DOCUMENTS

GENTABLE (INVOICE PLUGIN)
Administration und Konfiguration

VERSION 5.0

Inhaltsverzeichnis

1. Einleitung 4

1.1 Überblick 4
1.2 Die wichtigsten Eigenschaften & Features 4
1.3 Einsatz als Invoice-Plugin 5

2. Konfiguration & Anpassung 6

2.1 Einbindung in DOCUMENTS 5 6
2.2 Definition der Konfigurationsdatei 10
2.2.1 Die Grundeinstellungen 10
2.2.2 Die Definition der einzelnen Tabellenspalten/Felder 10
2.2.3 Einschränkungen des Inhalts 12
2.2.4 Einschränkungen der Sichtbarkeit 12
2.2.5 Abfrage externer Datenbanken 15
2.2.6 Definition von Buttons 16
2.2.7 Mehrere Konfigurationsdateien für einen Mappentypen 17
2.3 Anpassung der benutzerdefinierten Skript-Datei 17
2.4 Internationalisierung 18
2.4.1 Internationalisierung in der Konfigurationsdatei 18

3. Implementierung eigener Funktionalität 19

3.1 Einführung 19
3.2 Registrierung und Callback-Funktionen 19
3.3 Callback-Funktionen für Buttons 19
3.4 GentableContext 20
3.5 GentableGridModel 20
3.6 GentableGridColumnModel 22
3.7 GentableGridRowModel 22

4. Einsatzbeispiele & Standardfunktionen 23

5. Fehler & Ursachen 26

6. Abbildungsverzeichnis 28

1. Einleitung

1.1 Überblick

Die webbasierte Software-Komponente **GenTable** ist bereits in **DOCUMENTS 5** integriert und muss nicht separat installiert werden. Sofern eine gültige Lizenz vorhanden ist, kann **GenTable** direkt verwendet werden.

GenTable eignet sich generell dafür, beliebige tabellarische bzw. positionsbezogene Daten auf benutzerdefinierte Weise darzustellen. Diese Daten können daraufhin durch den Anwender bearbeitet und gespeichert werden.

Weiterhin ist die Integration selbst implementierter Funktionalität möglich. Auch externe Daten aus relationalen Datenbanken lassen sich einbinden.

Wichtige Merkmale sind darüber hinaus die Speicherung der Tabellendaten und der Konfigurationsdaten im XML-Format sowie die clientseitige Ausführung eines Großteils der Anwendung.

1.2 Die wichtigsten Eigenschaften & Features

Die wichtigsten Eigenschaften und Features der **GenTable**-Komponente:

- benutzerdefinierte Darstellung von beliebigen tabellarischen bzw. positionsbezogenen Daten
- Speicherung der Tabellendaten und der Konfigurationsdaten im *XML*-Format
- clientseitige Bearbeitung der Daten
- mögliche Elementtypen der Tabellenausgabe sind einzeilige und mehrzeilige Textfelder, Klapplisten, Kontrollkästchen (Checkboxen), statischer Text und Buttons
- anwendungsspezifischere Hilfs-Funktionen, z.B. Splittbuchung sowie automatisches Berechnen einzelner Positionsbeträge und des Gesamtbetrags bei Rechnungsmappen
- Erweiterbarkeit durch Integration eigener Funktionalität mit dem Client SDK
- benutzerdefinierte Integration von beliebigen externen Daten aus relationalen Datenbanken in ein- und mehrzeilige Textfelder, Klapplisten etc.
- externe Daten können zusätzlich vom eingeloggten Benutzer, von beliebigen Werten der Mappe oder von Werten derselben Zeile abhängig sein
- Tabellenzeilen können abhängig von der Definition sichtbar, unsichtbar oder readonly (nicht änderbar) sein
- automatisch ausgeführte Ereignisse mit bestimmten Tabellenspalten bzw. -feldern verknüpfen

1.3 Einsatz als Invoice-Plugin

Das bisher bekannteste Einsatzbeispiel von **GenTable** ist die Verwendung für die Eingangsrechnungsprüfung in Rechnungsmappen als sogenanntes **Invoice-Plugin**. Die Tabellenspalten entsprechen in diesem Fall den einzelnen Kategorien einer Rechnung (z.B. Bestellnummer, Menge, Einzelpreis, Gesamtpreis, Kostenstelle etc.) und die Tabellenzeilen entsprechen den einzelnen Rechnungspositionen.

2. Konfiguration & Anpassung

2.1 Einbindung in DOCUMENTS 5

Um **GenTable** in **DOCUMENTS 5** zu konfigurieren, muss zunächst der **DOCUMENTS-Manager** gestartet werden. In der Baumstruktur im linken Teil des Hauptfensters werden die einzelnen vorhandenen Mappentypen unter den Punkten *Documents / Alle Mappentypen* aufgelistet.

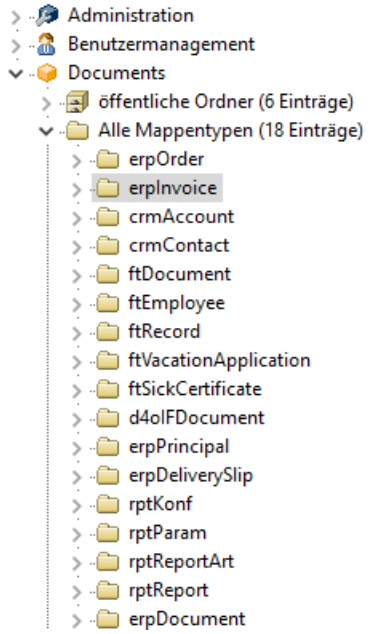


Abb. 1: Baumstruktur im DOCUMENTS-Manager

Nun öffnet man entweder einen der vorhandenen Mappentypen durch Doppelklick auf den jeweiligen Mappentyp-Eintrag in der Baumstruktur, oder man erstellt einen neuen Mappentyp über den Button *Neu* links oben im Hauptmenü. Dazu muss der Eintrag *Alle Mappentypen* in der Baumstruktur zuvor aktiviert worden sein.

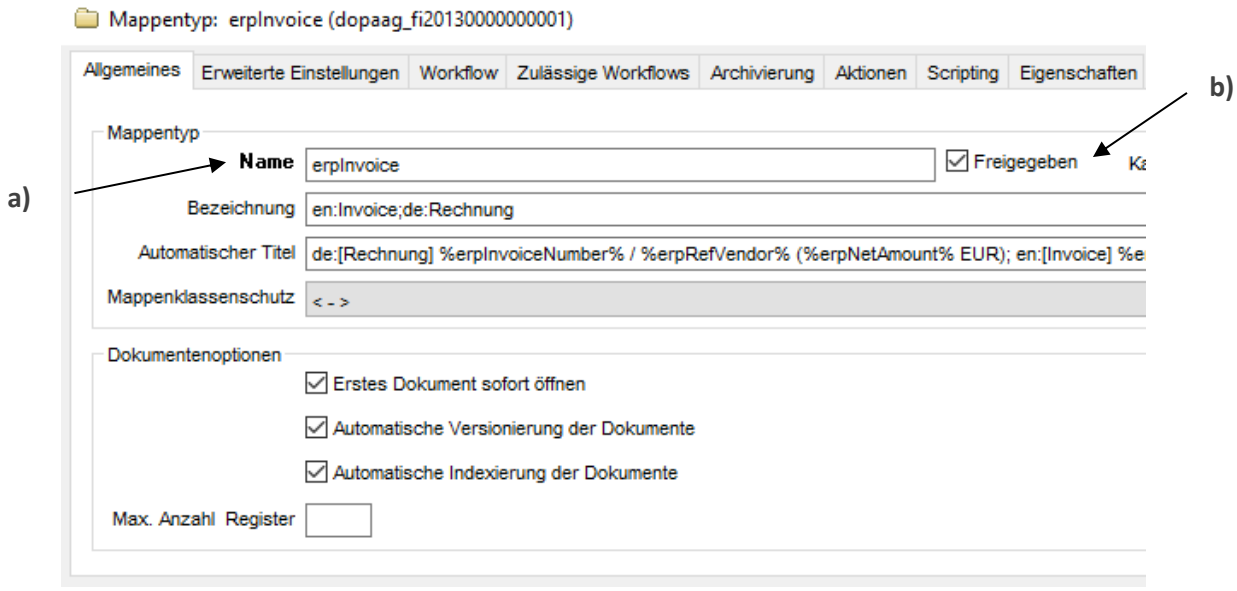


Abb. 2: Bearbeiten bzw. Erstellen eines Mappentyps

Wichtige Einstellungen bei einem Mappentyp sind die beiden folgenden Punkte in der Kategorie *Allgemeines*:

