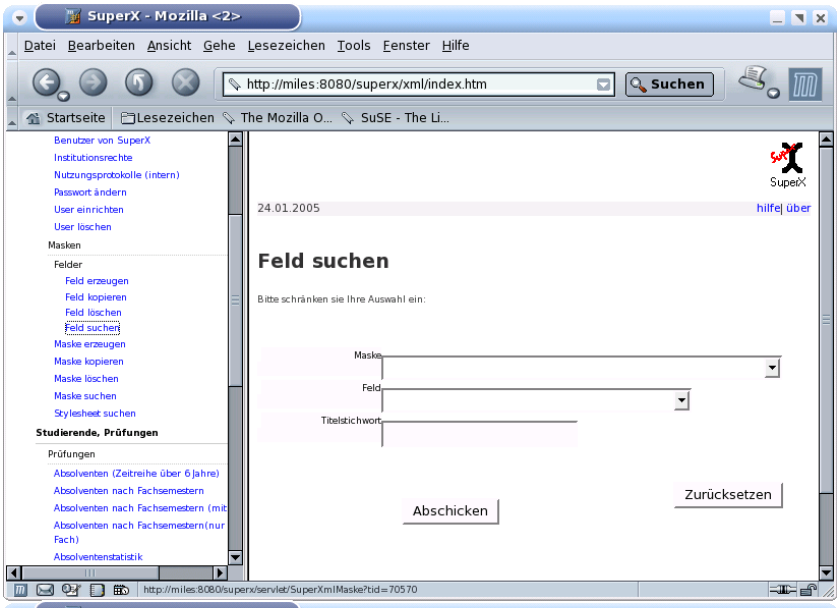
tid	eintrag
19921	SS 1992
19922	WS 1992/1993
19931	SS 1993
19932	WS 1993/1994
19941	SS 1994
19942	WS 1994/1995
19951	SS 1995
19952	WS 1995/1996
19961	SS 1996
19962	WS 1996/1997
19971	SS 1997
19972	WS 1997/1998
19981	SS 1998
19982	WS 1998/1999
19991	SS 1999
19992	WS 1999/2000
20001	SS 2000
20002	WS 2000/2001
20011	SS 2001
20012	WS 2001/2002
20021	SS 2002
20022	WS 2002/2003
20031	SS 2003

2.1.1.2.2 Speichern der Felddefinition: die Tabelle felderinfo

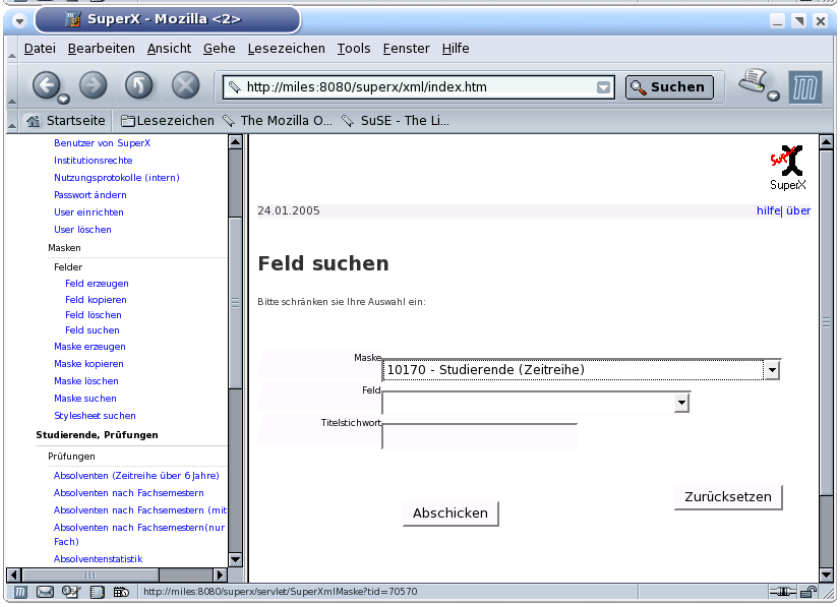
Wo wird nun in SuperX die Felddefinition gespeichert? Viele Scripte in SuperX werden selbst in Datenbanktabellen abgelegt, die Tabelle `felderinfo` enthält die relevanten Angaben für die Felder.

Um dies zu sehen, öffnen wir ein Formular im XML-Frontend, dort befinden sich Bearbeitungsformulare für Felder und Masken.

Im Themenbaum des XML-Frontends finden wir den Menüpunkt "Feld suchen". Rechts erscheint ein leeres Formular.



In dem Formular wählen wir die Abfrage Studierende Zeitreihe aus. Zusätzlich sehen wir auch die Nummer der Maske (10170), das ist bei der Maskenbearbeitung ganz nützlich.

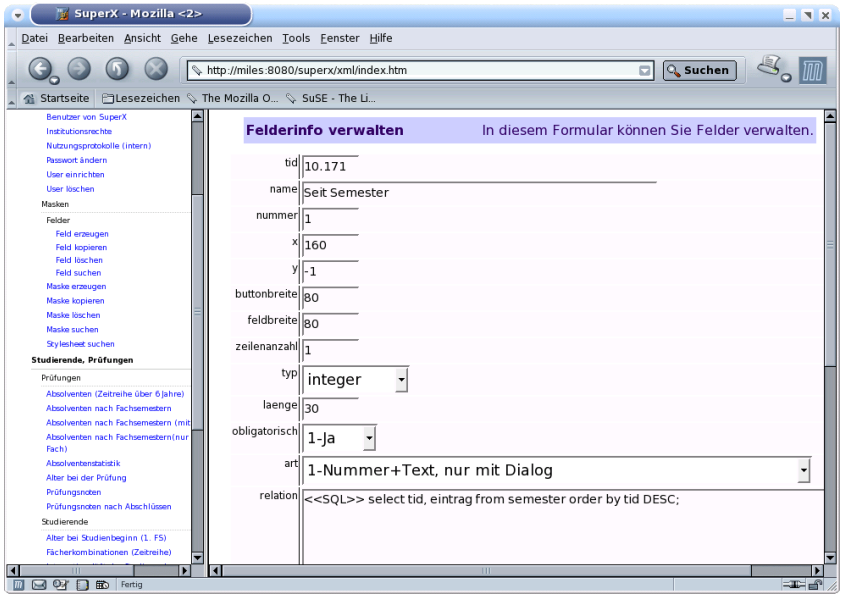


Wenn wir hier Abschicken drücken, erscheint folgendes Bild:

Die Maske enthält fünf Felder; wir sehen die Nummer des Feldes und den Namen. Rechts daneben befindet sich ein Knopf zum Bearbeiten des Feldes.

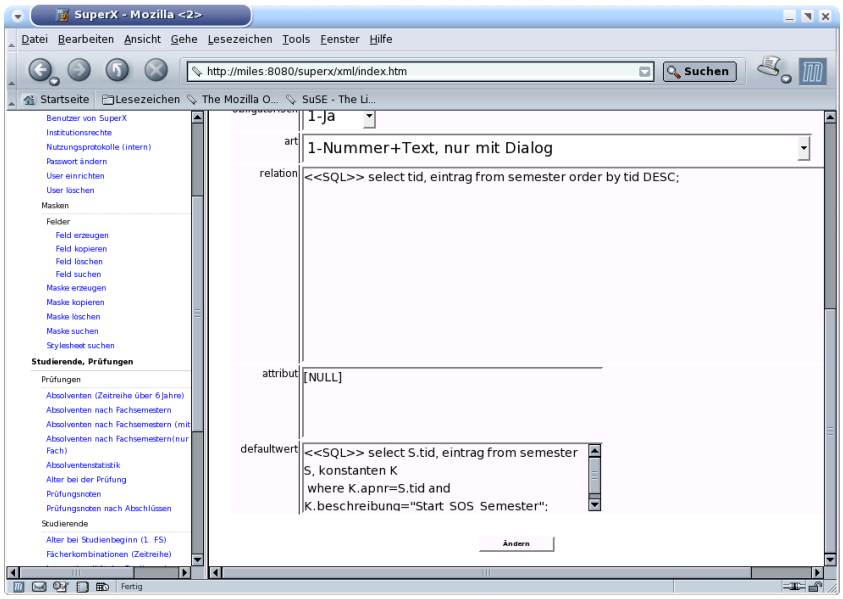
Ein kleiner Hinweis an dieser Stelle: Die Felder werden in der Tabelle `masken_felder_bez` der Maske Nr. 10170 zugeordnet. Wir zählen also bei Feldnummern in Einer-Schritten von der Maskennummer aus hoch. Aus diesem Grunde wählen wir bei Maskennummern Intervalle von mindestens 10, die nächste Maske wäre also mit 10180 nummeriert.

Wir wählen nun das Feld "Seit Semester", und gelangen in ein Bearbeitungsformular der Tabelle `felderinfo`. Wir sehen Name, Nummer, Position auf der Maske, Breite und Typ des Feldes (ganzzahlig). Das Feld ist obligatorisch, und von der Art Nr. 1 (Nummer + Text, mit Dialog).



Ganz unten sehen Sie das Feld "`relation`", in dem nach einem Steuerungszeichen `<<SQL>>` der SQL-Befehl steht, den wir vorhin in der Java-Konsole gesehen haben.

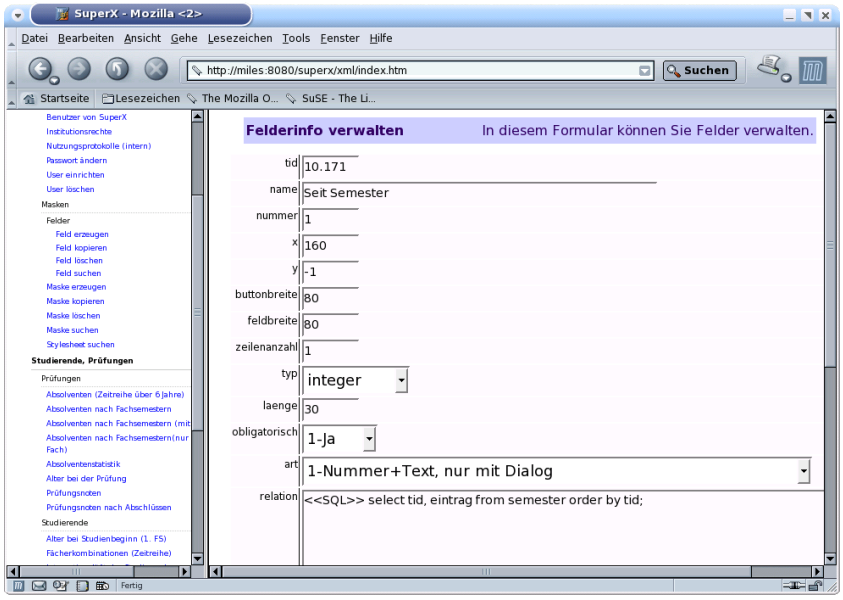
Der Vollständigkeit halber zeigen wir hier den Rest der Tabelle. Unten ist noch der Defaultwert für das Feld angegeben, ebenfalls ein SQL-Ausdruck.



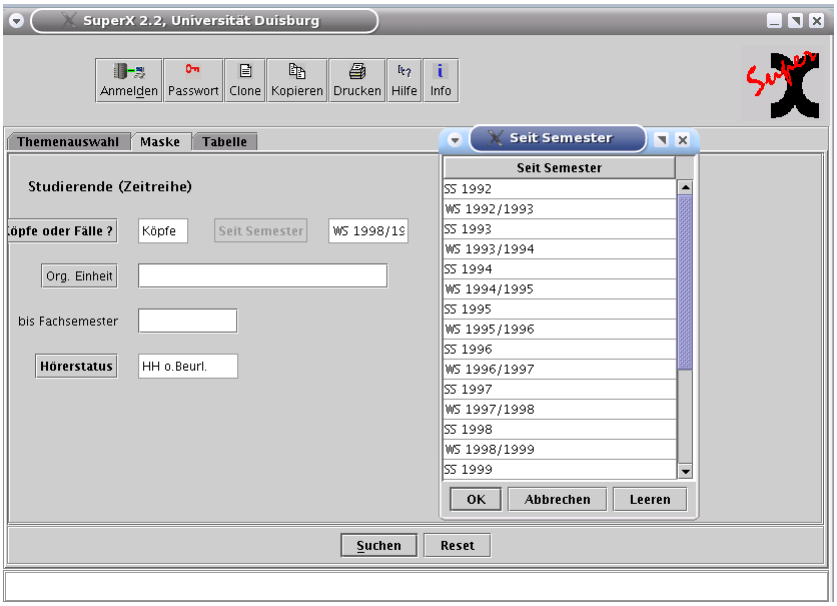
SuperX liest also aus der Datenbank die Scripte für eine Maske bzw. für ein Feld aus einer Tabelle, und führt Sie dann in der Datenbank aus.

2.1.1.2.3 Änderung einer Felddefinition

Wir können nun spassenshalber das Feld ändern, um die Sortierung anzupassen. Wir löschen im feld Relation das Wort "DESC", und schicken die Änderung ab.



Nun müssen wir im Applet die Maske einmal neu in der Themenauswahl öffnen, dann wird das neue Script aus der Datenbank geladen. Wenn wir dann in der Maske auf "Seit Semester" klicken, erscheinen die Semester in aufsteigender Reihenfolge.



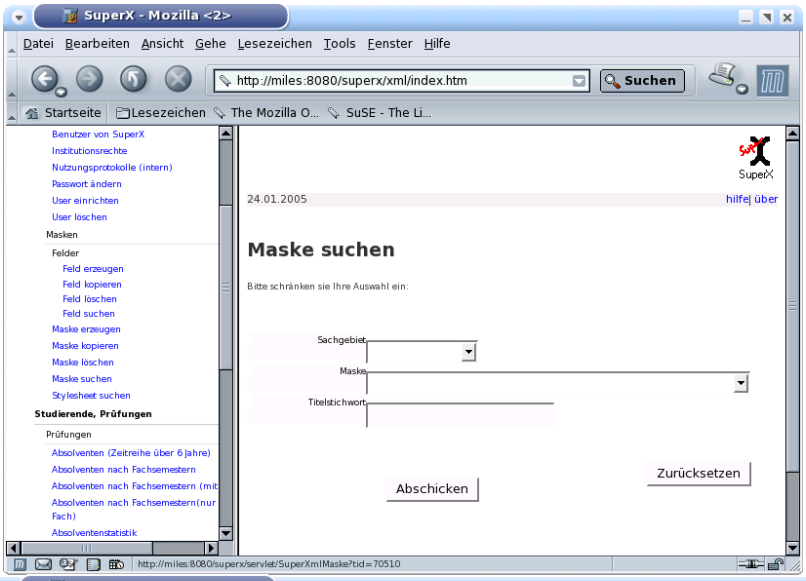
Auf diese Art und Weite können wir alle Maskenfelder bearbeiten. Die restlichen Attribute in der Tabelle `felderinfo` sind im **Adminstrationshandbuch Kernmodul** erläutert.

2.1.1.3 Maskendefinition

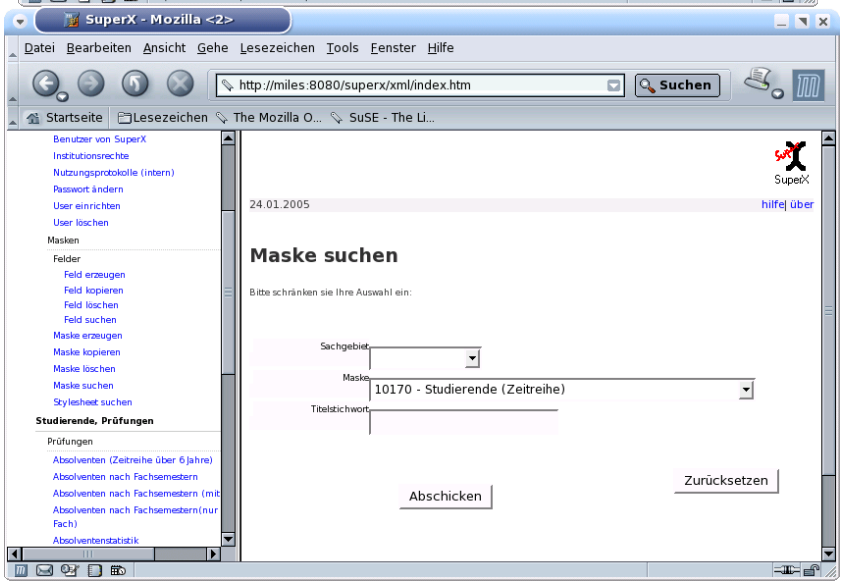
So weit so gut, wir können nun also Felder ändern. Wie können wir nun die Ergebnistabellen bearbeiten?

SuperX arbeitet hier ebenfalls mit SQL-Scripten, die als Felder in einer Tabelle gespeichert sind. Die Tabelle lautet `maskeninfo`. Wir können uns diese Tabelle ebenfalls im XML-Frontend anschauen:

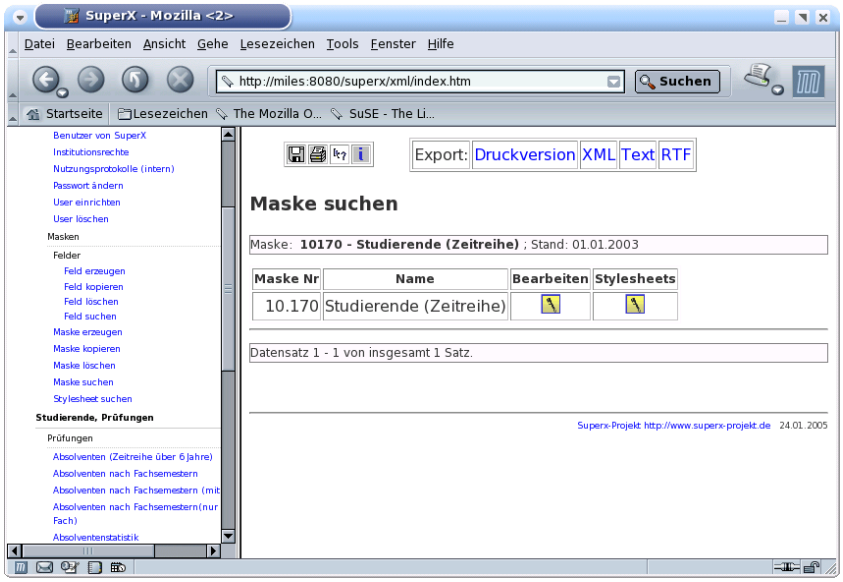
Im Themenbaum wählen wir Maske suchen. Es erscheint rechts eine Auswahlmaske ohne Vorbelegung.



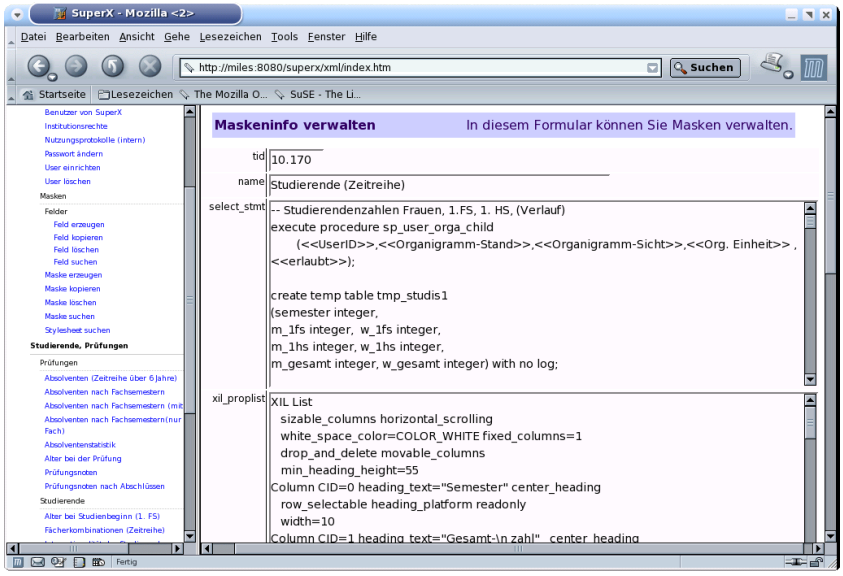
Dort wählen wir wieder die Abfrage **Studierende (Zeitreihe)** aus.



Als Ergebnis sehen wir unsere Maske sowie zwei Bearbeitungsbuttons. Wir wählen den ersten Button, **Bearbeiten**.



Wir gelangen in das Bearbeitungsformular der Maske. Neben der Nummer der Maske sehen wir den Namen und die Felder `select_stmt` und `xil_proplist`. Das Feld `select_stmt` enthält das SQL-Script, und `xil_proplist` die Ergebnisdarstellung.



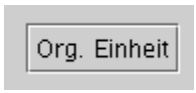
2.1.1.3.1 Abfragen in Maskendefinitionen

Das Script in `select_stmt` ist relativ lang, wir wollen es daher nur Auszugsweise kommentieren. Allgemein formuliert arbeiten wir so:

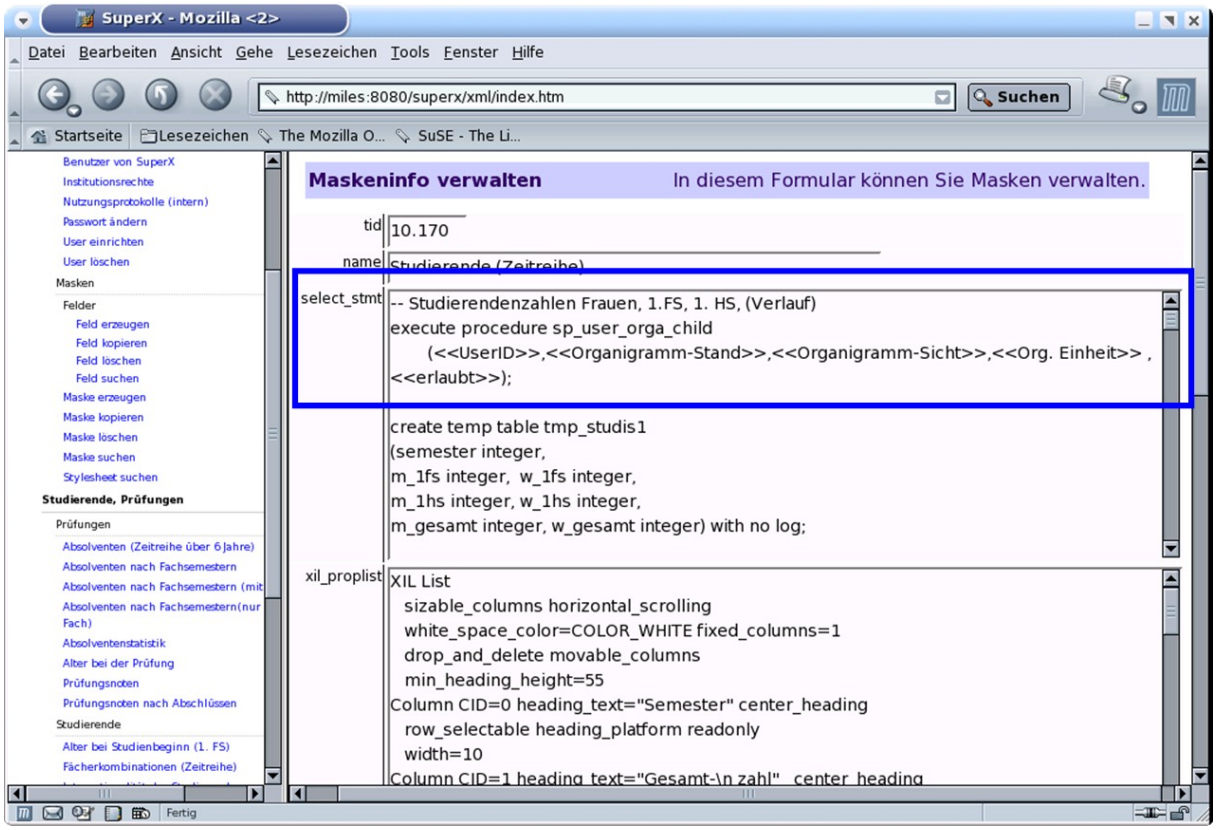
1. Zunächst werden die Eingaben in den Auswahlfeldern ausgewertet und eine Ergebnismenge ermittelt, meist in Form einer temporären Tabelle.
2. Diese Tabelle wird mit den Hilfstabellen in der Datenbank gejoined, und es wir eine Ergebnistabelle berechnet. Ggf. werden noch Summen oder Prozente berechnet, meist benötigen wir dazu weitere temporäre Tabellen.
3. Der letzte select im Feld select_stmt enthält die Ergebnistabelle, die das Applet empfängt. Mit Hilfe der xil_proplist werden die Spaltenüberschriften- und Breiten gesetzt, und das Ergebnis wird angezeigt.

4.Direkt danach wird die letzte temporäre Tabelle gedroppt, und die Datenbankverbindung wird an das SuperX-Servlet zurückgegeben.

Am Anfang eines SQL-Scriptes werden die Auswahlfelder ausgewertet, die der Anwender angeklickt hat, bevor er **Suchen** gedrückt hat. So wird z.B. das Feld:



wie folgt interpretiert:

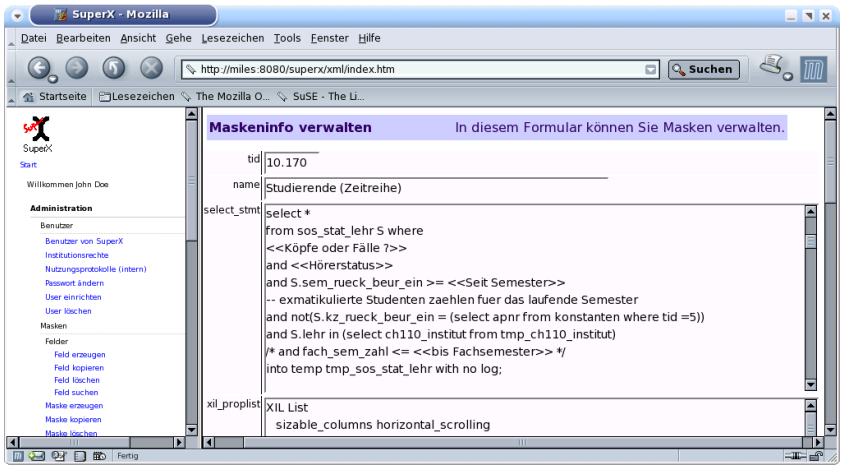


Die Prozedur "execute procedure…" steht am Anfang von fast jeder SuperX-Abfrage und ermittelt eine temporäre Tabelle `tmp_ch110_institut`, die die ausgewählten Lehreinheitsnummern enthält[2]. Konkret wird der Passus "`<<Org. Einheit>>`" durch den Schlüssel ersetzt, der in der Maske ausgewählt wurde. Es handelt sich also bei SuperX-Abfragen um dynamisches SQL.

[2] Das ist grob verkürzt dargestellt, aber im Augenblick für Abfragen im Bereich Studium ausreichend. Die Prozedur ermittelt außerdem noch die Institutionen, zu denen Ein User Leserechte hat. Das versteckte Feld <<UserID>> ist in jeder Maske vorhanden, und die zugehörigen Institutionen (und deren "Kinder") werden aus der Tabelle `user_institution` und `organigramm` ermittelt.

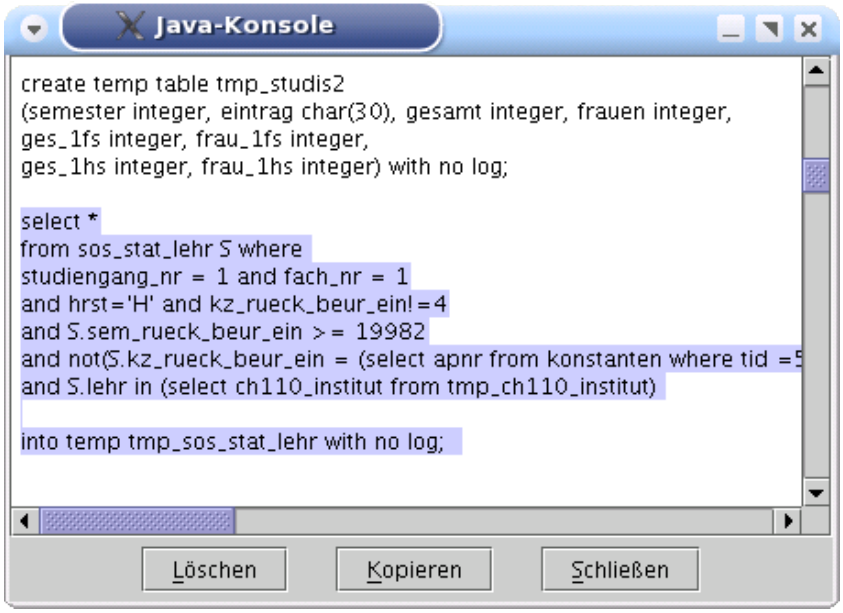
Die Lehreinheiten wiederum werden dann mit der Hilfstabelle `sos_stat_lehr` gejoined, die eine Statistik von allen Lehreinheiten und Semestern enthält[3] . Die Einschränkungen durch die Maskenfelder Köpfe oder Fälle, Seit Semester und Hörerstatus sieht in SQL wie folgt aus:

Für "Köpfe oder Fälle" etc. finden wir hier nur Platzhalter. Alle relevanten Sätze werden in die temporäre Tabelle `tmp_sos_stat_lehr` selektiert (Syntax von Informix, bei Postgres sieht das etwas anders aus).



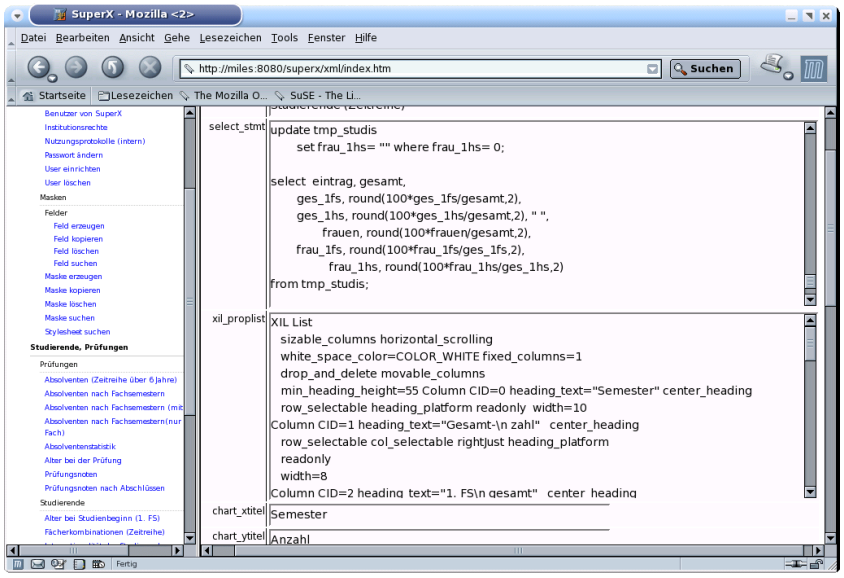
Wenn das Script abläuft, werden die Platzhalter ersetzt. In der Java-Konsole sieht das so aus:

Statt "**<<Köpfe oder Fälle>>**" finden wir den SQL-Ausdruck `studiengang_nr=1 and fach_nr=1`, der **Hörerstatus** ist auf "H" codiert, und die **Beurlaubten** (Status 4) werden ausgefiltert. Das **Semester** muss `>= 19982` sein, die **Exmatrikulierten** (Status 5) werden ebenfalls ausgefiltert, und die relevanten **Lehrheinheiten** werden ausgewählt.



[3] In anderen SuperX-Abfragen wie z.B. Studierende (allgemein) wird auch mit der Tabelle `lehr_stg_ab` gejoined, um die zugehörigen Studiengänge abzurufen - dies brauchen wir in dieser Abfrage nicht.

Der letzte Select enthält die Ergebnistabelle: "select eintrag…".
Das Feld `eintrag` enthält den Volltext des Semesters, `gesamt` die Gesamtsumme, `ges_1fs` die Gesamtsumme der Studierenden im 1.FS etc.
In der XIL-Proplist sehen Sie schon die Überschriften in der Ergebnistabelle.



Ein kurzer Hinweis zur `xil_proplist`: Spaltenüberschriften sind von 0 aufsteigend durchnummeriert, und die Überschriften selbst können durch "\n" mit Zeilenumbrüchen versehen werden. Pro Überschrift wird ein Absatz formuliert, u.a. auch die Breite der Spalte. Alle anderen Angaben werden zur Zeit noch nicht auswertet[4] . Wichtig ist dass eine Spalte pro Absatz definiert ist.

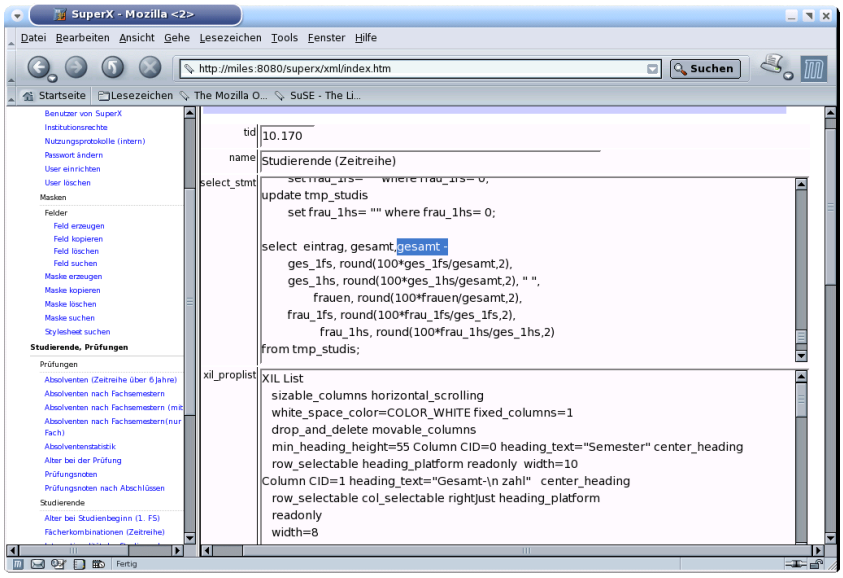
2.1.1.3.2 Änderung einer Abfrage

Stellen wir uns nun vor, will wollten die Abfrage dahingehend ändern, dass wir in der Ergebnistabelle statt der Studierenden im 1. FS die Studierendem im 2.-Xten Fachsemester sehen wollen. Das ist eine recht anschauliche Übung, dazu ändern wir den letzten `select` [5] .

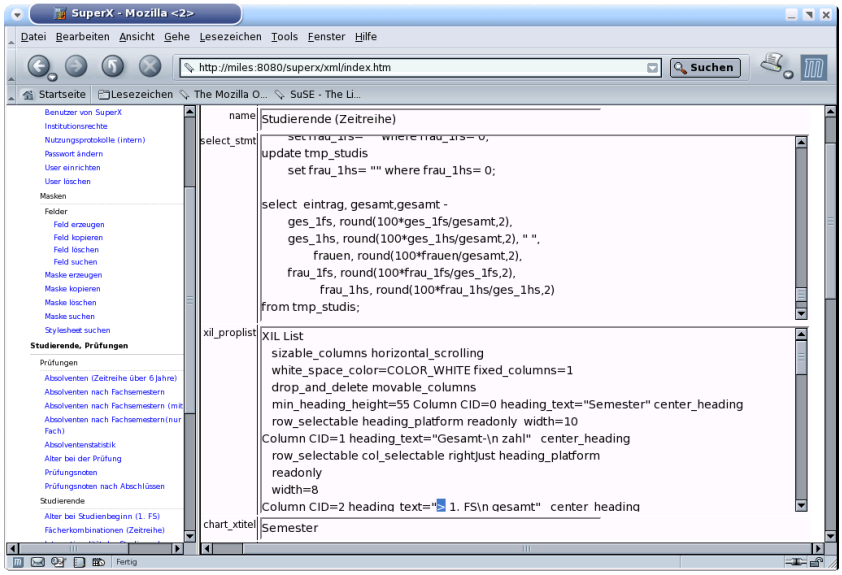
[4] Eine Erblast des alten Win32-Client, wenn wir irgendwann wirklich mal Zeit haben widmen wir uns diesem Problem. Die Syntax ist nicht gerade elegant, ebenfalls ein Überbleibsel vom SuperX-Client 1.x (XVT-Compiler). Aus Gründen der Abwärtskompatibilität weichen wir noch nicht davon ab.

[5] In der Praxis würden wir nicht so arbeiten, sondern die Abfrage zunächst zu einer neuen TID kopieren, und dann ändern.

Wir schreiben vor das Feld `ges_1fs` den Ausdruck "`gesamt -`".

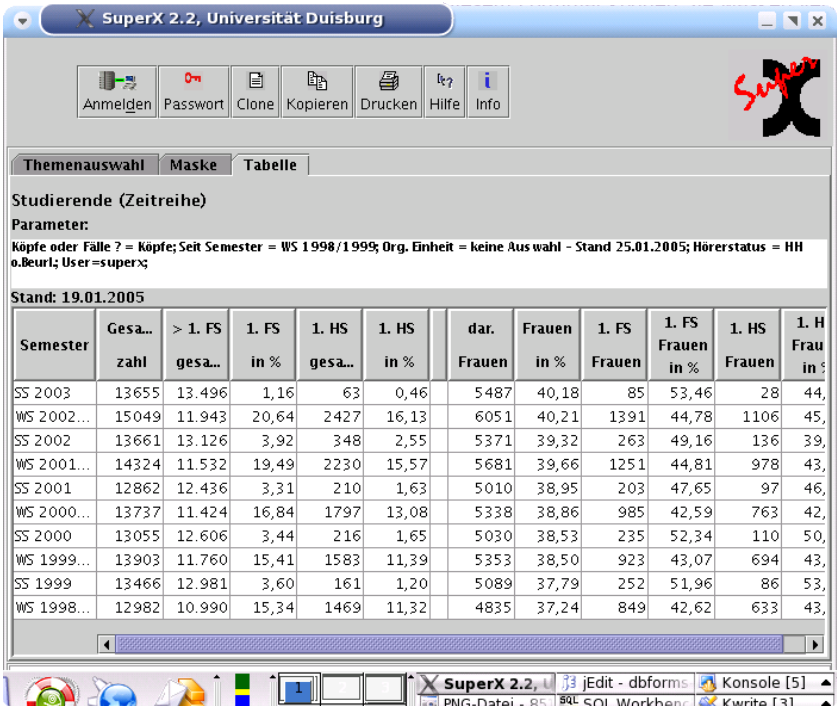


Dann müssen wir die Spaltenüberschrift ändern, Spalte 2 ergänzen wir am Anfang um ein ">"-Zeichen.



Und wenn wir dann die Maske im Themenbaum neu auswählen und ablaufen lassen, erhalten wir folgendes Ergebnis:

Die dritte Spalte zeigt nicht mehr die Studierenden im 1. Fachsemester, sondern den Rest. Alle anderen Spalten haben sich nicht verändert.



Semester	Gesa... zahl	> 1. FS gesa...	1. FS in %	1. HS gesa...	1. HS in %	dar. Frauen	Frauen in %	1. FS Frauen	1. FS Frauen in %	1. HS Frauen	1. H Frau in
SS 2003	13655	13.496	1,16	63	0,46	5487	40,18	85	53,46	28	44,
WS 2002...	15049	11.943	20,64	2427	16,13	6051	40,21	1391	44,78	1106	45,
SS 2002	13661	13.126	3,92	348	2,55	5371	39,32	263	49,16	136	39,
WS 2001...	14324	11.532	19,49	2230	15,57	5681	39,66	1251	44,81	978	43,
SS 2001	12862	12.436	3,31	210	1,63	5010	38,95	203	47,65	97	46,
WS 2000...	13737	11.424	16,84	1797	13,08	5338	38,86	985	42,59	763	42,
SS 2000	13055	12.606	3,44	216	1,65	5030	38,53	235	52,34	110	50,
WS 1999...	13903	11.760	15,41	1583	11,39	5353	38,50	923	43,07	694	43,
SS 1999	13466	12.981	3,60	161	1,20	5089	37,79	252	51,96	86	53,
WS 1998...	12982	10.990	15,34	1469	11,32	4835	37,24	849	42,62	633	43,

2.1.2 Konventionen

Für die Erstellung von SQL-Abfragen gibt es ein paar wichtige Konventionen:

•wenn Sie Abfragen schreiben, sollten Sie temporäre Tabellen immer mit "tmp_" vorab benennen. Grund: es könnte mal sein, dass wir die Tabelle auch als statische Tabelle nutzen. Dann gibt es einen SQL-Fehler.

•vermeiden Sie bei temporären Tabellen Umlaute in Spaltennamen (z.B. "Fakultät"). Das klappt zwar unter Informix, aber nicht unter Postgres.

•Nutzen Sie, wenn möglich, die ANSI-SQL-Standards. Postgres ist hier standardkonformer als Informix. Hinweise zu Unterschieden zwischen Postgres und Informix finden Sie hier: http://www.super-ics.-de/superx/postgres/f_StandardSQLUntersttz.htm

2.1.3 Fazit

So viel zu unserem Einstieg in die Abfragengestaltung. Wir sehen, dass mit den Bordmitteln der Datenbank (Stored Procedures) und dem dynamischen SQL fast beliebige Statistiken generierbar sind. Die hohe Flexibilität erkaufen wir uns mit einer recht hohen Hürde beim Einstieg, außerdem ist die Arbeit nicht gerade "visuell".

Nützliche Hilfsmittel sind SQL-Generatoren wie z.B. die SQLWorkbench von Thomas Kellerer (www.kellerer.org); bitte beachten Sie dabei, dass die Syntax der zugrunde liegenden Datenbanken berücksichtigt werden muss.

Der nächste Schritt für Sie wäre:

•Gestaltung von Abfragenlayouts im XML-Frontend

•Erkunden weiterer Feldarten- und Typen (z.B. Text- und Datumsfelder), insbes. unseren "Organigramm-Button" für Auswertungen im Bereich Haushalt, Personal etc.

•Zuordnen neuer Felder zu Masken bzw. Entfernen von Feldern

•Entwerfen von Stored Procedures

2.2 Erweiterte Maskenprogrammierung: Freemarker Templates

Mit SuperX 3.0 wird eine stark erweiterte Möglichkeit zur Abfragenentwicklung eingeführt.

Die OpenSource-Bibliothek *FreeMarker* (www.freemarker.org) wird als Template-Engine eingesetzt.

Damit Sie in einer Abfrage die Freemarker Funktionalität benutzen können, muss im Kopf des `select_stmt` eine Hinweiszeile

--FREEMARKER TEMPLATE

enthalten sein.

2.2.1 Klassische Verarbeitung

Die einzelnen Abfragen (auch synonym Masken genannt) enthalten SQL Befehle Platzhalter.

z.B.

```
select monat,sum(betrag) from cob_busa
where  monat=<<Monat>>.
```

Auf der Maske gibt es ein Feld Monat. Vorm Abschicken des SQL wird <<Monat>> durch den gewählten Wert ersetzt.

Ausdrücke die zwischen /* und */ stehen, werden entfernt, falls kein Wert ausgewählt wurde.

Wenn man auf einer Maske z.B. optional auf einen Geldgeber einschränken kann.

Aus

z.B.

```
select monat,sum(betrag) from cob_busa
where  monat=<<Monat>>.
/* and gege=<<Geldgeber>> */
```

wenn kein Geldgeber ausgewählt wurde

```
select monat,sum(betrag) from cob_busa
where  monat=1
```

wenn aber ein Geldgeber ausgewählt wurde statt dessen

z.B.

```
select monat,sum(betrag) from cob_busa
where  monat=1
and gege=3;
```

Achtung:

Der Ausdruck in `<<XXX>>` darf nur einmal in dem optionalen Block vorkommen.

Falls er zweimal benötigt wird, muss es auf zwei Blöcke aufgeteilt werden.

z.B.

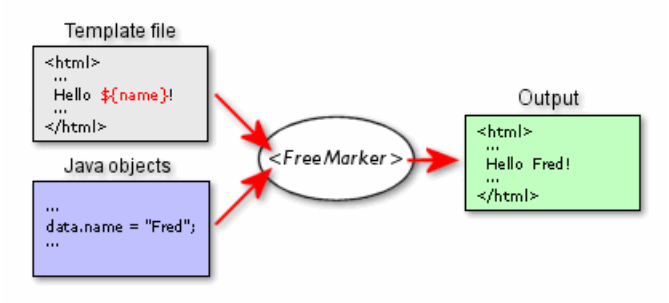
```
/* and (dr in (<<Deckungsring>>) */
/* or dr2 in (<<Deckungsring>>))*/
```

2.2.2 FreeMarker Transformation

2.2.2.1 Übersicht

Nach der klassischen Transformation mit generateSql folgt ggfs. die FreeMarker Transformation.

FreeMarker transformiert eine Vorlage (template) mit Hilfe eines Datenmodells (mit Java Objekten) zu einem Ausgabetext.



Sehr oft wird es zur Erzeugung von HTML benutzt, wir produzieren statt dessen SQL.

Die Java-Objekte im Datenmodell sind die Felder, die auf der Maske zur Auswahl stehen.

