Inhaltsverzeichnis

Masken- und Tabellengestaltung

Vorbemerkung SuperX/HISinOne-BI

Vorbemerkung:
Die im folgenden vorgestellten Techniken funktionieren in SuperX und in der HISinOne-BI in der Oberfläche "Business Intelligence"-> "Standardberichte" (ggf. mit dem Zusatz "(konfigurieren)". Nicht alle der im Folgenden dargestellten Möglichkeiten funktionieren in der Angular Oberfläche von HISinOne-BI.

Selektionsmasken anpassen

Checkboxen und Querabhängigkeiten

Checkboxen können mit Feldart 10 definiert werden. Bei einfachen Masken, kann felderinfo.relation leer bleiben und in felderinfo.default auch ein fester Wert stehen wie true. Wenn es auf der Maske 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
```

und im Defaultwert:

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

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(..,"<>",....) 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.

Felder zum Verarbeiten von Eingaben per Javascript

Vorbemerkung:
Diese Auswertung wird mit Release Kernmodul 5.1 bzw. HISinOne-BI 2024.06 ausgeliefert.

Ab können Sie vor dem Abschicken einer Maske benutzerdefiniterte Scripte ausführen, die z.B.

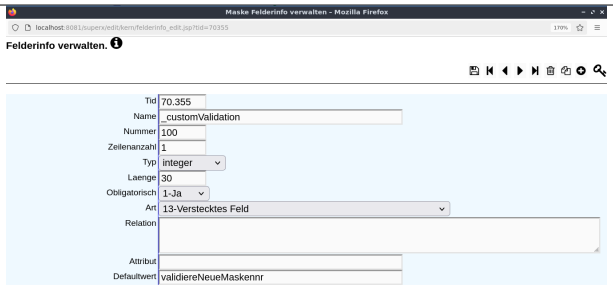
- bei Feldinhalt a von Feld x das Feld y mit dem Inhalt b befüllen
  - Beispiel fürs Land Sachsen: wenn im Feld "Bericht" der Wert "Stellenbewirtschaftungsbericht (Excel)" ausgewählt wird, wird
    - das Feld "tablestylesheet" mit dem entsprechenden JasperReport befüllt
    - das Feld "Ausgabeformat" mit dem Wert für Excel befüllt.
- Es gibt noch viele andere Szenarien, z.B. können bestimmte Eingaben geprüft und, bei Warnungen, entsprechende Meldungen ausgegeben werden, und das Abschicken wird verhindert.

Technisch ist das wie folgt umgesetzt: Wenn eine Maske ein Feld mit dem Namen "_customValidation" hat, wird

- der defaultwert des Feldes ausgewertet, dieser enthält den Namene einer Javascript-Methode
- diese Methode wird *vor* dem Abschicken einer Maske ausgeführt und je nachdem ob "true" oder "false" zurückgeliefert wird, wird die Maske tatsächlich abgeschickt oder nicht.

Hier ein Beispiel:

- Die Maske "Maske kopieren" ermöglicht es, Maskenkopien anzulegen. Die Zielnummer sollte nach den Konventionen von SuperX in einem 10er oder 20er Nummernkreis bestehen. Wenn eine andere Zahl eingegeben wird, sollte eine Warnung erscheinen.
- das Maskenfeld sieht so aus:



- Im Maskenfeld befindet sich eine Referenz auf die Javascript-Funktion *validiereNeueMaskennr*. Dieser Funktion wird vom Server immer der Parameter des Formulars und der Plattform (superx oder HISinOne-BI) übergeben. Hier der Quellcode:

```
function validiereNeueMaskennr(frm
```

- Im Ergebnis passiert nun folgendes, wenn man eine "ungerade" Maskennummer als "Neue Tid" eingibt und abschickt:

Es erscheint eine Warnung, und die Maske wird nicht abgeschickt.

Felder für Benutzergruppen verstecken

Es ist möglich, auf einzelnen Masken Felder für einzelne Gruppen zu verstecken. Dies dient z. B. dazu, das Feld "Hörerstatus" für die Gruppe "Externe Anwender" auszublenden, d.h. es würde immer "Alle" selektiert. Um dies zu realisieren geht man wie folgt vor:

1. Ermitteln Sie zunächst die Feldnummer unter Administration->Masken -> Felder -> Feld suchen.
2. Nutzen Sie die Selektionsparameter, um das gewünschte Feld zu suchen.
3. Klicken Sie auf "Abschicken". Die Nummer steht in der Spalte "tid".
4. Öffnen Sie Administration-> Tabelle suchen
5. Geben Sie als Stichwort "pref" ein.
6. Es erscheinen ein Listen- und ein Detailformular für die Tabelle.
7. Klicken Sie auf "Bearbeiten". Es öffnet sich ein Detailformular.
8. Wählen Sie die gewünschte "Gruppe" sowie das "Feld" aus.
9. Wählen Sie unter Voreinstellung "Versteckt", damit wird das Feld für die jeweilige Gruppe ausgeblendet.
10. Klicken Sie auf "Neuen Datensatz einfügen" bzw. "Speichern" .
11. Nach dem Speichern können Sie diesen Datensatz "kopieren" oder einen "Neuen Datensatz anlegen" . Auch ein Blättern durch die angelegten Datensätze ist nun möglich.

Anpassung von Feld-Vorbelegungen

Manche Masken haben Vorbelegungen, die bei der Hochschule nicht passen, z.B. Hörerstatus "Alle" im Bereich Studierende. Die in den jeweiligen Komponenten ausgelieferten Masken lassen sich zwar ändern, aber beim Einspielen eines neuen Releases würden diese überschrieben. Um dies zu vermeiden gibt es zwei Wege:

- Sie kopieren die Maske in einen eigenen Nummernkreis, dann ist sie vor Upgrades "geschützt".
- Sie ändern die Maske, und führen danach eine Customize-Regel ein, die nach jedem Upgrade ausgeführt wird.

Beide Varianten haben Vor- und Nachteile. Die erste Variante ist besser, wenn Sie nur eine Maske ändern wollen, und ggf. auch noch andere Layouts (z.B. auch Spaltenlayouts der Ergebnistabelle) ändern wollen. Die zweite Variante ist besser, wenn Sie auf einen Schlag mehrere Masken bzgl. einer Kleinigkeit ändern wollen. Das obige Beispiel "Hörerstatus" wäre also besser mit der zweiten Variante lösbar, weil es das Feld in vielen Masken gibt. Hier ein Beispiel wie man das macht:

Ändern Sie die Feldvorbelegung über die Maske Administration-> Masken -> Felder -> "Feld suchen", z.B. beim Feld "Hörerstatus" in der Maske "Studierende und Studienanfänger (Zeitreihe)":

Bisheriger Wert in Spalte Defaultwert:

```
<<SQL>> select apnr, eintrag from hoererstatus
```

Neuer Wert:

```
<<SQL>> select apnr, eintrag from hoererstatus
```

Speichern Sie die Änderung, und testen Sie die Maske. Wenn das Ergebnis Sie zufrieden stellt, können Sie es wie folgt vor Änderungen durch Releases schützen:

Erzeugen Sie eine Datei $SOS_PFAD/conf/customize.sql

und schreiben Sie den Inhalt hinein:

