

unidb and DH

Developer Guide

Version 1.0

server2/unidb/unidb.html

Hilfe Einstellungen abmelden neues Dokument
manuelle Eingabe gel. Dok. zeigen
Administration

Inhalt

- Administration
- Aktien
- aktuell
- Benutzerhandbuch
- Bilder
- DH & unidb
- Formulare
- Sicherungen

suchen

Formular Tabellenansicht speichern 1 / 83 Filter: Filter erstellen Sortierung

Status

Notizen_ID 2 Aufwand in h 3 erledigt in Bearbeitung zurückgestellt offen

Priorität

Betreff Formular - Feldeigenschaften - editierbar - aktiviert - sichtbar

Datenfelder müssen die Eigenschaften editierbar, aktiviert und sichtbar bekommen. Beispiel: Notizen_ID in diesem Formular sollte nur sichtbar, aber nicht editierbar sein.

Dokument Version Abfrage Datenbank Tabellen Abfragetyp Beschriftung

User_Tags

User_Akt

Ergebnis Abfrage

Point_Path	Tagname	Point_ID	Timestamp	Value
/	T25	250	2021-09-21 12:45:34	12.937
/	T25	250	2021-09-21 12:45:34	12.937
/	T25	250	2021-09-21 12:46:41	13
/	T25	250	2021-09-21 12:46:41	13
/	T25	250	2021-09-21 12:47:47	12.812
/	T25	250	2021-09-21 12:47:47	12.812

SQL

```
SELECT 'User_Tags'.'Point_Path', 'User_Tags'.'Tagname', 'User_akt'.'Point_ID', 'User_akt'.'Timestamp', 'U'.
```

ALTERNATIVEDRUCK

Feld	Point_Path	Tagname	Point_ID	Timestamp	Value
Funktion					
Alias					
Tabelle	User_Tags	User_Tags	User_akt	User_akt	User_akt
Sortierung					
gruppieren					
anzeigen					
Kriterium	like 'T25'				
Kriterium					

Dokument	Spalten & Zeilen	Format	Rahmen	DH Funktionen	Hilfe
	(A)	(B)	(C)	(D)	
0					
1	/L2				
2	Rohwerte in mm	Zeitstempel	Unix Zeitstempel		
3	1406	2019-03-03 16:00:00	1551625200		
4	1397	2019-03-09 13:00:00	1552132800	83097	
5	1371	2019-03-24 11:15:22	1553422522	199181	
6	1363	2019-03-31 11:07:28	1554023248	84060	
7	1344	2019-04-20 15:45:15	1555767915	164401	
8	1327	2019-05-01 19:50:23	1556733023	67140	
9	1315	2019-05-11 14:45:58	1557578758	78518	
10	1305	2019-05-18 19:47:49	1558201669	57211	

Hilfe Gruppe bearbeiten löschen verschieben

PI 1 belegter Speicher	benötigter RAM	2020-01-11 17:47:24	344.3	MByte
PI 1 Buffers	Puffer	2020-01-11 17:47:24	161692	KByte
PI 1 Cached	Cached	2020-01-11 17:47:24	394728	KByte
PI 1 CPU Temp	PI 1 CPU Temp	2020-01-11 17:47:24	38.5	°C
PI 1 Load_avg	PI 1 Auslastung	2020-01-11 17:47:24	9	%
PI 1 Memfree	freier Speicher	2020-01-11 17:47:24	39324	KByte
PI 1 Speicher frei	freier Speicher	2020-01-11 17:47:24	62.8	%
PI 1 SwapCached	Auslagerungsspeicher	2020-01-11 17:47:24	72	KByte
PI 1 root	Server Root Partition	2020-01-11 17:47:24	16.855	GByte

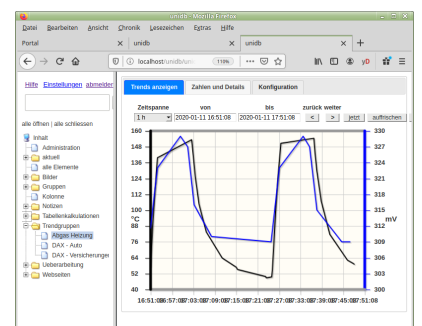
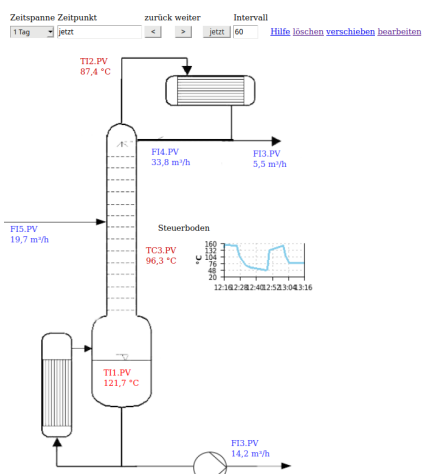


table of contents

Terms and abbreviations	2
Formatting used	2
1. unidb	3
1.1. Elements and general dialogs	3
1.1.1. Elements and their properties	3
1.1.2. conditional formatting.....	10
1.1.3. own JS code	13
1.2. Edit documents.....	13
1.2.1. query	13
1.2.2. Form.....	16
1.2.3. Report	26
1.3. Functions	33
1.4. programming	36
2. DH	38
2.1. Collect, process and provide data	38
2.2. Edit DH documents.....	38
2.2.1. Groups	38
2.2.2. drawings.....	43
2.3. settings	45
2.3.1. Manage building blocks for images.....	45
2.3.2. Manage multistates for drawings.	45
2.3.3. Configure devices	47
2.4. functions	47

Terms and abbreviations

DataHistorian	Database application which continuously collects data in the format data point - time stamp - value and makes this available again in a processed form.
DH	Figreviation for DataHistorian
TimeSeriesDatabase	An alternative name for a DataHistorian.
DB	Figreviation for database.
Tag	Data point in a DataHistorian
Point	Another name for a tag / data point.
Point_ID	Unique number of a tag, comparable to a serial number.
WYSIWYG	What You See Is What You Get (What you see is what you get.) There are two types of HTML editors. The simplest way is the one where you can see the HTML code and have to write it yourself. The second type is the type that is easier for the user. The HTML code remains hidden and instead you can see the fully formatted text (WYSIWYG).

Formatting used

bold font	Name of a control element (switch, text field, ...)
<i>italic</i>	Names of properties, etc.
highlighted in color	You should pay a little more attention to a text with a colored background.

1. unidb

This manual only deals with editing the query, form and report document types in design mode. The unidb uses MariaDB as a database. **Therefore, you should already have some knowledge of relational databases.** It would go beyond the scope of this manual if we first dealt with the basics of relational databases.

1.1.Elements and general dialogs

Dialogs can be called up via the headers of the documents, which can be found again and again in the design mode of forms, reports or images.
A document consists of elements that can occur in the same form in other document types.
It makes sense if we look at the elements and dialogs before we start working on the documents.

1.1.1.Elements and their properties

When you create a new document, it is initially empty. Although we can edit the properties of the document, we still only see an empty document. Only when we start to insert some objects, or better, elements, do we see a little more of them.
Real world vs. virtual world:
Please imagine we're building a house. We are the craftsmen and the future users are the residents of the house. First of all, a foundation has to be found. We have already done this in the virtual world by creating an empty document. A house consists of many parts, such as walls, ceilings, roof, windows, doors, etc. In the virtual world, this corresponds to the elements in our document. Every component of the house has certain properties such as width, height, color, etc. In our virtual world it is no different. Properties, such as the position in the document, the width, the height, etc. we find in every element. In addition, the elements also have special properties. For example, a data field in a form has the property database field. This property makes no sense in a simple text element that only displays a fixed text. Therefore we only see this property in the element database field.
Another parallel between the real and the virtual world is the grouping of objects or elements. Example: The roof of the house consists of beams, insulation, shingles, rain gutters, etc. A tab on a form is made up of tabs. These in turn contain further elements or even a further register.

Create element:

To create a new element, simply press the Create button in the Element area of the header. The dialog shown below opens.

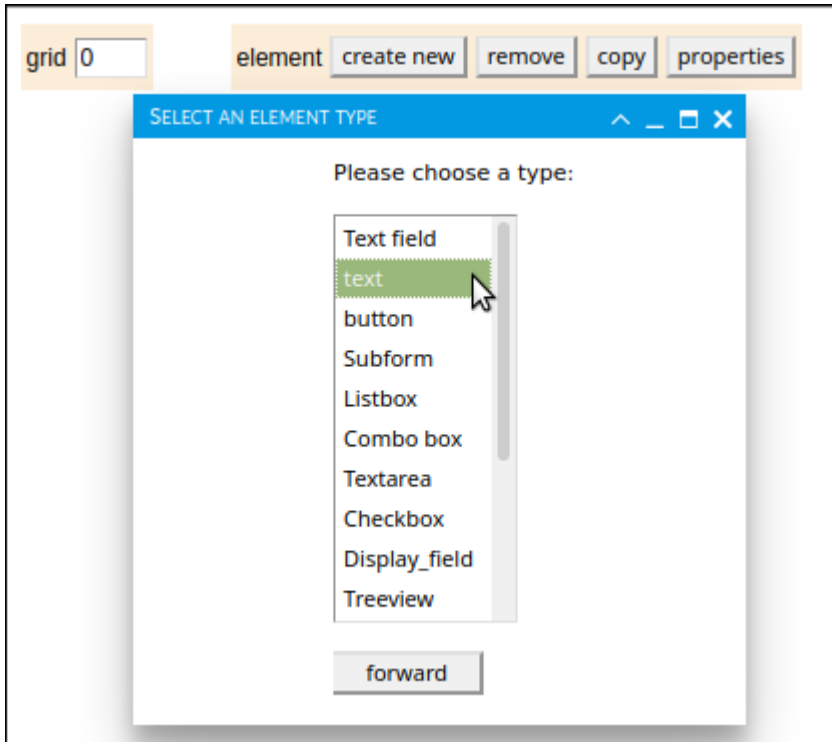


Fig 1: Element selection

In this example we select the Text option. A simple element is then created, which only displays a specified text.



Fig 2: newly created element

The new element appears as a red rectangle. The red color ensures that you can easily find the new element. If it were white, you would have to look for it on the white background. We cannot use the element the way it is presented. It's in the wrong position, the text in it is wrong and the formatting leaves something to be desired. So let's change the properties by highlighting it by clicking the left mouse button. The red color disappears. It is no longer necessary because we have found the element.

The element now looks like this:

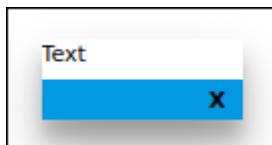


Fig 3: marked element

A click with the right mouse button on the element presents us with a context menu from which we select the Properties option.

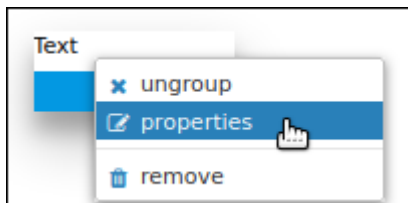


Fig 4: Select properties

A dialog opens like the one we see in Figure 5. You can also open this dialog via the Properties button in the Element of the header area. It is only important that the element is marked.

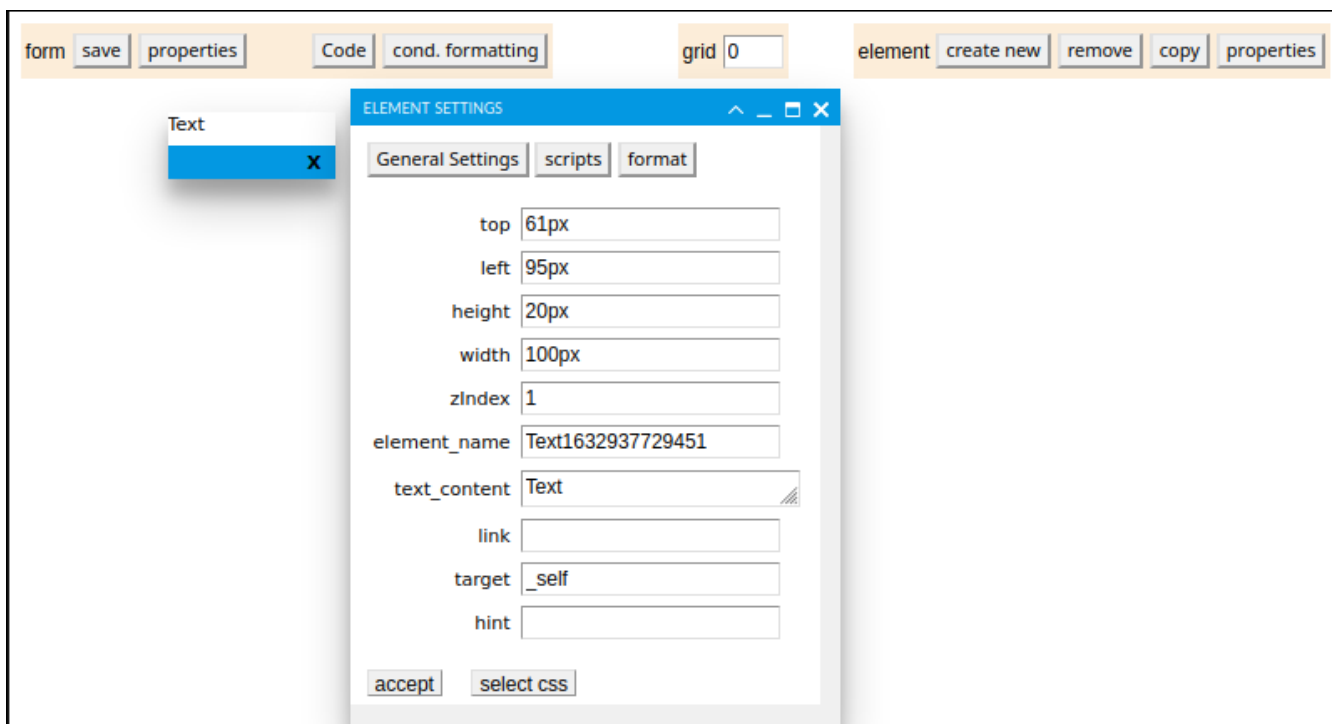


Fig 5: Element settings dialog

So that the dialog does not take up too much space and the overview is not lost, it is divided into three areas. The areas can be selected using the buttons at the top of the dialog. The general settings area is already preset. The Scripts area is only of interest if you want to execute your own code for a specified event. In the Format area you will find all the properties that have to do with the appearance of the element. General settings area: The properties that appear in the first area (general settings) depend on the element type. All elements have the properties top, left, height, width, zIndex, note and element name. The information on top and left always relates to the top left corner of the document and the element. You can either set the position and size of the element in this dialog, or move the element with the left mouse button pressed, e.g. change its size. The zIndex property determines the level in which the element is placed. The first letter z stands for the vertical axis. The lower the value in this field, the further down the element is placed. For example, if you want to place an element on a colored background, create a text element for the background, set the background color and leave the value for the zIndex at 1. Now create another element and place it over the text element. So that the new element actually appears above the text element, select a zIndex greater than 1 for this element. A text can be entered in the Note field, which is displayed to the user when he positions the mouse pointer over it. The property element name is automatically assigned by the unidb. It is made up of the element type and a Unix time stamp. If you want to use this field for a calculated field, a conditional formatting, or in a script, then you should give the field a more descriptive name. That makes life easier. Make sure, however, that you only use a name once.

Further fields depending on the element type:

Property	Element type (s)	description
Text content	text	The text to be displayed.
link	Text, display_fields, graphics	Address of a linked document or another place in the document.
target	Text, display_fields, graphics	Determines whether the target of the link should appear in the same frame or tab, or in a new tab or window.
locked	Text field, list field, text area, checkbox, tree view, option group, register, graphic, calculated	Locks the field from being edited by the user.
field	Text field, field_display, list field, text area, checkbox, option group, option	Specifies the database field.
Data format	Text field	Appearance of the field. The types text, number, date, time, slider and password are available. Text is the simplest form. Just try out what fits best.
min	Text field	Contains the smallest permitted value if the text field is set to the data type number or slider.
max	Text field	Contains the highest permitted value if the text field is set to the data type number or slider.
Increment	Text field	Specifies how far the value of the text field changes when a switch is pressed up or down (type number), or when the slider is moved minimally.
Link_Field	Subform	Database field in the main form, via which the link to the subform is established.
Link_Feld_Unterform	Subform	Database field in the subform that is used to link to the main form.
address	Subform	Link to the subform. You can get the link by right-clicking the subform in the tree structure and selecting the Copy link option in the context menu. The following must be entered after the link: UForm=1 .
List of values	List box, combo box	List of values that can be selected, each separated by a semicolon. If you fill this field, the next SQL field will not be taken into account. Example: Toyota; BMW; Renault
SQL	List box, combo box	SQL - SELECT, which can be used as an alternative to the value list to generate the selection options from a database table. Two fields are expected from the table. The first field stands for the value that should be given to this field and the second field is used for the text that is displayed when the selection is made. If the text available for selection is also to be the value to be saved in the field, please use the database field twice.