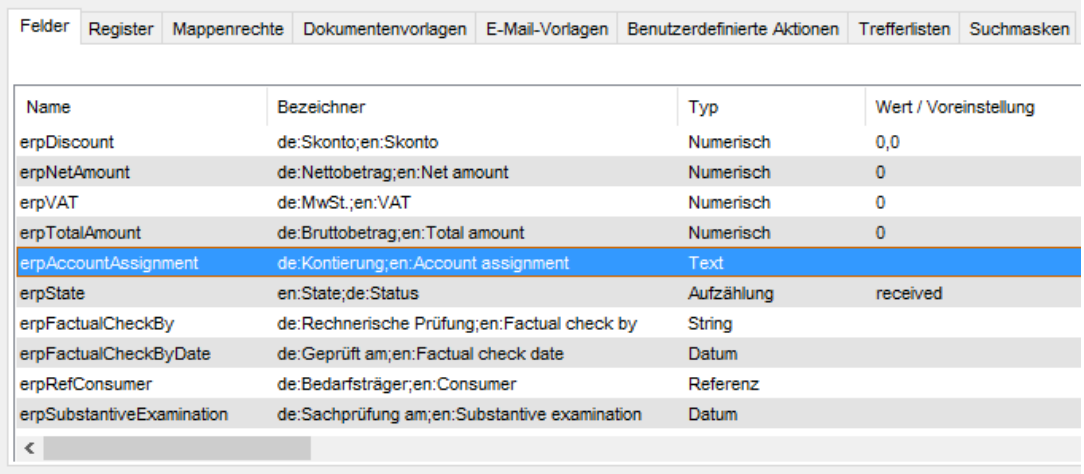
- **a)** Im Textfeld *Name* muss der eindeutige Name des Mappentyps angegeben werden
- **b)** Das Häkchen bei der Checkbox *Freigegeben* muss gesetzt sein, um den Mappentyp in **DOCUMENTS 5** freizugeben

Um **GenTable** in den Mappentyp zu integrieren, muss folgende Einstellung in der Kategorie *Eigenschaften* vorgenommen werden:

Über einen Rechtsklick in diesem Fenster und die dann folgende Auswahl der Funktion *neue Eigenschaft hinzufügen* muss eine neue Eigenschaft mit der Bezeichnung `hasInvoicePlugin` mit dem Wert `true` erstellt werden.

Abb. 3: Eigenschaften eines Mappentyps

Im unteren Teil des Fensters zur Bearbeitung des Mappentyps sind in der Kategorie *Felder* die einzelnen Felder des Mappentyps aufgelistet. Die vorhandenen Felder können hier durch einen Doppelklick auf den jeweiligen Eintrag bearbeitet werden. Durch einen Rechtsklick in diesem Teil des Fensters und Auswahl der Funktion *Neuen Datensatz anlegen und in Felder einfügen* können weitere Felder erstellt werden.



Name	Bezeichner	Typ	Wert / Voreinstellung
erpDiscount	de:Skonto;en:Skonto	Numerisch	0,0
erpNetAmount	de:Nettobetrag;en:Net amount	Numerisch	0
erpVAT	de:MwSt.;en:VAT	Numerisch	0
erpTotalAmount	de:Bruttobetrag;en:Total amount	Numerisch	0
erpAccountAssignment	de:Kontierung;en:Account assignment	Text	
erpState	en:State;de:Status	Aufzählung	received
erpFactualCheckBy	de:Rechnerische Prüfung;en:Factual check by	String	
erpFactualCheckByDate	de:Geprüft am;en:Factual check date	Datum	
erpRefConsumer	de:Bedarfsträger;en:Consumer	Referenz	
erpSubstantiveExamination	de:Sachprüfung am;en:Substantive examination	Datum	

Abb. 4: Bearbeiten bzw. Erstellen von Feldern

Für **GenTable** ist es erforderlich, dass ein zusätzliches, nicht-sichtbares Feld für die Speicherung der XML-Tabellendaten existiert bzw. erstellt wird. In diesem Beispiel ist dies das Feld *erpAccountAssignment*.

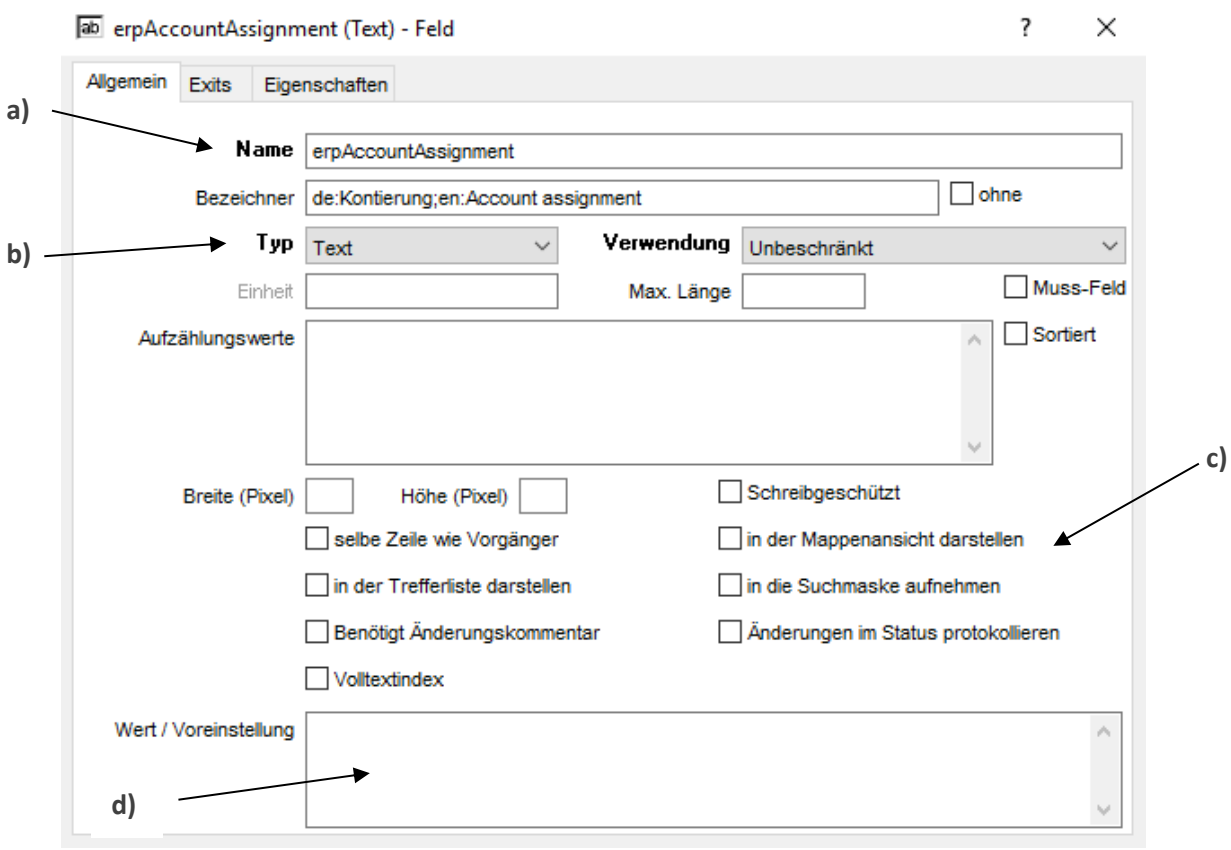


Abb. 5: Mappenfeld-Dialog

Die wichtigsten Punkte bei der Konfiguration des unsichtbaren Feldes für die XML-Tabellendaten sind die folgenden Einstellungen:

a) Im Textfeld *Name* muss ein eindeutiger Name für dieses Feld vergeben werden. Dieser Name muss später auch in der XML-Konfigurationsdatei von **GenTable** unter `<xmlfield>` verwendet werden.

b) Der Typ des Feldes für die XML-Tabellendaten sollte *Text* oder *String* sein.

c) Da das Feld mit den XML-Tabellendaten nicht angezeigt werden darf, wird das Häkchen bei der Checkbox *in der Mappenansicht darstellen* nicht gesetzt.