```
--Änderung  xx.xx.xxxx
Maskenvorbelegung  von
```

Damit wird diese Änderung nach jedem Upgrade ausgeführt. Sie können die Änderung auch direkt für alle Masken vornehmen, indem Sie die Where-Bedingung "and tid=16004" entfernen.

Hier noch ein Beispiel für das Feld "Status": wir ändern den Default von "Alle ohne Beurl." nach "Alle ohne Beurl., ohne Exmatr.":

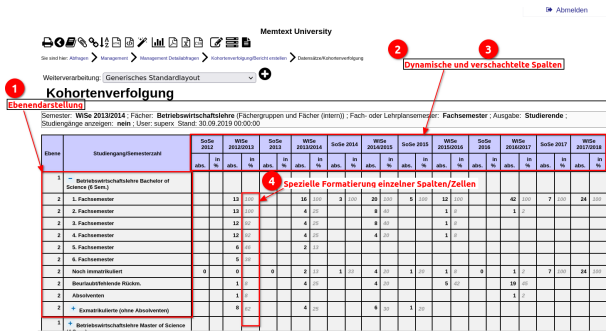
```
--Änderung  xx.xx.xxxx:
Standardwert  für  Feld
```

Ergebnistabellen anpassen

Vorschau: Erweiterte Ergebnistabellen

Im Masken-Tutorial wird erläutert wie Quellcodes für Masken erstellt werden. Damit werden grundlegende Tabellenlayouts erzeugt. Darüber hinaus gibt es viele weitere Möglichkeiten, diese Layouts anzupassen.

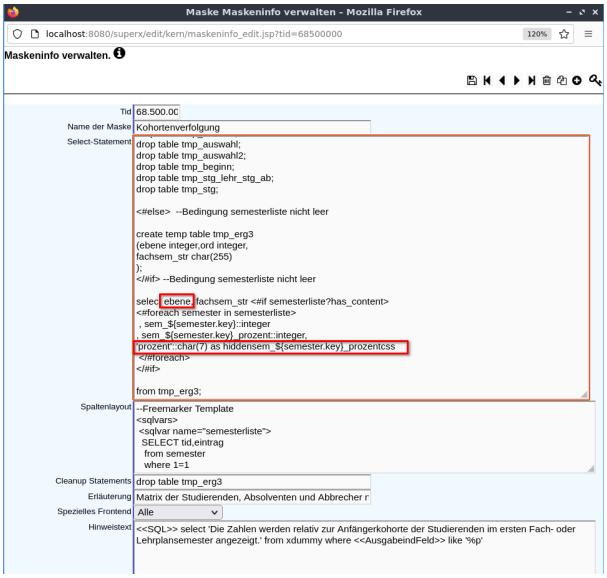
Die folgende Abbildung zeigt, was jenseits von "normalen" Ergebnistabellen so möglich ist, am Beispiel einer Tabelle aus dem Managementmodul:



Sie sehen

1. Eine Baumdarstellung inkl. Autklappmöglichkeit für die Zeilen
2. Dynamische Spalten z.B. bei Semesterzeitreihen
3. Verschachtelte Spaltenüberschriften
4. Einzelne Zellen werden mit CSS formatiert

Die folgende Abbildung zeigt den Quellcode, im letzten SELECT kann das Layout gesteuert werden.

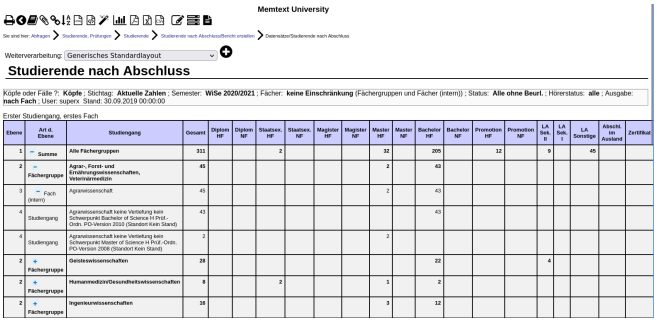


Dies wird unten näher erläutert. Aber dies ist nur ein Teil der Möglichkeiten, die es hier gibt. Sie können auch Hyperlinks in die Tabellenzellen legen, oder kleine Balkengrafiken einbauen.

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:



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.

Die Baumdarstellung lässt sich auch dynamisch mit Sichtenzugriff erstellen. Dazu kann mit .elements auf Sichten zugegriffen werden. Für Studiengangssichten also Studiengang.elements:

```
create table tmp_aggr (
Ebene integer,
strukturStr varchar(255),
name varchar (255),
key varchar(255),
anzahl integer,
nextsem integer,
anteil float,
inzweisem integer,
anteilzwei float,
abschlussarbeit integer
);

<#foreach einElement in Studiengang.elements<#foreach einElement in Studiengang.elements>
insert into tmp_aggr(Ebene,strukturStr, name, key,anzahl,nextsem,anteil,inzweisem,anteilzwei,abschlussarbeit)
select
${einElement.level}::smallint,
'${einElement.strukturStr}'::char(50),
'${einElement.name}'::char(200),
'${einElement.key}'::char(10),
count(E.matrikel_nr),
sum(E.nextsem),
(sum(E.nextsem)*100/count(E.matrikel_nr))::integer,
sum(E.inzweisem),
(sum(E.inzweisem)*100/count(E.matrikel_nr))::integer,
sum(E.abschlussarbeit)
from tmp_ergebnis E
where E.tid in ${einElement.subkeys}
order by 1,2,3,4
;
```

Aufruf eines Stylesheets

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.).

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
```

und zum Beispiel für Defaultwert:

```
<<SQL>> select distinct
```

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
```

Beispiel für Excel als defaultwert:

```
<<SQL>> select
element_value,description
```

Navigationsspalten

Eigene Navigationsspalten zum Anklicken

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 Maske übergeben, dabei kann bei mehreren Spalten Nummern angehängt werden.

SQL für vorherigen Screenshot

```
select  benutzer,name,email,nexttable,  nextedit,nextedit2  from  tmp_ergebnis;
```

- **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://hostname/superx/servlet/SuperXmlTabelle?tid=20010&id=2044

- Die Ergebnisseite wird dann um einen Button ergänzt.
- **nextwindowtable**: 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://hostname/superx/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: Auf der Ergbnisseite erscheint ein Delete-Button.
- **nextedit**: Link auf ein DBForms-Formular relativ zur URL des Servlets. Auf der Ergbnisseite erscheint ein Bearbeiten-Button.
- **nextmail**: Feldinhalte werden um einen Mailto-Tag ergänzt.
  - z.B. info@superx-projekt.de wird zu info@superx-projekt.de

- **url**: Feldinhalte werden um einen href-Tag, sowie wenn nötig um ein "http" ergänzt.
  - z.B. www.superx-projekt.de wird zu www.superx-projekt.de

- **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 Link durch ein Trennzeichen "|" getrennt sind.
  - So wird z.B. der Feldwert "SuperX-Projekt|http://www.superx-projekt.de" wie folgt ersetzt: SuperX-Projekt

- 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 Details zur Hochschule

- **nextgenericlink**: für generische Links
- **nextwindowgenericlink** :wie nextgenericlink nur mit Target _blank, Zusatzfunktion (ab HisInOne 2023.06) ein img am Anfang stellt eine Grafik für den Link da , z.B. "img:/superx/images/dms_abruf.svg|DocMan?...."
- **nextserverlink**:Link auf eine andere Seite auf dem aktuellen Server
- **nextdokulink**:Link auf eine integrierte Wiki-Seite (nur HISinOne-BI)

Ergebniszellen anklickbar machen

Statt eigener Navigationsspalten ist es auch möglich einzelne Zahlen bzw. Zellen in einem Bericht selbst anklickbar zu machen.

Dies kann z.B. genutzt werden, um per "DrillDown"

- von Kontoständen (wie 80.000 Euro) zu den zugehörigen Buchungen zu kommen,
- sich bei einer Summe von Studierenden die zugehörigen Matrikelnummern ausgeben zu lassen oder
- bei einem Flächenbericht mit Quadratmetern sich beim Anklicken der Quatdratmeterzahl die zugehörigen Räume anzeigen zu lassen.

Beispiel:

In einem einfachen Flächenbericht gibt es eine Ergebnisspalte "flaeche", die anklickbar gemacht werden soll und wenn eine Zahl angeklickt wird, soll ein Detailbericht mit den Räumen dazu (10250) aufgerufen werden.

Für die Ergebnisspalte wie

