Gentable

Das Gentable kann über ein "Gentable"-Objekt realtiv einfach ausgelesen und neu geschrieben werden.

```
var docFile = context.file;
var gentable = new Gentable(docFile);

/* Erstellt ein Array aus Zeilen-Objekten *
 * Zahlen, Datumswerte und Boolsche Werte haben bereits das passende Format */
if( gentable.readFromField()===false || gentable.Result===false ){
  util.log(Gentable.Log); /* Bei Fehlern kann das Log ausgegeben werden */
  context.errorMessage = gentable.Error; /* Fehlermeldung in Client-Sprache */
  return -1;
}

/* Beispiel um Zeilen zu iterieren */
for( var row=0; row<gentable.Rows.length; row++ ){
  var net = gentable.Rows[row]["Net"]; /* Netto aus der Zeile ausgeben */
  gentable.Rows[row]["Gross"] = net; /* Brutto auf den Netto-Wert setzen */
}

/* Summe aus Spaltenwerten bilden. */
var sumNet = gentable.sumLineAmounts("Net");

/* Gentable in die Datenbank-Tabelle 'Invoice_Posting_Pos' schreiben.
 * Die eindeutige Zeilen ID wird zurück in die Gentable-Spalte "ID" geschrieben.
 * Hierüber wird beim nächsten Aufruf ein Update ausgeführt. */
gentable.insertIntoDB();

/* Gentable-String in das Feld 'Gentable' schreiben.
 * Über den Parameter kann optional direkt ein sync() auf das DocFile-Objekt ausgeführt
werden. */
var syncFile = true;
gentable.createGentableStr(syncFile);
```

Gentable User-Exits

An einigen Skriptstellen wurden User-Exit - Funktionen direkt auf das Gentable-Objekt hinzugefügt. Dadurch muss das Gentable nicht nochmals umständlich ausgelesen werden. Über "this" kann direkt auf das Gentable-Objekt zugegriffen werden.

Die Funktionen selber müssen keinen Rückgabewert zurückgeben. Wenn "this.Result" auf den boolschen Wert "false" gesetzt wird kann über "this.Error" eine Fehlermeldung mitgegeben werden.

```
/** Wird bei der Erstellung der initialen Gentable-Zeile aufgerufen **/
Gentable.prototype.createInitialRow = function (){
}
```

```
/** Wird am Ende des Skripts beim Laden der offenen Bestellpositionen ausgeführt **/
Gentable.prototype.adjustLoadedOrderPositions = function (){
}
```

```
/** User-Exit beim Speichern einer Rechnungs-Akte
 * @since Invoice 1.0.200
 **/
Gentable.prototype.ue_InvoiceOnSave = function (){
	this.Result = false;
	this.Error = "Fehlermeldung";
	var docFile = this.DocFile;
	for( var r=0; r<this.Rows.length; r++){
		var row = this.Rows[r];
		this.Rows[r]["Net"] = 0;
	}
}
```

```
/** User-Exit bei Abschluss einer Workflow-Aktion
 * @since Invoice 1.0.200
 **/
Gentable.prototype.ue_OnActionEnd = function (){
	this.Result = false;
	this.Error = "Fehlermeldung";
	var docFile = this.DocFile;
	for( var r=0; r<this.Rows.length; r++){
		var row = this.Rows[r];
```

```
□□this.Rows[r]["Net"] = 0;
□}
}
```

```
/** Wird nach dem Auslesen des Gentable aber noch vor dem Anwenden einer Standardkontierung
ausgeführt.
 * @param {Accounting} standardAccounting Das Standardkontierungs-Objekt
 * @since Invoice 1.1.300
**/
Gentable.prototype.ue_BeforeStandardAccounting = function(standardAccounting){
  /* Beispiel: Kürzt das Gentable auf die gleiche Anzahl an Positionen wie die
Standardkontierung *
    accObj.getAccountingPositions(accObj.AccountingName);
    var accPos = accObj.Accounting[accObj.AccountingName];
    if( this.Rows.length>accPos.length ){
        for( r=this.Rows.length-1; r>=accPos.length ;r-- ){
            if( this.Logg instanceof Log ){
                this.Logg.info("["+r+"] this.Rows.pop();");
            }
            this.Rows.pop();
        }
    }
    */
}

/** Wird nach dem Anwenden einer Standardkontierung ausgeführt.
 * Im Anschluss wird automatisch gentable.createGentableStr(true) ausgeführt.
 * @param {Accounting} standardAccounting Das Standardkontierungs-Objekt wird ab Invoice
1.1.300 mitgegeben
 * @since Invoice 1.0.300
**/
Gentable.prototype.ue_AfterStandardAccounting = function(standardAccounting){
  ;
}
```