d) In das Textfenster *Wert/Voreinstellung* können optional vorab definierte XML-Tabellendaten eingetragen werden. Die hier eingetragenen Daten werden bei jedem Neuanlegen einer Mappe dieses Typs in der Tabelle von **GenTable** standardmäßig eingetragen. Durch einfaches Setzen der XML-Tags für Zeilen `<tr></tr>` und Spalten`<td></td>` kann so schon vorher festgelegt werden, wie viele Zeilen beim Neuanlegen automatisch angezeigt werden. Einträge von Spalten mit statischem Text oder von Spalten, die als nicht editierbar definiert sind, können so über die *Voreinstellung* gesetzt werden, da diese nicht mehr vom Benutzer vorgenommen werden können. Die Tabellendaten werden immer in folgender XML-Struktur gespeichert (hier beispielhaft eine Tabelle mit drei Spalten und drei Zeilen):

```
<table>
  <tr>
        <td title="Spalte 1">Beispieleintrag1</td>
        <td title="Spalte 2">Beispieleintrag2</td>
        <td title="Spalte 3">Beispieleintrag3</td>
  </tr>
  <tr>
        <td title="Spalte 1">Beispieleintrag4</td>
        <td title="Spalte 2">Beispieleintrag5</td>
        <td title="Spalte 3">Beispieleintrag6</td>
  </tr>
  <tr>
        <td title="Spalte 1">Beispieleintrag7</td>
        <td title="Spalte 2">Beispieleintrag8</td>
        <td title="Spalte 3">Beispieleintrag9</td>
  </tr>
</table>
```

Die grundlegenden Einstellungen zur Einbindung von **GenTable** in den **DOCUMENTS 5** Mappentypen sind damit abgeschlossen.

2.2 Definition der Konfigurationsdatei

Jede Konfiguration von **GenTable** bezieht sich in der Regel auf einen bestimmten Mappentypen in **DOCUMENTS 5**. Dazu wird im Verzeichnis `.../tomcat8/webapps/documents/web/WEB-INF/classes/` der **DOCUMENTS 5** - Installation eine XML-Konfigurationsdatei mit dem Namen `<FileTypeName>_Def.xml` hinterlegt. `<FileTypeName>` wird hier durch den technischen Namen des Mappentypen (z.B. `erpInvoice_Def.xml`) ersetzt. Als Alternative dazu kann die Konfiguration auch als sogenannte externe Ressource in **DOCUMENTS 5** eingebunden werden. - Im Folgenden werden die einzelnen XML-Elemente der Konfigurationsdatei detailliert erläutert.

2.2.1 Die Grundeinstellungen

`<table_def name="">`: Das Wurzelelement der Konfigurationsdatei. Das Attribut `name` enthält den eindeutigen Namen der Konfiguration. Alle weiteren Elemente werden als Kindelemente von `<table_def>` definiert.

`<xmlfield></xmlfield>`***:*** Der Name des nicht-sichtbaren Mappenfeldes, in welchem die XML-Daten von **GenTable** gespeichert werden sollen.

Mit dem Element `<database></database>` können die benötigten Parameter für die Verbindung zu einer Datenbank via JDBC für diese Tabelle festgelegt werden. JDBC-URL der Datenbank, JDBC-Treiber, Benutzername und Passwort können über die Unterelemente `<url> </url>`, `<driver> </driver>`, `<user> </user>` und `<password> </password>` angegeben werden.

`<indexNumbers></indexNumbers>`***:*** Legt mit `true` oder `false` fest, ob vor jeder Zeile eine fortlaufende Zeilennummer angezeigt werden soll. Der Default-Wert ist `true`.

`<indexCheckboxes></indexCheckboxes>`***:*** Legt mit `true` oder `false` fest, ob vor jeder Zeile eine Checkbox zur Auswahl der Zeilen angezeigt werden soll. Der Default-Wert ist `true`.

`<lastroweditable>`: Standardmäßig ist **bei der Neuanlage** einer Zeile **jedes Feld** editierbar. Das gilt auch für die Felder, die für das Element `<editable>` den Wert `false` haben. Falls jedoch `<lastroweditable>` auf den Wert `false` gesetzt wird, sind diese Felder auch bei der Neuanlage einer Zeile nicht editierbar. Der Default-Wert ist `true`.

`<searchable>`: Zeigt bei *true* in der Buttonzeile ein Suchfeld zum clientseitigen Filtern der angezeigten Tabellenzeilen an. Der Default-Wert ist `false`.

`<alwaysShowToolbar>`: Legt mit `true` oder `false` fest, ob die Buttonzeile auch im Lesemodus der aktuellen Mappe angezeigt werden soll. Der Default-Wert ist `false`.

`<rowHeight>`: Legt die Standardhöhe einer Zeile in Pixel fest (z.B. `50`).

2.2.2 Die Definition der einzelnen Tabellenspalten/Felder

Mit dem Element `<field></field>` wird eine Spalte der Tabelle definiert. Es muss stets **mindestens eine Tabellenspalte** definiert werden.

Die Detaileinstellungen der jeweiligen Spalte werden mit den folgenden Unterelementen festgelegt.

`<title></title>` (Pflichtfeld): Enthält den technischen Namen der Spalte.

`<label></label>`***:*** Enthält eine optionale Spaltenüberschrift, welche in der Benutzeroberfläche angezeigt wird. (Kann internationalisiert werden, siehe Abschnitt 2.4.1)

`<type></type>`***:*** Datentyp des Feldes. Es stehen folgende Datentypen zur Verfügung:

- **TEXT,** einzeiliges Textfeld
- **TEXTAREA,** mehrzeiliges Textfeld für längeren Text
- **SELECT,** Klappbox mit Liste von vordefinierten Einträgen
- **CHECKBOX,** Checkbox
- **STATICTEXT,** statischer, nicht-änderbarer Text
- **BUTTON,** Schaltfläche
- **DATE,** Datumsfeld mit Kalender
- **TIMESTAMP,** Zeitstempel mit Kalender und Uhrzeit

`<default></default>`: Ausgangswert des Feldes bei Erzeugung einer Zeile.

`<editable></editable>`: Bestimmt, ob die Tabellenspalte bearbeitet werden kann (`true`) oder nicht (`false`).

`<maxLength></maxLength>`***:*** Gibt die maximale Anzahl von Zeichen für Felder vom Typ TEXT an. Für Felder anderer Datentypen hat die Einstellung keine Funktion.

`<width></width>`: Breite der jeweiligen Tabellenspalte in Pixeln. Standardmäßig richtet sich die Breite der Spalte nach der Spaltenüberschrift.

`<class></class>`: Name einer zusätzlichen CSS Klasse für die Zellen der Spalte.

`<event type=""></event>`: Registriert ein bestimmtes Ereignis (Event) mit dem Eingabeelement der Spalte. Als `type` wird hier der Name des Events verwendet (z.B. `onFocus`, `onBlur` oder `onChange`). Das Element `event` muss den Namen einer JavaScript Callback-Funktion enthalten, die beim Eintritt des Events ausgeführt werden soll.

`<option value=""></option>`: (darf mehrmals vorkommen). Definiert die Optionen, wenn die Spalte vom Typ **SELECT** ist. Im Attribut `value` kann ein technischer Wert angegeben werden, der gespeichert wird, wenn die Option ausgewählt ist. Dieser Wert kann internationalisiert werden. (siehe Abschnitt 2.4.1).

`<clearExisting></clearExisting>`: TODO Falls bei einer Tabellenspalte externe dynamische Daten in Klapplisten (SELECT) integriert werden, aber der aktuell gespeicherte Wert nicht mehr in der Datenbankabfrage vorhanden ist, wird dieser Eintrag aus der Klappliste entfernt (`true`, default). Ansonsten wird dieser Eintrag zusätzlich mit in die Auswahl genommen (`false`).

`<buttonLabel></buttonLabel>`: Enthält die Aufschrift des Buttons, wenn die Spalte vom Typ **BUTTON** ist.

`<img></img>`: Um herkömmliche Buttons als Elementtyp der Spalte zu grafischen Image-Buttons zu machen, kann hier eine Bilddatei (Dateiname mit Endung `.png` oder `.gif`) oder ein Entypo Font-Icon (Notation: „entypo:<Iconname>“) angegeben werden.

2.2.3 Einschränkungen des Inhalts

Für ein Feld können verschiedene Einschränkungen eingestellt werden:

Für ein Textfeld mit numerischer Eingabe muss ein Constraint `<constraint>NUMBER</constraint>` hinzugefügt werden. Das Feld wird dann beim Speichern auf die korrekte Eingabe überprüft.

Wenn das Feld als Pflichtfeld angelegt werden soll kann `<constraint>MANDATORY</constraint>` für dieses Feld angegeben werden.