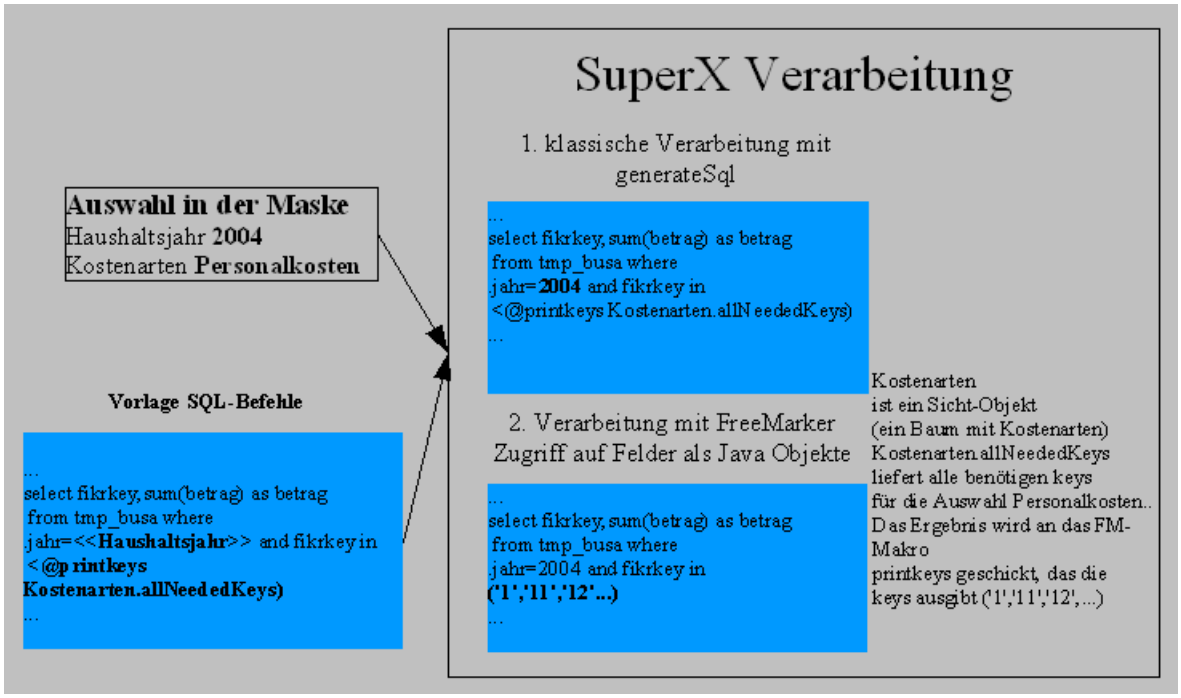
Als einfachsten Anwendungsfall könnten wir also für eine Maske mit einem Monatsfeld statt des klassischen SuperX-Tags

```
select monat,sum(betrag) from tmp_busa where monat=<<Monat>>
```

auch die FreeMarker Notation nehmen.

```
select monat,sum(betrag) from tmp_busa where monat=${Monat}
```

Ein komplexes Beispiel:



2.2.2.2 Interaktion Freemarker mit Maskenfeldern

Das Servlet verarbeitet bei der Maskenausführung zuerst die Maskenfelder, die mit den Steuerungszeichen "<<" und ">>" in das `select_stmt` eingefügt werden.

So wird z.B. bei der Anweisung

```
<#if (einFach.strukturStr = "Fach (intern)" && <<Aggregierung Fach>>=10 )>
```

im Servlet zunächst die Variable "Aggregierung Fach " aufgelöst, und dann erst startet der Freemarker-Parser. Für das o.g. Statement würde also zunächst:

```
<#if (einFach.strukturStr = "Fach (intern)" && 10=10 )>
```

ersetzt, und dann die Freemarker-IF-Bedingung ausgewertet.

So können Sie z.B. auch Checkboxen auswerten, wenn sie nicht angekreuzt sind: nehmen wir an das Feld heißt "nur aktuelle Stg." und hat den Typ "char" und die art=10, dann wäre der Code:

```
<#if "<<nur aktuelle Stg.>>" ="">
select 'Beispiel: nur aktuelle Stg. nicht angekreuzt' from xdummy;
</#if>
```

Wenn das Feld nicht angekreuzt ist, macht der SuperX-Parser daraus zunächst

```
<#if "" ="">
```

und Freemarker würde diese if-Bedingung mit "wahr" beantworten und den SQL `"select 'Beispiel: nur aktuelle Stg. nicht angekreuzt' from xdummy;"` ausführen.

Wenn das Feld angekreuzt wird, macht der SuperX-Parser daraus zunächst

```
<#if "'true'" ="">
```

und Freemarker würde diese if-Bedingung mit "falsch" beantworten, und den SQL `"select 'Beispiel: nur aktuelle Stg. nicht angekreuzt' from xdummy;"` nicht ausführen

2.2.2.3 Programmieren mit FreeMarker

FreeMarker unterstützt praktisch alle Konzepte klassischer Programmiersprachen.

Die Tags sind HTML-ähnlich.

2.2.2.3.1 Zugriff auf Java-Objekte im Datenmodell

Generell geht der Zugriff auf Java-Objekte im Datenmodell mit der Notation

`${varname}.`

z.B. -- ausgewählte Aggegrierung: `${Aggregierung}`

innerhalb von FreeMarker-Befehlen muss `${}` weggelassen werden

```
<#if Aggregierung="stark">
<@printkeys Kostenarten.allNeededKeys/>

Bei Sichten ist unter dem ${Kostenstellen} das Field-Objekt mit verschiede-
nen Methoden enthalten.
Bei Feldart 1 ist unter ${Aggregierung} die ausgewählten Werte, wenn man auf
das Field-Objekt Zugriff haben möchte, kann man dies mit
 ${AggregierungObject} tun. Man kann z.B. abfragen, ob eine Auswahl bei Ag-
gregierung möglich ist mit <#if AggregierungObject.containsElements> (ab
Kern4.5)
oder ab kern4.5 ${Aggregierung.allNeededKeys} wie bei
```

2.2.2.3.2 if-Abfragen

Mit Freemarker können Sie z.B. **if-then-Abfragen** in normales SQL einbauen, z.B. um je nach gewünschter Aggregierungsstufe einen unterschiedlichen insert zu benutzen

if-then in normalem SQL

```
<#if "<<Aggregierung>>"="stark">
insert into .. select ...
<#elseif "<<Aggregierung>>="mittel">
insert into .. select ...
<#else>
insert into .. select ...
</#if>
```

Der klassische SuperX-Tag `<<Aggregierung>>` wird vor der FreeMarker Transformation ersetzt, sodass FreeMarker effektiv zwei Strings vergleicht (`if "stark"="stark"`).

Alternativ könnte den ausgewählten Wert des Felds Aggregierung im Java-Objekt direkt ansprechen.

```
<#if Aggregierung="stark">
```

Hier braucht kein ${} um Aggregierung, da wir ja schon innerhalb einer FreeMarker-Anweisung sind.

2.2.2.3.3 Variablen

assign

Mit assign kann man eigene Variabeln definieren.

z.B.

```
<#assign sortnr=0>
<#assign sortnr=sortnr+1>

insert into ... values (${sortnr},...)


<#if "<<Stichtagsbezogen>>"="NEIN">
  <#assign quelltabelle = "sos_statistik">
<#else>
  <#assign quelltabelle = "sos_statistik">
</#if>
```

select .... from ${quelltabelle} where ...

`<<Stichtagsbezogen>>` wird von der SuperX-Transformation ersetzt, sodass Freemarker vergleicht:

`<#if "JA"="NEIN">` oder `<#if "NEIN"="NEIN">`

Auf alle Felder einer Maske kann man neben der klassischen SuperX-Notation `<<FELDNAME>>` auch per FreeMarker zugreifen. Es ginge z.B. auch

```
<#if Haushaltsjahr=2005>...
insert into ... select ... from … where jahr=${Haushaltsjahr}
```

(Außerhalb von FreeMarker-Anweisungen, muss die Variable mit ${} umschlossen sein, wenn Leerzeichen oder Sonderzeichen vorkommen muss man die .vars-Notation benutzen,

z.B. `"alles aufsummieren?")`)

Wenn man wissen möchte, ob eine Variable mit Inhalt gefüllt ist, kann man dies mit has_content Abfragen, z.B.

```
<#if lehr_abg?has_content >
```

Folgender Effekt ist schon mal aufgetreten:

Wenn man ein einer Abfrage z.B. schreibt

```
<#assign sortnr=sortnr+1>
insert into tmp_rs_base
(struktur,text,  ch30_fach, kz_fach, fach_nr, ch35_ang_abschluss,  sortnr,
...
```

dann klappt das nicht, man muss unter dem assign eine Leerzeile machen:

```
<#assign sortnr=sortnr+1>

insert into tmp_rs_base
(struktur,text,  ch30_fach, kz_fach, fach_nr, ch35_ang_abschluss,  sortnr,
```

Für Fortgeschrittene: sqlvars

Manchmal wünscht man sich mit FreeMarker auf Variablen zugreifen zu können, die aus der Datenbank gefüllt werden müssten, weil sie nicht als Felder auf der Maske vorkommen.

Allerdings muss die Freemarker-Transformation ja schon laufen, bevor der fertige SQL an die Datenbank geschickt wird, weil Postgres/Informix ja mit FreeMarker-Befehlen nichts anfangen können.

Man kann also nicht so etwas

```
<#assign gegename="select name from fin_geldgeber G where G.xyz=<<XYZ>>">
```

machen, weil die Variable gegename dann einfach nur select... als String enthält, Freemarker hat mit der Datenbank keinen direkten Kontakt.

SuperX gibt Abhilfe:

man legt einen Block an

```
<sqlvars>
<!-- einfache Variable-->
<sqlvar name="fin_geldgeber_exists">
select sp_table_exists('fin_geldgeber') from xdummy
</sqlvar>

<!-- Listenvariable -- 1. Spalte key (darf nicht null sein!), 2. Spalte name-->
<sqlvar name="geldgeber">
select ggnr,ggname1 from fin_geldgeber
</sqlvar>
</sqlvars>
```

Anschließend kann man in der Abfrage mit FreeMarker auf diese aus der Datenbank gefüllten Variablen zugreifen:

```
<#if fin_geldgeber_exists=1>
insert into .. select * from fin_geldgeber;
<#else>
insert into .. select * from fin_geldgeber;
</#if>
```

Die Variable wird als einfacher Wert erkannt, weil nur eine Spalte im SQL selektiert wurde.

(Achtung: Sofern der Select mehrere Zeilen liefert, wird nur der letzte gefundene Wert hinterlegt, wenn man abfragen möchte, ob überhaupt etwas in der Datenbank gefunden wurde kann man `<#if fin_geldgeber_exists?has_content>` benutzen)

Die zweite Art von Variable (Geldgeber) wird als Liste von Items geführt und kann in der Abfrage benutzt werden z.B. mit

```
where ggnr in (
```

```
<#foreach gg in geldgeber>
${gg.key},
</#foreach>
```

Die erste per SQL eingelesene Spalte,ist das Attribut key, die zweite name.

optional kann auch noch dritte/vierte Spalte mit Strukturinfo eingelesen werden

```
<sqlvar name="geldgeber">
select ggnr,ggname1,klr_geldgeber,strukturint from fin_geldgeber
</sqlvar>
```

Auf die dritte Spalte kann man zugreifen über das Attribut ".strukturStr", und die vierte Spalte mit .strukturInt, also für das orbige Beispiel mit:

```
<#foreach gg in geldgeber>
-- Hier steht der Inhalt der Spalte klr_geldgeber: ${gg.strukturStr}
...
```

Analog für strukturInt.

In den sqlvars kann auch freemarker-syntax und repository/konstanten benutzt werden, obstruses Beispiel:

```
<sqlvar name="spezial_gege">
<#if K_FIN_Quellsystem=1>
  select ggnr,ggname1 from fin_geldgeber where ggnr in ${FIN_Drittmittel}
<#else>
  select ggnr,ggname2 from fin_geldgeber where ggnr in ${FIN_Drittmittel}
</#if>
</sqlvar>
```

Achtung: bei den SQL-Statements innerhalb von SQLVAR-Abschnitten dürfen Sie keine "<" oder ">"-Zeichen nutzen, sondern mit der html-Notation < und > umschreiben, also statt:

```
<sqlvar name="Semester">
select distinct tid, eintrag
from semester
where tid <=20101
order by 1;
</sqlvar>
```

schreiben Sie besser:

```
<sqlvar name="Semester">
select distinct tid, eintrag
from semester
where tid &lt;=20101
order by 1;
</sqlvar>
```

Bei Problemen mit sqlvars, deren SQL dynamisch generiert werden, ist es schwierig, sich den produzierten SQL anzusehen (insb. wenn nur Browserzugriff besteht).

Beispiel

```
<sqlvar name="hhanssum"><![CDATA[
```

```
select sum(hhans) from fin_konto_aggr where
   rechnungsjahr= <<Haushaltsjahr>>
and ch110_institut in ${Kostenstelle.allNeededElements}
</sqlvar>
```

Trick: Einen SQL-Syntaxfehler einbauen.

```
<sqlvar name="hhanssum"><![CDATA[
select XXXXX sum(hhans) from fin_konto_aggr where
 rechnungsjahr= <<Haushaltsjahr>>
and ch110_institut in ${Kostenstelle.allNeededElements}
</sqlvar>
```

Dadurch wird ein SQL-Fehler erzeugt, der im Browser angezeigt wird und man kann sehen, wie <<Haushaltsjahr>> ersetzt wurde und Freemarker arbeitet.

Man kann auch Inhalt von Freemarker-Variablen bzw. vorhergehenden SQLVars anzeigen lassen. Beispiel Konstante FIN_Quellsystem und vorhergehende sqlvar instname:

```
<sqlvar name="instname">select drucktext from fin_inst where key_apnr=<<Kostenstelle>></sqlvar>
<sqlvar name="hhanssum"><![CDATA[
 select '${K_FIN_Quellsystem} - ${instname}' from xdummy;
select XX sum(hhans) from fin_konto_aggr where
rechnungsjahr= <<Haushaltsjahr>>
and ch110_institut in ${Kostenstelle.allNeededElements}
</sqlvar>
```

neu:

```
in sqlvars kann man auch auf die vorherigen sqlvars zugreifen
<sqlvars>
<sqlvar name="v1">select ..</sqlvar>
<sqlvar name="v2"> <![CDATA[select .. where <#if v1=1> feld=1<#else>feld=2</#if>]]></sqlvar>
</sqlvars>

geht bisher nur innerhalb von if-Anweisungen o.ä.
CDATA ist wichtig, damit wohlgeformtes xml

<sqlvar name="COB_FIN_STARTJAHR">select apnr from konstanten where beschreibung='  COB_FIN_STARTJAHR
'</sqlvar>
<sqlvar name="vorhandene_fin_monate"><![CDATA[select distinct klrjahr*12+klrmonat,'monat' from
fin_buch_akt where klrjahr>0 and klrjahr>=${COB_FIN_TO_BUSA_STARTJAHR}]]></sqlvar>

hier kommt Fehlermeldung smallint Operator >= existiert nicht
Hintergrund
${COB_FIN_TO_BUSA_STARTJAHR} wird als $-Kommentar vom klassischen general-sql gelöscht und
select distinct klrjahr*12+klrmonat,'monat' from fin_buch_akt where klrjahr>0 and klrjahr>=
abgeschickt
```

neu typ: hash / hashsequence

```
<sqlvar name="kennzahl" type="hash">\
select id, shortname,sqlchunk,calcratio,decimalplaces,linksub,linktimeline from man_catalogue \
where id=<<Kennzahl>>;\
</sqlvar>
```

kann beliebig viele Spalten enthalten, bei concat as benutzen wie id||'-'||name as bezeichnung

auf diese kann man später so zugreifen

t ${kennzahl.shortname} .

```
 Falls mehrere Werte gefunden werden können benutzt man am besten type='hashse-
quence' (neu Mai 2013)

 man kann zwar auch type='hash' benutzen, falls aber weniger als zwei Einträge ge-
funden werden und man ein foreach macht:
 <#foreach entry in entries>
 ${entry.id}','${entry.shortname}','${entry.description} '${entry.bezeichnung}
```

kommt der fehler:

Expected collection or sequence. auswertungids evaluated instead to de.superx.common.TemplateProcessor$SxHash

```
also am besten, wenn nur ein wert gefunden werden kann type='hash', wenn es mehrere sein können
type='hashsequence'


Neu in Kern4.2
Man kann sich auch in einem SQL-File, das per DOSQL ausgeführt wird, eine Sicht bauen.
Im select_stmt von Masken  geht dies nicht, da gibt es ja Sichten ja schon als Feld.
Beispiel:
<sqlvars>
<sqlvar name="auswertungsdefinition" type="sicht" name_intern="fibu_auswertung_def"
stand="1.1.1900"/>
</sqlvars>
- (Stand ist optional als default wird today genommen)

<#foreach def in auswertungsdefinition.elements>
${def.name} ${def.key}
</#foreach>
```

SuperX Konstanten

Es gibt in SuperX Konstanten, welche in der Maskenprogrammierung immer zur Verfügung stehen.

in HttpSession für Zugriff in jsp ist

- username
- UserIsAdmin (true/false),
- UserMaskRights z.B. ,23000,23030,23050,
- UserGroups z.B. ,2,3,4,
- UserGroupnames z.B. ,Administration,Dezernent,Test,

(Am Anfang und Ende ist jeweils ein Komma, dann kann z.B. Rechte prüfen

mit indexOf(",23000,") und bekommt keine Probleme wenn es eine Maske

523000 gibt)

Ähnlich und noch komfortabler auch in Freemarker

- ${username},
- ${UserMaskRights} z.B. ,23000,23030,23050,
- ${UserGroups} z.B. ,2,3,4,
- ${UserGroupnames} z.B. ,Administration,Dezernent,Test,

```
<#if UserIsAdmin>admin</#if>
<#if UserHasMaskRight("17070,23000")> ERLAUBT<#else>VERBOTEN </#if>
--Klammer kann ein oder mehrere Maskennummern enthalten
<#if IsUserInGroup("2")> in ADMIN-GROUP<#else> nicht in
ADMIN-group</#if> -- eine Gruppe in Klammer
<#if IsUserInGroupWithName("Administratoren")> NAME ADMIN-GROUP<#else>
NAME nicht in ADMIN-group</#if>
 --eine Gruppe in Klammer
<#if IsUserInAtLeastOneGroup("3,2,4")>
```

user ist in mindestens einer

Gruppe <#else> user ist in keiner der angegebenen Gruppen</#if>

```
<#if UserHasSachgebRight("152")>
```

neu in kern44 ab Nov 2014

```
<#if UserHasAllKostenstellenRights> ...
```

2.2.2.3.4 has_content

Wenn man wissen möchte, ob eine Variable mit Inhalt gefüllt ist, kann man dies mit has_content Abfragen, z.B.

```
<#if lehr_abg?has_content >
```

2.2.2.3.5 ForEach

FreeMarker kann nicht nur primitive Datentypen wie Strings oder Zahlen verarbeiten, sondern auch Collections. Wenn im Datenmodell eine Collection hinterlegt ist, kann man forEach benutzten.

Das sieht ungefähr so aus

```
<#foreach eineKostenart in Kostenarten.elements>
-- Auswertung für ${eineKostenart}
</#foreach>
```

Details siehe bei Sichtfeldern-Schleifen.

2.2.2.3.6 For …Next …-Schleifen: List

FreeMarker kann auch eine For-Next-Schleife mit 1-er Schritten erzeugen.

Das sieht z.B. so aus:

Erzeugung einer temp. Tabelle mit Altersintervallen m_a18, …,m_a20,…

```
create temp table tmp_aggre \
(struktur char(50),text char(200), ch30_fach char(3),sortnr
int,
 <#list 0..30 as i>
m_a${18+(i*2)} decimal(7,2),
w_a${18+(i*2)} decimal(7,2),
</#list>
gesamt decimal(7,2));
```

ergibt nach Freemarker-Transformation

```
create temp table tmp_aggre
(struktur char(50),text char(200), ch30_fach char(3),sortnr
int,
m_a18 decimal(7,2),
w_a18 decimal(7,2),
m_a20 decimal(7,2),
w_a20 decimal(7,2),
m_a22 decimal(7,2),
w_a22 decimal(7,2),
[…]
m_a78 decimal(7,2),
w_a78 decimal(7,2),
gesamt decimal(7,2));
```

2.2.2.3.7 Freemarker in dynamischen Felderrelationen

Damit ein Feld Freemarker Variablen auslesen kann wie Kostenstelle, muss es als dynamisch markiert sein. SuperX erkennt ein Feld als dynamisch wenn << darin vorkommt.

Daher

```
<<SQL>>
--Freemarker Template
<#include "SQL_lingua_franca"/>
<#include "SuperX_general"/>\
select distinct buchungsab_fb,trim(buchungsab_fb)||'-'||max(ba_name) from
fin_used_inst\
where kostenstelle in <@printkeys Kostenstelle.allNeededKeysList />
```

klappt nicht, aber

```
<<SQL>>
--Freemarker Template
<#include "SQL_lingua_franca"/>
<#include "SuperX_general"/>\
select distinct buchungsab_fb,trim(buchungsab_fb)||'-'||max(ba_name) from
fin_used_inst\
```

```
where kostenstelle in <@printkeys Kostenstelle.allNeededKeysList />
/*  --quatsch <<Kostenstelle>> */
```

2.2.2.3.8Makros und Funktion

Es könnten auch Makros und Funktionen (die Werte zurückliefern) definiert werden.

```
<#macro macroname param1 param2>
</#macro>
```

Auf gerufen werden sie mit

<@makroname> bzw. <@makroname param1 param2 ..>

Die Reihenfolge in der Makros definiert werden spielt keine Rolle.

Einige Makros für Datenbankunabhängigkeit (SQL lingua franca) und allgemeine Makros sind in der Tabelle `fm_templates` hinterlegt.

Diese können mit

<#include "xx"/> eingebunden werden.

2.2.2.4 Special tricks

Kostenarten name = xxx-Name

nur Namen ausgeben:

```
'${eineKostenart.name[(eineKostenart.name?index_of('-')+1)..(eineKostenart.name?length-1)]}'
```

```
<#assign maxEbene=99>
<#if "<<Filter bis Ebene>>"!="">
/* <#assign maxEbene=<<Filter bis Ebene>>> */
</#if>
```

```
Bei Feldart 1 z.B. Buchungsart - Name des selektierten Items auswerten
<#if BuchungsartObject.selectedItems[0].name='Bewilligung'><#assign tab_name='fin_buch_man'/></#if>
<#if BuchungsartObject.selectedItems[0].name='verfuegbares budget'><#assign
tab_name='fin_indiv_wert'/></#if>
<#if BuchungsartObject.selectedItems[0].name='Drittmittelsperre'><#assign
tab_name='fin_indiv_wert'/></#if>
```

```
<<SQL>> /* execute procedure sp_user_orga_child(<<UserID>>,mdy (1, 1,year(<<Organigramm-
Stand>>)),<<Organigramm-Sicht>>,<<Institution>>, <<erlaubt>>); --notinxmlfrontend */
```

```
/* execute procedure sp_user_orga(<<UserID>>,mdy (1, 1,year(<<Organigramm-Stand>>)),<<Organi-
gramm-Sicht>>); create temp table tmp_ch110_institut (ch110_institut char(10)); insert into
tmp_ch110_institut select key_apnr from tmp_organigramm; drop table tmp_organigramm;  --notinapplet
*/
select distinct fb, fb || " - " || ktobez from mbs_projekte_konto where 1=1 and inst_nr in (select
ch110_institut from tmp_ch110_institut) /* and dr in (<<Deckungsring>>)  */ /* and jahr = <<Haush-
altsjahr>> */ /* and kap in (<<Kapitel>>) */ order by 2;
```

Achtung - Absatzmarke scheint auch wichtig zu sein

Eurozeichen € in Namen

Eurozeichen in Namen können Probleme mit XSL machen, daher am besten namen mit Freemarker setzen und replace vornehmen. Beispiel:

```
<sqlvar name="sachkontonamen"> select sachkonto,bezeichnung from fin_fibu_kont whe-
re haushaltsjahr::integer&lt;=<<Jahr>> and (hhjende is null or hhjende::integer &gt;=
<<Jahr>> );</sqlvar>

create index ix_tmp_erg1 on tmp_erg (issachkonto, sachkonto);
   <#foreach s in sachkontonamen>
   update tmp_erg set name='${s.name?replace("€"," EUR")?replace("\x20AC"," EUR")?
replace("\x0080"," EUR")}' where issachkonto=1 and sachkonto='${s.id}';

   </#foreach>
```

Neu in kern4.5

Einen Defaultwert setzen nur wenn der auch in der Auswahlliste vorkommt.

<<SQL>> select apnr,apnr||'-'||druck from gxstage_cifx where key=3 and apnr='9999' and <#if .vars["Haushaltsprog./InnenauftragObject"]("containselementwithid","9999")> 1=1<#else> 1=0</#if>;

2.2.3 neue Funktionen 2011: UserRights Dbversion

```
UserMaskRights Komma getrennte Liste, am Anfang auch führendes Komma
UserHasMaskRight
UserIsAdmin boolean
UserName
UserGroups --tids
UserGroupnames -namen
IsUserInGroup (tid) boolean
IsUserInGroupWithName (name) boolean
IsUserInAtLeastOneGroup boolean
today als String
yeartoday aktuelles Jahr
databaseMajorVersion int
databaseMinorVersion int
```

```
Beispiele:
select '${UserMaskRights}' from xdummy;
select '${today}' from xdummy;
select '${yeartoday}' from xdummy;
select '${databaseMajorVersion}' from xdummy;
select '${databaseMinorVersion}' from xdummy;
select '${Username}' from xdummy;
select '${UserGroups}' from xdummy;
select '${UserGroupnames}' from xdummy;

<#if UserHasMaskRight(23230)> select 'ja' from xdummy; <#else> select 'nein' from
xdummy</#if>
<#if UserIsAdmin()||IsUserInGroup(4)||IsUserInGroupWithName('Administratoren')||IsU-
serInAtLeastOneGroup()>

<#macro truncate table>
<#if SQLdialect='Postgres'||(SQLdialect=='Informix'&&databaseMajorVersion&gt;10)>
truncate table ${table};
<#else>
delete from ${table};
</#if>
</#macro>

<@truncate table='fin_fikr_neu'/>


Teilweise auch in session verfügbar für JSP,
z.B.
String maskRights=request.getSession().getAttribute("UserMaskRights").toString();
if (maskRights.indexOf(",14571000,")>-1)
{hasFinRights=true;}
```

2.2.3.1 SQL-Lingua Franca

Um Abfragen nicht separat für Informix/Postgres entwickeln zu müssen, gibt es FreeMarker SQL-Lingua Franca[6] Makros.

Einfaches Beispiel:
Aufruf einer Prozedur

[6] aus der Sprachwissenschaft – Verständigungssprache für Sprecher verschiedener Sprachen, prominentestes Beispiel: Englisch

Informix execute procedure sp_proc();
Postgres select sp_proc();

FreeMarker **<@procedure sp_proc/>;** wird transformiert →	Informix execute procedure sp_proc();
	Postgres select sp_proc();

procedure ist ein FreeMarker Makro, das definiert ist in der Tabelle fm_templates, id "SQL_lingua franca"

`<@informixnolog/>`
wenn Informix als Datenbanksystem benutzt wird, wird der Zusatz *with no log* ausgegeben.

selectintotmp

Informix und Postgres unterscheiden sich darin, wie ein select into temp table aufgebaut ist.

Beispiel

Informix

```
select key,sum(betrag) from cob_busa where ... group by …. into temp
tmp_busa ;
```

und

Postgres

```
select key,sum(betrag) into temp tmp_busa from cob_busa where ... group by
…;
```

Um dieses datenbankunabhängig zu halten, gibt es das Makro selectintotmp mit den parametern select (die Spalten), source (Quelltabelle/n) und target (Zieltabelle).

Eine gegebenenfalls nötige where-Bedingung und group by muss als "Body" innerhalb des selectintotmp-Aufrufs angegeben werden.

Die obigen Beispiele können mit FreeMarker mit folgendem Makroaufruf automatisch erzeugt werden.

```
<@selectintotmp select="key,sum(betrag)" source="cob_busa"
target="tmp_busa">
```

```
where ... group by ..
</@selectintotmp>
```

2.2.3.2 Allgemeine FM-Makros/Funktionen

Es gibt eine Reihe von allgemeinen Makros, die für alle Abfragen interessant sind.

Standardmäßig werden Kommentare von der klassischen SuperX-Verarbeitung (generateSql) gelöscht, damit nicht soviel an die DB geschickt wird.

Wenn man aber bei Entwicklungszwecken noch Kommentare drin haben will, kann man das Makro addcomment benutzen.

Da FreeMarker erst nach dem klassischen generateSql ausgeführt wird, kann man Kommentare wieder einfügen.

```
<@addcomment comment="Hier wirds interessant"/>
```

printkeys

Druckt die Schlüssel aus.

```
<@printkeys Kostenarten.allNeededKeys/>
```

schreibt z.B. wenn Personalkosten ausgewählt wurde ('1','11','12',...)

Entwicklungsmöglichkeit:

Man könnte bei Bedarf gewisse "künstliche" Schlüssel raus filtern.

2.2.3.3 Spezielle Möglichkeiten bei Sicht-Feldern

Bei Sicht-Feldern (Feldart 12) gibt es besondere Möglichkeiten. Mittels Java-Reflection kann FreeMarker auf Methoden der Objekte im Datenmodell zugreifen.

Bei Sichtfeldern wie Org. Einheit oder Kostenart sind folgende Methoden interessant.

2.2.3.3.1 allNeededKeys – für temporäre Datentabellen

Diese Methode liefert alle benötigten Schlüssel.

Wenn bei Org. Einheit oder Kostenart nichts ausgewählt wurde, werden einfach alle im Baum vorhandenen Schlüssel geliefert.

Wenn z.B. Personalkosten ausgewählt wurde, wird nur der Schlüssel von Personalkosten ('1') und dessen Unterknoten (z.B. '11','12') geliefert.

Dafür wird noch das allgemeine Makro printkeys benutzt.

```
<@printkeys Kostenarten.allNeededKeys/>
```

Beispiel für Erstellung einer temporären Datentabelle

```
execute procedure sp_user_orga_child
     (<<UserID>>,<<Organigramm-Stand>>,0,<<Institution>>, <<erlaubt>>);
 Create temp table tmp_erg  (fikr varchar(200), betrag decimal (14,2)) with
no log;



select fikrkey,sum(betrag) as betrag from cob_busa B,tmp_ch110_institut T
where
 B.ch110_institut=T.ch110_institut and
 B.jahr=<<Haushaltsjahr>> and fikrkey in
<@printkeys Kostenarten.allNeededKeys />
 group by fikrkey
  into temp tmp_busa;
```

(Statt Benutzung der Prozedur sp_user_orga_child könnte man analog verwenden:

where B.110_institut in <@printkeys Institition.allNeededKeys/>)

Ggfs. versteckte Knoten werden hier mit ausgegeben.

Bei Kostenstellen-Feldern werden nur erlaubte Einträge ausgegeben.

Neu ab kern4.5 diese Methode kann man auch für Feldart 1-Felder benutzen, z.B. mit

${Haushaltsprogramm.allNeededKeys}.

2.2.3.3.2 ContainsElements

Neu ab Kern4.5 für Sichten und auch Feldart 1, ob eine Auswahl möglich ist.

z.B. <#if Kostenstelle.containsElements> … </#if>

Oder bei Feldart 1 : <#if HaushaltsprogrammObject.containsElements> .. </#if>

2.2.3.3.3keysToRoot – für Verteilschritte

Verteilschritte sind ein ungewöhnliches Konzept.

1

2

3

Schritt 3 ist die Summe aus 1-3.

In Abfragen wird <@printkeys Verteilschritt.keysToRoot/> benutzt

(Ab Kernmodul 3.0rc3)

2.2.3.3.4 elements– für Schleife über ausgewählte Knoten

Diese Methode liefert eine Collection, entweder über alle Knoten im Sichtbaum oder nur über einen ausgewählten Knoten und deren Kinder.

Beispiel:

```
<#foreach eineKostenart in Kostenarten.elements>
-- Auswertung für ${eineKostenart}

</#foreach>
```

elements liefert die Knoten genau in der Reihenfolge, in der sie auch im Baum sind.
Alternativ kann man `breadthFirstElements` oder `depthFirstElements` angeben.
Dann wird beim Baum zunächst in die Breite/Tiefe gegangen.

Wichtig:
Für eine foreach-Schleife werden auch bei Kostenstellen-Feldern bei eingeschränkten Usern immer alle Knoten ausgegeben
z.B. nur Rechte auf 11 und 13
root-Hochschule (Auswahl)
fak-Fakultäten (Auswahl)
1-fak1 (Auswahl)
11-Institut 1
13 – Institut 3
Es werden alle Knoten durchlaufen, weil (Teil)summenzeilen interessant sein können.
Anders ist es bei Berechnung (Methode subkeys) da werden nur die tatsächlich erlaubten Schlüssel ausgegeben.

2.2.3.3.5 Zugriff auf einzelne Knoten im Baum

Im Rahmen einer forEach Schleife bekommt man Zugriff auf einzelne Elemente eines Sichtenbaums.
Für die einzelnen Knoten kann FreeMarker wiederum mittels Java-Reflection auf bestimmte Methoden zugreifen.

id, name – Zugriff auf den Schlüssel und den Namen des Knotens

Insert into tmp_erg (fikr , betrag)
SELECT "${eineKostenart.**id**}" || " " || "${eineKostenart.**name**}", sum(betrag)
FROM tmp_busa
…

subkeys – liefert eine Liste mit dem Schlüssel des aktuellen Knotens (z.B. Personalkosten '1') und aller seiner Unterknoten (z.B. '11','12') (auch von versteckten Knoten!)

Insert into tmp_erg (fikr , betrag)
SELECT "${eineKostenart.**id**}" || " " || "${eineKostenart.**name**}", sum(betrag)
FROM tmp_busa
where fikrkey in <@printkeys **eineKostenart.subkeys** }
group by 1 ;
Bei Kostenstellen-Sichten werden nur die erlaubten Knoten ausgegeben,
z.B.
root-Hochschule (Auswahl)
fak-Fakultäten (Auswahl)
1-fak1 (Auswahl)
11-Institut 1
13 – Institut 3
wenn nichts ausgewählt wurde, root, fak, oder fak1 wird trotzdem nur 11,13 als subkeys ausgegeben, weil nur die selbst erlaubt sind. -> bei foreach (Methode elements ist es anders root, fak, fak1 werden auch mit durchlaufen, weil (Teil)summenzeilen interessant sein können

strukturInt,strukturStr - beschreibt Art oder Struktur des Knotens
Beispiel: orgstruktur im Organigramm beschreibt, ob ein Knoten eine Lehreinheit oder ein Fachbereich ist (20 bzw. 30)
Diese Strukturinformation ist im Knoten hinterlegt, sofern beim Einlesen der Sicht an Postition 4 und/oder 5 etwas angegeben wurde
(z.B. select name,key_apnr,parent,orgstruktur from organigramm).
Sie kann z.B. für if-Abfragen zur Aggregierung benutzt werden.

```
<#foreach eineInstitution in Institutionen>
   <#if Aggregierung="stark" and eineInstitution.strukturInt=30>
   ...
   <#else>
   ..
```

```
    </#if>
 </#foreach>
```

nodeattrib – ein Knotenattribut

wenn null oder 0: Knoten ganz normal, 1 versteckt, 2 nicht selektierbar im Baum

neu in kern4.5 3 = eingerückt bei Feldart 1

Beispiel für relations-SQL für Feldart 1

```
select key,name,0 as nodeattrib from tabelle where name  like '%LFB%'  -- 0 = normale Darstellung
union
select key,name,3 as nodeattrib from tabelle where name not like '%LFB%'  -- 3 = eingerueckt
```

Eine besonderer Trick ist, wenn man einen bestimmten Knoten aus dem Baum braucht, der nicht ausgwählt sein muss: Man nimmt den Feldnamen der Sicht und schreibt z.B.

```
Kostenarten("getSubkeys","21")
```

Dann kriegt man eine Schlüsselliste für alle Schlüssel 21 und untergeordnete.

2.2.3.4 Spezielle Möglichkeiten bei Feldart1 – Auswahlfeldern/Datenblätter

Bei einfachen Auswahlfeldern, z.B. Hörerstatus, kann man im SQL klassisch vorgehen

select ... from... where hrst =<<Hörerstatus>>

Wenn ein Schlüssel ausgewählt wurde, kann auch per Freemarker auf die Variable zugegriffen werden.

select ... from... where hsrt=${Hörerstatus}, wenn nichts ausgewählt wurde, enthält die Variable einen Leerstring.

Spezielle Anforderungen bei Datenblattabfragen

Beim Feld „Felder" sind verschiedene Felder hinterlegt, dort gibt es eine Querabhängigkeit vom ausgewählten Bericht. Je nach Berichtsstylesheet stehen im Art1-Feld „Felder" verschiedene Möglichkeiten zu Auswahl. Wenn der User einen bestimmten Bericht ausgewählt hat, sollen nur die zugehörigen Felder ausgelesen werden.

Dazu gibt es ein zusätzliche FreemarkerVariable <<Feldname>>Object, dass das Field-Objekt enthält sowie eine Methode allKeys(), die eine komma separierte Liste aller Auswahlmöglichkeiten liefert unabhängig davon, ob der User nur etwas ausgewählt hat oder nicht. Ergibt

${FelderObject.allKeys} bzw. ${.vars["Weitere TabellenObject"].allKeys)}

In der Datenblattmaske muss ein recht komplexe Definition von sqlvars mit Abfrage auf gewählte Tabellen erfolgen. z.B:

```
<sqlvar name="get_tables"><![CDATA[
SELECT distinct name,name   from sx_tables  where name in ('sos_stg_aggr'
```

```
   <#if <<tablestylesheet>>='tabelle_html.xsl'>
     /*,<<Weitere Tabellen>> */
     <#else>
    <#if .vars["Weitere TabellenObject"].allKeys?length&gt;0>,${.vars["Weitere Tabel-
lenObject"].allKeys}</#if>
    </#if>    )
   order by 2;]]>
</sqlvar>
```

Das gleiche wie der rote Block erledigt die FreemarkerMethode

```
${DatenblattTables(<<tablestylesheet>>,"<<Weitere Tabellen>>",.vars["Weitere Ta-
bellenObject"].allKeys)}
```

auch für Auswahl von Feldern gab es komplexen SQL in sqlvars

```
<#if <<tablestylesheet>>='tabelle_html.xsl'>
/*  and trim(table_name) || '.' || name in ( <<Felder>> ) */
  <#else>
 and trim(table_name) || '.' || name in (${FelderObject.allKeys})
  </#if>
```

Dies kann abgekürzt durch Freemarker-Methode DatenblattFields eingefügt werden:

```
${DatenblattFields(<<tablestylesheet>>,"<<Felder>>",FelderObject.allKeys)}
```

2.2.3.5 Grundgerüst für neue Abfragen

Voraussetzung:

Feld über das Schleife laufen soll, muss eine Sicht sein (Feldart 12)

Schritt	Beispiel
1. temporäre Datentabellen erstellen	<@selectintotmp select="ch30_fach,..." source="sos_statistik S" target="tmp_sosstatistik"> where <<Köpfe oder Fälle>> and <<Hörerstatus>> and ...

	and ch30_fach in <@printkeys Fächer.allNeededKeys/> </@selectintotmp> <@informixnolog/>;
2. Schleife über alle gewünschten Knoten	--Schleife, über jede Kostenart im ausgewählten Kostenarten-Baum, Reihenfolge genau wie im Baum <#foreach eineKostenart in Kostenarten.elements> --mit if-Anweisungen können ggfs. einzelne Einträge übersprungen werden (z.B. wegen Aggregierungsauswahl) Insert into tmp_erg (fikr , betrag) SELECT '${eineKostenart.id}'::char(10) \|\| ' ' \|\| '${eineKostenart.name}'::char(100), sum(betrag) FROM tmp_busa where fikrkey in ${eineKostenart.subkeys} --liefert nötige Schlüssel als ('1','12','13'..) für aktuelle Kostenart group by 1 ; </#foreach>
3. ggfs Weiterverarbeitung	udpate tmp_erg set …..
4. Abschluss-Select	select * from tmp_erg;

2.2.3.6 Mehrfachauswahl bei Sicht-Feldern mit Schleifenfunktion

Ab Kern4.5 kann man eine Mehrfachauswahl bei Sichtfeldern aktivieren, die als Schleife für die Ergebnisdarstellung benutzt werden.

Angenommen wir haben folgenden Finanzstellenbaum

- Hochschule
  - A1
    - A11
    - A12
  - B2
    - B21
    - B22

Bisher gab es ein Problem wenn ein User bei der Mehrfachauswahl A11 und B2 auswählte, weil dann die Ebenendarstellung durcheinander kam, die Ergebniszeile für A11 wurde gar nicht angezeigt, weil Ebene 2 < als Ebene 1 der zweiten Selektion B2.

Dies kann man jetzt umgehen, in dem man das Feld Finanzstelle obligatorisch macht, es muss also immer eine Selektion geben und dann für jede Finanzstelle nicht level benutzt, sondern levelFromSelection+1

```
<#foreach eineFistl in Finanzstelle.elements>
<#assign sortnr=sortnr+1/>
insert into tmp_erg (sortnr,ebene,key,name,
  budgetsumme,einnahmen,mittelbindung,ausgaben,verfuegbar_of,verfuegbar)
select ${sortnr},${eineFistl.levelFromSelection+1},'${eineFistl.key}','${eineFist-
l.name[(eineFistl.name?index_of('-')+1)..(eineFistl.name?length-1)]}',
sum(budgetsumme),sum(einnahmen),sum(mittelbindung),sum(ausgaben),sum(verfuegbar_of),s
um(verfuegbar)
from tmp_konto2
where kst_nr in <@printkeys eineFistl.subkeys/>
group by 1,2,3,4;
</#foreach>
```

Eine Summenberechnung bei mehr als einer Zeile geht dann so

```
<#if Finanzstelle.selectionCount&gt;1>
create table...
insert into tmp_sum  (
budgetsumme ,
 einnahmen  ,
 ausgaben  ,
 mittelbindung  ,
 verfuegbar ,
 verfuegbar_of )
 select
 sum(budgetsumme) ,
 sum(einnahmen)  ,
 sum(ausgaben)  ,
 sum(mittelbindung)  ,
 sum(verfuegbar) ,
 sum(verfuegbar_of)
 from tmp_konto2
 where kst_nr in (
 <#foreach eineFistl in Finanzstelle.elements>'${eineFistl.key}' <#if
eineFistl_has_next>,</#if></#foreach>
 );
</#if>
Ein Beispiel ist die Maske SAP Budget nach Finanzstellen 33060.
```

2.2.3.7 Kern 4.5: Rechte für beliebige Sichtarten

Es gibt eine neue Kerntabelle sichtart_rechttabelle:

Beispielhaft Füllung

Tid	Art	Tabelle	Feldname	Additionalkeyssql
1	SAP-Finanzstel-len-Sicht	gxtage_user_finanzstellen	finanzstelle	
2	SAP-Pro-jekt-Sicht	gxtage_user_hhprog	hhprog	select distinct ktr_nr from gxstage_huel where kst_nr in <<SAP-Finanzstellen-Sicht>>

Für Eine Sichtart wird eine Rechtetabelle angegeben, dabei ist Konvention, dass die Userspalte userinfo_id heißt, der Feldname des Rechtfelds kann angegeben werden (finanzstelle bzw. hhprog im Beispiel).

```
CREATE TABLE gxstage_user_finanzstellen
(
   tid           serial         primary key,
   userinfo_id   integer        NOT NULL,
   finanzstelle  varchar(24)
);
```

In der letzten Spalte additionalkeyssql kann eine Besonderheit angegeben werden, zusätzliche Keys die die User sehen dürfen. Das kann im einfachen Fall z.B. eine zusätzliche tabelle sein für finanzstellen

select finanzstelle from finanzstelle_zusatzrechte where userinfo_id=<<UserID>>

Das obige Beispiel für HHProgramme ist für eine Querabhänigkeit

select distinct ktr_nr from gxstage_huel where kst_nr in <<SAP-Finanzstellen-Sicht>>
es werden aus der HUEL alle hhprogramme rausgesucht für Finanzstellen, die der User in einer SAP-Finanzstellen-Sicht sehen darf – unabhängig von der Gültigkeit der Finanzstellen!

(Interna im Log: Dafür wird die SAP-FInanzstellen-Sichtarten einmal mit dem Spezialdatum 1.1.1970 aufgebaut → das bedeutet ohne Gültigkeit)

Es gibt zwei zusätzliche Freemarker-Variablen die man Abfragen kann
UserHasAllSichtart-Rights
UserHasNoSichtart-Rights
z.B.

```
<#if .vars["UserHasNoSAP-Finanzstellen-SichtRights"]>
  insert into tmp_erg (name) values ('Keine Rechte für Finanzstellen gefunden');
 <#else>
```

2.2.3.8

Man kann sich auch die gewählte Sichtart/name/name_intern ausgeben lassen.

z.B. ${Finanzposition.sichtart}

${Finanzposition.sichtname_intern}

${Finanzposition.sichtname}

Beispiel

<#assign linkbuch="SuperXmlTabelle?tid=33000&cachingcontrol=clearmask&Jahr=<<Jahr>>"/>

<#if Finanzposition.sichtname_intern=='fipos_inkl_stats'>

<#assign linkbuch=linkbuch+"&Finanzpositionswahl=alle"/>

<#else>

<#assign linkbuch=linkbuch+"&Finanzpositionswahl=zahlungswirksam"/>

</#if>

2.2.4 Datenbankunabhängigkeit

Grundlage für die Datenbankunabhängigkeit (Informix/Postgres) von SuperX-Abfragen sind die Free-Marker Makros SQL Lingua Franca (s.o.).

Zusätzlich sollte man auf Folgendes achten.

entwickeln mit Postgres

Da Postgres bei einigen Dingen strenger ist, sollte man mit Postgres entwickeln, wo möglich.

einfache Hochkommata haben Vorrang

SELECT '${eineKostenart.id}'::char(10), sum(betrag)

FROM tmp_busa …

Informix würde auch "${eineKostenart.id}" verstehen, Postgres würde dies jedoch für einen festen Spaltennamen halten und meckern, dass es keine solche Spalte gibt.

Casting benutzen

Postgres muss öfter mittels Casting über den Datentyp informiert werden.

insert into …

select 'Summe'::char(100), ... from … group by 1,2,3;

Informix braucht das nicht unbedingt, Postgres meldet bei fehlendem Casting gerne:

ERROR: Unable to identify an ordering operator '<' for type '"unknown"'

Use an explicit ordering operator or modify the query

Andersherum ist es wichtig, bei unions auch null zu casten.

select x,y,z from test

```
union
select x, null::char(10),z from test2
```

Postgres kommt ohne Casting klar, weil es aus dem ersten Select den Datentyp erkennt. Informix meldet demgegenüber nur Syntax Error.

