Datenbanken und XML

Inhaltsverzeichnis

Einführung

MariaDB-Datenbanken sind leider nur bedingt portabel, zum Beispiel auf einen anderen Datenträger oder eine neuere Version. Deshalb speichere ich den Inhalt von MariaDB-Datenbanken zusammen mit der Tabellen-Struktur in XML-Dateien.

Dieses Verzeichniss enthält Hilfsmittel, um Daten aus einer Datenbank in eine XML-Datei zu übertragen und umgekehrt. Das Skript `dbdump` führt die typischen Aufgaben aus.

Dateien

`catalog/src/dbdump/DataInserter.pm`

: Perl-Modul `Herbaer::DataInserter`

`catalog/src/dbdump/MysqlAccess.pm`

: Perl-Modul `Herber::MysqlAccess`

`catalog/src/dbdump/dbdump.pl`

: Erzeugt aus einer MariaDB-Datenbank eine XML-Datei (Dump) `catalog/dbdump/`*`DBNAME`*`.xml` mit Angaben zur Struktur der Datenbank und deren Inhalt. *`DBNAME`* steht für den Namen der Datenbank.

`catalog/src/dbdump/dbname.pl`

: Liest zu einem Datenbankzugangs-Schlüssel den Namen der zugehörigen Datenbank aus der Geheimnisdatei `web/secrets` und gibt ihn aus.

`catalog/dbdump`

: Die Dateien in diesem Verzeichnis werden vom Script `dbdump` erzeugt.

`catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml`

Aufbau (XML-Namensraum `http://herbaer.de/xmlns/20201201/dbdump`) und Inhalt (XML-Namensraum `http://herbaer.de/xmlns/20201201/dbcontents/`*`DBNAME`*) der Datenbank *`DBNAME`*.

`catalog/dbdump/`*`DBNAME`*`_create.sql`

Erstellt die Datenbank *`DBNAME`* und deren Tabellen.

`catalog/dbdump/`*`DBNAME`*`_ht.xslt`

Vorlage zur HTML-Ansicht der Dateien `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml`

`catalog/dbdump/`*`DBNAME`*`_insert.pl`

Fügt den Datenbank-Inhalt aus einer Datenbank-Dump-Datei `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml` in eine Datenbank mit der gleichen Struktur ein.

`catalog/dbdump/`*`DBNAME`*`.rng`

Beschreibt den XML-Namensraum `http://herbaer.de/xmlns/20201201/dbcontents/`*`DBNAME`* für den Inhalt der Datenbank *`DBNAME`*.

`catalog/dbdump/`*`DBNAME`*`.xml`

Symbolischer Verweis auf eine der Dateien `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml`, meist auf die neueste.

`catalog/src/dbdump/ dbdump.rng`

Beschreibt den XML-Namensraum `http://herbaer.de/xmlns/20201201/dbdump`.

`catalog/src/dbdump/ dump_create.xslt`

Erzeugt aus einer Datenbank-Dump-Datei `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml` die Datei `catalog/dbdump/`*`DBNAME`*`_create.sql`.

`catalog/src/dbdump/dump_htinc.xslt`

Wird von den Dateien `catalog/dbdump/`*`DBNAME`*`_ht.xslt` eingebunden.

`catalog/src/dbdump/dump_htview.xslt`

Erzeugt aus einer Datenbank-Dump-Datei `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml` die Datei `catalog/dbdump/`*`DBNAME`*`_ht.xslt`.

`catalog/src/dbdump/ dump_insert.xslt`

Erzeugt aus einer Datenbank-Dump-Datei `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml` die Datei `catalog/dbdump/`*`DBNAME`*`_insert.pl`.

`catalog/src/dbdump/dump_rng.xslt`

Erzeugt aus einer Datenbank-Dump-Datei `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml` die Datei `catalog/dbdump/`*`DBNAME`*`.rng`.

`catalog/src/dbdump/dump_struct_ht.xslt`

HTML-Ansicht der Datenbank-Struktur aus einer Datenbank-Dump-Datei `catalog/dbdump/`*`DBNAME_TIMESTAMP`*`.xml`.

`catalog/src/dbdump/path.xslt`

Symbolischer Verweis auf die Datei `pool/path.xslt`. Diese Datei ist in diesem Zusammenhang enstanden. Der Verweis dient nur zur Dokumentation.

`catalog/src/dbdump/test_path.xml` , `catalog/src/dbdump/test_path.xslt`

Diese beiden Dateien dienen zum Test der Datei `pool/path.xslt`.

`catalog/src/dbdump/dbdump`

Skript, das die Dateien im Verzeichnis `catalog/dbdump` erstellt.

dbdump

[Quelltext]

Übersicht

`dbdump --help|--version`

`dbdump [ --verbose ... | --no_verbose ][ --overwrite|--no_overwrite ]`
`[ --data|--no_data ][ --struct|--no_struct ][ --endaction|--no_endaction ]`
`[ --srcdir` *`SRCDIR`* `][ --dumpdir` *`DUMPDIR`* `][ --eafile` *`EAFILE`* `]`
`--dump --rng --createsql --htview --insertscript`
`DBKEY` ...

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus.

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Meldungen über den Programmablauf werden nach STDOUT ausgegeben, Programme werden mit dem Argument `--verbose` aufgerufen.

`--no_verbose`

Diese Option hebt die Wirkung der Option `--verbose` auf.

`--overwrite`

Existierende Dateien werden überschrieben.

`--no_overwrite`

Existierende Dateien werden nicht überschrieben.

`--data`

Die Datenbank-Dump-Datei `dbdump/`*`DBNAME`*`.xml` enthält den Datenbank-Inhalt.

`--no_data`

Die Datenbank-Dump-Datei `dbdump/`*`DBNAME`*`.xml` enthält nicht den Datenbank-Inhalt.

`--struct`

Die Datenbank-Dump-Datei `dbdump/`*`DBNAME`*`.xml` enthält die Datenbank-Struktur.

`--no_struct`

Die Datenbank-Dump-Datei `dbdump/`*`DBNAME`*`.xml` enthält nicht die Datenbank-Struktur. Die Dateien `dbdump/`*`DBDNAME`*`_create.sql`, `dbdump/`*`DBDNAME`*`_ht.xslt`, `dbdump/`*`DBDNAME`*`_insert.pl` und `dbdump/`*`DBDNAME`*`.rng` werden nicht erstellt.

`--endaction`

Nach dem Ablauf des Skripts werden die Befehle aus der Datei *`EAFILE`* ausgeführt.

`--no_endaction`

Nach dem Ablauf des Skripts werden die Befehle aus der Datei *`EAFILE`* nicht ausgeführt.

`--srcdir` *`SRCDIR`*

Das Verzeichnis der Hilfs-Skripte. In diesem Verzeichnis werden die folgenden Dateien erwartet:

```
dbdump.pl
dbname.pl
dump_create.xslt
dump_htview.xslt
dump_insert.xslt
dump_rng.xslt
```

`--dumpdir` *`DUMPDIR`*

Das Ausgabe-Verzeichnis.

`--eafile` *`EAFILE`*

Datei mit Shell-Befehlen, die am Ende des Ablaufs dieses Skripts ausgeführt werden. Eine typische Aufgabe ist es, den Computer auszuschalten.

`--dump`

Datei *`DUMPDIR/DBNAME_TIMESTAMP`*`.xml` und darauf den symbolischen Verweis *`DUMPDIR/DBNAME`*`.xml` erstellen.

`--rng`

Datei *`DUMPDIR/DBNAME`*`.rng` erstellen.

`--createsql`

SQL-Skript *`DUMPDIR/DBNAME`*`_create.sql` erstellen.

`--htview`

Datei *`DUMPDIR/DBNAME`*`_ht.xslt` erstellen.

`--insertscript`

Perl-Skript *`DUMPDIR/DBNAME`*`_insert.pl` erstellen.

`--dbkey` *`DBKEY`*

Ein Schlüssel für einen Datenbank-Zugang, s. `MysqlAccess.pm`. Die Datenbank-Zugangsdaten werden aus der „Geheimnis-Datei” `web/secrets` gelesen.

Beschreibung

Dieses Skript erzeugt zu einer oder mehreren Datenbanken die Dateien

```
dbdump/DBDNAME_TIMESTAMP.xml
dbdump/DBDNAME.xml
dbdump/DBDNAME_create.sql
dbdump/DBDNAME_create.sql
dbdump/DBDNAME_ht.xslt
dbdump/DBDNAME_insert.pl
dbdump/DBDNAME.rng
```

Quelltext

[Beschreibung]

```
#!/bin/bash
# Datenbanken und XML
# 2020-12-10 Herbert Schiemann <h.schiemann@herbaer.de>


# Zunächst Funktionen,
# die an die konkrete Anwendung anzupassen sind.


# Zähler, Variable, Aktionen
declare_vars ()
{
   # Ein Leerzeichen als Wert bedeutet, dass Positionsargumente verarbeitet werden
   _argv=" ";

   # Suchpfad für rc-Dateien, : - getrennte Liste von Verzeichnispfaden
   # Falls leer, wird die Option --rc nicht speziell behandelt
   g_configpath= ;

   # Zähler
   g_counters="  \
      verbose    \
      overwrite  \
      data       \
      struct     \
      endaction  ";

   # Variable
   g_variables=" \
      srcdir     \
      dumpdir    \
      eafile     ";

   # Aktionen
   g_actions="      \
      dump          \
      rng           \
      createsql     \
      htview        \
      insertscript  ";
} # declare_vars


# setzt Vorgabe-Werte
set_defaults ()
{
   local b=$(realpath $0);
   b=${b%/catalog/src/*};
   [[ -n "$verbose"   ]] || verbose=1 ;
   [[ -n "$overwrite" ]] || overwrite=0 ;
   [[ -n "$endaction" ]] || endaction=0 ;
   if [[ -z "$data" && -z "$struct" ]]; then
      data=1;
      struct=1;
   elif [[ -z "$data" ]]; then
      data=1
      (( struct )) && data=0;
   elif [[ -z "$struct" ]]; then
      struct=1
      (( data )) && struct=0;
   fi;
   [[ -n "$srcdir"   ]] || srcdir="$b/catalog/src/dbdump";
   [[ -n "$dumpdir"  ]] || dumpdir="$b/catalog/dbdump";
   [[ -n "$eafile"   ]] || eafile="$b/var/endaction";
} # set_defaults
```

```
# Zeigt eine kurze Hilfe an
show_help ()
{
   local cmd=${0#*/} ;
   set_defaults ;
   cat << .HELP ;
$cmd --version
$cmd --help
$cmd (Aktion ...)  DBKEY ..

Aktionen
--dump             XML-Dump-Datei erstellen
--rng              Datenbank-spezifische RelaxNG-Datei
--createsql        SQL-Skript zum Anlegen einer Datenbank
--htview           Vorlage zur HTML-Ansicht der Daten
--insertscript     Perl-Script zum Einfügen der Daten

DBKEY              Schlüssel zu einem Datenbankzugang

Optionen
--[no_]verbose     Erhöht den Umfang der Ausgabe des Scripts ($verbose)
--[no_]overwrite   Existierende Dateien überschreiben ($overwrite)
--[no_]data        Datenbank-Inhalt in XML-Datei ($data)
--[no_]struct      Datenbank-Struktur in XML-Datei ($struct)
--[no_]endaction   Aktion am Ende ausführen ($endaction)
--srcdir  SRCDIR   Verzeichnis der Skripte
                   ($srcdir)
--dumpdir DUMPDIR  Ausgabeverzeichnis
                   ($dumpdir)
--eafile  EAFILE   Datei mit dem End-Befehl
                   ($eafile)
.HELP
} # show_help


# Zeigt die Version an
show_version ()
{
   cat << .VERSION ;
catalog/src/dbdump/dbdump
Datenbanken und XML
2020-12-10, Herbert Schiemann, h.schiemann@herbaer.de
GPL Version 2 oder neuer
.VERSION
} # show_version


# Variable und Zähler initialisieren
init_vars () {
   local v;
   declare_vars ;
   for v in $g_counters $g_variables $g_actions; do
      eval "$v=" ;
   done;
} # init_vars


# Argumente verarbeiten
read_args ()
{
   local wd ;
   local lastwd ;
   local var ;
   local ok ;

   has_actions=0 ;
   for wd in "$@"; do
      if [[ "$lastwd" = "--" ]]; then
          _argv="$_argv $wd";
      elif [[ -n "$lastwd" ]]; then
         if [[ "$wd" =~ ^[\ a-zA-Z0-9./_#-]+$ ]]; then
            ok=0 ;
            for var in $g_variables; do
               if [[ "$var" == "$lastwd" ]]; then
                  (( ++ok )) ;
                  eval "$var=\"$wd\"" ;
                  break ;
               fi ;
            done ;
            if (( ! ok )); then
               (( verbose )) && echo "Unbekannte Option --$lastwd $wd" ;
               exit 11 ;
            fi ;
         else
            (( verbose )) && echo "Ungültiger Optionswert --$lastwd $wd" ;
            exit 12;
         fi;
         lastwd= ;
      else
         case "$wd" in
            --version )
               show_version ;
               exit 0 ;
               ;;
            --help )
               show_version ;
               show_help ;
```

```
               exit 0 ;
               ;;
            -- )
               if [[ -n "$_argv" ]]; then
                  lastwd=--;
                  continue;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 13 ;
               fi ;
               ;;
            --* )
               if [[ "$wd" =~ ^--[a-z][a-z0-9_]*$ ]]; then
                  lastwd=${wd#--} ;
                  ok=0 ;
                  for var in $g_counters ; do
                     if   [[ "$lastwd" == $var    ]]; then
                        eval "(( ++$lastwd ))" ;
                     elif [[ "$lastwd" == "no_$var" ]]; then
                        eval "${lastwd#no_}=0" ;
                     else
                        continue;
                     fi;
                     (( ++ok )) ;
                     break ;
                  done;
                  if (( !ok )); then
                     for var in $g_actions; do
                        if [[ "$lastwd" == "$var" ]]; then
                           eval "(( ++$var ))" ;
                           (( ++ok ));
                           has_actions=1;
                           break;
                        elif [[ "$lastwd" == "no_$var" ]]; then
                           eval "(( ++no_$var ))" ;
                           (( ++ok ));
                           break;
                        fi;
                     done;
                  fi;
                  (( ok )) && lastwd=;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 14 ;
               fi ;
               ;;
            * )
               if [[ -n $_argv ]]; then
                   _argv="$_argv $wd";
               else
                    (( verbose )) && echo "Ungültige Option $wd" ;
                    exit 15 ;
               fi;
               ;;
         esac ;
      fi ;
   done ;
   if [[ -n $lastwd && "$lastwd" != "--" ]]; then
      (( verbose )) && echo "Unverarbeitete Option --$lastwd";
      exit 16 ;
   fi ;
   [[ "$_argv" =~ ^[[:space:]]+$ ]] && _argv="" ;
} # read_args


# Aktionen ausführen
run_actions ()
{
   local act ;
   for act in $g_actions; do
     eval "(( ! has_actions && ! no_$act || $act )) && process_$act";
   done;
} # run_actions


# show_variables VARNAME1 VARNAME2
# Werte der Variablen anzeigen
show_variables ()
{
   local v ;
   for v in $g_counters $g_variables $g_actions $1; do
      eval "echo \"$v = \$$v\"" ;
   done;
} # show_variables
```

```
# ist ein Befehl verfügbar?
# check_command xsltproc sed
check_command ()
{
    local f ;
    for f in "$@"; do
        if [[ -z "$(which $f)" ]]; then
            (( verbose )) && echo "Befehl $f ist nicht verfügbar.";
            return 1;
        fi;
    done;
} # check_command


# Können die Eingabedateien gelesen werden?
# check_infiles first/path/to/file path/to/second_file ;
check_infiles ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -s "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist leer";
         return 1;
      fi;
      if [[ ! -r "$f" ]]; then
         (( verbose )) && echo "Kann Datei \"$f\" nicht lesen";
         return 1;
      fi;
   done;
   return 0;
} # check_infiles


# Können die Ausgabedateien erstellt werden?
# erstellt fehlende Verzeichnisse und löscht existierende Dateien
# nach Maßgabe der Variablen overwrite
# check_outfiles first/path/to/file path/to/second_file ;
check_outfiles ()
{
   local fp;
   local dir;
   local verb;
   (( verbose )) && verb=--verbose ;
   for fp in "$@"; do
      if [[ ! -e $fp ]]; then
         dir=${fp%/*};
         if [[ -n $dir && ! -e $dir ]]; then
            mkdir -p $verb $dir ;
            if [[ ! -d $dir ]]; then
               (( verbose )) && echo "$dir ist kein Verzeichnis";
               return 1;
            fi;
         fi;
      elif [[ -d $fp ]]; then
         (( verbose )) && echo "$fp ist ein Verzeichnis";
         return 1;
      elif (( overwrite )); then
         (( verbose )) && echo "lösche $fp";
         rm $fp;
      else
         (( verbose )) && echo "$fp existiert";
         return 1;
      fi;
      (( verbose )) && echo "$fp";
   done;
   return 0;
} # check_outfiles


# Sind die Dateien ausführbar?
# check_executeable first/path/to/script path/to/second_srcipt ;
check_executeable ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -x "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine ausführbare Datei";
         return 1;
      fi;
   done;
   return 0;
} # check_executeable
```

```
# XML-Dump-Datei erstellen
process_dump ()
{
   (( verbose )) && echo "process_dump" ;
   local d="$srcdir/dbdump.pl";
   local n="$srcdir/dbname.pl";
   check_executeable $d $n || return;
   local verb;
   (( verbose )) && verb=--verbose ;
   local dbkey;
   local dbname;
   local o;
   local lnk;
   local spec=no_spec;
   local contents=no_contents;
   (( struct )) && spec=spec;
   (( data   )) && contents=contents;
   for dbkey in $_argv; do
      dbname=$($n $dbkey);
      o=$dumpdir/$dbname.xml ;
      [[ -L $o ]] && rm $o;
      [[ -e $o ]] && mv $o $o.$(date +%Y%m%d%H%M%S%N) ;
      check_outfiles $o || continue;
      $d --dumpdir "$dumpdir" --$spec --$contents --dbkey "$dbkey" $verb;
      check_infiles $o || continue;
      lnk=${o%.xml}_$(date +%Y%m%d%H%M%S%N).xml;
      mv $o $lnk;
      ln -s $lnk $o;
   done;
} # process_dump


# Datenbank-spezifische RelaxNG-Datei
process_rng ()
{
   (( struct )) || return;
   (( verbose )) && echo "process_rng" ;
   local t=$srcdir/dump_rng.xslt ;
   check_infiles $t || return;
   local dbkey;
   local dbname;
   local o;  # Ausgabedatei
   local d;  # Dump-Datei
   for dbkey in $_argv; do
      dbname=$($srcdir/dbname.pl $dbkey);
      d=$dumpdir/$dbname.xml ;
      check_infiles $d  || continue;
      o=$dumpdir/${dbname}.rng;
      check_outfiles $o || continue;
      xsltproc --stringparam p_xslt "../../pool/rng_ht.xslt" $t $d > $o;
   done;
} # process_rng


# SQL-Skript zum Anlegen einer Datenbank
process_createsql ()
{
   (( struct )) || return;
   (( verbose )) && echo "process_createsql" ;
   local t=$srcdir/dump_create.xslt ;
   check_infiles $t || return;
   local dbkey;
   local dbname;
   local o;  # Ausgabedatei
   local d;  # Dump-Datei
   for dbkey in $_argv; do
      dbname=$($srcdir/dbname.pl $dbkey);
      d=$dumpdir/$dbname.xml ;
      check_infiles $d  || continue;
      o=$dumpdir/${dbname}_create.sql;
      check_outfiles $o || continue;
      xsltproc $t $d > $o;
   done;
} # process_createsql


# Vorlage zur HTML-Ansicht der Daten
process_htview ()
{
   (( struct )) || return;
   (( verbose )) && echo "process_htview" ;
   local t=$srcdir/dump_htview.xslt ;
   check_infiles $t || return;
   local dbkey;
   local dbname;
   local o;  # Ausgabedatei
   local d;  # Dump-Datei
   for dbkey in $_argv; do
      dbname=$($srcdir/dbname.pl $dbkey);
      d=$dumpdir/$dbname.xml ;
      check_infiles $d  || continue;
      o=$dumpdir/${dbname}_ht.xslt;
      check_outfiles $o || continue;
      xsltproc --stringparam p_date $(date +%Y-%m-%d%TH%:M%:S) $t $d \
      | sed s/\$HB_DBNAME/$dbname/ > $o                              ;
   done;
} # process_htview
```

```
# Perl-Script zum Einfügen der Daten
process_insertscript ()
{
   (( struct )) || return;
   (( verbose )) && echo "process_insertscript" ;
   local t=$srcdir/dump_insert.xslt ;
   check_infiles $t || return;
   local dbkey;
   local dbname;
   local o;  # Ausgabedatei
   local d;  # Dump-Datei
   for dbkey in $_argv; do
      dbname=$($srcdir/dbname.pl $dbkey);
      d=$dumpdir/$dbname.xml ;
      check_infiles $d  || continue;
      o=$dumpdir/${dbname}_insert.pl;
      check_outfiles $o || continue;
      xsltproc $t $d > $o;
      check_infiles $o  || continue;
      chmod +x $o;
   done;
} # process_insertscript


# Sicherheit
export PATH=/bin:/usr/bin ;
IFS=$' \t\n' ;


init_vars ;


set -o noclobber ;   # existierende Dateien werden nicht überschrieben
shopt -s extglob nullglob ;


check_command realpath || exit 1;


read_args "$@" ;
set_defaults ;


check_command xsltproc date sed || exit 1;
(( verbose > 1 )) && show_variables;
(( struct )) || (( data )) || return;
run_actions ;
(( endaction )) && source "$eafile";
exit 0;
```