Gentable XML User-Exits

Die Gentable-XML wird automatisch über die WEB-Konfigurations-Tabellen generiert. Um nur mehr Flexibilität zu ermöglichen können die intern generierten Objekte nochmal angepasst werden.Die folgenden Funktionen befinden sich im Skript "**DEXPRO__UserExit_GentableXml**".

Globale XML-Einstellungen können zum Beispiel nur auf einen Wert gesetzt werden. Über die folgende Funktion können Eigenschaften abhängig von Feldwerten an der Mappe manipuliert werden.

```
/** User exit to manipulate gentable Xml settings such as "indexCheckboxes",
"lastRowEditable", "storeFormat", ...
 * Called within "GentableXml.createGentableXml()".
 * @param {any} settingObject Object with Gentable settings
 * @param {string} template Name of the DocFile-template ("Mailroom", "Invoice",
"lcmContract", ...)
 * @param {DocFile} docFile DocFile object
 * @returns {array} (Manipulated) Array with Gentable settings objects.
 **/
function ue_GentableXml_ManipulateGentableXmlSettings(settingObject, template, docFile){
    /* If value is null the standard value will be used:
        xmlSetting.fieldName              // Field name        | must value!!!     ->
technical docFile head field name to store Gentable data
        xmlSetting.database               // Database name     | no standard value -> database
name for database connections
        xmlSetting.alwaysShowToolbar      // "true" or "false" | no standard value -> shows
gentable-buttons even in read-mode
        xmlSetting.url                    // Database url      | no standard value -> database
url (will only be set if datase is not null)
        xmlSetting.driver                 // Database driver   | no standard value -> database
driver (will only be set if datase is not null)
        xmlSetting.user                   // Database user     | no standard value -> database
user (will only be set if datase is not null)
        xmlSetting.password               // Database pw       | no standard value -> database
user password (will only be set if datase is not null)
        xmlSetting.storeFormat            // "xml" or "json"   | Standard: "xml"   -> gentable
store format
        xmlSetting.indexCheckboxes        // "true" or "false" | Standard: "true"  -> sets
checkboxes in front of each row to select rows
        xmlSetting.indexNumbers           // "true" or "false" | Standard: "true"  -> shows
line number in front of each row
        xmlSetting.enableTextSelection    // "true" or "false" | Standard: "false" -> enables
text selection from more than one field (even in read-mode)
```

```
        xmlSetting.languagePropertiesFile // technical pf-name | no standard value ->
Properties file name to use pf
                                          //                                          example:
for "gentable_de.properties" set "gentable"
        xmlSetting.lastRowEditable        // "true" or "false" | Standard: "true"  -> set
"true" if you want to edit each column when you add new rows
        xmlSetting.quickFilter            // "true" or "false" | Standard: null    -> adds
search field for each column
        xmlSetting.saveAll                // "true" or "false" | Standard: null    -> saves
all values even if they are not listed in xml definition
        xmlSetting.searchable             // "true" or "false" | Standard: "false" -> shows
search field in head-button-line
        xmlSetting.stickyGroupRows        // "true" or "false" | Standard: null    -> sticks
head line while scrolling down
        xmlSetting.columnsAlwaysVisible   // "true" or "false" | Standard: "true"  -> use this
setting to hide column if each value is invisible
        xmlSetting.resizableRows          // "true" or "false" | Standard: "false" ->
resizable row width
        xmlSetting.resizeColumnsToContent // "true" or "false" | Standard: "false" ->
automatically resizes field size to it's content
                                          //                                          does not
work correct if the content is larger than the technical value (1000 -> 1.000,00)
        xmlSetting.rowHeight              // integer value     | Standard: 20      -> standard
pixel row height
        xmlSetting.moveRows               // "true" or "false" | Standard: "false" -> allows
changing row positions via drag and drop
        xmlSetting.showMoveColumn         // "true" or "false" | Standard: "false" -> shows
extra column to move rows
    */

    return settingObject;
}
```