2.2.5 Zugriff auf Konstantentabelle und Hochschulinfo

Die Konstantentabelle steht auch komplett als FreeMarker-Variablen zur Verfügung. Damit es keine etwaigen Überschneidungsprobleme mit Feldernamen gibt, wird ein K_ vorangestellt.

Die COB_Version könnte also z.B. so abgefragt werden

```
<#if K_COB_Version&gt;7 >
…
<#else>
…
</#if>
```

Die ersten Einträge der Tabelle hochschulinfo sind ebenfalls verfügbar als `K_Name`,`K_Adresse`,`K_hs_nr` und `K_Kapitel`.

2.2.6 Sx_repository

In der Tabelle sx_repository können Einträge hinterlegt werden, die in allen Abfragen zur Verfügung stehen, z.B. kann dort ein SQL hinterlegt werden, der bestimmt, wie sich ein Professor in SVA definiert (z.B. bvl=30000 and dienstbez like „Prof"). Die Tabelle ist folgendermaßen aufgebaut:

tid	serial		1
id	bpchar(200)	Sollte mit der Modulbezeichnung anfangen und keine Leerzeichen enthalten	SVA_Professor
content	text(-1)	der Inhalt, der in den Abfragen eingesetzt werden soll	bvl=30000 and dienstbez=’prof’
caption	bpchar(200)	Bezeichnung, die ggfs. auch auf dem Bildschirm mit ausgegeben werden soll	Professoren
comment	text(-1)		
version	int2		
art	bpchar(200)		
sachgebiete_id	int4		
sort1	int4		
sort2	int4		
sort3	int4		

gueltig_seit			
gueltig_bis			

Die Repository wird im Applet bei der Anmeldung mit übermittelt, im XML-Frontend wird es gecacht.
(Wenn Änderungen gemacht werden, muss einmal im SuperXManager der Server-Cache aktualisiert werden).

In den Abfragen kann man die Repository Einträge z.B. folgendermaßen verwenden
insert into tmp_erg (bezeichnung, wert)

```
select "Professoren", sum(value) from xxx where ${SVA_Professor}
```

anstatt den Beschreibungstext fest anzugeben, kann man auch die caption des Eintrags benutzen.

```
select "${SVA_Professor.caption}", sum(value) from xxx where ${SVA_Professor}
```

Wenn es nur einen Eintrag für eine ID gibt, reicht die beschriebene Vorgehensweise. Falls es aber für eine ID mehrere Einträge mit unterschiedlichen Gültigkeitszeiträumen gibt, schreibt man z.B.

```
select "Sachmittel", sum(value) from xxx where ${FIN_Sachmittel("1.1.<<Rechnungsjahr>>")}
```

dann wird der erste gefundene FIN_Sachmittel Eintrag ausgegeben, der zum angegebenen Datum gültig ist. Dabei kann bei Bedarf auch wieder auf einzelne Attribute zugegriffen werden, also z.B.

```
select "${FIN_Sachmittel("1.1.<<Rechnungsjahr>>")}.caption", sum(value) from xxx where ${FIN_Sachmittel("1.1.<<Rechnungsjahr>>")}
```

2.2.7 Abfragen mit variabler Spaltenzahl

Es ist nun mit FreeMarker auch möglich, Abfragen mit einer variablen Anzahl von Ergebnisspalten zu erzeugen. Dazu muss mit Freemarker an zwei Stellen Variabilität eingebracht werden:
- im SQL der Maske und
- in der XIL-Proplist, die die Definition der Ergebnisspalten enthält.

Einfaches Beispiel:
Im FIN-Modul gibt es zwei Quellsysteme (MBS und KAHIKA). Je nach Quellsystem soll in einer Abfrage unterschiedliche Spalten erscheinen. Das Quellsystem kann über die Konstante FIN_QUELLSYSTEM ermittelt werden (Zugriff über FreeMarker als K_FIN_QUELLSYSTEM, 1=MBS, 2=KAHIKA). Am Ende des Masken-SQLs kann beispielweise stehen:

```
<#if K_FIN_Quellsystem=2>
select eintrag, hhans,reste, bewegungen , fest, verfuegbar, verfuegbar-fest from
tmp_erg order by dr,sortnr,tit;
<#else>
```

```
select eintrag, hhans,reste, akts_ein, sperr,
angeordneta,offsoll_a,angeordnete,offsoll_e , fest, verfuegbar, verfuegbar-fest
from tmp_erg order by dr,sortnr,tit;
</#if>
.
```

Die definierten Spaltenüberschriften in der XIL-Proplist müssen ebenfalls variabel angelegt werden:

```
--freemarker template
XIL List
   drop_and_delete movable_columns sizable_columns horizontal_scrolling
   white_space_color=COLOR_WHITE fixed_columns=1
   min_heading_height=35
Column CID=0 heading_text="Titel / DR" center_heading
   row_selectable col_selectable heading_platform readonly
   width=20 text_size=8 explanation="@@@fin_titel_dr@@@"
Column CID=0 heading_text="Zuweisungen" center_heading
   row_selectable col_selectable heading_platform readonly
   width=15 text_size=8 explanation="@@@fin_zuweisungen@@@"
Column CID=0 heading_text="Reste" center_heading
   row_selectable col_selectable heading_platform readonly
   width=15 text_size=8 explanation="@@@fin_reste@@@"
<#if K_FIN_Quellsystem=2>
   Column CID=0 heading_text="Bewegungen" center_heading
   row_selectable col_selectable heading_platform readonly
   width=15 text_size=80 explanation="@@@fin_bewegungen@@@"
<#else>
Column CID=0 heading_text="Akt.Soll" center_heading
   row_selectable col_selectable heading_platform readonly
   width=15 text_size=8 explanation="@@@fin_akt_soll@@@"
   ...
</#if>
   Column CID=1 heading_text="Festgelegt" center_heading
   row_selectable col_selectable heading_platform readonly
   width=15 text_size=80 explanation="@@@fin_festgelegt@@@"
```

Grundprinzip ist also, dass mit FreeMarker die Variablität in den SQL und die XIL-Proplist gebracht werden muss.

Ein komplexes Beispiel:

In einer Abfrage soll ein variable Anzahl von Lehreinheiten in den Spalten erscheinen, diese können nicht direkt aus einem Sicht-Feld Lehreinheiten ermittelt werden (in FreeMarker wäre dann z.B. Zugriff über `<#foreach lehrein in Lehreinheiten>` möglich). Stattdessen wird die Gesamtliste aller gewünschten Lehreinheiten über die Felder *Semester* und *Anbietende Lehreinheit* definiert. Damit Freemarker vor der eigentlichen Transformation noch eine passende aus der Datenbank gefüllt Variable erhält, wird am Anfang der Abfrage eine sqlvar namens *lehr_abg* angelegt (vergl. Abschnitt zu Variabeln).

```
--Freemarker Template
```

```
<#include "SQL_lingua_franca"/>
<#include "SuperX_general"/>
<sqlvars>
<sqlvar name="lehr_abg">
SELECT distinct L.key_apnr,
       L.drucktext, L.name as strukturstr
 from gang_k_lehr_hs L, gang_k_semester S
  where
1=1
/* and L.key_apnr in(<<Anbietende Lehreinh.>>) */
/* and S.tid = <<Semester>> */
 and S.sem_beginn between L.gueltig_seit and L.gueltig_bis
order by 2;
</sqlvar>
</sqlvars>
```

Anschließend wird die Ergebnistabelle variabel erzeugt (has_content prüft, ob überhaupt Lehreinheiten gefunden wurden).

```
create temp table tmp_gang_cnw2 (
ord integer,
tid integer,
semester_von integer,
sem_beginn date,
lehr_abg character(10),
lehr_empf character(10),
lehr_empf_str char(255),
lehr_empf_sort char(255),
stg_empf_str nchar(255)
<#assign i=0 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
, lehr_${i} decimal(5,2)
</#foreach>
</#if>
)
```

An späterer Stelle können variabel updates auf die Ergebnistabelle gemacht werden, damit kann mit .key auf den Schlüssel der Lehreinheit zugegriffen werden.

```
<#assign i=0 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
update tmp_gang_cnw2 set lehr_${i} = (select sum( C.ca_wert)
from tmp_gang_cnw C
where C.lehr_abg=tmp_gang_cnw2.lehr_abg
 and C.lehr_empf=tmp_gang_cnw2.lehr_empf
and C.tid=tmp_gang_cnw2.tid
and tmp_gang_cnw2.lehr_abg= '${row_gang.key}' )
where ord=2;
</#foreach>
```

```
</#if>
```

Zum Schluss könnte man einfach `select * from tmp_xx`; machen, oder sicherheitshalber besser:

```
select ord as ebene, stg_empf_str
<#assign i=0 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
, lehr_${i}
</#foreach>
</#if>
 from tmp_gang_cnw4
```

Schließlich muss nur noch die XIL-Proplist variabel erzeugt werden, dabei kann mit .name auf den Namen der Lehreinheit zugegriffen werden.

```
XIL List
   sizable_columns horizontal_scrolling
   white_space_color=COLOR_WHITE fixed_columns=1
   min_heading_height=35
Column CID=0 heading_text="Ebene" explanation="" center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=30
Column CID=1 heading_text="NachfragendenLehreinheit\\nStudiengang" explanation="" center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=30
<#assign i=1 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
Column CID=${i} heading_text="${row_gang. name }" explanation="" center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=20
</#foreach>
</#if>
```

Die volle Flexibilität ist im XML-Frontend gegeben. Da sich das Applet Spaltendefinitionen merkt, kann es Unstimmigkeiten geben, wenn die Spaltenzahl von Auswahlfeldern der Maske abhängt.

2.2.8 Maskennummer – für ähnliche Masken gleiches select_stmt

Vererbung oder objektorientierte Entwicklung von Masken gibt es in SuperX zwar (zum Glück?) noch nicht.

Es gibt aber schon mal Fälle, wo man zwei oder drei Masken hat, die den gleichen Grundaufbau haben und sich nur in Details unterscheiden.

Effizient ist es dafür, wenn man den gleichen select_stmt für alle Masken nehmen kann und nur die Details per Freemarker anpasst.

Z.B. eine Datei 50000-50100.sql die den Sql für die beiden Masken 50000 und 50100 enthält.

Anfang

```
--freemarker template
create temp table tmp_erg (...);
insert into ...
...
--spezifisch je nach Maske
<#if Maskennummer=50000>
update tmp_erg set x=y*-1 where ...;
</#if>
<#if Maskennummer=51000>
delete from tmp_erg where z=0;
update tmp_erg set x=z*-1 where ....;
</#if>
--abschluss-select

select * from tmp_erg;
```

Die Datei kann man als select_stmt sowohl für Maske 50000 als auch für 50100 verwenden.

Wenn eine Maske in SuperX aufgerufen wird, wird je nach Maskennummer ein Abschnitt ausgeführt.

Damit erspart man sich das Copy&Paste in beide Masken und wenn man allgemeine Änderungen/Erweiterungen an beiden Masken macht, kann man nichts vergessen .

Praktisch Datei mit sx_masken_sql_update.x einspiel (s.unten)

Die Variable Maskennummer steht auch in der XIL-Proplist zur Verfügung, beim Einsatz von CUSTOM_XXX Repository-Variabeln für dynamische Masken wurde das sogar schon mal in einer CUSTOM_XXX-Variablen eingesetzt. Gleiche Vorlage für zwei Masken, je nach Maskennummer geringe Unterschiede.

Beispiel:

CUSTOM_50000_50100

<#if Maskennummer=50000>

<#assign customize={"resulttable":[
{"field":"name","caption":"Gliederung","width":14},
{"field":"akt_soll","caption":"Ansatz","width":15,"explanation":"Haushaltsansatz inkl.Einnahmen und Reste (Haushalterisch: Aktuelles Soll)"},

```
{"field":"einnahmen","caption":"Einnahmen","width":10},
{"field":"aus","caption":"Ausgaben","width":10},{"field":"fest","caption":"Festgelegt","width":10},
{"field":"verfuegbar","caption":"verfügbar","width":12,"explanation":"@@@fin_verfuegbar@@@"}
]}/>
<#else>
{"field":"name","caption":"Gliederung","width":14},
{"field":"verfuegbar","caption":"verfügbar","width":12,"explanation":"@@@fin_verfuegbar@@@"}
]}/>
</#if>
```

Bei Maske 50000 erscheinen alle Spalten, bei 50100 nur name und Verfügbar.

Man ergänzt dann vor dem Abschluss-select

```
<#if CUSTOM_50000_50100?exists>
<#assign inlineTemplate=CUSTOM_50000_50100?interpret>
<@inlineTemplate/>
</#if>

ginge natürlich auch mit zwei custom_variablen im repository
<#if Maskennummer=50000&& CUSTOM_50000?exists >
<#assign inlineTemplate=CUSTOM_50000?interpret>
<@inlineTemplate/>
</#if>
<#if Maskennummer=50100&& CUSTOM_50100?exists >
<#assign inlineTemplate=CUSTOM_50100?interpret>
<@inlineTemplate/>
</#if>
```

2.2.9 Tooleinsatz

2.2.9.1 Jedit

Wenn intensiv mit FreeMarker gearbeitet wird, ist es angenehm mit Jedit zu arbeiten.

Dazu speichert man das `select_stmt` der betreffenden Abfrage in einer Datei, z.B. 11300.sql.

Wenn man diese mit Jedit öffnet, hat man schon mal SQL-Syntax Highlighting, man kann aber bei Bedarf auch Freemarker-Syntax Highlighting einschalten (Menü Utilities / Buffer Options / Edit Mode -> Freemarker).

Sehr angenehm ist auch die Folding Funktion (s.u.).

Um ein geändertes Skript komfortabel einzuspielen, kann man das Skript `sx_masken_sql_update.x` benutzen.

2.2.9.1.1 FreeMarker-Syntax Highlighting

Menü Utilities / Buffer Options / Edit Mode -> Freemarker

2.2.9.1.2 Folding

Mit dem Jedit-PluginManager das Plugin "ConfigurableFoldHandler" installieren (klappte mit Jedit 4.2, bei 4.3 teilweise Probleme)

Einzelne Abschnitt mit --start und --end Kommentaren versehen

z.B.

```
--start Schleife
<#foreach ..
...
</#foreach>
--end Schleife
```

Menü Plugins/ConfigurableFoldHandler.

Use default ausschalten,

statt dessen

--start und --end als Fold String eingeben.

Menü Utilities/Buffer Options/Folding Mode: custom

und dann

Menü Folding / Collapse All Folds

2.2.9.2 emacs for freemarker

Emacs ermöglicht für lange Freemarker Scripte Folding und auch Indentation (Einrücken)

Man geht so vor:

1. Emacs per yast installieren
2. web-mode installieren (http://web-mode.org/) Download der Datei https://raw.github.com/fxbois/web-mode/master/web-mode.el speichern in ~/.emacs.de
3. Bearbeiten der Datei ~/.emacs bzw. diese hier benutzen

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; File name: ` ~/.emacs '
;;; --------------------
;;;
;;; If you need your own personal ~/.emacs
```

```
;;; please make a copy of this file
;;; an placein your changes and/or extension.
;;;
;;; Copyright (c) 1997-2002 SuSE Gmbh Nuernberg, Germany.
;;;
;;; Author: Werner Fink, <feedback@suse.de> 1997,98,99,2002
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;; Test of Emacs derivates
;;; -----------------------
(if (string-match "XEmacs\\|Lucid" emacs-version)
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;;; XEmacs
  ;;; ------
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (progn
     (if (file-readable-p "~/.xemacs/init.el")
        (load "~/.xemacs/init.el" nil t))
  )
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;;; GNU-Emacs
  ;;; ---------
  ;;; load ~/.gnu-emacs or, if not exists /etc/skel/.gnu-emacs
  ;;; For a description and the settings see /etc/skel/.gnu-emacs
  ;;;   ... for your private ~/.gnu-emacs your are on your one.
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (if (file-readable-p "~/.gnu-emacs")
      (load "~/.gnu-emacs" nil t)
    (if (file-readable-p "/etc/skel/.gnu-emacs")
     (load "/etc/skel/.gnu-emacs" nil t)))

  ;; Custom Settings
  ;; ===============
  ;; To avoid any trouble with the customization system of GNU emacs
  ;; we set the default file ~/.gnu-emacs-custom
  (setq custom-file "~/.gnu-emacs-custom")
  (load "~/.gnu-emacs-custom" t t)
;;;
(add-to-list 'load-path "~/.emacs.d/")
(load "~/.emacs.d/web-mode.el")
(require 'web-mode)
(add-to-list 'auto-mode-alist '("\\.sql\\'" . web-mode))
;;; Informix mag keine Tabs beim Einruecken fuert zu Syntax Error
```

```
(setq-default indent-tabs-mode nil)
;;;fuer sql files freemarker modus nutzen
(setq web-mode-engines-alist
      '(("freemarker"    . "\\.sql\\'")
      )
)

;;;beim Umschalten zu SQL keywords highlight
(add-hook 'sql-mode-hook 'sql-highlight-postgres-keywords)

)
;;;
```

Danach eine langes Freemarker Script z.B. trans_kenn_konto_aggr.sql öffnen mit `emacs trans_kenn_konto_aggr.sql`.

Standardmäßig ist man zunächst im Web-Modus, d.h. man hat Freemarker-Highlighting.

Im Menü Web-mode kann man wählen „Indent Buffer“ zum Einrücken. Kann hilfreich zur Übersicht sein.

Wenn man auf einem <#if oder <#macro oder ähnlichem Befehl mit dem Cursor steht, kann man im Web-Mode-Menü wählen „Fold/Unfold“ um den Block ein-/auszublenden.

Leider gehen bekannt Shortcuts für Copy/Paste nicht wie üblich, man kann das Menü nehmen.

Um einen längeren Abschnitt zu markieren drückt man CTRL+space und dann einfach den Cursor. Zum Suchen an den Anfang der Datei gehen, dann im Menü search Search String Forward, ggfs. im gleichen Menü Repeat Forward.

4. Um auch bei längeren SQL-Abschnitten den Überblick zu behalten, kann man auch „Pseudo-if“-statements einfügen, z.B.

```
<#if "Tabellenerzeugung“!="">
create temp table tmp_konto_aggr (
hs_nr INTEGER ,
jahr SMALLINT ,
quartal integer,
monat SMALLINT ,
fikrkey CHAR(12) ,
klr_geldgeber CHAR(10),
titelgruppe_gege varchar(24),
 titel char(10),
 dr char(5),
 dr2 char(5),
```

```
 titelart CHAR(1) ,
 kapitel char(5),
ch110_institut CHAR(10) ,
projnr char(10),
betragsart CHAR(1) ,
bund_fachgebiet char(10),
bvkr_art char(12),
extkotr char(30),
extkost char(30),
datum date ,
kfmkam smallint,
 hhans decimal (14,2),
 einnahmen decimal(14,2),
 ausgaben  decimal (14,2),
 --nur Sachsen benötigt
 akts_tit decimal(14,2),
 hhans_tit decimal (14,2),
 einnahmen_tit decimal(14,2),
 ausgaben_tit  decimal (14,2),
 reste_tit decimal (14,2)
);
<7#if>
```

Das IF ist natürlich nur pseudo weil immer true, man kann im String dem Block einen Namen geben und dadurch, dass es ein Freemarkerbefehl ist, Fold/Unfold benutzen. Bei Fold sieht man dann nur

```
<#if "Tabellenerzeugung"!=""></#if>
```

Die Unterstreichung weist darauf hin, dass ein Fold-Block dahinter liegt.

So kann man bei langen Scripten gut den Überblick behalten.

Man kann zu SQL Highlighting wechseln, indem man ESC loslässt und dann x drückt und dann sql-mode RETURN eingibt. Zurück zum web-modes mit Freemarker Funktionen kommt man mit `ESC x web-mode`.

Happy Coding :-)

2.2.9.3sx_masken_sql_update.x

Wenn man das select_stmt einer Abfrage in einer eigenen Datei bearbeitet (z.B. mit Jedit), kann man es mit dem Skript sx_masken_sql_update.x komfortabel in die Datenbank einspielen.

Die Syntax ist einfach

sx_masken_sql_update.x Maskennummer Pfad/zur/Datei.sql

sx_masken_sql_update.x 11300 /home/superx/entwicklung/11300.sql

2.3 Abfragenentwurf mit SuperX-Sichten

In SuperX können bei Auswahldialogen verschiedene Sichten angeboten werden.

Alternative Hierarchien bzw. Auswertungshierarchien aus COB werden automatisiert übernommen. Jede Hierarchie bekommt einen Eintrag in der Sichtentabelle.

> Achtung: Die Inhalte und Rechte der Sichten werden gecached. Nach einer Änderung der Inhalte einer Sicht im laufenden Betrieb musst man im SuperXManager den Cache leeren und sich neu anmelden.

Die Sichten werden in der folgenden Tabelle definiert:

Spalte	Datentyp	Beschreibung	Beispiel	Beispiel 2
tid	integer (serial)	eindeutiger Identifier, der beim Einlesen und auch zur Rechtevergabe benutzt wird Die tid 0 bis 9 sind für interne Sichten reserviert.	0	100
parent	integer	Uum zukünftig evtl. Hierarchien von Sichten abzubilden		
system info_id	integer	querverweis zur systeminfo 0 für allgemein	0	10
art	char	Art der Sicht. Bezeichnung sollte auf –Sicht enden, dann wird auch xxx-Sicht und xxx-Stand im Masken-SQL ersetzt	Organigramm-Sicht	Kosten-/Erlösarten-Sicht
type	integer	10 für reguläre Sichten 20 für alternative Hierarchien		
name	char	Name, der auf dem Bildschirm bei der Auswahl erscheint	Organisatorische Sicht	Standardsicht-Kostenarten
name_ intern	char	interner eindeutiger Name, wenn der interne Name mit memtext beginnt, sollte die Sicht nicht eigenständig geändert werden		
beschreibung	char	ausführliche Beschreibung der Sicht		
stand button	integer	soll man bei einer Sicht, den Stand ändern können muss für eine Sichtart (Organigramm-Sicht) einheitlich sein	1	1
label	integer	kann zur Charakterisierung einer Sicht benutzt werden in Duisburg-Essen intern für altes Feld lehre benutzt 0 = alle Org.Einheiten 1 = Nur Lehre anzeigen; Institutionen im Bereich Lehre gefiltert nach user_institution 2 = Nur Lehre anzeigen; Institutionen im Bereich Lehre NICHT gefiltert wird von Prozeduren (sp_user_org,sp_user_orga_child) noch unterstützt, in-	0	0

		tern aber auf neue Variante Label und User-Rechte umgesetzt		
u ser_ rechte	integer	sollen beim Aufbau der Hierarchie (vergl. Spalte quelle) Userrechte berücksichtigt werden, wird bisher nur von Duisburg-Essen benutzt	1	1
rechte-quelle	char(255)	für zukünftig erweiterte Rechteverwaltung		
sesamkey	char(100)	für zukünftig erweiterte Rechteverwaltung		
quelle	text	SQL zum Aufbau der Hierarchie vergl. ausführliche Beschreibung unten	sp_user_orga(<<UserID>>,<<Stand>>,<<Sicht>>); select name,key_apnr,parent,lehre,erlaubt from tmp_organigramm;drop table tmp_organigramm;	sp_cob_fikr_hier(<<UserID>>,<<Stand>>,<<Sicht>>);select name,key,parent from tmp_hier;drop table tmp_hier;
attr ibut1	char	bei Bedarf noch Attribute der Sicht hinterlegt werden, auf die man bei Bedarf Einschränkungen fahren kann		
attribut2	char			
attribut3	integer			
attribut4	integer			
alt_quelle	char	Tabelle mit Infos zu altn. Hierarchie		
alt_hier_id		id der alt.Hier in angegebenen Quelltabelle null bei reguläre Hierarchie		
treecfgtable	char	Tabelle mit Infos zu TreeView aus Cob		
treecfgid		id des benutzen trees aus Cob-Tabelle trees null bei regulärer Hierachie		
implie dTags	text	future use spezifische Tags die in select_stmts ersetzt werden. Syntax <<Tagname>>=valuel..l.. Diese Spalte wird von Memtext gepflegt und darf nur von Memtext angepasst werden.		
userTags	text	future use: Falls Sie selbst spezielle Tags für eine Sicht hinterlegen wollen, können Sie das analog zur implied-Tags-Spalte hier tun.		
xmlmaxentries	integer	wenn ein Wert angegeben ist, werden maximal die Anzahl von Einträgen in Comboboxen im Xml-Frontend angezeigt. Es werden soviele Einträge auf unteren Ebenen ausgeblendet wie nötig.		
sortnr	integer	kann für Sortierungen benutzt werden		
aktiv	integer	über das Feld aktiv können bei Bedarf Aktivierung/Deaktivierung vorgenommen werden, wenn man z.B. im Felderinfo relation schreibt <<SQL>>select tid from sichten where art='Organigramm-Sicht' and aktiv=1		
gueltig_se	date	Zukünftig für Gültigkeitszeiträume von Sichten		

it				
gueltig_bis	date			

Quelle

Es handelt sich um den SQL, der ausgeführt werden soll. <<SQL>> am Anfang ist optional.

Wenn der SQL mit sp_ anfängt, wird davor je nach Datenbankserver execute procedure oder select gesetzt. Für alle Sichten wird erwartet, dass mindestens drei Felder *name*, *key* und *parent* geliefert werden.

(der konkrete Name der Spalten ist irrelevant).

Anschließend können optional noch Strukturinformationen[7] folgen (entweder Integer oder String).

Beispiel:

```
sp_fin_inst_hier(<<UserID>>,<<Stand>>,<<Sicht>>);select name,key,parent,strukturint from tmp_hier
order by name; drop table tmp_hier;
```

strukturint (gefüllt aus fin_inst.orgstruktur) gibt mit Zahlen an, ob ein Eintrag eine Lehreinheit oder ein Fachbereich ist.

Auf diese Information kann später in Abfragen mit FreeMarker zugegriffen werden.

Beispiel

```
select
<#foreach eineInstitution in Institutionen>
   <#if Aggregierung="stark" and eineInstitution.strukturInt=30>
   ...
   <#else>
   ..
   </#if>
</#foreach>
```

2.3.1 Einträge verstecken oder nicht-selektierbar machen

Es kann gewünscht sein, dass Einträge versteckt hinterlegt werden, wenn z.B. immer bei Auswahl eines Instituts 0405 auch intern dessen alte Nummer 0205 berücksichtigt werden soll.

Der User soll in SuperX nur 0405 sehen und auswählen können. Bei Berechnung durch die Datenbank soll aber 0405 und 0205 berücksichtigt werden.

[7] Für die klassischen Organigramm-Sichten muss an Position 4 und 5 zunächst die Information lehre (0/1) und erlaubt(0/1) folgen.

Um einen solchen Effekt zu erreichen, muss einzelnen Einträgen für die Sicht der Wert nodeattrib=1 gegeben werden. Das geschieht, indem man den Quell-SQL der Sicht (s.o.) erweitert, um ein *explizit benanntes* Feld nodeattrib, z.B.

```
sp_fin_inst_hier(<<UserID>>,<<Stand>>,<<Sicht>>);select name,key,parent,strukturint ,nodeattrib
from tmp_hier order by name; drop table tmp_hier;
```

Die Position von `nodeattrib` muss nach `name,key` und `parent` liegen, ist ansonsten aber irrelevant. Entscheidend ist die Spaltenbezeichnung `nodeattrib`. Sichergestellt werden muss auch, dass die Hierarchie stimmig ist, also dass der Eintrag mit dem Schlüssel `0205` als Parent den Wert `0405` hat.

Noch ein Beispiel für eine spezielle Kostenartensicht, bei der Kostenarten mit langen Schlüsseln versteckt werden sollen.

```
select lbez,key,ueberg,0 as nodeattrib from cob_fikr where len(key)<=6 union
select lbez,key,ueberg,1 as nodeattrib from cob_fikr where len(key)>6
```

Weiterhin kann es gewünscht sein, dass man in einer Sicht nur spezielle Einträge auswählen kann.

So können man eine Lehreinheitsauswahl vielleicht so darstellen:

Uni XY
- Fak 1
  - Lehreinheit A
  - Lehreinheit B
- Fak 2
  - Lehreinheit C
  - Lehreinheit D

Die Kategorisierung Fak 1/2 dient der Klarheit der Darstellung, für eine Abfrage soll aber vielleicht **nur** eine konkrete Lehreinheit auswählbar sein. In dem Fall kann wie oben das nodeattrib 2 vergeben werden. Dadurch wird ein Eintrag als nicht-selektierbar markiert.

2.3.2 User-/Gruppenrechte

Außer Administratoren dürfen normale User zunächst einmal keine der nicht-internen Sichten benutzen.

In den Tabellen user_sichten und user_sichtarten kann hinterlegt werden, dass ein User eine einzelne Sicht (z.B. name_intern Kostenstellen-Hauptsicht) oder eine ganze Sichtart (z.B. Kostenarten) benutzen darf.

Für Gruppen gibt es analog die Tabellen group_sichten und group_sichtarten.

Außerdem sachgeb_sichten und sachgeb_sichtarten, wenn Leute die ganzes Sachgebiete sehen dürfen auch Sicht sehen sollen

Die Pflege dieser Tabellen kann bequem über Administrationsformular im XML-Frontend vorgenommen werden.

Achtung:
Nach Änderungen muss der Server-Cache aktualisiert werden (http://superx-rechner/superx/servlet/SuperXManager) und der User muss sich neu anmelden.
Alternativ könnte auch Tomcat neu gestartet werden.

Im Dateisystem unter $SUPERX_DIR/db/bin gibt es zwei Skripte userrechte_sichern.x und userrechte_einspielen.x.
Die entladen alle usertabellen (userinfo,groupinfo,user_sachgeb_bez etc)
und spielen sie wieder ein.
Diese können z.B. dafür benutzt werden um alle Einstellungen von Test auf Echtrechner
zu übernehmen.

2.3.3 Benutzung der Sichten in Masken

In Felderinfo muss für das Feld (Org. Einheit,Fächer,Kostenstelle,Kostenart,...) als Art 12 eingetragen werden.

Im Feld Relation muss ein Select stehen, der die Tids der gewünschten Sichten zurückliefert

z.B.

```
<<SQL>>select tid,type,name from sichten where art='Organigramm-Sicht' and aktiv=1 order by type,name
```

(Die aktiv-Einschränkung ist praktisch für Entwicklungs- und Wartungszwecke, weil man dann leicht einzelne Sichten de/aktivieren kann - Sortierung nach type, damit erst Hauptsichten und dann alternative Hierarchien angezeigt werden, dann nach Name)

Der select sollte **alle Sichtentids** liefern, die ein Adminstrator sehen darf.
Wenn ein User für einzelne Sichten keine Berechtigung hat, filtert der Server diese automatisch raus.

Bei einem Sichtenfeld werden Tags bezogen auf Feldname und Sichtenart ersetzt, z.B. beim Feld Org. Einheit sowohl <<Org. Einheit-Sicht>> und <<Org. Einheit-Stand>> als auch <<Organigramm-Sicht>> und <<Organigramm-Stand>>.

Empfehlenswert ist aber insbesondere bei Sichten, die Benutzung der FreeMarker-Technologie.

Wenn diese nicht benutzt wird, muss bei Abfragen mit Organigramm-Sichten ggfs. das select_stmt in Maskeninfo angepasst werden.

Beispiel:

```
execute procedure sp_user_orga_child
   (<<UserID>>,<<Organigramm-Stand>>,<<Organigramm-Sicht>>,<<Org. Einheit>> , <<erlaubt>>)
```

Freemarker Trick um festzustellen, ob bei einer Sicht etwas auswählbar ist

```
<#if Kostenstelle.elementsMaxEbene&gt;1>
...
</#if>
```

wenn nichts auswählbar ist (Anzeige: Keine Auswahl möglich) liefert

elementsMaxEbene 1.

Ansonsten die tiefste Ebene im Baum (z.B. 5)

2.3.4 Alt. Hierarchien aus CoB

Es werden Hauptsichten (Type 10) für Kostenstellen, Kosten-/Erlösarten und Kostenträger angelegt.

Außerdem für jede Auswertungshierarchie einen Eintrag mit Type 20. Im Feld quelle muss die Tabelle angegeben sein, in der die alt. Hierarchie definiert ist.

Eine Tabelle mit alternativen Hierarchie muss so aufgebaut sein:

hierarchie_id	integer
key	char(10)
parent	char(10)

Zum Aufbau der Hierarchie werden weiterhin alt_hier_id,treeviewtable und treeviewid benutzt.

2.4 Spezielle Details

2.4.1 Checkboxen und Querabhängigkeiten

Checkboxen können mit Feldart 10 definiert werden.

Bei einfachen Masken, kann feldeinfo.relation leer bleiben und in felderinfo.default auch ein fester Wert stehen wie true.

Wenn es auf der Maske auch dynamische Felder gibt, in deren SQL z.B. <<UserID>> steht, ist es wichtig, dass auch für checkboxen in relation und default datenbanktypische Definitionen der auswählbaren Werte stehen, z.B.

<<SQL>> select 'true','Ja' from xdummy union select 'false','Nein' from xdummy

<<SQL>> select 'false','Nein' from xdummy

2.4.2 Felder auf der Maske verstecken

Wenn Felder auf der Maske versteckt werden sollen, gibt es zwei Möglichkeiten:

•Feldart 13 -> das Feld ist versteckt und wird intern nicht aufgebaut, im Masken-XML ist es aber enthalten

•Feldart beliebig, Eintrag in Spalte attribut: hidden

Das Feld wird intern aufgebaut und kann auch im Masken-SQL per FreeMarker benutzt werden, es wird aber keine Auswahlmöglichkeit auf der Maske angezeigt (benutzt bisher für Feld Kostenstelle, das nicht angezeigt werden sollte, im Masken-SQL aber schon für Rechtekontrollen benutzt wird)

•Feldart 999 (ab SuperX3.5rc2): Feld wird gar nicht erst aus Datenbank eingelesen, also ob nicht existent

Bei Benutzung der erweiterten kameralen Rechte SxFinRechte:

Auch wenn auf der Maske nicht alle kameralen Felder benötigt werden (z.B. Titel) müssen diese als versteckte Felder vorhanden sein, damit Querabhängigkeiten in Maskenbuttons z.B. FB SxFinRechte(..,"<<Titel>>",....) aufgelöst werden können!

Bei sehr vielen versteckten Feldern rutscht der Abschicken-Button nach unten, da auch versteckte Felder (noch) für die absolute Positionierung berücksichtigt werden. Trick versteckte Felder in felderinfo auf y=-1 setzen, dann kommen sie nicht in Reihenzählung.

2.4.3 Inhalte benutzerspezifisch ausblenden

Es gibt in SuperX- Sichten bereits die Möglichkeit, Inhalte für spezielle Gruppen bzw. User ein- oder auszublenden. Außerdem kann man einzelne User für einzelne Masken berechtigen oder nicht. Darüber hinaus kann es aber auch die Anforderung geben, einzelne Felder userspezifisch anzuzeigen (oder nicht), oder Inhalte in Ergebnistabellen userspezifisch ein- oder auszublenden. Dies wird im folgenden erläutert.

Hier eine Anleitung am Beispiel des TT-Modul,. wie man in einigen Masken user-spezifische Maskenrechte abbildet. Es soll also gesteuert werden, daß gewisse Gruppen bestimmte Felder auf der Maske nicht sehen können, und in der Ergebnistabelle auch nicht angezeigt bekommen.

Das geht nur mit ein paar Tricks.

2.4.3.1 Felder

Siehe Kernmodul-Handbuch. Felder für Benutzergruppen verstecken

2.4.3.2 Tabellen

Dann legt man ein spezielles XSL-Stylesheet für die Maske an, das in der Schleife über alle Felder mit XSLT abfragt, ob der Wert 1 ist – nur dann wird das Feld angezeigt.

Im select_stmt muss man mit dem gleichen Select wie oben im Defaultwert abfragen, ob die Spalte in der Ergebnistabelle gefüllt werden soll.

2.4.4 Baumdarstellung

Um im XML-Frontend eine Baumstruktur (Treetable) zu erhalten die auf- und zugeklappt werden kann, muss nur in dem letzten select als erstes Feld die Spalte "ebene" angegeben werden. Dieses Feld soll Zahlen enthalten, welche die Ebene angeben (1= erste Ebene, 2= zweite Ebene …). Dabei ist darauf zu achten, dass es keine Sprünge zur übernächsten höheren Ebene gibt, z.B. nicht von Ebene 2 auf Ebene 4 gesprungen wird. Andersherum ist dies kein Problem, also z.B. von Ebene 4 auf Ebene 2. Wenn es nun zuerst eine Spalte mit Ebene 1 gibt und darauf mehrere mit der Ebene 2 sind diese alle unter der Spalte der Ebene 1. Die Reihenfolge der Auflistung entscheidet dabei den übergeordneten Knoten und nicht nur die Ebene.

Ein Beispiel:

Ebene	Kostenart
2	**1-Personal- ISTkosten**
3	11-Bezüge/Vergütungen/Löhne
4	112-Angestelltenvergütungen
4	113-Löhne der Arbeiter
3	12-Beschäftigungsentgelte
4	121-Vergütungen für Lehrkräfte
5	1211-Vergütungen für Lehraufträge
5	1212-Verg.f.Kolloquien, Vorträge ..
4	122-Verg.f.Hilfskräfte und Tutoren
4	125-Stipendien
4	126-Verg.f.sonst.Beschäftigte
3	13-Personalnebenkosten
2	**2-Objektkosten**
2	**3-Kosten Fahrzeuge,Masch.,Geräte**

Zum Nachladen von Ergebniszeilen wird das Stylesheet tabelle_html_rows.xsl benutzt.

Wenn man ein spezielles Stylesheet hat, das auch die nachgeladenen Zeilen besonders darstellt, kann man eine eigene Variante von tabelle_html_rows.xsl erstellen und den Dateinamen in der Maske bei chartx eintragen.

2.4.5 Hinweis auf Masken

in Der tabelle maskeninfo.hinweis kann man einen festen Hinweistext eintragen, z.B. Beta-Phase.
Bei Bedarf kann man auch SQL nutzen
<<SQL>> select 'ohne Lehramtsabschlüsse' from xdummy where <<Semester>> < 19961.
Seit 4.1 (10.1.2012) geht auch Freemarker, ein Beispiel
<<SQL>>

```
--freemarker template
<#if !UserIsAdmin&&!IsUserInGroupWithName("Administratoren")&&!
IsUserInGroupWithName("Personalabteilung")>
select 'ohne Personalkosten' from xdummy
</#if>
```

2.4.6 CSV Upload

Die Daten werden zunächst in eine temp Tabelle (Kopie der Zieltabelle) eingefügt und auf Datentyp, Foreign keys etc getestet, wenn der Import prinzipiell möglich ist, können im select_stmt der Maske zusätzliche Kontrollen erfolgen (Freemarker Variable testupload=1), z.B. ob die hs_nr auch die der gewählten Hochschule entspricht, entwaige Fehler können in die temp Tabelle tmp_error eingefügt werden. Falls Fehler eingefügt wurden, zeigt das System diese in der HTML-Vorschau an. Ansonsten wird eine Import-Vorschau im HTML angezeigt und ein „Jetzt importieren" Button. Wenn der Button angeklickt wurde, wird der SQL im select_stmt der Maske (testupload=0) ausgeführt, die Spaltennamen werden in der Freemarker Variablen $colnames hinterlegt.

Ein Feld mit Feldart 19 Datei wird benötigt. Per multipart/form-data wird übertragen.
Beispiel 18020
Diese Felder werden benötigt:
Tabelle = Zieltabelle
Trennzeichen
Kodierung
Feldnamen in 1.Zeile 0/1 = nein/ja
zip-komprimiert 0/1 nein/ja
Modus 1 hinzufügen 0 alles löschen und hinzufügen

Beispiel für select_stmt

```
--freemarker template
<#if testupload="1">
```

```
 insert into tmp_errors (bezeichnung)
 select distinct 'falsche Hochschulnummer '||hs_nr from tmp_${Tabelle} where hs_nr!
=<<Hochschule>>;

 <#else>
 <#if Modus="0"> --alles loeschen und einfuegen
 delete from ${Tabelle} where hs_nr=<<Hochschule>>;
 <#else>
 -- hinzufuegen - do nothing
 </#if>
 insert into ${Tabelle}
 <#if colnames!=""> ( ${colnames} ) </#if>
 select
 <#if colnames!=""> ${colnames}  <#else> *  </#if>
 from tmp_${Tabelle};

 insert into data_upload(
  tablename,
 --       filename,
        datatype,
        datadelimiter,
         /*withheader, --<<Feldnamen 1. Zeile>>*/
        zipped,
        dataencoding,
        ch110_institut,
        submission_userid,
        submission_email,
        submission_date,
        submission_mode
        )
        select
        <<Tabelle>>,
        <<Dateityp>>,
        <<Trennzeichen>>,
        /* <<Feldnamen 1. Zeile>>,*/
        <<zip-komprimiert>>,
        <<Kodierung>>,
        <<Hochschule>>,
        <<UserID>>,
        <<Email fuer Protokoll>>,
        now(),
        <<Modus>>
        from xdummy;
```

```
</#if>
```

2.4.7 Direkt aus einer Maske Jasper Excel/PDF erzeugen

2.4.7.1Direkter Aufruf eines JasperReport /Stylesheet

Wenn Sie ein Stylesheet (egal ob JasperReport oder XSLT) der Maske als alleiniges Tabellenstylesheet zuweisen, wird dieses direkt im Ergebnis aufgerufen. Wenn Sie dann noch in der Maske ein Feld "Ausgabeformat" integrieren, können die Anwender zwischen HTML, Excel, PDF etc. wählen (s.u.).

Standardmäßig werden im Ergebnis nur die ersten 30 Zeilen angezeigt. Damit Sie mehr Zeilen anzeigen lassen können, können Sie ein verstecktes Feld "`maxoffset`" in die Maske kopieren, mit der Feldart 13 (verstecktes Feld) und dem Defaultwert z.B. 10000, um die maximale Zeilenanzahl 10.000 zu erlauben. Dieser Wert ist eigentlich viel zu hoch, servletseitig wird im Parameter maxRows ein allgemeines Limit gesetzt, um die Stabilität des Servers zu gewährleisten.

2.4.7.2Auswahl des JasperReport in der Maske

Sie können auch eine Maske mit den Feldern "Bericht" und "Ausgabeformat" versehen, in dem die Anwender das entsprechende Layout (d.h. den JasperReport) auswählen kann. Ein funktionierendes Beispiel wäre z.B. die Maske "Studierende Datenblatt". Um so etwas zu erzeugen gehen Sie wie folgt vor:

Es muss auf der Maske ein Feld der Feldart 1 geben mit dem Namen

```
tablestylesheet
```

und der relation

```
 <<SQL>> select distinct filename,X.caption from sx_stylesheets
```

X,sx_mask_style S where X.tid=S.stylesheet_id and S.maskeninfo_id=<<Maskennummer>>

und zum Beispiel für Defaultwert:

```
<<SQL>> select distinct filename,X.caption
from sx_stylesheets X,sx_mask_style S
where X.tid=S.stylesheet_id
and S.maskeninfo_id=<<Maskennummer>>
and S.ord=2
order by 2
```

Für das Feld `tablestylesheet` wird automatisch eine Beschriftung hinterlegt, damit auf der Maske als Anzeige "Bericht" erscheint.

Außerdem müssen Sie ein Feld "Ausgabeformat" in der Maske ergänzen, ebenfalls Feldart=1:

relation

```
<<SQL>> select element_value,description from menu_element where
element='Ausgabeformat' and nature::smallint<100 order by nature::smallint
```

Beispiel für Excel als defaultwert:

```
<<SQL>> select element_value,description from menu_element where
element='Ausgabeformat' and description='Excel'
```

2.4.8 Navigationsspalten im XML-Frontend

Wenn die Ergebnistabelle an das XML-Frontend übergeben wird, dann können spezielle Spalten für die Navigation eingesetzt werden. Die Spaltennamen werden im letzten `select` des `select_stmt` einer Makse übergeben.

nexttable	Link auf eine andere SuperX-Tabelle; der Inhalt des Feldes wird dann um den Pfad zum Servlet, (optional auch den String der Sessionid) und den Passus "SuperXmlTabelle?tid=" ergänzt, d.h. dem Servlet wird als erster Parameter die `maskeninfo-tid` übergeben. So wird z.B. aus dem Inhalt:
	`20010&id=2044`
	der Link
	`http://<URL der Webapplikation>>/servlet/SuperXmlTabelle?tid=20010&id=2044`
	Die Ergebnisseite wird dann um einen Button ergänzt.

nextwin-dowtable	Das gleiche wie "nexttable", nur es wird ein neues Fenster geöffnet.
nextpage	Link auf eine andere SuperX-Tabelle wie **nexttable**, es wird nur ein anderes Icon und ein anderer Target genutzt.

nextmask	Link auf eine andere SuperX-Maske; der Inhalt des Feldes wird dann um den Pfad zum Servlet, (optional auch den String der Sessionid) und den Passus "SuperXmlMaske?tid=" ergänzt. So wird z.B. aus dem Inhalt:
	`20010&id=2044`
	der Link
	`http://<URL der Webapplikation>>/servlet/SuperXmlMaske?tid=20010&id=2044`
	Die Ergebnisseite wird dann um einen Button ergänzt.

nextdelete	Link auf eine andere SuperX-Maske; Im Unterschied zu nextmask wird hier ein anderes Icon gewählt: Die Ergebnisseite wird dann um einen Delete-Button  ergänzt.
nextedit	Link auf ein DBForms-Formular relativ zur URL des Servlets. die Ergebnisseite wird um einen "Bearbeiten"-Button  ergänzt.
nextmail	Feldinhalte werden um einen Mailto-Tag ergänzt. z.B. info@superx-projekt.de wird zu `<a mailto:" info@superx-projekt.de"> info@superx-projekt.de</a>`

url	Feldinhalte werden um einen href-Tag (sowie wenn nötig um ein "http" ergänzt. z.B.
	`www.superx-projekt.de`
	wird zu
	`<a href="http://www.superx-projekt.de">www@superx-projekt.de</a>`

nextlink	Link auf eine externe Seite oder eine andere SuperX-Tabelle; anders als bei nexttable wird ein frei wählbarer textueller Link angegeben, wobei der Volltext des Links und der eigentliche Linkt durch ein Trennzeichen "|" getrennt sind.
	So wird z.B. der Feldwert "Erläuterungen|http://www.erlaeuterungen.de" wie folgt ersetzt:
	Erläuterungen
	Wenn nach dem Trennzeichen keine externe Web-Adresse angeboten wird (erkennbar am vorangestellten "http:"), dann wird der Inhalt des Feldes um den Pfad zum Tabellen-Servlet ergänzt: So wird z.B. aus dem Inhalt:
	`Details zur Hochschule|20010&id=2044`
	der Link
	`<a href=../servlet/SuperXmlTabelle?tid=20010&id=2044>Details zur Hochschule</a>`

2.4.9 Einzelne Zellen/Spalten formatieren (CSS)

Beispiel Spalte verfuegbar, es muss eine versteckte Spalte hiddenverfuegbarcss geben, dass enthält hiddencss-Klasse, z.B. neu in kern44 td.boldnumber

Auszug