Herbaer::DataInserter

[Quelltext]

Anwendung

```
use Herbaer::DataInserter;

# Callback-Funktion
sub show_datarow {
   my ($table_name, $hashref_rowdata) = @_;
   my $field_name;
   my $value;
   while ( ($field_name, $value) = each %$hashref_rowdata ) {
      print "$table_name.$field_name = $value\n";
   }
}

my $path_to_xmlfile = "/path/to/dbdump/DBNAME.xml";
my $xml_namespace   = "http://herbaer.de/xmlns/20201201/dbcontents/DBNAME";
my $verbose         = 0;
Herbaer::DataInserter
   -> new ($xml_namespace, \&show_datarow, $verbose)
   -> parse_file ($path_to_xmlfile);
```

Funktionen

`$di = Herbaer::DataInserter -> new (`*`$xml_ns`*`, `*`$callback`*`, `*`$verb`*`)`

Der Konstruktor ergibt ein neues Objekt, das dazu dient, eine XML-Datei mit dem Inhalt einer Datenbank zu lesen.

`$xml_ns`

Der XML-Namensraum der XML-Elemente, die den Inhalt der Datenbank enthalten. Der XML-Namensraum ist für die Datenbank-Struktur spezifisch, typisch ist die Form `http://herbaer.de/xmlns/20201201/dbcontents/`*`DBNAME`*. Der Namensraum des XML-Wurzelelements kann ein anderer sein.

`$callback`

Eine Funktion, die für jede Datenreihe aufgerufen wird. In einer typischen Anwendung fügt sie eine Datenreihe in eine Datenbank ein. Die beiden Parameter sind:

`$table_name`

Der Name einer Datenbank-Tabelle.

`$hashref_rowdata`

Eine HASH-Referenz. Die HASH-Schlüssel sind die Namen der Felder der Datenreihe, die Werte die Feldinhalte.

`$verb`

Eine ganze Zahl, die den Umfang der Meldungen nach STDERR steuert. Dieser Parameter ist optional. Wenn er nicht definiert oder null ist, erfolgen keine Meldungen.

`$di -> parse_file (`*`$path_to_xmlfile`*`)`

Verarbeitet die XML-Datei unter dem Dateipfad *`$path_to_xmlfile`*.

Benutzte Module

`XML::SAX::ParserFactory`

Anwendung

Das Modul `Herber::DataInserter` wird von einem Skript benutzt, das von der Transformation `dump_instert.xslt` erzeugt wird.

Quelltext

[Beschreibung]

```
# Daten aus einer Dump-XML-Datei in eine Datenbank einfügen
# 2020-12-06 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::DataInserter ;


use XML::SAX::ParserFactory;


sub new {
   my ($class, $ns, $cb, $verb) = @_;
   $class = ref($class) || $class;
   $self = {
       "ns"      => $ns,
       "cb"      => $cb,
       # 0: erwarte Datenbank-Tag
       # 1: erwarte Tabelle
       # 2: erwarte Feld
       # 3: erwarte Felddaten
       "state"   => 0,
       "row"     => undef,
       "fld"     => undef,
       "verbose" => $verb // 1,
   };
   bless ($self, $class);
   $self -> {"parser"} = XML::SAX::ParserFactory -> parser (Handler => $self);
   return $self;
} # new


# SAX-Handler-Methoden
sub start_element {
   my ($self, $el) = @_;
   my $st = $self -> {"state"};
   if ($el -> {"NamespaceURI"} eq $self -> {"ns"}) {
      if ($st == 1) {
         $self -> {"row"} = {};
      }
      elsif ($st == 2) {
         $self -> {"fld"} = "";
      }
   }
   elsif ($st == 0) {
      return;
   }
   ++$self -> {"state"};
} # start_element


sub characters {
   my ($self, $chr) = @_;
   if ($self -> {"state"} == 3) {
      $self -> {"fld"} .= $chr -> {"Data"};
   }
} # characters


sub end_element {
   my ($self, $el) = @_;
   my $st = $self -> {"state"};
   if ($el -> {"NamespaceURI"} eq $self -> {"ns"}) {
      if ($st == 2) {
         $self -> {"cb"} -> ($el -> {"LocalName"}, $self -> {"row"});
      }
      elsif ($st == 3) {
         $self -> {"row"} -> {$el -> {"LocalName"}} = $self -> {"fld"};
      }
   }
   --$self -> {"state"} if $st;
} # end_element


# Verarbeitet eine Datei
sub parse_file {
   my ($self, $filename) = @_;
   my $fh;   # Dateihandle
   if (open ($fh, "<", $filename)) {
      $self -> {"parser"} -> parse_file ($fh);
      close ($fh);
   }
   else {
      print STDERR "Kann Datei \"$filename\" nicht lesen\n" if $self -> {"verbose"};
   }
} # parse_file


1;
```

Herbaer::MysqlAccess

[Quelltext]

Datenbank-Schlüssel

Daten für den Zugang zu einer MariaDB-Datenbank (Datenbankname, Anmeldename, Kennwort) stehen in einer „Geheimnis”-Datei. Die Geheimnis-Datei ist eine Textdatei mit Zeilen der folgenden Art:

```
dbkey.mysql.DBKEY=DBACCESS
mysql.DBACCESS.name=DBNAME
mysql.DBACCESS.user=DBUSER
mysql.DBACCESS.password=PASSWORD
```

DBNAME

DBNAME ist der Name der MariaDB-Datenbank.

DBUSER

DBUSER ist der Anmeldename, mit dem sich ein Anwender mit dem Kennwort *PASSWORD* an der Datenbank *DBNAME* anmelden kann.

PASSWORD

PASSWORD ist das Kennwort zu dem Anmeldenamen *DBUSER* für die Datenbank *DBNAME*.

DBACCESS

DBACCESS ist eine Bezeichnung für den Datenbank-Zugang, bestehend aus der Kombination von Datenbanknamen *DBNAME*, Anmeldenamen *DBUSER* und Kennwort *PASSWORD*.

Wenn die *DBNAME*-Zeile in der Geheimnis-Datei fehlt, dann ist *DBACCESS* gleichzeitig der Datenbankname.

DBKEY

DBKEY ist ein Schlüssel, dem ein Datenbank-Zugang zugeordnet ist.

Funktion

Das Modul `Herbaer::MysqlAccess` exportiert die eine Funktion `get_database`.

```
my $dba = get_database ($dbkey, $secrets, $verbose);
my $db_handle = $dba -> [0];
my $db_name   = $dba -> [1];
```

Das Ergebnis ist eine Liste (ARRAY-Ref) aus einem DBI-Datenbank-Handle und dem Datenbanknamen zu dem Datenbank-Schlüssel *$dbkey*. Im Falle eines Fehlers ist das Ergebnis undefiniert.

$dbkey

Der Schlüssel zu dem Datenbank-Zugang.

$secrets (optional)

Der Dateipfad der Geheimnis-Datei.

Wenn dieser Parameter nicht angegeben ist, sollte die Modul-Datei `MysqlAccess.pm` in oder unter dem Verzeichnis *KLEIDER*`/catalog` liegen. Der Teilpfad ab `/catalog/` wird durch `/web/secrets` ersetzt.

Das Thema Datenbanken gehört eher zum Kleider-Katalog als zur Website, aber die „Geheimnis”-Datei habe ich im Zusammenhang mit der Website entworfen. So sind die Dateipfade zu erklären.

`$verbose` (optional)

Wenn der Wert logisch wahr ist (Voreinstellung), dann werden Fehlermeldungen nach STDERR ausgegeben, wenn die erwarteten Einträge in der Geheimnis-Datei fehlen.

Benutzte Module

`DBI` für den Datenbank-Zugang.

Anwendung

Das Skript `dbdump.pl` benutzt dieses Modul `Herbaer::MysqlAccess`, ebenso das Skript, das von der Transformation `dump_insert.xslt` erzeugt wird.

Quelltext

[Beschreibung]

```
# Zugang zu einer Mysql-Datenbank über secrets-Datei
# 2020-12-06 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::MysqlAccess ;


BEGIN {
   use Cwd qw(realpath);
   use DBI;
   use Exporter;
   our $VERSION    = 20201206;
   our @ISA        = qw (Exporter);
   our @EXPORT     = qw (get_database);
   our @EXPORT_OK  = qw (get_database);
}


sub get_database {
   my ($dbkey, $secrets, $verbose) = @_;
   $verbose //= 1;
   if (!$secrets) {
      $secrets = realpath($INC{"Herbaer/MysqlAccess.pm"});
      $secrets =~ s/\/catalog\/.*$//;
      $secrets = "$secrets/web/secrets";
   }
   my $hnd;
   if (! open ($hnd, "<:encoding(utf-8)", $secrets)) {
      print STDERR "Kann Datei \"$secrets\" nicht lesen:$!\n" if $verbose;
      return undef;
   }
   my $line;
   my $dbname;
   my $dbn;
   my $user;
   my $password;
   while (defined ($line = <$hnd>)) {
      $line =~ s/\s*$//;
      if ( $line =~ /^\s*dbkey.mysql.([a-z0-9]+)\s*=\s*(.+)/ ) {
         if ($1 eq $dbkey) {
            $dbname = $2;
         }
      }
      elsif ($line =~ /^\s*mysql\.([a-z0-9]+)\.([a-z]+)\s*=\s*(.+)/) {
         if ($dbname && $1 eq $dbname) {
            if ($2 eq "name") {
               $dbn = $3;
            }
            if ($2 eq "user") {
               $user = $3;
            }
            elsif ($2 eq "password") {
               $password = $3;
            }
         }
      }
   }
   close $hnd;
   $dbn ||= $dbname;
   if (!$dbn) {
      print STDERR "Datenbank nicht gefunden: dbkey $dbkey\n" if $verbose;
      return undef;
   }
   if (!$user) {
      print STDERR "Datenbank-User nicht gefunden: dbname $dbname\n" if $verbose;
      return undef;
   }
   if (!$password) {
      print STDERR "Datenbank-Kennwort nicht gefunden: dbname $dbname\n" if $verbose;
      return undef;
   }
   my $dbh = DBI -> connect ("DBI:mysql:$dbn", $user, $password);
   if (!$dbh) {
      print STDERR "Keine Verbindung zur MySQL-Datenbank $dbn\n" if $verbose;
      return undef;
   }
   [$dbh, $dbn];
} # get_database


1;
```

dbdump.pl

[Quelltext]

Übersicht

`dbdump.pl --help | --version`

`dbdump.pl [ --verbose ... | --no_verbose ]`
`[ --spec | --no_spec ][ --contents | --no_contents ]`
`[ --dbkey` *`DBKEY`* `][ --secrets` *`SECRETS`* `][ --dumpdir` *`DUMPDIR`* `]`

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Erhöht den Umfang der Meldungen.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--spec`

Die Datenbankstruktur wird ausgeben (Tabellen und Felder)

`--no_spec`

Die Datenbankstruktur wird nicht ausgegeben.

`--contents`

Der Inhalt der Datenbank wird ausgegeben.

`--no_contents`

Der Inhalt der Datenbank wird nicht ausgegeben.

`--dbkey` *`DBKEY`*

Der Schlüssel zum Datenbank-Zugang, s. `MysqlAccess.pm`.

`--secrets` *`SECRETS`*

Der Dateipfad der Geheimnisdatei, optional, s. `MysqlAccess.pm`.

`--dumpdir` *`DUMPDIR`*

Das Ausgabeverzeichnis, optional. Das Skript erstellt die Ausgabedatei *`DUMPDIR/DBNAME`*`.xml`. *`DBNAME`* steht für den Namen der Datenbank.

Dieses Skript liegt in oder unter dem Verzeichnis *KLEIDER*/catalog. Wenn das Ausgabeverzeichnis nicht angegeben ist, wird der Dateipfad dieses Skripts ab /catalog durch /catalog/dbdump ersetzt.

Beschreibung

Dieses Programm gibt die Struktur (Datendefinition) und den Inhalt einer Datenbank (s. *DBKEY*) nach Maßgabe der Optionen --spec und --contents in einem XML-Format aus. Die Ausgabedatei ist *DUMPDIR*/*DBNAME*.xml. *DBNAME* steht für den Namen der Datenbank. Die Datei dbdump.rng beschreibt den verwendeten XML-Namensraum.

Für den Inhalt wird ein XML-Namensraum benutzt, der für die Datenbank spezifisch ist. Der lokale Name der umfassenden Elements für den Datenbankinhalt ist der Datenbankname. Die lokalen Namen der Kindelemente sind die Tabellennamen, die lokalen Namen dessen Kindelemente sind die Feldnamen. Der XML-Namensraum heißt http://herbaer.de/xmlns/20201201/dbcontents/*DBNAME*, s. dump_rng.xslt.

Benutzte Module

Das Programm ist mit Perl Version 5.28.1 entwickelt. Es benutzt die folgenden Module:

Cwd

Die Funktion realpath dient dazu, *DUMPDIR* zu bestimmen.

Herbaer::MysqlAccess

Die Funktion get_database liefert ein DBI-Datenbank-Handle und den Namen der Datenbank, s. MysqlAccess.pm.

Herbaer::Readargs

Die Funktionen read_args aus diesem Modul verarbeitet die Befehlszeilenargumente, die Funktion print_message_with_values gibt die Hilfe mit den aktuellen Einstellungen aus.