```
flaeche  decimal(14,2)
```

muss es zusätzliches Feld in Ergebnistabelle geben

```
hidden_flaeche  text
```

-> durch das hidden wird erkannt, dass die Spalte selbst nicht dargestellt werden sol und auch nicht nach Excel/PDF exportiert werden soll , _flaeche sorgt für die Zuordnung zu Ergebnisspalte.

Dann beim insert z.B.

```
insert into tmp_erg (kostenstelle,hidden_flaeche,flaeche)
select kostenstelle,'?tid=10250&Kostenstelle='||kostenstelle||'&cachingcontrol=clearmask',sum(flaeche)
from tmp_rohdaten group by 1,2;
```

Heißt: Für jede Ergebniszeile wird die Kostenstellennummer dargestellt und der Wert "flaeche" wird anklickbar gemacht (hidden_flaeche). Wenn die Flächenzahl in einer Ergebniszeile angeklickt wird, wird der Bericht 10250 mit dem Parameter z.B. Kostenstelle=1000 in einem neuen Tab aufgerufen.

Cachingcontrol=clearmask

sorgt dafür, dass die Maske neue geladen wird mit den Defaulteinstellungen und keine vorher auf anderen Masken getätigte Auswahl (z.B. Raumnutzungsart=500) eingetragen wird.

Aufpassen muss man noch, wenn man zusätzliche Felder auf der Maske hat.

z.B: Flächenart.

Wenn der User sich nicht die Gesamtfläche ausgeben lässt, sondern eine Einschränkung auf eine oder mehrere Flächenarten gemacht habt, muss diese Einschränkung ja auch an den Detailbericht mit übergeben werden.

Im einfachsten Fall bei obligatorischen numerischen Feld mit einfacher Auswahl - da ja immer etwas ausgewählt sein muss, wird der Tag <> auf jeden Fall ersetzt.

```
insert into tmp_erg (kostenstelle,hidden_flaeche,flaeche)
select kostenstelle,'?tid=10250&Kostenstelle='||kostenstelle||'&Flächenart=<>&cachingcontrol=clearmask',sum(flaeche)

from tmp_rohdaten group by 1,2;
```

Ist bei Flächenart eine Mehrfachauswahl oder ist nicht numerisch möglich, muss man evtl machen

```
insert into tmp_erg (kostenstelle,hidden_flaeche,flaeche)
select kostenstelle,'?tid=10250&Kostenstelle='||kostenstelle||'&Flächenart=${Flächenart?replace("|",",")}&cachingcontrol=clearmask',sum(flaeche)
from tmp_rohdaten group by 1,2;
```

Die Variable ${Flächenart} wird durch die Auswahl des Users gefüllt, dabei werden Pipezeichen bei Mehrfachauswahl durch das Replace mit Kommas ersetzt.

Wenn das Feld nicht obligatorisch ist oder für Gesamtsummenzeile nicht berücksichtigt werden soll, muss man etwas mit freemarker tricksen.

Man erstellt mit assign eine leere Freemarker-Variable für den Flächenartparameter (flart=""), wenn etwas ausgewählt ist, wird die Variable mit Inhalt gefüllt, z:B. &Flächenart=NF1,NF2 - für die zwei ausgewählten Flächenarten NF1 und NF2. In den Link wird dann der Inhalt der FreemarkerVariable mit ${flart} eingebaut.

```
<#assign flart=""/>
-- Wenn der User eine
```

Eine Beispielmaske (mit noch mehr Freemarker Tricks) ist der Bericht 10320 "Flächen nach Kostenstellen-Flächenarten-Gebäuden (Zeitreihe)"

Rechteverwaltung für die Detailmaske

Wenn abgefragt werden soll, ob ein User Rechte für die Untermaske hat und nur in dem Fall die Navigationsspalte angezeigt werden soll oder die Zahl anklickbar gemacht werden soll, kann man dies mit einer Freemarkerfunktion machen

z.B. im letzten Select-Statement

```
select kostenstelle,name   <#if UserHasMaskRight(23230)<#if UserHasMaskRight(23230)> , nextedit   from tmp_erg
```

analog in der XIL-Proplist.

Oder bei einer "Hidden"-spalte, bleibt diese leer außer wenn der Anwender Rechte für den Unterbericht hat

```
  <#if
UserHasMaskRight(23230)
```

Standardabfragen mit hochschulspezifischen Details versehen

Bei Standardabfragen, bei denen nur Kleinigkeiten hochschulspezifisch angepasst werden sollen, wird ein Block eingebaut, z.B.

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

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

Dies ist erweiterbar, indem ein Repository-Objekt CUSTOM_xxxxxx (Maskennummer) angelegt wird. Beispiel: FIN-Abfrage zeigt standardmäßig Festlegungen immer an, aber einige Hochschulen wollen Festlegungen nur für aktuelles Haushaltsjahr. Es wird in CUSTOM_xxxxxx eine Variable definiert:

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

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

```
<#assign FestlegungenNurAktuellesJahr=false/<#assign FestlegungenNurAktuellesJahr=false/>
<#if CUSTOM_xxxxxx?exists<#if CUSTOM_xxxxxx?exists>
<#assign inlineTemplate=CUSTOM_xxxxxx?interpret<#assign inlineTemplate=CUSTOM_xxxxxx?interpret>
<@inlineTemplate/>

-- ausführen der Definition überschreibt default-wert von FestlegungenNurAktuellesJahr   mit <#assign FestlegungenNurAktuellesJahr=true/<#assign FestlegungenNurAktuellesJahr=true/>


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

you#re welcome!

Spaltenlayout in Ergebnistabellen

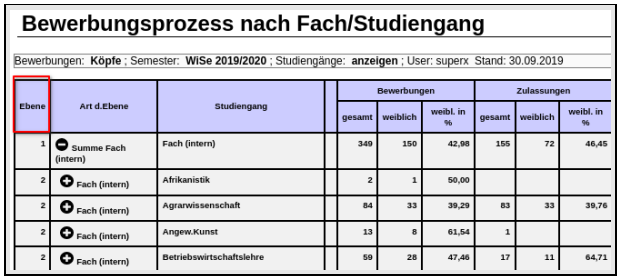
Wie im Tutorial gezeigt, wird das Spaltenlayout (Überschriften, Breite) in der sog. "xil_proplist" gesteuert (der Name stammt übrigens vom früher im SuperX-Windows-Client eingesetzten XVT-Compiler zur Layoutdarstellung). Das Format ist etwas eigenwillig und soll hier erläutert werden.