```
create temp table tmp_erg
(
titel char(5),titelbez varchar(255),sort smallint,
 hhans decimal (14,2),
 sperre decimal (14,2),
 reste decimal(14,2),
 akts decimal(14,2),
 einnahmen decimal(14,2),
 ausgaben  decimal (14,2),
 fest   decimal (14,2),
 verfuegbar  decimal (14,2),
 hiddenverfuegbarcss varchar(200)
) <@informixnolog/>;
…
update tmp_erg set hiddenverfuegbarcss='boldnumber' where sort=0;
select  titelbez, hhans,sperre ,reste,akts,einnahmen,ausgaben,fest,verfuegbar+fest,verfuegbar,
hiddenverfuegbarcss from tmp_erg order by sort,titel;
```

entsprechende XIL_proplist-Definition muss es auch geben

2.4.10Spaltenlayout in Ergebnistabellen

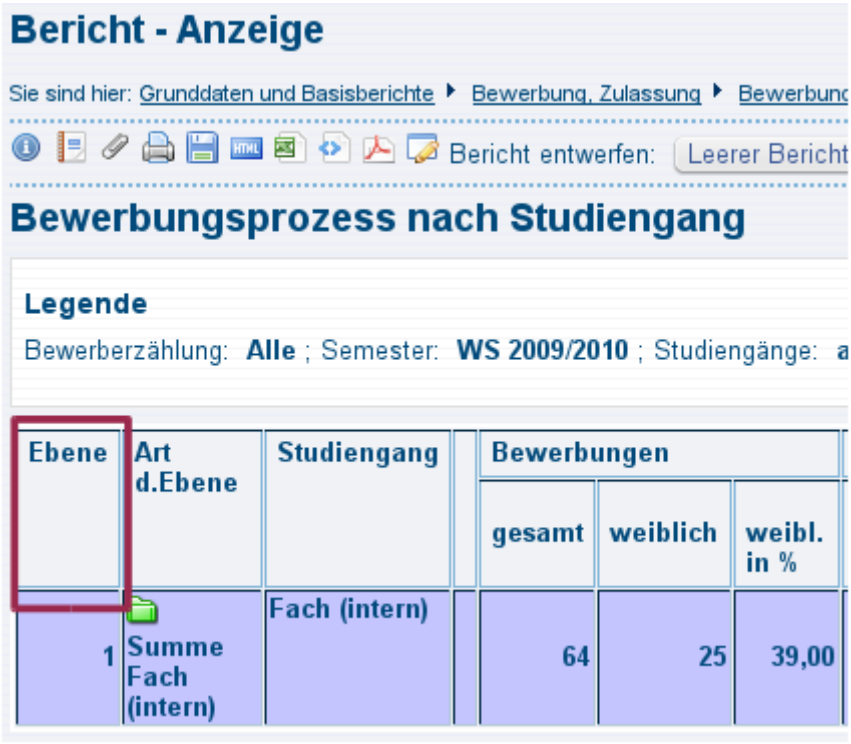
Wie im Tutorial gezeigt, wird das Spaltenlayout (Überschriften, Breite) in der sog. "xil_proplist" gesteuert (der Name stammt übrigens vom früher im SuperX-Windows-Client eingesetzten XVT-Compiler zur Layoutdarstellung). Das Format ist etwas eigenwillig und soll hier erläutert werden.

2.4.10.1 Die Attribute in der xil_proplist

Zunächst ein Beispiel: der Code für die Maske "Bewerbungsprozess nach Studiengang" im ZUL-Modul beginnt wie folgt:

```
XIL List
   sizable_columns horizontal_scrolling
 drop_and_delete movable_columns
 white_space_color=COLOR_WHITE fixed_columns=2
   min_heading_height=55
Column CID=0 heading_text="Ebene" center_heading
   row_selectable heading_platform readonly
   width=5 text_size=20 explanation=""
```

Hier sehen Sie die Ausgabe:



Der Code stammt wie gesagt von einem alten Windows-Client und wurde nur aus Gründen der abwärtskompatibilität übernommen. Nur die fett hervorgehobenen Code-Teile werden überhaupt ausgewertet. Wichtig ist aber, daß die Absatzstruktur des vorhandenen Dokuments beibehalten wird (d.h. jede Spalte

ist in einem Absatz definiert, der mit "Column" beginnt). Am Ende der gesamten `Xil_proplist` befindet sich die Endemarke "@@@" in einem neuen Absatz.

Hier eine Erläuterung der Attribute:

Name	Bedeutung
Column	Definiert eine neue Spalte. Achtung: die Anzahl der "Column"-Anweisungen muss mit der Anzahl der Spalten übereinstimmen, die beim **select_stmt** geliefert werden.
heading_text	Die Spaltenüberschrift in der Ergebnistabelle. Hier sind noch spezielle Layoutanweisungen möglich (s.u.), außerdem können Sie Glossare nutzen
width	Die Spaltenbreite in Zeichen. Diese Anweisung wird im HTML-Layout nicht ausgewertet. Im PDF-Layout wird sie relativ ausgewertet: Alle Spaltenbreiten werden addiert, und zum DIN-A-4-Querformat in Beziehung gesetzt, und dann werden alle Spalten prozentual auf cm heruntergerechnet. In Excel werden die Breiten in Zeichen umgesetzt.
explanation	Erläuterungstext, der zu der Ergebnistabelle in einem separation Fenster angezeigt werden kann. Achtung: Wenn Sie `explanations` einsetzen, müssen alle Spalten dieses Attribut haben. Bitte nicht nur einzelne Spalten dokumentieren. Im Notfall schreiben Sie nur die Spaltenüberschrift rein.

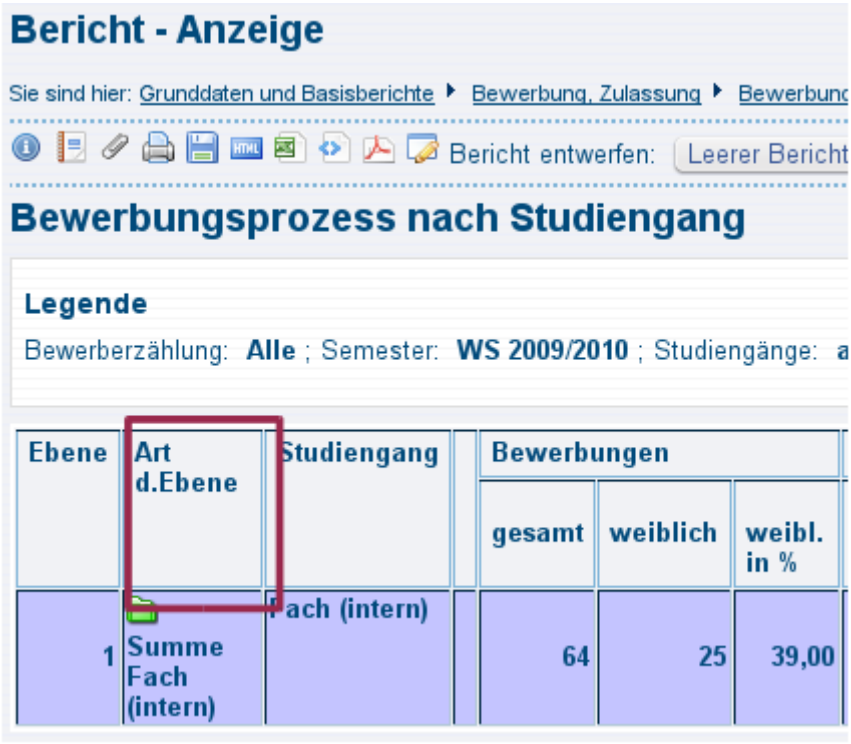
2.4.10.2 Mehrzeilige Spaltenüberschriften

Im Attribut "`heading_text`" können auch mehrzeilige Spaltenüberschriften definiert werden. Fügen Sie den Zeilenumbruch aber nicht direkt ein, sondern codieren Sie diesen als "\n". Bei der Ausgabe wird dies als Umbruch umgesetzt.

Beispiel: Der Code

```
Column CID=0 heading_text="Art\nd.Ebene" center_heading
   row_selectable heading_platform readonly
   width=10 text_size=20 explanation="@@@sos_ebene@@@"
```

sieht in der Ausgabe so aus:



2.4.10.3 Verknüpfte Spaltenüberschriften

Um Spaltenüberschriften zu verknüpfen, muss man wie folgt vorgehen:

Alle Zellen, die verknüpft werden sollen, müssen den gleichen Namen haben, und mit dem Steuerzeichen "`\000`" sowie einem Zeilenumbruch "`\n`" enden.

Beispiel: Der Code

```
Column CID=4 heading_text="Bewerbungen\000\n gesamt"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10 explanation="Bewerberanzahl "
Column CID=4 heading_text="Bewerbungen\000\n weiblich"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10 explanation=""
Column CID=5 heading_text="Bewerbungen\000\nweibl. in %"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10 explanation=""
```

sieht in der Ausgabe so aus:

2.4.10.4 dynamische Spaltenanzahl

kann mit Freemarker realisiert werden, Einfaches Beispiel

nur bei FIN_Quellsystem 1 (MBS) Soll Ansatz ausgegeben werden.

abschluss-Select in masken-sql

select name,<#if K_FIN_Quellsystem=1> hhans</#if>, einnahmen, ausgaben from fin;

XIL

```
Column CID=4 heading_text="Name"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
<#if K_FIN_Quellsystem=1>
Column CID=4 heading_text="Ansatz"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
</#if>
Column CID=4 heading_text="Einnahmen"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
Column CID=4 heading_text="Einnahmen"   center_heading
```

```
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
```

weiteres Beispiel:
Spalte Bewilligung soll nur angezeigt werden, wenn werte größer 0
in masken sql

```
<sqlvars>
<sqlvar bewilligungen>select sum(bewill) from fin where ins=<<Ins>> </sqlvar>
</sqlvars>
```

Abschluss-select

```
select name,ansatz<#if bewilligungen&gt;0> , bewill, </#if> , ausgaben,verfuegbar from tmp_erg.
```

XilPropliste
XIL

```
Column CID=4 heading_text="Name"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
<#if bewilligungen&gt;0 >
Column CID=4 heading_text="bewill"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
</#if>
Column CID=4 heading_text="Ausgaben"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
Column CID=4 heading_text="verfügbar"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=10
```

Bisher musste dafür in der XIL-Proplist der sqlvar-Block wiederholt werden, in der aktuellen superx-version stehen aber die sqlvars aus der masken-sql automatisch zu verfügung ;-)

hochschulspezifische Einstellungemöglichkeit
Der Standardbericht sollte so viele Spalten wie max benötigt enthalten, in der Standardauslieferung werden alle Spalten gezeigt.
Beispiel standard schluss-select

```
select name, akt_soll,einnahmen,aus,verfuegbar from tmp-erg
```

Wenn eine Hochschule nicht alle Spalten sehen möchte, oder die Spalten in einer anderen Reihenfolge (!), folgendes Vorgehen:

in sx_repository eine Variable definieren CUSTOM_XXXXXX (Maskennummer)

assign der einen Hash definiert

```
<#assign customize={"resulttable":[
{"field":"name","caption":"Gliederung","width":14},
{"field":"akt_soll","caption":"Ansatz","width":15,"explanation":"Haushaltsansatz inkl.Einnahmen und
Reste (Haushalterisch: Aktuelles Soll)"},
```

{"field":"einnahmen","caption":"Einnahmen","width":10},
{"field":"aus","caption":"Ausgaben","width":10},{"field":"fest","caption":"Festgelegt","width":10},
{"field":"verfuegbar","caption":"verfügbar","width":12,"explanation":"@@@fin_verfuegbar@@@"}
]}/>

es wird ein customize-hash defniert, darin sind felder definiert, die angezeigt werden sollen.
(Nach Eingabe/Änderung manager-cache leeren!)

So wie oben ist die gleiche Ausgabe wie Standard.
Reihenfolge ändern: verfügbar nach vorne

<#assign customize={"resulttable":[
{"field":"name","caption":"Gliederung","width":14},
{"field":"verfuegbar","caption":"verfügbar","width":12,"explanation":"@@@fin_verfuegbar@@@"},
{"field":"akt_soll","caption":"Ansatz","width":15,"explanation":"Haushaltsansatz inkl.Einnahmen und
Reste (Haushalterisch: Aktuelles Soll)"},
{"field":"einnahmen","caption":"Einnahmen","width":10},
{"field":"aus","caption":"Ausgaben","width":10},{"field":"fest","caption":"Festgelegt","width":10}
]}/>

Will eine Hochschule zum verfügbar vorn stehen haben, und akt_soll gar nicht sehen, muss Eintrag so aussehen

<#assign customize={"resulttable":[
{"field":"name","caption":"Gliederung","width":14},
{"field":"verfuegbar","caption":"verfügbar","width":12,"explanation":"@@@fin_verfuegbar@@@"},
{"field":"einnahmen","caption":"Einnahmen","width":10},
{"field":"aus","caption":"Ausgaben","width":10},{"field":"fest","caption":"Festgelegt","width":10}

]}/>

field ist Feldname in der tmp_erg-tabelle, caption Spaltenüberschrift, width für xil-proplist und explanation ggfs. auch

man muss genau auf richtige anzahl von { } , achten, sonst kommt beim Maskenaufruf interpretation error

```
im Masken-sql gegen Ende baut man ein
<#if CUSTOM_xxxxxx?exists>
<#assign inlineTemplate=CUSTOM_xxxxxx?interpret>
<@inlineTemplate/>
</#if>

wenn eine Hochschule ein custom_xxx angelegt hat, wird der Inhalt interpretiert und ein custo-
mize-Hash steht zur verfügung



 Abschluss-select prüft also
<#if customize?exists> --customize existiert, abschluss select daraus aufbauen
```

```
select <#foreach f in customize.resulttable>
 ${f.field} <#if h_has_next>,</#if>
</#foreach>
 from tmp_erg2 ;
 <#else> -- kein customize objekt existiert, standard abschluss select
 select name,akt_soll,einnahmen,ausgaben,verfügbar from tmp_erg;
 </#if>
```

Für die XIL-Proplist muss es genauso laufen:

```
XIL List
--freemarker template
 <#if CUSTOM_xxxx?exists>
<#assign inlineTemplate=CUSTOM_xxxx?interpret>
<@inlineTemplate/>
</#if>

<#if customize?exists>--wenn Hochschul-customizeobjekt exisit xil dynamisch aufbauen
<#foreach f in customize.resulttable>
Column CID=0 heading_text="${f.caption}"   center_heading explanation="<#if f.explanation?exists>$
{f.explanation}</#if>"
row_selectable col_selectable heading_platform readonly width=${f.width}
</#foreach>
<#else> --standard xil list
Column CID=0 heading_text="Name explanation="" center_heading    row_selectable col_selectable hea-
ding_platform readonly width=9 text_size=0
Column CID=0 heading_text="aktsoll explanation="" center_heading    row_selectable col_selectable
..
</#if>
```

complex, but coool, läuft schon mit 3/2010 superx4.0.jar

for the super nerds,

Technik kann man sogar mit eigenen freemarker funktionen verbinden, z.B. dynamsiche spalten nach customizing und Link-Spalten zur Einzelbuchungen nur anzeigen, wenn Rechte für Einzelbuchungen da sind

einfach aus dem Kontext

```
XIL
<#function isWanted field>
<#assign result=true>
<#if field?starts_with('linkbuch')&&Einzelbuchrecht?exists&&Einzelbuchrecht?is_number&&Einzelbuch-
recht=0><#assign result=false/></#if>
<#if (field='einnahmen'||field?starts_with('linkbuchein')||field='offsoll_e'||field?
starts_with('linkbuchoffsolle'))&&"<<Einnahmen anzeigen>>"="'nein'">
<#assign result=false/></#if>
<#return result>

<#if customize?exists>
<#foreach f in customize.resulttable>
<#if isWanted(f.field)>
Column CID=2 heading_text="${f.caption}"   center_heading explanation="<#if f.explanation?exists>$
{f.explanation}</#if>"
row_selectable col_selectable heading_platform readonly width=${f.width}
```

```
</#if>
</#foreach>
<#else>
...
```

2.4.10.5 Dezimalstellen variieren

Normalerweise werden Werte mit Dezimalstellen immer zweistellig wiedergegeben. Im Ausnahmefall kann man dies ändern, indem man eine Spalte mit dem Namensschema "hidden"+Spaltenname+"dp" hinter die jeweilige Spalte setzt, und der Inhalt der Spalte enthält die Zahl der Nachkommastellen (0-6 möglich.

Beispiel:

```
select ... plan_soll ,
1::integer as hiddenplan_solldp,...
```

bewirkt, dass die Spalte plan_soll einstellig dargestellt wird.

Achtung: dies klappt nicht bei verschachtelten Spaltenlayouts .

2.4.10.6 Standardabfragen mit hochschulspezifischen Details versehen

Bei ganz Standardabfragen, bei denen nur Kleinigkeiten hochschulspezifisch angepasst werden sollen. Kann man einen Block einbauen, z.B.

```
<#if K_hs_nr=6850> --HFT Stuttgart
update tmp_erg set fest=0 where jahr!=year(today()); --Festlegungen nur bei aktuellem Haushaltsjahr
</#if>
```

Freemarker greift mit K_hs_nr auf die Hochschulnr aus der Tabelle hochschulinfo zu,der update wird also nur an der HFT Stuttgart gemacht.

eine dynamsiche Spaltenanzahl kann man mit CUSTOM_xxxx repository-Objekt definieren, siehe Abschnitt vorher zu dyn. Spaltenanzahl.

Dies kann man auch erweitern, in dem man im CUSTOM_xxx Repository-objekt nicht nur cusotmize-Objekt defienert, sondern auch weitere Parameter, z.B. FIN-Abfrage zeigt Festlegungen immer an, einige Hoch shculen wollen Festlegungen nur für aktuelles Haushaltsjahr, man kann z.B. in CUSTOM_XXX zusätzliche Variable defnieren

```
<#assign FestlegungenNurAktuellesJahr=true/>
```

in masken-sql der Maske definiert man dann zunächst einen default-wert

```
<#assign FestlegungenNurAktuellesJahr=false/> -- default alle festelegungen
<#if CUSTOM_xxxxxx?exists> - falls custom_XXXX für die maske exisiert wird es aufgerufen
<#assign inlineTemplate=CUSTOM_xxxxxx?interpret>
<@inlineTemplate/>
-- ausführen der Definition überschreibt default-wert von FestlegungenNurAktuellesJahr  mit <#assign
FestlegungenNurAktuellesJahr=true/>
```

```
</#if>

<#if  FestlegungenNurAktuellesJahr>
 update tmp_erg set fest=0 where jahr!=year(today());
 </#if>
```

Standard in der Maske ist also alle Festlegungen anzeigen, wenn eine Hochschule nur die vom aktuellen Jahr will, kann sie custom_xxx repository-Objekt anlegen mit <#assign FestlegungenNurAktuellesJahr=true/>
und bekokmmt nur die aktuellen.

you#re welcome!

2.4.10.7Anzeigen von Balkendiagrammen in der Tabelle

Es gibt die Möglichkeit in der Ergebnistabelle vertikale Balken anzeigen zu lassen. Dies kann z.B. dafür genutzt werden um ein Balkendiagramm darzustellen. Die übergebene Zahl sollte vom Typ her Integer sein. Die Zahl entspricht dann den Pixelwert der Länge des Balkens. Wenn Ihnen der Wert zu klein ist, können Sie durch Multiplikation den Balken verlängern.
Um diese Funktion zu nutzen, muss der übergebene Feldname mit _graph beginnen. Damit wird diese Funktion aktiviert.
Hier ein Beispiel:

Note	Anzahl	Prozent	Graph
1,00	13,00	2,00	
1,30	116,00	17,85	
1,70	172,00	26,46	
2,00	181,00	27,85	
2,30	118,00	18,15	
2,70	37,00	5,69	
3,00	12,00	1,85	
3,30	1,00	0,15	

2.5Verbindung zu externen Datenbanken (operatives Reporting)

Ab Kern4.6 ist es möglich, Verbindungen zu externen Datenbanken aufzubauen.
Verbindungen zu externen Datenbanken werden in der Tabelle dbconnections hinterlegt.

```
CREATE TABLE dbconnections
(
  name      varchar(255)   primary key,
  driver    varchar(255),
  url       varchar(255),
```

```
    username    varchar(255),
    passwort    varchar(255),
    minidle     smallint,-- für ConnectionPool, kann leer bleiben
    maxidle     smallint, -- für ConnectionPool, kann leer bleiben
    maxactive   smallint,-- für ConnectionPool, kann leer bleiben
    testsql     varchar(255), -- für ConnectionPool zum Prüfen ob Connection läuft kann leer bleiben
);
```

Wenn eine Abfrage auf eine externe Datenbank gehen soll, kann man dies derzeit spezifizieren im

- felderinfo.relation
- felderinfo.defaultwert
- maskeninfo.select_stmt

Die Syntax lautet --#db:NAME#
Der Name bezieht sich auf die erste Spalte in der Tabelle dbconnections.

Zu Datenschutzzwecken kann man eingeschränkte User anlegen, die nur einzelne Tabellen oder auch nur Views darauf sehen dürfen.
Dabei kann z.B. auch ein View angelegt werden, der nicht den vollständigen Namen von Prüflingen enthält, sondern nur die Anfangsbuchstaben.

```
create view pos_schwarz as
SELECT L.mtknr,
L.abschl,
A.dtxt as abschluss_str,
L.stg,
G.dtxt as stg_str,
L.pversion,
L.pstatus,
L.pnote,
substring(S.nachname from 1 for 1) as nachname,
substring(S.vorname from 1 for 1) as vorname,
F.hrst,
L.psem
FROM  k_stg G, k_abint A, sos S, lab L left outer join stg F
on (L.mtknr=F.mtknr
and F.semester=L.psem
and F.stgnr=L.stgnr
)
where G.refstg=L.stg
and A.abint=L.abschl
and S.mtknr=L.mtknr
```

```
and L.pnr=9000;
```

2.6Abfragemakros (einschl. Schleifen u. Grafiken)

Makros sind Abfragen, die mehrere andere Abfragen hintereinander ablaufen lassen.

Makros haben die üblichen Einträge in `maskeninfo`, `felderinfo`, `masken_felder_bez`, `maske_sachgeb_bez` etc.

Die Spalte `macro` in der `maskeninfo` hatte im vorherigen Jahrtausend mal was damit zu tun, ob eine Maske ein Makro ist oder nicht – jetzt steuern die Einträge in der Spalte nur, ob eine Maske nur im Applet, nur im XML-Frontend oder in beiden erscheinen soll.

Der Eintrag sollte so sein, dass das Makro nur im XML-Frontend erscheint, da nur das XML-Frontend Makros unterstützt.

Welche Einzelabfragen ein Makro ausführen soll, wird in macro_masken_bez eingetragen.

maskeninfo_id1	maskeninfo-tid des makros,
maskeninfo_id2	maskeninfo-tid der Einzelabfrage,
active	0/1 deaktivieren möglich
sortnr	zur Reihenfolgebestimmung der Durchführung

also z.B.

maskeninfo_id1	maskeninfo_id2
12000	10000
12000	10050

Das Makro 12000 führt die Einzelabfragen 10000 und 10050 aus.

Das „`select_stmt`" eines Makros in der `maskeninfo` wird nicht benutzt, wichtig sind hingegen die Felder, z.B. Lehreinheit und Semester.

Diese Felder werden zur Auswahl angeboten, wenn das Makro im XML-Frontend angeklickt wird. Klickt man auf Abschicken werden die einzelnen Unterabfragen durchgeführt, wobei das ausgewählte Semester und die ausgewählte Lehreinheit auch automatisch in jeder Unterabfrage ausgewählt werden (Die Feldnamen müssen aber gleich sein, heißt das Feld in der Makromaske Lehreinheit, in der Einzelabfrage aber Org.Einheit, kann keine automatische Zuordnung erfolgen. Die Felder müssen gleich heißen oder man muss auf der Makromaske das Feld Org.Einheit zusätztlich aufnehmen).

2.6.1 Makros und Sichten

Wenn in den Submasken eines Makros verschiedene Sichten benutzt werden, müssen auf der Hauptmaske alle in den Untermasken benötigten Sichten auswählbar sein.

Sonst gelten ggfs. vorgenommene Standänderungen nur für die Sichten der Hauptmaske nicht für die der Untermasken.

Alternativ müsste man prüfen, ob Sichten von der Hauptmaske (Organigramm-Sicht) ihren Stand auch auf alle Sichten in den Untermasken übertragen, die von der gleichen Art sind (wahrscheinlich eher gefährlich).

2.6.2 Makro-Schachtelung

Makros sollen geschachtelt werden können, ein Makro soll also Untermakros aufrufen können.

Für ein Statistikheft könnte man dann einzelne Makros zu den Abschnitten des Hefts erstellen und testen und später dann ein „Mastermakro" für das ganze Heft.

Auf Datenbankseite könnte man das recht einfach ermöglichen, wenn auch in `maskeninfo_id2` können Makros angegeben sein können.

z.B. Lehrbericht 10600,
Untermakro Einschreibungen 10800,
Einzelne Maske 10060, 10050

makro_maske_bez

maskeninfo_id1	maskeninfo_id2	sort
10600	10800	1
106005	10050	2
10800	10060	1
10800	10070	2

2.6.3 Schleifenfunktion

2.6.3.1 Grundlagen

Ein Makro soll automatisch mehrmals durchgeführt werden, z.B. für alle Lehreinheiten.

Erweiterung von **makro_maske_bez**

makro_maske_bez

Feld	Beispiel	Kommentar
maskeninfo_id1	10800	
maskeninfo_id2	10050	
active	0/1	0 zum zeitweisen deaktivieren
sort	1	
schleifenrelation	`<<SQL>> select key_apnr,name from organigramm where lehre=1 and orgstrukur=30 order by 2`	
schleifenfeldname	Org.Einheit	
schleifenfstand	1.1.2005 oder <<Org. Einheit-Stand>> oder <<today>>	falls das schleifenfeld einen Stand benötigt
schleifenfsicht	13 oder <<Org. Einheit-Sicht>>	falls das schleifenfeld Sichten unterstützt (art=12) und mehr als eine Sicht zur Auswahl steht, muss die gewünschte Sicht (tid) angegeben werden
aktion		

Im Makro 10800 wird die Maske Stud. Allg (10050) aufgerufen.

Da Feld *schleifenrelation* gefüllt ist, wird `<<SQL>>` ausgeführt und Stud. Allg. mehrmals entsprechend der Anzahl gefundener Einträge ausgeführt, dabei wird immer ein Eintrag das Feld mit dem *schleifenfeldnamen* „Org. Einheit" eingesetzt.

2.6.3.2 Schleife über ein anderes Makro

Um nicht eine einzelne Maske, sondern einen Makro mit mehreren Abfragen mehrmals per Schleife aufzurufen, muss ein übergeordnetes Makro geschaffen werden.

Im folgenden Beispiel gibt es ein Makro, dass Studierenden und Absolventenzahlen ausgibt (15000). Dies wird vom übergeordneten Makro 16000 für alle Lehreinheiten aufgerufen.

maskeninfo_id1	maskeninfo_id2	sort	schleifenrelation	schleifenfeldname
16000	15000	1	<<SQL>> select key_apnr,name from organigramm where lehre=1 and orgstrukur=30 order by 2	Org.Einheit
15000	10050	1		
15000	11930	2		

2.6.4 Spezielle Auswahlwerte hinterlegen

Beim Durchführen von Makros soll es möglich sein, für einzelne Felder einzelner Masken spezielle Auswahlwerte zu hinterlegen.

z.B. bei einem Makro mit 10 Studierendenabfragen, die standardmäßig die Haupthörerzahl ausgeben, soll bei einer Maske stattdessen beim Feld Hörerstatus Nebenhörer ausgewählt sein.

Lehrbericht (10600) ist ein Makro

makro	maske	sort	
10600	10070	1	
	10050	2	
	10090	3	
	10100	4	
	10050	5	
	11500	6	
	10050	7	(hier Nebenhörer gewünscht)
	10050	8	

...

Felderinfo der „MakroMaske" (10600) definiert Felder, die sich auf alle Untermasken beziehen

bei einer einzelnen Untermaske (z.B. Stud. Allg. 10050) werden Felder gefüllt

NEU:

Ein solcher Eintrag überlagert ggfs. in der Makro-Maske vorkommende Felder gleichen Namens!

makro_feld_wert

Makro	Sortnr	feldername	value(char)	sicht	stand
10600	7	Hörerstatus	'N' für Nebenhörer	bei Sichtfeldern z.B. 13 oder <<Org. Einheit-Sicht>>	z.B. 1.1.2005 oder today oder <<Org. Einheit-Stand>>

Die Werte müssen genauso so eingetragen werden, wie sie vom SQL in der relation-Spalte des zugehörigen Felds geliefert werden.

z.B. steht für das Feld Staatsangehörigkeit in relation:

```
<<SQL>> select tid,text from aggre_bland where tid in (0,1) order by text;
```

Ergebnis

tid text

1 Ausland

0 Deutschland

Wenn Ausland vorbelegt werden soll, muss in makro_feld_wert als value nur

1

eingetragen werden.

TODO

Man sollte auch eine gewünschte Sicht mit einem gewünschten Stand hinterlegen können.

2.6.5 Zukünftig: Feldnamen-Synonyme

Wenn besonderes Problem tritt auf, wenn inhaltlich gleiche Felder in unterschiedlichen Masken unterschiedlich heißen, z.B. Institution, Lehreinheit, Org. Einheit oder Semester,Seit Semester, Start-Semester

Auf der Makro-Maske sollte idealerweise nur ein Feld stehen, z.B. Org. Einheit.

(Wenn verschiedene Sichten benutzt werden, müssen alle in den Untermasken benötigten Sichten auswählbar sein, s.o.).

Man müsste irgendwo Synomyme hinterlegen können.

z.B. makro_feld_alias

makro	integer	15000	
feldname	char	Org. Einheit	
alias	char	Lehreinheit	
sortnr	integer	null für grundsätzlich oder sonst sortnr in einem Makro heißt, nur für die Abfrage x and sortnr=3	

2.6.6 Aktionen (Grafikerzeugung)

2.6.6.1 Grundlagen

In der Tabelle macro_masken_bez gibt es eine Spalte `aktion`, in der verschiedene Aktionen definiert werden können.

Derzeit sind folgende Aktionen möglich

Name	**Beschreibung**
showTable	Die Ergebnistabelle anzeigen. Diese Aktion wird standardmäßig durchgeführt, wenn bei aktion nichts angegeben ist.
createChart-x	Eine Grafik erzeugen – s. Kapitel Grafikerzeugung.

Mehrere Befehle sind durch | zu trennen, die Reihenfolge spielt eine Rolle.

`showTable|createChart-1` zeigt erst die Tabelle und danach die Grafik

`createChart-1|showTable` zeigt erst die Grafik und danach die Tabelle

Sofern die Spalte aktion null oder einen Leerstring enthält, wird die Aktion `showTable` durchgeführt.

In den Beispielen wird nach Java Konvention das erste Wort klein und danach jeder Wortanfang groß geschrieben, Sie können aber auch alles klein oder alles groß schreiben.

2.6.6.2 Grafikerstellung

2.6.6.2.1 Grundlagen

Als Erstes muss irgendwo hinterlegt werden, dass nachdem eine Abfrage durchgeführt wurde auch eine Grafik erzeugt werden soll.

Dies geschieht im Feld `aktion` der Tabelle `makro_masken_bez`.

Das Kommando lautet:

createChart-graphicformatTid

z.B.

makro_maske_bez

maskeninfo_id1	maskeninfo_id2	sort	schleifenrelation	Schleifenfeldname	aktion
10800	10050	2			createChart-2

In der Tabelle graphicformat werden Attribute für eine Grafik hinterlegt, die graphicformatTid im Kommando verweist auf einen Eintrag

Tabelle **graphicformat**

tid	
charttype	bar (Säulen), hbar (Balken), pie (Torten), line (Linien)
caption	Studierendezahl (Personen) nach Semestern
width	400
height	400
caption x	Semester
caption y	Studierende
line x (Trennlinien)	0 oder 1
line y	0 oder 1
showvalues (Wert bei Balken anzeigen)	0 oder 1
moreattribs	

Da es ja noch eine ganze Menge weiterer Attribute, z.B. Farben von Balken 1-15, Intervalle auf der Y-Achse (1000,2000,3000 oder 1500, 3000, 5000) etc geben kann, haben wir überlegt, nicht für jedes potentielle Attribut eine eigene Spalte in der Tabelle anzulegen (irgendwas fehlt später stimmt auch noch), sondern ein BLOB-Feld moreattribs, in dem weitere Attribute nach der Syntax

attributname=wert| attributname =wert

eingetragen werden können

z.B.

color1=100,200,0|maxWertYAchse=18000

Wenn es in Ergebnistabellen Leerzeilen gibt, werden diese für die Grafikerstellung automatisch ignoriert.

2.6.6.2.2 MoreAttribs

Die Attribute sind nicht case-sensitive. Statt nach Java Convention ignoreRowsWith: kann man auch ignorerowswith oder IGNOREROWSWITH schreiben.

Die Reihenfolge der Attribute spielt keine Rolle.

Attribute, die bei allen Grafiken anwendbar sind

Name	**Beispiel / Erläuterung**
`ignoreRowsWith:`	ignoreRowsWith:Summe Alle Zeilen der Ergebnistabelle, bei denen in der ersten Spalte das Stichwort „Summe" vorkommt, werden bei der Grafikerstellung ignoriert.
`ignoreColNo:`	ignoreColNo:4 Spalte vier wird nicht ausgegeben
`colorX:`	color1:red|color2:155,0,30|color3:gray30% Definition von Farben über Systemname, mit Komma getrennten RGB-Werten oder Grauabstufung von 0% weiß bis 100% schwarz
`flipFlop`	Ganz zum Schluß werden Zeilen und Spalten ausgetauscht. Einträge für ignoreRows und ignoreColNo beziehen sich auf die ursprüngliche Tabelle

2.6.6.2.3 Säulendiagramme

charttype: bar oder bar3D

moreAttribs

`useOnlyColNo:`	useOnlyColNo:2 wenn nur 2. Spalte ausgegeben werden soll

2.6.6.2.4 Balkendiagramme

charttype hbar oder hbar3D (für horizontal bar)

MoreAttribs

`useOnlyColNo:`	useOnlyColNo:2 wenn nur 2. Spalte ausgegeben werden soll

2.6.6.2.5 Tortendiagramme

Bei einem Tortendiagramm wird üblicherweise die erste und zweite Spalte visualisiert.
Im Beispiel die Hauptnutzfläche aufgeteilt nach Gebäuden.

Ge- bäude	Hauptnutz- fläche (HNF)	Nebennutz- fläche (NNF)	Verkehrs- fläche	Funktions- fläche	Sonstige Fläche	Gesamt- fläche	
GA	31,56	14,00	0,00	0,00	0,00	31,56	
GABF	83,50	15,00	0,00	0,00	0,00	83,50	
GB	810,20	123,00	0,00	0,00	0,00	810,20	
GBCF	43,53	20,00	0,00	0,00	0,00	43,53	
UB – Bibliothek	944,29	183,00	0,00	0,00	0,00	944,29	

Falls nicht die Werte der zweiten Spalten visualiert werden sollen, kann in der Tabelle graphicformars unter moreAttribs der Parameter

`useNo:` angegeben werden, also z.B. `useNo:3` für eine Darstellung der Nebennutzfläche.

Sie können bei Tortendiagrammen auch das Attribut **`flipFlop`** angeben.
Dann wird standardmäßig die zweite Zeile (hier das Gebäude GA) mit HNF,NNF,Verkehrsfläche etc als Torte dargestellt.
Um die Gesamtfläche auszublenden fügen Sie noch **`ignoreColNo:7`** hinzu, das funktioniert da der Austausch von Zeilen und Spalten (flipFlop) ja grundsätzlich als Letztes passiert.
Um statt der zweiten Zeile (GA), die dritte Zeile (GABF) zu visualisieren müssten Sie `useNo:3` angeben.

2.6.6.3 spezielle Stylesheets benutzten

Zukünftig:

Für einzelne Masken innerhalb eines Makros können auch Stylesheets hinterlegt werden, die angewendet werden sollen

aktion

stylesheet-1;stylesheet-2

Nummer ist tid in sx_stylesheets

2.7 Dokumentation von Abfragen

2.7.1 Glossare

Die Statistiken in SuperX ist nicht immer für Außenstehende "selbsterklärend", und insbesondere bei Kennzahlen und kondensierten Werten sollten die Konzepte mit einem Glossar versehen sein.

Die Frontends von SuperX bieten drei Möglichkeiten der Dokumentation:

•Dialogelemente auf den Masken können mit einem "Tool-Tip" versehen werden, d.h. bei Mausbewegung über den Button wird eine Erläuterung angezeigt.

•Ergebnistabellen können mit einem Glossar versehen werden, das die in der Tabelle benutzten Begriffe auf einer zweiten Seite erläutert.

•Umfangreichere Hilfetexte sind über die kontextabhängigen Hilfetexte zu einer Maske und Ergebnistabelle verlinkt. Dies ist im Administrationshandbuch dokumentiert.

Für die ersten beiden Dokumentationsarten wird in SuperX die Tabelle `sx_captions` gepflegt, die Felderläuterungen und allgemeine Schlüsselwörter dokumentieren. Die Dokumentation ist sogar mehrsprachig möglich.

Referenzen auf die `sx_captions` werden über die ID gemacht, Dabei gilt:

•Beim Platzhalter `@@meine_id@@` wird das Feld Beschriftung (kurz) angezeigt

•Beim Platzhalter `@@@meine_id@@@` wird das Feld Beschriftung (lang) angezeigt.

•HTML-Codierung ist möglich.

2.7.1.1 Allgemeine Schlüsselwörter

Allgemeine Schlüsselwörter sind der Tabelle `sx_captions` definiert, man erkennt sie, daran dass die Spalte id gefüllt ist (table_name, field_name und record_no hingegen leer)

tid	id	table_name	field_name	record_no	locale	contents_short	contents_long	sachgebiete_id

1	**studiengang**				**de**	**Studiengang**	**Studiengänge definieren sich durch das Fach, die Vertiefungs-richtung, durch Haupt- oder Nebenfach sowie den Abschluss.**	**16**
2	**studiengang**				**en**	**Subject / Degree**	**A combination of subject and degree as well as the major-minor distinction**	**16**
3	stud_general				de	Studierende allgemein		
4	stud_general				en	students (general)		

Im Beispiel wird der Tag **studiengang** definiert.

Dieser Tag wird an beliebiger Stelle (Maskennamen, Überschriften, select_stmt, XIL-Proplist, XSL-Dateien, etc) durch den Eintrag `contents_short` der aktuellen Locale ersetzt.

2.7.1.2 Der Spezialfall Maskenfelder

Für die Erläuterung von Maskenfeldern können kurze und längere Hilfetexte hinterlegt werden. Die kurzen Texte dienen als Beschriftung des Feldes (überscheiben als den "Feldnamen"), und die langen Texte erscheinen als Tool-Tip bei Mausbewegung auf den Button. Im Ausdruck werden die Maskenfelder wahlweise auf einer separaten Seite dokumentiert.

Damit nicht für jedes einzelne Maskenfeld ein Eintrag gemacht werden muss, kann ein Hilfetext über seinen Namen auch mehreren Maskenfeldern zugeordnet werden; in diesem Fall ist die Spalte `record_no` leer.

Für Felder aus der Tabelle felderinfo schaut SuperX nach, ob in der Tabelle `sx_captions` ein Eintrag für die Tabelle `felderinfo`, `field_name studiengang` und `record_no = 10050` oder `null` vorhanden ist

Im folgenden Beispiel ist ein Maskenbutton "Studiengang" erläutert, der in dieser Weise und bei dem Feld Nummer `10050` dokumentiert sein soll.

tid	id	table_name	field_name	record_no	locale	contents_short	contents_long	sachgebiete_id
9		**felderinfo**	**studiengang**	**10050**	**de**	**Grundständiger Studiengang**	**Ein Studiengang im grundständigen Studium**	**16**
10		felderinfo	studiengang	10050	en	Degree program		16

Wenn Sie den Erläuterungstext bei allen Feldern mit dem Namen "`studiengang`" erscheinen lassen wollen, dann müssen Sie das Feld `record_no` leer lassen.

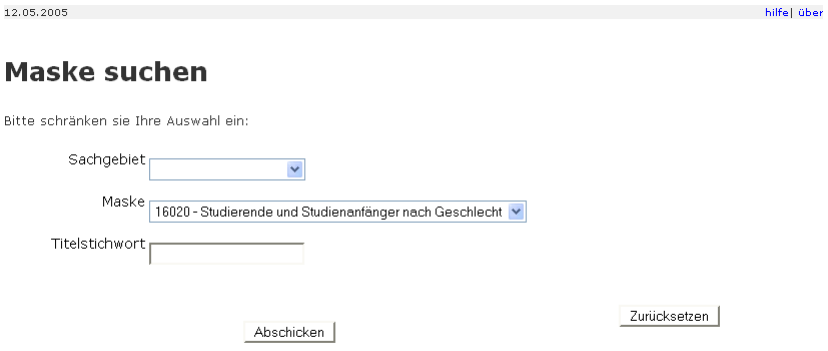
2.7.1.3 Änderung von Glossaren im XML-Frontend

Im XML-Frontend gibt es komfortable Möglichkeiten zur Änderung von Glossaren. Melden Sie sich als Administrator im XML-Frontend an. Zunächst zeigen wir, wie allgemeine Erläuterungen zur Makse sowie Erläuterugnen von Ergebnisspalten erzeugt werden, dann werden Felderläuterungen eingepflegt.

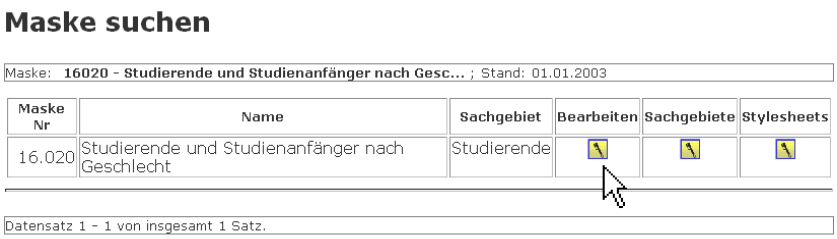
2.7.1.3.1 Maskenerläuterung

Masken bearbeiten wir im XML-Frontend im Menü "Maske Suchen", wie es im Tutorial beschrieben ist.

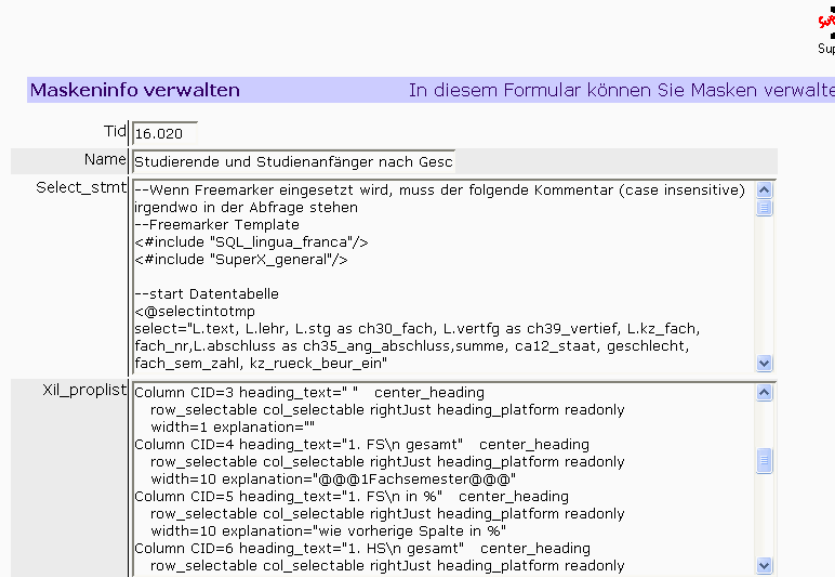
Wir wählen bei Maske suchen eine Maske, die wir dokumentieren wollen, z.B. die Abfrage **Studierende und Studienanfänger nach Geschlecht**.



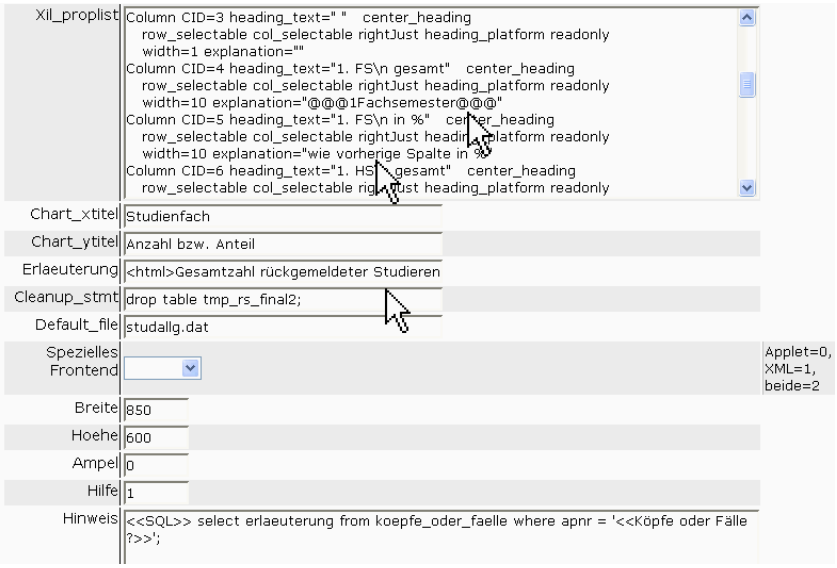
Wir schicken das Formular ab, und erhalten Bearbeitungsmöglichkeiten zur Maske. Wir wählen hier "Bearbeiten".



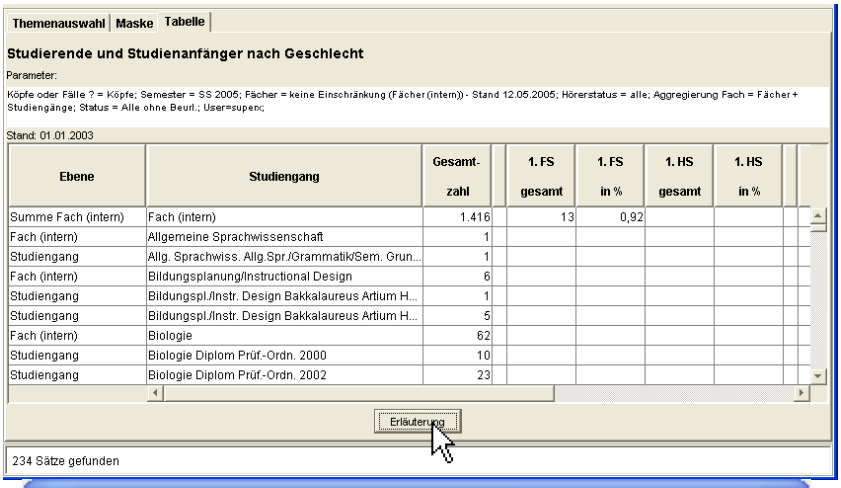
Wir werhalten das Beabrietungsformular der Maske. Diesmal kümmenr wir uns nicht um das Feld "select_stmt", alsod er Script, sondern um die Dokumentation.



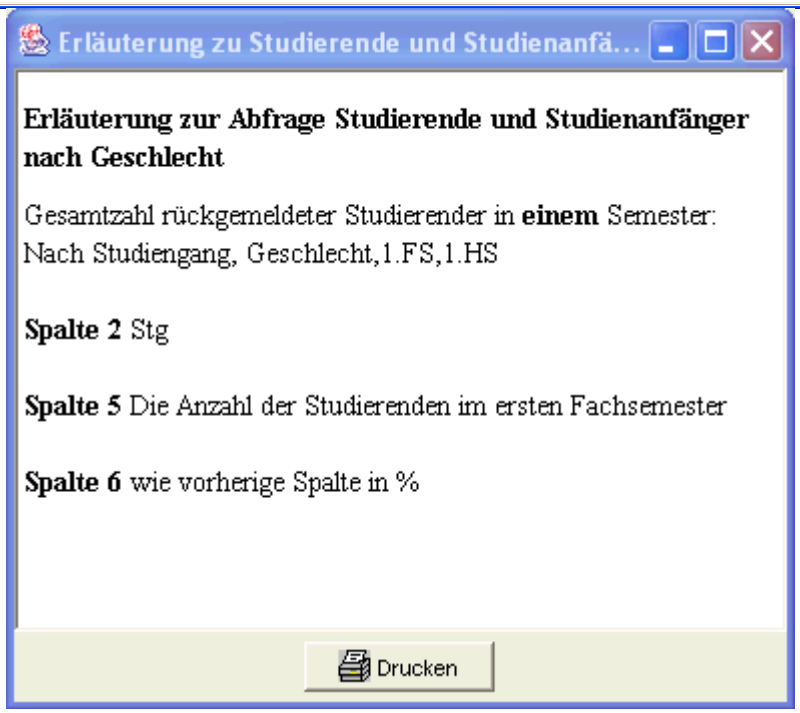
Weiter unten auf dem Formular stehen die Spaltenbeschriftungen, bei Spalte 5 z.B. "wie vorherige Spalte". Zunächst schauen wir uns die Erläuterung der Maske an (Feld "Erlaeuterung").



Im Applet wird der Erläuterungstext bei Mausklick auf den Buton "Erläuterung" angezeigt:

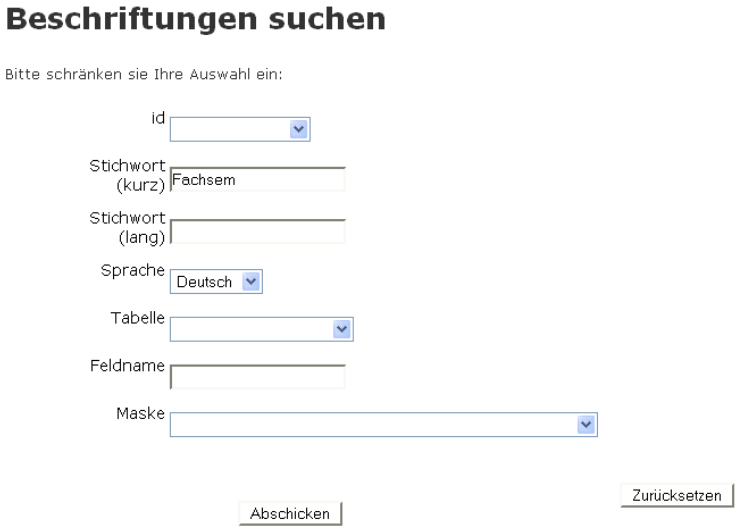


Wie sehen den Text aus der Maskenerläuterung sowie für jede dokumeniterte Spalte den Text der Spaltenerläuterung.



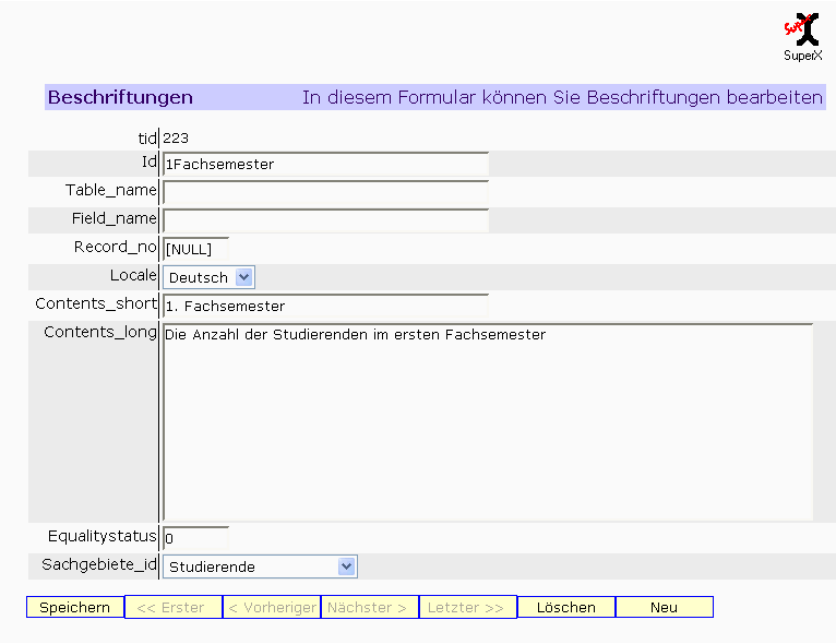
Der Erläuterungstext von Spalte 5 ist ein Sonderfall, hier wurde oben in der Maske mit einem Platzhalter namens "@@@1Fachsemester@@" gearbeitet. Der Inhalt für den Platzhalter wird in der Beschriftungstabelle gepflegt. Dazu müssen wir in das Menü "Administration -> Beschriftung suchen" gehen.

Wir wählen z.B. das Stichwort "**Fachsem**" aus, und schicken das Formular ab.



Es erscheinen als Ergebnis mehrer Zeilen, die erste Zeile enthält unseren Platzhalter (im Feld "id"). Wir bearbeiten diesen Eintrag.

Der Erläuterungstext zur Spalte steht im Feld `contents_long`. Genau dieser Text wurde oben in der Erläuterung der Spalte 5 angezeigt.
Sie speichern die Angaben mit "Speichern".



2.7.1.3.2 Feldbeschriftungen ändern

Für die Erläuterung von Maskenfeldern können kurze und längere Hilfetexte hinterlegt werden. Die kurzen Texte dienen als Beschriftung des Feldes (überschreiben also den "Feldnamen"), und die langen Texte erscheinen als Tool-Tip bei Mausbewegung auf den Button. Im Ausdruck werden die Maskenfelder wahlweise auf einer separaten Seite dokumentiert.

Damit nicht für jedes einzelne Maskenfeld ein Eintrag gemacht werden muss, kann ein Hilfetext über seinen Namen auch mehreren Maskenfeldern zugeordnet werden; in diesem Fall ist die Spalte `record_no` leer.

Für Felder aus der Tabelle felderinfo schaut SuperX nach, ob in der Tabelle `sx_captions` ein Eintrag für die Tabelle `felderinfo`, `field_name studiengang` und `record_no = 10050` oder `null` vorhanden ist

Im folgenden Beispiel ist ein Maskenbutton "Studiengang" erläutert, der in dieser Weise und bei dem Feld Nummer `10050` dokumentiert sein soll.

tid	id	table_name	field_name	record_no	locale	contents_short	contents_long	sachgebiete_id
9		**felderinfo**	**studiengang**	**10050**	**de**	**Grundständiger Studiengang**	**Ein Studiengang im grundständigen Studium**	**16**
10		felderinfo	studiengang	10050	en	Degree program		16

Wenn Sie den Erläuterungstext bei allen Feldern mit dem Namen "`studiengang`" erscheinen lassen wollen, dann müssen Sie das Feld `record_no` leer lassen.

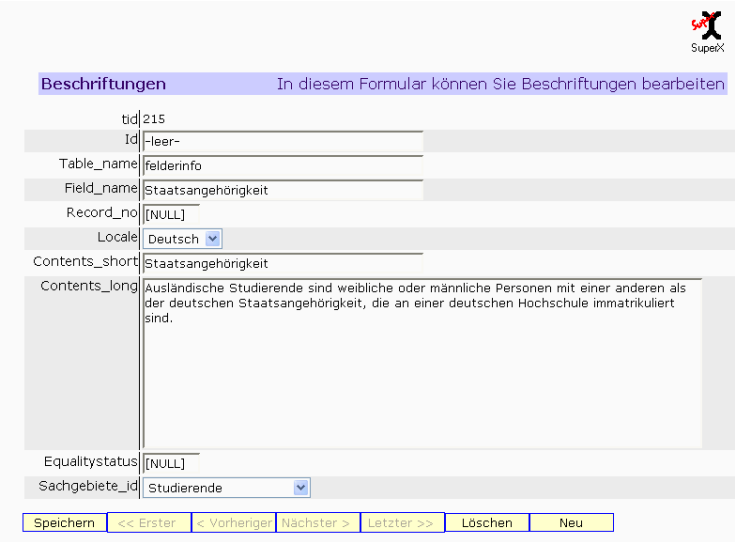
Im folgendne Beispiel wollen wir die Beschriftungen der Maskenfelder bearbeiten. Gehen Sie im Menübaum zum Eintrag

Administration -> **Beschriftungen suchen**.

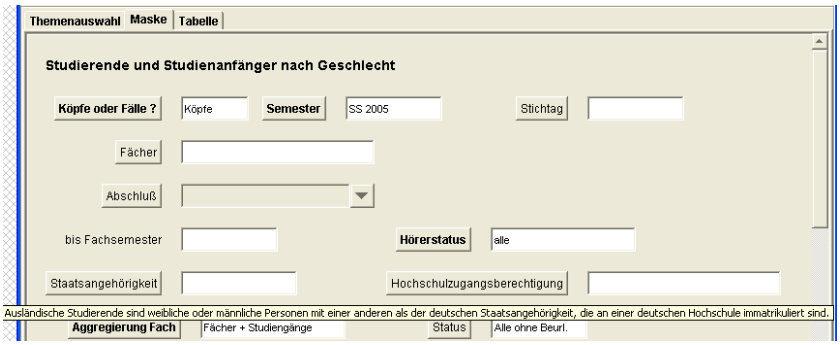
Wir wählen z.B. das Stichwort " **Staats**" aus, und schicken das Formular ab.

Es erscheint als Ergebnis ein Felderinfo-Eintrag zu diesem Thema. Wir sehen, dass nur die Kurzbeschriftung gefüllt ist. Wenn wir das Konzept mit einem ToolTip versehen wollen, gehen wir rechts auf "Bearbeiten"

Wir können eine Langbeschreibung im Feld "contents_long" einfügen. Der dortige Text wird als ToolTip angezeigt. Sie speichern die Angaben mit "Speichern".



Im Applet wird derText dann bei Mausbewegung über den Button **Staatsangehörigkeit** wie folgt angezeigt:



2.7.2 Erzeugung der SuperX-Hilfe im Javahelp-Format

Die SuperX-Hilfe im Applet besteht aus einem Archiv im Javahelp-Format. Die Hilfetexte sind in den Modulen erzeugt und können problemlos integriert werden. Falls Sie eigene Hilfetexte einbinden wollen, müssen Sie wie folgt vorgehen:

1.Erzeugen Sie html-Seiten mit der Hilfe (html 3.2)
2.Binden Sie die Dateien in die Mapping-Datei ein
(`$SUPERX_DIR/webserver/tomcat/webapps/superx/applet/javahelp/map.jhm`)
3.Falls die Hilfeseiten kontextabhängig abrufbar sein sollen, müssen die Titel der Mapping-Einträge folgenden Konventionen folgen:
- Allgemeine Beschreibungen der Abfragen lauten A<<TID>>.htm
- Beschreibungen der Masken lauten M<<TID>>.htm
- Beschreibungen der Ergebnistabellen lauten T<<TID>>.htm

Am Anfang ist es hilfreich, die vorhandenen Hilfetexte als Vorlage zu benutzen.

Die Javahilfe kann auch komfortabler mit dem Memtext-Autorensystem aus einer Word-Datei erzeugt werden. Details dazu siehe http://studio.memtext.de .

2.8 Werkzeuge zur Masken-Entwicklung

2.8.1 Übersicht

Die Masken werden in der Datenbank in der Tabelle `maskeninfo` (und anhängige Tabellen) gespeichert und können dort auch geändert werden. Bei der Installation / Upgrade des Moduls werden die Masken vom Dateisystem in die Datenbank eingespielt. Im Dateisystem liegen die Masken im Verzeichnis