Herbaer::XMLDataWriter

XML-Ausgabe

MIME::Base64

Die Funktion encode_base64 kodiert den Inhalt von Datenfeldern, deren Datentyp-Bezeichnung blob oder binary enthält.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# MySQL-Datenbank im XML-Format ausgeben
# 2020-11-29, Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


use utf8;
use Cwd qw(realpath);
use Herbaer::MysqlAccess ;
use Herbaer::Readargs ;
use Herbaer::XMLDataWriter;
use MIME::Base64;


binmode (STDIN,  ":encoding(utf-8)");
binmode (STDOUT, ":encoding(utf-8)");
binmode (STDERR, ":encoding(utf-8)");


=for comment


{database}
-> dbname
-> create      -- die Anweisung zum Erstellen der Datenbank
-> tabspec  [{tabspec}]
-> contents


{tabspec}
-> tabname
-> tabtype
-> create      -- Anweisung zum Erstellen der Tabelle
-> fields  [{fldspec}]
-> index   [{ixspec}]


{fldspec}
-> name
-> type
-> collate
-> null
-> key
-> default
-> extra
-> comment


{ixspec}
-> tabname
-> key
-> seq
-> colname
-> ixtype
-> nonuni
-> collate
-> card
-> subpart
-> packed
-> null
-> comment
-> ixcomm


=cut


# Kommandozeilen-Argumente
my $args = {
   "[cnt]verbose"  => 1,
   "[cnt]spec"     => 1,
   "[cnt]contents" => 1,
   "dbkey"         => "",    # Schlüssel zu den Datenbank-Zugangsdaten
   "secrets"       => undef, # Eingabedatei (Text)
   "dumpdir"       => undef,
};


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
dbdump.pl
MySQL-Datenbank im XML-Format
2020-11-29 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
}
$args -> {"[sr]version"} = sub { version (); exit 0; };
```

```
$args -> {"[sr]help"} = sub {
   set_defaults ($args);
   version ();
   print_message_with_values (<<"HELP", $args);
$0 OPTION ...
--[no_]verbose     Umfang der Ausgabe \${[cnt]verbose}
--[no_]spec        Datenbank-Struktur ausgeben \${[cnt]spec}
--[no_]contents    Datenbank-Inhalt ausgeben \${[cnt]contents}
--dbkey   DBKEY    Schlüssel zum Datenbankzugang \${dbkey}
--dumpdir DUMPDIR  optional: Ausgabeverzeichnis
                   \${dumpdir}
--secrets SECRETS  optional: Pfad der Geheimnis-Datei
                   \${secrets}
HELP
   exit 0;
};


sub set_defaults {
   my $args = shift;
   my $b = realpath ($0);
   $b =~ s/\/catalog\/.*//;
   $args -> {"dumpdir"} ||= "$b/catalog/dbdump";
}; # set_defaults


read_args ($args);
set_defaults ($args);


my $data = {
   "dbh"         => undef,    # Datenbank-Handle
   "xmlwriter"   => undef,
   "database"    => {},       # Datenbanken s.o.
};


sub get_dbh {
   my ($args, $data) = @_;
   my $ar = get_database (
      $args -> {"dbkey"},
      $args -> {"secrets"},
      $args -> {"[cnt]verbose"}
   );
   if ($ar) {
      $data -> {"dbh"} = $ar -> [0];
      $data -> {"database"} -> {"dbname"} = $ar -> [1];
      return 1;
   }
   else {
      return 0;
   }
} # get_dbh


sub get_create_database {
   my ($args, $data) = @_;
   my $dbh    = $data  -> {"dbh"};
   my $dbase  = $data  -> {"database"};
   my $dbname = $dbase -> {"dbname"};
   my $sh  = $dbh -> prepare ("SHOW CREATE DATABASE $dbname");
   $sh -> execute ();
   my $ar = $sh -> fetchrow_arrayref();
   if ($ar && @$ar > 1) {
      $dbase -> {"create"} = $ar -> [1];
   }
   1;
} # get_create_database


sub get_tables {
   my ($args, $data) = @_;
   my $verbose = $args -> {"[cnt]verbose"};
   my $dbh   = $data  -> {"dbh"};
   my $dbase = $data  -> {"database"};
   my $ts    = $dbase -> {"tabspec"} ||= [];
   my $sh  = $dbh -> prepare ("SHOW FULL TABLES");
   my $ar;
   $sh -> execute ();
   my $tabname;
   my $rv = 0;
   while ($ar = $sh -> fetchrow_arrayref()) {
      $tabname = $ar -> [0];
      $tabtype = $ar -> [1];
      push (@$ts, {
         "tabname" => $tabname,
         "tabtype" => $tabtype,
      });
      ++$rv;
      print "Table $tabname: $tabtype\n" if $verbose;
   }
   $rv;
} # get_tables
```

```
sub get_table_status {
   my ($args, $data) = @_;
   my $dbh   = $data  -> {"dbh"};
   my $tstat = {};  # Tabellenname als Schlüssel
   my $tsd   = {};  # Daten zu einer Tabelle aus SHOW TABLE STATUS

   # Felder aus SHOW TABLE STATUS
   # https://mariadb.com/kb/en/show-table-status/
   my $ts_name;    # Name: Table name
   my $ts_engine;  # Engine: Table storage engine
   my $ts_version; # Version: Version number from the table's .frm file.
   my $ts_rf;      # Row_format: Row format (see InnoDB, Aria and MyISAM row formats)
                   # hier nicht benutzt
   my $ts_rows;    # Rows: Number of rows in the table.
                   # Some engines, such as XtraDB and InnoDB may store an estimate.
   my $ts_arl;     # Avg_row_length: Average row length in the table
                   # hier nicht benutzt
   my $ts_datalen; # Data_length
=for comment
For InnoDB/XtraDB, the index size, in pages, multiplied by the page size.
For Aria and MyISAM, length of the data file, in bytes.
For MEMORY, the approximate allocated memory.
=cut
   my $ts_maxdl;   # Max_data_length: hier nicht benutzt
=for comment
Maximum length of the data file,
ie the total number of bytes that could be stored in the table.
Not used in XtraDB and InnoDB.
=cut
   my $ts_ixlen;   # Index_length: Length of the index file.
   my $ts_df;      # Data_free
=for comment
Bytes allocated but unused.
For InnoDB tables in a shared tablespace,
the free space of the shared tablespace with small safety margin.
An estimate in the case of partitioned tables - see the PARTITIONS table.
=cut
   my $ts_autoinc; # Auto_increment: Next AUTO_INCREMENT value.
   my $ts_cretime; # Create_time: Time the table was created.
   my $ts_updtime; # Update_time:
=for comment
Time the table was last updated.
On Windows, the timestamp is not updated on update,
so MyISAM values will be inaccurate.
In InnoDB, if shared tablespaces are used, will be NULL,
while buffering can also delay the update,
so the value will differ from the actual time of the last UPDATE, INSERT or DELETE.
=cut
   my $ts_chktime; # Check_time:
=for comment
Time the table was last checked.
Not kept by all storage engines, in which case will be NULL.
=cut
   my $ts_collate; # Collation: Character set and collation.
   my $ts_chksum;  # Checksum: Live checksum value, if any.
                   # hier nicht benutzt
   my $ts_creopt;  # Create_options: Extra CREATE TABLE options.
   my $ts_comment; # Comment: Table comment provided when MariaDB created the table.
=for comment
Weitere Felder werden hier nicht benutzt:
Max_index_length:
Maximum index length (supported by MyISAM and Aria tables). Added in MariaDB 10.3.5.
Temporary:
Placeholder to signal that a table is a temporary table.
Currently always "N",
except "Y" for generated information_schema tables and NULL for views.
Added in MariaDB 10.3.5.
=cut
   my $sh  = $dbh -> prepare ("SHOW TABLE STATUS");
   my $ar;
   $sh -> execute ();

   while ($ar = $sh -> fetchrow_arrayref()) {
      (
       $ts_name,    # Name: Table name
       $ts_engine,  # Engine: Table storage engine
       $ts_version, # Version: Version number from the table's .frm file.
       $ts_rf,      # Row_format: Row format (see InnoDB, Aria and MyISAM row formats)
       $ts_rows,    # Rows: Number of rows in the table.
       $ts_arl,     # Avg_row_length: Average row length in the table
       $ts_datalen, # Data_length
       $ts_maxdl,   # Max_data_length: hier nicht benutzt
       $ts_ixlen,   # Index_length: Length of the index file.
       $ts_df,      # Data_free
       $ts_autoinc, # Auto_increment: Next AUTO_INCREMENT value.
       $ts_cretime, # Create_time: Time the table was created.
       $ts_updtime, # Update_time:
       $ts_chktime, # Check_time:
       $ts_collate, # Collation: Character set and collation.
       $ts_chksum,  # Checksum: Live checksum value, if any.
       $ts_creopt,  # Create_options: Extra CREATE TABLE options.
       $ts_comment, # Comment: Table comment provided when MariaDB created the table.
      ) = (@$ar);
      $tsd -> {$ts_name} = {
         "engine"  => $ts_engine,
         "version" => $ts_version,
```

```
         "rows"    => $ts_rows,
         "datalen" => $ts_datalen,
         "ixlen"   => $ts_ixlen,
         "df"      => $ts_df,
         "autoinc" => $ts_autoinc,
         "cretime" => $ts_cretime,
         "updtime" => $ts_updtime,
         "chktime" => $ts_chktime,
         "collate" => $ts_collate,
         "creopt"  => $ts_creopt,
         "comment" => $ts_comment,
      };
   }
   my $dbase = $data  -> {"database"};
   my $ts    = $dbase -> {"tabspec"};
   my $td;
   my ($k, $v);
   for $tab (@$ts) {
      $td = $tsd -> {$tab -> {"tabname"}};
      while ( ($k, $v) = each %$td) {
         $tab -> {$k} = $v if $v;
      }
   }
} # get_table_status


# Spalten der Datenbanktabellen
sub get_columns {
   my ($args, $data) = @_;
   my $verbose = $args -> {"[cnt]verbose"};
   my $dbh   = $data  -> {"dbh"};
   my $dbase = $data  -> {"database"};
   my $ts    = $dbase -> {"tabspec"};
   my $tab;
   my $tabname;
   my $flds;
   my $fld;
   my $sh;
   my $ar;
   my $rv = 1;   # 1 ok, 0 Problem / eine Tabelle ohne Felder
   my $fcnt;     # Zahl der Felder einer Tabelle

   # https://mariadb.com/kb/en/show-columns/
   my $fnam;    # Field / name
                # indicates the column name.
   my $ftyp;    # Type / type
                # indicates the column data type.
   my $fcoll;   # Collation / collate
                # indicates the collation for non-binary string columns,
                # or NULL for other columns.
                # NULL wird nicht gespeichert
   my $fnull;   # Null
                # contains YES if NULL values can be stored in the column, NO if not.
                # Nur im Falle YES wird 1 gespeichert.
   my $key;     # Key
                # indicates whether the column is indexed:
                # gespeichert wird p für PRI, u für UNI, m für MUL

=for comment
If Key is empty, the column either is not indexed
or is indexed only as a secondary column in a multiple-column, non-unique index.
If Key is PRI, the column is a PRIMARY KEY
or is one of the columns in a multiple-column PRIMARY KEY.
If Key is UNI, the column is the first column of a unique-valued index
that cannot contain NULL values.
If Key is MUL, multiple occurrences of a given value are allowed within the column.
The column is the first column of a non-unique index
or a unique-valued index that can contain NULL values.

If more than one of the Key values applies to a given column of a table,
Key displays the one with the highest priority, in the order PRI, UNI, MUL.

A UNIQUE index may be displayed as PRI
if it cannot contain NULL values and there is no PRIMARY KEY in the table.
A UNIQUE index may display as MUL if several columns form a composite UNIQUE index;
although the combination of the columns is unique,
each column can still hold multiple occurrences of a given value.
=cut

   my $fdef;    # Default
                # indicates the default value that is assigned to the column.
   my $fx;      # Extra

=for comment
Any additional information that is available about a given column.

AUTO_INCREMENT    The column was created with the AUTO_INCREMENT keyword.
PERSISTENT        The column was created with the PERSISTENT keyword. (New in 5.3)
VIRTUAL           The column was created with the VIRTUAL keyword. (New in 5.3)
CURRENT_TIMESTAMP The column is a TIMESTAMP column that is automatically updated on INSERT and UPDATE.
=cut

   my $fpriv;   # Privileges
                # the privileges you have for the column.
   my $fcomm;   # Comment
                # any comment the column has.
```

```
   for $tab (@$ts) {
      $tabname = $tab -> {"tabname"};
      $flds = [];
      $tab -> {"fields"} = $flds;
      $sh  = $dbh -> prepare ("SHOW FULL COLUMNS FROM $tabname");
      $sh -> execute ();
      $fcnt = 0;
      while ($ar = $sh -> fetchrow_arrayref()) {
         ++$fcnt;
         ($fnam, $ftyp, $fcoll, $fnull, $fkey, $fdef, $fx, $fpriv, $fcomm) = (@$ar);
         $fld = {
            "name" => $fnam,
            "type" => lc ($ftyp),
         };
         push (@$flds, $fld);
         $fld -> {"collate"} = $fcoll  if $fcoll && $fcoll ne "NULL";
         $fld -> {"null"}    = 1       if $fnull eq "YES";
         if ($fkey) {
            $fld -> {"key"}  = "p"     if $fkey  eq "PRI";
            $fld -> {"key"}  = "u"     if $fkey  eq "UNI";
            $fld -> {"key"}  = "m"     if $fkey  eq "MUL";
         }
         $fld -> {"default"} = $fdef   if $fdef  && $fdef  ne "NULL";
         $fld -> {"extra"}   = lc($fx) if $fx;
         $fld -> {"comment"} = $fcomm  if $fcomm;
         # Datentypen s. https://mariadb.com/kb/en/data-types/
         # "encode" wird nur intern benutzt (evtl. auch ausgeben?)
         if ( $ftyp =~ /blob|binary/i ) {
            $fld -> {"encode"} = "b64";  # Base64-Kodierung für Binär-Daten
         }
         elsif ( $ftyp =~ /text|char/i ) {
            $fld -> {"encode"} = "esc";  # reservierte XML-Zeichen schützen
         }
      }
      $rv = 0 if !$fcnt;
   }
   $rv;
} # get_columns


# SQL-Anweisung zum Erstellen der Tabellen
sub get_create_table {
   my ($args, $data) = @_;
   my $dbh   = $data  -> {"dbh"};
   my $dbase = $data  -> {"database"};
   my $ts    = $dbase -> {"tabspec"};
   my $tab;
   my $sh;
   my $ar;
   for $tab (@$ts) {
      next if $tab -> {"tabtype"} ne "BASE TABLE";
      $tabname = $tab -> {"tabname"};
      $sh  = $dbh -> prepare ("SHOW CREATE TABLE $tabname");
      $sh -> execute ();
      $ar = $sh -> fetchrow_arrayref();
      if ($ar && @$ar > 1) {
         $tab -> {"create"} = $ar -> [1];
      }
   }
   1;
} # get_create_table


sub get_index {
   my ($args, $data) = @_;
   my $dbh   = $data  -> {"dbh"};
   my $dbase = $data  -> {"database"};
   my $ts    = $dbase -> {"tabspec"};
   my $tab;
   my $sh;
   my $ar;
   my $ixs;     # Liste der Index-Einträge
   my $ix;      # ein Index-Eintrag

                # https://mariadb.com/kb/en/show-index/
   my $tabname; # Table
                # Table name
   my $nonuni;  # Non_unique
                # 1 if the index permits duplicate values, 0 if values must be unique.
                # Nur 1 wird gespeichert
   my $key;     # Key_name
                # Index name. The primary key is always named PRIMARY.
   my $seq;     # Seq_in_index
                # The column's sequence in the index, beginning with 1.
   my $colname; # Column_name
                # Column name.
   my $collate; # Collation
                # Either A, if the column is sorted in ascending order in the index,
                # or NULL if it's not sorted.
   my $card;    # Cardinality
                # Estimated number of unique values in the index.
                # The cardinality statistics are calculated at various times,
                # and can help the optimizer make improved decisions.
   my $subpart; # Sub_part
                # NULL if the entire column is included in the index,
                # or the number of included characters if not.
                # NULL wird nicht gespeichert
```

```
   my $packed;  # Packed
                # NULL if the index is not packed, otherwise how the index is packed.
                # NULL wird nicht gespeichert
   my $null;    # Null
                # NULL if NULL values are permitted in the column,
                # an empty string if NULL's are not permitted.
                # Im Falle NULL wird 1 gespeichert, sonst nichts.
   my $ixtype;  # Index_type
                # The index type, which can be BTREE, FULLTEXT, HASH or RTREE.
                # See Storage Engine Index Types.
   my $comment; # Comment
                # Other information, such as whether the index is disabled.
                # Wird nur gespeichert, wenn nicht leer.
   my $ixcomm;  # Index_comment
                # Contents of the COMMENT attribute when the index was created.
                # Wird nur gespeichert, wenn nicht leer.

   for $tab (@$ts) {
      $ixs = $tab -> {"index"} ||= [];
      # next if $tab -> {"tabtype"} ne "BASE TABLE";
      $tabname = $tab -> {"tabname"};
      $sh  = $dbh -> prepare ("SHOW INDEX FROM $tabname");
      $sh -> execute ();
      while ($ar = $sh -> fetchrow_arrayref()) {
         ($tabname,
          $nonuni,
          $key,
          $seq,
          $colname,
          $collate,
          $card,
          $subpart,
          $packed,
          $null,
          $ixtype,
          $comment,
          $ixcomm) = (@$ar);
         $ix = {
            "tabname" => $tabname,
            "key"     => $key,
            "seq"     => $seq,
            "colname" => $colname,
            "ixtype"  => $ixtype,
         };
         $ix -> {"nonuni"}  = $nonuni  if $nonuni;
         $ix -> {"collate"} = $collate if $collate;
         $ix -> {"card"}    = $card    if $card;
         $ix -> {"subpart"} = $subpart if $subpart && $subpart ne "NULL";
         $ix -> {"packed"}  = $packed  if $packed  && $packed  ne "NULL";
         $ix -> {"null"}    = 1        if $null    && $null    eq "NULL";
         $ix -> {"comment"} = $comment if $comment;
         $ix -> {"ixcomm"}  = $ixcomm  if $ixcomm;
         push (@$ixs, $ix);
      }
   }
   1;
} # get_index


my $xmlopt = {
   '@fields' => ["", "field"],
   '$encode' => "IGNORE",
};


sub write_spec {
   my ($args, $data) = @_;
   my $verbose = $args -> {"[cnt]verbose"};
   my $dbase   = $data -> {"database"};
   my $dn      = $dbase -> {"dbname"};
   my $file    = join ("", $args -> {"dumpdir"}, "/", "$dn.xml");
   my $xdw     = $data -> {"xmlwriter"} ||= Herbaer::XMLDataWriter -> new ($xmlopt);
   $xdw -> open (
      $file,
      "utf-8", "http://herbaer.de/xmlns/20201201/dbdump",
      "${dn}_ht.xslt"
   ) || do {
      print STDERR "Kann Datei $file nicht erstellen\n" if $verbose;
      return;
   };
   if ($args -> {"[cnt]spec"}) {
      $xdw -> open_element ("dbdump");
      $xdw -> write ("dbname", {}, $dbase -> {"dbname"});
      $xdw -> write ("create", {}, $dbase -> {"create"});
      $xdw -> write ("table",  {}, $dbase -> {"tabspec"});
   }
} # write_spec


sub write_end {
   my ($args, $data) = @_;
   my $xdw     = $data -> {"xmlwriter"};
   $xdw -> close () if $xdw;
} # write_end
```