Die Attribute in der xil_proplist

Zunächst ein Beispiel: der Code für die Maske "Bewerbungsprozess nach Fach/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:



Bewerbungsprozess nach Fach/Studiengang

Bewerbungen: **Köpfe** ; Semester: **WiSe 2019/2020** ; Studiengänge: **anzeigen** ; User: superx Stand: 30.09.2019

Ebene	Art d.Ebene	Studiengang	Bewerbungen			Zulassungen		
			gesamt	weiblich	weibl. in %	gesamt	weiblich	weibl. in %
1	Summe Fach (intern)	Fach (intern)	349	150	42,98	155	72	46,45
2	Fach (intern)	Afrikanistik	2	1	50,00			
2	Fach (intern)	Agrarwissenschaft	84	33	39,29	83	33	39,76
2	Fach (intern)	Angew.Kunst	13	8	61,54	1		
2	Fach (intern)	Betriebswirtschaftslehre	59	28	47,46	17	11	64,71

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:

- **fixed_columns**: Das Attribut im Kopf steuert wie viele führende Spalten beim Scrollen nach rechts fixiert bleiben.
- **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.

Fixierte Spalten

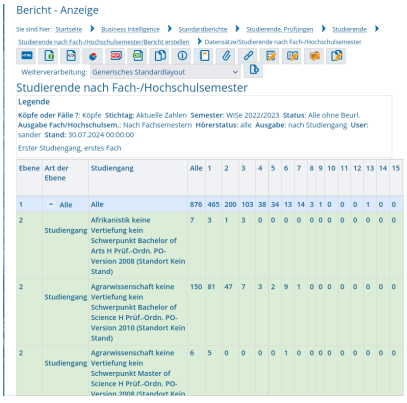
Bei breiten Tabellen ist es ggf. sinnvoll, die ersten X Spalten zu fixieren, damit sie beim Scrollen nach rechts stehen bleiben. Dies definieren Sie über

- eine Konstante "fixed_columns_aktiv", mit der Sie das Fixieren generell einschalten
- im Kopf der Spaltenbeschreibung (xil_proplist) mit dem Attribut "fixed_columns".

Beispiel:

```
XIL_List
sizable_columns horizontal_scrolling
drop_and_delete movable_columns
white_space_color=COLOR_WHITE fixed_columns=3
min_heading_height=40
```

Bei einer Ergebnistabelle in schmalem Viewport sieht die Originaltabelle so aus:



Bericht - Anzeige

Studierende nach Fach-/Hochschulsemester

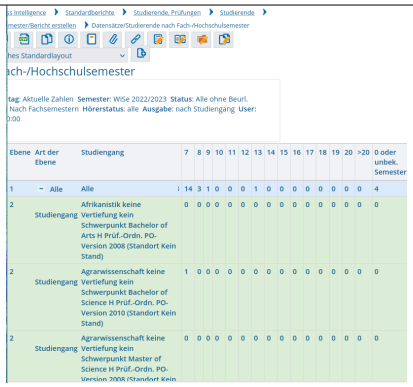
Legende

Köpfe oder Fälle ?: Köpfe Stichtag: Aktuelle Zahlen Semester: WiSe 2022/2023 Status: Alle ohne Beurl. Ausgabe Fach/Hochschulsem.: Nach Fachsemestern Hörerstatus: alle Ausgabe: nach Studiengang User: sander Stand: 30.07.2024 00:00:00

Erster Studiengang, erstes Fach

Ebene	Art der Ebene	Studiengang	Alle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	- Alle	Alle	876	465	200	103	38	34	13	14	3	1	0	0	0	1	0	0
2	Studiengang	Afrikanistik keine Vertiefung kein Schwerpunkt Bachelor of Arts H Prüf.-Ordn. PO-Version 2008 (Standort Kein Stand)	7	3	1	3	0	0	0	0	0	0	0	0	0	0	0	0
2	Studiengang	Agrarwissenschaft keine Vertiefung kein Schwerpunkt Bachelor of Science H Prüf.-Ordn. PO-Version 2010 (Standort Kein Stand)	150	81	47	7	3	2	9	1	0	0	0	0	0	0	0	0
2	Studiengang	Agrarwissenschaft keine Vertiefung kein Schwerpunkt Master of Science H Prüf.-Ordn. PO-Version 2008 (Standort Kein	6	5	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bei einer Ergebnistabelle in schmalem Viewport und Scrollen nach rechts sieht es das dann so aus:



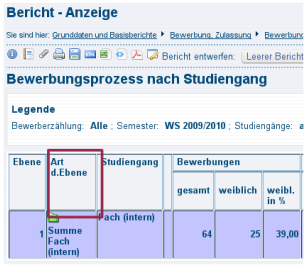
Mehrzeilige Spaltenüberschriften

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:



Verknüpfte Spaltenüberschriften

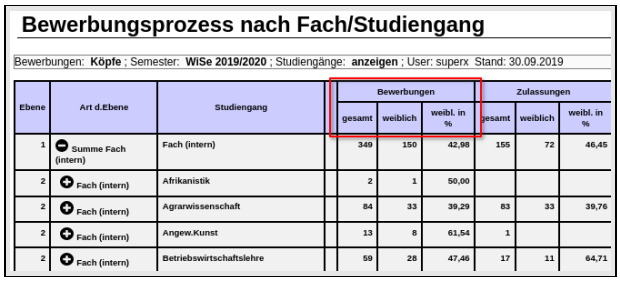
Um Spaltenüberschriften zu verknüpfen, muss man wie folgt vorgehen:

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\n weibl. in %" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=10 explanation=""
```

sieht in der Ausgabe so aus:



Dynamische Spaltenanzahl

Kann mit Freemarker realisiert werden, Einfaches Beispiel:

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

Abschluss-Select im Masken-SQL:

```
select name,<#if K_FIN_Quellsystem=1<#if K_FIN_Quellsystem=1> hhans, 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<#if K_FIN_Quellsystem=1>
Column CID=4 heading_text="Ansatz" 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
Column CID=4 heading_text="Einnahmen" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=10
```

Ein weiteres Beispiel:

Spalte Bewilligung soll nur angezeigt werden, wenn werte größer 0

Im Masken-SQL:

```
...