Lines	List box, combo box	The number of lines that should be visible at the same time. If there is a 1 here, the element looks like a normal text field, but with an arrow on the right-hand side with which the list can be expanded. If there is a value greater than 1, the element is displayed as a list.
SQL	Treeview	SQL - SELECT query. The tree is built from the result of the query.
headline	Treeview	Term that is displayed at the top of the tree structure. This is the root of the structure.
Index_field	Treeview	Provides a unique name for a node. These can be alphanumeric or numeric values.
Parent_field	Treeview	Index_field of the parent node.
text	Treeview	Text that is displayed in the tree structure for a node.
Link_field	Treeview	Database field from the query that contains the link that is assigned to the node.
Link_Target_Field	Treeview	Target for the link.
value	option	The value that is written to the database field of the option group to which this option belongs.
Tabs	register	Button with which tabs can be created, removed and sorted.
address	graphic	Link to the graphic to be displayed.
formula	calculated	Expression which should be calculated.

Scripts area:

Here you can enter the JS functions you have created yourself, which are to be executed in the event of the listed event. Please enter the function in this format: Function name () ;.

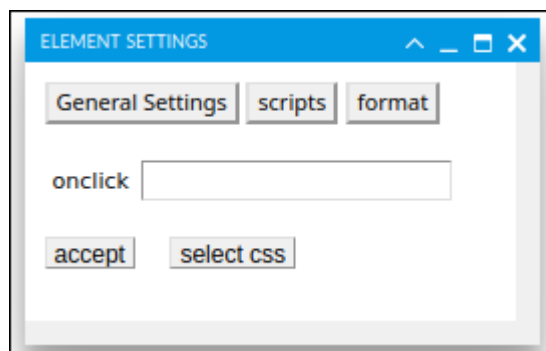


Fig 6: Scripts area

Format area:

The options in this area shouldn't need any description.

A note on the specification of the css_style:

A browser first uses the information from the integrated style sheets to format the elements. If there is still an area in the HTML document that defines these styles, this definition overwrites that of the stylesheets. The fields in the dialog shown below overwrite the other style definitions.

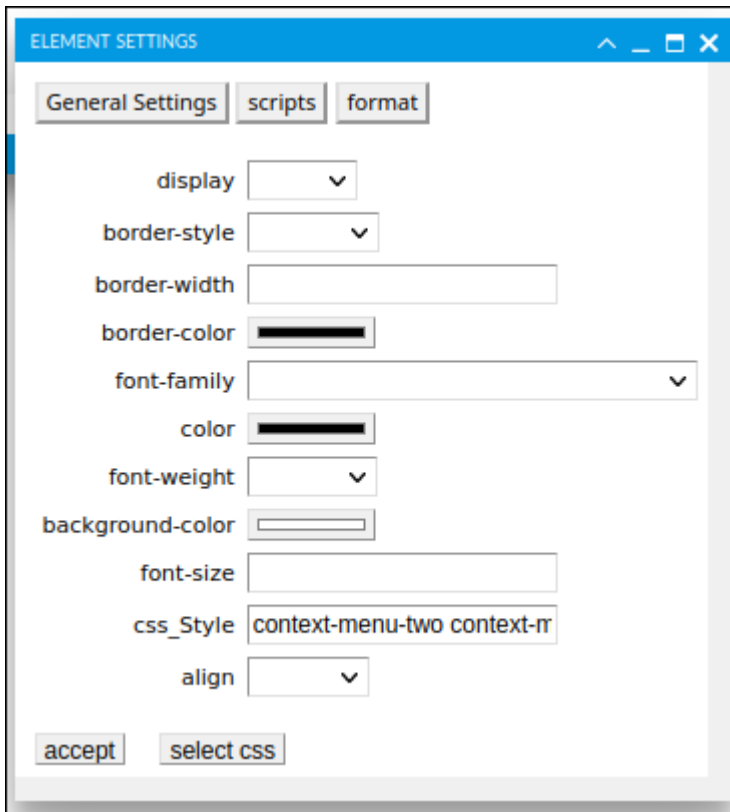


Fig 7: Format area

The button Select CSS opens a dialog via which one or more style definitions can be selected from the integrated style sheets. This saves having to set the same formatting for elements over and over again. You can integrate additional CSS stylesheets via the dialog for the settings of the document. These are then also displayed in the tree menu shown below.

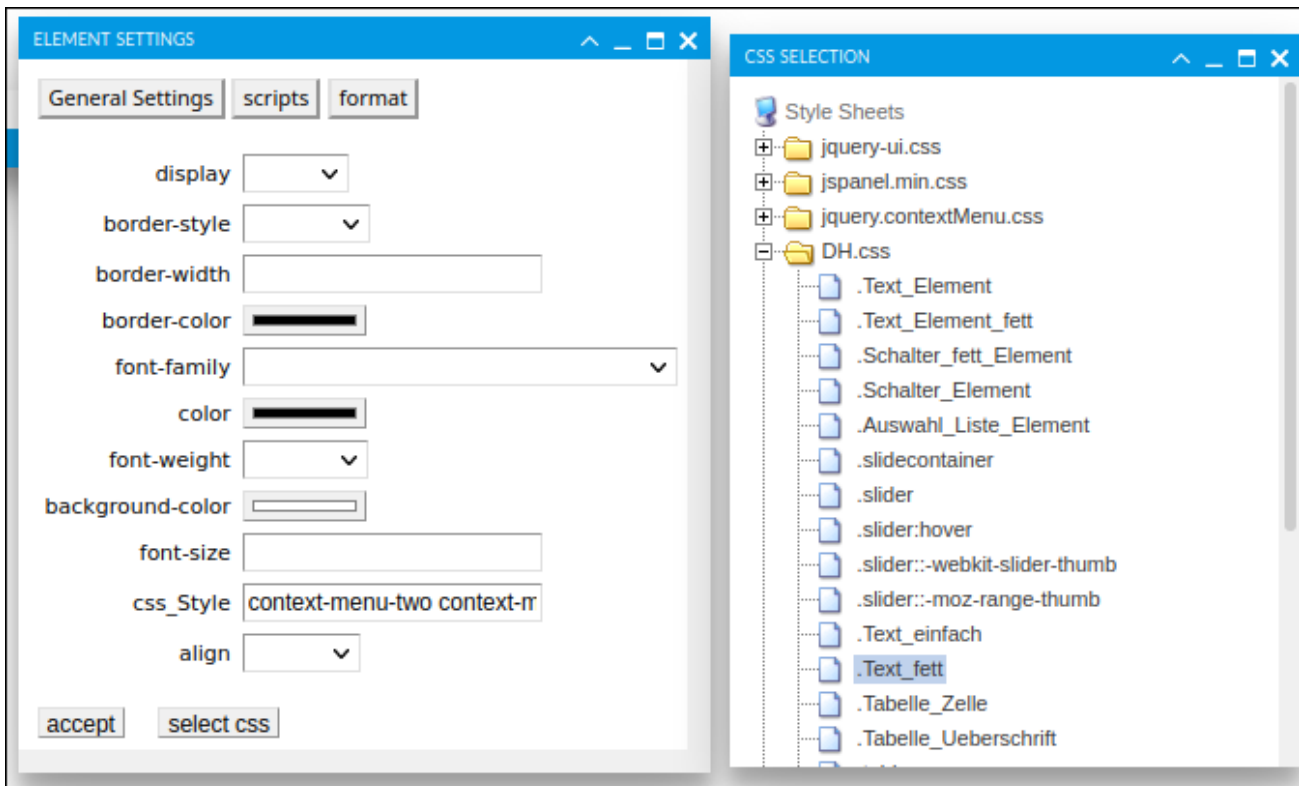


Fig 8: CSS selection

If you click on a branch in the tree structure, a small window appears showing the formatting.

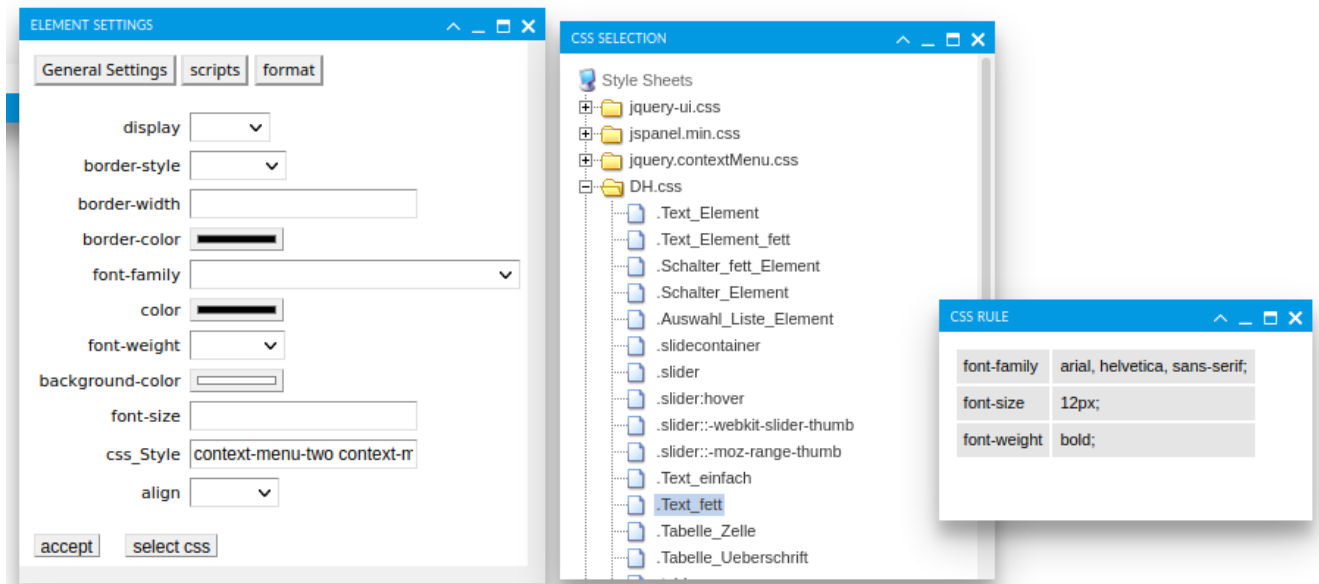


Fig 9: Select CSS rule

Positioning of elements:

You can position elements in three different ways.

Method 1: You enter the new position in the properties dialog of the element.

Method 2: You move the selected element while holding down the left mouse button.

Method 3: You move the unmarked element while holding down the left mouse button.

The difference between method 2 and 3 is that element (1) can be dragged onto another element (2) using method 3 and that from then on it belongs to element (2). If you now move element (2), both elements are moved. In the properties dialog of element (1) you can see that the properties above and on the left now refer to element (2).

Example:

You have created a register element. Now you want to place some text on the second tab. When you create the text element, it initially exists next to the tab element. Now left click on the text element and drag it directly to the tab. Please keep the left mouse button pressed, otherwise the text element would be marked and can therefore only be moved on the same level.

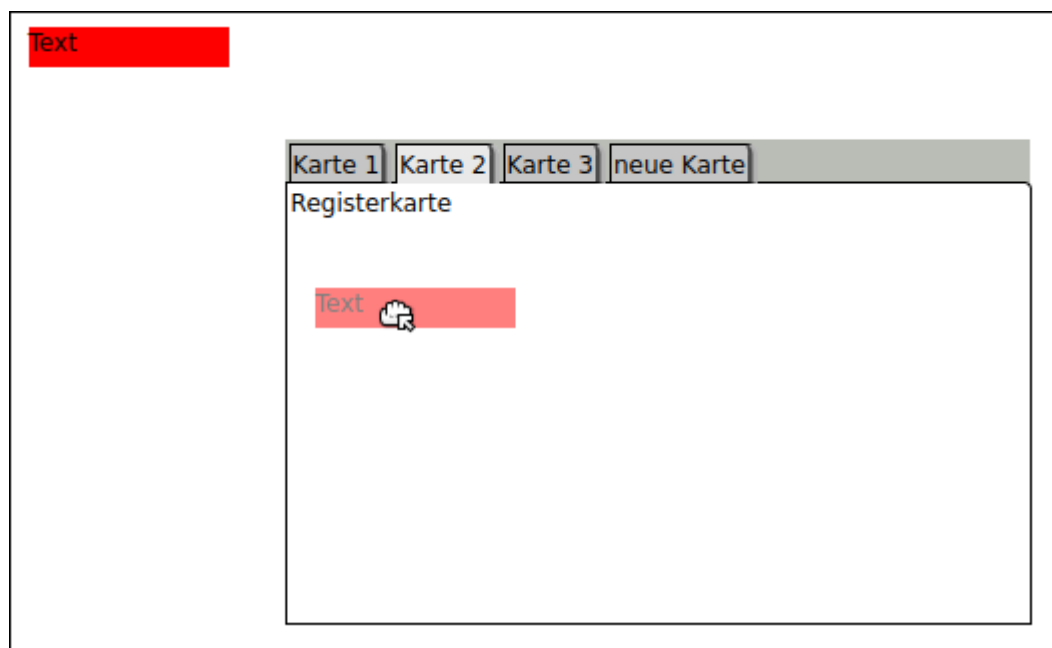


Fig. 10: Dragging the element with the mouse without selecting it

You mark several elements either by clicking on the elements while holding down the CTRL key, or by marking an area with a mouse click. To do this, click with the mouse on a corner of the area to be marked. The mouse pointer should now have the shape of a crosshair. Then click on the diagonal corner. With both methods, the marked elements are displayed pale. A double click on a free space in the document removes the marking again.

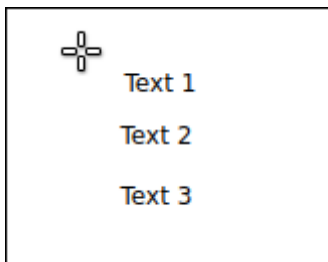


Fig 11: Start of the marking at the top left

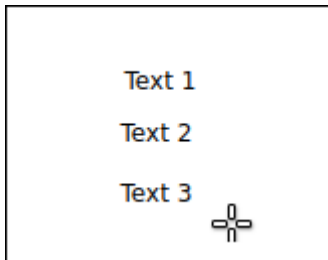


Fig. 12: End of the marking at the bottom right

A click with the right mouse button on one of the marked elements brings up a context menu via which the marked elements can be aligned or adjusted in size.

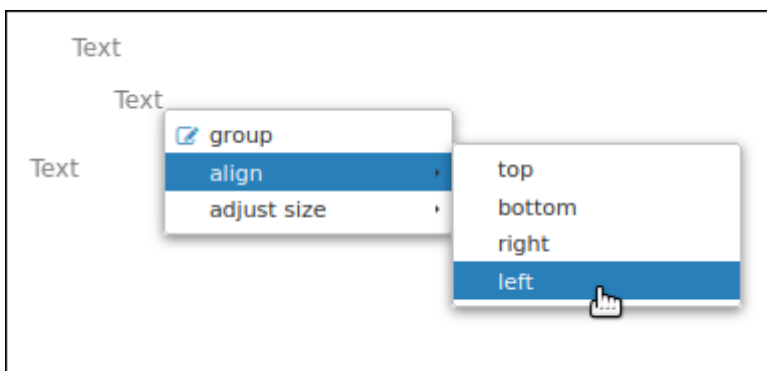


Fig. 13: Selected elements and context menu

The group option combines the selected elements in a group. This group can be moved like a single item.