```
# Datenbank-Inhalt lesen und schreiben
sub get_write_contents {
   my ($args, $data) = @_;
   my $dbh    = $data -> {"dbh"};
   my $xdw    = $data -> {"xmlwriter"};
   my $dbase  = $data -> {"database"};
   my $dbname = $dbase -> {"dbname"};
   $xdw -> open_element ("contents");
   $xdw -> open_element (
      $dbname,
      {"xmlns" => "http://herbaer.de/xmlns/20201201/dbcontents/$dbname"}
      );
   my $ts    = $dbase -> {"tabspec"};
   my $tab;
   my $tabname;
   my $fields;
   my $sh;
   my $ar;
   my $i;
   my $d;     # ein Datum
   my $enc;
   for $tab (@$ts) {
      $tabname = $tab -> {"tabname"};
      $fields  = $tab -> {"fields"};
      $sh  = $dbh -> prepare ("SELECT * FROM $tabname");
      $sh -> execute();
      while ($ar = $sh -> fetchrow_arrayref()) {
         $xdw -> open_element ($tabname);
         for ($i = 0; $i < @$ar; ++$i) {
            next unless defined $ar -> [$i];
            $d = $ar -> [$i];
            if ( $enc = $fields -> [$i] -> {"encode"} ) {
               if ($enc eq "b64") {
                  $d = encode_base64 ($d);
               }
               elsif ($enc eq "esc") {
                  $d =~ s/&/&amp;/g;
                  $d =~ s/</&lt;</g;
                  $d =~ s/>/&gt;</g;
               }
            }
            $xdw -> write ($fields -> [$i] -> {"name"}, {}, $d);
         }
         $xdw -> close_element ($tabname);
      }
   }
} # get_write_contents


get_dbh             ($args, $data) || exit 1;
get_create_database ($args, $data);
get_tables          ($args, $data) || exit 2;
get_create_table    ($args, $data);
get_table_status    ($args, $data);
get_columns         ($args, $data) || exit 3;
get_index           ($args, $data);
write_spec          ($args, $data);  # Datenbankstruktur wird ausgegeben
# Datenbank-Inhalte werden gelesen und ausgegeben
get_write_contents  ($args, $data) if $args -> {"[cnt]contents"};
write_end           ($args, $data);  # Dump-Datei (XMLWriter) wird geschlossen
```

dbname.pl

[Quelltext]

Übersicht

`dbname.pl --help | --version`

`dbname=$(dbname.pl DBKEY)`

Argumente

`--help , --version`

Gibt einen Hinweis zur Version und zur Anwendung aus.

`DBKEY`

`DBKEY` ist der Schlüssel zum Datenbank-Zugang. s. `MysqlAccess.pm`. Dieses Skript (`dbname.pl`) liest den Namen der zugehörigen Datenbank aus der Geheimnisdatei *`KLEIDER`*`/web/secrets` und gibt ihn aus.

Dieses Skript muss in oder unter dem Verzeichnis *`KLEIDER`*`/catalog/` liegen.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# Name einer MariaDB-Datenbank anhand des DBKEY
# 2020-12-10 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


use Cwd qw(realpath);
sub help {
   print <<"HELP";
dbdump/dbname.pl
Name einer MariaDB-Datenbank anhand des DBKEY
2020-12-10 Herbert Schiemann <h.schiemann\@herbaer.de>
dbname=\$($0 DBKEY)
HELP
   exit 0;
}


help () if @ARGV != 1;


my $dbkey = shift (@ARGV);
help () if $dbkey eq "--version" or $dbkey eq "--help";


my $secrets = realpath($0);
$secrets =~ s/\/catalog\/.*$//;
$secrets = "$secrets/web/secrets";


my $hnd;
if (! open ($hnd, "<:encoding(utf-8)", $secrets)) {
   print STDERR "Kann Datei \"$secrets\" nicht lesen:$!\n";
   exit 1;
}
my $line;
my $dbname;
my $dbn;
while (defined ($line = <$hnd>)) {
   $line =~ s/\s*$//;
   if ( $line =~ /^\s*dbkey.mysql.([a-z0-9]+)\s*=\s*(.+)/ ) {
      $dbname = $2 if ($1 eq $dbkey);
   }
   elsif ($line =~ /^\s*mysql\.([a-z0-9]+)\.name\s*=\s*(.+)/) {
      $dbn = $2 if ($dbname && $1 eq $dbname);
   }
}
close $hnd;
$dbn ||= $dbname;
if (!$dbn) {
   print STDERR "Datenbank nicht gefunden: dbkey $dbkey\n";
   exit 1;
}


print $dbn;
```

dbdump.rng - Struktur und Inhalt einer Datenbank

Namespace	http://herbaer.de/xmlns/20201201/dbdump
Wurzelelement	dbdump
(anything)	Beliebiger Inhalt
	Enthält: (anything) (*)
	Enthalten in: (anything), (foreign_el)
(foreign_att)	Attribute anderer XML-Namensräume
	Enthalten in: dbdump, dbname, create, table, tabname, tabtype, create, engine, version, rows, datalen, ixlen, df, autoinc, cretime, updtime, chktime, collate, creopt, comment, field, name, type, collate, null, key, default, extra, comment, label, index, tabname, key, seq, colname, ixtype, nonuni, collate, card, subpart, packed, null, comment, ixcomm, contents
(foreign_el)	Elemente anderer XML-Namensräume
	Enthält: (anything) (*)
	Enthalten in: dbdump, table, field, index, contents
dbdump	Das XML-Wurzelelement des Dokuments
	Es enthält Daten zur Struktur und zum Inhalt einer MariaDB-Datenbank. Die hier definierten Elemente beschreiben im wesentlichen die Struktur einer Datenbank: die Tabellen, die Felder und Indizes.
	Enthält: (foreign_att), dbname, create (?), table (+), contents (?), (foreign_el)
	Enthalten in: Wurzel

```
<element name="dbdump">
  <ref name="foreign_att"/>
  <interleave>
    <ref name="el_dbname"/>
    <optional>
        <ref name="el_create"/>
    </optional>
    <oneOrMore>
        <ref name="el_table"/>
    </oneOrMore>
    <optional>
        <ref name="el_contents"/>
    </optional>
    <ref name="foreign_el"/>
  </interleave>
</element>
```

dbname	Der Name der Datenbank
	Enthält: Datentyp word
	Enthalten in: dbdump

```
<element name="dbname">
  <ref name="foreign_att"/>
  <data type="word"/>
</element>
```

create	Eine SQL-Anweisung, die die Datenbank anlegt. Sie ist das Ergebnis des SQL-Befehls `SHOW CREATE DATABASE`. Siehe `https://mariadb.com/kb/en/show-create-database/`
	Enthält: Text, (foreign_att)
	Enthalten in: dbdump

```
<element name="create">
  <ref name="foreign_att"/>
  <text/>
</element>
```

table — Beschreibt eine Tabelle.

Die Elemente entsprechen den Zeilen der Ausgabe des SQL-Befehls `SHOW FULL TABLES`, siehe `https://mariadb.com/kb/en/show-tables/`

Enthält: (foreign_att), tabname, tabtype (?), create (?), engine (?), version (?), rows (?), datalen (?), ixlen (?), df (?), autoinc (?), cretime (?), updtime (?), chktime (?), collate (?), creopt (?), comment (?), field (+), index (+), (foreign_el)

Enthalten in: dbdump

```
<element name="table">
  <ref name="foreign_att"/>
  <interleave>
    <ref name="el_tabname"/>
    <optional>
        <ref name="el_tabtype"/>
    </optional>
    <optional>
        <ref name="el_tab_create"/>
    </optional>
    <optional>
        <ref name="el_tab_engine"/>
    </optional>
    <optional>
        <ref name="el_tab_version"/>
    </optional>
    <optional>
        <ref name="el_tab_rows"/>
    </optional>
    <optional>
        <ref name="el_tab_datalen"/>
    </optional>
    <optional>
        <ref name="el_tab_ixlen"/>
    </optional>
    <optional>
        <ref name="el_tab_df"/>
    </optional>
    <optional>
        <ref name="el_tab_autoinc"/>
    </optional>
    <optional>
        <ref name="el_tab_cretime"/>
    </optional>
    <optional>
        <ref name="el_tab_updtime"/>
    </optional>
    <optional>
        <ref name="el_tab_chktime"/>
    </optional>
    <optional>
        <ref name="el_tab_collate"/>
    </optional>
    <optional>
        <ref name="el_tab_creopt"/>
    </optional>
    <optional>
        <ref name="el_tab_comment"/>
    </optional>
    <oneOrMore>
        <ref name="el_field"/>
    </oneOrMore>
    <oneOrMore>
        <ref name="el_index"/>
    </oneOrMore>
    <ref name="foreign_el"/>
  </interleave>
</element>
```

tabname — Der Name einer Tabelle

Enthält: Datentyp word

Enthalten in: table

```
<element name="tabname">
  <ref name="foreign_att"/>
  <data type="word"/>
</element>
```

tabtype — Typ einer Tabelle

Erlaubte Werte: "BASE TABLE", "VIEW", "SEQUENCE"

Enthält: (foreign_att)

Enthalten in: table

```
<element name="tabtype">
  <ref name="foreign_att"/>
  <choice>
    <value>BASE TABLE</value>
    <value>VIEW</value>
    <value>SEQUENCE</value>
  </choice>
</element>
```

create — SQL-Anweisung, die die Tabelle erzeugt. Der SQL-Befehl `SHOW CREATE TABLE` ergibt diese Anweisung, s. `https://mariadb.com/kb/en/show-create-table/`.

Enthält: Text, (foreign_att)

Enthalten in: table

```
<element name="create">
  <ref name="foreign_att"/>
  <text/>
</element>
```

engine — Table storage engine

Enthält: Text, (foreign_att)

Enthalten in: table

```
<element name="engine">
  <ref name="foreign_att"/>
  <text/>
</element>
```

version — Version number from the table's .frm file.

Enthält: Text, (foreign_att)

Enthalten in: table

```
<element name="version">
  <ref name="foreign_att"/>
  <text/>
</element>
```

rows — Number of rows in the table. Some engines, such as XtraDB and InnoDB may store an estimate.

Enthält: Datentyp integer

Enthalten in: table

```
<element name="rows">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

datalen — For InnoDB/XtraDB, the index size, in pages, multiplied by the page size. For Aria and MyISAM, length of the data file, in bytes. For MEMORY, the approximate allocated memory.

Enthält: Datentyp integer

Enthalten in: table

```
<element name="datalen">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

ixlen — Length of the index file.

Enthält: Datentyp integer

Enthalten in: table

```
<element name="ixlen">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

df — Bytes allocated but unused. For InnoDB tables in a shared tablespace, the free space of the shared tablespace with small safety margin. An estimate in the case of partitioned tables - see the `PARTITIONS` table.

Enthält: Datentyp integer

Enthalten in: table

```
<element name="df">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

autoinc — Der nächste `AUTO_INCREMENT`-Wert

Enthält: Datentyp integer

Enthalten in: table

```
<element name="autoinc">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

cretime — Zeit, zu der die Tabelle angelegt wurde

Enthält: Datentyp datetime

Enthalten in: table

```
<element name="cretime">
  <ref name="foreign_att"/>
  <data type="datetime"/>
</element>
```

updtime — Time the table was last updated. On Windows, the timestamp is not updated on update, so MyISAM values will be inaccurate. In InnoDB, if shared tablespaces are used, will be `NULL`, while buffering can also delay the update, so the value will differ from the actual time of the last `UPDATE`, `INSERT` or `DELETE`.

Enthält: Datentyp datetime

Enthalten in: table

```
<element name="updtime">
  <ref name="foreign_att"/>
  <data type="datetime"/>
</element>
```

chktime — Time the table was last checked. Not kept by all storage engines, in which case will be NULL.

Enthält: Datentyp datetime

Enthalten in: table

```
<element name="chktime">
  <ref name="foreign_att"/>
  <data type="datetime"/>
</element>
```

collate — Character set and collation.

Enthält: Datentyp string

Enthalten in: table

```
<element name="collate">
  <ref name="foreign_att"/>
  <data type="string"/>
</element>
```

creopt — Weitere Optionen zu `CREATE TABLE`.

Enthält: Text, (foreign_att)

Enthalten in: table

```
<element name="creopt">
  <ref name="foreign_att"/>
  <text/>
</element>
```

comment — Kommentar zum Anlegen der Tabelle

Enthält: Text, (foreign_att)

Enthalten in: table

```
<element name="comment">
  <ref name="foreign_att"/>
  <text/>
</element>
```

field — Beschreibt ein Feld einer Tabelle.

Jedes Element entspricht einer Zeile der Ausgabe des SQL-Befehls `SHOW FULL COLUMNS FROM tabname`, s. `https://mariadb.com/kb/en/show-columns/`.

Enthält: (foreign_att), name, type, collate (?), null (?), key (?), default (?), extra (?), comment (?), label (?), (foreign_el)

Enthalten in: table

```
<element name="field">
  <ref name="foreign_att"/>
  <interleave>
    <ref name="el_field_name"/>
    <ref name="el_field_type"/>
    <optional>
        <ref name="el_field_collate"/>
    </optional>
    <optional>
        <ref name="el_field_null"/>
    </optional>
    <optional>
        <ref name="el_field_key"/>
    </optional>
    <optional>
        <ref name="el_field_default"/>
    </optional>
    <optional>
        <ref name="el_field_extra"/>
    </optional>
    <optional>
        <ref name="el_field_comment"/>
    </optional>
    <optional>
        <ref name="el_field_label"/>
    </optional>
    <ref name="foreign_el"/>
  </interleave>
</element>
```

name — Der Name eines Feldes

Enthält: Datentyp word

Enthalten in: field

```
<element name="name">
  <ref name="foreign_att"/>
  <data type="word"/>
</element>
```

type — Der Datentyp eines Feldes in Kleinbuchstaben.

Enthält: Datentyp string

Enthalten in: field

```
<element name="type">
  <ref name="foreign_att"/>
  <data type="string"/>
</element>
```

collate — Die Sortierfolge für nicht-binäre Zeichenketten-Felder.

Enthält: Datentyp word

Enthalten in: field

```
<element name="collate">
  <ref name="foreign_att"/>
  <data type="word"/>
</element>
```

null — Dieses Element zeigt an, dass das Feld auch keinen Wert (`NULL`) enthalten kann. Der SQL-Befehl `SHOW FULL COLUMS` zeigt in diesem Fall in der Spalte `Null` den Wert `YES`.

Enthält: (foreign_att)

Enthalten in: field

```
<element name="null">
  <ref name="foreign_att"/>
  <value>1</value>
</element>
```

key — Dieses Element zeigt an, dass das Feld Bestandteil eines Index ist.

`p`

Die Spalte ist Teil des primären Index.

`u`

Das Feld ist die erste Komponente eines eindeutigen Index und darf nicht `NULL` sein.

`m`

Das Feld ist die erste Komponente eines nicht-eindeutigen Index oder eines eindeutigen Index, der `NULL`-Werte enthalten kann.

Erlaubte Werte: "p", "u", "m"

Enthält: (foreign_att)

Enthalten in: field

```
<element name="key">
  <ref name="foreign_att"/>
  <choice>
    <value>p</value>
    <value>u</value>
    <value>m</value>
  </choice>
</element>
```

default — Der Default-Wert des Feldes.

Enthält: Text, (foreign_att)

Enthalten in: field

```
<element name="default">
  <ref name="foreign_att"/>
  <text/>
</element>
```

extra — Zusätzliche Information zum Feld. Das Feld kann unter anderen Wörtern die folgenden Wörter enthalten:

`auto_increment`

Das Feld ist ein AUTO_INCREMENT - Feld.

`persistent`

Das Feld ist mit dem Schlüsselwort `PERSISTENT` angelegt.

`virtual`

Das Feld ist mit dem Schlüsselwort `VIRTUAL` angelegt.

`current_timestamp`

Das Feld ist ein Zeitstempel und wird automatisch bei jedem Einfügen oder Aktualisieren aktualisiert.

Enthält: Text, (foreign_att)

Enthalten in: field

```
<element name="extra">
  <ref name="foreign_att"/>
  <text/>
</element>
```

comment — Eine Anmerkung zum Feld

Enthält: Text, (foreign_att)

Enthalten in: field

```
<element name="comment">
  <ref name="foreign_att"/>
  <text/>
</element>
```

label — Eine kurze Beschriftung des Feldes zur Ausgabe. MariaDB kennt keine Spaltenbeschriftung, daher ist dieses Element niemals in der Ausgabe des Skripts `dbdump.pl` enthalten. In einer MariaDB - Datenbank kann der Spaltenkommentar als Spaltenbeschriftung dienen.

Enthält: Text, (foreign_att)

Enthalten in: field