2.2.4 Einschränkungen der Sichtbarkeit

Es ist möglich, die Sichtbarkeit von Zeilen und Spalten im Gentable-Plugin abhängig von verschiedenen Bedingungen zu steuern. Sowohl auf Zeilen- als auch auf Spaltenebene können Regeln definiert werden die:

- Ein Feld einer Zeile mit einem festen Wert vergleichen
- Ein Feld einer Zeile mit einem festen Wert vergleichen und prüfen ob der aktuelle Benutzer Mitglied eines bestimmten Zugriffsprofils ist
- Prüfen, ob der aktuelle Benutzer Mitglied eines bestimmten Zugriffsprofils ist
- Ein Mappenfeld mit einem festen Wert vergleichen
- Ein Mappenfeld mit einem festen Wert vergleichen und prüfen, ob der aktuelle Benutzer Mitglied eines bestimmten Zugriffsprofils ist
- Ein Mappenfeld mit dem Wert eines Feldes einer Zeile vergleichen
- Ein Mappenfeld mit dem Wert eines Feldes einer Zeile vergleichen und prüfen, ob der aktuelle Benutzer Mitglied eines bestimmten Zugriffsprofils ist
- Ein Mappenfeld und/oder ein Feld einer Zeile mit dem Wert eines systemweiten Autotexts oder einer Eigenschaft des aktuellen Benutzers vergleichen

Am Ende dieses Abschnitts werden diese verschiedenen Kombinationen in einer Beispiel-Definitionsdatei verwendet.

Zusätzliche Einstellungsmöglichkeiten:

- Es gibt zwei Regeltypen „`READONLY`“ und „`INVISIBLE`“
- Wird in einer Regel das Attribut „`invert`“ auf „`true`“ gesetzt, wird die Logik der Regel umgekehrt (Siehe Beispiel für `Field3`). Hier wird die Spalte READONLY für alle Benutzer die NICHT Mitglied des Zugriffsprofils „Administrator“ sind.

Wichtiger Hinweis:

Die Einschränkung der Sichtbarkeit hat lediglich Auswirkungen in der Web-Oberfläche. Das zugrunde liegende Datenmodell steht dem Browser grundsätzlich vollständig zur Verfügung, so dass die Einschränkung der Sichtbarkeit nicht als eine Einschränkung der Benutzerrechte verstanden werden darf.

Beispieldefinitionsdatei TODO

```
<?xml version="1.0" encoding="UTF-8"?>
<table_def name="erpInvoice">
  <xmlfield>erpAccountAssignment</xmlfield>
  <rowCondition>
        <rule type="READONLY" field="Field1" value="2222" />
        <rule type="READONLY" filefield="Creditor" value="ALFKI" />
        <rule type="READONLY" accessprofile="Warehouse" />
        <rule type="READONLY" field="Field5"
              value="Field5" accessprofile="Directors" />
        <rule type="READONLY" filefield="Creditor"
              value="OTRIS" accessprofile="Administration" />
        <rule type="READONLY" autotext="%currentUser.$AllowedAmount%"
              field="Field4" />
        <rule type="READONLY" autotext="%currentUser.$AllowedAmount%"
              filefield="Amount" />
  </rowCondition>

  <field number="1">
        <label>Field1</label>
        <title>Field1</title>
        <type>TEXT</type>
        <editable>true</editable>
        <condition>
               <rule type="READONLY" field="Field2" value="Field2" />
        </condition>
  </field>

  <field number="2">
        <label>Field2</label>
        <title>Field2</title>
        <type>TEXT</type>
        <editable>true</editable>
        <condition>
                 <rule type="READONLY" filefield="DocumentNr" value="1000" />
        </condition>
  </field>

  <field number="3">
        <label>Field3</label>
        <title>Field3</title>
```

```
        <type>TEXT</type>
        <editable>true</editable>
        <condition>
          <rule type="READONLY" invert=”true”
                accessprofile="Administration" />
        </condition>
</field>

<field number="4">
        <label>Field4</label>
        <title>Field4</title>
        <type>TEXT</type>
        <editable>true</editable>
        <condition>
         <rule type="READONLY" field="Field3" value="2000"
               accessprofile="Administration" />
        </condition>
</field>

<field number="5">
        <label>Field5</label>
        <title>Field5</title>
        <type>TEXT</type>
        <editable>true</editable>
        <condition>
           <rule type="READONLY" filefield="DocumentNr"
                 value="1111" accessprofile="Warehouse" />
        </condition>
</field>

<field number="6">
        <label>Field6</label>
        <title>Field6</title>
        <type>TEXT</type>
        <editable>true</editable>
        <condition>
          <rule type="READONLY" autotext="file:%Creditor%"
                field="Field6" />
        </condition>
</field>

<field number="7">
```

```
        <label>Field7</label>
        <title>Field7</title>
        <type>TEXT</type>
        <editable>true</editable>
        <condition>
          <rule type="READONLY"
                autotext="%currentUser.$AllowedAmount%"
                filefield="CostCenter" />
        </condition>
  </field>
</table_def>
```

2.2.5 Abfrage externer Datenbanken

`<sql></sql>`: Kann verwendet werden, um die Inhalte des Feldes aus einer externen Datenquelle zu beziehen. Hierbei wird die am Anfang des Dokuments angegebene Datenbankverbindung genutzt, um die Daten zu beziehen.

Die Abfrage der Datenbank kann für das Feld im Element `<sql></sql>` mit folgenden Unterelementen eingestellt werden:

`<query></query>`: Enthält den SQL-Befehl der zur Abfrage verwendet werden soll. Bei der Angabe des Befehls können verschiedene Platzhalter verwendet werden.

- `%userLogin%` **:** Wird durch den Loginnamen des aktuell angemeldeten Benutzers ersetzt.
- `%FILE_FIELD:Mappenfeldname%` **:** Wird durch den Inhalt des jeweiligen Mappenfeldes der aktuellen Mappe ersetzt.
- `%FIELD:Spaltenname%` **:** Wird durch den Wert der jeweiligen Spalte der aktuellen Tabellenzeile ersetzt.

Es ist darauf zu achten, dass alle Platzhalter immer in **Hochkommata** (z.B. `'%FIELD:Artikel%'`) eingeschlossen sind. Externe Daten können in Klapplisten, Textfelder und mehrzeilige Textfelder integriert werden und TODO Checkboxen können davon abhängig angekreuzt werden (DB-Eintrag `true`) oder nicht (DB-Eintrag `false`).

`<result></result>`: Enthält den Namen der Datenbankspalte, die für die Abfrage der Daten verwendet werden soll.

`<key></key>`: Enthält den Namen der Datenbankspalte, die den technischen Wert des Ergebnisses enthält.

2.2.6 Definition von Buttons

Mit dem Element `<button></button>` können **optional** ein oder mehrere benutzerdefinierte Buttons erstellt werden.

Die Buttons können jeweils über folgende Unterelemente eingestellt werden.

`<label></label>`: Aufschrift des Buttons. (Kann internationalisiert werden, siehe Abschnitt 2.4.1).

`<function></function>`: Funktionsaufruf, der ausgeführt wird, wenn der Button angeklickt wird. **ACHTUNG**: Funktionen immer ohne Parameter eintragen, auch wenn deren Signaturen Parameter enthalten!

`<accessKey></accessKey>`: Enthält einen Buchstaben. Durch Drücken der ALT-Taste zusammen mit diesem Buchstaben kann die Funktion des Buttons über die Tastatur ausgelöst werden.

`<img></img>`: Um benutzerdefinierte Buttons zu grafischen Image-Buttons zu machen, kann hier eine Bilddatei (Dateiname mit Endung `.png` oder `.gif`) oder ein Entypo Font-Icon (Notation: „entypo:<Iconname>“) angegeben werden.

2.2.7 Mehrere Konfigurationsdateien für einen Mappentypen

Es ist möglich, für einen Mappentypen die Verwendung von mehreren Definitionsdateien zu ermöglichen. Hierbei kann der Benutzer entscheiden, welche Definitionsdatei für das Gentable-Plugin zu einer bestimmten Zeit bzw. bei einem bestimmten Workflow-Zustand verwendet werden soll. Dies kann über die Eigenschaft „genTableDefField“ eingestellt werden. Der Wert dieser Eigenschaft muss dem technischen Namen eines Auswahlfeldes im Mappentypen entsprechen. Dieses Auswahlfeld muss als Auswahlmöglichkeiten die Namen der verschiedenen zur Verfügung stehenden Definitionsdateien beinhalten.