...
```

Die im Masken-SQL definierten sqlvars (in diesem Falle bewilligungen) sind auch in der XIL-Proplist zur verfügbar.

Abschluss-Select im Masken-SQL:

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

XIL:

```
Column CID=4 heading_text="Name" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=10
<#if bewilligungen><#if bewilligungen>>
Column CID=4 heading_text="bewill" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=10


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
```

Hochschulspezifische Einstellmöglichkeit

Ein Bericht sollte nur so viel Spalten haben wie maximal benötigt werden. In der Standardauslieferung werden alle Spalten gezeigt. Ein Beispiel für ein Standard-Schluss-select:

```
select name, akt_soll,einnahmen,aus,fest,verfuegbar from tmp_erg
```

Wenn eine Hochschule nicht alle Spalten oder die Spalten in einer anderen Reihenfolge sehen möchte, wird in x_repository eine Variable CUSTOM_XXXXXX (Maskennummer) angelegt, welche einen customize-Hash und in diesem die Felder, welche angezeigt werden sollen, definiert. field ist Feldname in der tmp_erg-tabelle, caption Spaltenüberschrift, width für xil-proplist und bei Bedarf kann auch explanation ergänzt werden:

```
<#assign customize={"resulttable":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@@@"}}/>
```

Dies wäre die gleich Ausgabe wie im Standard.

Nach Eingabe/Änderung den Manager-Chache leeren

Möchte eine Hochschule beispielsweise verfügbar weiter vorne stehen haben und akt_soll gar nicht sehen, wird der Eintrag folgendermaßen geändert:

```
<#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}
]}/>
```

Im Masken-sql gegen Ende baut man ein

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

Falls eine Hochschule ein custom_xxxxxx angelegt hat, wird der Inhalt interpretiert und ein customize-Hash steht zur Verfügung.Das Abschluss-select prüft also

```
<#if customize?exists<#if customize?exists>
select
<#foreach f in customize.resulttable<#foreach f in customize.resulttable>
${f.field} <#if h_has_next<#if h_has_next>,



from tmp_erg2 ;

<#else> -- kein customize objekt existiert, standard abschluss select

select name,akt_soll,einnahmen,ausgaben,fest,verfügbar from tmp_erg;
```

Für die XIL-Proplist muss es genauso laufen:

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



<#if customize?exists<#if customize?exists>
<#foreach f in customize.resulttable<#foreach f in customize.resulttable>
Column CID=0 heading_text="${f.caption}" center_heading explanation="<#if f.explanation?exists<#if f.explanation?exists>${f.explanation}"

row_selectable col_selectable heading_platform readonly width=${f.width}

<#else> --standard xil list

Column CID=0 heading_text="Name explanation="" center_heading row_selectable col_selectable heading_platform readonly width=9 text_size=0
Column CID=0 heading_text="aktsoll explanation="" center_heading row_selectable col_selectable
..
```

complex, but coool,

for the super nerds,

Technik kann man sogar mit eigenen Freemarker-Funktionen verbinden, z.B. dynamsiche Spalten nach customizing und Link-Spalten zur Einzelbuchung nur anzeigen, wenn Rechte für Einzelbuchungen da sind. Aus dem Kontext, XIL:

```
<#function isWanted field<#function isWanted field>
<#assign result=true<#assign result=true>
<#if field?starts_with('linkbuch')&&Einzelbuchrecht?exists&&Einzelbuchrecht?is_number&&Einzelbuchrecht=0<#if field?starts_with('linkbuch')&&Einzelbuchrecht?exists&&Einzelbuchrecht?is_number&&Einzelbuchrecht=0><#assign result=false/

<#if (field='einnahmen'||field?starts_with('linkbuchein')||field='offsoll_e'||field?starts_with('linkbuchoffsolle'))&&"<>"=""nein"">

<#assign result=false/<#assign result=false/>

<#return result<#return result>
<#if customize?exists<#if customize?exists>
<#foreach f in customize.resulttable<#foreach f in customize.resulttable>
<#if isWanted(f.field)<#if isWanted(f.field)>
Column CID=2 heading_text="${f.caption}" center_heading explanation="<#if f.explanation?exists<#if f.explanation?exists>${f.explanation}"

row_selectable col_selectable heading_platform readonly width=${f.width}



<#else>

...
```

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):

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

bewirkt, dass die Spalte plan_soll einstellig dargestellt wird.

Dies klappt nicht bei verschachtelten Spaltenlayouts

Einzelne Zellen oder Spalten formatieren (CSS)

Wenn im Masken-Resultset zum Beispiel eine Spalte "verfuegbar" existiert, kann über eine zusätzliche Spalte nach dem Muster

```
hidden - Spaltenname - css
```

Also hier z.B. eine versteckte Spalte **hiddenverfuegbarcss** mit dem Inhalt des css-Klassennamens, z.B. neu in kern44 *td.boldnumber*:

Auszug

```
create temp table tmp_erg
,
```

Eine entsprechende XIL_proplist-Definition der "column" muss es auch geben, hier ist der Name allerdings beliebig, sollte aber eindeutlig sein, d.h. eine Tabelle sollte nicht zwei gleiche Spaltenüberschriften haben.

Detailansicht verlinken

Es ist möglich die Werte einer bestimmten Spalte als Links auszugeben, sodass mit einem Klick auf diesen weitere Details in einer zweiten Ergebnistabelle gezeigt werden. Dabei wird über Klick auf diesen Link eine zweite Maske aufgerufen, die Maskenfelder dieser Maske mit den übergebenen Parametern gefüllt und die Ergebnistabelle dieser Maske ausgegeben.

Im Folgenden eine Erläuterung anhand eines Beispiels mit Prüfungsanmeldungen. Die Tabelle listet pro Modul die Anzahl der Anmeldung und in einer weiteren Spalte die Anzahl der Studierenden, welche zu ihrem dritten Prüfungsversuch antreten. Die Werte der letzten Spalte mit den Drittversuchen wird verlinkt:

Nach Klick auf einen der Links erscheint eine weitere Tabelle, welche statt der Anzahl die einzelnen Matrikelnummern inklusive Studiengang ausgibt:

In diesem Falle fungieren die Werte der Spalte drittversuch als Link. Um dies umzusetzen geben wir eine weitere Spalte hidden_drittversuch aus. Durch hidden wird diese nicht dargestellt und _drittversuch dient als Referenz auf die Spalte drittversuch. Die Spalte hidden_drittversuch enthält den Teil der URL, welcher die Parameter beim Maskenaufruf definiert. Das sieht beispielsweise so aus:

```
'?tid=30610360&Prüfungskurztext='||pktxt||'&&cachingcontrol=clearmask' as hidden_drittversuch
```

'||pktxt||' wird ersetzt durch den Wert des Feldes pktxt derselben Zeile:

```
select
pktxt,
pdtxt,
(case when pstatus='AN' then 1 else 0 end) as anmeldung,
(case when pversuch=3 then 1 else 0 end) as drittversuch,
'?tid=30610360&Prüfungskurztext='||pktxt||'&&cachingcontrol=clearmask' as hidden_drittversuch
into temp tmp_ergebnis
```

Werden beim Maskenaufruf der Detailansicht weitere Felder benötigt, wird für diese eine Variable angelegt:

```
#assign stg=""/>
<#if "<"!=""<#assign stg="&Studiengang=<"?replace("'","")?replace("|",",")/
```

So ist gewährleistet, dass der Parameter nicht in die URL übernommen wird, falls das Maskenfeld leer ist. Zudem werden einfache Hochkommata ' und Pipes | ersetzt, um zugewährleisten, dass die URL funktioniert. Einfache Hochkommata ' könnten beispielsweise bei Schlüsseln, welche als charakter definiert sind übergeben werden und Pipes | bei Aufzählungen. Die Variable wird in den select der Spalte hidden_drittversuch eingebunden:

```
'?tid=30610360&Prüfungskurztext='||pktxt||'${stg}&&cachingcontrol=clearmask' as hidden_drittversuch
```

Sonderfall: Falls eine Maske in einem Link auf sich selbst verweist wird empfohlen statt der Maskennummer die automatisch gefüllte Freemarker-Variable ${Maskennummer} zu benutzen:

'?tid=${Maskennummer}&Prüfungskurztext='||pktxt|'${stg}&&cachingcontrol=clearmask' as hidden_drittversuch