```
<element name="label">
  <ref name="foreign_att"/>
  <text/>
</element>
```

index — Die Index-Felder der Tabelle. Jedes Element entspricht einer Zeile der Ausgabe des SQL-Befehls `SHOW INDEX FROM tabname`, s. https://mariadb.com/kb/en/show-index/

Enthält: (foreign_att), tabname, key, seq, colname, ixtype (?), nonuni (?), collate (?), card (?), subpart (?), packed (?), null (?), comment (?), ixcomm (?), (foreign_el)

Enthalten in: table

```
<element name="index">
  <ref name="foreign_att"/>
  <interleave>
    <ref name="el_index_tabname"/>
    <ref name="el_index_key"/>
    <ref name="el_index_seq"/>
    <ref name="el_index_colname"/>
    <optional>
        <ref name="el_index_ixtype"/>
    </optional>
    <optional>
        <ref name="el_index_nonuni"/>
    </optional>
    <optional>
        <ref name="el_index_collate"/>
    </optional>
    <optional>
        <ref name="el_index_card"/>
    </optional>
    <optional>
        <ref name="el_index_subpart"/>
    </optional>
    <optional>
        <ref name="el_index_packed"/>
    </optional>
    <optional>
        <ref name="el_index_null"/>
    </optional>
    <optional>
        <ref name="el_index_comment"/>
    </optional>
```

```
    <optional>
        <ref name="el_index_ixcomm"/>
    </optional>
    <ref name="foreign_el"/>
  </interleave>
</element>
```

tabname — Der Name der Tabelle des Index-Feldes.

Im Falle eines `VIEW` ist der Tabellenname anders als der `VIEW`-Name.

Enthält: Datentyp word

Enthalten in: index

```
<element name="tabname">
  <ref name="foreign_att"/>
  <data type="word"/>
</element>
```

key — Der Name des Index. Der primäre Index hat immer den Namen `PRIMARY`.

Enthält: Datentyp word

Enthalten in: index

```
<element name="key">
  <ref name="foreign_att"/>
  <data type="word"/>
</element>
```

seq — Die Position des Feldes im Index, beginnend mit `1`.

Enthält: Datentyp integer

Enthalten in: index

```
<element name="seq">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

colname — Der Name des Feldes

Enthält: Datentyp word

Enthalten in: index

```
<element name="colname">
  <ref name="foreign_att"/>
  <data type="word"/>
</element>
```

ixtype — Typ des Index. Mögliche Werte sind

`BTREE`
`FULLTEXT`
`HASH`
`RTREE`

Erlaubte Werte: "BTREE", "FULLTEXT", "HASH", "RTREE"

Enthält: (foreign_att)

Enthalten in: index

```
<element name="ixtype">
  <ref name="foreign_att"/>
  <choice>
    <value>BTREE</value>
    <value>FULLTEXT</value>
    <value>HASH</value>
    <value>RTREE</value>
  </choice>
</element>
```

nonuni — Dieses Element zeigt an, dass der Index nicht eindeutig ist.

Enthält: (foreign_att)

Enthalten in: index

```
<element name="nonuni">
  <ref name="foreign_att"/>
  <value>1</value>
</element>
```

collate — Sortierung nach der Spalte im Index

A

Der Index ist nach aufsteigenden Werten in dieser Spalte sortiert.

NULL

Der Index ist nicht nach dem Werten dieser Spalte sortiert.

Erlaubte Werte: "A", "NULL"

Enthält: (foreign_att)

Enthalten in: index

```
<element name="collate">
  <ref name="foreign_att"/>
  <choice>
    <value>A</value>
    <value>NULL</value>
  </choice>
</element>
```

card — Geschätzte Zahl eindeutiger Werte in diesem Feld.

Enthält: Datentyp integer

Enthalten in: index

```
<element name="card">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

subpart — Die Anzahl der einbezogenen Zeichen des Feldes, falls nicht der ganze Inhalt des Feldes in den Index einbezogen ist,

Enthält: Datentyp integer

Enthalten in: index

```
<element name="subpart">
  <ref name="foreign_att"/>
  <data type="integer"/>
</element>
```

packed — Falls der Index gepackt ist, gibt dieses Element an, wie der Index gepackt ist.

Enthält: Text, (foreign_att)

Enthalten in: index

```
<element name="packed">
  <ref name="foreign_att"/>
  <text/>
</element>
```

null — Dieses Element zeigt an, dass das Feld leer (NULL) sein kann.

Enthält: (foreign_att)

Enthalten in: index

```
<element name="null">
  <ref name="foreign_att"/>
  <value>1</value>
</element>
```

comment — Weitere Information zum Index

Enthält: Text, (foreign_att)

Enthalten in: index

```
<element name="comment">
  <ref name="foreign_att"/>
  <text/>
</element>
```

ixcomm — Der Kommentar zum Anlegen des Index.

Enthält: Text, (foreign_att)

Enthalten in: index

```
<element name="ixcomm">
  <ref name="foreign_att"/>
  <text/>
</element>
```

contents — Der Inhalt der Datenbank. Die Elemente, die den Inhalt der Datenbank enthalten, gehören zu einem Namensraum, der für die Datenbank spezifisch ist.

Enthält: (foreign_att), (foreign_el)

Enthalten in: dbdump

```
<element name="contents">
  <ref name="foreign_att"/>
  <ref name="foreign_el"/>
</element>
```

dump_create.xslt

[Quelltext]

Allgemeines

SQL-Skript zum Anlegen der Datenbank aus Datenbank-Dump

Erzeugt aus den Angaben zur Datenbank-Struktur in einer Datenbank-Dump-Datei ein SQL-Skript, das eine Datenbank gleicher Struktur anlegt.

Namensräume

Präfix	Namensraum
xml	http://www.w3.org/XML/1998/namespace
dd	http://herbaer.de/xmlns/20201201/dbdump
d	http://herbaer.de/xmlns/20051201/doc
xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	text
Encoding	utf-8

Parameter

Parameter p_date

Erstellungszeit

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Schlüssel

Schlüssel index

Name	index
Match	dd:table/dd:index
Use	concat (dd:tabname, '.', dd:key)

Der Schlüssel wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage dd:table

Muster-Vorlagen (matching templates)

Muster-Vorlage /dd:dbdump

Wurzelelement: Datenbank anlegen

Verwendete globale Parameter oder Variable:

Parameter p_date

Muster-Vorlage dd:table

Tabelle anlegen

Verwendete Modus:

index

Muster-Vorlage dd:table/dd:engine

Datenbank-Engine einer Tabelle

Muster-Vorlage dd:table/dd:autoinc

Nächster Auto-Increment-Wert

Muster-Vorlage dd:table/dd:collate

Sortierung

Muster-Vorlage dd:table/dd:comment

Kommentar

Muster-Vorlage dd:table/dd:field

Ein Feld einer Tabelle

Muster-Vorlage dd:field/dd:name

Name eines Feldes

Muster-Vorlage dd:field/dd:type

Datentyp eines Feldes

Muster-Vorlage dd:field/dd:default

Defaultwert eines Feldes

Muster-Vorlage dd:field/dd:extra

Weitere Optionen zu einem Feld

Muster-Vorlage dd:field/dd:comment

Kommentar zu einem Feld

Muster-Vorlage dd:table/dd:index, index

Ein Index

Muster-Vorlage dd:index/dd:ixtype

Typ eines Index

Muster-Vorlage dd:index/dd:ixcomm

Kommentar zu einem Index

Modus

Modus index

Die folgenden Vorlagen implementieren den Modus index:

Muster-Vorlage dd:table/dd:index, index

Der Modus index wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:table

Quelltext

[Beschreibung]

```
<?xml version = "1.0" encoding = "utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:dd  = "http://herbaer.de/xmlns/20201201/dbdump"
  version   = "1.0"
>
<!--
  SQL-Skript zum Anlegen der Datenbank aus Datenbank-Dump
  2020-12-05 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Deutschland
  GPL Version 2 oder neuer
  Jede Gewährleistung ist ausgeschlossen.
-->
<xsl:param name = "p_date"/>

<xsl:output method = "text" encoding = "utf-8"/>

<xsl:key name = "index" match = "dd:table/dd:index"
         use = "concat (dd:tabname, '.', dd:key)"/>

<xsl:template match = "/dd:dbdump">
  <xsl:variable name = "dbname" select = "dd:dbname"/>
  <xsl:text>-- automatisch erstellt </xsl:text>
  <xsl:if test = "string-length ($p_date) &gt; 0">
    <xsl:value-of select = "$p_date"/>
  </xsl:if>
  <xsl:text>
</xsl:text>
  <xsl:choose>
    <xsl:when test = "dd:create">
      <xsl:value-of select = "dd:create"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>CREATE DATABASE IF NOT EXISTS </xsl:text>
      <xsl:value-of select = "$dbname"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>;
USE </xsl:text><xsl:value-of select = "$dbname"/><xsl:text>;

</xsl:text>
  <xsl:apply-templates select = "dd:table"/>
</xsl:template>

<xsl:template match = "dd:table">
  <xsl:variable name = "tnam" select = "dd:tabname"/>
  <xsl:text>-- Tabelle </xsl:text>
  <xsl:value-of select = "$tnam"/>
  <xsl:text>
</xsl:text>
  <xsl:choose>
    <xsl:when test = "dd:create">
      <xsl:value-of select = "dd:create"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>CREATE TABLE </xsl:text>
      <xsl:value-of select = "$tnam"/>
      <xsl:text> (
  </xsl:text>
      <xsl:apply-templates select = "dd:field"/>
      <xsl:apply-templates
          select = "dd:index [
            generate-id(.) = generate-id(key('index', concat (dd:tabname, '.', dd:key)))
            ]"
          mode = "index"
        />
      <xsl:text>)
    </xsl:text>
      <xsl:apply-templates select = "dd:engine | dd:autoinc | dd:collate | dd:comment"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>;

</xsl:text>
</xsl:template>
```

```
<xsl:template match = "dd:table/dd:engine">
  <xsl:text>ENGINE </xsl:text>
  <xsl:value-of select = "."/>
  <xsl:if test = "position() &lt; last()">
    <xsl:text>,
    </xsl:text>
  </xsl:if>
</xsl:template>


<xsl:template match = "dd:table/dd:autoinc">
  <xsl:text>AUTO_INCREMENT </xsl:text>
  <xsl:value-of select = "."/>
  <xsl:if test = "position() &lt;last()">
    <xsl:text>,
    </xsl:text>
  </xsl:if>
</xsl:template>


<xsl:template match = "dd:table/dd:collate">
  <xsl:text>COLLATE </xsl:text>
  <xsl:value-of select = "."/>
  <xsl:if test = "position() &lt; last()">
    <xsl:text>,
    </xsl:text>
  </xsl:if>
</xsl:template>


<xsl:template match = "dd:table/dd:comment">
  <xsl:text>COMMENT `</xsl:text>
  <xsl:value-of select = "."/>
  <xsl:text>`</xsl:text>
  <xsl:if test = "position() &lt; last()">
    <xsl:text>,
    </xsl:text>
  </xsl:if>
</xsl:template>


<xsl:template match = "dd:table/dd:field">
  <xsl:apply-templates select = "dd:name"/>
  <xsl:apply-templates select = "dd:type"/>
  <xsl:if test = "not (dd:null)">
    <xsl:text>NOT NULL </xsl:text>
  </xsl:if>
  <xsl:apply-templates select = "dd:default"/>
  <xsl:apply-templates select = "dd:extra"/>
  <xsl:apply-templates select = "dd:comment"/>
  <xsl:if test = "position() &lt; last() or ../dd:index">
    <xsl:text>,</xsl:text>
  </xsl:if>
  <xsl:text>
  </xsl:text>
</xsl:template>


<xsl:template match = "dd:field/dd:name">
  <xsl:value-of select = "."/>
  <xsl:text> </xsl:text>
</xsl:template>


<xsl:template match = "dd:field/dd:type">
  <xsl:value-of select = "."/>
  <xsl:text> </xsl:text>
</xsl:template>


<xsl:template match = "dd:field/dd:default">
  <xsl:text>DEFAULT `</xsl:text>
  <xsl:value-of select = "."/>
  <xsl:text>` </xsl:text>
</xsl:template>


<xsl:template match = "dd:field/dd:extra">
  <xsl:value-of select = "."/>
  <xsl:text> </xsl:text>
</xsl:template>


<xsl:template match = "dd:field/dd:comment">
  <xsl:text>COMMENT `</xsl:text>
  <xsl:value-of select = "."/>
  <xsl:text>` </xsl:text>
</xsl:template>
```

```
<xsl:template match = "dd:table/dd:index" mode = "index">
  <xsl:variable name = "key" select = "dd:key"/>
  <xsl:choose>
    <xsl:when test = "dd:key = 'PRIMARY'">
      <xsl:text>PRIMARY KEY </xsl:text>
    </xsl:when>
    <xsl:when test = "dd:nonuni">
      <xsl:value-of select = "concat ('INDEX `', $key, '`')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select = "concat ('UNIQUE KEY `', $key, '`')"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:apply-templates select = "dd:ixtype"/>
  <xsl:text> (</xsl:text>
  <xsl:for-each select = "../dd:index [dd:key = $key]">
    <xsl:sort select = "dd:seq" data-type = "number"/>
    <xsl:value-of select = "concat ('`', dd:colname, '`')"/>
    <xsl:if test = "position() &lt; last()">, </xsl:if>
  </xsl:for-each>
  <xsl:text>) </xsl:text>
  <xsl:apply-templates select = "dd:ixcomm"/>
  <xsl:if test = "position() &lt; last()">,</xsl:if>
  <xsl:text>
  </xsl:text>
</xsl:template>


<xsl:template match = "dd:index/dd:ixtype">
  <xsl:value-of select = "."/>
  <xsl:text> </xsl:text>
</xsl:template>


<xsl:template match = "dd:index/dd:ixcomm">
  <xsl:text>COMMENT `</xsl:text>
  <xsl:value-of select = "."/>
  <xsl:text>` </xsl:text>
</xsl:template>


</xsl:stylesheet>
```

dump_htinc.xslt

[Quelltext]

Allgemeines

HTML-Vorlagen für Datenbank-Ansicht

Die Stylesheet-Datei, die von `dump_htview.xslt` erzeugt wird, bindet diese Datei ein.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	**Namensraum**
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://www.w3.org/1999/xhtml
*	d	http://herbaer.de/xmlns/20051201/doc
*	dd	http://herbaer.de/xmlns/20201201/dbdump
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8
Indent	yes

Eingebundene Stylesheets

/pool/path.xslt - Hilfsvorlagen zu Dateipfaden

Pfade relativ zur Stylesheet-Datei

Im allgemeinen sind relative Dateipfade absoluten Dateipfaden vorzuziehen. Wenn man eine XML-Datei im Dateisystem verschiebt, die eine xml-stylesheet-Anweisung mit einem relativen Verweis-Pfad enthält, dann ist die xml-stylesheet-Anweisung anzupassen. XSLT-Anweisungen zur HTML-Ansicht erzeugen oft Verweise auf CSS- und Javascript-Dateien. Die Pfade zu den Verweiszielen müssen ebenfalls angepasst werden. Die Lösung: die XSLT-Datei enthält Dateipfade, die relativ zu der XSLT-Datei sind, und kombiniert diese Dateipfade mit dem Dateipfad aus der xml-stylesheet-Anweisung zum Ziel-Dateipfad. Diese Datei enthält die Vorlagen zu dieser Lösung.

Zum Test dienen die Dateien *`KLEIDER`*`/catalog/src/dbdump/test_path.xml` und *`KLEIDER`*`/catalog/src/dbdump/test_path.xslt`.

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Wurzel

Aufgerufene benannte Vorlagen:

path.stylepath

Muster-Vorlage dd:dbdump

Datenbank-Struktur

Muster-Vorlage dd:table

Struktur einer Tabelle

Muster-Vorlage dd:field

Parameter

tnam

Ein Datenbank-Feld

Quelltext

[Beschreibung]

```
<?xml version = "1.0" encoding = "utf-8"?>
<?xml-stylesheet href="xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:dd  = "http://herbaer.de/xmlns/20201201/dbdump"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns     = "http://www.w3.org/1999/xhtml"
  version   = "1.0"
  exclude-result-prefixes = "d dd"
>
<!--
  HTML-Vorlagen für Datenbank-Ansicht
  2020-12-13 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Deutschland
  GPL Version 2 oder neuer
  Jede Gewährleistung ist ausgeschlossen.
-->
<xsl:output
  method   = "xml"
  encoding = "utf-8"
  indent   = "yes"
/>


<xsl:include href = "/pool/path.xslt"/>


<xsl:template match = "/">
  <xsl:variable name = "title">
    <xsl:text>Datenbank </xsl:text>
    <xsl:value-of select = "dd:dbdump/dd:dbname"/>
  </xsl:variable>
  <html>
    <head>
      <title><xsl:value-of select = "$title"/></title>
      <link rel = "stylesheet">
        <xsl:attribute name = "href">
          <xsl:call-template name = "path.stylepath">
            <xsl:with-param name = "pth">../../pool/folding.css</xsl:with-param>
          </xsl:call-template>
        </xsl:attribute>
      </link>
      <style>
thead       {font-weight: bold; }
thead tr td {cursor: pointer;   }
td + td     {padding-left: 1em; }
td.number   {text-align: right; }
      </style>
      <xsl:if test = "dd:dbdump/dd:table">
        <style>
#d_structure span.tabname     { font-weight: bold; }
#d_structure tr:first-child   {
  background-color: #dddddd ;
  cursor: n-resize ;
}
#d_structure tbody.collapsed tr:first-child { cursor: s-resize ; }
#d_structure tbody.collapsed tr + tr        { display: none; }
        </style>
      </xsl:if>
      <xsl:if test = "dd:dbdump/dd:contents/*">
        <script>
          <xsl:attribute name = "src">
            <xsl:call-template name = "path.stylepath">
              <xsl:with-param name = "pth"
              >../../web/src/addstory/tabsort.js</xsl:with-param>
            </xsl:call-template>
          </xsl:attribute>
        </script>
      </xsl:if>
      <script>
        <xsl:attribute name = "src">
          <xsl:call-template name = "path.stylepath">
            <xsl:with-param name = "pth">../../pool/folding.js</xsl:with-param>
          </xsl:call-template>
        </xsl:attribute>
      </script>
      <script>
        <xsl:text>
onload = function() {
  var f = new Folder();
  f.apply_to_doc_parent_child (document, "div",   "h2");
        </xsl:text>
        <xsl:if test = "dd:dbdump/dd:table">
  f.apply_to_doc_parent_child (document.getElementById ("d_structure"), "tbody", "tr");
        </xsl:if>
        <xsl:if test = "dd:dbdump/dd:contents/*">
  tabsort_initialize();
        </xsl:if>
        <xsl:text>
```