```
SuperX: $SUPERX_DIR/db/module/<<Modulname>>/masken
Edustore: tomcat/webapps/superx/WEB-
INF/conf/edustore/db/module/<<Modulname>>/masken
```

Achtung: wenn Sie die Masken in der Datenbak (d.h. über den Browser oder das Access-Frontend) ändern, dürfen Sie keinen Upgrade/Installe/Uninstall eines Moduls ausführen, die Masken werden dann nämlich ersetzt bzw. gelöscht.

Für die Verwaltung der Masken gibt es Shell-Scripte (Linux) und Edustore-DBI-Scripte (Windows). Zentrale Steuerungsinstanz ist aber die Modul-XML-Datei, in dieser Datei werden die Masken deklariert und in den Themenbaum eingehängt. Und: wenn diese Datei geändert wurde, müssen die Verwaltungsscripte (Installation, Sicherung, Upgrade) neu generiert werden.

2.8.2 Shell-Scripte

2.8.2.1 Masken-Verwaltung

Zum Erzeugen und Verändern von Masken gibt es unter UNIX eine Kommandoschnittstelle, die auf dem Gebrauch folgender Skripte beruht. Die Skripte stehen unter dem Verzeichnis

`$SUPERX_DIR/db/masken`

und erzeugen oder verwenden Dateien in dem gegenwärtigen Arbeitsverzeichnis. Nach dem Einspielen der Datenbank sollten Sie darauf achten, den Dateien Ausführungsberechtigung (`chmod 750 sx_*`) zu geben.

2.8.2.1.1 Eine Maske suchen

Wenn Sie eine Maske suchen, sollten die die Felder `tid` oder `name` in der Tabelle `maskeninfo` durchsuchen. Das folgende Script macht dies automatisch:

sx_search_mask

Aufruf:	sx_search_mask <String>
Aktion:	sx_search_mask sucht die Masken, deren Name <String> enthält
Ausgabe: .	tid, name der gefundenen Masken

2.8.2.1.2 Eine Maske sichern und entladen

Um eine Maske zu sichern, müssen Sie die entsprechenden Einträge in den Tabellen

1. `felderinfo`,
2. `masken_felder_bez`,
3. `maskeninfo`,
4. `sachgeb_maske_bez`,
5. `maske_system_bez`

selektieren und sichern. Für dies gibt es das Script `sx_select_mask`.

sx_select_mask

Aufruf:	`Informix: sx_select_mask <TID>` `Postgres: sx_select_mask_xil <TID>`
Aktion:	`sx_select_mask` entlädt alle Metadaten aus den Tabellen `maskeninfo`, `felderinfo`, `masken_felder_bez`, `sachgeb_maske_bez`, `maske_system_bez` zur Maske mit tid = <TID>.
Ausgabe:	Fünf Dateien: 1.<TID>_felderinfo.unl, 2.<TID>_masken_felder_bez.unl, 3.<TID>_maskeninfo.unl, 4.<TID>_sachgeb_maske_bez.unl, 5.<TID>_maske_system_bez.unl

2.8.2.1.3 Eine Maske neu einfügen

Um eine Maske neu einzufügen, müssen Sie die entsprechenden Einträge in den Tabellen

1. `felderinfo`,
2. `masken_felder_bez`,
3. `maskeninfo`,
4. `sachgeb_maske_bez`,
5. `maske_system_bez`

einfügen. Dafür gibt es das Script `sx_insert_mask`.

sx_insert_mask

Aufruf:	`sx_insert_mask <TID> [<neue TID>] [j]`
Aktion:	sx_insert_mask lädt den Inhalt der fünf Dateien 1.<TID>_felderinfo.unl, 2.<TID>_masken_felder_bez.unl, 3.<TID>_maskeninfo.unl, 4.<TID>_sachgeb_maske_bez.unl, 5.<TID>_maske_system_bez.unl in die jeweiligen Tabellen der SuperX-Datenbank. Mit "j" wird die Sicherheitsabfrage umgangen.

Falls <neue TID> nicht angegeben wird, werden die Metadaten wieder mit der alten TID in die Datenbank eingespielt (=Update).

Falls <neue TID> angegeben wird, werden die Metadaten mit der neuen TID in die Datenbank eingespielt (=Insert). Dabei werden alle TIDs in den abhängigen Tabellen angepasst. So können Masken sehr einfach kopiert werden. Eine neue TID bekommt man durch die Wahl der nächsten Zehnerzahl, die größer als die größte vorkommende Nummer ist. Die größte vorkommende Nummer erhält man durch Ausführung des folgenden SQL-Ausdrucks mit Hilfe des Kommandos SQL-Client:

```
select max(tid) from maskeninfo;
```

2.8.2.1.4 Eine Maske löschen

Um eine Maske zu löschen, müssen Sie die Einträge in den oben genannten Tabellen entfernen. Dafür gibt es das Script `sx_delete_mask`

sx_delete_mask

Aufruf:	`sx_delete_mask <TID>`
Aktion:	`sx_delete_mask` löscht alle Metadaten aus den Tabellen `maskeninfo` , `felderinfo` , `masken_felder_bez` , `sachgeb_maske_bez` und `maske_system_bez` zur Maske mit tid = <TID>.

2.8.2.2 Änderungen an einer Maske vornehmen

1. Selektieren der Metadaten der betreffenden Maske: `sx_select_mask <TID>`
2. Editieren der fünf Metadaten-Dateien „<TID>_..."

3. Abspeichern der neuen Metadaten: `sx_insert_mask <TID>`

2.8.3 Maskenverwaltung in Edustore

Für Windows-Anwender bieten sich die Werkzeuge in Edustore an. Hier werden keine Shellscripte benutzt, sondern XML-Scripte, die mit der HIS-Technologie DB-Interface arbeiten. Diese werden mit Java über die Kommandozeile aufgerufen, einige sind auch direkt in der Webanwendung über den Browser verfügbar.

2.8.3.1 Masken einspielen

Beim Upgrade eines Moduls werden die Masken von der Festplatte wieder neu in die Datenbank eingespielt, d.h. das Modul wird auf seinen Auslieferungszustand zurückgesetzt.

Sie können die Masken über die Kommandozeile einfügen. Das folgende Script benötigt unter DOS ein paar Umgebungsvariablen:

```
java -cp %QIS_CLASSPATH% de.his.edustore.bin.ExecuteDBInterface
databases_<<SPEZIALMODUL>>.xml %QISSERVER_PFAD%\..\superx\WEB-INF\conf\edus-
tore\db\module\<<Modulname>>\conf\his1\edustore_install\edustore_<<Modulna-
me>>_masken_einspielen.xml "$<<Modulname
(großgeschrieben)>>_PFAD=<<WEBAPPS_PFAD>>\superx\WEB-
INF\conf\edustore\db\module\<<Modulname>>"
```

z.B. für das SOS-Modul:

```
java -cp %QIS_CLASSPATH% de.his.edustore.bin.ExecuteDBInterface
databases_meins.xml %QISSERVER_PFAD%\..\superx\WEB-INF\conf\edustore\db\mo-
dule\sos\conf\his1\edustore_install\edustore_sos_masken_einspielen.xml
"$SOS_PFAD=%QISSERVER_PFAD%\..\superx\WEB-INF\conf\edustore\db\module\sos"
```

In Eclipse im Dialog "Run..." brauchen Sie die Umgebungsvariablen nicht, es reicht, die Klasse sowie die Parameter anzugeben. Siehe unten das Beispiel fürs Entladen.

2.8.3.2 Masken entladen

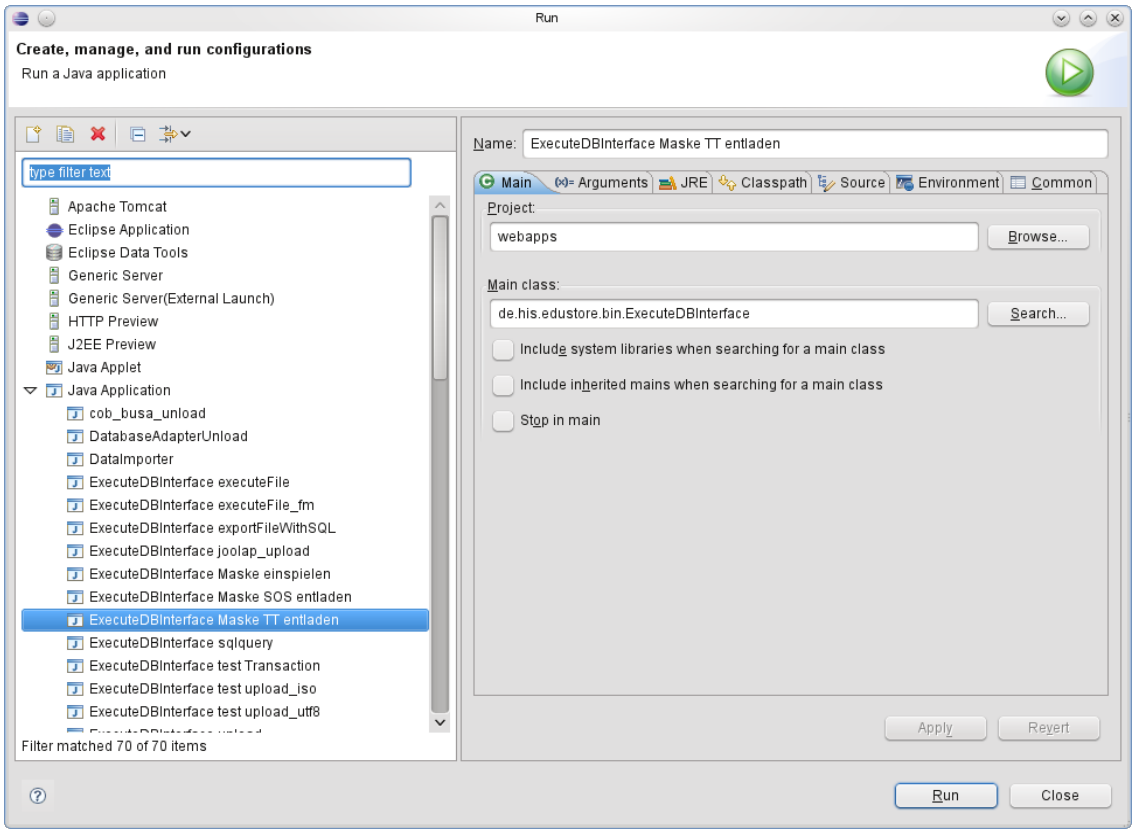
Sie können die Masken eines Moduls wie folgt aus der Datenbank entladen. Das folgende Script benötigt unter DOS ein paar Umgebungsvariablen:

```
java -cp %QIS_CLASSPATH% de.his.edustore.bin.ExecuteDBInterface
databases_<<SPEZIALMODUL>>.xml %QISSERVER_PFAD%\..\superx\WEB-INF\conf\edus-
tore\db\module\<<Modulname>>\conf\his1\edustore_install\edustore_<<Modulna-
me>>_masken_entladen.xml "$<<Modulname
(großgeschrieben)>>_PFAD=<<WEBAPPS_PFAD>>\superx\WEB-
INF\conf\edustore\db\module\<<Modulname>>"
```

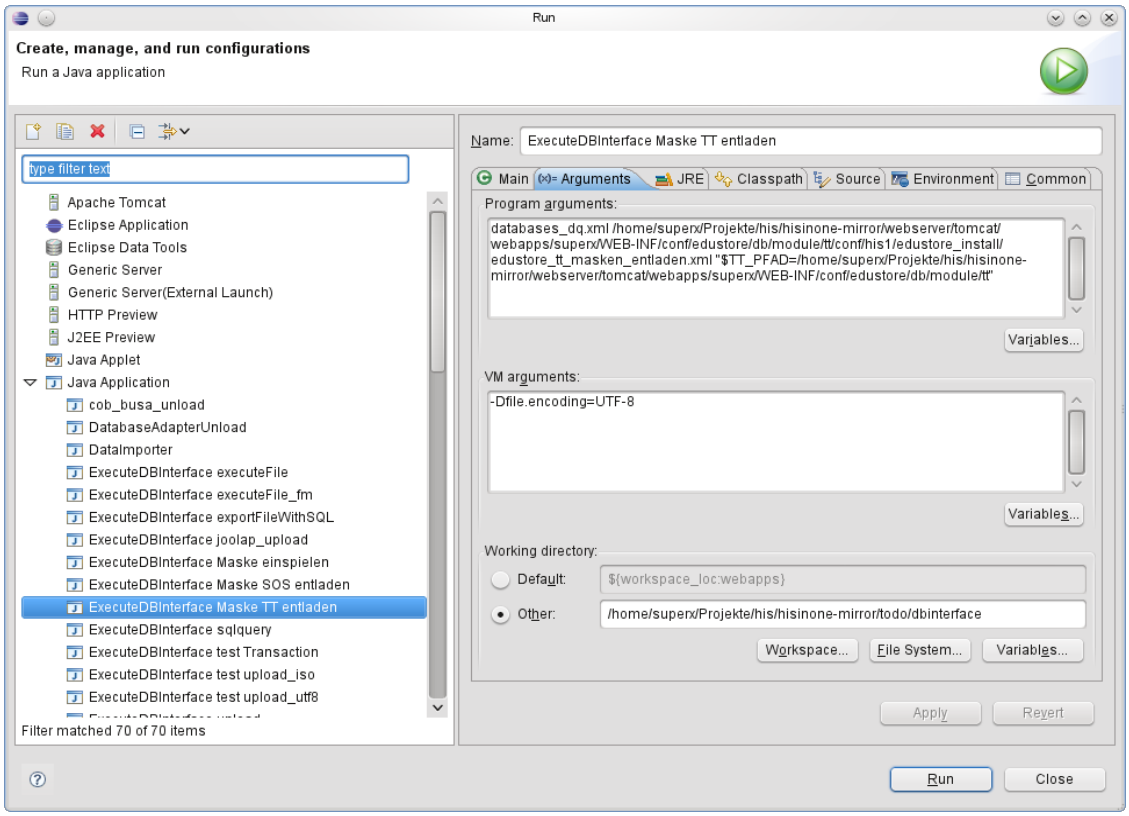
z.B. für das SOS-Modul:

```
java -cp %QIS_CLASSPATH% de.his.edustore.bin.ExecuteDBInterface
databases_meins.xml %QISSERVER_PFAD%\..\superx\WEB-INF\conf\edustore\db\mo-
dule\sos\conf\his1\edustore_install\edustore_sos_masken_entladen.xml
"$SOS_PFAD=%QISSERVER_PFAD%\..\superx\WEB-INF\conf\edustore\db\module\sos"
```

In Eclipse im Dialog "Run..." brauchen Sie die Umgebungsvariablen nicht, es reicht, die Klasse sowie die Parameter anzugeben. Hier zwei Screenshots für Linux und das TT-Modul unter Postgres:



Wichtig ist auch der Parameter `file.encoding`, wenn das zugrunde liegende Betriebssystem ISO als Standardcodierung nutzt:



2.8.4 Webanwendung

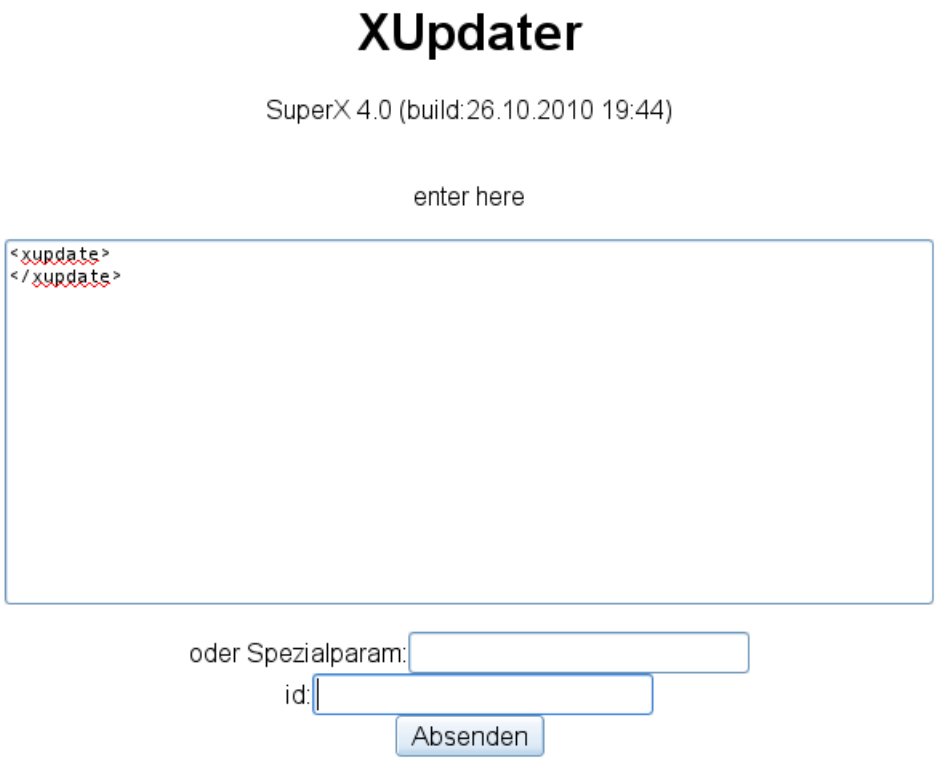
Wenn nur die Grafische Oberfläche zur Verfügung steht, gibt es auch hier eine Möglichkeit die Maske zu sichern (Xupdater).

Aufruf von SuperXManager als Admin

http://<<ServerIP>>:<<Port>>/superx/servlet/SuperXManager?xupdater=true

Beispiel: http://localhost:8080/superx/servlet/SuperXManager?xupdater=true

Die Seite die sich öffnet sollte folgendermaßen aussehen:



Hier gibt es drei Eingabefelder:

•enter here: ist zum Einspielen von Updates/Masken in die Datenbank oder wird als Ausgabe für den zu speichernden Quelltext einer Maske benutzt.

•Spezialparam: hier wird der Parameter eingegeben. Bisher gibt es nur den Parameter "maske".

•Id: ist die Masken-ID. Bei Verwendung des Spezialparams "maske" muss hier die ID der Maske eingegeben werden, welche gesichert werden soll.

Zum Auslesen von Masken bei Feld Spezialparam "maske" eingeben und bei id die Nummer der Maske z.B. 16420.

Dann auf "Absenden" klicken. Im Textfeld erscheint dann der XML mit allen Infos zur Maske. Das ist ein `<xupdate>` mit verschiedenen Unterknoten.

`<themenbaum>` macht Themenbaumeinträge

`<sql> ... </sql>` führt einzelne SQL aus z.B. `delete from maskeninfo where tid = 16420;` oder `insert into felderinfo..`

Für die Text/Blob-Felder gibt es Knoten

```
<text table="maskeninfo" field="select_stmt" where="tid=16420">  .xxx...
</text>
```

Bedeutet: Es wird per JDBC ein Update aufgebaut für die Tabelle `maskeninfo`, mit dem Feld `select_stmt` mit der where-Bedingung `tid=16420` und der Inhalt des Knoten lautet dann: `xxx`

Dank JDBC geht das auch für Informix

Schwierigkeit war wenn in dem Feld `select_stmt` oder `xil_proplist` selbst bei Freemarker-sqlvars ein `CDATA` vorkam, `CDATA` innerhalb von `CDATA` geht per XML nicht. Dies ist nun so gelöst, dass `<[CDATA[[` innerhalb der Felder durch festen Text `CDATASTART` ersetzt wird, und dann beim Einspielen `CDATASTART` wieder durch `<[CDATA[[`. (ob das so 100% klappt bitte beobachten)

Beispieltext:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xupdate>

<!-- fuer Maske 16420-->
<themenbaum maskentid="16420" parentname="Administration Studierende,
Prüfungen"/>
<!--macht beim Einspielen Themenbaumeintrag-->
<!--ggfs. vorhandene Daten Löschen
<sql>delete from maskeninfo where tid = 16420;</sql>
....
<sql><![CDATA[insert into maskeninfo
(tid,name,chart_xtitel,chart_ytitel) values
(16420,'','Fachsemester','Anzahl');]]></sql>
<sql><![CDATA[update maskeninfo  set name='Studiengangsverzeichnis
(lehr_stg_ab)' where tid=16420;]]></sql>
<sql>insert into maske_system_bez (maskeninfo_id,systeminfo_id) values
(16420,7);</sql>
..
<text table="maskeninfo" field="select_stmt"
where="tid=16420"><![CDATA[--Freemarker Template
<#include "SQL_lingua_franca"/>
<#include "SuperX_general"/>
...
</text>
…
```

Um eine Maske einzuspielen, kann man den xupdate-Text nehmen (bei Windows auf Codierung achten!). Nach Aufruf von

`http://<<ServerIP>>:<<Port>>/superx/servlet/SuperXManager?xupdater=true`

den xupdate-Text in das große Textfeld "enter here" kopieren und absenden.

(zum Einspielen darf nichts in Spezialparam oder id stehen).

Wie man an den Knoten sieht, kann man das nicht nur für Maskeneinspielen nutzen, sondern für beliebigen SQL.

Beispiel:

```
<xupdate>
<sql> update felderinfo set buttonbreite=120 where name='Kostenstelle'</sql>
```

```
<text table="sx_captions" field="contents_long"
where="id='fin_dritt'">Erläuterung zu Drittmitteln</text>
</xupdate>
```

Hier gibt es viele Möglichkeiten aber es ist auch Vorsicht geboten, wenn bei

```
<text table="sx_captions" field="contents_long">
```

die Wherebedingung fehlt, heißt das wie in klassischem SQL

```
update sx_captions set contents_long='';
```

und alle Zeilen in der ganzen Tabelle werden auf einen Leerstring gesetzt.

Übrigens ist DOSQL erweitert, so dass man xupdate auch per Shell nutzen kann. Z.B. module_cifx_install.sql

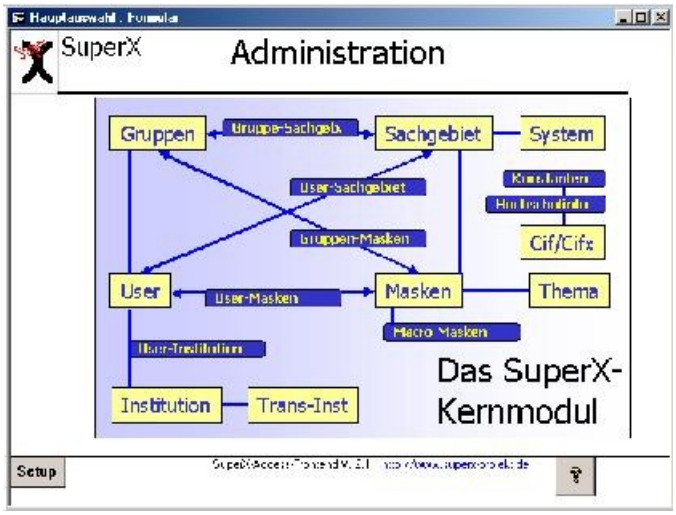
```
<xupdate>
<sql>DELETE FROM fm_templates WHERE id = 'KERN_CIFX_UPDATE';</sql>

<sql>INSERT INTO fm_templates
(id, description, content, version)
select 'KERN_CIFX_UPDATE', 'Makro cifx-update',content, 1 from
fm_templates where id='SuperX_general';

--Content ist leider als not null definiert, daher erst mit anderem Wert
füllen und dann überschreiben
</sql>

<text table="fm_templates" field="content" where="id='KERN_CIFX_UPDATE'">
<![CDATA[
<#macro MODUL_CIFX_UPDATE>
select 'Quellsystem_var ${Quellsystem_var} '::char(30) from xdummy;
select '${SQLdialect}' from xdummy;
...
</text>
```

Man kann mit dem Element <postsql> auch ein oder mehrere SQL-Befehle nach dem TEXT-Update ausführen.

2.8.5 Das Access-Frontend

Die Access-Datenbank enthält die Tabellen des Kernmoduls als Verknüpfungen und ermöglicht so ein leichtes Administrieren der Datenbank. Die Installation ist in der Installationsanleitung für ODBC-Quellen beschrieben. Die folgende Abbildung zeigt das Hauptmenü:

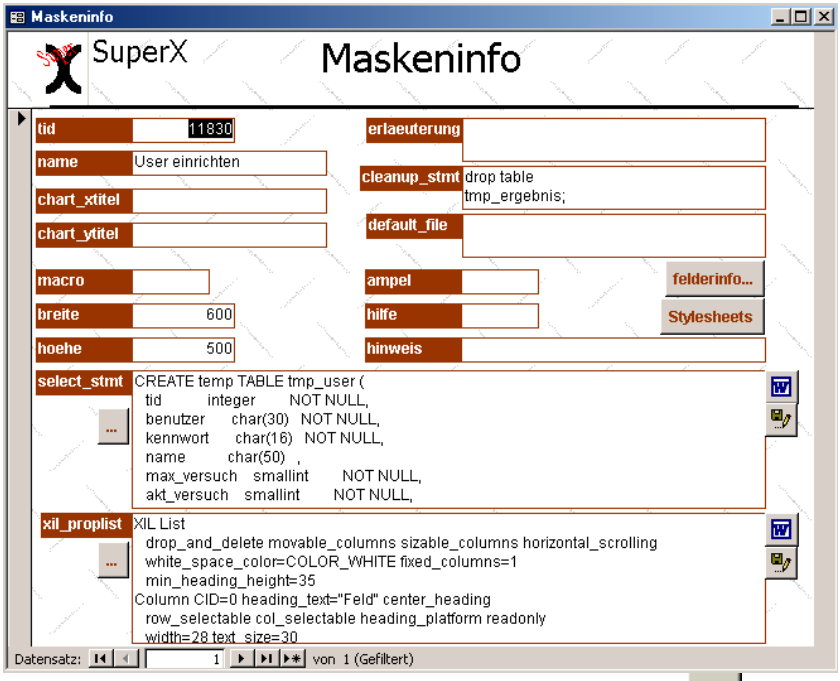
Das Frontend eignet sich zur Verwaltung von Usern, Gruppen, Sachgebieten und Masken sowie deren relationalen Verküpfungen (blaue Kästchen). Darüberhinaus sind Formulare für das Systeminfo, den Themenbaum und das Organigramm vorgesehen.



Probleme mit der Bedienung von Access gibt es immer dann, wenn Tabellen keine Primärschlüssel haben oder wenn die Felder mit den Primärschlüsseln nicht gefüllt sind. Mit der Version 2.1 erhalten alle Tabellen in SuperX (außer Datentabellen und Hilfstabellen, weil diese normal nicht manuell bearbeitet werden) Primärschlüssel. Wenn es dennoch Probleme gibt, empfehlen wir die Java-basierte SQLWorkbench .

Das Access-Frontend ermöglicht die bequeme Änderung von Abfragen (für die Eingabe neuer Masken und Felder empfehlen wir eher die Abfragen im normalen Themenbaum). Nach dem Öffnen der Datei `/db/superx_frontend.mdb` können Sie unter *Masken* die einzelnen Masken von SuperX anwählen und öffnen. Sie erhalten im Formular `maskeninfo` ein Formular, das Eingaben oder Änderungen in der Tabelle `maskeninfo` ermöglicht.

Das Formular ermöglicht es, Masken zu ändern und zu erzeugen. Sie können eine TID vergeben und einen Namen eintragen.



Das `select_stmt` ist ein großes Textfeld und läßt sich besser durch Drücken der  -Taste in einem separaten Fenster bearbeiten. Leider werden Tabulatoren im normalen Windows-Editor nicht korrekt dargestellt, deshalb befinden sich rechts noch zwei Buttons, mit denen Sie Masken in Word[8] editieren können. Mit dem Button  öffnen Sie das `select_stmt` in Word, und können dort Änderungen vornehmen. Mit dem Button  speichern Sie die Änderungen in der Datenbank, und Word wird geschlossen. Bitte beachten Sie, dass Sie die Dateien in Word nicht speichern müssen. Analog können Sie verfahren, wenn Sie das Feld `xil_proplist` bearbeiten. Um in Access sicherzustellen, dass Feldänderungen wirklich in der Datenbank gespeichert werden, sollten Sie sich einen Button zum Speichern von Datensätzen in die Access-Symbolleiste setzen (Extras -> Anpassen -> Befehle -> Datensatz speichern in eine häufig benutzte Symbolleiste ziehen).

Mit dem Button `Felderinfo` gelangen Sie zu den Feldern dieser Maske. Sie können die Felder dort bearbeiten. Bein Hinzufügen neuer Felder müssen Sie allerdings die jeweiligen `tids` manuell in die Tabelle `masken_felder_bez` eintragen.

Analog funktioniert die Bearbeitung der individuellen Stylesheets für eine Maske.

[8] Warum ausgerechnet Word? Das Access-Frontend ist in Visual-Basic-for-Applications programmiert, und nach unserer Erfahrung ist dies der am meisten verfügbare Editor mit VBA-Unterstützung, wenn auch Access (als Teil von MS Office) installiert ist. Der Editor WordPad z.B. bietet keine VBA-Schnittstelle. Uns war außerdem eine ausgefeilte Such- und Undo-Funktion wichtig. Theoretisch könnte man in der mitgelieferten Dokumentvorlage `editblob.dot` im gleichen Verzeichnis auch Autotexte und Makros hinterlegen. Daher: Auch wenn es ungewöhnlich ist, Word als IDE zu benutzen: nach unserer Erfahrung ist es recht praktisch. Fehlt nur noch die farbige Syntaxunterstützung...

2.8.6 Weitere Tools

Durch die odbc- und jdbc-Treiber können beliebige Datenbankfrontends eingesetzt werden. Gute Erfahrungen gerade mit Tabellen ohne Primärschlüssel haben wir mit der SQLWorkbench von Thomas Kellerer gemacht. Exemplarisch für andere jdbc-Clients haben wir dieses Programm näher beschrieben.

2.8.6.1 SQLWorkbench

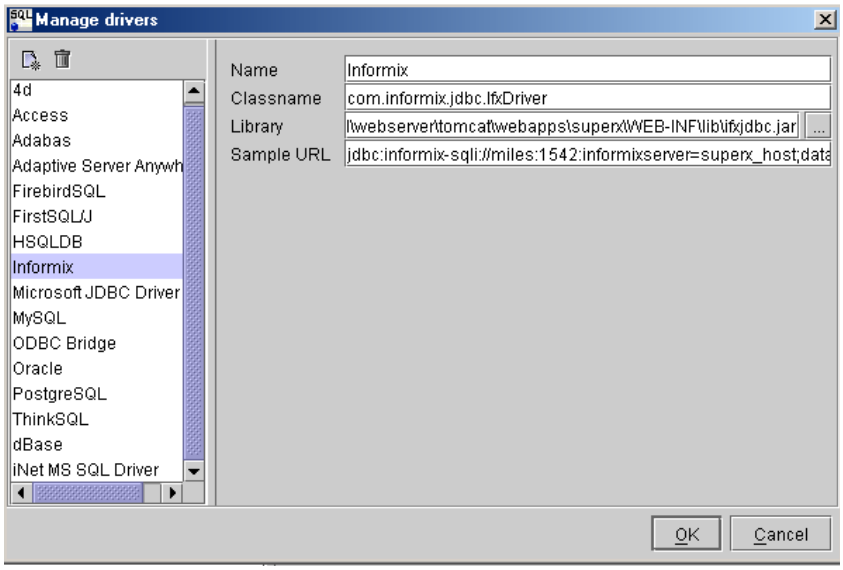
Die SQLWorkbench arbeitet mit dem jdbc-Treiber jeweils von Postgres oder Informix. Sie ist Teil des SuperX-Clientpakets 3.0 oder höher, das Sie vom Download-Bereich des SuperX-Projektes laden können. In dem Clientpaket sind alle notwendigen Treiber und Profile bereits enthalten, deshalb empfehlen wir den Einsatz des Clientpakets.