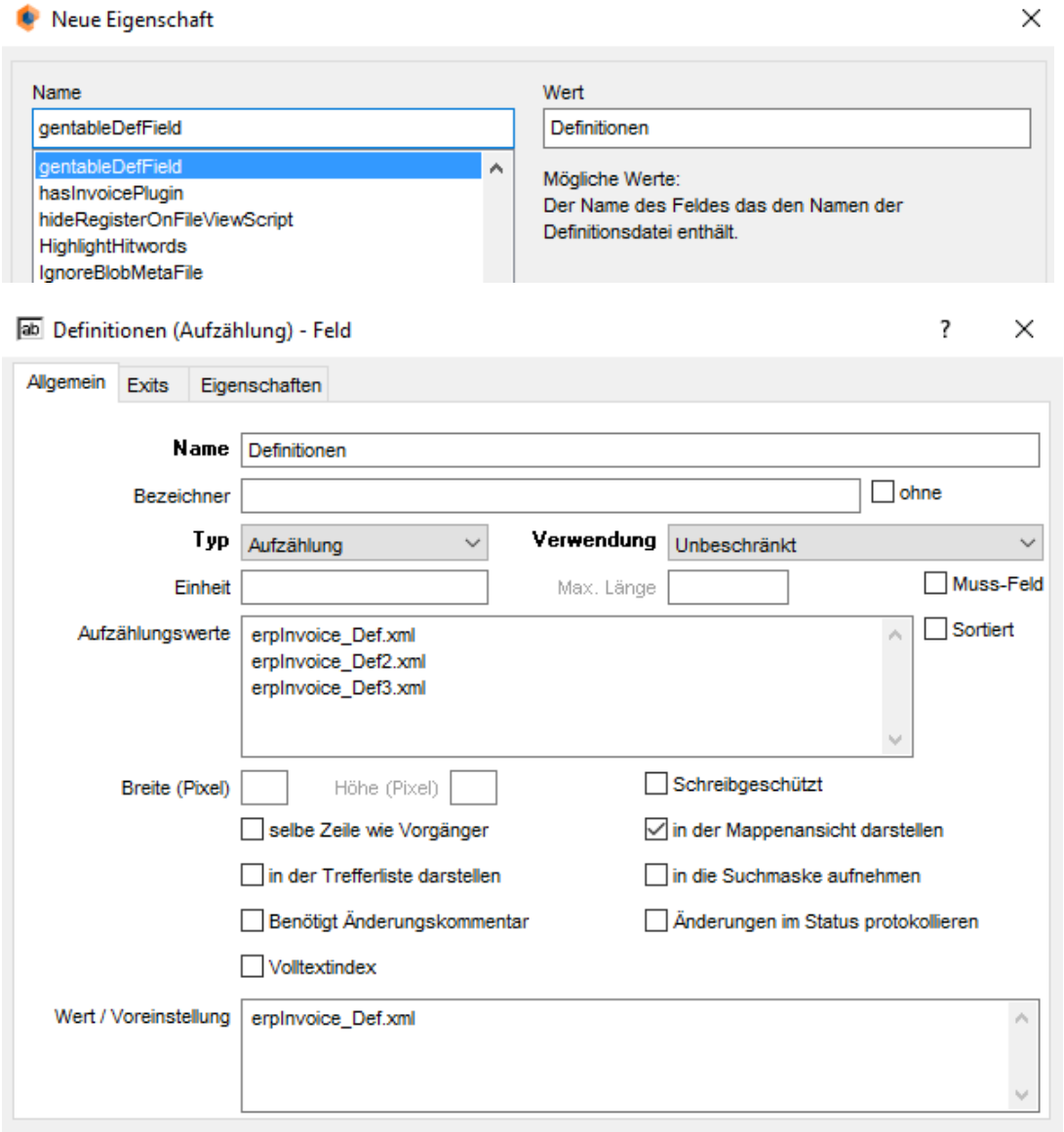


Abb. 6: Eigenschaft gentableDefField

2.3 Anpassung der benutzerdefinierten Skript-Datei

Es gibt die Möglichkeit, eine Skript-Datei bzw. JSP-Datei mit benutzerdefinierten JavaScript-Funktionen in **GenTable** einzubinden. Im Standardfall ist dies die mitgelieferte Datei `UserdefinedScripts.jsp`. In dieser Datei sind wichtige Funktionalitäten implementiert wie z.B. das Kopieren von Tabellenzeilen, Löschen von Tabellenzeilen, Anhängen einer neuen

Tabellenzeile und Verschieben von Tabellenzeilen. Des Weiteren sind aber auch spezifischere Funktionalitäten für den Einsatz zur Rechnungseingangsprüfung als **Invoice-Plugin** implementiert. Dies sind z.B. Funktionen zum Berechnen der einzelnen Positionspreise, zum Berechnen und Prüfen der Rechnungs-Gesamtsumme, zum Formatieren der Preisfelder und zur Splittbuchung.

2.4 Internationalisierung

GenTable kann mehrsprachig konfiguriert und angezeigt werden. So werden z.B. sämtliche Meldungen der Software-Komponente in der aktuellen Sprache des angemeldeten Benutzers ausgegeben. Vorkonfiguriert sind die Sprachen Deutsch und Englisch. Die einzelnen Meldungen können leicht geändert werden und es können zusätzlich weitere Sprachen hinzugefügt werden. Die originalen Sprachdateien befinden sich im Unterverzeichnis `.../tomcat8/webapps/documents/web/WEB-INF/classes/` der **DOCUMENTS 5** – Installation. Als Alternative dazu können die Dateien auch als sogenannte externe Ressourcen in **DOCUMENTS 5** eingebunden werden. Die Sprachdateien tragen alle den Namen `DocumentsMessages_<language>.properties`. Anstelle von `<language>` im Dateinamen steht jeweils das Kürzel der Sprache nach ISO-Norm (z.B. `de` für Deutsch, `en` für Englisch). In diesen `properties`-Dateien sind die einzelnen Sprachmeldungen als Wertepaare definiert. Die Standardsprache ist Deutsch und die zugehörigen Meldungen dafür sind in der Datei `DocumentsMessages_de.properties` hinterlegt. Alle für **GenTable** relevanten Meldungen der `properties`-Dateien beginnen stets mit dem Bezeichner `Documents.Gentable`.

Zum Erstellen einer neuen Sprachdatei ist einfach eine der vorhandenen Sprachdateien zu kopieren, das Sprachkürzel im Dateinamen entsprechend zu ändern und die einzelnen Sprachausgaben in der Datei an die entsprechende Sprache anzupassen.

2.4.1 Internationalisierung in der Konfigurationsdatei

Alle `<label></label>` und `<option></option>` Elemente in der Konfigurationsdatei können internationalisierte Inhalte enthalten. Hierzu wird die folgende Syntax verwendet:

```
<label>Default-Wert;de:Deutsch;en:Englisch;fr:Französich</label>
```

Der Default-Wert wird im Falle von `<label>` angezeigt wenn die Sprache, die für den aktuellen Benutzer eingestellt ist, nicht in der Beschreibung vorhanden ist. Im Falle von `<option>` dient der Default-Wert zusätzlich noch als technischer Wert für die Speicherung der Option.

3. Implementierung eigener Funktionalität

3.1 Einführung

Zur Anpassung von **GenTable** ist es möglich, weitere selbst implementierte Funktionalität zu integrieren. Dafür werden jedoch grundlegende Programmierkenntnisse in der Sprache JavaScript vorausgesetzt. Die wichtigsten Klassen für die Programmierung mit **GenTable** stellen der `GentableContext`, das `GentableGridModel`, das `GentableGrid ColumnModel` und das `GentableGridRowModel` dar, welche im weiteren Verlauf des Kapitels detailliert vorgestellt werden.

Das `GentableGridModel` bringt bereits schon einen großen Teil wichtiger Standard-Funktionalitäten und auch auf den Einsatz als **Invoice-Plugin** zugeschnittene Funktionalitäten mit, die dort und in weiteren `Gentable`-Klassen in unterschiedlichen Funktionen implementiert sind. Eine vollständige Auflistung und Beschreibung aller `Gentable`-Klassen finden sie zusätzlich in der API Dokumentation für das **DOCUMENTS 5** Client SDK.

3.2 Registrierung und Callback-Funktionen

Jegliche, selbst implementierte fachliche Funktionalität wird innerhalb sogenannter Callbacks implementiert. Diese Callbacks sind gewöhnliche JavaScript-Funktionen, welche unter einem bestimmten Namen oder für einen bestimmten Zeitpunkt registriert werden.