```
};
        </xsl:text>
      </script>
    </head>
    <body>
      <h1><xsl:value-of select = "$title"/></h1>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>


<xsl:template match = "dd:dbdump">
  <xsl:if test = "dd:table">
    <div id = "d_structure">
      <h2>Struktur</h2>
      <table>
        <xsl:apply-templates select = "dd:table"/>
      </table>
    </div>
  </xsl:if>
  <xsl:apply-templates select = "dd:contents/*"/>
</xsl:template>


<xsl:template match = "dd:table">
  <xsl:variable name = "name" select = "dd:tabname"/>
  <tbody id = "tab_{$name}">
    <xsl:apply-templates select = "dd:field">
      <xsl:with-param name = "tnam" select = "$name"/>
    </xsl:apply-templates>
  </tbody>
</xsl:template>


<xsl:template match = "dd:field">
  <xsl:param name = "tnam"/>
  <tr id = "fld_{$tnam}.{dd:name}">
    <td>
      <xsl:if test = "position() = 1">
        <span class = "tabname"><xsl:value-of select = "$tnam"/></span>
      </xsl:if>
    </td>
    <td>
      <xsl:value-of select = "dd:name"/>
    </td>
    <td>
      <xsl:value-of select = "dd:type"/>
    </td>
    <td>
      <xsl:if test = "not(dd:null)">not null</xsl:if>
    </td>
    <td>
      <xsl:choose>
        <xsl:when test = "dd:key = 'p'">primary</xsl:when>
        <xsl:when test = "dd:key = 'u'">unique</xsl:when>
        <xsl:when test = "dd:key = 'm'">index</xsl:when>
        <xsl:when test = "string-length(dd:default) &gt; 0">
          <xsl:text>Default: </xsl:text>
          <xsl:value-of select = "dd:default"/>
        </xsl:when>
        <xsl:otherwise/>
      </xsl:choose>
    </td>
  </tr>
</xsl:template>


</xsl:stylesheet>
```

dump_htview.xslt

[Quelltext]

Allgemeines

Vorlage zur HTML-Ansicht aus Datenbank-Dump

Diese Transformation erzeugt aus einem Datenbank-Dump eine XSLT-Datei zur HTML-Ansicht des Datenbank-Dump. XML-Namensräume für XSLT 1.0 mehr oder weniger fest vorgegeben. In der Ausgabedatei ist daher noch der Platzhalter *$HB_DBNAME* durch den Namen der Datenbank zu ersetzen.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://www.w3.org/1999/xhtml
	dc	http://herbaer.de/xmlns/20201201/dbcontents/$HB_DBNAME
*	dd	http://herbaer.de/xmlns/20201201/dbdump
	d	http://herbaer.de/xmlns/20051201/doc
	xsa	xsl/alias
	xsl	http://www.w3.org/1999/XSL/Transform

Namensraum-Alias

Stylesheet-Präfix	Stylesheet-URI	Ausgabe-Präfix	Ausgabe-URI
xsa	xsl/alias	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8
Indent	yes

Parameter

Parameter p_date

Erstellungszeit

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Parameter p_xslt

Stylesheet-Anweisung in der Ausgabe

Select: '../../pool/xslt_ht.xslt'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Parameter p_valuetemplates

Vorlagen für einzelne Werte ausgeben: yes/no

Select: 'no'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump
Muster-Vorlage dd:dbdump/dd:table, templ

Muster-Vorlagen (matching templates)

Muster-Vorlage /dd:dbdump

Wurzelelement

Verwendete Modus:

ref
templ

Verwendete globale Parameter oder Variable:

Parameter p_date
Parameter p_xslt
Parameter p_valuetemplates

Muster-Vorlage dd:dbdump/dd:table, ref

Vorlagen für Tabellen anwenden

Verwendete Modus:

thead

Muster-Vorlage dd:table/dd:field, class

Attribut `class` einer Tabellen-Zelle (eines Datenbankfeldes). Das erste Wort ist ein Datentyp, das zweite Wort ein Datentyp-Hinweis für die Sortierung (s.`tabsort.js`).

Muster-Vorlage dd:table/dd:field, thead

Spaltenüberschriften

Verwendete Modus:

class

Muster-Vorlage dd:dbdump/dd:table, templ

Vorlage für Tabelle definieren

Verwendete Modus:

tbody
value

Verwendete globale Parameter oder Variable:

Parameter p_valuetemplates

Muster-Vorlage dd:table/dd:field, tbody

Ein Datenbankfeld in der Tabelle

Verwendete Modus:

class

Muster-Vorlage dd:table/dd:field, value

Parameter

tnam

Tabellenname

Inhalt eines Datenbankfeldes

Modus

Modus ref

Die folgenden Vorlagen implementieren den Modus ref:

Muster-Vorlage dd:dbdump/dd:table, ref

Der Modus ref wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Modus templ

Die folgenden Vorlagen implementieren den Modus templ:

Muster-Vorlage dd:dbdump/dd:table, templ

Der Modus templ wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Modus thead

Die folgenden Vorlagen implementieren den Modus thead:

Muster-Vorlage dd:table/dd:field, thead

Der Modus thead wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:dbdump/dd:table, ref

Modus class

Die folgenden Vorlagen implementieren den Modus class:

Muster-Vorlage dd:table/dd:field, class

Der Modus class wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:table/dd:field, thead
Muster-Vorlage dd:table/dd:field, tbody

Modus tbody

Die folgenden Vorlagen implementieren den Modus tbody:

Muster-Vorlage dd:table/dd:field, tbody

Der Modus tbody wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:dbdump/dd:table, templ

Modus value

Die folgenden Vorlagen implementieren den Modus value:

Muster-Vorlage dd:table/dd:field, value

Der Modus value wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:dbdump/dd:table, templ

Quelltext

[Beschreibung]

```
<?xml version = "1.0" encoding = "utf-8"?>
<?xml-stylesheet href="xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:xsa = "xsl/alias"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:dd  = "http://herbaer.de/xmlns/20201201/dbdump"
  xmlns:dc  = "http://herbaer.de/xmlns/20201201/dbcontents/$HB_DBNAME"
  xmlns     = "http://www.w3.org/1999/xhtml"
  version   = "1.0"
  exclude-result-prefixes = "dd"
>
<!--
  Vorlage zur HTML-Ansicht aus Datenbank-Dump
  2020-12-04 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Deutschland
  GPL Version 2 oder neuer
  Jede Gewährleistung ist ausgeschlossen.
-->
<xsl:namespace-alias stylesheet-prefix = "xsa" result-prefix = "xsl"/>


<xsl:param name = "p_date"/>


<xsl:param name = "p_xslt" select = "'../../pool/xslt_ht.xslt'"/>


<xsl:param name = "p_valuetemplates" select = "'no'"/>


<xsl:output method   = "xml" encoding = "utf-8" indent = "yes"/>


<xsl:template match = "/dd:dbdump">
  <xsl:variable name = "dbname" select = "dd:dbname"/>
  <xsl:if test = "string-length ($p_xslt) &gt; 0">
    <xsl:processing-instruction name = "xml-stylesheet">
      <xsl:text>href="</xsl:text>
      <xsl:value-of select = "$p_xslt"/>
      <xsl:text>" type="application/xml"</xsl:text>
    </xsl:processing-instruction>
  </xsl:if>
  <xsa:stylesheet version = "1.0">
    <d:info xmlns="http://herbaer.de/xmlns/20051201/doc">
      <title>
        <xsl:value-of select = "$dbname"/>
        <xsl:text>_ht.xslt</xsl:text>
      </title>
      <subtitle>
        <xsl:text>HTML-Ansicht der Datenbank </xsl:text>
        <xsl:value-of select = "$dbname"/>
      </subtitle>
      <xsl:if test = "string-length ($p_date) &gt; 0">
        <date><xsl:value-of select = "$p_date"/></date>
      </xsl:if>
      <annotation>
        <para>
Automatisch erzeugt von <filename>KLEIDER/catalog/src/dbdump/dump_htview.xslt</filename>
        </para>
      </annotation>
    </d:info>
    <xsa:include href = "../src/dbdump/dump_htinc.xslt"/>
    <xsa:template match = "dc:{$dbname}">
      <xsl:apply-templates select = "dd:table" mode = "ref"/>
    </xsa:template>
    <xsl:apply-templates select = "dd:table" mode = "templ"/>
    <xsl:if test = "$p_valuetemplates != 'yes'">
      <xsa:template match = "dc:*">
        <xsa:value-of select = "."/>
      </xsa:template>
    </xsl:if>
  </xsa:stylesheet>
</xsl:template>
```

```
<xsl:template match = "dd:dbdump/dd:table" mode = "ref">
  <xsl:variable name = "tnam" select = "dd:tabname"/>
  <xsa:if test = "dc:{$tnam}">
    <div>
      <xsl:attribute name = "id">div_tab_<xsl:value-of select = "$tnam"/></xsl:attribute>
      <h2>
        <xsl:choose>
          <xsl:when test = "dd:comment">
            <xsl:value-of select = "dd:comment"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:text>Tabelle </xsl:text>
            <xsl:value-of select = "$tnam"/>
          </xsl:otherwise>
        </xsl:choose>
      </h2>
      <table class = "tabsort">
        <xsl:attribute name = "id">tab_<xsl:value-of select = "$tnam"/></xsl:attribute>
        <thead>
          <tr>
            <xsl:apply-templates select = "dd:field" mode = "thead"/>
          </tr>
        </thead>
        <tbody>
          <xsa:apply-templates select = "dc:{$tnam}"/>
        </tbody>
      </table>
    </div>
  </xsa:if>
</xsl:template>


<xsl:template match = "dd:table/dd:field" mode = "class">
  <xsl:variable name = "tp" select = "dd:type"/>
  <xsl:variable name = "cls">
    <xsl:choose>
      <xsl:when test = "starts-with ($tp, 'bool')"     >boolean text</xsl:when>
      <xsl:when test = "starts-with ($tp, 'tinyint')"  >integer number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'smallint')" >integer number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'mediumint')">integer number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'int')"      >integer number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'bigint')"   >integer number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'dec')
                     or starts-with ($tp, 'numeric')
                     or starts-with ($tp, 'fixed')
                     or starts-with ($tp, 'number')">decimal number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'float')"    >number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'double')"   >number</xsl:when>
      <xsl:when test = "starts-with ($tp, 'bit')"      >bitfield</xsl:when>
      <xsl:when test = "starts-with ($tp, 'binary')"   >binary text</xsl:when>
      <xsl:when test = "starts-with ($tp, 'datetime')" >datetime date</xsl:when>
      <xsl:when test = "starts-with ($tp, 'timestamp')">datetime date</xsl:when>
      <xsl:when test = "starts-with ($tp, 'date')"     >date</xsl:when>
      <xsl:when test = "starts-with ($tp, 'time')"     >time text</xsl:when>
      <xsl:when test = "starts-with ($tp, 'year')"     >year number</xsl:when>
      <xsl:when test = "contains ($tp, 'blob')"        >blob text</xsl:when>
      <xsl:otherwise                                   >text</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:if test = "string-length($cls) &gt; 0">
    <xsl:attribute name = "class">
      <xsl:value-of select = "$cls"/>
    </xsl:attribute>
  </xsl:if>
</xsl:template>


<xsl:template match = "dd:table/dd:field" mode = "thead">
  <td>
    <xsl:apply-templates select = "." mode = "class"/>
    <xsl:choose>
      <xsl:when test = "dd:label">
        <xsl:value-of select = "dd:label"/>
      </xsl:when>
      <xsl:when test = "dd:comment">
        <xsl:value-of select = "dd:comment"/>
      </xsl:when>
      <xsl:when test = "dd:name">
        <xsl:value-of select = "dd:name"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select = "concat ('(', position(), ')')"/>
      </xsl:otherwise>
    </xsl:choose>
  </td>
</xsl:template>
```

```
<xsl:template match = "dd:dbdump/dd:table" mode = "templ">
  <xsa:template match = "dc:{dd:tabname}">
    <tr>
      <xsl:apply-templates select = "dd:field" mode = "tbody"/>
    </tr>
  </xsa:template>
  <xsl:if test = "$p_valuetemplates = 'yes'">
    <xsl:apply-templates select = "dd:field" mode = "value">
      <xsl:with-param name = "tnam" select = "dd:tabname"/>
    </xsl:apply-templates>
  </xsl:if>
</xsl:template>


<xsl:template match = "dd:table/dd:field" mode = "tbody">
  <td>
    <xsl:apply-templates select = "." mode = "class"/>
    <xsa:apply-templates select = "dc:{dd:name}"/>
  </td>
</xsl:template>


<xsl:template match = "dd:table/dd:field" mode = "value">
  <xsl:param name = "tnam"/>
  <xsl:if test = "dd:type | dd:label | dd:comment">
    <d:para>
      <xsl:if test = "dd:type">
        <xsl:text>Datentyp: </xsl:text>
        <xsl:value-of select = "dd:type"/>
        <xsl:if test = "dd:label | dd:comment">,</xsl:if>
        <xsl:text>
</xsl:text>
      </xsl:if>
      <xsl:if test = "dd:label">
        <xsl:value-of select = "dd:label"/>
        <xsl:if test = "dd:comment">,</xsl:if>
        <xsl:text>
</xsl:text>
      </xsl:if>
      <xsl:if test = "dd:comment">
        <xsl:value-of select = "dd:comment"/>
        <xsl:text>
</xsl:text>
      </xsl:if>
    </d:para>
  </xsl:if>
  <xsa:template match = "dc:{$tnam}/dc:{dd:name}">
    <xsa:value-of select = "."/>
  </xsa:template>
</xsl:template>


</xsl:stylesheet>
```

dump_insert.xslt

[Quelltext]

Allgemeines

Perl-Skript zum Einfügen von Daten aus Datenbank-Dump

Erzeugt aus den Angaben zur Datenbank-Struktur in einer Datenbank-Dump-Datei ein Perl-Skript, das die Datenbank-Inhalte aus der (einer) Dump-Datei in eine Datenbank (gleicher Struktur) einfügt.

Namensräume

Präfix	**Namensraum**
xml	http://www.w3.org/XML/1998/namespace
(default)	http://relaxng.org/ns/structure/1.0
xl	http://www.w3.org/1999/xlink
dd	http://herbaer.de/xmlns/20201201/dbdump
d	http://herbaer.de/xmlns/20051201/doc
xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	text
Encoding	utf-8

Parameter

Parameter p_date

Erstellungszeit

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Globale Variable

Variable g_dbname

Name der Datenbank

Select: /dd:dbdump/dd:dbname

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Muster-Vorlagen (matching templates)

Muster-Vorlage /dd:dbdump

Wurzelelement: Skript ausgeben

Verwendete Modus:

sth
exec

Verwendete globale Parameter oder Variable:

Parameter p_date
Variable g_dbname

Muster-Vorlage dd:table, sth

Statement-Handle für eine Tabelle

Verwendete Modus:

list
ph

Muster-Vorlage dd:field, list

Liste der Feldnamen

Muster-Vorlage dd:field, ph

Platzhalter für Werte

Muster-Vorlage dd:table, exec

Datenzeile einfügen

Verwendete Modus:

val

Muster-Vorlage dd:field, val

Modus

Modus sth

Die folgenden Vorlagen implementieren den Modus sth:

Muster-Vorlage dd:table, sth

Der Modus sth wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Modus exec

Die folgenden Vorlagen implementieren den Modus exec:

Muster-Vorlage dd:table, exec

Der Modus exec wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Modus list

Die folgenden Vorlagen implementieren den Modus list:

Muster-Vorlage dd:field, list

Der Modus list wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:table, sth

Modus ph

Die folgenden Vorlagen implementieren den Modus ph:

Muster-Vorlage dd:field, ph

Der Modus ph wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:table, sth

Modus val

Die folgenden Vorlagen implementieren den Modus val:

Muster-Vorlage dd:field, val

Der Modus val wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage dd:table, exec

Quelltext

[Beschreibung]

```
<?xml version = "1.0" encoding = "utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:dd  = "http://herbaer.de/xmlns/20201201/dbdump"
  xmlns:xl  = "http://www.w3.org/1999/xlink"
  xmlns     = "http://relaxng.org/ns/structure/1.0"
  version   = "1.0"
>
<!--
  Perl-Skript zum Einfügen von Daten aus Datenbank-Dump
  2020-12-06 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Deutschland
  GPL Version 2 oder neuer
  Jede Gewährleistung ist ausgeschlossen.
-->
<xsl:param name = "p_date"/>