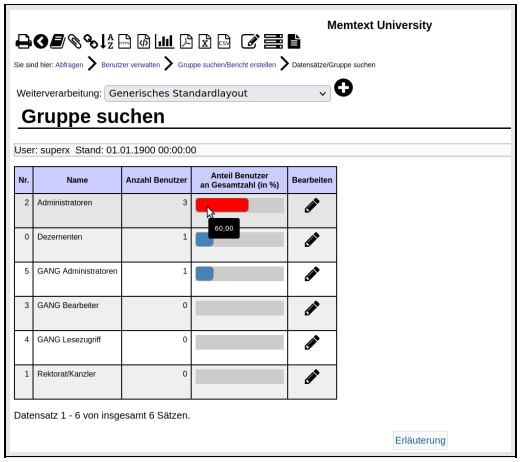
Hintergrund: Falls man eine Kopie des entsprechenden Berichtes angelegt hat, um diesen mit eingeschränkten Funktionen (bspw. für User mit eingeschränkten Rechten) bereitzustellen, bietet sich die sximport-Funktion an. Würde dann statt ${Maskennummer} die tid verwandt, so würde der Link in der Berichtskopie auf den Hauptbericht verweisen, was für User mit eingeschränkten Rechten nicht erwünscht ist.

Datenbalken

Datenbalken in Standardberichten allgemein

Es gibt die Möglichkeit, in der Ergebnistabelle horizontale Balken (In Excel nennt man dies "Datenbalken") anzeigen zu lassen. Dies kann z.B. dazu genutzt werden, ein Balkendiagramm darzustellen.

Hier ein Beispiel: Maske Administration -> Benutzer -> Gruppe suchen -> dort im Ergebnis die Spalte "Anteil Benutzer an Gesamtzahl (in %) "



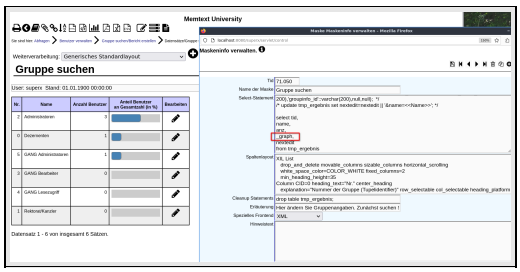
Voreingestellt ist auch, dass beim MouseOver ein Tooltip mit der Zahl angezeigt wird.

Es gibt die Möglichkeit, die Datenbalken zu variiieren:

- Balken links- oder rechtsbündig
- Die Breite der Balken wird in Pixel angegeben, Sie können die Breite aber anpassen
  - Der jew. Balken kann relativ zu einem Gesamtwert dargestellt werden
  - Der gesamte Balken kann in der Breite angepaßt werden
- Auch Balken mit Serien sind möglich

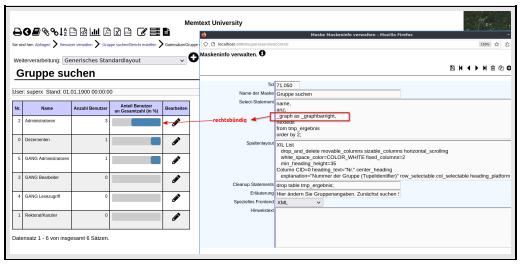
Datenbalken linksbündig

Der Default des Datenbalkens ist linksbündig, wenn sie also "_graph" als Spaltenüberschrift nehmen, wird der Balken linksbündig gezeichnet.



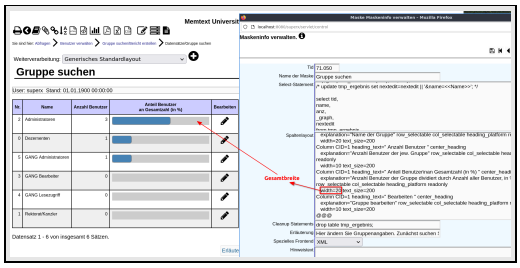
Datenbalken rechtsbündig

Wenn sie "_graphbarright" als Spaltenüberschrift nehmen, wird der Balken rechtsbündig gezeichnet.



Datenbalken Breite

Die Gesamtbreite der Spalte mit dem Datenbalken (wie gewohnt anteilig zu allen Spalten) wird in der xil_proplist festgelegt. Im folgenden Beispiel wurde die Balkenbreite von 10 auf 20 erhöht:

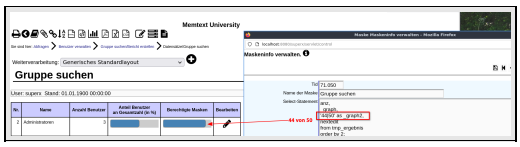


Wenn der Wert für den Datenbalken kein Prozentwert ist (also nicht die Skala 1-100 hat), können Sie den Gesamtwert mit "|" dahinter setzen. Dadurch wird die Breite anteilig berechnet. Beispiel:

Eine Spalte hat den Wert 44 von einem Gesamtwert von 50. Damit der Datenbalken nicht schmaler als die Häfte ist geben Sie die Gesamtsumme dahinter an:

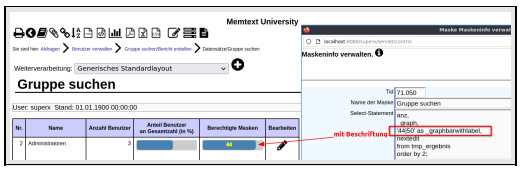
```
select ..., '44|50' as _graph2 ...
```

Damit wird der Datenbalken mit der Hälfte der Gesamtbreite angezeigt.



Datenbalken mit Beschriftung

Sie können den Datenbalken auch mit einer Beschriftung versehen, indem Sie als Spaltenname "_graphbarwithlabel" nutzen:



Je nach Länge der Zahl und Breite des Balkens kann es zu unschönen Darstellungen kommen.

Datenbalken mit Serien

Wenn Sie den Spaltennamen beginnend mit "_graphbarxseries" liefern werden Datenbalken mit Serien ausgegeben. Dazu wird als Wert eine Liste mit Komma-getrennten Ganzzahlen erwartet, normiert auf die Breite der Spalte in der **_xil_proplist**. Hier ein Beispiel:

Die Farben der Serie können Sie im Template

```
<xsl:template
name="myBarColorFg" >...
```

in der Datei /superx/xml/pageComponents_html_final.xsl überlagern. Die Auslieferung befindet sich in der Datei /superx/xml/graphtools.xsl.

Datenbalken in xCube

Da in xCube die Masken nicht änderbar sind, können Sie die Funktion über die Maske bzw. das Maskenfeld "Spaltenvisualisierung" direkt nutzen.

- Siehe in xCube

Exporte anpassen

Exportformate

Im XML-Frontend können Abfragen direkt nach html (Druckversion), XML, PDF oder XLS (->Excel) exportiert werden. Die zugehörigen Stylesheets lauten:

- **html (Druckversion)**

- $SUPERX_DIR/webserver/tomcat/webapps/superx/xml/tabelle_html_p.xls

- **PDF**
  - $SUPERX_DIR/webserver/tomcat/webapps/superx/xml/tabelle_fo_pdf.xsl
- **XLS**
  - $SUPERX_DIR/webserver/tomcat/webapps/superx/xml/tabelle_xls.xsl

Der PDF-Konverter arbeitet mit der OpenSource-Bibliothek FOP, der Excel-Konverter mit POI. Die Vorlagen können als Grundlage für eigene Stylesheets verwendet werden.