> Das Clientpaket läßt sich unter Windows und Linux einsetzen, Voraussetzung ist lediglich eine Java Runtime 1.5 oder höher. Bei Nutzung unter Linux und Zeichencodierung der Datenbank im ISO Format sollten Sie in der Shell, die das Script sqlWorkbench.sh ausführt, die Umgebungsvariable `LANG=de_DE@euro` setzen. Sonst werden Umlaute nicht richtig erkannt.

Beim ersten Aufruf der Workbench können Sie Profile für Treiber und Datenbanken eingeben. Musterprofile für viele gängige Datenbanksysteme liegen vor. Leider ist der Informix-Treiber nicht dabei, deshalb muss dieser "von Hand" registriert werden. Gehen Sie dazu über File->Connect in das Feld "Manage Drivers". Dort können Sie einen Namen vergeben und die jdbc-Parameter übertragen. Die folgende Abbildung zeigt ein Beispiel:

Der Dialog zur Einrichtung von Datenbanktreibern am Beispiel Informix.
Die Parameter entsprechen denen, die Sie für das SuperX-Servlet in db.properties definieren.

Der Informix-Treiber `ifxjdbc.jar` muss lokal gespeichert sein.

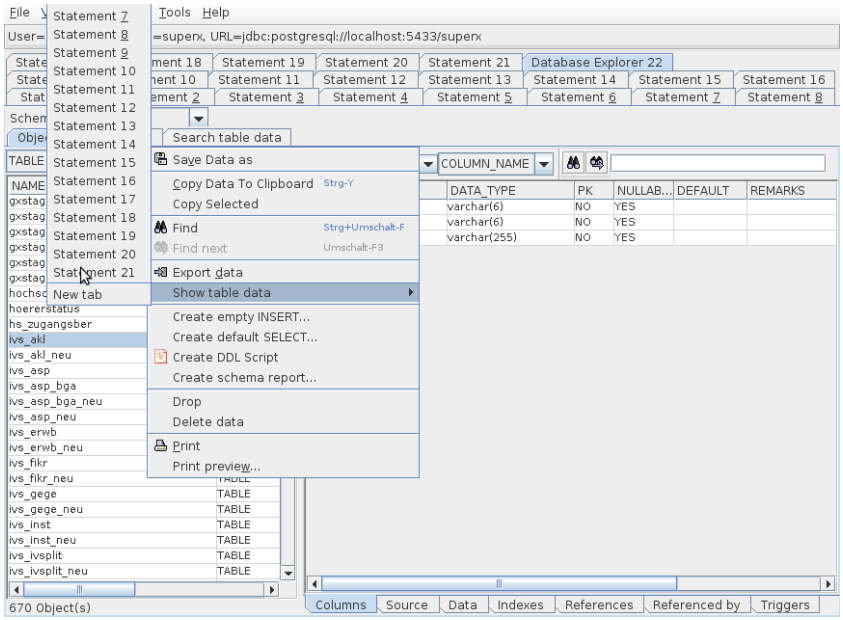


Im Dialog File -> Connect können Sie dann eine Datenquelle eintragen, und die Verbindungsparameter vervollständigen (Username, Passwort). Autocommit sollten Sie immer einschalten.

Interessant ist der Datenbank-Explorer (Tools -> Database Explorer), der es ermöglicht, die Datenbank nach Tabellen / Prozeduren etc. zu durchsuchen. Wenn eine Tabelle ausgewählt ist, kann sie auch über die Registerkarte "Data" editiert werden. Achten Sie darauf, dass Sie das Feld **Max. Rows** auf einen sinnvollen Wert setzen, z.B. 2000. Die SQLWorkbench ist gerade für die Arbeit mit Tabellen ohne Primärschlüssel geeignet, weil jede Äderung intern als Update formuliert wird. Der Nachteil ist, dass das Tool manchmal recht langsam ist, und dass nicht mehrere Zellen über Zwischenablage geändert / eingefügt werden können.

Sehr praktisch für die Entwicklung von SQL-Abfragen ist die Möglichkeit, zu jeder Tabelle eine select-String zu formulieren.

Markieren Sie die Tabelle im Database Explorer, und gehen Sie über das Kontextmenü auf Show table data, und wählen Sie ein Editorfenster aus. Der Select-String wird dann angezeigt.



Das Tool bietet außerdem eine Makrofunktion, und in neueren Versionen auch ETL-Funktionen über einen "Data Pumper", was es natürlich für SuperX besonders interessant macht. Weitere Tipps und Hilfen erhalten Sie im (gelungenen, aber englischen) Benutzerhandbuch.

2.8.7 Diagnose-Tool (jsp)

Um Rechteeinstellungen nach der Anmeldung im Browser für den aktuellen User zu prüfen kann man /superx/xml/diagnosetool.jsp aufrufen.

(Voraussetzung, dass auch FIN-Modul existiert, weil die entsprechenden Tabellen gelesen werden).

Unten sind auch Sichten dargestellt, die enthalten direkt nach der Anmeldung noch keine Einträge, weil sie erst bei Bedarf (Maskenaufruf mit der entsprechenden Sicht) geladen werden.

Bei Bedarf kann man diagnosetool.jsp?sichstand=true übergeben, dann werden auch die Sichten alle geladen – experimentelles Feature: Evtl. kann es danach bei Maskenaufruf zu Problemen kommen.

2.8.8 Entwicklungsservlet für SuperX-Abfragen

Neu ab Kernmodul3.5 ist ein spezielles Entwicklungsservlet, das die (verschiedenen Stadien der) Verarbeitung und Berichtserstellung in SuperX transparent macht.

Bisher wurde zu Entwicklungszwecken insbesondere das Applet oder die SQL-Angaben im SuperXManager genutzt, nun gibt es ein spezielles Entwicklungsservlet, das die Arbeit erleichtern soll.

2.8.8.1 Aufrufseite des Entwicklungsservlets

Es ist auf Ihrem Webserver unter der Adresse `http://RECHNER:PORT/superx/servlet/de.superx.servlet.-Entwicklung` erreichbar.

Wenn Sie eine kürze URL einrichten möchten, können Sie in der Datei

`webserver/tomcat/webapps/superx/WEB-INF/web.xml`

einen Eintrag hinzufügen:

```
<servlet>
        <servlet-name>
            Entwicklung
        </servlet-name>
        <servlet-class>
            de.superx.servlet.Entwicklung
        </servlet-class>
</servlet>
```

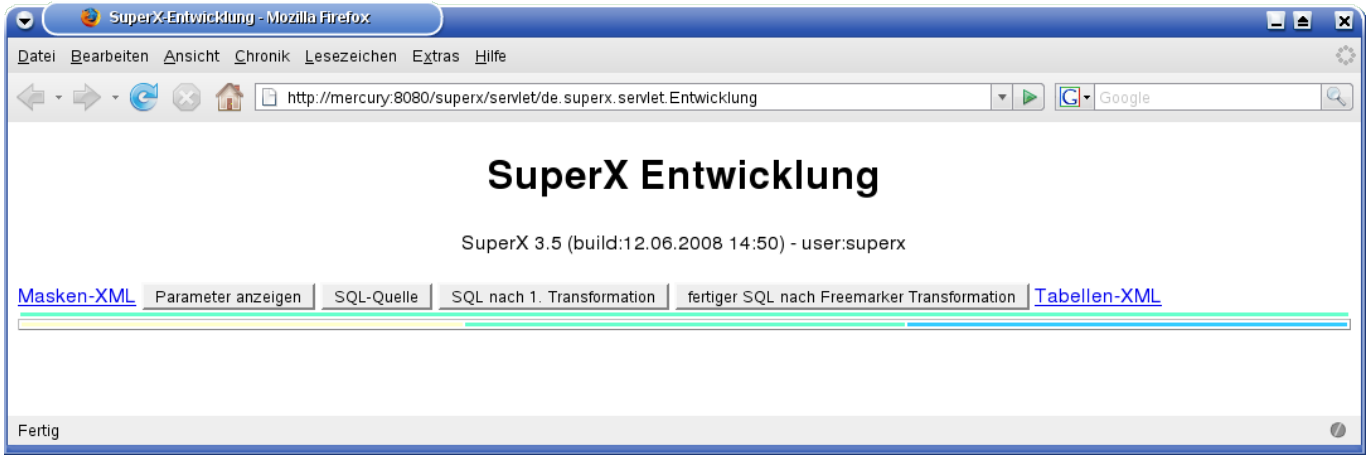
Dann können Sie nach einem Tomcatneustart, das Servlet auch unter der folgenden Adresse aufrufen:

`http://RECHNER:PORT/superx/servlet/Entwicklung`

2.8.8.2 Funktionalität des Entwicklungsservlets

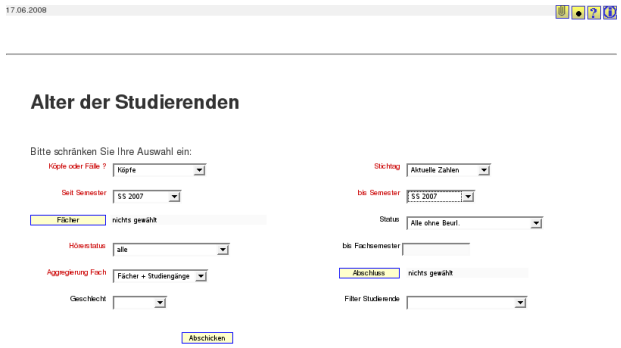
Mit dem Servlet kann man sich den Ausgangs-SQL, die Anpassungen durch die klassische SuperX-Transformation sowie den fertigen SQL nach FreeMarker-Transformation und auch die übergebenen Parameter ansehen. Außerdem kann man sich den Masken- und Tabellen-XML runterladen.

Hier eine Abbildung der Startseite:



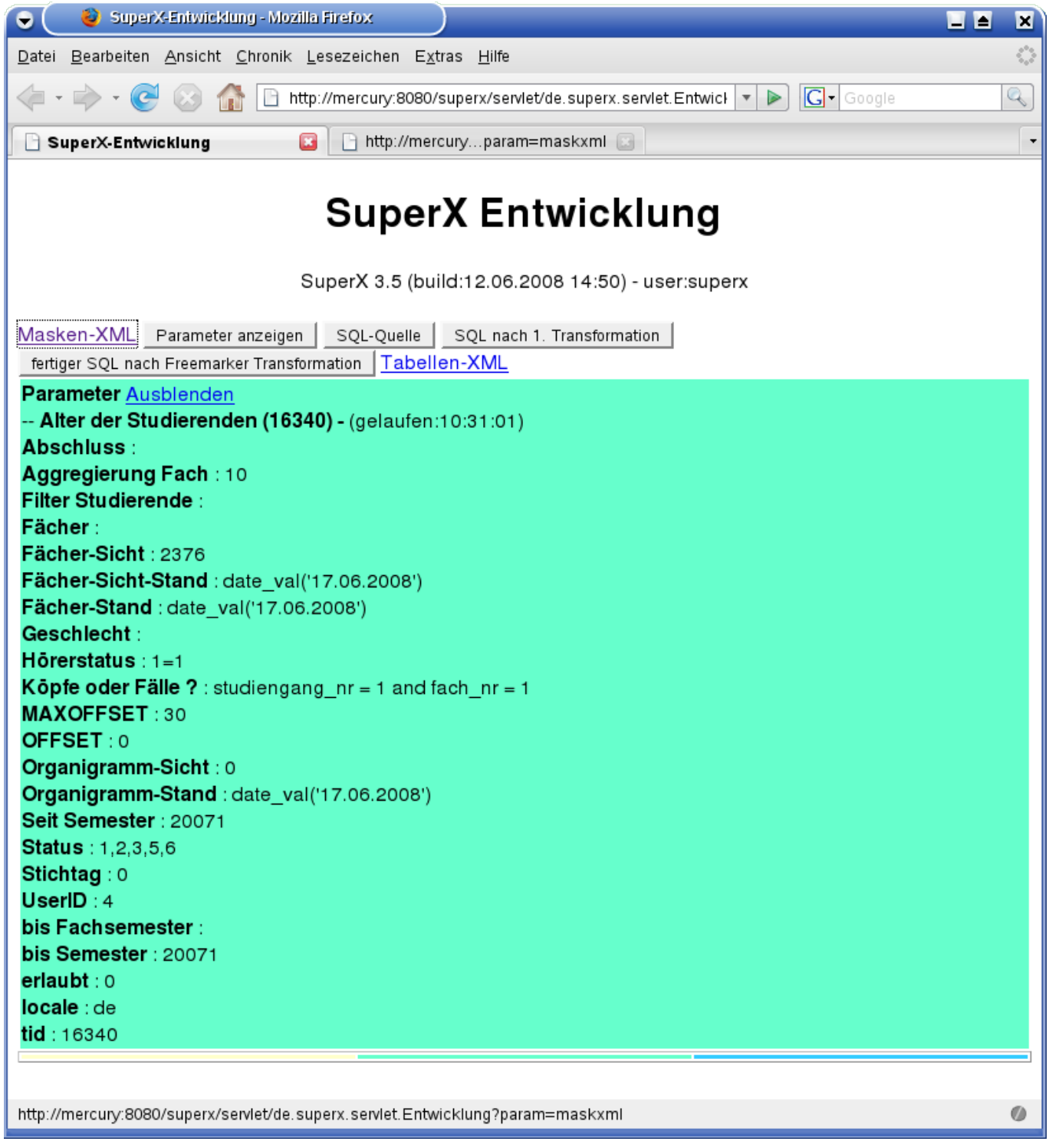
Wenn Sie eine Abfrage ausführen, können Sie zunächst mit dem Link "Masken-XML" den XML-Code der Maske aufrufen (klicken Sie jeweils auf die Grafik, um sie zu vergrößern):

Hier die Maske:

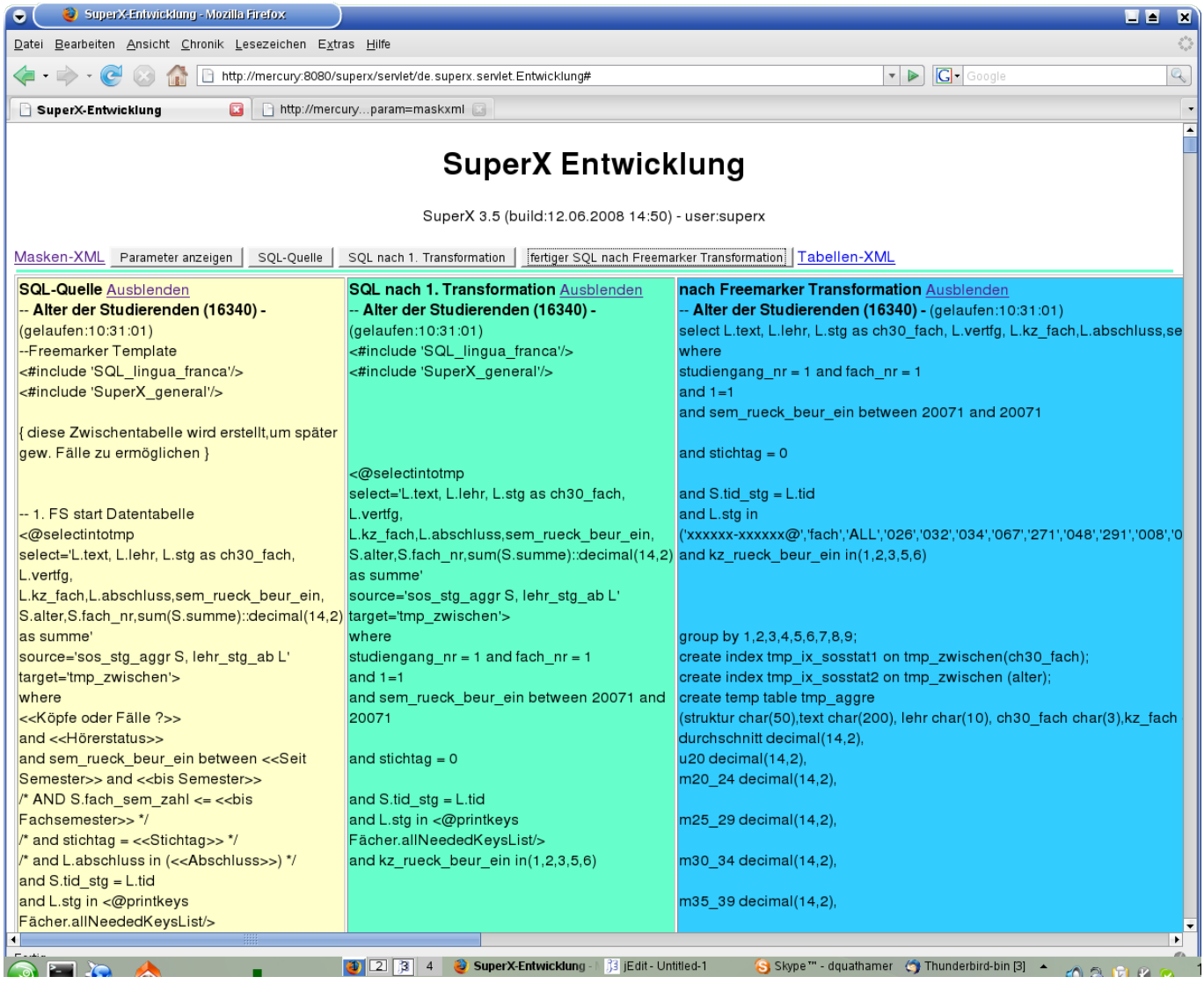


und hier der XML-Code:

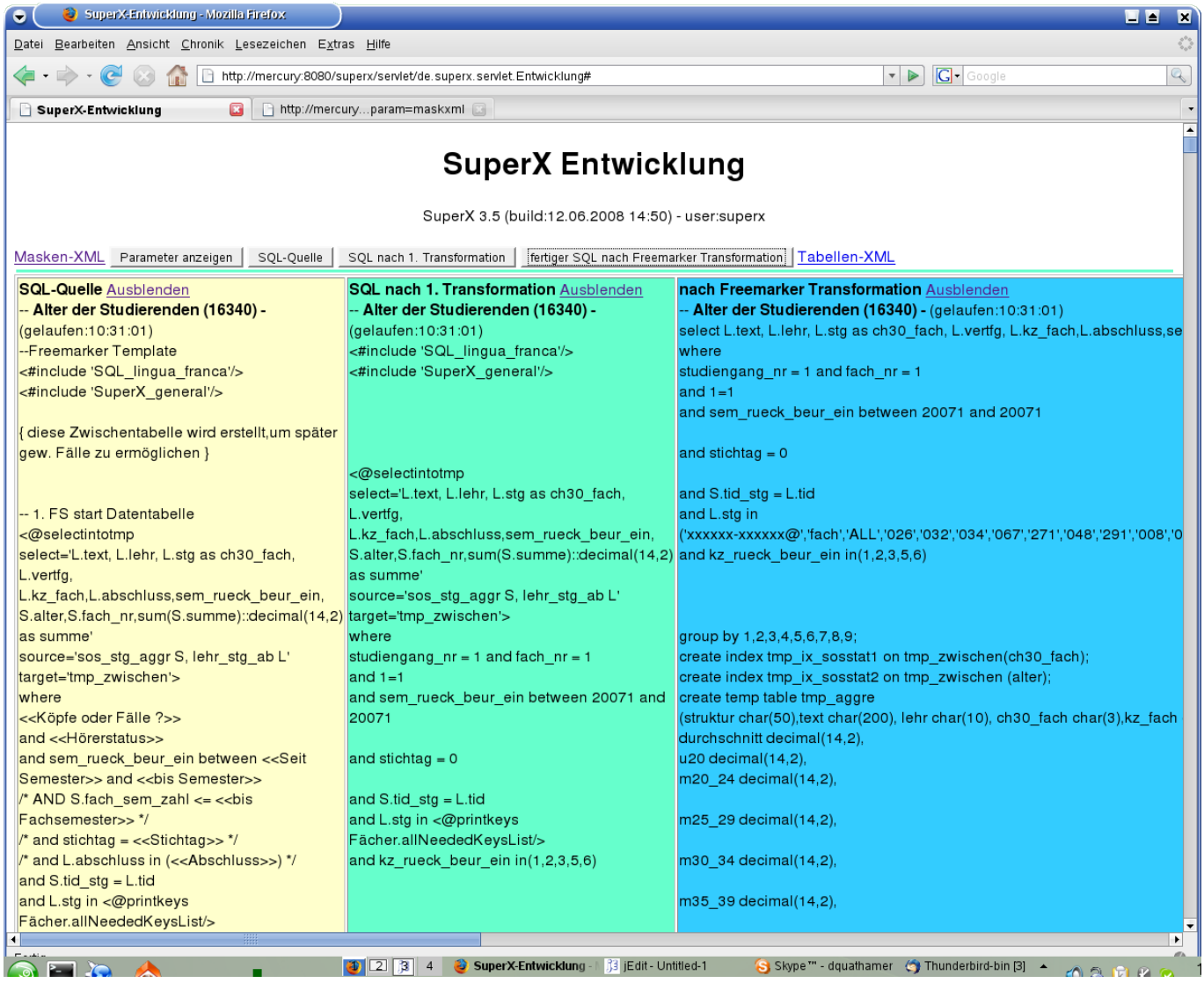
Im nächsten Schritt können die vom Benutzer eingegebenen Parameter sichtbar gemacht werden:


SuperX-Entwicklung - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://mercury:8080/superx/servlet/de.superx.servlet.Entwicl
Google
SuperX-Entwicklung
http://mercury...param=maskxml
SuperX Entwicklung
SuperX 3.5 (build:12.06.2008 14:50) - user:superx
Masken-XML
Parameter anzeigen
SQL-Quelle
SQL nach 1. Transformation
fertiger SQL nach Freemarker Transformation
Tabellen-XML
Parameter Ausblenden
-- Alter der Studierenden (16340) - (gelaufen:10:31:01)
Abschluss :
Aggregierung Fach : 10
Filter Studierende :
Fächer :
Fächer-Sicht : 2376
Fächer-Sicht-Stand : date_val('17.06.2008')
Fächer-Stand : date_val('17.06.2008')
Geschlecht :
Hörerstatus : 1=1
Köpfe oder Fälle ? : studiengang_nr = 1 and fach_nr = 1
MAXOFFSET : 30
OFFSET : 0
Organigramm-Sicht : 0
Organigramm-Stand : date_val('17.06.2008')
Seit Semester : 20071
Status : 1,2,3,5,6
Stichtag : 0
UserID : 4
bis Fachsemester :
bis Semester : 20071
erlaubt : 0
locale : de
tid : 16340
http://mercury:8080/superx/servlet/de.superx.servlet.Entwicklung?param=maskxml

Die Parameter werden dem nun folgenden SQL-Quellcode der Abfrage übergeben (SQL-Quelle), daraus wird dann ein Script erzeugt, das nur noch die Freemarker-Befehle enthält (SQL nach 1. Transformation). Das tatsächlich nach der Freemarker-Transformation in der Datenbank ausgeführte Script wird in der rechten Spalte angezeigt.

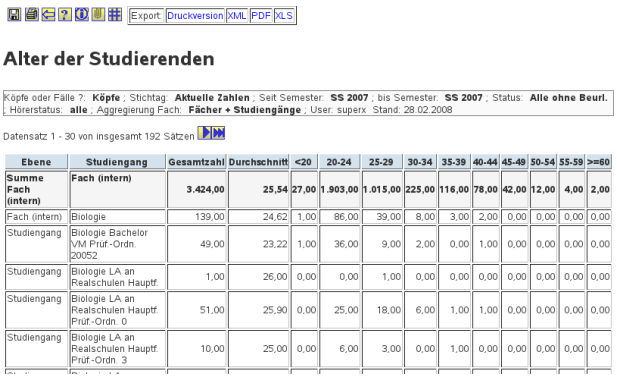


Schlussendlich können Sie noch den XML-Code der Ergebnistabelle anzeigen (Link Tabellen-XML):



Schlussendlich können Sie noch den XML-Code der Ergebnistabelle anzeigen (Link Tabellen-XML):

Hier die Tabelle:



Export: Druckversion XML PDF XLS

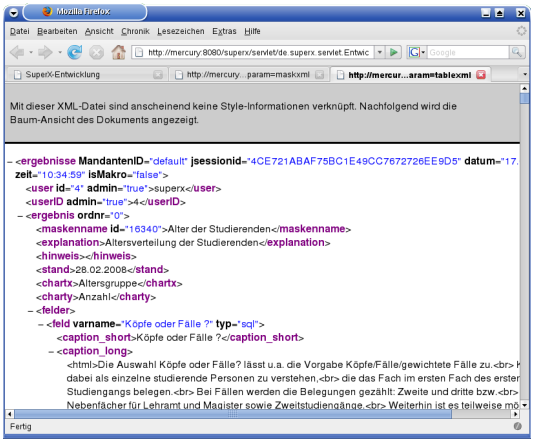
Alter der Studierenden

Köpfe oder Fälle ?: **Köpfe** ; Stichtag: **Aktuelle Zahlen** ; Seit Semester: **SS 2007** ; bis Semester: **SS 2007** ; Status: **Alle ohne Beurl.** ; Hörerstatus: **alle** ; Aggregierung Fach: **Fächer + Studiengänge** ; User: superx Stand: 28.02.2008

Datensatz 1 - 30 von insgesamt 192 Sätzen

Ebene	Studiengang	Gesamtzahl	Durchschnitt	<20	20-24	25-29	30-34	35-39	40-44	45-49	50-54	55-59	>=60
Summe Fach (intern)	**Fach (intern)**	**3.424,00**	**25,54**	**27,00**	**1.903,00**	**1.015,00**	**225,00**	**116,00**	**78,00**	**42,00**	**12,00**	**4,00**	**2,00**
Fach (intern)	Biologie	139,00	24,62	1,00	86,00	39,00	8,00	3,00	2,00	0,00	0,00	0,00	0,00
Studiengang	Biologie Bachelor VM Prüf.-Ordn 20052	49,00	23,22	1,00	36,00	9,00	2,00	0,00	1,00	0,00	0,00	0,00	0,00
Studiengang	Biologie LA an Realschulen Hauptf.	1,00	26,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Studiengang	Biologie LA an Realschulen Hauptf. Prüf.-Ordn. 0	51,00	25,90	0,00	25,00	18,00	6,00	1,00	1,00	0,00	0,00	0,00	0,00
Studiengang	Biologie LA an Realschulen Hauptf. Prüf.-Ordn. 3	10,00	25,00	0,00	6,00	3,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00

und hier der XML-Code:



Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

http://mercury:8080/superx/servlet/de.superx.servlet.Entwic

SuperX-Entwicklung | http://mercury...param=maskxml | http://mercur...aram=tablexml

Mit dieser XML-Datei sind anscheinend keine Style-Informationen verknüpft. Nachfolgend wird die Baum-Ansicht des Dokuments angezeigt.

```
– <ergebnisse MandantenID="default" jsessionid="4CE721ABAF75BC1E49CC7672726EE9D5" datum="17.
zeit="10:34:59" isMakro="false">
  <user id="4" admin="true">superx</user>
  <userID admin="true">4</userID>
  – <ergebnis ordnr="0">
    <maskenname id="16340">Alter der Studierenden</maskenname>
    <explanation>Altersverteilung der Studierenden</explanation>
    <hinweis></hinweis>
    <stand>28.02.2008</stand>
    <chartx>Altersgruppe</chartx>
    <charty>Anzahl</charty>
    – <felder>
      – <feld varname="Köpfe oder Fälle ?" typ="sql">
        <caption_short>Köpfe oder Fälle ?</caption_short>
        – <caption_long>
          <html>Die Auswahl Köpfe oder Fälle? lässt u.a. die Vorgabe Köpfe/Fälle/gewichtete Fälle zu.<br> H
          dabei als einzelne studierende Personen zu verstehen,<br> die das Fach im ersten Fach des erster
          Studiengangs belegen.<br> Bei Fällen werden die Belegungen gezählt: Zweite und dritte bzw.<br>
          Nebenfächer für Lehramt und Magister sowie Zweitstudiengänge.<br> Weiterhin ist es teilweise mö
```

Fertig

2.8.8.3 Berechtigung für das Entwicklungsservlet

Standard ist, dass das Servlet aus Sicherheitsgründen nur für Adminstratoren zugänglich ist, wenn Sie es auch als eingeschränkter User nutzen möchten, müssen Sie es für diesen User in der `tomcat/webapps/superx/WEB-INF/web.xml` freigeben. Ergänzen Sie einen `<init-param>`-Block zur obig aufgeführten <servlet>-Definition, Inhalt ist eine mit Komma getrennte Liste der Userkennungen, die Zugriff erhalten sollen,z.B.

```
<servlet>
        <servlet-name>
            Entwicklung
        </servlet-name>
        <servlet-class>
            de.superx.servlet.Entwicklung
        </servlet-class>
<init-param>
<param-name>authorized_users</param-name>
<param-value>test,test2</param-value>
</init-param>
</servlet>
```

Hinweis: Wenn Sie einen Eintrag wie oben machen, gilt der für die aufrufende URL `http://RECHNER:PORT/superx/servlet/Entwicklung`.

Nicht jedoch für die Langversion http://RECHNER:PORT/superx/servlet/de.superx.servlet.Entwicklung- ggfs. also zwei <servlet>-Einträge für die Servletnamen Entwicklung und de.superx.servlet.Entwicklung anlegen.

2.8.9 Masken für das XML-Frontend vorbereiten

Das XML-Frontend arbeitet mit den vorhandenen Masken und stellt dort grundlegende Funktionen zur Verfügung. Darüber hinaus bietet das Frontend die Möglichkeit, einzelne Abfragen individuell zu gestalten. Hierzu sind allerdings grundlegende XML-Kenntnisse erforderlich.

Ein großer Vorteil des XML-Frontends ist, dass Anwender sich ihre Bericht im XML-Format herunterladen können und ohne Datenbankkenntnisse ihre Berichte "maßschneidern" können.

Es ist z.B. damit möglich, auf beliebte Berichte mit gesetzten Parametern einen Bookmark zu legen.

2.8.9.1 Erzeugen eines Stylesheets

Es ist möglich für Spezialfunktionen eigene Stylesheets für einzelne Masken zu hinterlegen.

Zunächst muss für das Ergebnis ein neues Stylesheet erzeugt werden. Als Vorlage für Masken können Sie das Muster-Stylesheet

`$SUPERX_DIR/webserver/tomcat/webapps/superx/xml/maske_html.xsl`

Für Ergebnistabellen können Sie das Muster-Stylesheet

`$SUPERX_DIR/webserver/tomcat/webapps/superx/xml/tabelle_html.xsl`

verwenden. Speichern Sie das Stylesheet unter einem anderen Namen im gleichen Verzeichnis ab, und ändern Sie das Stylesheet. Dann fügen Sie das Stylesheet in die Tabelle `sx_stylesheets` ein.

sx_stylesheets : Tabelle

	tid	filename	caption	description	relation	userag	contenttype
▶	1	tabelle_html.xsl	Generisches St	Generisch	table		text/html; charset=ISO-8859-1
	2	tabellenfeld_bearbeiten.xsl	Generisches St	Generisch	table		text/html; charset=ISO-8859-1
	3	maske_html.xsl	Generisches St	Generisch	mask		text/html; charset=ISO-8859-1
	4	maske_html_tabfeld.xsl	Generisches St	Generisch	mask		text/html; charset=ISO-8859-1
	5	tabelle_html_11570.xsl	Berichtsblatt K	Kurze Zusamm	table		text/html; charset=ISO-8859-1
	7	tabelle_fo_rtf.xsl	RTF	Export in Textv	table		application/msword
	6	tabelle_fo_pdf.xsl	PDF	Export in PDF (	table		application/pdf
*							

Datensatz: 1 von 7

Das Beispiel zeigt einige Styleheets, das erste ist bereits Teil des Kernmoduls, das fünfte befindet sich im COB-Modul. Zu den Feldern:

•**filename** kennzeichnet den Dateinamen relativ zum Verzeichnis `$SUPERX_DIR/webserver/tomcat/webapps/superx/xml`.

•**caption** dient als Kurzüberschrift, die im Ergebnisblatt als Button angezeigt wird.

•**description** stellt einen Erläuterungstext für den Button dar.

•**relation** bezieht sich auf die Beziehung des Stylesheets; mögliche Werte sind "mask" für eine Maske und "table" für Tabelle.

•**useragent** beitet die Möglichkeit, ein Stylesheet für spezielle Lesegeräte anzubieten, z.B. WAP-Handys oder Braille-Zeilen.

•**contenttype** entspricht dem useragent und kennzeichnet den `content-type`, der dem Lesegerät im http-header übermittelt werden soll. Möglich sind derzeit die obigen Varianten (svg oder excel sind in Vorbereitung).

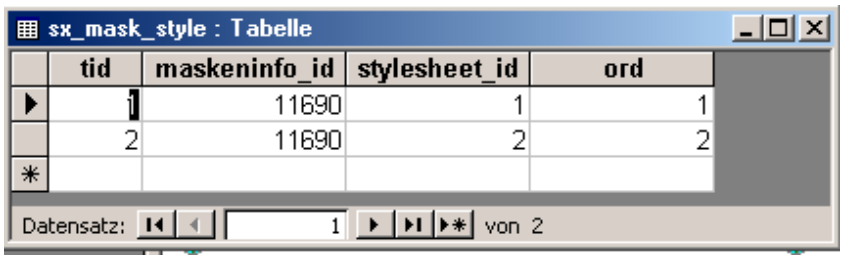
Bitte bei Upgrade auf SuperX 4.1 und Treetable-Tabellen Abschnitt unter 5.1 beachten.

2.8.9.2 Zuordnung einer Maske zu einem Stylesheet

Konkret arbeitet SuperX so: Wenn einer Abfrage ein oder mehrere Stylesheets zugeordnet sind, dann werden die in der Reihenfolge angezeigt, in der sie definiert sind. Wenn kein Stylesheet definiert ist, dann wird das Standard-Stylesheet von SuperX benutzt: `maske_html.xsl` für Masken sowie `tabelle_html.xsl` für Tabellen.

Die Zuordnung eines Stylesheets geschieht in der Tabelle `sx_mask_style`. Der Tupelidentifier des Stylesheets wird in der Tabelle `sx_mask_style` im Feld `stylesheet_id` eingetragen.

Das Beispiel zeigt, dass die beiden oben beschriebenen Stylesheets der Maske 11690 zugeordnet werden.



Das Feld `ord` kennzeichnet die Reihenfolge der anzubietenden Stylesheets. Wir sehen hier, dass zuerst das generische Standard-Stylesheet angezeigt wird, und dann das Stylesheet Nr.2.

Defaultmäßig sind die Stylesheets für html (Druckversion in neuem Fenster), xml, Excel und PDF in jeder Ergebnistabelle enthalten. Andere Stylesheets pfg müssen in der obigen Tabelle zugeordnet werden - dies ist sinnvoll, da die Standard-Stylesheets zunächst mit der in Frage kommenden Maske erprobt werden muss.

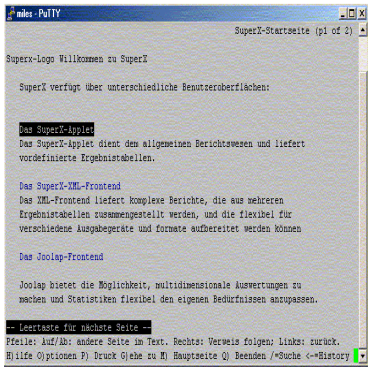
2.8.9.3 Anpassung an Lesegeräte

Der Vorteil von XML-Berichten ist, dass sie sich an individuelle Lesegeräte anpassen lassen. So können Sie die Standardoberfläche automatisch für das jeweilige Lesegerät anpassen und dadurch ganz individuelle Designs erzielen, z.B. auch für barrierefreie Angebote.

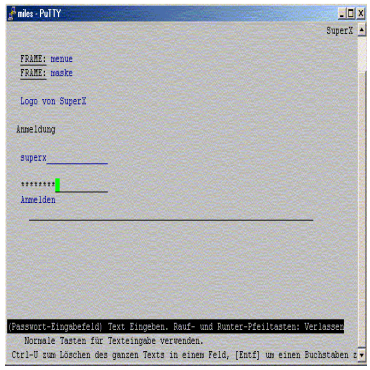
Das folgende Beispiel zeigt dies anhand des textbasierten HTML-Browsers **lynx**, der sich (zumindest am Anfang) gut zum Testen für barrierefreie Angebote eignet.

Klicken Sie jeweils auf die Grafik, um sie zu vergrößern.

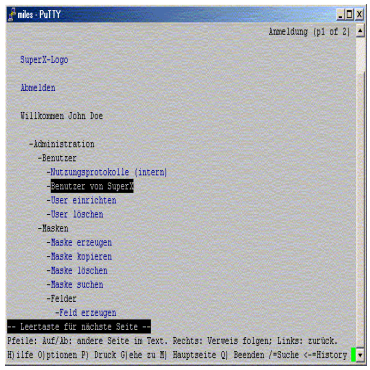
Die rechte Abbildung zeigt die SuperX-Homepage in einer Konsole im Browser `lynx`.



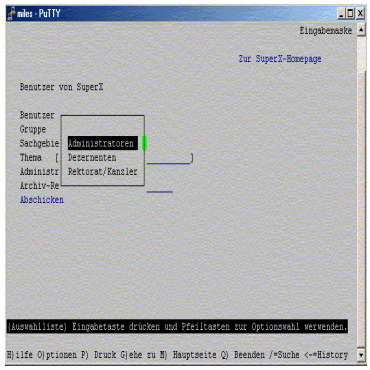
Wir gehen auf das XML-Frontend, und erhalten die Anmeldemaske. Die Frame-Tags ignorieren wir.



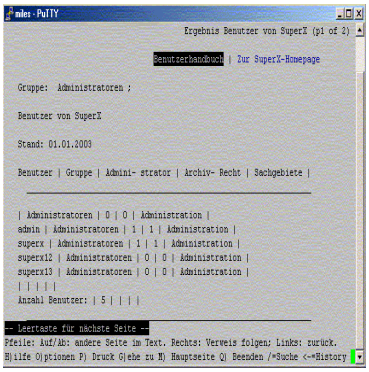
Nach erfolgreicher Anmeldung erscheint das Menü aus dem Themenbaum. Wir wählen hier als Beispiel die Abfrage **Benutzer von SuperX**.



Nun wird die Maske von dieser Abfrage angezeigt. Bei Kombinationsfeldern gehen wir auf das Feld, und drücken die Return-Taste. Es erscheinen die Auswahleinträge. Zum Abschluss gehen wir auf "Abschicken".



Es erscheint die Ergebnisanzeige. Dies sieht natürlich noch nicht besonders gut aus, weil textbasierte Browser und Tabellen sich nicht gut vertragen. Via Stylesheet lassen sich aber ganz übersichtlich Darstellungen entwerfen.



Das Beispiel zeigt, dass durch XML und XSL keine Grenzen bei der Gestaltung von Benutzeroberflächen für SuperX existieren. Die obigen Stylesheets befinden sich als Muster im Verzeichnis `$SUPERX_DIR/webserver/tomcat/webapps/superx/xml`, und haben jeweils den Zusatz "html2" (für einfaches HTML Version 2.0) im Dateinamen, z.B. `maske_html2.xsl`.

Wein kleiner Tipp noch für lynx: Wenn Sie das produzierte html überprüfen wollen, dann starten Sie lynx wie folgt:

```
lynx –trace http://localhost:8080/superX/xml/
```

Eine Logdatei `lynx.trace` wird in das aktuelle Verzeichnis geschrieben.

2.8.9.4 Eigene XSL-Stylesheets für Masken oder Tabellen erstellen

Mit eigenen XSL-Stylesheets kann man das Aussehen von Masken oder Ergebnistabellen sehr individuell anpassen.

Klassisch war das Vorgehen, dass man eine Kopie von maske_html/pdf/xls.xsl bzw tabelle_html/pdf/xls.xsl machte und darin Änderungen vornahm.

Da inzwischen immer wieder Erweiterungen an den Standardstylesheets vorgenommen werden, ist es nur blöd, dass diese Erweiterungen dann nicht in die kopierten Spezialstylesheets kommen.

Daher sind die Stylesheets inzwischen etwas „objektorientierter" und kleine Änderungen kann auch per page_compontents_final.xsl machen.

Beispiel aus der Praxis, bei 2,3 Masken sollte unter dem Maskennamen noch ein Link zu PDF-Dateien erscheinen.

In der Standard maske_html.xsl ist nach dem Titel eine template-Aufruf definiert.

```
<p class="maskentitel"><xsl:value-of select="maske/@name" /></p>
<xsl:call-template name="pccustomize"> <xsl:with-param name="position" se-
lect="'mask_post_title'"></xsl:with-param> </xsl:call-template>
```

Das template ist leer in Page-Compnents.xsl definiert. Man kann es in pageCompontents_final.xsl definieren, dann wird es überschrieben.

```
<xsl:template name="pccustomize">
 <xsl:param name="position"/>
 <xsl:if test="$position='mask_post_title'">
   <!-- hier kommt der individuelle Inhalt rein -->
  </xsl:if>
</xsl:template>
```

Die Links werden nur bei den entsprechenden Masken eingebaut.

Außerdem wird standardmäßig leere maskonload überschrieben um den div der folgenden maskenfelder etwas tiefer zu setzen, damit Platz für den größeren Titel ist (sonst war Link nicht anklickbar).

Um bei bestimmten Masken export Buttons auszublenden, kopieren Sie das entsprechende Template (hier: exportButtons) aus der Page-Compnents.xsl in die pageCompontents_final.xsl und fügen dort eine if-Bedingung ein.

Beispiel für entfernen des PDF export Buttons für die Masken mit der tid 16000 und 17000:

```
 <xsl:template name="exportButtons" >
...
<xsl:if test="/ergebnisse/ergebnis/maskenname/@id!='16000' and
/ergebnisse/ergebbnis/maskenname/@id!='17000'">
 <!-- hier kommt der PDF-Button -->
</xsl:if>
...
</xsl:template>
```

Will man eine spezielles Tabellenstylesheet erzeugen, braucht man tabelle_html.xsl nicht mehr kopieren, sondern erzeugt eine xsl Datei mit den Standard imports und fügt dort den import für tabelle_html.xsl hinzu.

```
<xsl:import href="xsl_functions.xsl" />
<xsl:import href="resultset_html.xsl" />
<xsl:import href="interLinks_html.xsl" />
<xsl:import href="pageComponents_html.xsl" />
<xsl:import href="tabelle_html.xsl" />
<xsl:import href="pageComponents_html_final.xsl" />
```

als weiteres braucht man nur das Template von tabelle_html.xsl zu überlagern, was geändert werden soll.

Einfachstes Beispiel – keine Erläuterungslinks anzeigen,

Template explanation wird überlagert

```
<xsl:template name="explanation"/>
```

Bei Bedarf kann man auch die standardmäßig leeren Funktionen wie

```
<xsl:call-template name="tablecustomize"><xsl:with-param name="position" se-
lect="'table_post_center'"></xsl:with-param></xsl:call-template>
```

überlagern.

Am Ende folgt dann, um die Tabelle aufzubauen.

```
<xsl:template match="/">
<xsl:call-template name="table"/>
</xsl:template>
```

it's fricking complex, but enjoy!

2.8.9.5Eigene XSL-Stylesheets für Mandanten

In Mandantensystemen wird trotz des Mandantenverzeichnisses die pageComponents_html_final.xsl aus dem superx/xml Verzeichnis genommen. Auch die css Dateien werden aus dem Allgemeinen Verzeichnis verwendet. Um Individuelle Anpassungen vornehmen zu können gibt es die Möglichkeit mit Bedingung auf die MandantenID in der pageComponents_html_final.xsl Veränderungen vorzunehmen. Die Bedingung auf die MandantenID ist hier notwendig, damit andere Hochschulen nicht von der Änderung betroffen werden. Hier ist also Vorsicht geboten!

Die pageComponents_html_final.xsl sollte auch nicht ersetzt werden sondern erweitert werden, falls andere Hochschulen schon Individuelle Anpassungen vorgenommen haben.

Hier ein Beispiel:

```
<xsl:choose>
<xsl:when test="/maske/@MandantenID='MANDANTENID'">
<!-- Hier der Code für den Mandant MANDANTENID-->
</xsl:when>
<xsl:otherwise>
<!-- Hier der Code für alle anderen -->
</xsl:otherwise>
</xsl:choose>
```

Falls Änderungen im Themenbaum stattfinden soll, dann muss folgender Code in die test Bedingung:

```
/menue/mandantenid='FHHN'
```

Falls z.B. der Link zum Applet oder Passwort ändern raus soll.

2.8.9.6 Besonderes XML zu ALLEN Masken hinzufügen

Für Management-Modul gibt es ganz besondere Erweiterung.