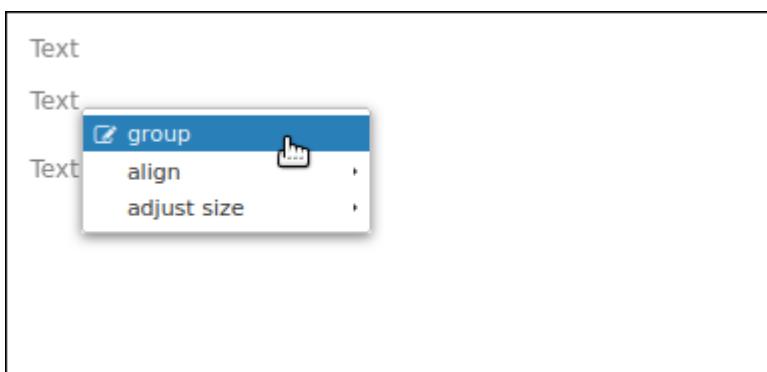


Fig. 14: Grouping of the aligned elements

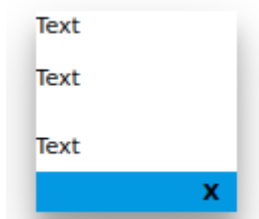


Fig 15: grouped elements

1.1.2.conditional formatting

In dialogue **cond. Formatting** you can specify any number of conditions that ensure that the formatting of an element changes as soon as the condition is met. Example: A field contains an amount. As soon as the amount is negative, it should appear in red. If it is positive, the original formatting (e.g. black) is retained.

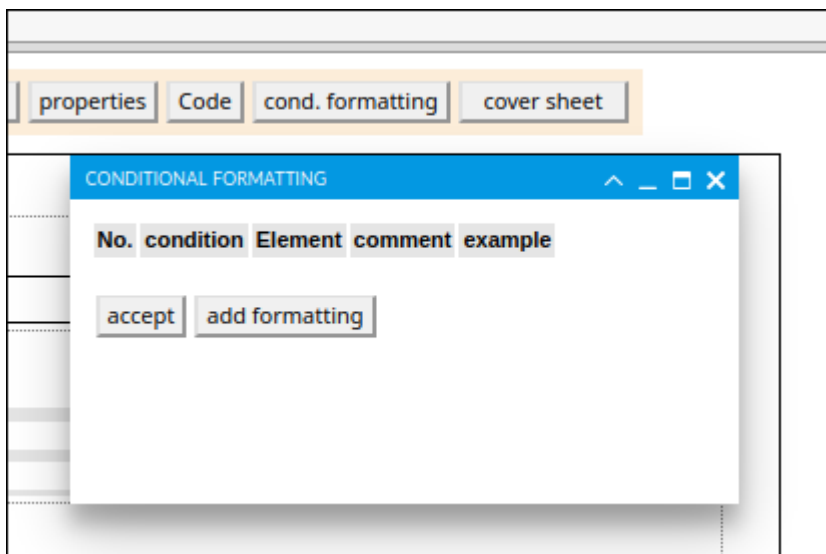


Fig. 16: Overview of conditional formatting that has already been created

Via the **cond. Formatting** in the header open the dialog. Initially it does not contain any entries. The **add formatting** button opens another dialog in which you can define the condition and the formatting.

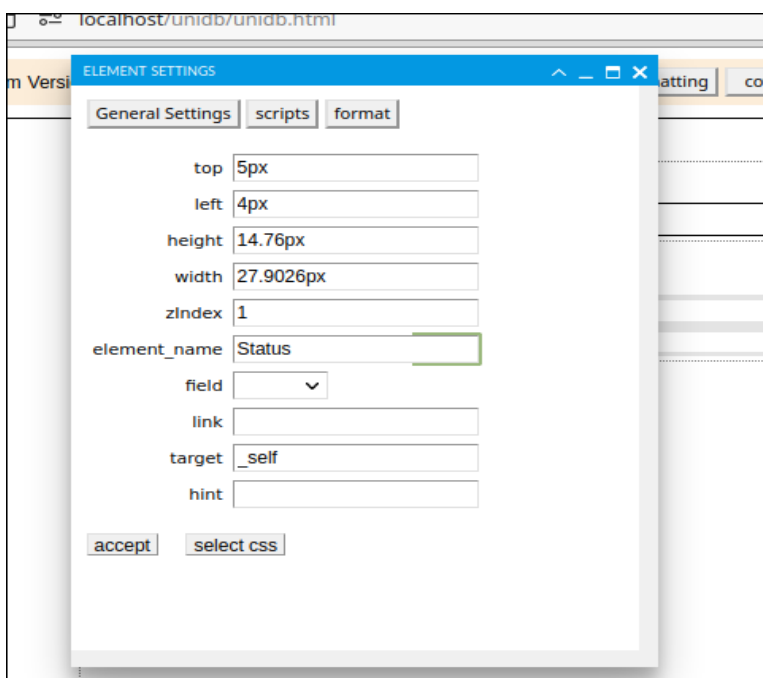


Fig. 17: Dialog for creating a new conditional formatting

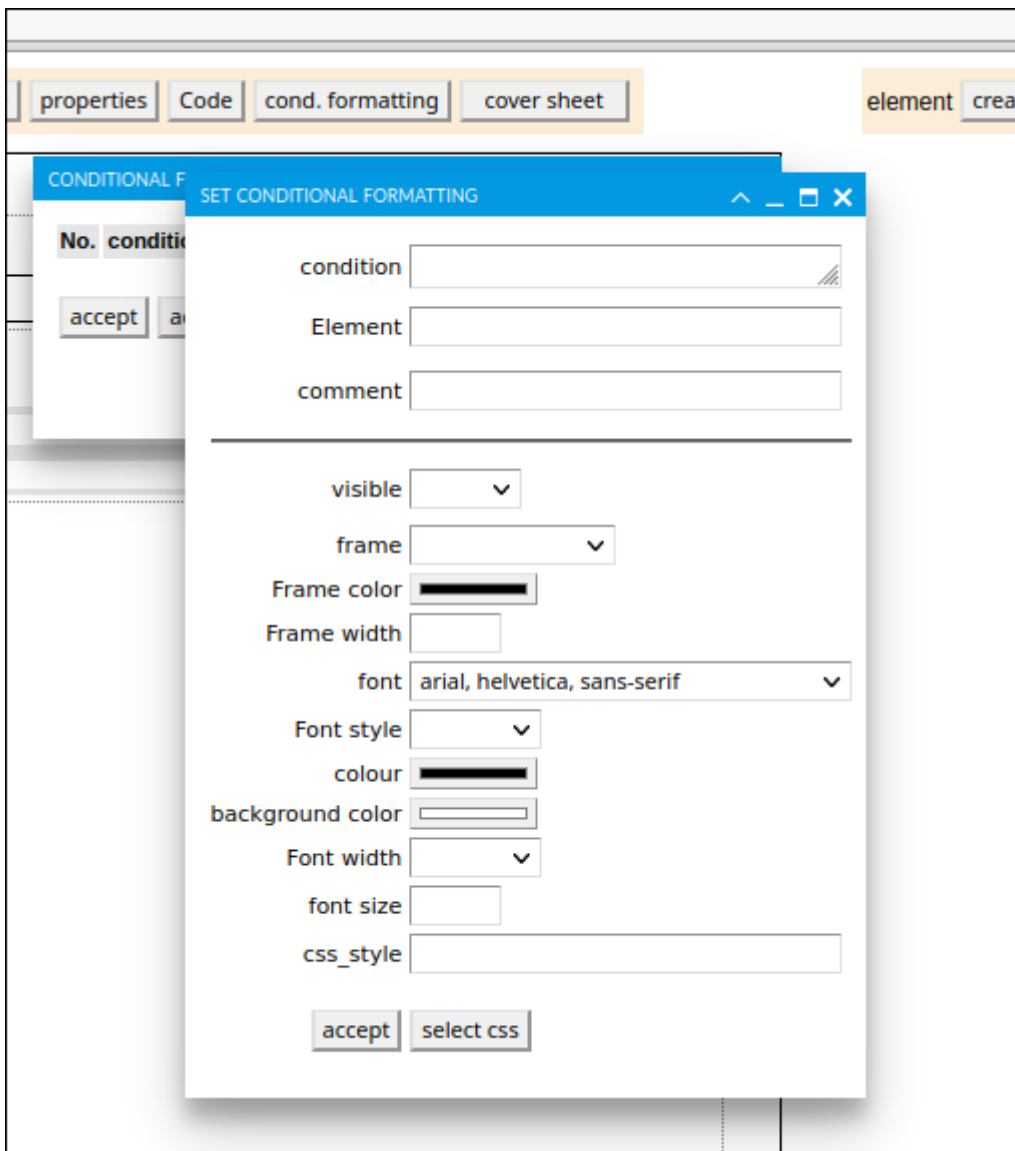


Fig. 18: Element name changed to "Status"

The top three fields contain the condition that leads to the formatting change, the field to be formatted and optionally a comment.

All of the following fields are used to set the desired formatting.

Before you edit this dialog, however, it is advisable to give the elements concerned meaningful names. When you create a new element, a name is automatically assigned, which consists of the element type and the current Unix time stamp. Of course, such a name says less than a self-assigned name. In the next picture you can see the dialog for the element properties. There the automatically generated name has been replaced by the name Status. This is more convenient to work with than with a name such as field_display1629109057579.

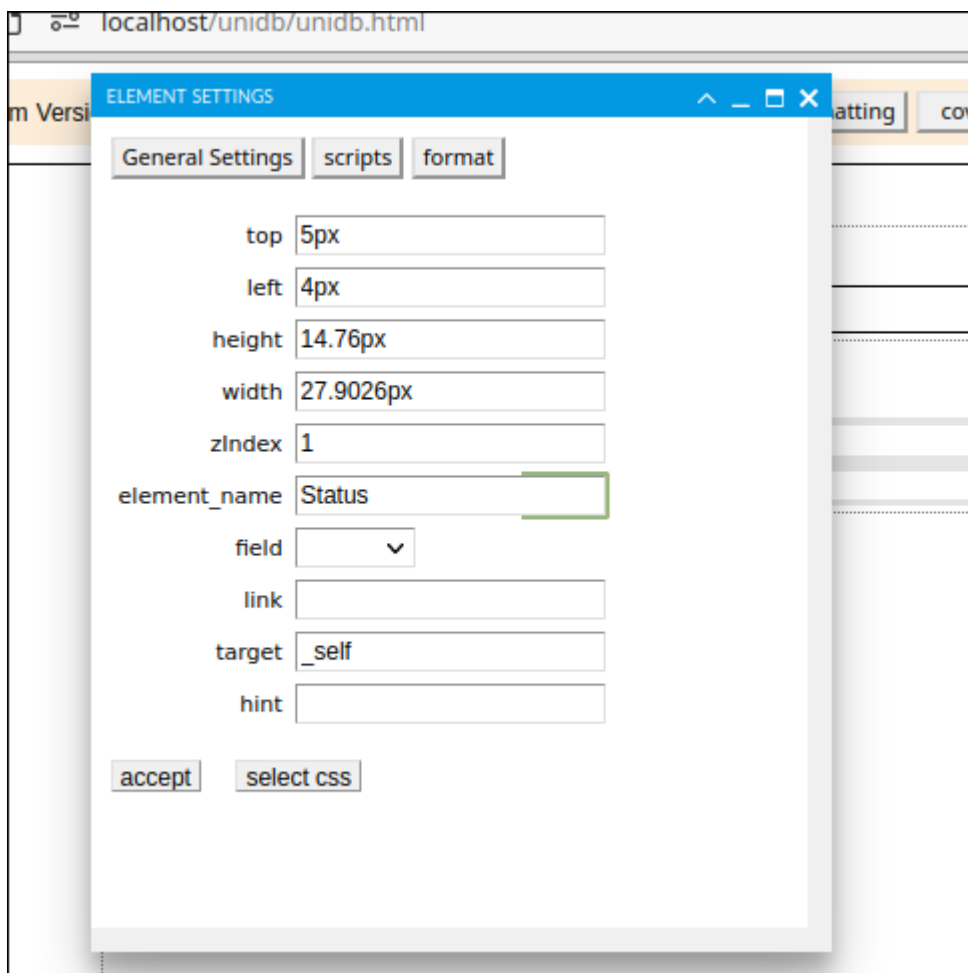


Fig. 19: Completed dialog for conditional formatting

The content of the **Condition** field should be self-explanatory. The element whose formatting is to be changed is in the **Element** field. In the area for formatting only the background color was set to red.

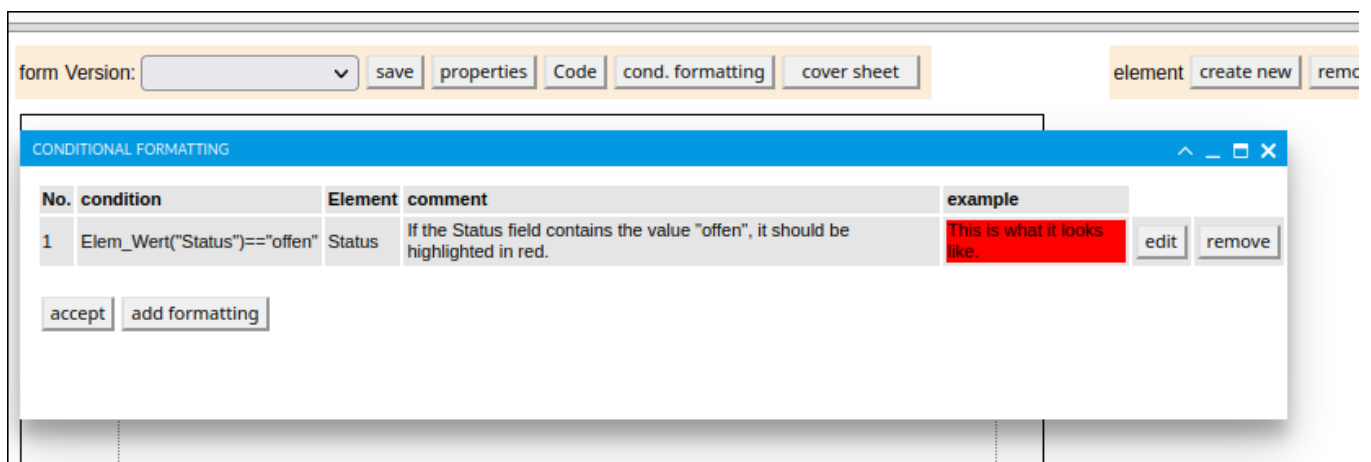


Fig. 20: Finished conditional formatting in the overview window

We press the button **accept** and see the result of our efforts in the first dialogue. The **example** column shows how the field is displayed when the condition is met.



Fig. 21: Conditional formatting in action

The example was created for a report. In the picture below you can see the result. In the first record, the status "deferred" is displayed as defined in the draft report. In the second data record, the Status field has the value "open" and is highlighted in red as required.

1.1.3. own JS code

You can integrate your own JS functions into some document types. Write your functions in the dialog that you can access in the header using the Code button. The dialog only consists of a simple text area and a button to take over your functions.

You can enter your functions in the dialog for the element properties for the desired event.

Please do not forget to click the Apply button at the bottom of the dialog before you close the dialog.

Enter the JS code to be executed after the document has been loaded in the dialog for the document properties.

You can find out more about this in Chapter 3.

1.2. Edit documents

The document types query, form and report are of little interest to ordinary users. The creation / editing is also not an everyday task for these users. Therefore, the draft mode of these document types is not discussed in the User's Guide, but here.

1.2.1. query

Note:

To use the Query document type, you should be familiar with querying databases. A course on database queries is beyond the scope of this help.

However, anyone who has already worked with graphical query editors will find their way around here immediately.

Menu bar:

The Document menu looks very similar to the Document menu found in the other document types we'll discuss here. The Query menu, on the other hand, has been specially designed for this document.

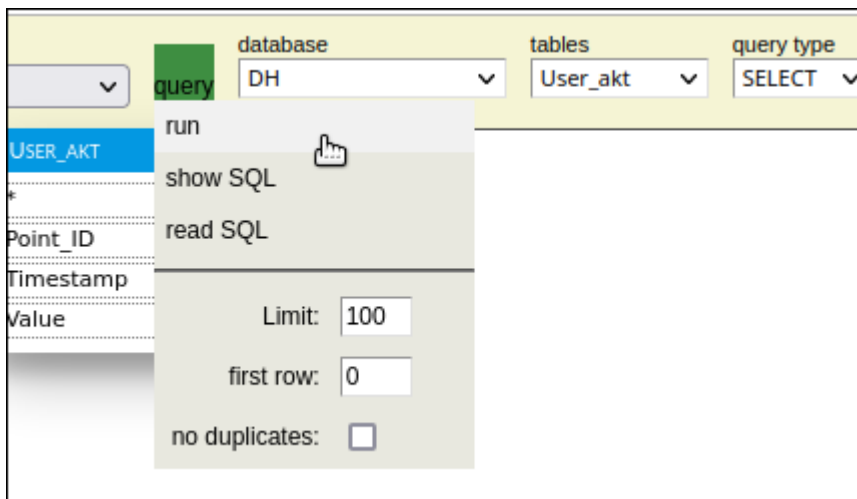


Fig 22: Query menu

- **Run** at the top entry should be clear.
- **Show SQL** opens a window that contains the SQL code that was created from the tables displayed and the settings in the selection area.
- **Read SQL** works the other way around. Enter your own SQL code in the window with the SQL code, or if you have changed the displayed SQL code, this code is read in and the tables are displayed accordingly. The selection area is filled in accordingly.
- **Limit** is set to 1000 by default. This is a caution in the event that you are querying tables with several million records. If you do not want any limitation, simply empty the field.
- **first row** shows the data records from the specified data record onwards. If there is a 1000 here, then the data records 1000 to 2000 are displayed.
- **No duplicates** places a DISTINCT after the SELECT at the beginning of the query.