<xsl:output method = "text" encoding = "utf-8"/>


<xsl:variable name = "g_dbname" select = "/dd:dbdump/dd:dbname"/>


<xsl:template match = "/dd:dbdump">
  <xsl:text>#!/usr/bin/perl -w
# file KLEIDER/catalog/dbdump/</xsl:text>
  <xsl:value-of select = "$g_dbname"/>
  <xsl:text>_insert.pl generated </xsl:text>
  <xsl:value-of select = "$p_date"/>
  <xsl:text>


use Cwd qw(realpath);
use Herbaer::DataInserter ;
use Herbaer::MysqlAccess ;
use Herbaer::Readargs;
use MIME::Base64;
use utf8;


binmode (STDIN,  ":encoding(utf-8)");
binmode (STDOUT, ":encoding(utf-8)");
binmode (STDERR, ":encoding(utf-8)");


my $args = {
   "[cnt]verbose" => 1,
   "dbkey"        => undef,
   "dumpfile"     => undef,
};


sub version {
   print &amp;lt;&amp;lt;'VERSION' ;
Daten aus einer Dump-XML-Datei in eine Datenbank einfügen
automatisch erstellt durch dump_insert.xslt </xsl:text>
  <xsl:value-of select = "$p_date"/>
  <xsl:text>
VERSION
}
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   print_message_with_values (&amp;lt;&amp;lt;"HELP", $args);
$0 OPTION...
--[no_]verbose      Umfang der Ausgabe \${[cnt]verbose}
--dbkey    DBKEY    Schlüssel zum Datenbankzugang \${dbkey}
--dumpfile DUMPFILE Pfad der Dump-Datei
                    \${dumpfile}
HELP
   exit 0;
};


read_args   ($args);


if (!$args -> {"dumpfile"} || !$args -> {"dbkey"}) {
   $args -> {"[sr]help"} -> ($args);
   exit 1;
}
```

```
my $dbh = get_database ($args -> {"dbkey"});
exit unless $dbh;
$dbh = $dbh -> [0];
</xsl:text>
  <xsl:apply-templates select = "dd:table" mode = "sth"/>
  <xsl:if test = "//dd:type [contains (., 'blob') or contains (., 'binary')]">
sub dec_blob {
   my $v = shift;
   $v ? decode_base64 ($v) : $v;
}
  </xsl:if>
  <xsl:text>sub insert {
   my ($t, $v) = @_;
</xsl:text>
  <xsl:apply-templates select = "dd:table" mode = "exec"/>
  <xsl:text>}

Herbaer::DataInserter
   -> new ("http://herbaer.de/xmlns/20201201/dbcontents/</xsl:text>
  <xsl:value-of select = "$g_dbname"/>
  <xsl:text>", \&amp;insert, $args -> {"[cnt]verbose"})
   -> parse_file ($args -> {"dumpfile"});

# end of file KLEIDER/catalog/dbdump/</xsl:text>
  <xsl:value-of select = "$g_dbname"/>
  <xsl:text>_insert.pl generated_</xsl:text>
  <xsl:value-of select = "$p_date"/>
  <xsl:text>
</xsl:text>
</xsl:template>


<xsl:template match = "dd:table" mode = "sth">
  <xsl:variable name = "tnam" select = "dd:tabname"/>
  <xsl:text>
my $st_</xsl:text>
  <xsl:value-of select = "$tnam"/>
  <xsl:text> = $dbh -> prepare (
  "INSERT INTO </xsl:text>
  <xsl:value-of select = "$tnam"/><xsl:text> (</xsl:text>
  <xsl:apply-templates select = "dd:field" mode = "list"/>
  <xsl:text>) VALUES (</xsl:text>
  <xsl:apply-templates select = "dd:field" mode = "ph"/>
  <xsl:text>)"
   );
</xsl:text>
</xsl:template>

<xsl:template match = "dd:field" mode = "list">
  <xsl:value-of select = "dd:name"/>
  <xsl:if test = "position() &lt; last()">, </xsl:if>
</xsl:template>

<xsl:template match = "dd:field" mode = "ph">
  <xsl:text>?</xsl:text>
  <xsl:if test = "position() &lt; last()">,</xsl:if>
</xsl:template>

<xsl:template match = "dd:table" mode = "exec">
  <xsl:variable name = "tnam" select = "dd:tabname"/>
  <xsl:text>   </xsl:text>
  <xsl:if test = "position() &lt; 1">els</xsl:if>
  <xsl:text>if ($t eq "</xsl:text>
  <xsl:value-of select = "$tnam"/>
  <xsl:text>") {
      $st_</xsl:text>
  <xsl:value-of select = "$tnam"/>
  <xsl:text> -> execute (
</xsl:text>
  <xsl:apply-templates select = "dd:field" mode = "val"/>
  <xsl:text>      );
   }
</xsl:text>
</xsl:template>

<xsl:template match = "dd:field" mode = "val">
  <xsl:variable name = "fn" select = "dd:name"/>
  <xsl:choose>
    <xsl:when test = "contains (dd:type, 'blob') or contains (dd:type, 'binary')">
      <xsl:text>         dec_blob ($v -> {"</xsl:text>
      <xsl:value-of select = "$fn"/>
      <xsl:text>"})</xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>         $v -> {"</xsl:text>
      <xsl:value-of select = "$fn"/>
      <xsl:text>"}</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>,
</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

dump_rng.xslt

[Quelltext]

Allgemeines

RelaxNG aus Datenbank-Dump

Erzeugt aus den Angaben zur Datenbank-Struktur in einer Datenbank-Dump-Datei eine RelaxNG - Datei, die die XML-Elemente beschreibt, die für die Inhalte der Datenbank benutzt werden.

Der Name des Top-Level-Elements ist der Datenbank-Name, die Namen der Kind-Elemente sind die Tabellennamen, die Namen derer Kind-Elemente sind die Feldnamen.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	**Namensraum**
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://relaxng.org/ns/structure/1.0
	xl	http://www.w3.org/1999/xlink
*	dd	http://herbaer.de/xmlns/20201201/dbdump
	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8
Indent	yes

Parameter

Parameter p_date

Erstellungszeit

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Parameter p_xslt

Stylesheet-Anweisung in der Ausgabe

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /dd:dbdump

Muster-Vorlagen (matching templates)

Muster-Vorlage /dd:dbdump

Wurzelelement

Verwendete Modus:

ref
spec

Verwendete globale Parameter oder Variable:

Parameter p_date
Parameter p_xslt

Muster-Vorlage dd:dbdump/dd:table, ref

Verweise auf die Elemente, die die Tabellen spezifizieren.

Muster-Vorlage dd:dbdump/dd:table, spec

Die Spezifikation der Tabellen

Verwendete Modus:

ref
spec

Muster-Vorlage dd:table/dd:field, ref

Parameter

tnam

Verweise auf die Elemente, die die Felder einer Tabelle spezifizieren

Muster-Vorlage dd:table/dd:field, spec

Parameter

tnam

Die Felder einer Tabelle, Datentypen s. `https://mariadb.com/kb/en/data-types/`

Muster-Vorlage dd:field/dd:type

Datentyp eines Feldes in der Anmerkung

Modus

Modus ref

Die folgenden Vorlagen implementieren den Modus ref:

Muster-Vorlage dd:dbdump/dd:table, ref
Muster-Vorlage dd:table/dd:field, ref

Der Modus ref wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage /dd:dbdump
Muster-Vorlage dd:dbdump/dd:table, spec

Modus spec

Die folgenden Vorlagen implementieren den Modus spec:

Muster-Vorlage dd:dbdump/dd:table, spec
Muster-Vorlage dd:table/dd:field, spec

Der Modus spec wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage /dd:dbdump
Muster-Vorlage dd:dbdump/dd:table, spec

Quelltext

[Beschreibung]

```
<?xml version = "1.0" encoding = "utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:dd  = "http://herbaer.de/xmlns/20201201/dbdump"
  xmlns:xl  = "http://www.w3.org/1999/xlink"
  xmlns     = "http://relaxng.org/ns/structure/1.0"
  version   = "1.0"
  exclude-result-prefixes = "dd"
>
<!--
  RelaxNG-Schema aus Datenbank-Dump
  2020-12-04 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Deutschland
  GPL Version 2 oder neuer
  Jede Gewährleistung ist ausgeschlossen.
-->
<xsl:param name = "p_date"/>


<xsl:param name = "p_xslt"/>


<xsl:output method   = "xml" encoding = "utf-8" indent = "yes"/>


<xsl:template match = "/dd:dbdump">
  <xsl:variable name = "dbname" select = "dd:dbname"/>
  <xsl:variable name = "ns"
    select = "concat ('http://herbaer.de/xmlns/20201201/dbcontents/', $dbname)"
    />
  <xsl:if test = "string-length ($p_xslt) &gt; 0">
    <xsl:processing-instruction name = "xml-stylesheet">
      <xsl:text>href="</xsl:text>
      <xsl:value-of select = "$p_xslt"/>
      <xsl:text>" type="application/xml"</xsl:text>
    </xsl:processing-instruction>
  </xsl:if>
  <grammar ns = "{$ns}">
    <d:info xmlns="http://herbaer.de/xmlns/20051201/doc">
      <title>
        <xsl:text>db_</xsl:text>
        <xsl:value-of select = "$dbname"/>
        <xsl:text>.rng</xsl:text>
      </title>
      <subtitle>
        <xsl:text>Inhalt der Datenbank </xsl:text>
        <xsl:value-of select = "$dbname"/>
      </subtitle>
      <xsl:if test = "string-length ($p_date) &gt; 0">
        <date><xsl:value-of select = "$p_date"/></date>
      </xsl:if>
      <annotation>
        <para>Automatisch erzeugt</para>
      </annotation>
    </d:info>


    <start>
      <ref name = "el_{$dbname}"/>
    </start>


    <define name = "anything">
      <d:para>Beliebiger Inhalt</d:para>
      <zeroOrMore>
        <choice>
          <element>
            <anyName/>
            <ref name="anything"/>
          </element>
          <attribute>
            <anyName/>
          </attribute>
          <text/>
        </choice>
      </zeroOrMore>
    </define>
```

```
    <define name = "foreign_att">
      <d:para>Attribute anderer XML-Namensräume</d:para>
      <zeroOrMore>
        <attribute>
          <anyName>
            <except>
              <nsName ns = ""/>
              <nsName ns = "{$ns}"/>
            </except>
          </anyName>
        </attribute>
      </zeroOrMore>
    </define>


    <define name = "foreign_el">
      <d:para>Elemente anderer XML-Namensräume</d:para>
      <zeroOrMore>
        <element>
          <anyName>
            <except>
              <nsName ns = "{$ns}"/>
            </except>
          </anyName>
          <ref name = "anything"/>
        </element>
      </zeroOrMore>
    </define>


    <define name = "el_{$dbname}">
      <d:para>Das Wurzelelement: Inhalt der Datenbank</d:para>
      <element name = "{$dbname}">
        <ref name = "foreign_att"/>
        <interleave>
          <ref name = "foreign_el"/>
          <xsl:apply-templates select = "dd:table" mode = "ref"/>
        </interleave>
      </element>
    </define>


    <xsl:apply-templates select = "dd:table" mode = "spec"/>
  </grammar>
</xsl:template>


<xsl:template match = "dd:dbdump/dd:table" mode = "ref">
  <zeroormore>
    <ref name = "el_tbl_{dd:tabname}"/>
  </zeroormore>
</xsl:template>


<xsl:template match = "dd:dbdump/dd:table" mode = "spec">
  <xsl:variable name = "tnam" select = "dd:tabname"/>
  <define name = "el_tbl_{$tnam}">
    <d:para>
      <xsl:text>Tabelle </xsl:text>
      <xsl:value-of select = "$tnam"/>
    </d:para>
    <element name = "{$tnam}">
      <ref name = "foreign_att"/>
      <interleave>
        <ref name = "foreign_el"/>
        <xsl:apply-templates select = "dd:field" mode = "ref">
          <xsl:with-param name = "tnam" select = "$tnam"/>
        </xsl:apply-templates>
      </interleave>
    </element>
  </define>
  <xsl:apply-templates select = "dd:field" mode = "spec">
    <xsl:with-param name = "tnam" select = "$tnam"/>
  </xsl:apply-templates>
</xsl:template>


<xsl:template match = "dd:table/dd:field" mode = "ref">
  <xsl:param name = "tnam"/>
  <xsl:choose>
    <xsl:when test = "dd:null">
      <optional>
        <ref name = "el_fld_{$tnam}_{dd:name}"/>
      </optional>
    </xsl:when>
    <xsl:otherwise>
      <ref name = "el_fld_{$tnam}_{dd:name}"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

```
<xsl:template match = "dd:table/dd:field" mode = "spec">
  <xsl:param name = "tnam"/>
  <xsl:variable name = "tp" select = "dd:type"/>
  <define name = "el_fld_{$tnam}_{dd:name}">
    <d:para>
      <xsl:value-of select = "concat ('Feld ', $tnam, '.', dd:name)"/>
      <xsl:apply-templates select = "dd:type"/>
    </d:para>
    <element name = "{dd:name}">
      <ref name = "foreign_att"/>
      <xsl:choose>
        <xsl:when test = "starts-with ($tp, 'bool')">
          <data type = "boolean"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'tinyint')">
          <data type = "integer"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'smallint')">
          <data type = "integer"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'mediumint')">
          <data type = "integer"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'int')">
          <data type = "integer"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'bigint')">
          <data type = "integer"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'dec')
                          or starts-with ($tp, 'numeric')
                          or starts-with ($tp, 'fixed')
                          or starts-with ($tp, 'number')
                          ">
          <data type = "decimal"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'float')">
          <data type = "number"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'double')">
          <data type = "number"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'bit')">
          <data type = "bitfield"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'binary')">
          <data type = "binary"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'datetime')">
          <data type = "datetime"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'date')">
          <data type = "date"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'timestamp')">
          <data type = "datetime"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'time')">
          <data type = "time"/>
        </xsl:when>
        <xsl:when test = "starts-with ($tp, 'year')">
          <data type = "year"/>
        </xsl:when>
        <xsl:otherwise>
          <text/>
        </xsl:otherwise>
      </xsl:choose>
    </element>
  </define>
</xsl:template>


<xsl:template match = "dd:field/dd:type">
  <xsl:text>: Datentyp </xsl:text>
  <xsl:value-of select = "."/>
</xsl:template>


</xsl:stylesheet>
```

dump_struct_ht.xslt

[Quelltext]

Allgemeines

Datenbank-Struktur als HTML

Das Skript `dbdump` benutzt diese Datei nicht.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	**Namensraum**
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://www.w3.org/1999/xhtml
*	d	http://herbaer.de/xmlns/20051201/doc
*	dd	http://herbaer.de/xmlns/20201201/dbdump
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8
Indent	yes

Eingebundene Stylesheets

/pool/path.xslt - Hilfsvorlagen zu Dateipfaden

Pfade relativ zur Stylesheet-Datei

Im allgemeinen sind relative Dateipfade absoluten Dateipfaden vorzuziehen. Wenn man eine XML-Datei im Dateisystem verschiebt, die eine xml-stylesheet-Anweisung mit einem relativen Verweis-Pfad enthält, dann ist die xml-stylesheet-Anweisung anzupassen. XSLT-Anweisungen zur HTML-Ansicht erzeugen oft Verweise auf CSS- und Javascript-Dateien. Die Pfade zu den Verweiszielen müssen ebenfalls angepasst werden. Die Lösung: die XSLT-Datei enthält Dateipfade, die relativ zu der XSLT-Datei sind, und kombiniert diese Dateipfade mit dem Dateipfad aus der xml-stylesheet-Anweisung zum Ziel-Dateipfad. Diese Datei enthält die Vorlagen zu dieser Lösung.

Zum Test dienen die Dateien *`KLEIDER`*`/catalog/src/dbdump/test_path.xml` und *`KLEIDER`*`/catalog/src/dbdump/test_path.xslt`.

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Wurzel

Aufgerufene benannte Vorlagen:

path.stylepath

Muster-Vorlage dd:dbdump

Datenbank-Struktur

Muster-Vorlage dd:table

Struktur einer Tabelle

Muster-Vorlage dd:field

Parameter

tnam

Tabellenname

Ein Datenbank-Feld

Quelltext

[Beschreibung]

```
<?xml version = "1.0" encoding = "utf-8"?>
<?xml-stylesheet href="xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:dd  = "http://herbaer.de/xmlns/20201201/dbdump"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns     = "http://www.w3.org/1999/xhtml"
  version   = "1.0"
  exclude-result-prefixes = "d dd"
>
<!--
  Datenbank-Struktur als HTML
  2020-12-12 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Deutschland
  GPL Version 2 oder neuer
  Jede Gewährleistung ist ausgeschlossen.
-->
<xsl:output
  method   = "xml"
  encoding = "utf-8"
  indent   = "yes"
/>


<xsl:include href = "/pool/path.xslt"/>


<xsl:template match = "/">
  <xsl:variable name = "title">
    <xsl:text>Datenbank </xsl:text>
    <xsl:value-of select = "dd:dbdump/dd:dbname"/>
  </xsl:variable>
  <html>
    <head>
      <title><xsl:value-of select = "$title"/></title>
      <link rel = "stylesheet">
        <xsl:attribute name = "href">
          <xsl:call-template name = "path.stylepath">
            <xsl:with-param name = "pth">../../../pool/folding.css</xsl:with-param>
          </xsl:call-template>
        </xsl:attribute>
      </link>
      <style>
thead       {font-weight: bold; }
thead tr td {cursor: pointer;   }
td + td     {padding-left: 1em; }
td.number   {text-align: right; }
      </style>
      <script>
        <xsl:attribute name = "src">
          <xsl:call-template name = "path.stylepath">
            <xsl:with-param name = "pth">../../../pool/folding.js</xsl:with-param>
          </xsl:call-template>
        </xsl:attribute>
      </script>
      <script>
onload = function() {
  var f = new Folder();
  f.apply_to_doc_parent_child (document, "div",   "h2");
  f.apply_to_doc_parent_child (document, "tbody", "tr");
};
      </script>
      <style>
tr:first-child          {
  background-color: #dddddd ;
  cursor: n-resize ;
}
tbody.collapsed tr:first-child { cursor: s-resize ; }
tbody.collapsed tr + tr        { display: none; }
      </style>
    </head>
    <body>
      <h1><xsl:value-of select = "$title"/></h1>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>


<xsl:template match = "dd:dbdump">
  <xsl:if test = "dd:table">
    <div>
      <h2>Struktur</h2>
      <table>
        <xsl:apply-templates select = "dd:table"/>
      </table>
    </div>
  </xsl:if>
</xsl:template>
```