Anwendungsfall, ein Navigationsmenü soll in allen Masken bereitgestellt werden, es soll aber auch möglich sein, ganz normale SuperX-Masken einzubinden.

also Aufruf z.B. http://localhost:8080/superx/servlet/SuperXmlTabelle?tid=xxy&param=1&stylesheet=tabelle_html_man.xsl

Legen Sie dazu in Repository ein Feld ID=CUSTOMXMLADD an. Da kann fester XML-Drin stehen, der wird zu allen Tabellen-xmls hinzugefügt.

zB..

```
<navigation>
<item1 .../>
<item2 ../>
</navigation>
```

der wird dann im Tabellen XML unter ergebnisse/ergebniselement hinzufügt und kann ausgewertet werden.

z.B:

```
<div id="Navigation">
<xsl:for-each select="/ergebnisse/ergebnis/ergebniselement/navigation/item">
<p class="ebene0">
<a class="ebene1" target="_self">
<xsl:attribute name="href"><xsl:value-of select="concat(HtmlUtils:encodeURL('SuperXmlTabelle',/ergebnisse/@jsessionid ),'?tid=',href)" /></xsl:attribute>
<![CDATA[]]><xsl:value-of select="caption" /><![CDATA[]]>
            </a>
            </p>
</xsl:for-each>
```

Richtig cool wird es aber noch, wenn der hinzuzufügende XML dynamisch mit Freemarker generiert wird, z.B.:

```
<xupdate>
 <text table="sx_repository" field="content" where="id='CUSTOMXMLADD'">
```

```
   <![CDATA[<#include "MAN_MAKROS"/><sqlvars> <sqlvar name="entries"
type="hash">select R.catalogue_id,K.shortname,sqlchunk,calcratio,decimalpla-
ces,linksub,linktimeline from man_catalogue K,man_catalogue_rpt R where
K.id=R.catalogue_id and R.active=1 order by R.sortnr, R.sortnr2 </sqlvar>
</sqlvars>  <#if !Semester?exists><#assign Semester=""></#if> <#if !Bezugs-
semester?exists><#assign Bezugssemester=""/></#if> <#if !Jahr?exists><#as-
sign Jahr=""></#if> <#if !faecherkeys?exists><#assign faecherkeys=""></#if>
<#if !Fächer?exists><#assign Fächer={"selectedKey":""}></#if>  <#if !Kosten-
stelle?exists><#assign Kostenstelle={"selectedKey":""}></#if>  <#if !
(.vars["Datum (Personal)"]?exists)><#assign "Datum (Personal)"=""></#if>
<#if !(.vars["Beschäftigungsstelle (Person)"]?exists)><#assign "Beschäfti-
gungsstelle
(Person)"=""></#if><navigation><item><href>CDATASTART888880190&Jahr=$
{Jahr}&Semester=${Semester}&Datum%20(Personal)=${.vars["Datum
(Personal)"]}CDATAEND</href><caption>Start</caption></item><#if entries?
is_sequence><#foreach e in entries> <#assign linktimeline=e.linktimeline?in-
terpret/> <#assign linkt><@linktimeline/></#assign> <#assign linksub=e.-
linksub?interpret/> <#assign links><@linksub/></#assign>  <doubleitem id="$
{e.id}"> <aktuell>CDATASTART${links?replace('Fakultäten|','')}CDATAEND</ak-
tuell> <zeitreihe>CDATASTART${linkt?
replace('Zeitreihe|','')}CDATAEND</zeitreihe> <caption>CDATASTART${e.short-
name}CDATAEND</caption> </doubleitem> </#foreach> </#if></navigation>]]>
  </text>

</xupdate>
```

2.8.9.7 Erweiterungen des XML-Frontends

Das XML-Frontend bietet gegenüber dem Applet einige Erweiterungen, die insbesondere für aufwändiger gestaltete Webapplikationen nützlich sind:

- Die Ergebnissseiten werden nicht komplett geladen, sondern im Rahmen von frei definierbaren Intervallen, z.B. 30 Datensätze pro Seite. Am Seitenende wird dann eine Navigationsmöglichkeit gelifert (Vorherige Seite / Nächste Seite). Der Intervall wird in `$SUPERX_DIR/webserver/tomcat/webapps/superx/WEB-INF/web.xml` definiert (Parameter `maxOffset`).
- Die Ergebnisseiten können verlinkt werden, über spezielle Navigationsspalten (s.u.).
- In Feldern können Links zu anderen Masken definiert werden (Feldart 15).

2.8.9.7.1 Navigationsspalten im XML-Frontend

Wenn die Ergebnistabelle an das XML-Frontend übergeben wird, dann können spezielle Spalten für die Navigation eingesetzt werden. Die Spaltennamen werden im letzten `select` des `select_stmt` einer Makse übergeben.

nexttable Link auf eine andere SuperX-Tabelle; der Inhalt des Feldes wird dann um den Pfad zum Servlet, (optional auch den String der Sessionid) und den Passus "SuperXmlTabelle?tid=" ergänzt, d.h. dem Servlet wird als erster Parameter die `maskeninfo-tid` übergeben. So wird z.B. aus dem Inhalt:

```
20010&id=2044
```

der Link

```
http://<URL der Webapplikation>>/servlet/SuperXmlTabelle?tid=20010&id=2044
```

Die Ergebnisseite wird dann um einen Button  ergänzt.

nextpage Link auf eine andere SuperX-Tabelle wie **nexttable**, es wird nur ein anderes Icon und ein anderer Target genutzt.

nextmask Link auf eine andere SuperX-Maske; der Inhalt des Feldes wird dann um den Pfad zum Servlet, (optional auch den String der Sessionid) und den Passus "SuperXmlMaske?tid=" ergänzt. So wird z.B. aus dem Inhalt:

```
20010&id=2044
```

der Link

```
http://<URL der Webapplikation>>/servlet/SuperXmlMaske?tid=20010&id=2044
```

Die Ergebnisseite wird dann um einen Button  ergänzt.

nextdelete Link auf eine andere SuperX-Maske; Im Unterschied zu nextmask wird hier ein anderes Icon gewählt: Die Ergebnisseite wird dann um einen Delete-Button  ergänzt.

nextedit Link auf ein DBForms-Formular relativ zur URL des Servlets. die Ergebnisseite wird um einen "Bearbeiten"-Button  ergänzt.

nextmail Feldinhalte werden um einen Mailto-Tag ergänzt. z.B.

info@superx-projekt.de

wird zu

```
<a mailto:" info@superx-projekt.de"> info@superx-projekt.de</a>
```

url Feldinhalte werden um einen href-Tag (sowie wenn nötig um ein "http" ergänzt. z.B.

```
www.superx-projekt.de
```

wird zu

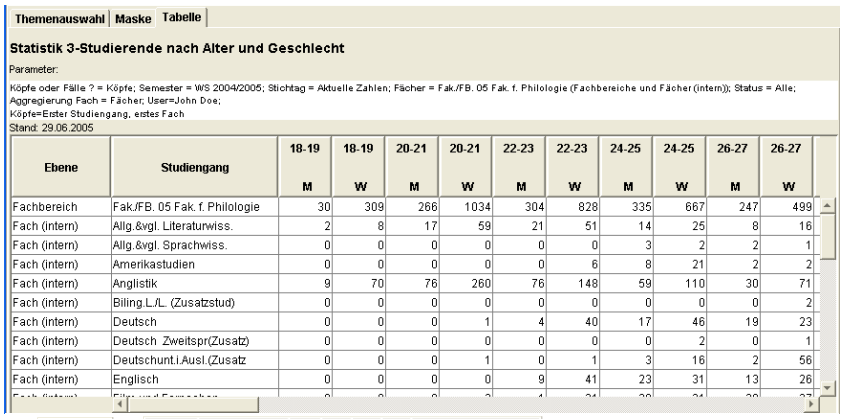
```
<a href="http://www.superx-projekt.de">www@superx-projekt.de</a>
```

nextlink Link auf eine externe Seite oder eine andere SuperX-Tabelle; anders als bei nexttable wird ein frei wählbarer textueller Link angegeben, wobei der Volltext des Links und der eigentliche Linkt durch ein Trennzeichen "|" getrennt sind.
So wird z.B. der Feldwert "Erläuterungen|http://www.erlaeuterungen.de" wie folgt ersetzt:
Erläuterungen
Wenn nach dem Trennzeichen keine externe Web-Adresse angeboten wird (erkennbar am vorangestellten "http:"), dann wird der Inhalt des Feldes um den Pfad zum Tabellen-Servlet ergänzt: So wird z.B. aus dem Inhalt:
`Details zur Hochschule|20010&id=2044`
der Link
`<a href=../servlet/SuperXmlTabelle?tid=20010&id=2044>Details zur Hochschule</a>`

2.8.9.7.2 Hierarchieebenen in Ergebnisspalten

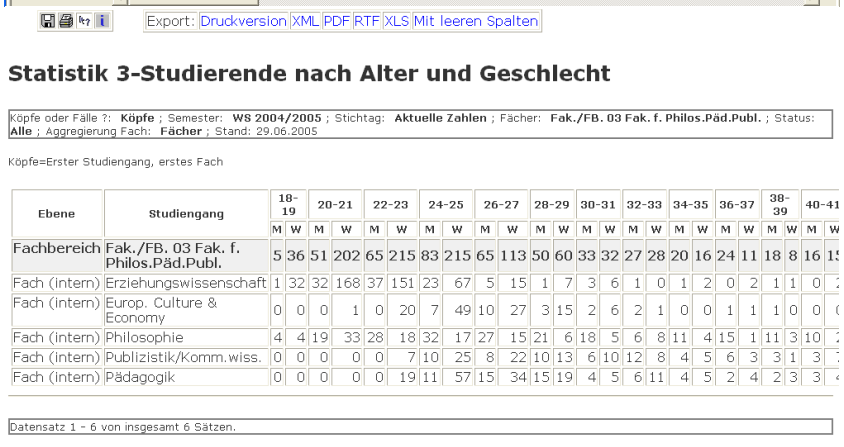
In Ergebnistabellen wird oft gewünscht, Tabellenüberschriften ineiner zu verschacheln. So wird z.B. aus folgender Tabelle:

Im Applet…



Ebene	Studiengang	18-19 M	18-19 W	20-21 M	20-21 W	22-23 M	22-23 W	24-25 M	24-25 W	26-27 M	26-27 W
Fachbereich	Fak./FB. 05 Fak. f. Philologie	30	309	266	1034	304	828	335	667	247	499
Fach (intern)	Allg.&vgl. Literaturwiss.	2	8	17	59	21	51	14	25	8	16
Fach (intern)	Allg.&vgl. Sprachwiss.	0	0	0	0	0	0	3	2	2	1
Fach (intern)	Amerikastudien	0	0	0	0	0	6	8	21	2	2
Fach (intern)	Anglistik	9	70	76	260	76	148	59	110	30	71
Fach (intern)	Biling.L./L. (Zusatzstud)	0	0	0	0	0	0	0	0	0	2
Fach (intern)	Deutsch	0	0	0	1	4	40	17	46	19	23
Fach (intern)	Deutsch Zweitspr(Zusatz)	0	0	0	0	0	0	0	2	0	1
Fach (intern)	Deutschunt.i.Ausl.(Zusatz	0	0	0	1	0	1	3	16	2	56
Fach (intern)	Englisch	0	0	0	0	9	41	23	31	13	26

Im XML-Frontend…



Ebene	Studiengang	18-19 M	18-19 W	20-21 M	20-21 W	22-23 M	22-23 W	24-25 M	24-25 W	26-27 M	26-27 W	28-29 M	28-29 W	30-31 M	30-31 W	32-33 M	32-33 W	34-35 M	34-35 W	36-37 M	36-37 W	38-39 M	38-39 W	40-41 M	40-41 W
Fachbereich	Fak./FB. 03 Fak. f. Philos.Päd.Publ.	5	36	51	202	65	215	83	215	65	113	50	60	33	32	27	28	20	16	24	11	18	8	16	15
Fach (intern)	Erziehungswissenschaft	1	32	32	168	37	151	23	67	5	15	1	7	3	6	1	0	1	2	0	2	1	1	0	
Fach (intern)	Europ. Culture & Economy	0	0	0	1	0	20	7	49	10	27	3	15	2	6	2	1	0	0	1	1	1	0	0	
Fach (intern)	Philosophie	4	4	19	33	28	18	32	17	27	15	21	6	18	5	6	8	11	4	15	1	11	3	10	
Fach (intern)	Publizistik/Komm.wiss.	0	0	0	0	0	7	10	25	8	22	10	13	6	10	12	8	4	5	6	3	3	1	3	
Fach (intern)	Pädagogik	0	0	0	0	0	19	11	57	15	34	15	19	4	5	6	11	4	5	2	4	2	3	3	

Die Spalten werde also verknüpft. Wie geht das?

Versehen Sie in der XIL_PROPLIST die Spaltenüberschrift mit einem Steuerzeichen "\000², also z.B.

Das Steuerzeichen "\000" zur Verknüpfung von Spaltenüberschriften kommt direkt nach dem "gemeinsamen" Teil der Überschrift

```
Column CID=1 heading_text="Studiengang" center_heading
   row_selectable heading_platform readonly
   width=40 text_size=60
Column CID=2 heading_text="18-19 \000 \n  M"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=8
Column CID=3 heading_text="18-19 \000 \n  W"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=8
Column CID=4 heading_text="20-21 \000 \n  M"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=8
Column CID=5 heading_text="20-21 \000 \n  W"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=8
Column CID=6 heading_text="22-23 \000 \n  M"   center_heading
   row_selectable col_selectable rightJust heading_platform readonly
   width=8
```

Zusätzlich kann es gewünscht sein, für diese Hierarchieebenen im Browser eine Auf- und Zuklappmöglichkeit zu haben.

Beispielsweise könnte man bei einer Abfrage „Übersicht über Kennzahlenlieferungen" drei Spalten zu Flächeninformationen haben (2005,2006,2007) in denen angegeben wird, ob geliefert wurde:

Flächen		
2005	2006	2007
X		
	X	X

wenn man den Punkt Flächen zuklappt, soll eine Zahl erscheinen, wieviele Lieferungen es für die Jahre 2005-2007 gegeben hat:

Um dies zu erreichen, müssen von der Datenbank vier Spalten geliefert werden flaeche2005,flaeche2006,flaeche2007 und flaeche_gesamt.

Der entsprechende Abschnitt in der XIL-Proplist muss so aussehen wie vorher mit Steuerzeichen \000 und allen vier Spalten

```
Column CID=2 heading_text="Flächen\000 2005" center_heading explanation=""
   row_selectable col_selectable heading_platform readonly  width=15 text_size=100
Column CID=2 heading_text="Flächen\000 2006" center_heading explanation=""
   row_selectable col_selectable heading_platform readonly  width=15 text_size=100
Column CID=2 heading_text="Flächen\000 2007" center_heading explanation=""
   row_selectable col_selectable heading_platform readonly  width=15 text_size=100
 Column CID=2 heading_text="Flächen\000 2005-7" center_heading explanation=""
   row_selectable col_selectable heading_platform readonly  width=15 text_size=100
```

und jetzt kommt der Clou:

Am Ende der XIL-Proplist macht man noch eine Angabe, welche Spalten den zu versteckende Aggregierungsspalten sind, also

```
hiddenAggregationColumns="Flächen\000 2005-7"
```

(Wenn es mehrere gibt, mit | getrennt angeben)

Dadurch weiß der Server, dass Flächen 2005,Flächen 2006 und Flächen 2007 Detailspalten sind und zeigt zunächst zur diese an. Wenn der Punkt Flächen zugeklappt wird, werden die Detailsspalten ausgeblendt und statt dessen wird die versteckte Aggregierungsspalte Flächen 2005-7 angezeigt.

Beim Auf- und Zuklappen wird vom Server nachgeladen, dass dauert zwar einen Moment, dafür ist der Server aber informiert und auch Druckversion und Excel-/PDF-Export können angepasst werden.

Diese Funktionalität wird im XML-Frontend ausgewertet, im Applet wird das Steuerzeichen sowie `hiddenAggregationColumns` einfach ignoriert.

2.8.9.7.3 PDF-Export

Kurz ein paar Hinweise:

Am besten nimmt man zur Bearbeitung eine bestehende pdf-Vorlage.

Tabellen:

Für jede Spalte muss direkt unter fo:table fin table-column Knoten mit der Breite kommen (im mm)

```
<fo:table>
<fo:table-column column-width=”30mm”>
<fo:block font-size=”10pt” text-align=”start/end” font-weight=”bold” font-
family=”serif” line-height=”9pt” space-before.optimum=”6pt” space-after.op-
timum=”6pt” language=”en” hyphente=”true”>
<xsl:value-if select=”format-number(/ergebnisse…,’#.###.##0,00,’German’)”/>
German groß ist wichtig!!!
</fo:block>
</fo:table-column>
<fo:table-header>
<fo:table-row>
<fo:table-cell border-width=”0.1mm” border-style=”solid” padding-
left=”0.5mm” padding-right=”0.5mm”>..
<fo:table-body>
<fo:table-row><fo:table-cell>..
```

```
<fo:block text-align=”center” font-size=”9pt” font-weight=”bold”
hyphenate=”false”>
```

Blöcke zusammenhalten

```
<fo:block keep-together.within-page=”always”>
Block1
Block2
</fo:block>
```

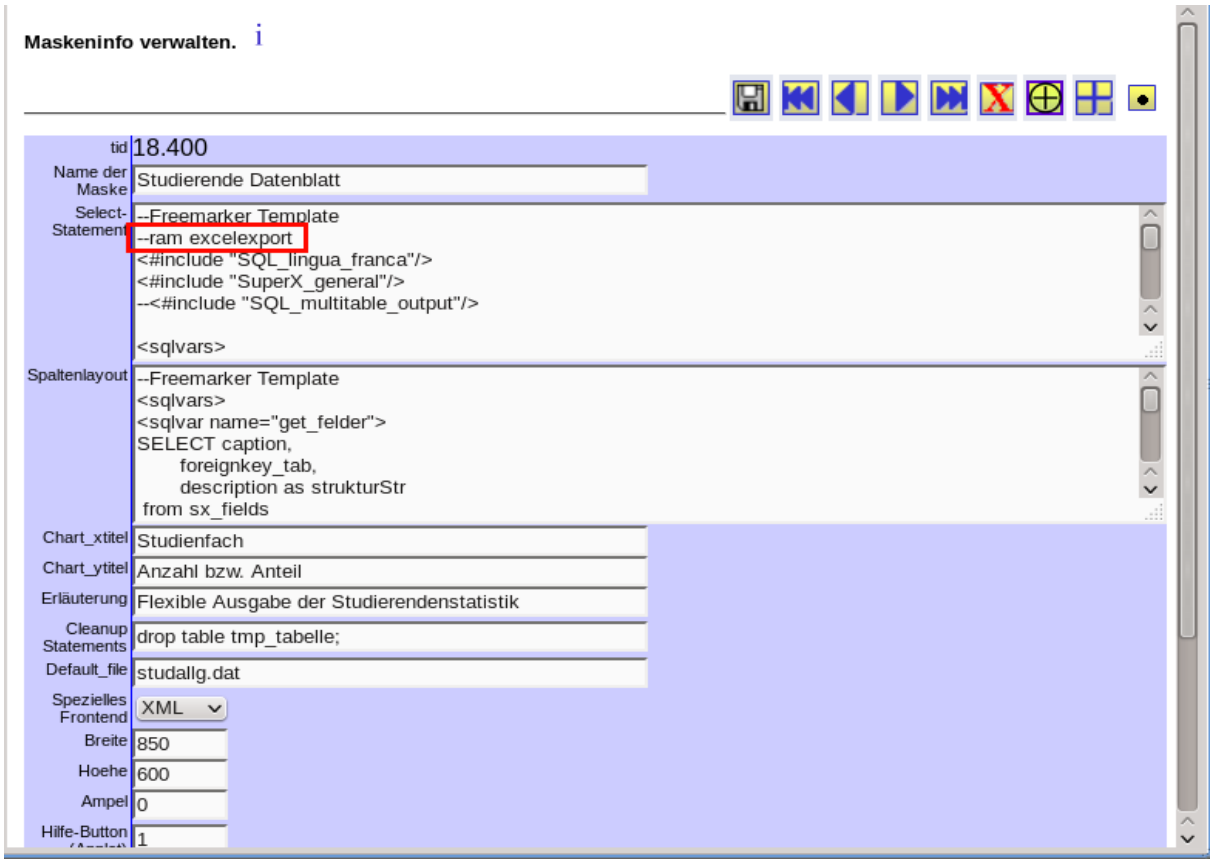
```
Um lokal zu testen gibt es Java-klasse de.superx.bin.ExcelPdfCreator
Params
```

-in/home/superx/iaf-ausgaben.xml -xsl/home/superx/tabelle_fo_pdf_xxxx.xsl -out/home/superx/test.pdf

(Dateiendung legt fest,dass PDF erzeugt werden soll)

2.8.9.7.4 Excelexport

Ab Kernmodul 4.5 kann man einen Performance-optimierten Excelexport anstoßen, indem man im Berichtskopf den Kommentar "`--ram excelexport`" setzt:



2.8.9.7.5Anpassung des Layout beim Excelexport

Kurz ein paar Hinweise:

Am besten nimmt man zur Bearbeitung eine bestehende xsl-Vorlage.

Man kann eine bestehende Exceldatei als Vorlage nehmen (attribut vorlage des xls_workbook Knotens).

Dies ist praktisch, um nicht direkt erzeugbare Einstellungen zu hinterlegen,z.B.

- Skalierung auf 70 %
- wiederholende Tabellenüberschrift (Seite einrichten / Tabelle)
- Extras/Schutz/Blattschutz (Poi kann trotzdem reinschreiben!)

Wenn man

```
<xls_workbook vorlage="vorlage1.xls" removeAdditionalSheets="true">
```

Wenn man Tabellen auf Vorrat angelegt hat, kann man mit dem Tag removeAdditionalSheets=true überflüssige Tabellen entfernen.

Es werden alle Zellen neu erzeugt, man kann jedoch einzelne Zeilen oder Zellen überspringen, um in der Excelvorlage Enthaltenes nicht zu überschreiben:

```
<xls_row jumpover="true">
<xls_cell jumpover="true"></xls_cell>
```

<xsl_sheet> ist ein Tabellenblatt.

<xsl_row> kann Attribute haben ebene=summe

Zellen

Für Zahlen <xls_cell style="body_dec" numeric="true">

mögliche Attribute: width (gilt logischweise für ganze Spalte)

```
Um lokal zu testen gibt es Java-klasse de.superx.bin.ExcelPdfCreator
Params
```

-in/home/superx/iaf-ausgaben.xml -xsl/home/superx/tabelle_xls_xxxxxx.xsl -out/home/superx/test.xls

(Dateiendung legt fest,dass Exceldatei erzeugt werden soll)

2.9 Erstellung von Datenblattberichten

Datenblattberichte sollen es für Hochschulen vereinfachen eigene Berichte z.B. mit JasperReports zu erzeugen. Diese Datenblattberichte müssen daher für jeden Geschmack die richtigen Informationen liefern ohne dabei Verluste bei Geschwindigkeit und Komfort zu haben. Für diese Herausforderung gibt es nun eine Lösung welche hier vorgestellt wird.

2.9.1Tabellendefinition

Ein Datenblattbericht hat als Datengrundlage immer eine Faktentabelle und beliebig viele Dimensionstabellen, die mit der Faktentabelle verknüpft sind. Die Definition der Tabellen wird im jew. Modul ausgeliefert, in der sog. "Modul-XML-Datei".

2.9.2Vorgehensweise

Um ein Datenblattbericht zu erstellen gehen Sie am besten folgendermaßen vor:

•Maske kopieren welche Ihrer zukünftigen Felderauswahl am nähesten kommt.
•Felder anpassen. Die Felder: Bericht, Weitere Tabellen, Felder, Schlüssel anzeigen und Ausgabeformat aus einem Bestehenden Datenblattbericht in Ihre neue Maske kopieren.
•Masken SQL (select_stmt) und Tabellenformat (xil_proplist) anpassen.
•Stylesheets der Maske zuordnen
•Spalten (Felder) dem Stylesheet zuordnen

2.9.3 Felder anpassen

Bei den Feldern Bericht, Weitere Tabellen und Felder müssen nach dem kopieren noch Änderungen vorgenommen werden.

Bei dem Feld Bericht muss bei relation und defaultwerd die MaskenID (tid) der neuen Maske eingetragen werden.

Bei den Feldern Weitere Tabellen und Felder müssen die Tabellen auf die sich die alte Maske bezogen hat mit denen der neue Maske ersetzt werden.

2.9.4 Masken SQL

Am einfachsten ist es, wenn Sie den Quellcode einer vorhandenen Datenblattmaske übernehmen und diesen dann anpassen.

Um den Code der Maske (select_stmt) übersichtlicher zu gestalten und da er sich zum Teil für jeden Datenblattbericht wiederholen würde, haben wir einiges in Macros gesteckt und diese in das Kernmodul (verfügbar ab Version 4.1) gepackt. Über `<#include "SQL_multitable_output"/>` werden die benötigten Macros in der Maske geladen und stehen zur verfügung.

Die erstel Zeilen eines Datenblattberichts werden also immer volgendermaßen aussehen:

```
--Freemarker Template
<#include "SQL_lingua_franca"/>
<#include "SuperX_general"/>
<#include "SQL_multitable_output"/>
```

Danach kommen die 4 SQL Vars: get_tables, get_table_joins, get_felder und get_felder_tk. Dieser sql-var Block untersheidet sich bei den Datenblattberichten nur an der Bezugstabelle. Bei Studierende Datenblatt ist es z.B. die Tabelle sos_stg_aggr. Diese müssen Sie dann nur mit Ihrer Tabelle ersetzten.

Danach werden 3 Macros aufgerufen:

```
<@generate_multitable_list />
<@generate_field_list_multitable aggregationsfeld="summe"
aggregatfunktion="sum("  />
<@generate_foreign_fields_multitable  />
```

Bei dem Macro `generate_field_list_multitable` muss eventuell das Aggregationsfeld geändert werden. Das Aggregationsfeld muss addierbar sein (also kein Text oder Datumsfeld) und sollte die Summenzeile der jeweiligen Tabelle sein.

Darauf folgt die Zuweisung der Filter. Die Filter müssen hier der Variable filter zugewisen werden. Die funktioniert mit z.B.:

```
<#assign filter="
/* and <<Hörerstatus>> */
/* and <<Semester>> */
" />
```

Die Filter die mehrfachauswahl erlauben oder eine Baumstruktur besitzen müssen speziell eingefügt werden. Dabei ist folgendes zu beachten:

1. Start des Kommentar Tags "/*" eine Zeile zuvor
2. filter zu dem Filter hinzufügen
3. Macro @printkeys geht hier nicht. Daher die Syntax von dem Beispiel unten verwenden
4. Buttonfeld immer am Ende Auskommentiert angeben. Z.B.: `--<<Studiengang>>`

```
/*
<#assign filter = filter + " and tid_stg in
"+Studiengang.allNeededKeysList />  --<<Studiengang>> */
```

Zum Schluss kommt noch das Macro, welches die bisherigen Eingaben verarbeitet. Dies wird unverändert an das Ende angehängt.

```
 <@generate_resultset_multitable tabellen=table_list
p_show_keys=<<Schlüssel anzeigen>>
p_field_list_select=field_list_select
p_foreign_fields=foreign_fields
p_filter=filter
p_field_list_groupby=field_list_groupby
p_join_clause=join_clause
/>
```

Und fertig ist der Datenblattbericht. Nun muss nur noch die XIL_Problist eingerichtet werden. Hier ist es am einfachsten, wenn die XIL_Problist eines vorhandenen Datenblattberichts genommen wird und in

der SQLVAR get_felder in der where Bedingung der Name der Bezugstabelle ausgetauscht wird. In dem Beispiel ist es die Tabelle sos_stg_aggr.

```
where table_name in ('sos_stg_aggr'
```

2.9.5Schlüssel-Anzeigen (data integrity)

Wenn Sie eine eigene Tabelle als Grundlage haben, müssen in der modul.xml data-integrity Einträge gepflegt werden, damit die Anzeige von Schlüsseln Ja/Nein funktioniert.

Beispiel:

```
<relation from="semester" to="man_facts" delete="FALSE" displayType="select"
visibleFields="eintrag" format="%s">
<relation-column from="tid" to="sem" />
</relation>
```

2.9.6Performance bei Datenblatt-Berichten

Datenblatt Berichte sind mitunter recht ressourcenintensiv, über die Konstante "Datenblatt max.Zeilenzahl" kann man steuern, wie viele Zeilen man zulassen will. Siehe http://www.superx-projekt.de.de/doku/kern_modul/admin/f_KonfigurationderDatenblatt-BerichtemaxZeilenanzahl.htm

Außerdem sollten Sie den Performanc-optimierten Excelexport aktivieren.

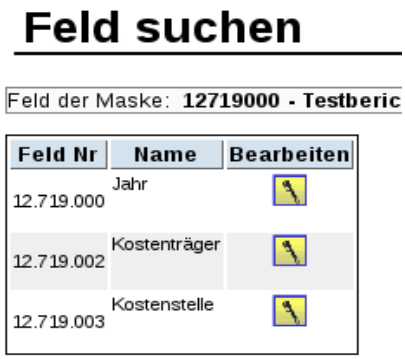
2.10 Eine einfache SAP-Abfrage

Ziel ist es, eine ganz einfache SAP-Abfrage zu zeigen, die per Kostenstelle und Projekte zeigt, wieviel Budget gebucht wurde, wieviel Einnahmen und Ausgaben es gegeben hat, wieviele Festlegungen sowie wieviel Geld noch verfügbar ist. Vorrang liegt dabei auf der Rechtekontrolle nach Kostenstelle und Projekt.

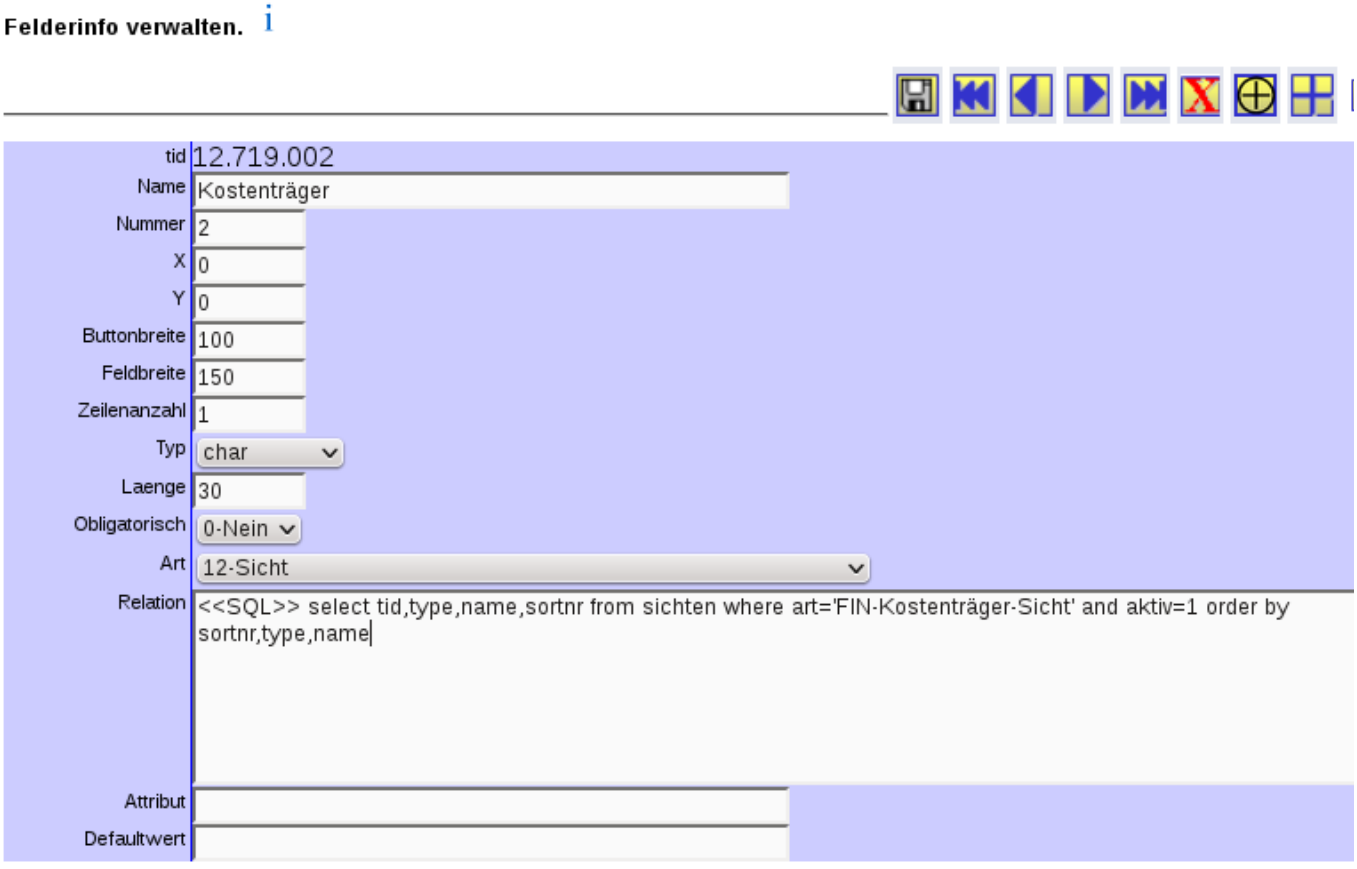
Die Maske sieht so aus:

Wenn man unter Administration/Masken/Felder Felder suchen auswählt, kann man die Maske auswählen.

Es erscheint eine Übersicht:



Klicken Sie z.B. in der Zeile Kostenträger auf den „Bearbeiten“-Button. Es öffnet sich ein Dialog.



Interessant ist die Zeile Relation, die angibt, welche Sichten für den Button aktiv sein sollen.

Sie können diese kopieren und in einem Datenbanktool wie pg_access oder der SQLWorkbench ausführen. Ergänzen Sie vielleicht noch um die Spalte Quelle.

```
select tid,type,name,sortnr,quelle from sichten where art='FIN-Kostenträger-Sicht' and aktiv=1 order by sortnr,type,name
```

Result | Messages

tid	type	name	sortnr	quelle
990	10	reguläre Sicht ...	0	sp_fin_ktr_hier(<<UserID>>,<<Stand>>,<<Sicht>>);select name,key,parent,strukturstr from tmp_hier order by name; drop table tmp_
1013	20	Forschungsprojekte ...	0	cob_alt_keys
1014	20	KoTr/Proj-Hier. (Stand Update 8.0) ...	0	cob_alt_keys

Type=10 ist die reguläre Sicht (Typ 20 sind alternative Hierarchien aus COB).

Bei Quelle der regulären Sicht sieht man, dass die Prozedur sp_fin_ktr_hier mit verschiedenen dynamischen Parametern aufgerufen wird. <<UserID>> wird zur Laufzeit mit der Userid (=tid in der Tabelle userinfo) des angemeldeten Users ersetzt, <<Stand>> mit dem gewünschten Standdatum und <<Sicht>> mit der Sichtnummer (noch nicht relevant).

Als Ergebnis liefert ein select auf die temporäre tabelle tmp_hier die Projekte, die der User sehen darf und diese werden dann im Baum beim Button Kostenträger dargestellt.

Bei einem eingeschränkten User sieht der Baum z.B. so aus, wenn er auf den Button Kostenträger klickt:

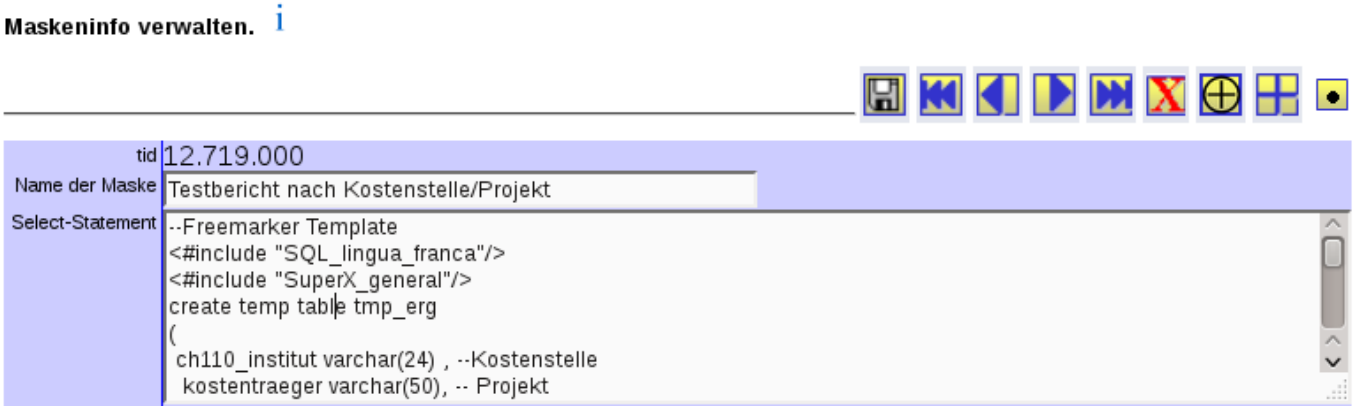
Die Rechteverarbeitung in der Maske wollen wir zunächst am Beispiel von Kostenstellen veranschaulichen.

Für die Berechnung der Werte ist das sogenannte select_stmt in der Tabelle Maskeninfo mit der tid=12719000 zuständig.

Sie können es sich ansehen, wenn Sie unter Administration / Masken / Maske suchen, die Maske auswählen:

Klicken Sie anschließend auf den Bearbeiten Button:

Das select-Statement steht im sich öffnenden Dialog an zweiter Stelle:



Sie können den Inhalt in einem Texteditor wie Jedit bearbeiten, was Vorteile wie Syntax Highlighting bietet, achten Sie aber darauf, dass die Kodierung mit Ihrem SuperX-System übereinstimmt, sonst können die Umlaute kaputt gehen.

Der Abschnitt im Select_statement für die Kostenstellen-Rechteverwaltung lautet:

```
--Start Rechtekontrolle auf Kostenstelle.
and  ch110_institut in <@printkeys Kostenstelle.allNeededKeysList />.
--Ende Rechtekontrolle auf Kostenstelle.
```

ch110_institut ist das Feld für die Kostenstelle. Das Makro printkeys Kostenstelle.allNeededKeysList listet alle berechtigten und gewünschten Kostenstellen auf.

Hat ein User z.B. nur Rechte auf Kostenstellen A und B wird daraus

```
 and  ch110_institut in   ('A','B')
```

Wähle er vorher im Baum aus, dass er nur Daten für die Kostenstelle A möchte, ergibt das Makro

```
 and  ch110_institut in   ('A').
```

Bei einem Admin oder einem User, der alle Kostenstellenrechte hat und keine Einschränkung auf dem Maskenfeld Kostenstelle macht, wird daraus ein langer SQL, z.B.

```
 and  ch110_institut in   ('root','A','B','C',.....).
```

Buchungen, bei denen keine explizite Kostenstelle angegeben wurde, erhalten als ch110_institut den obersten Knoten ('root').

Die Rechtekontrolle für Kostenträger ist komplizierter.

```
1  --Start Rechtekontrolle auf Projekte.
2    <#if K_FIN_PROJ_RECHTE=1||UserIsAdmin||(K_FIN_PROJ_RECHTE=2&&UserHasAllKostenstellenRights)>.
3  --Kostenträgerauswahl ist optional.
4   <#if Kostenträger?is_hash && Kostenträger.sichtname="externe Kostenträger">.
5    -- für externe Kostenträger ist ein Join über proj nötig, Kostenträger enthält Schlüssel .
6  /* and  (kostentraeger in (select projnr from fin_proj P where P.key_extkotr in <@printkeys Kostenträger.allNeededKeys/>) or <<Kostenträger>>='null')  */
7   <#else>.
8  /* and  (kostentraeger in <@printkeys Kostenträger.allNeededKeys/> or <<Kostenträger>>='null' )*/.
9  </#if> .
10 <#else>.
11 --nur berechtigte Kostenträger anzeigen.
12  <#if Kostenträger?is_hash && Kostenträger.sichtname="externe Kostenträger">.
13   -- für externe Kostenträger ist ein Join über proj nötig, Kostenträger enthält Schlüssel .
14  and  kostentraeger in (select projnr from fin_proj P where P.key_extkotr in <@printkeys Kostenträger.allNeededKeys/>)   .
15  <#else>.
16  and  kostentraeger in <@printkeys Kostenträger.allNeededKeys/>  .
17 </#if> .
18 .
19 </#if>.
20  --Ende  Rechtekontrolle auf Projekte.
```

Es wird zunächst unterschieden, ob die klassische Projekterechteverwaltung genutzt wird (K_FIN_PROJ_RECHTE=1), der User Admin ist oder alle Kostenstellenrechte hat, dann ist die Kostenträgerauswahl optional (Zeile 3-10), ansonsten werden nur berechtigte Kostenträger angezeigt (Zeile 12-19). Innerhalb der beiden Blöcke wird noch mal unterschieden, ob als Sicht „externe Kostenträger“ ausgewählt ist oder nicht (Zeile 4,12). Falls ja muss ein Unterselect gemacht werden, sonst nicht.

Bei der optionalen Kostenträgerauswahl wird durch die Notation /* ….. */ und or <<Kostenträger>>='null' dafür gesorgt, dass die Einfügung der Zeile optional ist, d.h. wenn in dem Maskenfeld „Kostenträger“ der Maske kein Wert ausgewählt wurde, wird die Zeile komplett entfernt.

Welcher SQL konkret erzeugt wird, kann man sich im Webanwendungs-Manager ansehen.
Rufen Sie dazu den Webanwendungs-Manager im Themenbaum auf.

Im mittleren Block können Sie sich den letzten SQL ansehen, der gelaufen ist.

letzter Masken sql (vom XML-Frontend)

```
--Abfrage Maske Testbericht nach
Kostenstelle/Projekt (12719000) durchführen
11:26:57
--Abfrage
create temp table tmp_erg
(
 ch110_institut varchar(24) ,   kostentraeger
varchar(50),  budget  decimal (14,2),
 einnahmen  decimal (14,2),
 ausgaben  decimal (14,2),
 festgelegt  decimal (14,2),
 verfuegbar  decimal (14,2)
)
;

insert into tmp_erg
select
ch110_institut,kostentraeger,sum(hhans_dr),
sum(einnahmen_dr),sum(ausgaben_dr),sum(festgelegt_
dr), sum(verfuegbar_dr)
from fin_konto_aggr
```

Der Webanwendungsmanager ist nur als Admin aufrufbar. Um mit einem eingeschränkten User zu testen, ist es am einfachsten, sich in einem Browser (z.B. InternetExplorer) als Admin anzumelden und den Webanwendungsmananger zu nutzen und in einem anderen Browser (z.B. Firefox) als eingeschränkter User die Abfrage durchzuführen. Wenn Sie dann im Webanwendungsmanager auf Log aktualisieren klicken, sehen Sie, welcher SQL für den eingeschränkten User erzeugt wurde.

3 Modulverwaltung

3.1 Modulverwaltung mit ANT

Das Datenmodell eines Edustore-Moduls ist in der $TOMCAT_HOME/webapps/superx/WEB-INF/conf/edustore/db/module/<<Modulname>>/conf/<<Modulname>>.xml abgelegt und dokumentiert (Ausnahme: Das Kernmodul liegt in $TOMCAT_HOME/webapps/superx/WEB-

INF/conf/edustore/db/install/conf/kern.xml). Aus dieser Datei kann man mit XSLT die entsprechenden HIS1-Dateien erzeugen, die für die Installation, den Upgrade, die Deinstallation und die Laderoutine eines Moduls benötigt werden. Bei manchen Modulen (z.B. GANG) werden sogar alle Datenbankformulare aus dieser Datei erzeugt.

Die *.tab und *.idx Dateien werden dann in sch.edustore erzeugt, view-Dateien in view.edustore

Die Klasse, die im Servlet/Browser die Installation ermöglicht, lautet

```
de.his.edustore.modules.WebFrontendForModuleInstall
```

Diese Klasse zeigt zunächst eine Liste der installierbaren Module. Wenn ein Modul ausgewählt wurde, geht die Klasse die o.g. Schritte durch:

1.Tabellen und Views des Moduls anlegen: Kopieren der oben erzeugten Schema-Dateien von $SUPERX_DIR/db/module/conf/his1/* nach qisserver/WEB-INF/conf/dbconv/conf/sch/sch.edustore sowie view.edustore, und Start der Methode headlessCloneDB.invokeEngine("check","edustore","edustore");

2.Bei HIS/CloneDB werden dann die Tabellen erzeugt, aber Angaben zum Spaltenmerkmal "default" werden ignoriert (in der Methode von HIS de.his.dbutils.schema.createTable). Deshalb fügen wir ggf. bei Spalten mit Default-Attributen das Merkmal mit "alter table ..." an, dabei wird jeweils für Postgres und Informix die richtige Syntax gewählt.

3.Stored Procedures anlegen: hier wird in der Moduldatei (z.B. $SUPERX_DIR/db/install/kern/conf/kern.xml) das Element <functions><function> ausgelesen, die Funktionen werden installiert.

4.Installationsscript für spezielle Modulscripte und Tabelleninhalte [derzeit Work in progress, hier müssen wir komplexere SQL-Scripte zur Installation ausführen. Hierzu müssen wir vermutlich dbinterface nutzen.]

5.Masken einspielen. Ersatz für das Shellscript $SUPERX_DIR/db/install/kern_masken_einspielen_pg.x (bzw. *_ids.x für Informix)

6.Datei $MODULPFAD/conf/customize.sql ausführen, wenn vorhanden (hier legen Hochschulen eigene Scripte ab)

7.dbforms-config.xml anpassen (Tabellendefinitionen für dbforms)

Nach der Modulinstallation ist ein Tomcat-Neustart notwendig.

Die Module können über Browser oder Kommandozeile bzw. direkt aus Eclipse als Java-Anwendung installiert werden. Hier ein Beispiel für die Linux-Kommandozeile

```
java -Xmx700M -cp "$QIS_CLASSPATH" de.his.edustore.modules.WebFrontendForMo-
duleInstall databases_meins.xml sos install '$SOS_PFAD=../../webserver/tom-
cat/webapps/superx/WEB-INF/conf/edustore/db/module/sos'
```

Achten Sie darauf daß Sie beim Start als Java-Anwendung in Eclipse keine einfachen Anführungszeichen benutzen. Ein Beispiel für Eclipse:

1.Menü "Run"->"Open Run Dialog"->Neue Java Application

2.Im Reiter "Main" bei Project den Projektnamen, und als Main Class "de.his.edustore.modules.WebFrontendForModuleInstall" eintragen

3.Im Reiter "Arguments" im Feld "Program Arguments" eingeben: databases_meins.xml sos install $SOS_PFAD=superx/WEB-INF/conf/edustore/db/module/sos [wenn Sie bei "Working directory: Default ${Workspace loc:webapps}]

3.1.1 Module-Scripts-Create mit Ant

In Edustore werden Datenbank-Metadaten in XML-Dateien gepflegt (s.o.). Aus diesen XML-Dateien werden Installations- und Pflegescripte erzeugt sowie Datenbankdokumentationen.