3.3 Callback-Funktionen für Buttons

Um einen Button auf der **GenTable** Toolbar zu erzeugen, welcher eine selbst programmierte Funktion ausführen soll, sind grundsätzlich zwei Schritte erforderlich. Zunächst muss der Button in der **GenTable** XML-Konfigurationsdatei definiert werden (siehe 2.2.62.2.6). Die Definition beinhaltet neben dem Label (Tag `<label>`) auch den Namen der Callback-Funktion (Tag `<function>`), welche ausgeführt werden soll, sobald der Benutzer den Button betätigt hat. Im folgenden Beispiel ist das für den Button `Neue Zeile` die Funktion `appendNewRow`.

```
<button>
  <label>Neue Zeile</label>
  <function>appendNewRow</function>
</button>
```

Im zweiten Schritt muss in der JavaScript-Datei, in der die Funktionalität für den Button hinterlegt werden soll, die Callback-Funktion unter dem in der Definition angegebenen Namen registriert und implementiert werden.

Das folgende Code-Beispiel zeigt, wie eine Callback-Funktion für das einfache Hinzufügen einer Zeile für einen Mappentypen `ftInvoice` unter dem Namen `appendNewRow` in der `gentableRegistry` registriert wird. Die eigentliche Implementierung folgt gleich darauf innerhalb des Callbacks `function(documentsContext) { ... }`.

```
documents.sdk.gentable.gentableRegistry.registerCallback("ftInvoice",
    "appendNewRow", function(documentsContext) {

    var gridModel = documentsContext.getGentableContext().getGridModel(),
        row = gridModel.addRow();
});
```

3.4 GentableContext

Der `GentableContext` vereint sämtliche Funktionen, die im Zusammenhang mit der Implementierung von eigener Funktionalität für **GenTable** von Bedeutung sind.

Die Funktion `getGridModel()` gibt das `GentableGridModel` der aktuellen Tabelle zurück. Das `GentableGridModel` stellt verschiedene grundlegende Funktionen zur Abfrage und Manipulation von Spalten und Zeilen der Tabelle bereit.

3.5 GentableGridModel

Das `GentableGridModel` stellt das komplette clientseitige Datenmodell einer Tabelle in **GenTable** bereit. Die Spalten und Zeilen dieses Modells werden jeweils durch Objekte der beiden Typen `GentableGridColumnModel` und `GentableGridRowModel` abgebildet, welche vom `GentableGridModel` aus über verschiedene Funktionen direkt erreichbar sind. Es ist grundlegend wichtig zu verstehen, dass sich jegliche Manipulation dieses Datenmodells i.d.R. unmittelbar auf die Anzeige von **GenTable** auswirkt. Das bedeutet, dass in diesen Fällen entweder die gesamte Tabelle oder nur einzelne Teilbereiche vom Browser automatisch neu gerendert werden, sobald beispielsweise Zeilen hinzugefügt, gelöscht, manipuliert, kopiert oder verschoben werden. Zugriffe und Manipulationen der Tabelle über das browserinterne Document Object Model (DOM) sind somit grundsätzlich nicht mehr erforderlich und entfallen im Vergleich zu früheren Versionen von **GenTable** komplett. Dadurch wird die Implementierung eigener Funktionalität erheblich vereinfacht. Von jedweder manuellen Manipulation der **GenTable**-GUI über das native DOM (z.B. mittels jQuery o.ä.) hingegen wird im Übrigen grundsätzlich abgeraten, da in diesem Fall weder Kompatibilität zu neueren Versionen noch ein einwandfreies Laufzeitverhalten gewährleistet werden kann. Es sollte stets die in diesem Kapitel vorgestellte API verwendet werden.

Um sich in selbst implementierten Callback-Funktionen eine Referenz auf das aktuelle `GentableGridModel` zu beschaffen, sollte zu Beginn stets der nachfolgend abgebildete Code verwendet werden.

```
var gridModel = documentsContext.getGentableContext().getGridModel();
```

Es folgt nun eine Auflistung der wichtigsten Funktionen auf dem `GentableGridModel`.

Die Funktion `getColumn(name)` gibt das Datenmodell einer beliebigen Spalte aus dem `GentableGridModel` zurück. Zur eindeutigen Bestimmung der Spalte muss der Parameter `name` angegeben werden. Der Rückgabewert ist vom Typ `GentableGridColumnModel`. Auf diesem Modell lassen sich u.a. verschiedene Informationen einer Spalte abfragen.

Die Funktion `getColumns()` gibt sämtliche Spalten auf dem `GentableGridModel` als `GentableGridColumnCollection` zurück.

Die Funktion `columnsSize()` gibt die Anzahl der Spalten auf dem aktuellen `GentableGridModel` als `Integer` zurück.

Die Funktion `getRow(index)` gibt das Datenmodell einer beliebigen Zeile aus dem `GentableGridModel` zurück. Zur eindeutigen Bestimmung der Zeile muss der Parameter `index` angegeben werden. Es ist zu beachten, dass die erste Zeile stets dem Index `0` entspricht. Der Rückgabewert ist vom Typ `GentableGridRowModel`. Auf diesem Modell lassen sich u.a. die einzelnen Zellen einer Zeile lesen und mit neuen Werten besetzen.

Die Funktion `getRows()` gibt sämtliche Zeilen auf dem `GentableGridModel` als `GentableGridRowCollection` zurück.

Die Funktion `rowsSize()` gibt die Anzahl der Zeilen auf dem aktuellen `GentableGridModel` als `Integer` zurück.

Die Funktion `addRow(row)` fügt dem `GentableGridModel` eine neue Zeile hinzu. Der Rückgabewert ist vom Typ `GentableGridRowModel`.

Die Funktion `getSelectedRows()` gibt die Menge aller aktuell über die Checkbox selektierten Zeilen auf dem `GentableGridModel` als `GridCollection` zurück.

Die Funktion `setSelectedRows(rowIndexes)` setzt die Selektion zunächst komplett zurück, so dass keine Zeile mehr selektiert ist. Der Parameter `rowIndexes` bestimmt alle neu zu auszuwählenden Zeilen als `Array` von `Integer`. Es ist zu beachten, dass die erste Zeile stets dem Index `0` entspricht. So würde beispielsweise mit dem Argument `[0, 2]` die erste und die dritte Zeile selektiert werden.

Die Funktion `resetSelection()` setzt die aktuelle Selektion sämtlicher Zeilen komplett zurück, so dass keine Zeile mehr selektiert ist.

Die Funktion `copyRow(srcIndex, dstIndex)` kopiert eine komplette Zeile. Dabei bestimmen die Parameter `srcIndex` und `dstIndex` den Quell- und Zielindex der zu kopierenden Zeile.

Die Funktion `moveRow(srcIndex, dstIndex)` verschiebt eine komplette Zeile innerhalb des `GentableGridModel`. Dabei bestimmen die Parameter `srcIndex` und `dstIndex` den Quell- und Zielindex der zu verschiebenden Zeile.

Die Funktion `removeRow(index)` entfernt eine komplette Zeile aus dem `GentableGridModel`. Dabei bestimmt der Parameter `index` den Index der zu löschenden Zeile.

3.6 GentableGridColumnModel

Das `GentableGridColumnModel` repräsentiert das Datenmodell einer Spalte in **GenTable**. Da die Definition der einzelnen Spalten in der XML-Konfiguration festgelegt wird, kann an dieser Stelle nur ein lesender Zugriff erfolgen. Es folgt nun eine kurze Auflistung der wichtigsten Funktionen auf dem `GentableGridColumnModel`.

Die Funktion `getName()` gibt den technischen Namen der Spalte zurück.

Die Funktion `getLabel()` gibt das sprachenabhängige Label der Spalte zurück.

Die Funktion `getDataType()` gibt den Datentyp der Spalte als `String` zurück.

Die Funktion `isVisible()` gibt `true` zurück, falls die Spalte sichtbar ist, ansonsten `false`.

Die Funktion `isEditable()` gibt `true` zurück, falls die Spalte editierbar ist, ansonsten `false`.

3.7 GentableGridRowModel

Das `GentableGridRowModel` repräsentiert das Datenmodell einer Spalte in **GenTable**.Es folgt nun eine kurze Auflistung der wichtigsten Funktionen auf dem `GentableGrid RowModel`.

Die Funktion `index()` gibt den Index der Zeile zurück. Es ist zu beachten, dass die erste Zeile stets dem Index `0` entspricht.

Die Funktion `get(columnName)` gibt den Wert einer bestimmten Zelle in der Zeile zurück. Der Parameter `columnName` bestimmt dabei die gewünschte Spalte.