As soon as an entry is selected in the **tables** field, a new window is created which represents the selected table. The fields of the table are listed one below the other in the window and can be dragged with the mouse into the selection area or into another table window.

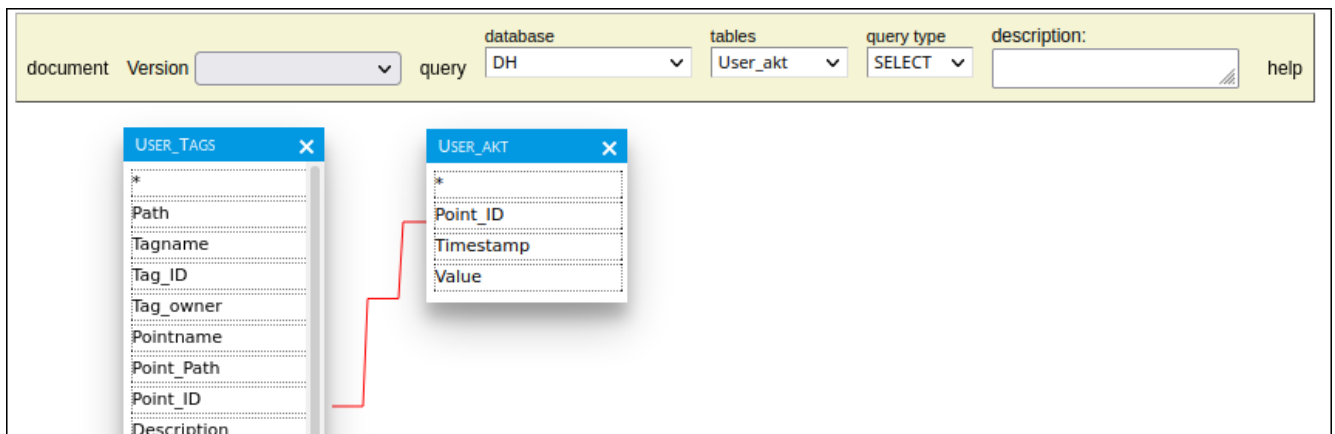


Fig. 23: two tables with an INNER JOIN connection

In the figure above, the Point_ID field was dragged from the Tags table to the field of the same name in the Archive table. The two tables are thus linked to one another (INNER JOIN). To change the connection type, simply right-click the connection and select Properties from the context menu. In the dialog you can then change the connection type to RIGHT JOIN or LEFT JOIN.

Selection area:

The handling of this area should not really need any explanation.

All criteria that are in the same line are logically AND linked. All lines are logically ORed with one another.

Either drag the required fields with the mouse from the tables into the top line of the selection area, or you first select the table in the selection area and then the field from the list boxes.

Each column contains a switch + and -. The switch - removes the column and the switch + inserts a new empty column to the right of the current column.

Here is a small example:

We want to see the raw values for day T25. The values should be before 01/01/2020 and be sorted in descending order by time stamp. We do not need the tag name and the value type (raw value, rV) in the results table. We therefore remove the check mark in these two columns.

SELECTION AREA							
	-	+	-	+	-	+	-
field	Point_Path	Tagname	Point_ID	Timestamp	Value		
function							
alias							
table	User_Tags	User_Tags	User_akt	User_akt	User_akt		
sort							
group	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
show	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
criteria		like 'T25'		>'2020-01-01'			
criteria							

Fig 24: Selection area

Don't worry, you can use as many lines as you want to specify the criteria. As soon as the program notices that there is only one empty line left, it creates a new line. The same applies to the columns.

Via the **Query / Show SQL** menu, you get the SQL code for your selection:

```
SQL
SELECT `Archiv`.`Timestamp`, `Archiv`.`Value` FROM `Tags` INNER JOIN `Archiv` ON `Tags`.`Point_ID` = `Archiv`.`Point_ID` WHERE (`Tags`.`Tagname` = "T25" AND `Archiv`.`Timestamp` < '2020-01-01' AND `Archiv`.`Art` = "rV") ORDER BY `Archiv`.`Timestamp` DESC LIMIT 0,1000;
```

Fig 25: SQL Code

The **query / execute** menu provides the query result:

QUERY RESULT				
Point_Path	Tagname	Point_ID	Timestamp	Value
/	T25	250	2021-10-01 12:20:29	12.875
/	T25	250	2021-10-01 12:20:29	12.875
/	T25	250	2021-10-01 12:27:07	13.125
/	T25	250	2021-10-01 12:27:07	13.125

Fig. 26: Result of the query

Please note:

When specifying a criterion, always include the operator. Please always put alphanumeric values in double quotation marks.

And now one last note about the queries:

Please do not forget to save the document via the Document menu before calling up another document.

1.2.2. Form

1.2.2.1. Form properties

After you have created a new form, you should first look at the form properties. To do this, we call up the dialog for the form settings in the header.

The top two fields in this dialog should be understandable.

The standard view is set in the third line. The form therefore always appears in this view when it is opened. The **Rows per Page** parameter determines how many rows should be displayed in the table view at the same time. The last parameter in this line can be used to specify whether or not the navigation area including the switches for a new data record and for deleting a data record should be displayed. you can thus build forms that do not display data records and instead serve as a menu or something similar.

If you would like to make the database available on all servers of the collective, please mark the **Synchronization** field. This means that all changes to a data record are transferred to all servers in the collective.

The **data source** is a SELECT query that supplies the data records for the form. If you are unsure about SQL, you can also create a query using the **Query** document type. There you can display the generated SQL text via the menu and insert it here via the clipboard.

The field **additional header lines** is intended for the integration of further JS libraries or CSS files.

JS code is entered in the **JS onload** field, which is to be executed as soon as the form has been loaded. The current field can be written with JS code, which is called as soon as a record has been loaded.

FORM SETTINGS

form name: Pflege

background color: [color picker]

presentation: form lines per page: 15 Show navigation area: ☒

database: unldb

synchronization: ☐

data source: `SELECT * FROM `Pflege` ORDER BY `Pflege_ID` ASC;`

additional header lines: [text area]

Js - onload: [text area]

current: [text area]

changes: accept

Fig. 27: Form settings dialog

1.2.2.2. Data Source

If you want to edit data with the form, the data source must be a SELECT query, which is based on only one table. The table must also contain an index field. If you do not include this index field in the query, the unldb will try to add it. A query based on several tables can be used, but it is not possible to edit the data. In this case, please use a subform instead.

1.2.2.3. Subform

A subform is inserted as an element. This element has the properties **address**, **link_field** and **link_field_subform**. Because a subform is a separate form, it is inserted into the main form as an iFrame. If possible, a relative link to the subform is entered in the Address field. Simply copy the address by right-clicking on the subform in the tree structure and then selecting the option Copy link - Copy address. Depending on the browser used, this option can be named a little differently. You then enter the address copied in this way in the dialog for the settings of the element in the Address field. You may then have to make a relative link out of the link. This link then looks something like this:

./Formular.php?Baum_ID=xxx&Server_ID=y&UForm=1

Instead of xxx and y, however, there are numbers. Now you have to add the part marked in bold. The parameter UForm = 1 tells the unidb that the form will be called up as a subform.

Since the content of the sub-form depends on the data record of the main form, a relationship must now be established between the two forms. This is done via the two fields Verkn_Feld and Verkn_Feld_Unterform. The field contents of these two fields will always be the same in display mode. It becomes clearer with an example:

Let's say we have a customer database. This database consists of a table called Masterdata, an Invoices table and another invoice_items table.

We would now like to create a form that shows us the customers and the respective invoices. If necessary, we would also like to edit the data records.

The database was only created for this example and is therefore very simple.

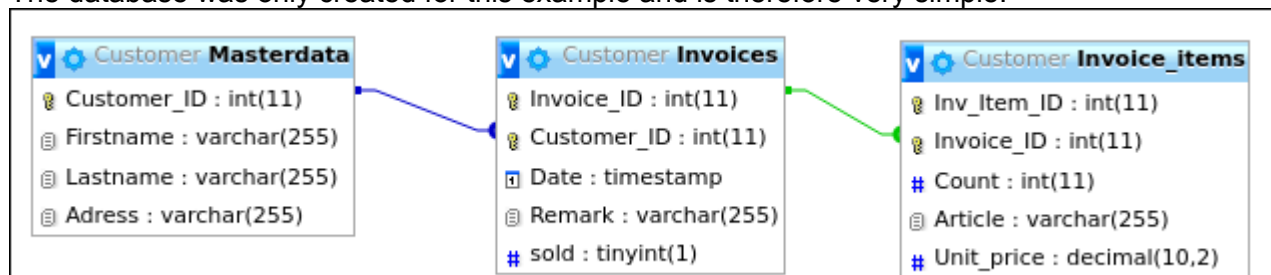


Fig 28: Customer database

We will initially ignore the invoice items. The invoices should appear in a subform. The main form will only show the masterdata.

The first form will be the Invoices subform. In the form settings, we specify Table as the standard view, as we only want to see the invoices in the subform as a list. The latest invoices should always be at the top of the list.

FORM SETTINGS

form name: Pflege

background color: [color picker]

presentation: form lines per page: 15 Show navigation area: ☒

database: unidb

synchronization: ☐

data source: `SELECT * FROM `Rechnungen` ORDER BY `Datum` DESC;`

additional header lines: [text area]

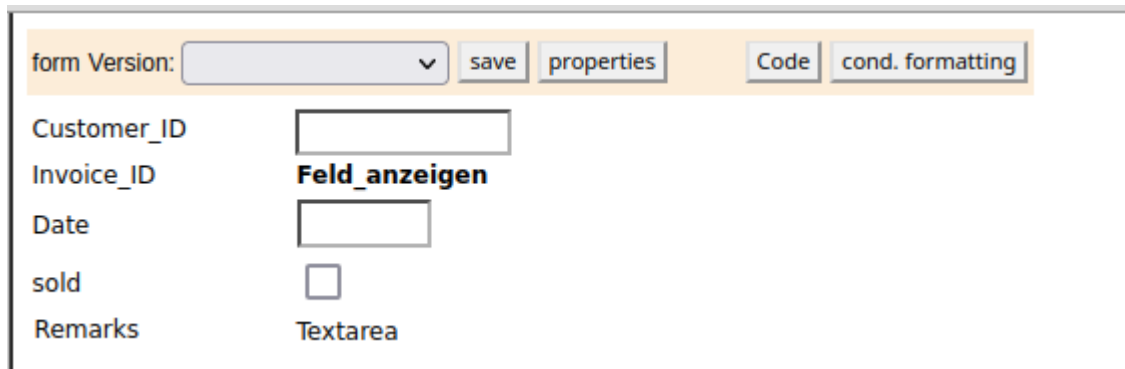
Js - onload: [text area]

current: [text area]

changes: accept

Fig. 29: Form settings for the invoices form

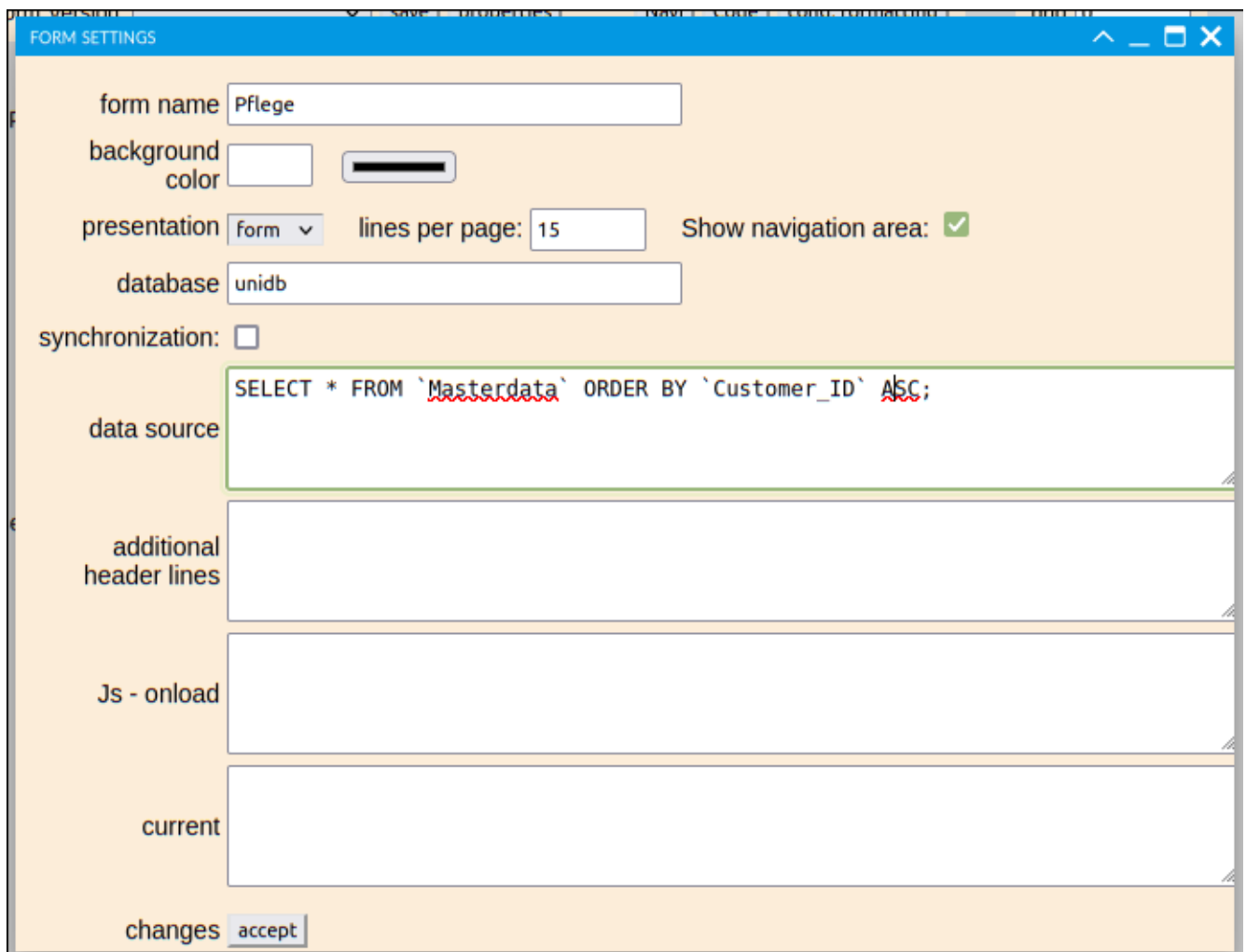
Since we also want to be able to edit the data, we need to create a form view as well.



The image shows a draft view of a form for invoices. At the top, there is a toolbar with buttons for 'form Version:', 'save', 'properties', 'Code', and 'cond. formatting'. Below the toolbar, the form fields are listed on the left and their corresponding input types on the right: 'Customer_ID' (text box), 'Invoice_ID' (text box with the label 'Feld_anzeigen' next to it), 'Date' (text box), 'sold' (checkbox), and 'Remarks' (Textarea).

Fig. 30: Draft view of the invoices form

Now we still need the main form.

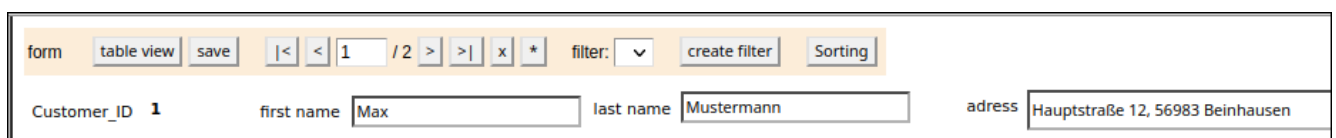


The image shows the 'FORM SETTINGS' dialog box. It contains several configuration options: 'form name' (Pflege), 'background color' (a color picker), 'presentation' (form), 'lines per page' (15), 'Show navigation area' (checked), 'database' (unidb), 'synchronization' (unchecked), 'data source' (SELECT * FROM `Masterdata` ORDER BY `Customer_ID` ASC;), 'additional header lines' (empty text area), 'Js - onload' (empty text area), 'current' (empty text area), and 'changes' (accept button).