Der PDF-Export funktioniert zwar technisch, aber leider sehen die Ergebnisse oft nicht "schön" aus, da die Berichte in SuperX generell über die Seitenbreite hinaus gehen. Wir empfehlen daher, die Exporte nur bei speziell geeigneten Berichten (mit weniger Spalten) zu verwenden. Außerdem gibt es für Volltexte keine Silbentrennung.

Der XLS-Export wurde mit MS Excel und OpenOffice getestet. Da die Produkte automatisch auf Seitenbreite skalieren können, sieht der Export hier deutlich besser aus.

Außerdem können grundlegende Layoutelemente wie Kopf- und Fußzeilen und Seitenzahlen individuell angepasst werden, ohne zwingend XSLT-Kenntnisse zu haben.

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 ein Knoten mit der Breite kommen (im mm)

```
<fo:table>
<fo:table-column  column-
```

Blöcke zusammenhalten

```
<fo:block  keep-
together.within-
```

Um lokal zu testen gibt es die 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 ein PDF erzeugt werden soll).

Schriftgröße des PDF-Exportes anpassen

Bei Bedarf kann man die Schriftgröße des PDF-Exportes anpassen. Dazu einfach beim letzten Select des "select_stmts" der Maske noch angeben

```
select ..... , 6::smallint as hidden_pdf_fontsize from tmp_erg order by ...;
```

und in der XIL-Proplist als Spalte "hidden_pdf_fontsize" hinterlegen.

Oder falls die Spalten über eine CUSTOM_ Variable im Repository definiert werden, kann man einfach ergänzen:

```
<#assign CUSTOM_1234={"resulttable":as hidden_pdf_fontsize","caption":"hidden_pdf_fontsize","width":1}

}/>
```

Excelexport

Es wird nicht vorausgesetzt, dass Sie Microsoft Excel nutzen. Es wird zwar das Excel-Format erzeugt (*.xslx), aber in der Regel sind diese Dateien auch für andere Softwareprodukte, z.B. LibreOffice, nutzbar.

Performance-optimiertes Excel

Man kann einen Performance-optimierten Excelexport anstoßen, indem man im Berichtskopf den Kommentar "--ram excelexport" setzt.

Dies ist z.B. bei Datenblattberichten standardmäßig der Fall.

Excel-Vorlagen

Diese Funktionalität wird derzeit überarbeitet, weil sie nur mit dem alten Excel Format (*.xls) funktioniert.

Am besten nimmt man zur Bearbeitung eine bestehende xsl-Vorlage.

Man kann eine bestehende Excel-Datei 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 man trotzdem reinschreiben!)

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 Excel-Vorlage enthaltenes nicht zu überschreiben:

Tabellenblatt

Zelle kann Attribute haben ebene=summe

Zellen

Für Zahlen

mögliche Attribute: width (gilt logischerweise für ganze Spalte)

Um lokal zu testen gibt es Java-Klasse de.superx.bin.ExcelPdfCreator

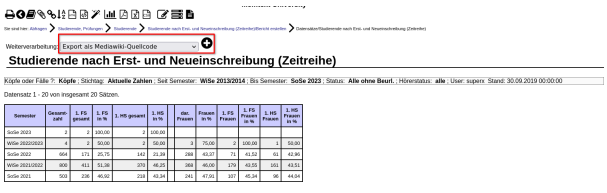
Params

-in/home/superx/iaf-ausgaben.xml -xsl/home/superx/tabelle_xls_xxxxxx.xsl -out/home/superx/test.xls

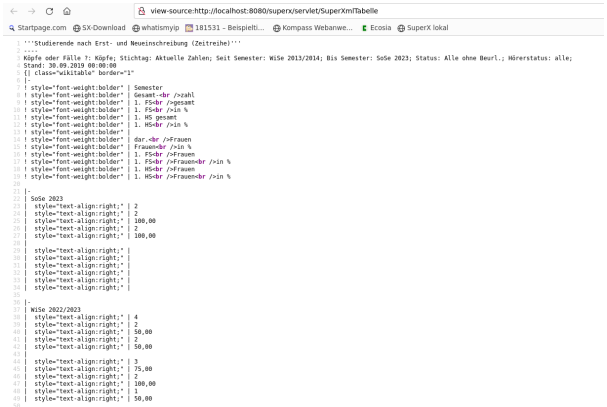
(Dateiendung legt fest, dass eine Excel-Datei erzeugt werden soll)

Export als Mediawiki-Quellcode

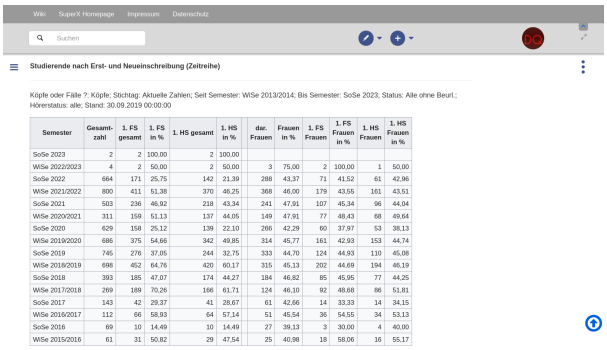
Wenn sie die Software Mediawiki nutzen, können Sie Tabellenexporte auch direkt als Quellcode exportieren:



Der Quellcode wird im Browser in einem neuen Fenster ausgegeben:



Diesen Quellcode können Sie per Copy-Paste in Ihre Mediawiki übernehmen, in der Variante Bluespice sieht das dann so aus:



Semester	Gesamt-zahl	1. FS gesamt	1. FS in %	1. HS gesamt	1. HS in %	dar. Frauen	Frauen in %	1. FS Frauen	1. FS Frauen in %	1. HS Frauen	1. HS Frauen in %
SoSe 2023	2	2	100,00	2	100,00						
WiSe 2022/2023	4	2	50,00	2	50,00	3	75,00	2	100,00	1	50,00
SoSe 2022	664	171	25,75	142	21,39	288	43,37	71	41,52	61	42,96
WiSe 2021/2022	800	411	51,38	370	46,25	368	46,00	179	43,55	161	43,51
SoSe 2021	503	236	46,92	218	43,34	241	47,91	107	45,34	96	44,04
WiSe 2020/2021	311	159	51,13	137	44,05	149	47,91	77	48,43	68	49,64
SoSe 2020	629	168	25,12	139	22,10	266	42,29	60	37,97	53	38,13
WiSe 2019/2020	686	375	54,66	342	49,85	314	45,77	161	42,93	153	44,74
SoSe 2019	745	276	37,05	244	32,75	333	44,70	124	44,93	110	45,08
WiSe 2018/2019	698	452	64,76	420	60,17	315	45,13	202	44,69	194	46,19
SoSe 2018	393	185	47,07	174	44,27	184	46,82	85	45,95	77	44,25
WiSe 2017/2018	269	189	70,26	166	61,71	124	46,10	92	48,68	86	51,81
SoSe 2017	143	42	29,37	41	28,67	61	42,66	14	33,33	14	34,15
WiSe 2016/2017	112	66	58,93	64	57,14	51	45,54	36	54,55	34	53,13
SoSe 2016	69	10	14,49	10	14,49	27	39,13	3	30,00	4	40,00
WiSe 2015/2016	61	31	50,82	29	47,54	25	40,98	18	58,06	16	55,17

Masken- und Tabellenlayouts mit XSLT

Stylesheets verwalten

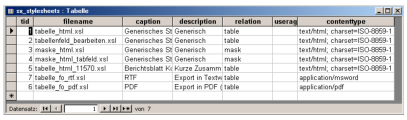
Es ist möglich für Spezialfunktionen eigene Stylesheets für einzelne Masken zu hinterlegen. Zunächst muss für das Ergebnis ein neues Stylesheet erzeugt werden. Als Vorlage für Masken können Sie das Muster-Stylesheet

$WEBAPP/xml/maske_html.xsl

verwenden. Für Ergebnistabellen können Sie das Muster-Stylesheet

$WEBAPP/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.