Die Funktion `getInt(columnName)` gibt den Wert einer bestimmten Zelle in der Zeile als `Integer` zurück. Der Parameter `columnName` bestimmt dabei die gewünschte Spalte.

Die Funktion `getFloat(columnName)` gibt den Wert einer bestimmten Zelle in der Zeile als `Float` zurück. Der Parameter `columnName` bestimmt dabei die gewünschte Spalte.

Die Funktion `set(columnName, columnValue)` setzt den Wert einer bestimmten Zelle in der Zeile. Die Parameter `columnName` und `columnValue` bestimmen dabei die gewünschte Spalte und den neuen Wert der Zelle.

Die Funktion `isVisible()` gibt den Wert `true` zurück, falls die Zeile sichtbar ist, ansonsten `false`.

4. Einsatzbeispiele & Standardfunktionen

Im Folgenden werden die am häufigsten verwendeten Funktionen für den Einsatz als **Invoice-Plugin** erläutert. Nach dem Laden einer Mappe des Typs *„ftInvoice“* sieht die Anzeige in etwa so aus:

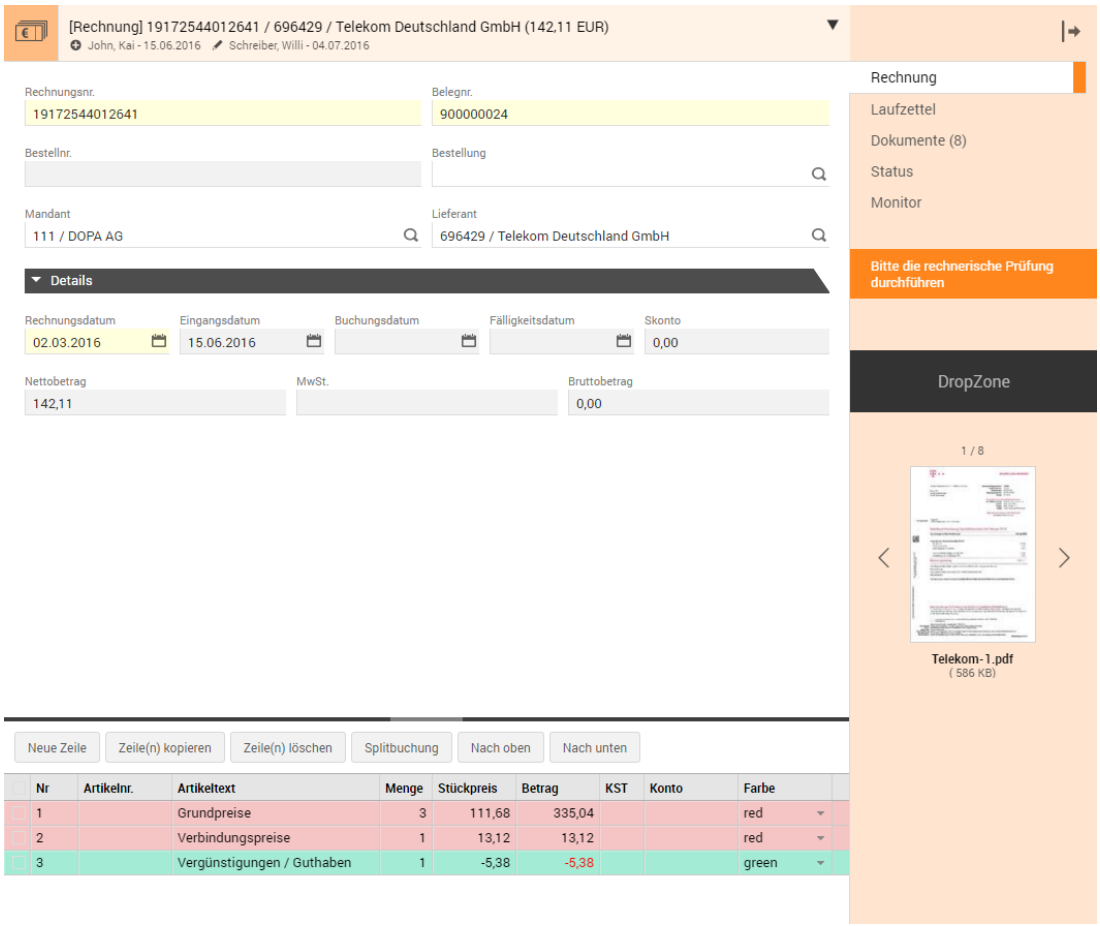


Abb. 7: Mappe mit Invoice-Plugin

In den Feldern der Mappe und in den Rechnungs-Positionen (Tabellenzeilen) wurden einige Beispieleinträge gemacht. In der nachfolgenden Abbildung ist durch den entsprechenden Button eine neue Zeile angefügt worden.

Abb. 8: Neue Zeile anfügen

Neue Zeile | Zeile(n) kopieren | Zeile(n) löschen | Splitbuchung | Nach oben | Nach unten

	Nr	Artikelnr.	Artikeltext	Menge	Stückpreis	Betrag	KST
☑	1		Grundpreise	3	111,68	335,04	
☐	2		Verbindungspreise	1	13,12	13,12	
☑	3		Vergünstigungen / Guthaben	1	-5,38	-5,38	
☐	4						

Abb. 9: Zeile(n) auswählen

Die zwei über die Index-Checkboxen ausgewählten Zeilen sollen kopiert werden.

Neue Zeile | Zeile(n) kopieren | Zeile(n) löschen | Splitbuchung | Nach oben | Nach unten

	Nr	Artikelnr.	Artikeltext	Menge	Stückpreis	Betrag	KST
☐	1		Grundpreise	3	111,68	335,04	
☑	2		Grundpreise	3	111,68	335,04	
☐	3		Verbindungspreise	1	13,12	13,12	
☐	4		Vergünstigungen / Guthaben	1	-5,38	-5,38	
☑	5		Vergünstigungen / Guthaben	1	-5,38	-5,38	
☐	6						

Abb. 10: Zeile(n) kopieren

Nach dem Kopieren der Zeilen werden nun Zeilen ausgewählt und danach gelöscht.

Neue Zeile | Zeile(n) kopieren | Zeile(n) löschen | Splitbuchung | Nach oben | Nach unten

	Nr	Artikelnr.	Artikeltext	Menge	Stückpreis	Betrag	KST
☐	1		Grundpreise	3	111,68	335,04	
☐	2		Verbindungspreise	2	13,12	26,24	
☐	3		Vergünstigungen / Guthaben	1	-5,38	-5,38	
☐	4		Vergünstigungen / Guthaben	1	-5,38	-5,38	

Abb. 11: Zeile(n) löschen

Über die Funktion *„Splittbuchung“* (hier Zeile 2) werden ausgewählte Zeilen gleichzeitig kopiert und die Einträge für Menge und Betrag jeweils halbiert.

Neue Zeile | Zeile(n) kopieren | Zeile(n) löschen | Splitbuchung | Nach oben | Nach unten

	Nr	Artikelnr.	Artikeltext	Menge	Stückpreis	Betrag	KST
☐	1		Grundpreise	3	111,68	335,04	
☑	2		Verbindungspreise	1	13,12	13.12	
☑	3		Verbindungspreise	1	13,12	13.12	
☐	4		Vergünstigungen / Guthaben	1	-5,38	-5,38	
☐	5		Vergünstigungen / Guthaben	1	-5,38	-5,38	

Abb. 12: Splittbuchung durchgeführt

Auch können bestimmte Felder aus den Einträgen anderer Felder automatisch berechnet werden, wie z.B. der Betrag aus Menge und Stückpreis.

Beim Prüfen des Nettobetrags erhält man je nach Ergebnis des Vergleiches mit dem Mappen-Betrag eine bestimmte Meldung.