Fig. 31: Form properties of the main form

Here we select form as the representation. The customers will be sorted in ascending order according to their customer number.

After you have made the settings for the form, you should press the save button once. The new settings are adopted and the database fields are available for selection when the elements are created.



The image shows the main form view. At the top, there is a toolbar with buttons for 'form', 'table view', 'save', navigation buttons, 'filter:', 'create filter', and 'Sorting'. Below the toolbar, the form fields are displayed: 'Customer_ID' (1), 'first name' (Max), 'last name' (Mustermann), and 'adress' (Hauptstraße 12, 56983 Beinhausen).

Fig 32: Main form

Now we add the subform for the invoices. We set the element so large that you can see 15 lines of the

table and the header without having to scroll.

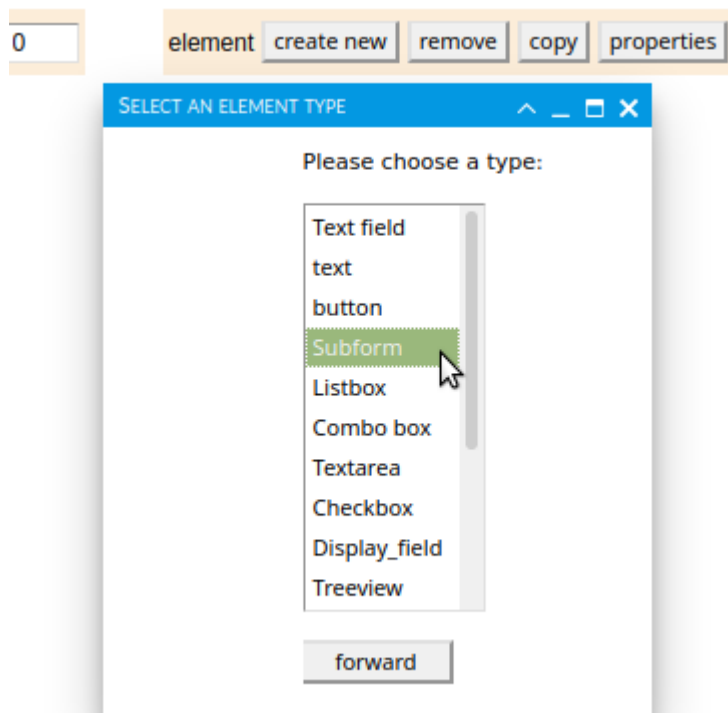


Fig. 33: Create a new element of the type subform

Caution, stumbling block!

Since the element is an iFrame, the selection of the element is a bit treacherous. It can only be selected by clicking on the edge of the element. A little sensitivity is required for this. So that at least the context menu can be easily reached, the edge of the selected element is displayed extremely wide. A right click on this border then opens the context menu. We are already working on a solution to this inconvenience.

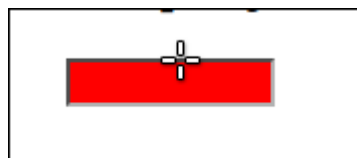


Fig. 34: The element can only be selected from the edge.

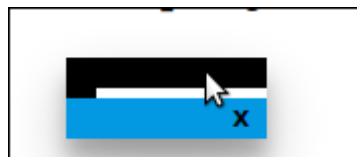


Fig. 35: Selected element with a temporarily wide border

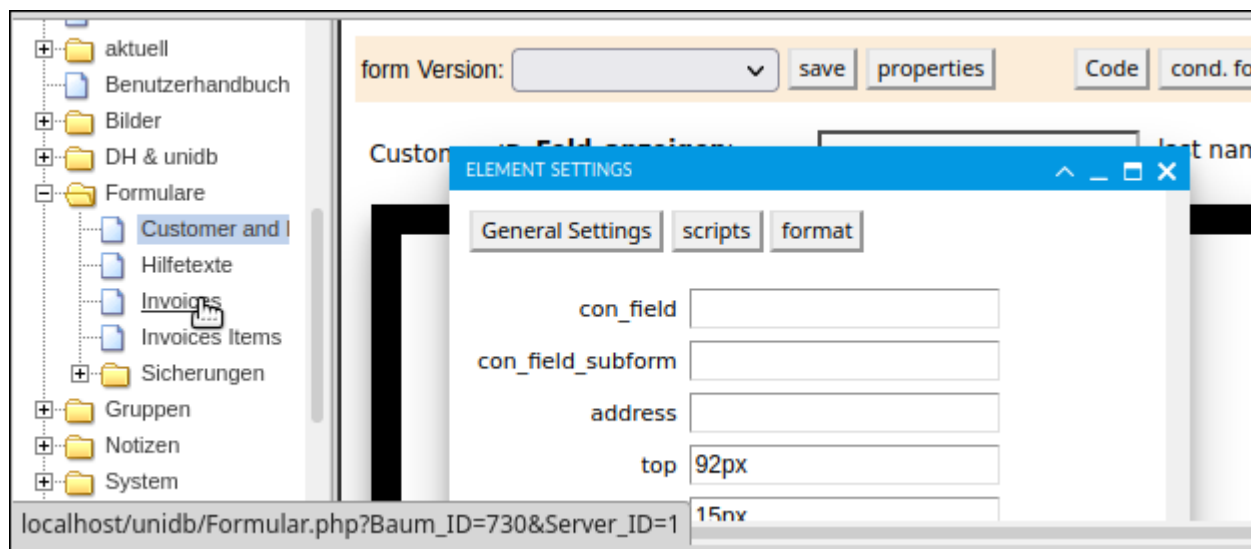


Fig. 36: Element settings for the subform

In Figure 36, the mouse pointer is in the tree structure directly over the Invoices form. At the bottom left of the picture we can see the associated URL `http://localhost/unidb/Formular.php? Baum_ID = 730 & Server_ID = 1`.

We copy the URL via the clipboard into the Address field of the dialog. There we turn the absolute link into a relative link: `./Formular.php?Baum_ID=730&Server_ID=1`
 Now we just have to add a little something to the link so that the unidb also knows that this is a subform: `./Formular.php?Baum_ID=730&Server_ID=1&UForm=1`.
 The common field in both forms is called `Customer_ID`. The completed dialog for the properties of the element now looks like this:

Fig. 37: Element properties of the subform

As soon as we apply the settings, the subform appears in the element. Since there are no invoices yet, it is of course still empty.

Fig. 38: Customers form with invoices subform in the design view

Invoice_ID	Customer_ID	Date	Remark	sold
17	1	2021-10-01 00:00:00		0

Fig. 39: Customer form with subform in the user view.

It's all well and good, but we would also like to see the invoice items. To do this, we simply create another form and then incorporate this into the invoice form as a sub-form. This gives us a cascade that

consists of three nested forms.

FORM SETTINGS

form name: Pflege

background color: [color picker] [black bar]

presentation: form lines per page: 15 Show navigation area: ☒

database: unidb

synchronization: ☐

data source: `SELECT * FROM `invoice_items` ORDER BY `Inv_Item_ID` ASC;`

additional header lines: [text area]

Js - onload: [text area]

current: [text area]

changes: accept

Fig. 40: Form properties of the invoice items form

In the form view, we have also added a calculated field that multiplies the number of items by the individual price.

form table view save |< < 1 / 8 > >| x * filter: [dropdown] create filter Sorting

Inv_Item_ID 1

Invoice_ID 15

Count: 13

Article: Schrauben M20 X 60

Unit price: 2.34

Summary: 30.42

Fig. 41: User view of the invoice items form

Now all we have to do is add it to the Invoices form as a subform.

The common data field of the two forms is called Invoice_ID and is entered in the corresponding fields of the element properties.

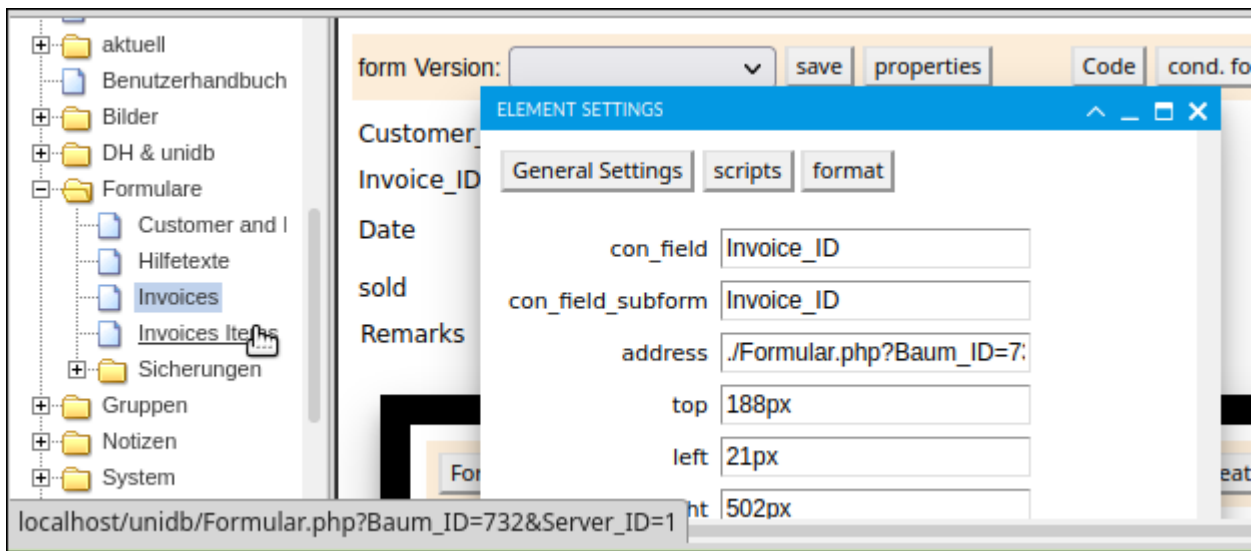


Fig 42: Settings for the new subform

We build the address from the URL of the invoice items form and the addition & UForm = 1. Here we turn the absolute link into a relative link. We then enter the result `./Formular.php?Baum_ID=732&Server_ID=1&UForm=1` as the address. The new subform appears immediately after we have applied the settings.

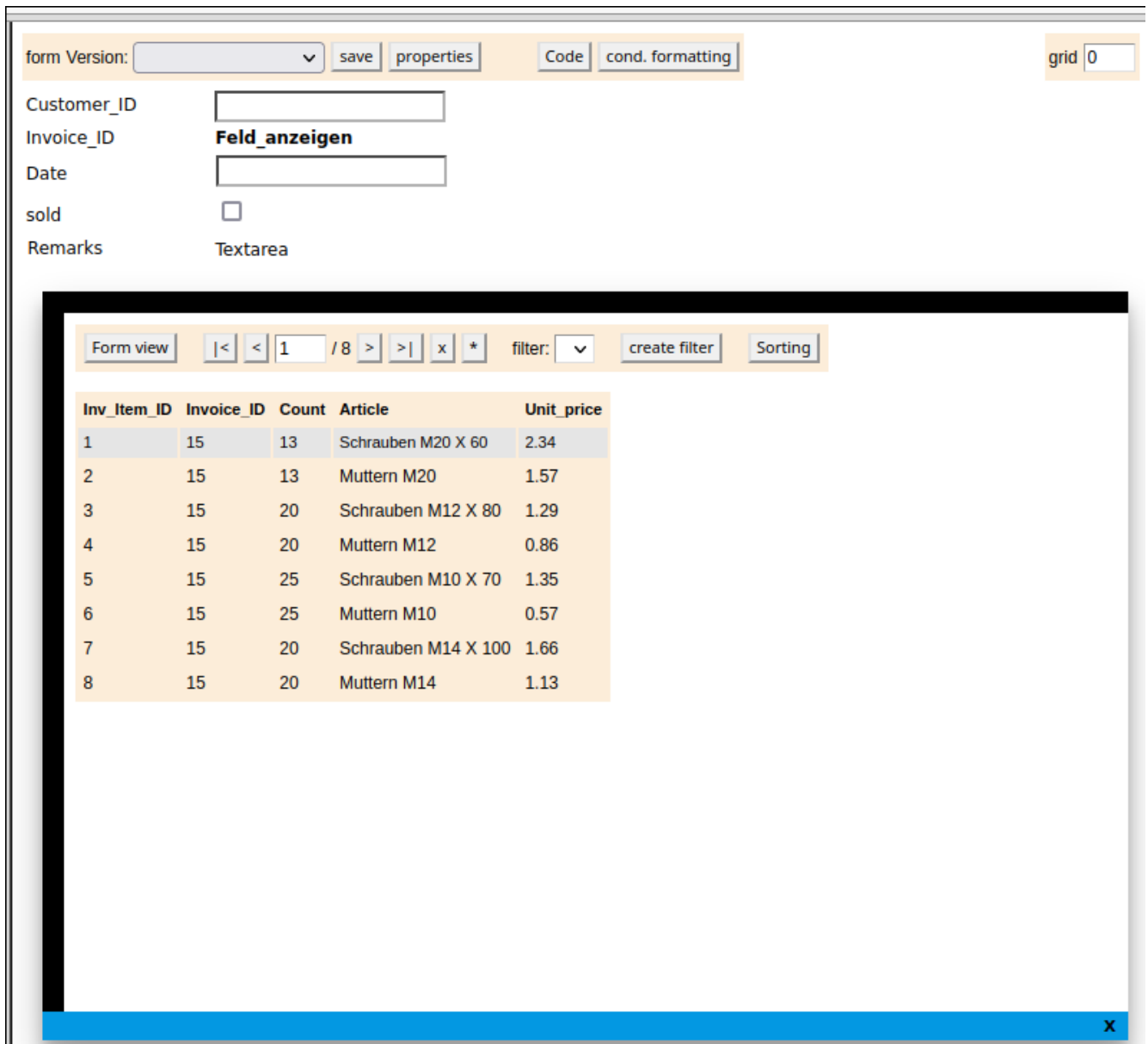


Fig. 43: The new Invoice Items subform in the Invoices form.

Please do not forget to save the changes. Wouldn't it be a shame about the work?

A German proverb says: "Order is half life!". It follows that we spend the other half looking for order. ;-)
Joking aside, good organization saves work. So it is time to organize the forms hierarchically.

This

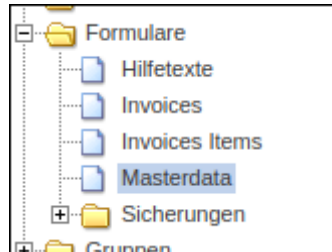


Fig 44: Mess

will change to this

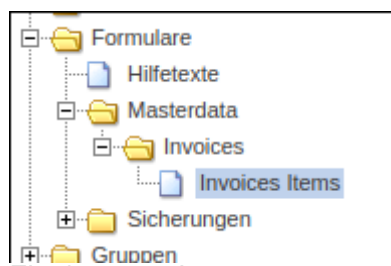


Fig 45: even better

The hierarchy is now clearly shown. If we now give the master data form a reasonable name, then we can actually be satisfied.