In der XML können auch Zeilen-Bedingungen definiert werden. Hierüber kann ein Schreibschutz auf ganze Zeilen gelegt werden bzw. können Zeilen abhängig von Feldwerten ausgeblendet werden.

```
/** User exit to build up Gentable Xml row conditions.
 * First type "READONLY" disables  field editing for all row-fields.
 * Second type "INVISIBLE" hides complete row if the rule matches.
 * Called within "GentableXml.createGentableXml()".
```

```
 * @param {String} template Name of the DocFile-template ("Mailroom", "Invoice",
"lcmContract", ...)
 * @param {DocFile} docFile DocFile object
 * @returns {String} Gentable row condition xml tag.
**/
function ue_GentableXml_AddRowConditions(template, docFile){
    var rowCondition = '';
    /* EXAMPLES from Otris Gentable Administration guide: *
    rowCondition = '<rowCondition><rule type="READONLY" field="Field1" value="2222"
/></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" filefield="Creditor" value="ALFKI"
/></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" accessprofile="Warehouse"
/></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" field="Field5" value="Field5"
accessprofile="Directors" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" filefield="Creditor" value="OTRIS"
accessprofile="Administration" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY"
autotext="%currentUser.$AllowedAmount%" field="Field4" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY"
autotext="%currentUser.$AllowedAmount%" filefield="Amount" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="INVISIBLE" invert="true"
accessprofile="Administration" /></rowCondition>';
    */
    return rowCondition;
}
```

Über die Feldkonfigurationen in der WEB-Administration können die Feldeigenschaften bereits abhängig von Kopfwerten geändert werden. Die folgende Funktion wurde zum Beispiel standardmäßig erweitert, um bei einem Klick auf irgendein Feld innnerhalb einer Zeile, diese Zeile als 'aktiv' zu kennzeichnen. Hierüber wird im Squeeze-Viewer zu einer Rechnung die passende Rechnungsposition angezeigt. Über die Funktion können sogar Feldtypen geändert werden.

Einen sehr interessanter Anwendungsbereich bietet die Funktion bei der Auflistung von Auswahl-Listen. Diese können über die WEB-Konfiguration nur starr angegeben werden. Über das Skripting können zum Beispiel sehr einfach Benutzer und Benutzergruppen mit technischem Wert und Anzeige-Wert angegeben werden.

```
/** User exit to manipulate field configuration. This user exit is called for each Gentable
field.
 * Called within "GentableXml.createGentableXml()".
```

```
 * @param {FieldConfig} fieldConfigObject Field configuration object (Standard+Changes)
 * @param {String} template Name of the DocFile-template ("Mailroom", "Invoice",
"lcmContract", ...)
 * @param {DocFile} docFile DocFile object
 * @returns {FieldConfig} Field configuration object.
 **/
function ue_GentableXml_ManipulateFieldSettings(fieldConfigObject, template, docFile){
    var attribute     = "name";
    var techFieldName = fieldConfigObject.TechFieldName;

    /* Beispiel für ein Feld, welches alle Zugriffsprofile listet */
    if( techFieldName==="EnumAccessProfileField" ){
        fieldConfigObject.FieldDataType = "SELECT";
        fieldConfigObject.SelectOption  = "<option value=\"\"></option>"; // Leere Feldwert-
Option
        var iterAP = context.getAccessProfiles(false);
        for( var ap=iterAP.first(); ap; ap=iterAP.next() ){
            var apDisplayName = (ap.getAttribute(attribute)===""")? ap.name :
ap.getAttribute(attribute);
            fieldConfigObject.SelectOption += "<option
value=\""+ap.name+"\">"+apDisplayName+"</option>";
        }
    }

    /* Setzt die angeklickte Zeile als 'aktiv', um im Squeeze-Viewer die entsprechende Zeile
zu markieren */
    if( fieldConfigObject.JsEventType1!=="onFocus" &&
fieldConfigObject.JsEventType2!=="onFocus"  && fieldConfigObject.JsEventType3!=="onFocus" ){
        fieldConfigObject.JsEventType4 = "onFocus";
        fieldConfigObject.JsEvent4     = "setActiveRow";
    }
    return fieldConfigObject;
}
```

Revision #17
Created 2020-12-07 13:15:35 UTC by Markus Meisner
Updated 2023-08-22 10:23:43 UTC by Markus Meisner