```
<xsl:template match = "dd:table">
  <xsl:variable name = "name" select = "dd:tabname"/>
  <tbody id = "tab_{$name}">
    <xsl:apply-templates select = "dd:field">
      <xsl:with-param name = "tnam" select = "$name"/>
    </xsl:apply-templates>
  </tbody>
</xsl:template>


<xsl:template match = "dd:field">
  <xsl:param name = "tnam"/>
  <tr id = "fld_{$tnam}.{dd:name}">
    <td>
      <xsl:if test = "position() = 1">
        <xsl:value-of select = "$tnam"/>
      </xsl:if>
    </td>
    <td>
      <xsl:value-of select = "dd:name"/>
    </td>
    <td>
      <xsl:value-of select = "dd:type"/>
    </td>
    <td>
      <xsl:if test = "not(dd:null)">not null</xsl:if>
    </td>
    <td>
      <xsl:choose>
        <xsl:when test = "dd:key = 'p'">primary</xsl:when>
        <xsl:when test = "dd:key = 'u'">unique</xsl:when>
        <xsl:when test = "dd:key = 'm'">index</xsl:when>
        <xsl:when test = "string-length(dd:default) &gt; 0">
          <xsl:text>Default: </xsl:text>
          <xsl:value-of select = "dd:default"/>
        </xsl:when>
        <xsl:otherwise/>
      </xsl:choose>
    </td>
  </tr>
</xsl:template>


</xsl:stylesheet>
```

path.xslt

[Quelltext]

Allgemeines

Hilfsvorlagen zu Dateipfaden

Im allgemeinen sind relative Dateipfade absoluten Dateipfaden vorzuziehen. Wenn man eine XML-Datei im Dateisystem verschiebt, die eine xml-stylesheet-Anweisung mit einem relativen Verweis-Pfad enthält, dann ist die xml-stylesheet-Anweisung anzupassen. XSLT-Anweisungen zur HTML-Ansicht erzeugen oft Verweise auf CSS- und Javascript-Dateien. Die Pfade zu den Verweiszielen müssen ebenfalls angepasst werden. Die Lösung: die XSLT-Datei enthält Dateipfade, die relativ zu der XSLT-Datei sind, und kombiniert diese Dateipfade mit dem Dateipfad aus der xml-stylesheet-Anweisung zum Ziel-Dateipfad. Diese Datei enthält die Vorlagen zu dieser Lösung.

Zum Test dienen die Dateien *KLEIDER*`/catalog/src/dbdump/test_path.xml` und *KLEIDER*`/catalog/src/dbdump/test_path.xslt`.

Namensräume

Präfix	Namensraum
xml	http://www.w3.org/XML/1998/namespace
xsl	http://www.w3.org/1999/XSL/Transform
d	http://herbaer.de/xmlns/20051201/doc

Globale Variable

Variable path.dir_sspi

Verzeichnis-Teil des Dateipfades der xml-stylesheet - Anweisung

Aufgerufene benannte Vorlagen:

path.dir

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Benannte Vorlage path.cat
Benannte Vorlage path.stylepath

Benannte Vorlagen

Benannte Vorlage path.dir

Parameter

pth

Default: .

der Pfad

Der Verzeichnis-Teil eines Dateipfades bis zum letzten '/'

Die Vorlage wird aufgerufen in:

Benannte Vorlage path.dir
Variable path.dir_sspi
Benannte Vorlage path.cat

Aufgerufene benannte Vorlagen:

path.dir

Benannte Vorlage path.file

Parameter

pth

Default: .

der Pfad

Der Dateiname-Teil eines Dateipfades nach dem letzten '/'

Die Vorlage wird aufgerufen in:

Benannte Vorlage path.file
Benannte Vorlage path.cat

Aufgerufene benannte Vorlagen:

path.file

Benannte Vorlage path.cat

Parameter

prf

Default: $path.dir_sspi

der erste Pfad

sfx

Default: .

der Unter-Pfad

Verkettet zwei Pfade *prf* (Wert "*PRF*") und *sfx* (Wert "*SFX*").

Wenn *prf* leer oder "." ist oder *sfx* mit "/" beginnt, ist das Ergebnis "*SFX*".

Wenn *sfx* leer ist, ist das Ergebnis "*PRF*".

Wenn *prf* "/" oder "/." ist, ist das Ergebnis "*/SFX*".

Wenn *prf* den Wert "*./PRFREST*" hat, wird diese Vorlage mit den Parametern "*PRFREST*" und "*SFX*" aufgerufen.

Wenn *sfx* den Wert "*./SFXREST*" hat, wird diese Vorlage mit den Parametern "*PRF*" und "*SFXREST*" aufgerufen.

Wenn *prf* den Wert "*PRFSTART/*" oder "*PRFSTART/.*" hat, wird diese Vorlage mit den Parametern "*PRFSTART*" und "*SFX*" aufgerufen.

Wenn *sfx* nicht mit "*../*" anfängt, ist das Ergebnis "*PRF/SFX*".

sfx hat den Wert "*../SF*", *prf* hat den Wert "*PDIRPLOCAL*", wobei "*PLOCAL*" nicht leer ist und nicht das Zeichen "*/*" enthält.

Wenn "*PLOCAL*" "*..*" ist, ist das Ergebnis "*PRF/SFX*".

Andernfalls wird diese Vorlage (`path.cat`) mit den Parametern "*PDER*" und "*SF*" aufgerufen.

Die Vorlage wird aufgerufen in:

Benannte Vorlage path.cat
Benannte Vorlage path.stylepath

Aufgerufene benannte Vorlagen:

path.cat
path.file
path.dir

Verwendete globale Parameter oder Variable:

Variable path.dir_sspi

Benannte Vorlage path.stylepath

Parameter

pth

Default: .

der (relative) Pfad

Verkettet den Pfad aus der xml-stylesheet - Anweisung mit einem relativen Pfad.

Aufgerufene benannte Vorlagen:

path.cat

Verwendete globale Parameter oder Variable:

Variable path.dir_sspi

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  version   = "1.0"
>
<!--
  Hilfsvorlagen zu Dateipfaden
  2020-12-12 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  GPL Version 2 oder neuer
-->
<xsl:template name = "path.dir">
  <xsl:param name = "pth" select = "."/>
  <xsl:if test = "contains ($pth, '/')">
    <xsl:value-of select = "concat(substring-before ($pth, '/'), '/')"/>
    <xsl:call-template name = "path.dir">
      <xsl:with-param name = "pth" select = "substring-after ($pth, '/')"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>


<xsl:template name = "path.file">
  <xsl:param name = "pth" select = "."/>
  <xsl:choose>
    <xsl:when test = "contains ($pth, '/')">
      <xsl:call-template name = "path.file">
        <xsl:with-param name = "pth" select = "substring-after ($pth, '/')"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select = "$pth"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:variable name = "path.dir_sspi">
  <xsl:call-template name = "path.dir">
    <xsl:with-param name = "pth" select = "
      substring-before (
      substring-after (
      substring-after (/processing-instruction('xml-stylesheet'), 'href'),
      '&amp;quot;'),
      '&amp;quot;')
    "/>
  </xsl:call-template>
</xsl:variable>


<xsl:template name = "path.cat">
  <xsl:param name = "prf" select = "$path.dir_sspi"/>
  <xsl:param name = "sfx" select = "."/>
  <xsl:choose>
    <xsl:when test = "string-length ($prf) = 0 or $prf = '.' or starts-with ($sfx, '/')">
      <xsl:value-of select = "$sfx"/>
    </xsl:when>
    <xsl:when test = "string-length ($sfx) = 0">
      <xsl:value-of select = "$prf"/>
    </xsl:when>
    <xsl:when test = "$prf = '/' or $prf = '/.'">
      <xsl:value-of select = "concat ('/', $sfx)"/>
    </xsl:when>
    <xsl:when test = "starts-with ($prf, './')">
      <xsl:call-template name = "path.cat">
        <xsl:with-param name = "prf" select = "substring ($prf, 3)"/>
        <xsl:with-param name = "sfx" select = "$sfx"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:when test = "starts-with ($sfx, './')">
      <xsl:call-template name = "path.cat">
        <xsl:with-param name = "prf" select = "$prf"/>
        <xsl:with-param name = "sfx" select = "substring ($sfx, 3)"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:when test = "substring ($prf, string-length ($prf)) = '/'">
      <xsl:call-template name = "path.cat">
        <xsl:with-param name = "prf"
                        select = "substring ($prf, 1, string-length ($prf) - 1)"/>
        <xsl:with-param name = "sfx" select = "$sfx"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:when test =
      "string-length($prf) &gt; 1 and substring ($prf, string-length($prf) - 1) = '/.'"
      >
      <xsl:call-template name = "path.cat">
        <xsl:with-param name = "prf"
```

```
                      select = "substring ($prf, 1, string-length ($prf) - 2)"/>
        <xsl:with-param name = "sfx" select = "$sfx"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:when test = "not (starts-with ($sfx, '../'))">
      <xsl:value-of select = "concat ($prf, '/', $sfx)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:variable name = "f">
        <xsl:call-template name = "path.file">
          <xsl:with-param name = "pth" select = "$prf"/>
        </xsl:call-template>
      </xsl:variable>
      <xsl:choose>
        <xsl:when test = "$f = '..'">
          <xsl:value-of select = "concat ($prf, '/', $sfx)"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:call-template name = "path.cat">
            <xsl:with-param name = "prf">
              <xsl:call-template name = "path.dir">
                <xsl:with-param name = "pth" select = "$prf"/>
              </xsl:call-template>
            </xsl:with-param>
            <xsl:with-param name = "sfx" select = "substring ($sfx, 4)"/>
          </xsl:call-template>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template name = "path.stylepath">
  <xsl:param name = "pth" select = "."/>
  <xsl:call-template name = "path.cat">
    <xsl:with-param name = "prf" select = "$path.dir_sspi"/>
    <xsl:with-param name = "sfx" select = "$pth"/>
  </xsl:call-template>
</xsl:template>


</xsl:stylesheet>
```

Datei test_path.xml

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="test_path.xslt" type="application/xml"?>
<!--
  2020-12-12 Herbert Schiemann <h.schiemann@herbaer.de>

  Diese Datei dient in Verbindung mit der Datei
  KLEIDER/catalog/src/dbdump/test_path.xslt
  zum Test der XSLT-Datei KLEIDER/pool/path.xslt.
  Kopiere diese Datei in irgendein Verzeichnis
  und passe die xml-stylesheet - Anweisung in der zweiten Zeile an,
  dass sie auf KLEIDER/catalog/src/dbdump/test_path.xslt verweist
  (absoluter oder relativer Pfad)
-->
<test xmlns = "http://herbaer.de/xmlns/20201212/testpath">
  <!--
    die folgenden Pfade werden in einen Verzeichnis-Teil einschließlich
    des letzten '/' und einen Datei-Teil (ohne '/') zerlegt.
  -->
  <path>../auto/bahn/bau/stelle</path>

  <!--
    Verzeichnis-Komponente des Dateipfades aus der xml-stylesheet-Anweisung
  -->
  <dir_sspi/>

  <!--
    Verkettet zwei Pfade prf und sfx
  -->
  <cat>
    <prf></prf>
    <sfx>./../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>.</prf>
    <sfx>./../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>./irgend/ein/pfad/./..</prf>
    <sfx>/ein/absoluter/pfad</sfx>
  </cat>
  <cat>
    <prf>./der/unterpfad/ist/leer/...</prf>
    <sfx></sfx>
  </cat>
  <cat>
    <prf>/</prf>
    <sfx>./../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>/.</prf>
    <sfx>./../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>././../erster/pfad/mit/selbstkuerzel</prf>
    <sfx>./.././.../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>unterpfad/mit/selbstkuerzel</prf>
    <sfx>./relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>unterpfad/mit/selbstkuerzel</prf>
    <sfx>./../../.../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>./hauptpfad/mit/gedoens/.///././/</prf>
    <sfx>./../../.../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>basis/pfad/./</prf>
    <sfx>../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>basis/pfad/mit/clash/./..</prf>
    <sfx>../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>basis/pfad/normal/./../...</prf>
    <sfx>../relativer/pfad</sfx>
  </cat>
  <cat>
    <prf>basis/pfad/mit/clash/und_gedoens/./.././/.</prf>
    <sfx>../relativer/pfad</sfx>
  </cat>

  <!--
    Pfad verkettet mit der Verzeichnis-Komponente aus der xml-stylesheet-Anweisung
  -->
  <stylepath>../hier/ist/das/ziel</stylepath>
</test>
```

test_path.xslt

[Quelltext]

Namensräume

Präfix	Namensraum
xml	http://www.w3.org/XML/1998/namespace
t	http://herbaer.de/xmlns/20201212/testpath
d	http://herbaer.de/xmlns/20051201/doc
xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	text
Encoding	utf-8

Eingebundene Stylesheets

/pool/path.xslt - Hilfsvorlagen zu Dateipfaden

Im allgemeinen sind relative Dateipfade absoluten Dateipfaden vorzuziehen. Wenn man eine XML-Datei im Dateisystem verschiebt, die eine xml-stylesheet-Anweisung mit einem relativen Verweis-Pfad enthält, dann ist die xml-stylesheet-Anweisung anzupassen. XSLT-Anweisungen zur HTML-Ansicht erzeugen oft Verweise auf CSS- und Javascript-Dateien. Die Pfade zu den Verweiszielen müssen ebenfalls angepasst werden. Die Lösung: die XSLT-Datei enthält Dateipfade, die relativ zu der XSLT-Datei sind, und kombiniert diese Dateipfade mit dem Dateipfad aus der xml-stylesheet-Anweisung zum Ziel-Dateipfad. Diese Datei enthält die Vorlagen zu dieser Lösung.

Zum Test dienen die Dateien *KLEIDER*`/catalog/src/dbdump/test_path.xml` und *KLEIDER*`/catalog/src/dbdump/test_path.xslt`.

Muster-Vorlagen (matching templates)

Muster-Vorlage t:test

Wurzelelement

Muster-Vorlage t:path

Pfad zerlegen in Verzeichnis-Komponente und Dateinamen

Aufgerufene benannte Vorlagen:

newline
path.dir
path.file

Muster-Vorlage t:dir_sspi

Verzeichnis aus der xml-stylesheet-Anweisung

Aufgerufene benannte Vorlagen:

newline

Muster-Vorlage t:cat

Verkettung zweier Pfade

Aufgerufene benannte Vorlagen:

newline
path.cat

Muster-Vorlage t:stylepath

Stylesheet-relativer Pfad

Aufgerufene benannte Vorlagen:

newline
path.cat
path.stylepath

Benannte Vorlagen

Benannte Vorlage newline

Zeilenende

Die Vorlage wird aufgerufen in:

Muster-Vorlage t:path
Muster-Vorlage t:dir_sspi
Muster-Vorlage t:cat
Muster-Vorlage t:stylepath

Quelltext

[Beschreibung]

```
<?xml version = "1.0" encoding = "utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:t   = "http://herbaer.de/xmlns/20201212/testpath"
  version   = "1.0"
>
<!--
  2020-12-12 Herbert Schiemann <h.schiemann@herbaer.de>
  Diese Datei dient in Verbindung mit der Datei
  KLEIDER/catalog/src/dbdump/test_path.xml
  zum Test der XSLT-Datei KLEIDER/pool/path.xslt.
-->

<xsl:output method = "text" encoding = "utf-8"/>


<xsl:include href = "/pool/path.xslt"/>


<xsl:template name = "newline">
  <xsl:text>
</xsl:text>
</xsl:template>


<xsl:template match = "t:test">
  <xsl:apply-templates select = "*"/>
</xsl:template>


<xsl:template match = "t:path">
  <xsl:text>Pfad        </xsl:text>
  <xsl:value-of select = "."/>
  <xsl:call-template name = "newline"/>
  <xsl:text>Verzeichnis </xsl:text>
  <xsl:call-template name = "path.dir"/>
  <xsl:call-template name = "newline"/>
  <xsl:text>Datei       </xsl:text>
  <xsl:call-template name = "path.file"/>
  <xsl:call-template name = "newline"/>
  <xsl:call-template name = "newline"/>
</xsl:template>


<xsl:template match = "t:dir_sspi">
  <xsl:text>Verzeichnis aus der xml-stylesheet-Anweisung</xsl:text>
  <xsl:call-template name = "newline"/>
  <xsl:value-of select = "$path.dir_sspi"/>
  <xsl:call-template name = "newline"/>
  <xsl:call-template name = "newline"/>
</xsl:template>


<xsl:template match = "t:cat">
  <xsl:text>1. Pfad    </xsl:text>
  <xsl:value-of select = "t:prf"/>
  <xsl:call-template name = "newline"/>
  <xsl:text>2. Pfad    </xsl:text>
  <xsl:value-of select = "t:sfx"/>
  <xsl:call-template name = "newline"/>
  <xsl:text>Verkettung </xsl:text>
  <xsl:call-template name = "path.cat">
    <xsl:with-param name = "prf" select = "t:prf"/>
    <xsl:with-param name = "sfx" select = "t:sfx"/>
  </xsl:call-template>
  <xsl:call-template name = "newline"/>
  <xsl:call-template name = "newline"/>
</xsl:template>


<xsl:template match = "t:stylepath">
  <xsl:text>Stylesheet-relativ          </xsl:text>
  <xsl:value-of select = "."/>
  <xsl:call-template name = "newline"/>
  <xsl:text>Dokument-relativ (default)  </xsl:text>
  <xsl:call-template name = "path.cat"/>
  <xsl:call-template name = "newline"/>
  <xsl:text>Dokument-relativ (explizit) </xsl:text>
  <xsl:call-template name = "path.stylepath"/>
  <xsl:call-template name = "newline"/>
  <xsl:call-template name = "newline"/>
</xsl:template>


</xsl:stylesheet>
```