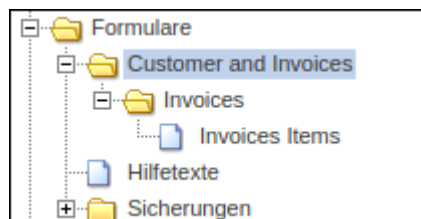
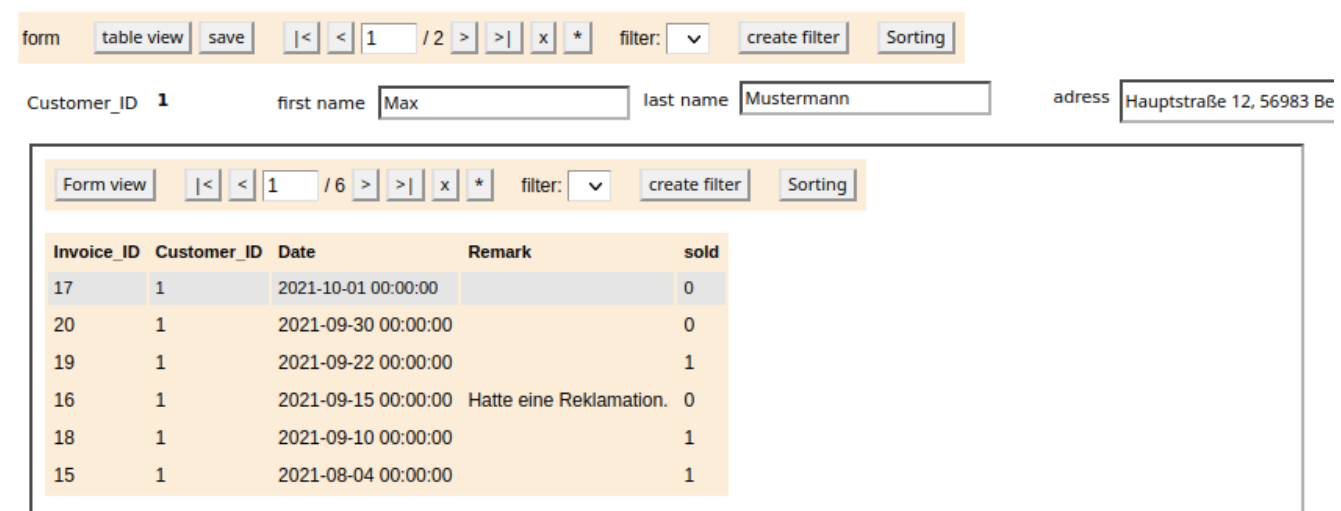


Fig. 46: Masterdata became Customers and Invoices

And now the views of the users:



Invoice_ID	Customer_ID	Date	Remark	sold
17	1	2021-10-01 00:00:00		0
20	1	2021-09-30 00:00:00		0
19	1	2021-09-22 00:00:00		1
16	1	2021-09-15 00:00:00	Hatte eine Reklamation.	0
18	1	2021-09-10 00:00:00		1
15	1	2021-08-04 00:00:00		1

Fig. 47: Customers and invoices form with the invoices subform

If we now double-click on an invoice, such as the Invoice_ID 15, we see the Invoices subform in the form view with the record of the Invoice_ID 15.

The screenshot shows a form cascade. The top form is for a customer, with fields for Customer_ID (1), first name (Max), last name (Mustermann), and address (Hauptstraße 12, 56983 Beinhausen). Below this is a form for an invoice, with fields for Invoice_ID (15), Date (2021-08-04 00:00:00), and a checkbox for 'sold' (checked). The bottom form is a table view of invoice items for invoice 15.

Inv_Item_ID	Invoice_ID	Count	Article	Unit_price
1	15	13	Schrauben M20 X 60	2.34
2	15	13	Muttern M20	1.57
3	15	20	Schrauben M12 X 80	1.29

Fig. 48: View of the entire form cascade

So here we see invoice no. 15 that was sent to customer Max Mustermann. We also see all invoice items for this invoice.

1.2.2.4. Register element

We create a new element of the register type and drag it to the desired size with the mouse. Your work will be made easier if you set a background color for the register that clearly stands out from the form background. When you have finished creating the form, you can reset the background color.

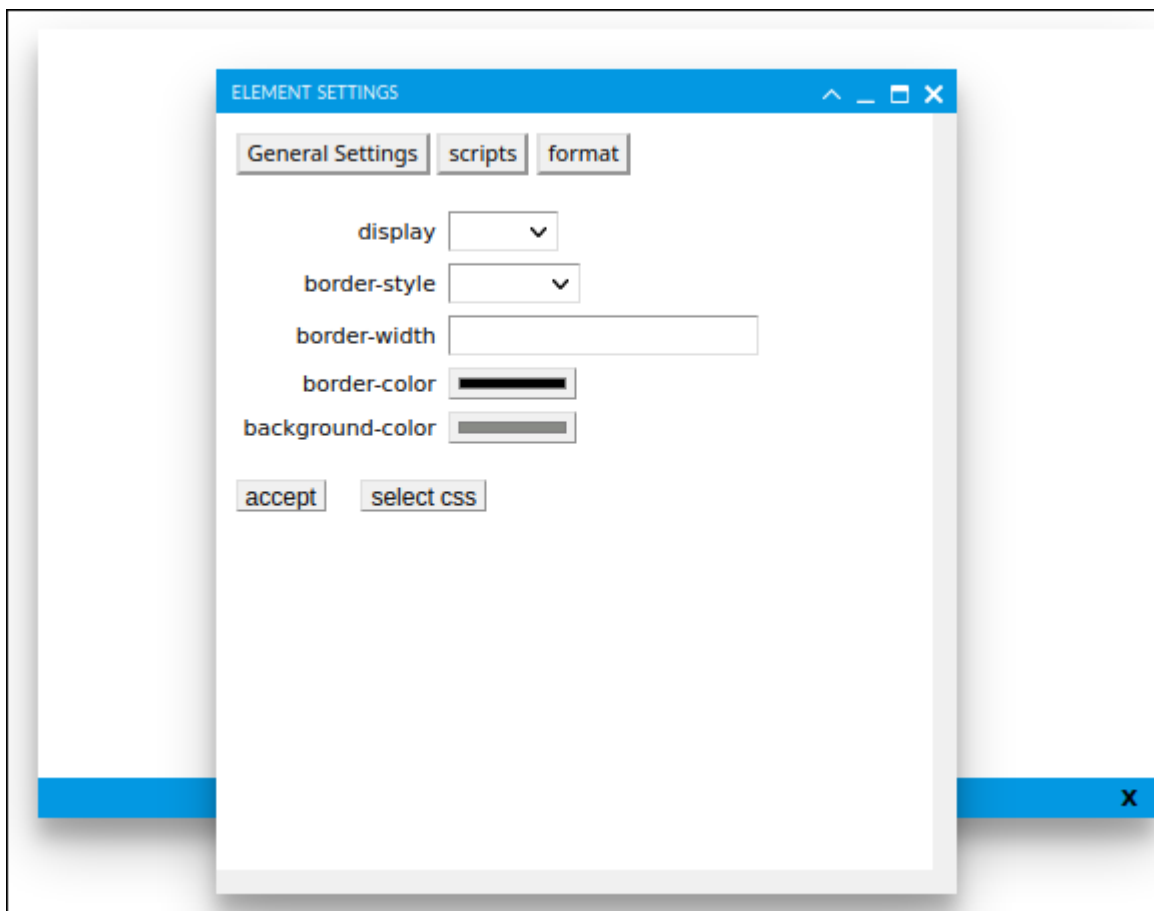


Fig. 49: Setting the background color of the register.

The new tab does not yet have tabs. We change this by pressing the **Edit tabs** button in the element properties dialog under **general settings**. Another dialog opens. This dialog initially only contains a button labeled **New tab**.

If you click this button you should see the following:

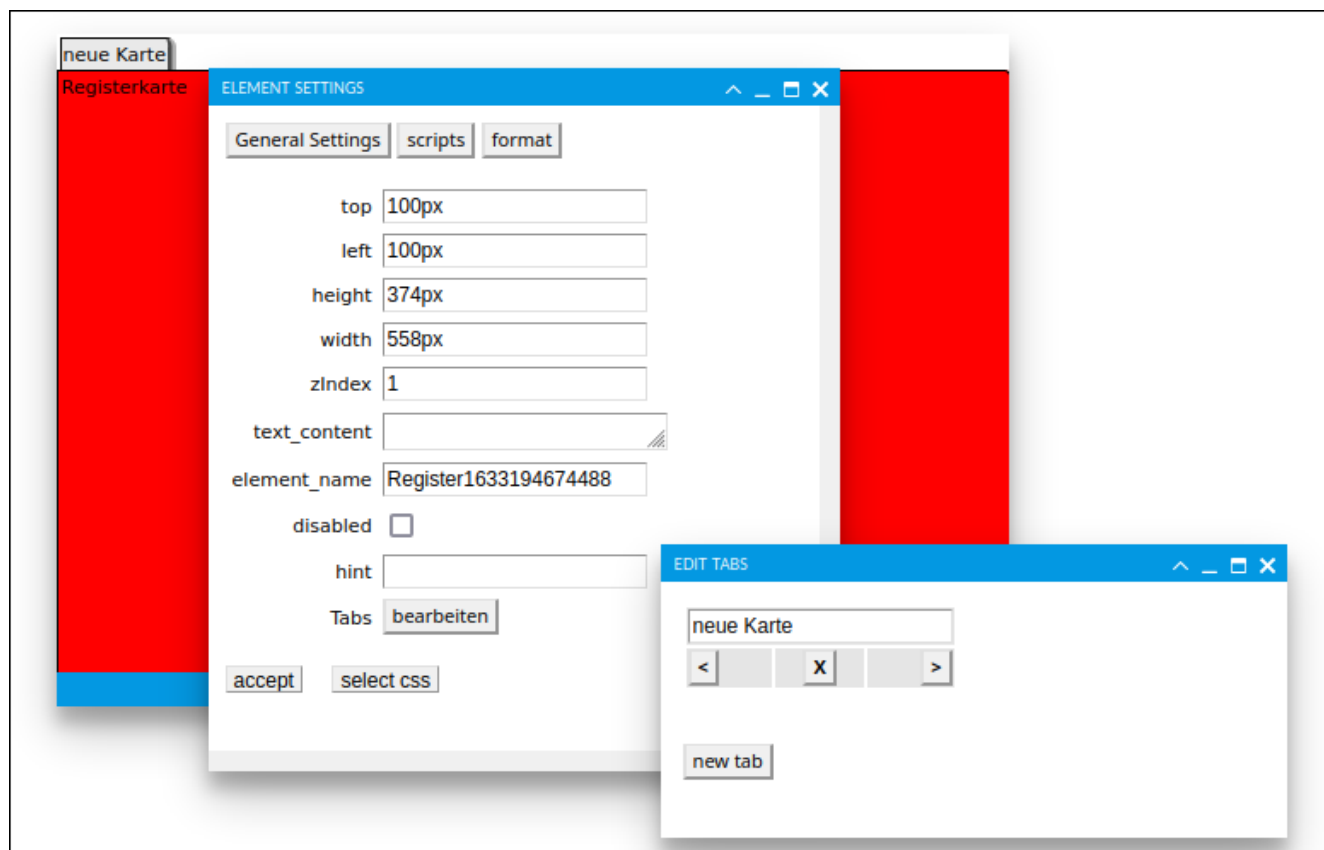


Fig 50: new tab added.

A new tab has been added. Since a tab is only an element, it is displayed in the background as a new, red element.

We can now give the tab a reasonable name and, if necessary, create additional tabs. The tabs can be moved in the order using the arrow buttons.

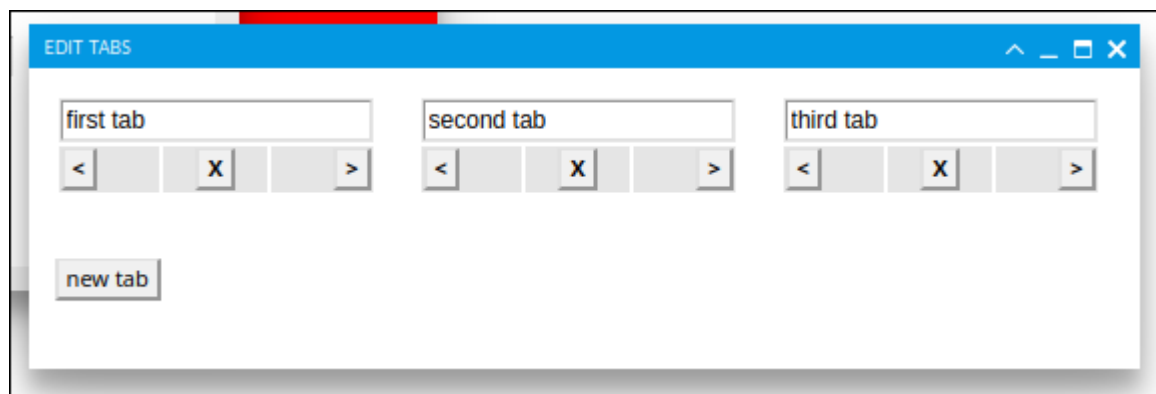


Fig. 51: Register with three tabs.

When we have finished editing the tabs, we simply close the dialog and apply the settings for the tab.

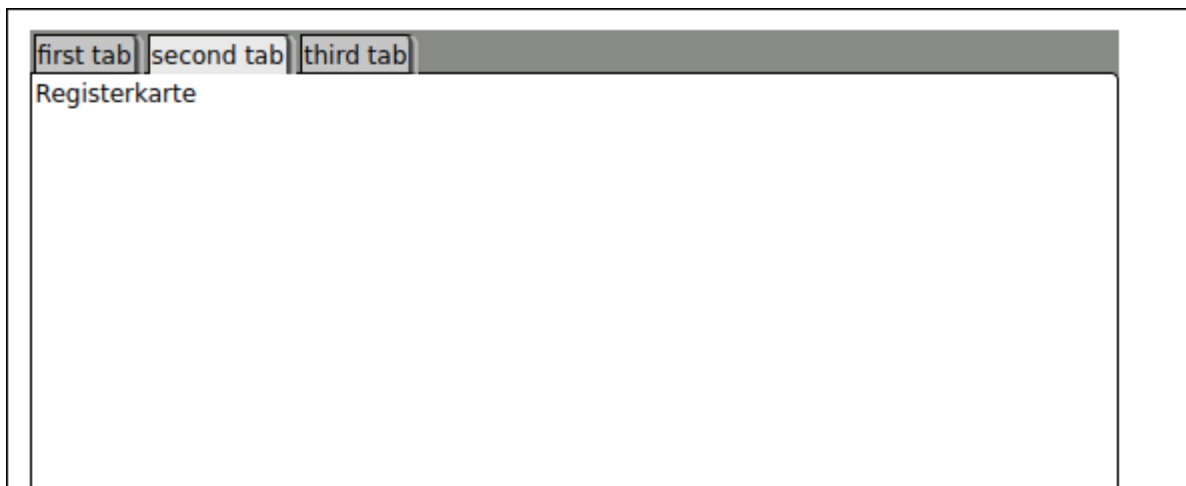


Fig. 52: Finished register with three tabs

We create another element. It's a simple text element. new elements are always created directly on the form. If we now want to place our text element on the second tab, we first select the tab and then drag the text element out of the tab with the mouse. Please make sure that you keep the left mouse button pressed until you drop the element. If you click the element before you move it, it will remain at the form level and will not be subordinated to the tab.

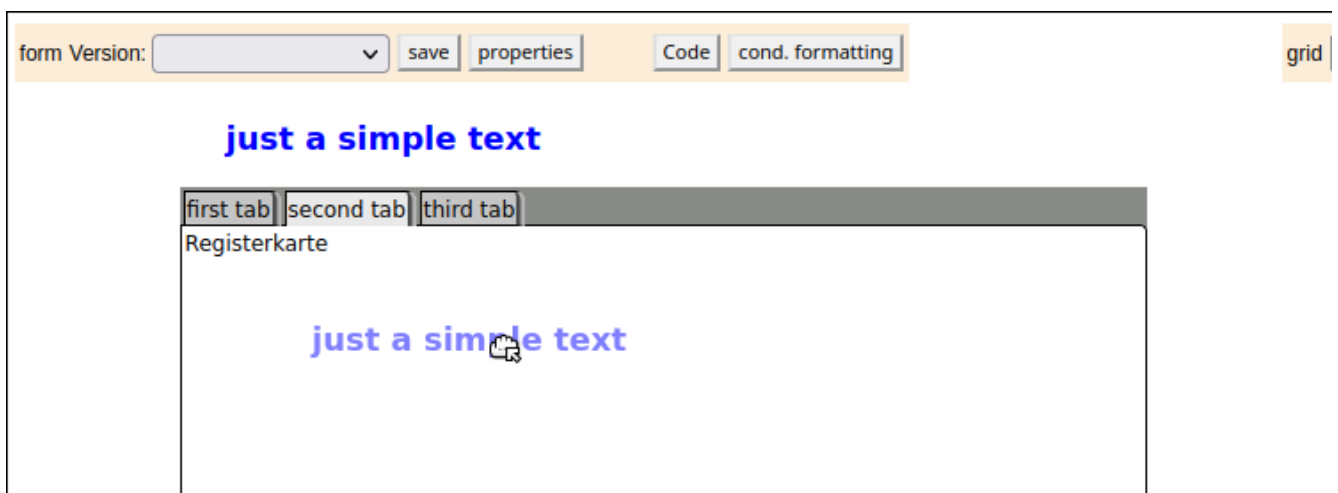


Fig. 53: The text element is dragged onto tab 2.