3.1.2 Umgebung einrichten

Als erstes müssen die Variablen JAVA_HOME und ANT_HOME gesetzt sein.

Nun gehen Sie in das Verzeichnis:

```
webapps/superx/WEB-INF/conf/edustore/db/module/<<Modulname>>
```

(Ausnahme Kernmodul: webapps/superx/WEB-INF/conf/edustore/db/install)

Um das Script nun auszuführen geben Sie folgenden Befehl ein:

```
ant -f ../../conf/module_scripts_create_ant.xml -DMODULE=<<MODULNAME>> -DDA-
TABASE=<<DATENBANKSYSTEM (POSTGRES, INFORMIX oder HSQLDB>> <<PARAMETER>>
```

3.1.3 Beispiele:

Für das SOS-Modul müsste zuerst in das Verzeichnis:

```
webapps/superx/WEB-INF/conf/edustore/db/module/sos
```

gewechselt werden. Mit einer Postgres Datenbank würde der Befehl um das Script komplett auszuführen folgendermaßen aussehen:

```
ant -f ../../conf/module_scripts_create_ant.xml -DMODULE=sos
-DDATABASE=POSTGRES all
```

Für das FIN-Modul müsste zuerst in das Verzeichnis:

webapps/superx/WEB-INF/conf/edustore/db/module/fin

gewechselt werden. Mit einer Informix Datenbank würde der Befehl um das Script nur zum erzeugen der Dokumentation auszuführen folgendermaßen aussehen:

```
ant -f ../../conf/module_scripts_create_ant.xml -DMODULE=fin -DDATABASE=IN-
FORMIX doc
```

Beim Kernmodul befinden wir uns in der Shell in db/install, daher muss man, um nach db/conf/ zu gelangen, nur ein Verzeichnis nach oben gehen. Hier ein Beispiel für Windows mit Postgres Datenbank welches nur die Scripte für die Modulinstallation erzeugt:

```
ant -f ..\conf\module_scripts_create_ant.xml -DMODULE=kern -DDATABASE=POST-
GRES install
```

3.1.4 Parameter:

all	Modulscripte komplett erzeugen
backup	Erzeugt Scripte für Backup/Restore und kürzen und wiederherstellen der Felderinhalte
dbforms	Erzeugt DB-Forms (jsp-Seiten, Muster für dbforms-config.xml)
deinst	Erzeugt die Deinstallation-Scripte (SuperX-Platform)
doc	Erzeugt die Datenbank-Dokumentation (Entladeroutine, Laderoutine, Modulbeschreibung im HTML- bzw. RTF-Format)
edustore_etl	Erzeugt die ETL-Scripte für HIS-Edustore (HISinOne-Platform)
edustore_install	Erzeugt die Installations-Scripte (HISinOne-Platform) inkl. Upgrade-Scripte und Maskenverwaltung im Verzeichnis conf/his1/edustore_install
etl	Erzeugt ETL-Scripte (SuperX-Platform)
ice	Erzeugt die Scripte zum Füllen der Metadaten-Tabellen und Hilfstabellen für ICE
install	Erzeugt die Modulscripte für die Modulinstallation (SuperX-Platform)
mask	Erzeugt die Verwaltungsscripte für die Masken des Moduls (Laden, Entladen)
mediawiki	Erzeugt die Datenbankdokumentation im Mediawiki-Format im Verzeichnis conf/his1/edustore_doc
meta	Erzeugt die SQL Dateien, die die Metadaten-Tabellen und Felder füllen/leeren
upgrade	Erzeugt das Script zum Datenbank-Upgrade (SuperX-Platform) (Aktualisierung einer vorhandenen Modulinstallation auf die aktuelle Version)

Mit dem Befehl:

```
ant -f ../../conf/module_scripts_create_ant.xml help
```

wird auch nochmal eine Erläuterung in der Shell ausgegeben.

3.1.5 Erläuterung zum Aufbau der ANT-Datei

Die Targets, die zum Starten gedacht sind, stehen ganz am Anfang unter dem Punkt "Start - Parameter" . Diese Targets enthalten alle den Parameter:depends="init", da in init die später benötigten Variablen gefüllt werden. In den Start-Targets steht in der Regel nur ein Aufruf der zugehörigen Target-Gruppe mit antcall. In den Target-Gruppen werden dann die einzelnen Targets aufgerufen.

Namensgebung:

Start-Targets: <<Ober-Bezeichnung>>

Gruppen-Targets: modul_<<Ober-Bezeichnung>>

Targets: modul_<<Ober-Bezeichnung>>_<<Target-Bezeichnung>>_superx

3.2 Modul XML

Das XML-Format hat den Vorteil, dass die Scripte dynamisch für Postgres und Informix erzeugt werden können, und dass die Scripte vereinheitlicht werden. Aus dieser Datei werden die Scripte erzeugt, die das Modul jeweils für Postgres und Informix installieren / updaten /aktualisieren / überprüfen und entfernen.

Im folgenden finden Sie eine genaue Erläuterung der Elemtente:

Um neue Tabellen in der Datenbank in ein Modul zu integrieren sind einige Schritte erforderlich. In dieser Anleitung werde ich anhand von einem Beispiel zeigen, wie es geht. Wichtig dabei ist auch penibel auf die Bezeichnungen zu achten, da die Groß- und Kleinschreibung bei Tabellen- und Feldnamen berücksichtigt wird. Daher ist "Gang" nicht gleich "gang" und zusätzliche Leerzeichen führen auch zu Fehlern.

Als Beispiel nehme ich die Tabelle gang_stg_astat, die wie der Name schon verrät, ins GANG-Modul soll.

Zuerst müssen Sie die XML-Datei des entsprechenden Moduls öffnen. Diese liegt in: „$SUPERX_DIR/db/module/<<MODUL>>/conf". Die Datei heißt <<MODUL>>.xml.

In meinem Beispiel wäre dies die gang.xml die in „$SUPERX_DIR/db/module/gang/conf" liegt. Die Modul-Dateien sind in der Regel mehrere tausend Zeilen lang und daher unübersichtlich, wenn man keine Legende hat. Daher würde ich JEdit zum bearbeiten empfehlen, da Sie sich hier auf der linken Seite diese Legende einblenden lassen können.

Nachdem nun die Modul-Datei geöffnet ist können Sie in JEdit in der Legende sehr schön die Aufteilung der Datei in 6 Unterbereiche sehen: "database", "install", "uninstall", "upgrade", "etl" und "dbforms".

3.2.1 Database

Im "database"-Element werden die Tabellen, Views, Themen und Masken Integriert.

3.2.2 Tabellendefinitionen in der Modul-XML-Datei

3.2.2.1 Allgemeines

Dies ist der Kernbereich. Um hier eine Tabelle einzutragen, muss eine genaue Syntax eingehalten werden, damit es funktioniert. Das Hauptelement, die Tabelle, startet mit `<table>` und endet mit `</table>`. Wie Sie sehen werden die Elemente in Größer - Kleinerzeichen gesetzt und das Ende wird einfach mit einem Schrägstrich vor dem Namen bekannt gemacht. In dem Element Tabelle kommen noch die Elemente "description", "columns" und "primaryKeys". Zusammen würde das dann folgendermaßen aussehen:

```
<table>
     <description></description>
     <columns></columns>
```

```
    <primaryKeys></primaryKeys>
</table>
```

Das ist sozusagen das Grundgerüst. Bestimmte Eigenschaften und der Name der Tabelle kommen wie in der HTML Programmierung mit in die Größer - Kleinerzeichen. In meinem Beispiel würde es dann so aussehen:

```
<table name="gang_stg_astat" caption="Studienfächer (amtlich) zu Lehreinhei-
ten" version="0.1" thema="Studiengänge" typ="Schlüsseltabelle"
datenquelle="1">
```

Beschreibung:

name="Tabellenname"

caption="Beschreibung"

version="Version der Tabelle"

thema="Das Thema, welchem die Tabelle zugeordnet werden kann"

typ="Art der Tabelle"

datenquelle=""

Das Element "description" bietet Ihnen die Möglichkeit eine ausführlichere Beschreibung der Tabelle zu hinterlassen. Der Text wird wie folgt eingefügt:

```
<description>Hier kommt die Beschreibung rein</description>
```

In dem Element "columns" geht es um die Spalten der Tabelle. Hier wird für jede Spalte ein Element "culumn name" eingerichtet, welches bestimmte Eigenschaften zugeordnet wird. Dies sieht z.B. so aus:

```
<column name="lehreinheit" type="CHAR" size="10" notnull="false" descripti-
on="Lehreinheit" currentlyUsed="true"><comment>(hochschulinterner
Schlüssel)</comment></column>
```

Beschreibung:

```
name="Name der Spalte"
type="Datentyp"
size="größe des Datentyps"
notnull="ob das Feld leergelassen werden darf"
description="Beschreibung der Spalte. Dieser Text wird auch in Daten-
blatt-Berichten als Feldname angezeigt"
currentlyUsed="true/false" (in der Doku wird angezeigt dass das Feld nicht
ausgewertet wird, und in Datenblattberichten wird die Spalte nicht im Feld
"Felder" angezeigt.
```

Sowie als Unterelement

```
<comment>Hier kann noch ein zusätzliches Kommentar rein</<comment>
```

Im Element "primaryKeys" werden die Primärschlüssel der Tabelle eingetragen. Hier gilt folgende Syntax:

```
<rs>
<row>
```

```
 <fld name="table_cat">superx</fld>
 <fld name="table_schem">superx</fld>
 <fld name="table_name">gang_stg_astat</fld>
 <fld name="column_name">tid</fld>
 <fld name="key_seq">1</fld>
 <fld name="pk_name">gang_stg_stat_tid</fld>
</row>
</rs>
```

Die Elemente "fld name" müssen hier von "rs" und "row" umschlossen werden. Für einen Weiteren "primaryKey" wird ein neues Element "row" in dem selben "rs" geschrieben.

```
Zur Beschreibung:
table_cat="Databasenamen"
table_schem="Databasename"
table_name="Tabellenname in dem der Primarykey steht"
column_name="Spaltenname vom Primarykey"
key_seq="Der wievielte Primarykey ist dies von der Tabelle?"
pk_name="Name vom Primarykey, üblicherweise <<Tabellenname>>_<<Spaltenname>>"
```

Im gesamten sieht es in meinem Beispiel von der gang_stg_astat so aus:

```
<table name="gang_stg_astat" caption="Studienfächer (amtlich) zu Lehreinheiten" version="0.1"
      thema="Studiengänge" typ="Schlüsseltabelle" datenquelle="1">
 <description>Zuordnung amtlicher Studienfächer einer Hochschule zu internen    Lehreinheiten.</descrip-
tion>
 <columns>
  <column name="tid" type="SERIAL" size="4" default="" notnull="true" description="Laufnummer"        is-
Key="true"/>
  <column name="lehreinheit" type="CHAR" size="10" notnull="false"      description="Lehreinheit"><com-
ment>(hochschulinterner Schlüssel)</comment></column>
  <column name="beschreibung" type="CHAR" size="150" notnull="false" description="Bezeichnung des
      Studienfachs"/>
  <column name="astat" type="CHAR" size="3" notnull="false" description="Externer Schlüssel"/>
  <column name="studienbereich" type="CHAR" size="10" notnull="false" description="Schlüssel   Studienbe-
reich"/>
  <column name="aktiv" type="SMALLINT" notnull="false" description="Aktiv"><comment>Nur aktive Stati  (1)
werden im Studiengangsbaum angezeigt</comment></column>
  <column name="bund_key" type="CHAR" size="10" notnull="false" description="Bundesschlüssel   Lehrein-
heit"/>
  <column name="datenquelle" type="INTEGER" size="4" default="1" notnull="false"       description="Da-
tenquelle für diesen Datensatz"/>
  <column name="erzeugt_am" type="DATE" size="4" default="today()" notnull="false"     description="Da-
tum der Erzeugung des Datensatzes" currentlyUsed="false"/>
  <column name="geaendert_am" type="DATE" size="4" default="" notnull="false" description="Datum der
      letzten Änderung des Datensatzes" currentlyUsed="false"/>
  <column name="geaendert_von_id" type="INTEGER" size="4" default="" notnull="false"   description="Use-
rID des Ändernden" currentlyUsed="false"/>
  <column name="erzeugt_von_id" type="INTEGER" size="4" default="" notnull="false"     description="Use-
rID des Erzeugers" currentlyUsed="false"/>
 </columns>
 <primaryKeys>
  <rs>
   <row>
    <fld name="table_cat">superx</fld>
    <fld name="table_schem">superx</fld>
    <fld name="table_name">gang_stg_astat</fld>
    <fld name="column_name">tid</fld>
    <fld name="key_seq">1</fld>
    <fld name="pk_name">gang_stg_stat_tid</fld>
    </row>
  </rs>
```

```
 </primaryKeys>
</table>
```

3.2.2.2 Tabellen umbenennen HOWTO

Zuerst wird das Datenbankschema in der Modul-XML-Datei angepaßt, also hier: superx/WEB-INF/conf/edustore/db/install/conf/kern.xml . Dann werden daraus die Installationsscripte erzeugt (http://wiki.his.de/mediawiki/index.php/Edustore:_Datenbank-_und_Modulinstallation#Module-Scripts-Create_mit_Ant) . Wenn die Tabelle auch gefüllt werden soll, muss eine Einfügeoperation mit den Elementen loadtable erzeugt werden.

```
<loadtable refresh="true" delimiter="^" header="false"
tabname="schluessel"><file path="$SUPERX_DIR/db/install/rohdaten/schlues-
sel.unl"/></loadtable>
```

Zu Klären ist, ob auch bisherigen Anwender die Tabellen Upgraden können. Es müsste ein Upgrade-SQL-Script entwickelt werden, das zuerst prüft, ob die alten Tabellen mit den alten Feldnamen noch existieren, und wenn ja, müssen sie entfernt werden. Derzeit planen wir das nur für eine Neuinstallation.

Die Erzeugung der neuen Tabellen passiert beim Install/Upgrade automatisch, darum muss man sich nicht kümmern.

Beispiel Tabelle schluessel: neuer Name menu_element:

Alt:

```
<table name="schluessel" version="2.0" thema="Administration" typ="Schlüs-
seltabelle">
<columns><column name="id" type ="INTEGER" size ="4" default ="" notnull
="true" />
<column name="variable" type ="CHAR" size ="50" default ="" notnull ="false"
/>
<column name="wert" type ="CHAR" size ="255" default ="" notnull ="false" />
<column name="beschreibung" type ="CHAR" size ="255" default ="" notnull
="false" />
<column name="typ" type ="CHAR" size ="255" default ="" notnull ="false" />
<column name="erlaeuterung" type ="CHAR" size ="255" default ="" notnull
="false" />
</columns>
```

Neu:

```
<table name="menu_element" version="4.0" thema="Administration" typ="Schlüs-
seltabelle">
<columns><column name="id" type ="SERIAL" size ="4" default ="" notnull
="true" />
```

```
<column name="element_name" type ="VARCHAR" size ="255" default ="" notnull
="false" />
<column name="element_value" type ="VARCHAR" size ="255" default ="" notnull
="false" />
<column name="element_descr" type ="VARCHAR" size ="255" default ="" notnull
="false" />
<column name="element_type" type ="VARCHAR" size ="255" default ="" notnull
="false" />
<column name="element_comment" type ="VARCHAR" size ="255" default ="" not-
null ="false" />
</columns>
```

Tabelle beim Install/Upgrade füllen in dem Script

```
<loadtable refresh="true" delimiter="^" header="false"
tabname="schluessel"><file path="$SUPERX_DIR/db/install/rohdaten/schlues-
sel.unl"/></loadtable>
```

Achtung: es gibt auch views, die auf der Tabelle schluessel basieren.

Tabelle hinweise und hinweis_akzept: neuer Name user_dialog und user_dialog_accept, Vorgehen analog. Bitte die Spalten auch auf englisch übersetzen, und als Primärschlüssel "id" statt "tid". Text-Datentypen bekommen immer "varchar" statt "char".

Dann muss der Quellcode nach dem betr. Tabelle durchsucht werden. Tabellen des Kernmoduls werden auch im Java Code der Webanwendung abgefragt.

```
webapps/superx/WEB-INF/edit/**/*.jsp
webapps/superx/WEB-INF/conf/edustore/db/**/*.sql|*.xml
webapps/superx/WEB-INF/conf/edustore/db/**/masken/*.unl

src/de/superx/**/*.java
```

Dann gibt es noch eine spezielle Herausforderung in HIS1: Die alten Tabellen/Views liegen als Schema-Dateien noch in

```
/qisserver/WEB-INF/conf/dbconv/conf/sch/sch.eduetl/*
/qisserver/WEB-INF/conf/dbconv/conf/sch/view.eduetl/*
```

sowie im SVN in

```
/superx/WEB-INF/conf/edustore/db/install/conf/his1/dbconv/sch.eduetl/*
/superx/WEB-INF/conf/edustore/db/install/conf/his1/dbconv/view.eduetl/*
```

Bei Neuinstallationen ist das kein Problem, aber bei vorhandenen Installationen müssen diese Dateien unbedingt gelöscht werden, sonst würden sie beim Neuaufbau der Datenbank wieder angelegt.

3.2.3 Views

Für die Views wird wie bei den Tabellen eine bestimmte Syntax verfolgt. Diese erläutere ich an dem Beispiel der gang_k_sb:

```
<view name="gang_k_sb" version="0.1">
<description>View Studienbereiche (Landesschlüssel)</description>
<columns>
<column name="apnr" type ="char" size ="2" default ="" notnull ="true" />
<column name="kurz" type ="VARCHAR" size ="150" default ="" notnull ="false"
/>
 <column name="druck" type ="VARCHAR" size ="255" default ="" notnull ="fal-
se" />
<column name="lang_1" type ="VARCHAR" size ="255" default ="" notnull ="fal-
se" />
<column name="astat" type ="SMALLINT" size ="2" default ="" notnull ="false"
/>
</columns>
<sql dbsystem=""><![CDATA[SELECT apnr,kurz,druck ,lang_1,astat FROM
gang_cifx where key=3 ]]></sql>
</view>
```

Die Vorgehensweise ähnelt hier der, der Tabellen. In "view" wird der Name und die Version des Views eingetragen und in "description" die Beschreibung. In "columns" wird für jede Spalte ein "column" Element angelegt mit den entsprechenden Eigenschaften, wie "name", "type", "size" ... . In "sql" wird dann der select des Views eingetragen. Die Besonderheit hier ist, dass "CDATA". Durch diesen Befehl wird der eingeschlossene Text nicht von XSLT ausgewertet, sondern direkt übergeben.

3.2.4 Themen

Hier handelt es sich um die Überschriften im Themenbaum, z.B. "Auswertungen zu Studiengängen". Im Attribut Parent wird das übergeordnete Element definiert.

```
<thema name="Auswertungen zu Studiengängen" parent="Studiengänge">Abfragen
zur Akkreditierung etc.</thema>
```

In der jew. Modul-XML gibt es zwei Varianten:

-Das oberste Element zum Themenbaum steht ganz oben, z.B. "Studierende, Prüfungen" in der sos.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<module name="sos" version="1.0b" sachgebiet_id="16" sachgebiet="Studieren-
de" systeminfo_id="7" system="Studierende, Prüfungen" thema="Studierende,
Prüfungen" thema_parent="Abfragen">
<description>Abfragen und Berichte im Bereich der Studierenden- und Prü-
fungsverwaltung, d.h. Statistiken zu Studierenden oder Absolventen nach ver-
schiedenen Merkmalen. @@sos_doku_benutzer_url@@</description>
```

Bei den Themen darunter steht z.B. in der sos.xml:

```
<thema name="Administration Studierende, Prüfungen" parent="Studierende,
Prüfungen">Abfragen zur Administration des SOS-Moduls @@sos_doku_admin_ur-
l@@</thema>
```

Nach Generierung der Modulscripte wird der Themenbaum entsprechend aufgebaut, und das Thema entsprechend mit dem Inhalt unter Informationen hinterlegt. Wichtig ist aber: das passiert nur bei der Erstinstallation bzw. beim Upgrade nur dann, wenn das Thema noch nicht existiert und neu eingefügt ist.

Warum? Damit die Hochschulen das bei Bedarf ändern können.

Wenn Sie den Inhalt des Feldes `themenbaum.erlaeuterung` auch bei vorhandenen Installationen ändern wollen, müssen Sie das als speziellen Upgrade-Step codieren.

Beispiel: in sos.xml:

```
...
<upgrade-step>
...
<nativeaction sql="" scriptfile="$SOS_PFAD/schluesseltabellen/themenbaum_up-
date.sql"
     database=""/>
...
</upgrade-step>
```

Wir machen das bisher selten, weil es wie gesagt eigentlich nicht so gedacht ist.

Fazit: um Release Captions zu ändern muss man also

1. die jew. Modul-XML anpassen, beim obersten Element unter /module/description, für elemente darunter im Bereich <thema... />
2. für vorhandene Installationen einen Upgrade step einrichten, nach dem Motto: `update themenbaum set erlaeuterung=... where name=...;`
3. Modulscripte generieren
4. In der jew. Installation das Modul dann Upgraden

3.2.5Masken

Hier werden die Masken bzw. Berichte des Moduls deifnieret. Diese Definitionen werden z.B. genutzt, um die Masken zu installieren oder zu entladen.

```
<maske tid="25240" name="Zulassung" thema="Auswertungen zu Studiengängen">
<src><path>$GANG_PFAD/masken</path><author>D.
Quathamer</author><lastChange>23.5.2007</lastChange><dbsystem>INFORMIX</db-
system></src>
<src><path>$GANG_PFAD/masken</path><author>D.
Quathamer</author><lastChange>23.5.2007</lastChange><dbsystem>POSTGRES</db-
system></src>
</maske>
```

3.2.6 Data-integrity

In der Data-integrity werden die Verknüpfungen der Tabellen eingetragen. Dies ist einmal für die <<Modul>>.html wichtig, da dann die Verknüpfungen auch als Link zu der entsprechenden Tabelle erscheinen und damit in SuperX in den entsprechenden Tabellen nicht nur eine Nummer, sondern auch ein aussagekräftiger Text steht.

Außerdem kann mit den Relationen in HISinOne automatisch ein Foreign Key-Ausdruck erzeugt werden.

3.2.6.1 Beispiel 1: Deskriptive Eigenschaften einer Relation

In meinem Beispiel verknüpfe ich die Tabellen `gang_k_lehr_hs` mit der `gang_stg_astat`. Auch die Reihenfolge spielt hier eine wichtige Rolle. Die erste Tabelle ist die, in der der entsprechende Inhalt und der Schlüssel in einer Spalte stehen,a lso die "Schlüsseltabelle". Die zweite Tabelle enthält nur den Schlüssel, von dem auf die Schlüsseltabelle referenziert wird. Daher wird in der <<Modul>>.xml in der zweiten Tabelle der entsprechende Fremdschlüssel eingetragen. Im folgenden Beispiel steht also der Schlüssel in der Spalte "gang_stg_astat.lehreinheit", und dies referenziert auf die Spalte "gang_k_lehr_hs.key_apnr". Der Bezeichnungstext wird dann aus der "gang_k_lehr_hs.drucktext" geholt:

```
<!--gang_stg_astat -->
<relation from="gang_k_lehr_hs" to="gang_stg_astat" delete="FALSE" display-
Type="select" visibleFields="drucktext" format="%s">
<relation-column from="key_apnr" to="lehreinheit" />
</relation>
```

In visibleFields wird die entsprechende Spalte eingetragen, die ausgegeben werden soll. Mit zwei Pipes "||" können auch mehrere Spalten miteinander verbunden werden, wenn z.B. der Key und der Text ausgegeben werden soll.

Beispiel:

```
<!--gang_stg_astat -->
<relation from="gang_k_lehr_hs" to="gang_stg_astat" delete="FALSE" display-
Type="select" visibleFields="key_apnr || trim(drucktext)" format="%s">
<relation-column from="key_apnr" to="lehreinheit" />
</relation>
```

Zur Beschreibung "relation":

```
from="Tabelle mit Schlüssel und zugehörigen Text"
to="Tabelle nur mit Schlüssel"
delete=""
displayType=""
visibleFields="welche Spalten angezeigt werden sollen"
format="Format der Anzeige"
```

Zur Beschreibung "relation-column":

```
from="Key-Spalte der ersten Tabelle"
```

```
to="Key-Spalte der zweiten Tabelle"
```

3.2.6.2 Beispiel 2: Erzeugung von Fremdschlüsseln

Die Relation in der data-integrity kann auch für die Generierung von (datenbankbasierten) Fremdschlüsseln genutzt werden. Die Syntax ist dieselbe wie oben, nur das Attritut `createForeignKey="true"` muss hinzugefügt werden. Beispiel:

```
<relation from="tabellenmerkmale" to="tabellenschluessel" delete="FALSE"
displayType="select" visibleFields="merkmal" format="%s"
createForeignKey="true">
<relation-column from="mschluessel" to="mschluessel" />
</relation>
```

Daraus wird folgende Datei erzeugt:

```
<<Modulname>>/conf/his1/dbconv/sch.edudata/tabellenschluessel.fk
```

mit dem Inhalt:

```
alter table tabellenschluessel ADD CONSTRAINT fktabellenschluessel_mschlues-
sel FOREIGN KEY (mschluessel) REFERENCES tabellenmerkmale(mschluessel) ;
```

Auch Fremdschlüssel über mehrere Spalten können erzeugt werden, hier ein Beispiel:

```
<relation from="tabellenschluessel" to="zusvirtuell" delete="FALSE" display-
Type="select" visibleFields="merkmal" format="%s" createForeignKey="true">
<relation-column from="mschluessel" to="merkmal" />
<relation-column from="aschluessel" to="ausprerg" />
</relation>
```

Resultat:

```
alter table zusvirtuell ADD CONSTRAINT fkzusvirtuell_merkmal_ausprerg FOR-
EIGN KEY (merkmal, ausprerg) REFERENCES tabellenschluessel(mschluessel,
aschluessel) ;
alter table zusvirtuell ADD CONSTRAINT fkzusvirtuell_merkmal_ausprsummand
FOREIGN KEY (merkmal, ausprsummand) REFERENCES tabellenschluessel(mschlues-
sel, aschluessel) ;
```

Hinweis: wenn eine Tabelle mehrere Fremdschlüssel hat, ist es übersichtlicher, wenn Sie am Anfang einen Kommentar mit dem Ziel-Tabellennamen setzen, also oben z.B.

```
<!-- zusvirtuell-->
 <relation from="tabellenschluessel" to="zusvirtuell"...
<relation from="tabellenschluessel" to="zusvirtuell"...
```

3.2.6.3 Beispiel 3: Referenztabelle hat zusätzliche Filter

Wenn eine Fremdschlüsselbeziehung auf mehrere Spalten geht, dann können auch mehrere "relatio-columns" angegeben werden. Im folgenden Beispiel referenziert das Feld `lehr_stg_ab.lehr` auf die Tabelle `organigramm`, allerdings nur auf Werte mit der `orgstruktur=30`:

```
<relation from="organigramm" to="lehr_stg_ab" delete="FALSE"
displayType="select" visibleFields="name" format="%s">
<relation-column from="orgstruktur" to="30" />
<relation-column from="key_apnr" to="lehr" />
</relation>
```

Vorsicht bei zusätzlichen Filtern, die vom Typ CHAR sind:hier muss das '-Zeichen mit einem Backslash maskiert werden. Beispiel:

```
<relation from="menu_element" to="felderinfo" delete="FALSE"
displayType="select" visibleFields="description" format="%s">
<relation-column from="element" to="\'Feldtyp\'" />
<relation-column from="element_value::char(15)" to="typ" />
</relation>
```

3.3 Install, uninstall , ETL und upgrade

Hier werden Aktionen und Ladeschritte bei der jeweiligen Aktion eines Moduls definiert. Das Hauptaugenmark liegt hier auf "nativeaction" und "loadtable". Eine "nativeaction wäre z.B.:

```
<nativeaction sql="" scriptfile="$GANG_PFAD/conf/gang_tabellen_fuellen.sql"
     database=""/>
```

Hier wird der Inhalt der SQL Datei genommen und ausgeführt. Es ist auch möglich einen SQL Befehl direkt in das Attribut "sql" zu schreiben. Falls erforderlich, kann auch noch als Bedingung die Datenbank angegeben werden (z.B. POSTGRES oder INFORMIX), für die diese "nativeaction" gedacht ist.

Für "loadtable" ist folgendes Beispiel:

```
 <loadtable refresh="false" delimiter="^" header="false"
tabname="gang_cifx"><file
path="$GANG_PFAD/schluesseltabellen/gang_cifx.unl"/></loadtable>
```

Hier wird der Inhalt einer Tabelle in eine Datei abgespeichert. "delimiter" steht für Feldtrenner, "header" ob die Spaltenüberschrift übernommen werden soll und "tabname" für den Tabellennamen. in "file" in dem Attribut "path" wird der Pfad mit Dateinamen angegeben, wo der Inhalt abgespeichert werden soll.

3.3.1 Spezialität bei ETL

In dem ETL Modul sind 6 Stufen eingebaut. Zu beachten ist hier auch, wenn eine Stufe nicht richtig verarbeitet wird, bricht das Script komplett ab.

Zu den einzelnen Stufen:

Unload: Hier werden die Daten aus dem Vorsystem entladen.

Load: Hier werden die Daten in das Datawarehouse geladen.

Trans: Hier wird die Transformation durchgeführt. D.h. es wird eine Schlüsselharmonisierung durchgeführt.

Aggregation: Hier werden die Hilfstabellen gefüllt.

System: Hier wird das Datum aktualisiert, welches die Aktualität angibt.

Test: Hier findet die Prüfroutine statt.

3.3.2 olap-system

Hier werden die Dimensionen und Faktentabellen für ICE definiert.

Beispiel Dimension:

```
<dimension-blueprint system_key="bluep_stg" name="Studienfach (intern)"
keyandparentcoltype="varchar"  datatype="nominal"/>
```

Beispiel Faktentabelle:

```
<fact-table name="sos_stg_cube" >
     <calc-function>sum</calc-function>
</fact-table>
```

3.4 Patches

In SuperX gibt es Scripte um Patches zu erstellen, welche über weitere Scripte an der Hochschule automatisch eingespielt werden können. Dies ermöglicht es kleinere Änderungen in ein SuperX System zu übernehmen ohne gleich ein Modulupgrade durchzuführen. Außerdem kann ein Patch auch Änderungen für mehrere Module beinhalten.

Zugehörige Dateien:

`db/bin/patch_scripts_create.x` – Dieses Script startet die Pachtgenerierung mit ANT .

`db/conf/patch_scripts_create_ant.xml` – Wird von ANT aufgerufen. Dieses Script verwaltet den Ablauf.

`db/conf/patch_generate.xsl` – Hier wird der Patch erzeugt. Es werden temporäre Scripte erzeugt, welche das Patch-Paket schnüren und Scripte, welche in das Patch-Paket eingefügt werden.

`WEB-INF/patch/xml/patch_2011-xx-xx.xml` – In dieser Datei stehen die Änderungen, welche von den zuvor erwähnten Dateien verwendet werden um das Patch-Paket zu erstellen.

`WEB-INF/patch/patch-archive` – ist der Ordner in dem die fertigen Patches abgelegt werden.

3.4.1 Anleitung zur Erstellung von Patches

Die Patches werden aus der Datei patch-2011-xx-xx erstellt. Wobei xx-xx für Monat und Tag steht. Die PatchID ist hierbei ein zentraler Punkt der die Zugehörigkeit aller Dateien erkennen lässt und natürlich auch den Zeitpunkt an dem der Patch erstellt wurde. Die PatchID besteht aus Jahr, Monat und T ag, z .B.: "2011-05-23". Der Name der PatchXML darf diesem Muster nicht abweichen und in der XML Datei muss diese ID genau so auch angegeben werden.

Der Aufbau der PatchXML ähnelt sehr der Modul XML Dateien. Wer sich also damit schon etwas beschäftigt hat, wird sich hier schnell zurecht finden.

Am einfachsten ist es, ein vorhandene XML Datei zu nehmen und diese den eigenen Wünschen anzupassen.

Wenn die PatchXML Datei fertig ist, muss nur noch die `patch_scripts_create.x` gestartet werden und der Patch wird automatisch erstellt. Ein Beispielaufruf für einen Patch mit der ID "2011-05-23" (wenn der PatchOrdner in `/home/superx/svn_his/superx/trunk/superx/WEB-INF/patch liegt`) wäre:

```
cd /home/superx/svn_his/superx/trunk/superx/WEB-INF/conf/edustore/db/bin
patch_scripts_create.x  2011-05-23 /home/superx/  git  /superx/superx/WEB-INF/patch
```

3.4.1.1 Dokumentation

In jedem Element von files, masken und upgrade kann das Element patch-description hinzugefügt werden. Wenn dies der Fall ist, wird die Beschreibung zusammen mit dem Element, in dem die Beschreibung steht, in der Dokumentation des Patches erwähnt. Zusätzlich ist es möglich in der Beschreibung die FeautureID zu hinterlassen, um weitere Informationen zu der Änderung zu bekommen.

3.4.2 Patches einspielen

Ab Kernmodul 4. 1 gibt es eine komfortable Möglichkeit, Patches einzuspielen. Wichtige Vorbedi n gungen sind:

•Vorhandensein des Scriptes `$SUPERX_DIR/db/bin/patch_apply.x` (ab Kernmodul 4. 1). Wenn Sie das Kernmodul 4.0rc1 einsetzen, laden Sie die Datei herunter:
http://www.superx-projekt.de/dist/kernmodul40rc1_patch_apply_iso.zip (iso)
http://www.superx-projekt.de/dist/kernmodul40rc1_patch_apply_utf8.zip (utf8)
und entpacken Sie sie in `$SUPERX_DIR`

•Auf dem System muss das Programm "unzip" installiert sein

Wenn diese Voraussetzungen gegeben sind:

•Laden Sie die Patch-Datei, die zu Ihrem System paßt (SuperX oder HISinOne, iso oder utf-8-Codierung) , herunter, z.B. nach `$SUPERX_DIR/patches/patch_2011-06-15_superx_iso.zip`

•Gehen Sie in das Verzeichnis `$SUPERX_DIR/patches` , und geben Sie ein: `p atch_apply.x patch_2011-06-15_superx_iso.zip`

•Das Ergebnis wird in die Datei `patch_2011-06-15_superx_iso.zip.log` geloggt. Wenn Fehler auftreten, kommt direkt eine Meldung in der Shell.

3.5 dbforms

In der Modul-XML werden auch die Vorlagen für die DBforms gemacht. Dabei wird mit XSLT aus der abstrakten Beschreibung des Formulars eine funktionstüchtige JSP-Seite erzeugt. Hier werde ich nur auf die wichtigsten Felder eingehen.

3.5.1 Erläuterung der XML-Elemente

3.5.1.1 Gesamtstruktur

Als erstes sollte der Tabellenname eingetragen werden. Dieser wird hier an mehreren stellen benötigt. (Bei: "table", "path", "customfield", "table", "path" ..) Zu beachten ist das der Tabellenname sowie das Modul in den Pfad- ("path") Angaben auch richtig eingetragen wird.

Im ersten Block wird meist in den Feldern "name" und "caption" das selbe eingetragen und zwar eine kurze aber aussagekräftige Bezeichnung der Tabelle. Diese kann auch in dem zweiten Block übernommen werden. Dort gibt es genau die selben Felder.

Eine ausführlichere Beschreibung der Tabelle wird in das Feld "description" geschrieben. Diese Beschreibung sollte aber auch nicht länger sein als ein normaler Satz.

Wenn man <field-selection complete="true" /> wählt, dann werden automatisch alle Felder der Tabelle im Formular angezeigt. Wenn Sie nur einige Felder anzeigen wollen, nutzen Sie das Element <customfield>. Außerdem kann man mit customfield spezielle Layoutanweisungen kombinieren, z.B. Feldgröße.

```
<form name="Studienfächer (amtlich) zu Lehreinheiten (Liste)"
 table="gang_stg_astat"
 path="/edit/gang/gang_stg_astat_list.jsp"
 followUp=""
 caption="Studienfächer (amtlich) zu Lehreinheiten (Liste)"
 orderBy="astat"
 gotoHt=""
 helpfile=""
 maxRows="*"
 mode="update_insert">
<description>Zuordnung amtlicher Studienfächer einer Hochschule zu internen
Lehreinheiten.</description>
<filters>
</filters>
<field-selection  complete="false"  />
<customfield name="tid" nullFieldValue="" />
<customfield name="astat" type="label" nullFieldValue="" visibleSize="10"/>
<customfield type="link" name="Details"
path="/superx/edit/gang/gang_stg_astat_edit.jsp" linkid="tid"/>
</form>
```

```
<form name="Studienfächer (amtlich) zu Lehreinheiten"
 table="gang_stg_astat"
 path="/edit/gang/gang_stg_astat_edit.jsp"
 followUp=""
 caption="Studienfächer (amtlich) zu Lehreinheiten bearbeiten"
```

```
 orderBy="astat"
 gotoHt="tid"
 helpfile=""
 maxRows="1"
 mode="full">
<description>Zuordnung amtlicher Studienfächer einer Hochschule zu internen
Lehreinheiten.</description>
<filters>

</filters>
<field-selection  complete="false"  />
<customfield name="tid" nullFieldValue="" />
<customfield name="lehreinheit" nullFieldValue="" visibleSize="10"/>
<customfield name="beschreibung" nullFieldValue="" visibleSize="30"/>
<customfield name="astat" nullFieldValue="" visibleSize="100"/>
<customfield name="studienbereich" nullFieldValue="" visibleSize="30"/>
<customfield name="aktiv" nullFieldValue="" visibleSize="10"/>
<customfield name="bund_key" nullFieldValue="" visibleSize="30"/>
<customfield name="Datensatzstatus" type="include" path="/edit/gang/daten-
satzstatus.inc"/>
</form>
```

3.5.1.2 Filter

Mit dem Element `<filters> ... </filters>` kann man Parameter definieren, die in der URL zum DB-FORM übergeben werden sollen. Z.B. mit dem Passus

```
<filters>
<filter mandatory="true" type="equals">tid</filter>
</filters>
```

definiert man den Parameter "tid" als Pflichtfeld. Siehe Beispiel `<form name="zuordnung"` in der gang.xml.

3.5.2 Test der Formulare

Die Formulare werden in der Modul-XML-Datei definiert (s.o.). Die JSP-Seiten werden dann generiert, indem man das module_scripts_create ausführt.

Danach muss man einen Modulupgrade ausführen, damit die Formulare in der Datenbank gespeichert werden. Und dann kann man die Formulare testen, indem man im Menü Administration-> "Tabelle suchen"-> Bearbeiten anklickt.

Wenn Sie die Bearbeitungsformulare auch im Themenbaum anzeigen wollen, müssen Sie eine Suchmaske vorschalten, in der

- die Datensätze gefiltert werden können
- über einen Bearbeiten-Button zugänglich gemacht werden.

Sie z.B. die Maske "Tabelle suchen", diese ist ein gutes, einfaches Beispiel für eine Kombination aus Suchformular und Bearbeitungsformular.

4 Build der Java Quellen

Wenn Sie edustore-trunk aus Subversion in Ihren Tomcat (Verzeichnis webapps) ausgecheckt haben, gibt es im webapps-Verzeichnis drei neue Verzeichnisse:

src: Hier liegen die Java-Quellen für SuperX

superx: Hier liegt die SuperX-Webanwendung

superx-build: Hier liegt die build-xml, um aus den Java Quellen in src die superx<<Versionsnr>>.jar in WEB-INF/lib zu erzeugen, und um das superx.war-Archiv zu erzeugen. Auch die Dateien für den Build des SuperX-Applet liegen hier.

4.1 Umgebung für ANT

Um die build.xml korrekt nutzen zu können, müssen Sie die Umgebung für ANT einrichten.

•Gehen Sie in der Shell in das Verzeichnis superx-build

•Wenn Sie planen, die fertigen JAR-Dateien an Kunden auszuliefern: Um abwärtskompatibel zu kompilieren, sollte ANT mit einer Java-1.4-Runtime gestartet werden. Setzen Sie ggf. die Umgebungsvariable JAVA_HOME und JAVACMD auf eine Java 1.4-Installation, z.B.

```
PATH=/usr/java/j2sdk1.4.1_01/bin:$PATH
export PATH
JAVA_HOME=/usr/java/j2sdk1.4.1_01
export JAVA_HOME
```

4.2 Build des SuperX-Servlets

Um einen build des Servlets zu erzeugen, gehen Sie wie folgt vor:

- rufen Sie auf:

ant distServer

- Wenn eine Fehlermeldung kommt: "package javax.servlet.http does not exist", dann sind Sie ggf. im falschen Verzeichnis. ANT sucht relativ zum aktuellen Verzeichnis in "../../common/lib" (Tomcat 5.5) oder "../../lib" (Tomcat 6) nach der Datei servlet-api.jar, diese wird vom Java-Compiler benötigt.

Nach einem build der superx<<Versionsnr>>.jar sollten Sie mit dem Target

```
ant cleanBuildPath
```

die *.class-Dateien im Verzeichnis WEB-INF/classes löschen, und Tomcat neu starten. Bitte committen Sie **keinesfalls** *.class-Dateien in das SVN.

4.2.1.1Build des SuperX-Applet

Der ANT-Target **distApplet** erzeugt einen build des SuperX-Applet. Die fertige JAR-Datei lautet webapps/superx/applet/superx.jar. Um sie möglichst klein zu halten, enhält sie nur die benötigten Klassen aus dem SuperX-Projekt sowie benötigte Libraries von Fremdherstellern (JavaHelp-lassen, Freemarker, Log4J). Außerdem wird sie nach dem build signiert, um die Druckfunktion für Clients zu aktivieren. Um den Build ausführen zu können, gehen Sie wie folgt vor:

(einmalig:) Erzeugen Sie ein Zertifikat z.B. mit den Befehlen

```
keytool -genkey -alias superx_applet -keyalg RSA
keytool -selfcert -alias superx_applet -validity 365
```

Hiermit wird ein Zertifikat provisorisch zertifiziert für 365 Tage. Als Passwort geben Sie z.B. "changeit" an. Wenn Sie ein anderes Passwort eintragen, müssen Sie es in der build.xml ändern. Der Passus in der build.xml lautet

```
<signjar jar="${APPLET_DIR}/superx.jar" alias="superx_applet"
storepass="changeit" />
```

Danach geben Sie ein

ant distApplet

Damit werden die Quellen nach `WEB-INF/classes` kompiliert, und dann werden die nicht benötigten Dateien entfernt. Dazu wird das Tool Proguard Obfuscator genutzt (zugehörige Dateien befinden sich in superx-build). Wenn die superx.jar erzeugt wurde, werden alle Dateien unterhalb von WEB-INF/classes gelöscht.

5 Nötige Änderung bei Upgrade

5.1 auf Version 4.1

5.1.1 Änderungen an vorhandenen speziellen XSL-Stylesheets

Wenn dort in der Ergebnistabelle einTreeTable dargestellt wird (Aufklappfunktionzeilen) und nicht das Standardexportmenü aus pageComponents, muss man in dem Exportmenü
document.forms[0].stylesheet='';....document.forms[0].submit();
ändern in
prepareExport('excel'); pdf/druckversion
wenn man den Auswahldialog alle Daten/nur sichtbare Zabellenzeilen dort auch haben möchte.
Wenn man dies nicht macht, werden wie bisher immer alle Zeilen exportiert.

Hinweis:
in tabelle_fo_pdf.xsl:
Standard Header/Footer Definitionen ausgelagert nach tabelle_fo_pdf_kopffusszeile.xsl

Bei speziellen PDF-Stylesheets die noch auf der alten tabelle_fo_pdf.xsl basieren, funktioniert unterschiedliche Fußzeilen für 1./weitere Seiten noch nicht.

nötige Korrekturen im Stylesheet
in

```
 <fo:simple-page-master master-name="first">
<fo:region-before extent="40mm" region-name="first-region-before"/>
<fo:region-after extent="10mm"/>
```

bei region-after muss ein Name eingetragen sein,
in neuer Version per template rest-region-after

bei

```
 <fo:simple-page-master master-name="rest">
  <fo:region-before extent="10mm"/>
<fo:region-after extent="10mm"/>
```

ebenfalls Namen eintragen, jetzt per template

```
 <fo:region-before extent="10mm" region-name="rest-region-before"/>
  <fo:region-after extent="10mm" region-name="rest-region-after"/>
```

unter

```
 <xsl:template name="pdfpage">
<fo:page-sequence master-name="sequence" master-reference="sequence">
<!-- Anzeige: Aktuelle Seite / Seitenanzahl -->
FALSCH gilt für alle Seiten
<fo:static-content flow-name="xsl-region-after">
 <fo:block>
<fo:inline align="left" font-size="8pt" space-end="224mm">
Erzeugungsdatum: <xsl:value-of select="/ergebnisse/@datum" />
</fo:inline>
<fo:inline align="right" font-size="8pt">
<fo:page-number />/<fo:page-number-citation ref-id="endofdoc" />
</fo:inline>
</fo:block>
</fo:static-content>

<fo:static-content flow-name="first-region-before">
```

neu:

```
 <xsl:template name="pdfpage">
<fo:page-sequence master-name="sequence" master-reference="sequence">
<fo:static-content flow-name="first-region-before"><xsl:call-template
name="first_page_header"/> </fo:static-content>
<fo:static-content flow-name="first-region-after"> <xsl:call-template
name="first_page_footer"/> </fo:static-content>
```

```
<fo:static-content flow-name="rest-region-before"><xsl:call-template
name="rest_page_header"/> </fo:static-content>
<fo:static-content flow-name="rest-region-after"> <xsl:call-template
name="rest_page_footer"/> </fo:static-content>
```