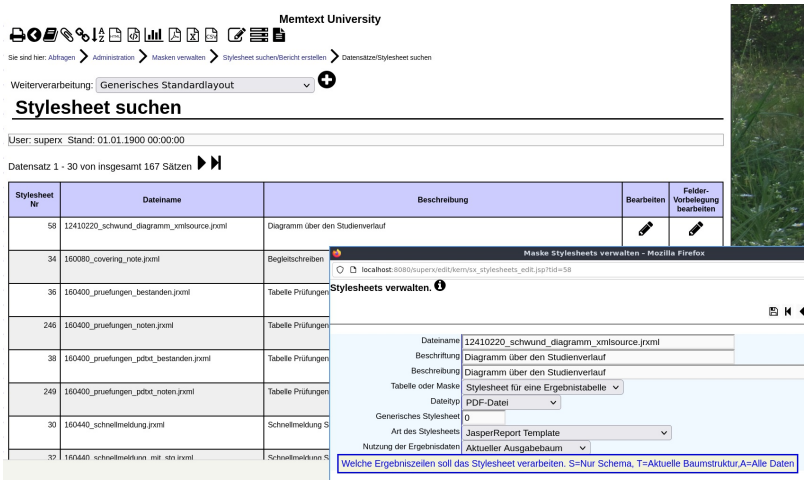


id	filename	caption	description	relation	useragent	contenttype
1	tabelle_html.xsl	Generisches St	Generisch	table		text/html; charset=ISO-8859-1
2	tabellenfeld_bearbeiten.xsl	Generisches St	Generisch	table		text/html; charset=ISO-8859-1
3	maske_html.xsl	Generisches St	Generisch	mask		text/html; charset=ISO-8859-1
4	maske_html_tabfeld.xsl	Generisches St	Generisch	mask		text/html; charset=ISO-8859-1
5	tabelle_html_11570.xsl	Berichtsblatt Ki	Kurze Zusamm	table		text/html; charset=ISO-8859-1
7	tabelle_fo_rtf.xsl	RTF	Export in Texte	table		application/msword
6	tabelle_fo_pdf.xsl	PDF	Export in PDF (	table		application/pdf

Das Beispiel zeigt einige Stylesheets, 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* bietet die Möglichkeit, ein Stylesheet für spezielle Lesegeräte anzubieten, z.B. WAP-Hanys 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).
- *is_generic*. Generisches Stylesheet für alle Ergebnistabellen (1=ja)
- *usage_resultset_data*. Nutzung der Ergebnisdaten: Welche Ergebniszeilen soll das Stylesheet verarbeiten. S=Nur Schema, T=Aktuelle Baumstruktur,A=Alle Daten. Die Ausprägung T=aktuelle Baumstruktur wird nur ausgewertet, wenn die Ergebnistabelle in den Zeilen eine Aufklappfunktion bietet.
- *stylesheet_type*. Art des Stylesheets (XSL,JRXML,XSL_JRXML,XSL_FO,XML,NATIVE,XLSX). Der Wert XSL_JRXML wird nur ausgewertet wenn das RPTA-Modul installiert ist (Berichtsassistent)
- *jr_datasource*. Datenquelle (JasperReports). Wird nur bei JasperReports ausgewertet. Mögliche Werte: RS (Resultset vom Servlet) oder XMLSOURCE (XML vom Servlet). Ersteres ist schneller, zweites ist flexibler.

Im Browser können Sie die Daten mit dieser Maske bearbeiten:

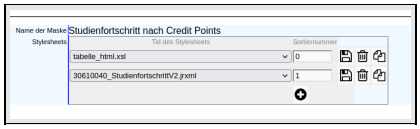


Zuordnung eines Stylesheets zu einer Maske

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. Die Tabelle lässt sich über ein Bearbeitungsformular füllen. Dieses wird erreicht wenn die Maske gesucht wird, welcher ein Stylesheet zugeordnet wird:

Das Feld Sortiernummer (ord) kennzeichnet die Reihenfolge der anzubietenden Stylesheets. Wir sehen hier, dass zuerst das generische Standard-Stylesheet angezeigt wird und dann ein JasperReport als weiteres Stylesheet.



Defaultmäßig sind die Stylesheets für html (Druckversion in neuem Fenster), xml, Excel und PDF in jeder Ergebnistabelle enthalten. Andere Stylesheets werden in der obigen Tabelle zugeordnet.

Einzelne Templates anpassen

Mit eigenen XSL-Stylesheets kann man das Aussehen von Masken oder Ergebnistabellen sehr individuell anpassen. Oben beschrieben 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, kommen diese Erweiterungen dann jedoch nicht in die kopierten Spezialstylesheets.

Daher sind die Stylesheets inzwischen etwas „objektorientierter" und kleine Änderungen kann man auch in der Datei

.../webapps/superx/xml/page_components_final.xsl

vornehmen.

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 ein template-Aufruf definiert.

```
<p  class="maskentitel">
<xsl:value-of
```

Das Template ist als leer in Page-Components.xsl definiert. Man kann es in pageComponents_final.xsl definieren, dann wird es überschrieben.

```
<xsl:template
name="pccustomize">
```

Die Links werden nur bei den entsprechenden Masken eingebaut.

Außerdem wird standardmäßig das leere maskonload überschrieben, um den div der folgenden Maskenfelder etwas tiefer zu setzen, damit Platz für einen größeren Titel ist.

Um bei bestimmten Masken Export-Buttons auszublenden, kopieren Sie das entsprechende Template (hier: exportButtons) aus der Page-Components.xsl in die pageComponents_final.xsl und fügen dort eine if-Bedingung ein.

Beispiel für das Entfernen des PDF-Export-Buttons für die Masken mit der tid 16000 und 17000:

```
 <xsl:template
name="exportButtons"  >
```

Will man ein spezielles Tabellenstylesheet erzeugen, braucht man tabelle_html.xsl nicht mehr kopieren, sondern erzeugt eine xsl-Datei mit den Standard-Importen und fügt dort den Import für tabelle_html.xsl hinzu.

```
<xsl:import
href="xsl_functions.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">
```

überlagern.

Am Ende folgt dann, um die Tabelle aufzubauen.

```
<xsl:template  match="/">
<xsl:call-template
```

Besonderes XML zu ALLEN Masken hinzufügen

Für das Management-Modul gibt es eine 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¶m=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  .../>
```

der wird dann in Tabellen XML unter ergebnisse/ergebniselement hinzufügt und kann ausgewertet werden.

z.B:

```
<div  id="Navigation">
<xsl:for-each
```

Der hinzuzufügende XML wird dynamisch mit Freemarker generiert, z.B.:

```
<xupdate>
 <text
```