If you then click on another tab, the text field disappears.

1.2.3. Report

A report-type document compiles records in such a way that they can be printed on paper, or better yet, in a PDF. Because no data is processed here, there are correspondingly fewer elements to choose from. You can only find the element type **grouping** here exclusively. A **grouping** is the equivalent of a subform in a form document. The formats A0 to A6 are available for selection as the page format.

As usual with a report, it consists of different areas. The header and footer appear on every page of the report. They are intended for page numbers, dates, and titles. The report header and report footer are only displayed once at the beginning, resp. displayed at the end of the report. The detail header is displayed in front of the data records. He repeated on each side. This makes it suitable for column headings. The detail footer is only displayed once at the end of the detail area. All areas can be adjusted in height or even hidden completely. The only area that cannot be hidden is the detail area. Incidentally, this area is repeated for each data record.

1.2.3.1. Settings report

FORM SETTINGS

General Settings

form name

background color

database

data source

additional header lines

Js - onload

page format

size Format double-sided printing ☒

margins in mm left right top bottom

hide areas

Page header ☐ Report header ☒ Detail head ☐

Side footer ☐ Report footer ☒ Detail footer ☒

changes

Fig. 54: Document properties of a report

The dialogue is divided into three areas. We already know the first part from the forms. The **page format** area should actually be understandable. However, a note about the option **double-sided printing** could still be helpful. When you print out a document and print on both sides of the paper, you usually want the margin that will be punched a little wider. On pages with an odd number, this is the left margin. On the other hand, on pages with an even page number, it is the right margin. It is also more pleasant if, for example, you can always find the page number on the outside.

The option **double-sided printing** ensures that the left and right margins are swapped for pages with an even page number. The head area and the foot area are divided into three zones. There is a left zone, which takes up 25% of the width of the page, a middle zone, which takes up 50% of the width of the page, and then a right zone, which is again 25% wide. When printing on both sides, the left and right zones are also swapped. Both the information for the margins and the zones always relate to the first page, i.e. the pages with an odd page number.

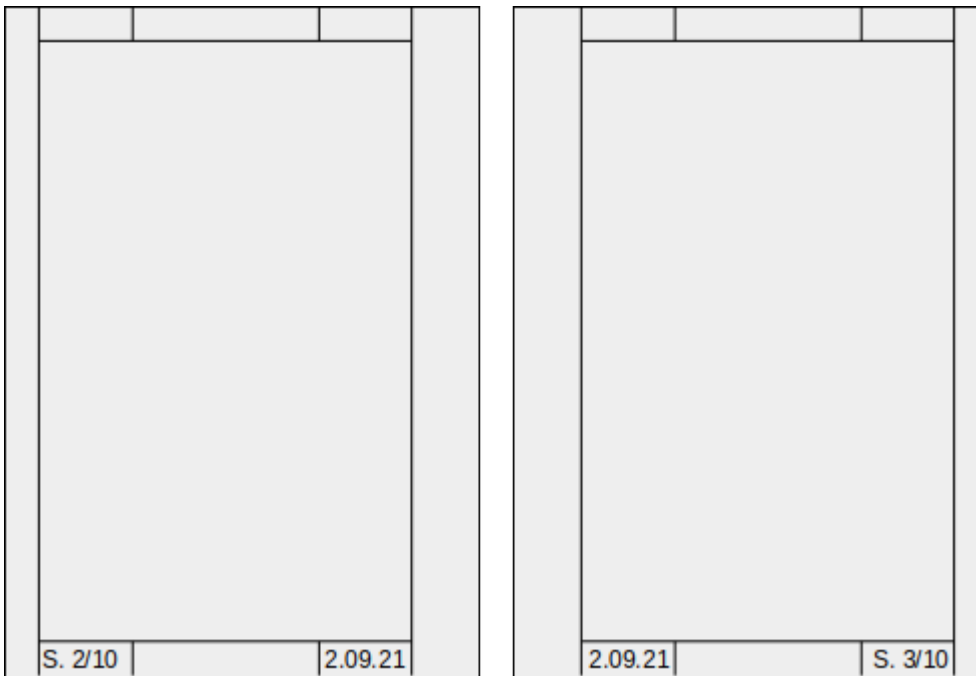


Fig. 55: **double-sided printing** option switched on



Fig. 56: Option **double-sided printing** switched off

The section **hide areas** lists all areas that can be hidden. A cross means that the area has a height of 0 pixels and is therefore not visible in the report. You can also hide an area via the context menu by right-clicking on the corresponding area. In the dialog for the report properties, the cross can be removed at any time in order to display the area again at the previously set height.

1.2.3.2. cover sheet

A report can be provided with a cover sheet. The cover sheet consists of only one large area. If you display page numbering in the report, the cover sheet is not included. To insert a cover sheet, simply press the Cover Sheet button in the header.

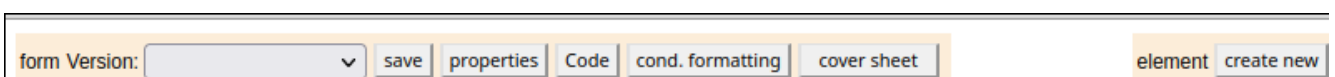


Fig. 57: Header with the **cover sheet** button

The sheet is represented by a solid line. The set margins are marked with a dashed line. You can't remove these lines, but don't worry, they only appear in draft mode.

As soon as you place at least one element on the cover sheet, the cover sheet is automatically shown in the user view. However, if you leave the cover sheet blank, it will not appear in the user view.

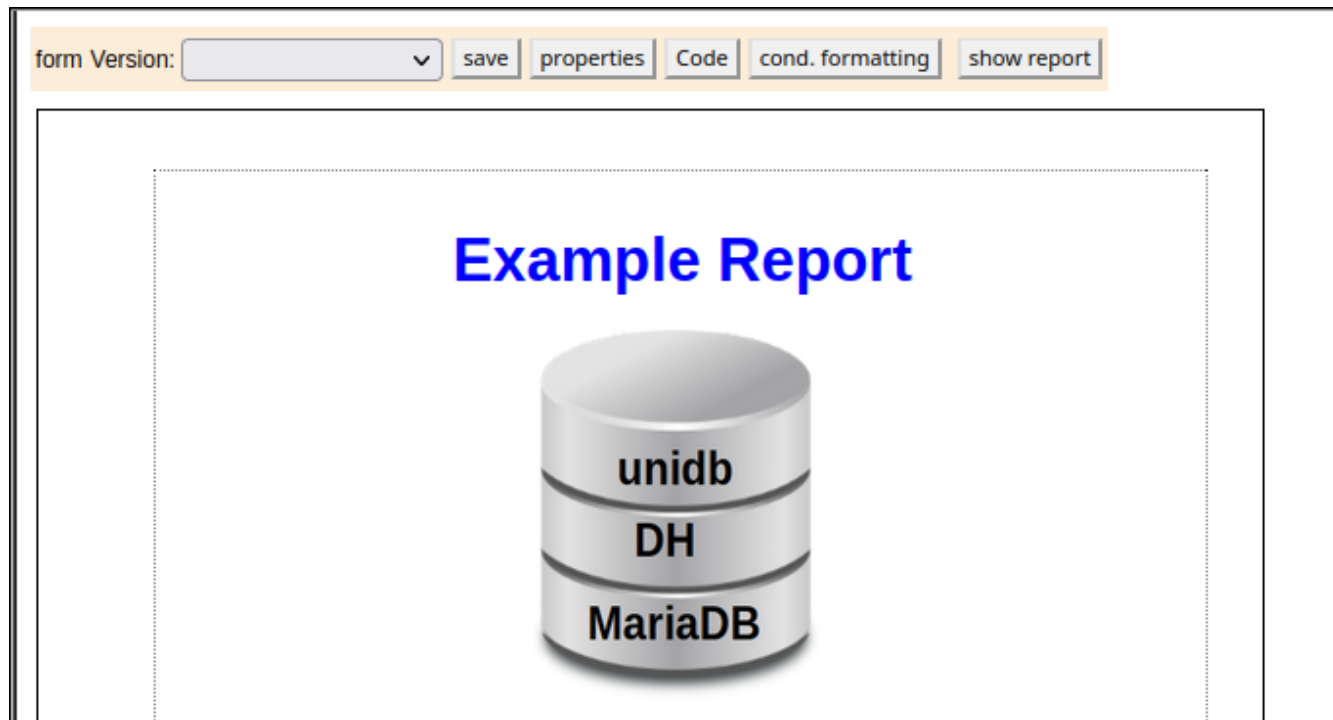


Fig. 58: Cover sheet with one text and one graphic element each

Instead of the Cover Sheet button, you will see the **show Report** button in this view. This will bring you back to the previous view.

1.2.3.3. Scaling

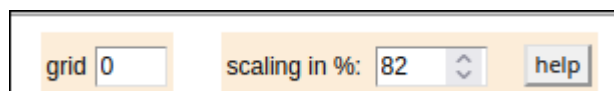


Fig. 59: Setting the scaling in the header

The presentation of the report draft can be continuously enlarged or reduced on the screen. With a scaling of 200% you can place elements precisely and with 70% you have a complete overview of the page. When scaling, the dimensions and positions of all elements are recalculated.

Attention, watch out!

If you create a text element with a scaling of 200% and select 12px for the font size, then the actual font size when printing the report is only 6px, since the report is of course always 100% of the size.

1.2.3.4. Header and footer

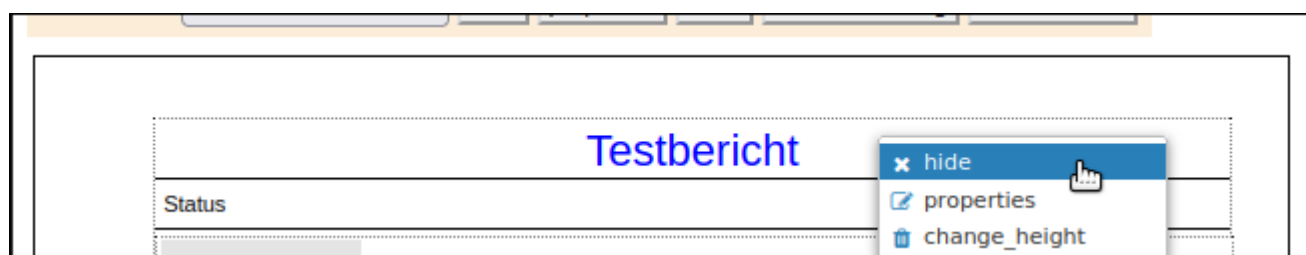


Fig. 60: Header (text test report) clicked with the right mouse button

A click with the right mouse button over an area brings up a context menu with three options. The **hide** option sets the height of the area to 0px and hides it, but does not delete the content.



Fig. 61: Changing the height of the area

The **change height** option marks the area. You can now change the height of the area with the mouse button. As soon as you click on another area, the marking is canceled.

Additional properties of the selected area can be set using the dialog that you open using the Properties option in the context menu. This dialog contains significantly more setting options for the header and footer than for the other areas. We will first consider the setting options that are available for all areas.

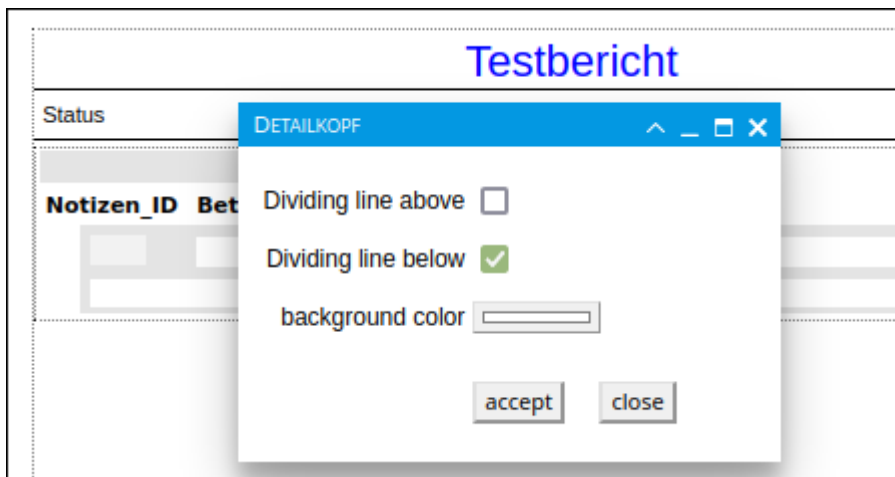


Fig. 62: Dialog for the properties of the detail header area

Sometimes you don't know which area you have chosen. Therefore, the selected area is always displayed in the header of the dialog.

There is not much to do here. **Dividing line above** draws a thin, 1px thick line at the top of the area. **Dividing line below** does the same thing at the bottom of the area. In the example from Figure 62, the area in which the word Status appears has been selected. In this example, the option Separation line below has been checked for the page header and the detail header. Since the report header is hidden, it appears that both options are displayed for the detail header. The area can be colored using the Background color button.

Tip:

The default setting for the background color of the elements is white. You can make your work easier if you temporarily color an area. This makes the elements more visible. Once you have created all the elements, you simply set the background color of the area back to white.

The dialog box is titled "SEITENKOPF" and contains the following settings:

- Section 1:**
 - Name of the report: ☒
 - free text:
 - alignment: center (dropdown)
 - formatting:
- Section 2:**
 - date: ☐
 - Date / time: ☐
 - alignment: center (dropdown)
 - formatting:
- Section 3:**
 - Page / Pages: ☐
 - page: ☐
 - alignment: center (dropdown)
 - formatting:
- Bottom Section:**
 - Dividing line above: ☐
 - Dividing line below: ☒
 - background color:
- Buttons:** accept, close

Fig. 63: Settings dialog for the page header

This dialog is divided into four sections. The lowest section contains the options already discussed, which are available for all areas.

In the three remaining sections you can set what should appear in the three sections of the page header or page footer. In the first section it is indicated whether a freely selectable text, the name of the document, or neither should be displayed. If you do not put a cross here and leave the text field empty, nothing will be displayed. In the example from Figure 63 it was checked that the name of the document should be displayed. In the Orientation field, the middle section of the page header was selected for display. As a reminder, the middle section takes up half the available width. Nothing was checked in the second section. This means that neither the date nor the date and time when the report was generated is displayed. The Orientation field is therefore irrelevant in this case. The next section shouldn't need any further explanation.

We want to see the time at which the report is generated in the left corner of the page footer. The current page and the number of pages should appear in the right corner. Since we specified in the dialog for the report properties that we would like to print on both sides of the paper, the left margin is set a little wider than the right margin. The setting refers to the first page and thus to all odd page numbers. The date and time should always appear inside. Therefore we select left as alignment in the settings dialog for the page footer. The page number should always be on the outside. Therefore the alignment for this is set to the right.

Figure 64 shows the completed dialog including a preview of the footer.

On the occasion: In Design view you only see the variable names. Later, when the report is generated, the current date and time will be displayed instead. It is the same with the page number.