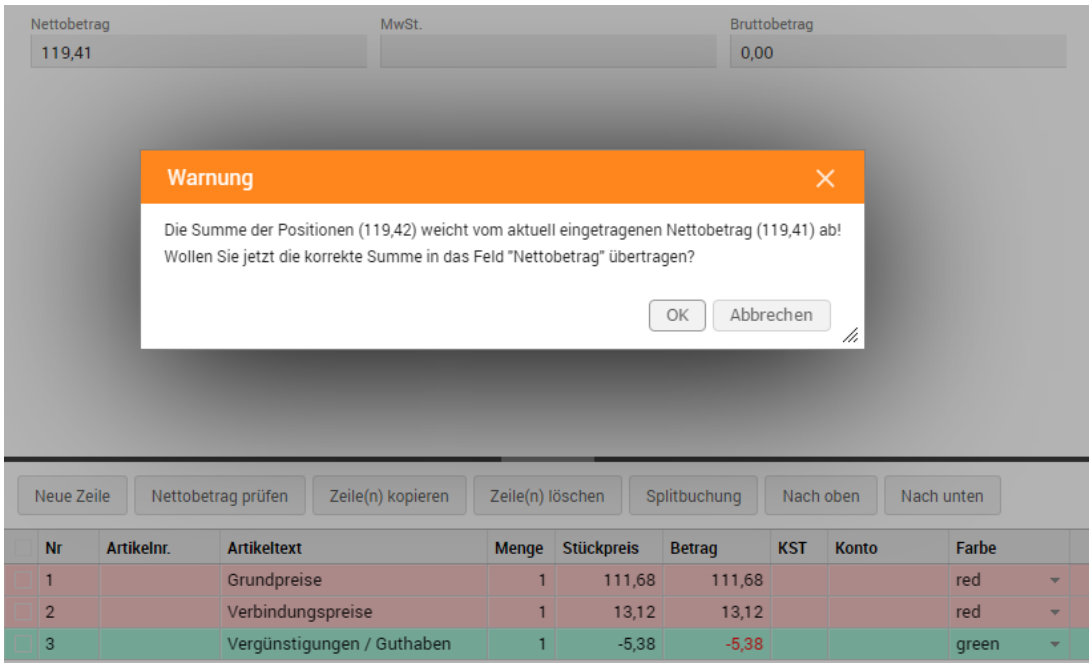


Abb. 13: Nettobetrag prüfen

Wenn bestimmte Tabellenspalten/Felder je nach Konfiguration nur Zahlenwerte enthalten dürfen oder nicht leer bleiben dürfen, erscheint beim Speichern eine entsprechende Meldung. Erst wenn alle Bedingungen erfüllt sind, kann die Tabelle gespeichert werden.

Abb. 14: Überprüfung beim Speichern

5. Fehler & Ursachen

Falls keine XML-Konfigurationsdatei für den jeweiligen Mappentyp existiert, der Name falsch geschrieben wurde oder die Datei leer ist, kann **GenTable** nicht gestartet werden und wird mit folgender Fehlermeldung abgebrochen:

Abb. 15: Fehlende XML-Konfigurationsdatei

Wenn die XML-Konfigurationsdatei nicht korrekt gelesen werden kann, die Datei kein valides XML beinhaltet oder bestimmte, nicht erlaubte Sonderzeichen als XML-Einträge (z.B. „<“, „>“, „&“ etc.) verwendet werden, so wird die Ausführung von **GenTable** mit folgender Fehlermeldung abgebrochen:

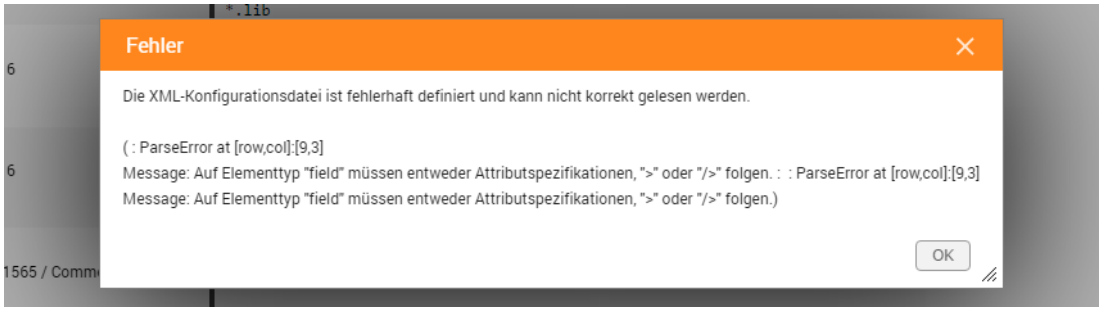


Abb. 16: XML-Konfigurationsdatei fehlerhaft

Der folgende Fehler tritt auf, falls der XML-Eintrag `<xmlfield>` in der XML-Konfigurationsdatei fehlt, falsch definiert oder leer ist.

Abb. 17: Kein Konfigurationseintrag für Mappenfeld oder leer

Folgende Fehlermeldung erscheint, wenn das in der XML-Konfigurationsdatei definierte Mappenfeld für die Tabellendaten nicht im Mappentyp selbst definiert bzw. vorhanden ist:

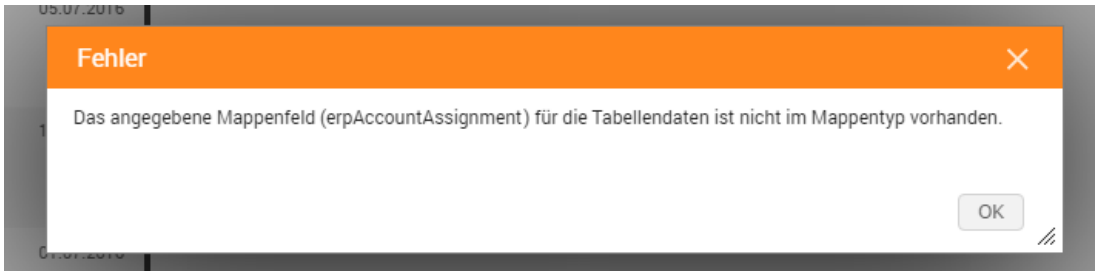


Abb. 18: Mappenfeld nicht vorhanden

Es muss stets mindestens eine Tabellenspalte in der XML-Konfigurationsdatei definiert sein:

Abb. 19: Keine Tabellenspalten

Falls externe Daten aus einer Datenbank integriert werden sollen, so muss ein passender JDBC-Datenbank-Treiber vorhanden sein:

Abb. 20: Datenbanktreiber

Die SQL-Syntax muss immer korrekt eingehalten werden, ansonsten treten Fehler bei der Datenbank-Anfrage auf. Nähere Hinweise werden in der Fehlermeldung angezeigt:

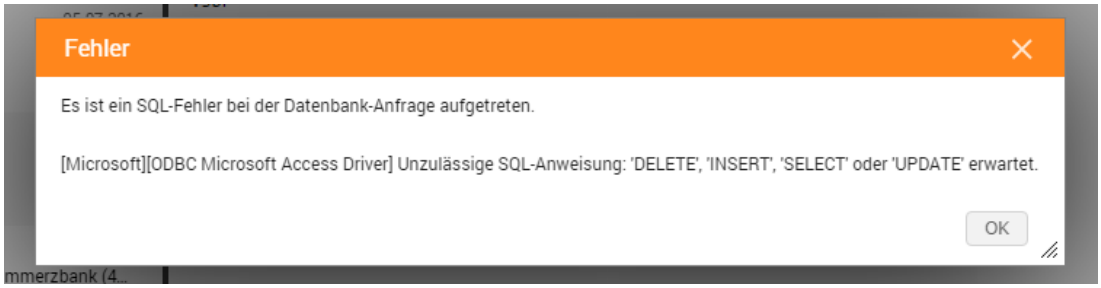


Abb. 21: Datenbankabfrage

6. Abbildungsverzeichnis

Abb. 1: Baumstruktur im DOCUMENTS-Manager 6
Abb. 2: Bearbeiten bzw. Erstellen eines Mappentyps 7
Abb. 3: Eigenschaften eines Mappentyps 7
Abb. 4: Bearbeiten bzw. Erstellen von Feldern 8
Abb. 5: Mappenfeld-Dialog 8
Abb. 6: Eigenschaft gentableDefField 17
Abb. 7: Mappe mit Invoice-Plugin 23
Abb. 8: Neue Zeile anfügen 23
Abb. 9: Zeile(n) auswählen 24
Abb. 10: Zeile(n) kopieren 24
Abb. 11: Zeile(n) löschen 24
Abb. 12: Splittbuchung durchgeführt 24
Abb. 13: Nettobetrag prüfen 25
Abb. 14: Überprüfung beim Speichern 25
Abb. 15: Fehlende XML-Konfigurationsdatei 26
Abb. 16: XML-Konfigurationsdatei fehlerhaft 26
Abb. 17: Kein Konfigurationseintrag für Mappenfeld oder leer 26
Abb. 18: Mappenfeld nicht vorhanden 27
Abb. 19: Keine Tabellenspalten 27
Abb. 20: Datenbanktreiber 27
Abb. 21: Datenbankabfrage 27