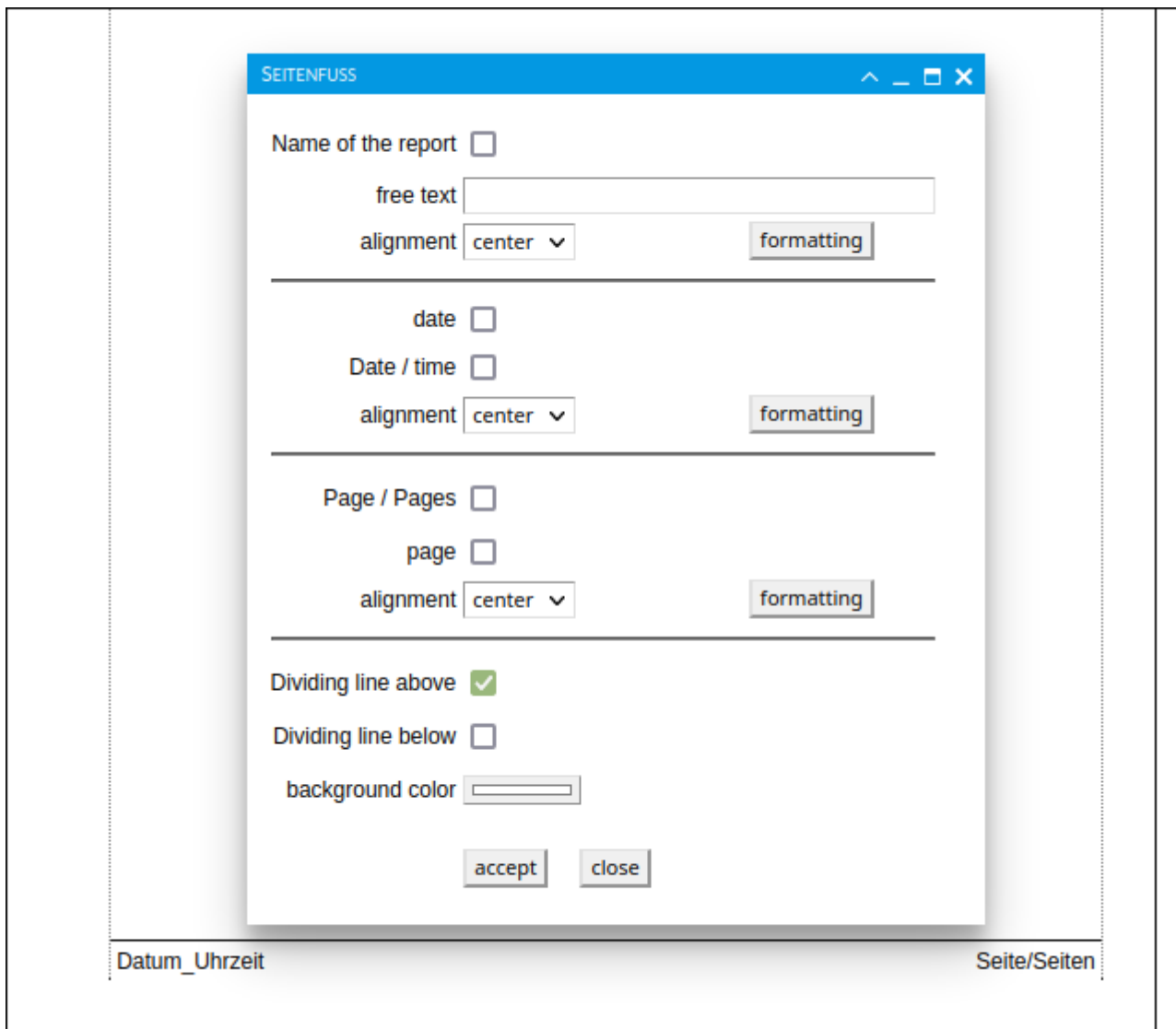


Fig. 64: Settings dialog for the footer

1.2.3.5. Place elements

With forms, a new element always appears on the base. It's a little different with reports. A new element is always created on the currently selected area. To select an area, simply click anywhere on an empty space in the area.

A new element is also colored red here.

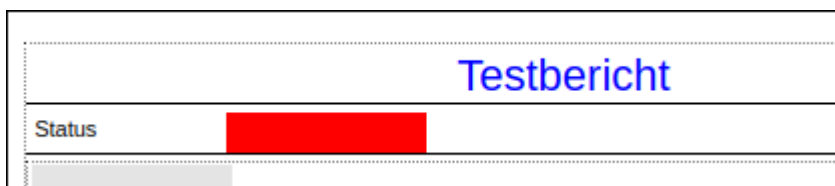


Fig. 65: New text element in the detail header

When you click it, it looks different than the forms. The reason for this is that the elements have to be arranged much more precisely when creating a report. A display as with the forms would be a bit more comfortable, but would make the precise alignment of the element more difficult.

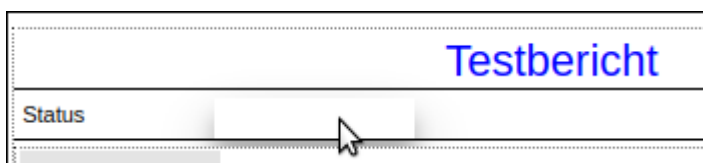


Fig. 66: selected text element.

Figure 66 shows how difficult it is to see a white element on a white background. This can be remedied

by temporarily changing the background color of the detail header.

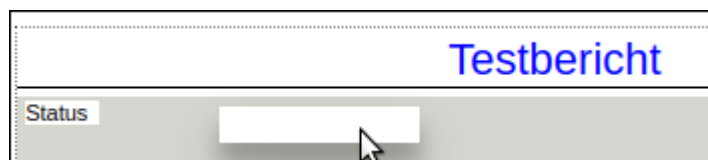


Fig. 67: colored background of the detail head

It's getting better that way, isn't it?

1.2.3.6. Detail area

The details area again fetches with every data record. The unidb tries to keep a data record together on one page. This means that when creating the report, the unidb ensures that the current data record fits completely into the remaining space on the page. If this is not the case, the complete data record is written on the next page.

Fields are displayed variable in height. This means that a field is output sometimes smaller and sometimes larger, depending on its content. When designing the detail area, you do not need to worry about whether the entire field content is displayed if it should be a bit more extensive than usual. If you select the detail area too large, i.e. with plenty of space below the fields, this space is repeated for each data record. This can result in an extremely long report with a lot of free space between the records.

1.2.3.7. grouping

What the subforms are for the forms, the groupings are for the report.

Unlike in some other report generators, a grouping in a unidb report always has its own data source. The connection to the detail area in which the grouping is located is made via a common database field. So we actually have the same principle here as with the subforms. The difference is that a subform is a separate form. With a grouping, however, it is only a matter of making the field contents available. Therefore the data source is entered in the dialog for the element properties.

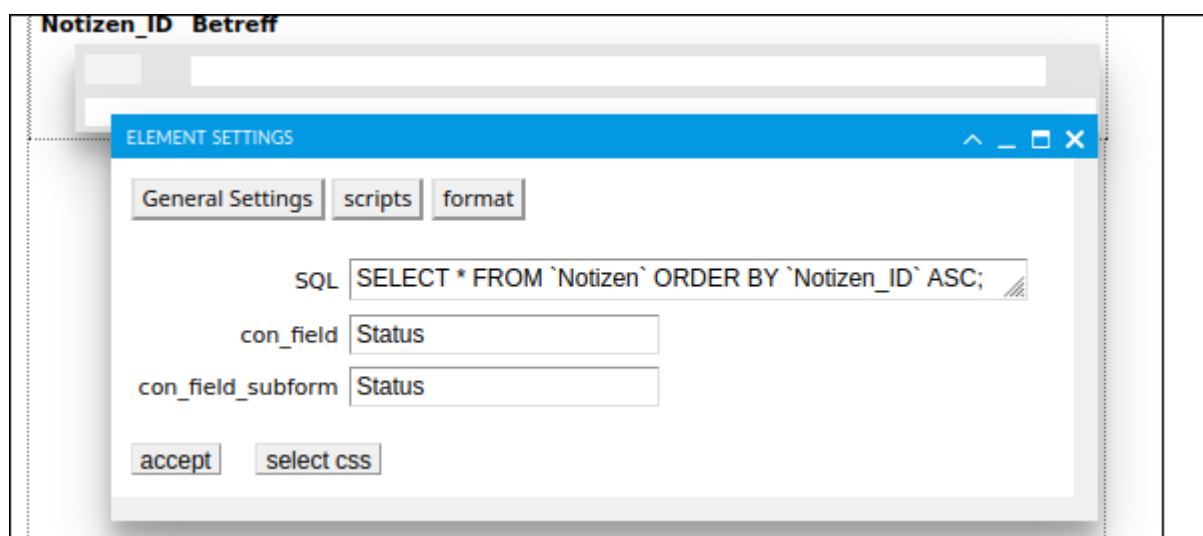


Fig. 68: Dialog for the properties of a group

Since we do not need any formatting or other properties for a grouping, the dialog is very clear. The database query for the detail area is written in the SQL field. In the example from Figure 68, the Status field is used for grouping.

You can nest groupings as well as subforms. The linked field of a nested grouping always refers to the corresponding field of the superordinate grouping.

1.3. Functions

Functions are used for conditional formatting, calculated fields and in scripts. In order to be able to access elements easily, the unidb provides you with the functions Elem () and Elem_Wert (). Both functions take the element name as an argument. We therefore recommend that you give an element

that you want to use in one of these two functions a meaningful name. You can do this simply via the Element Properties dialog. The unidb automatically assigns a name when an element is created. This name consists of the element type and the Unix time stamp for the time the element was created. That name will be unique, but it won't tell you much. A self-assigned name is much better.

important!

Please make sure that you only use a name once.

Elem (element name)

Returns the unique ID of an element. You can find the element name in the Properties dialog of an element.

Elem_Wert (element name)

Returns the field content of a database field in a form or report. You can find the element name in the Properties dialog of an element.

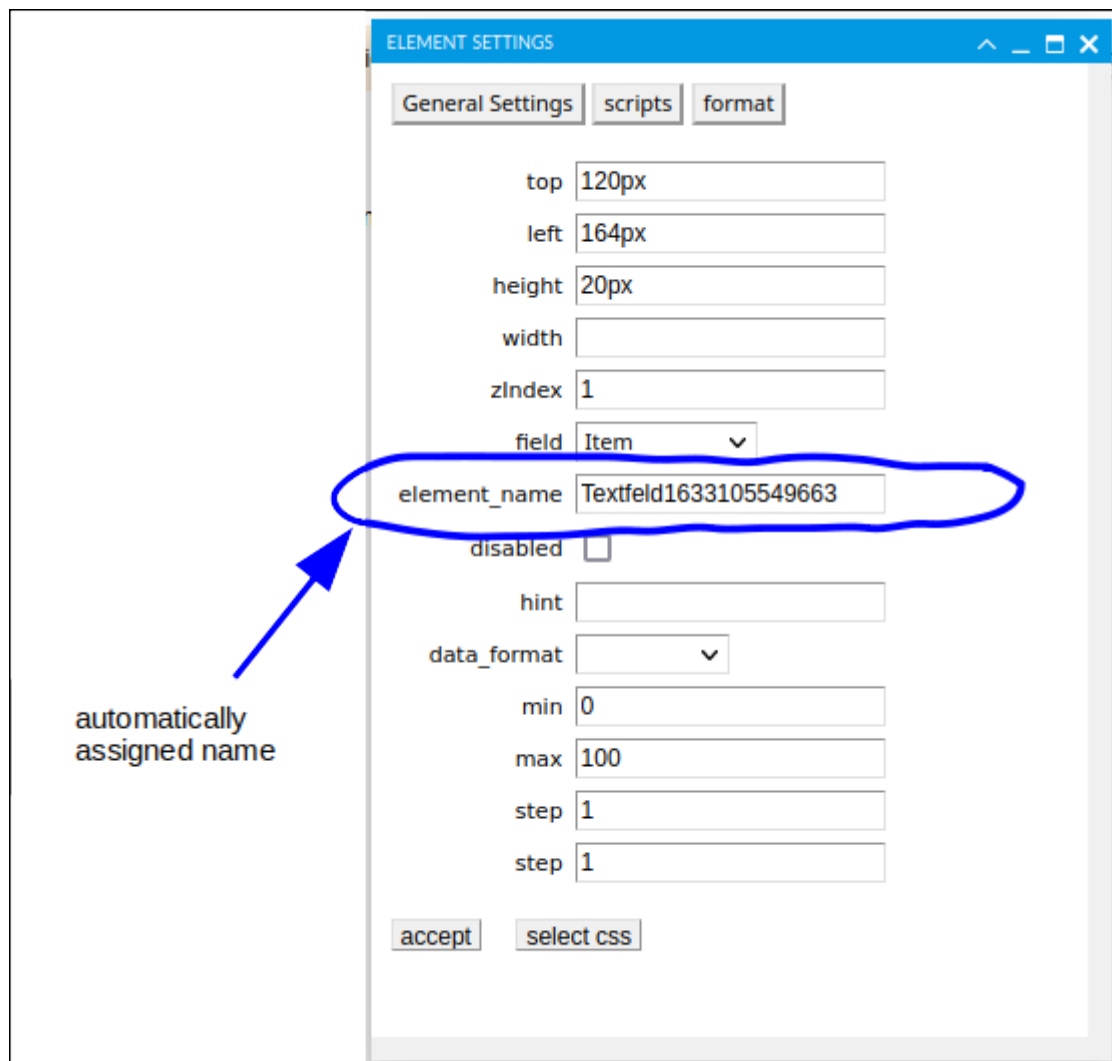


Fig. 69: Settings dialog for an element with automatically assigned element names

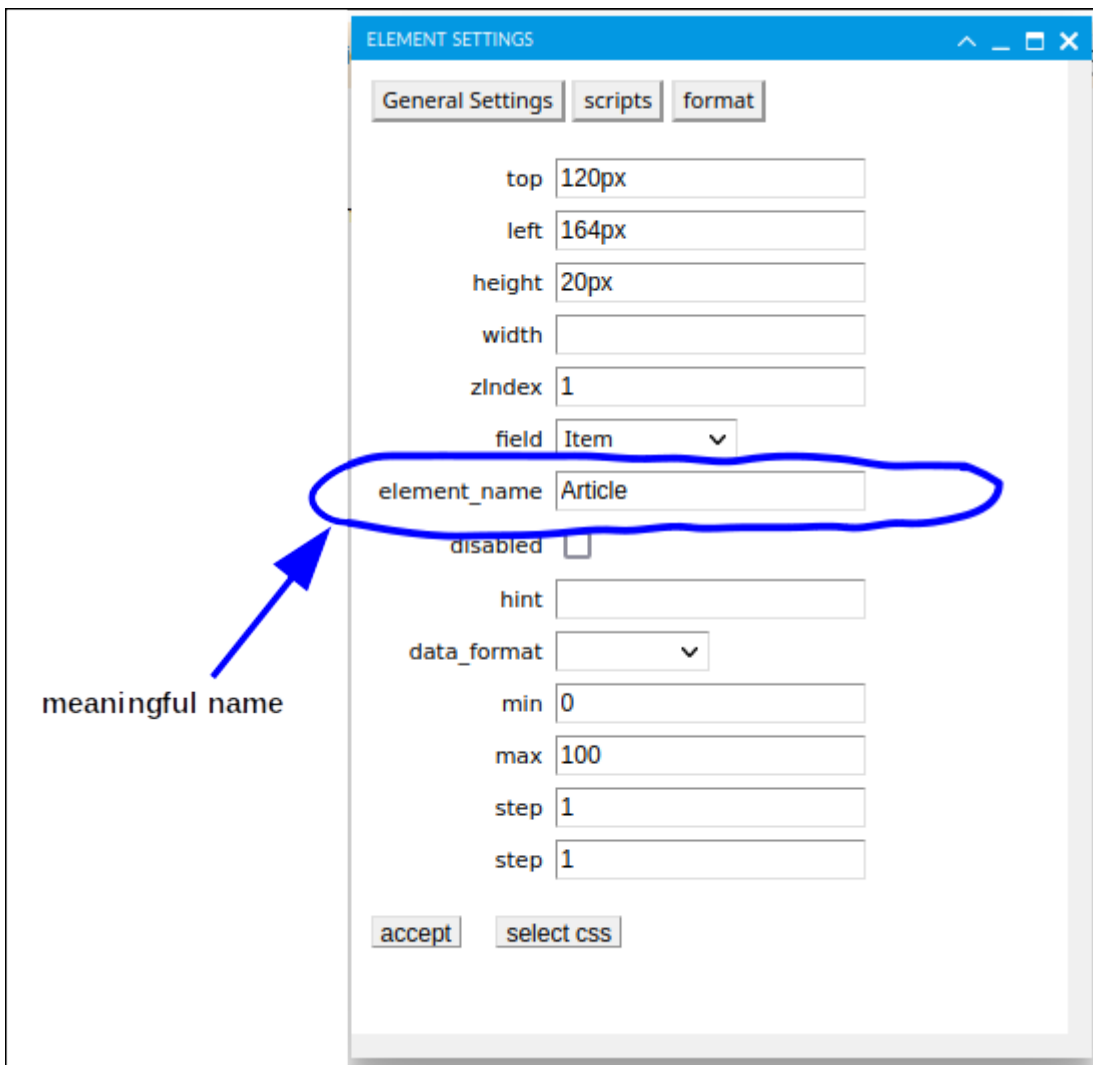


Fig. 70: meaningful element name

DBQ (database, function, field, table, condition)

This function returns exactly one value from a database table.

Parameter:

- Database:
This parameter is optional, but you should always include it. If no database is specified, the unidb uses the database currently in use.
- Function:
A MariaDB / MySQL function such as MIN, MAX, COUNT, ...
The specification of the function is optional. If no function is specified, the unid returns the first value it finds.
- Field:
The database field that contains the desired value.
- Tabel:
The database table.
- Condition:
Specifying a condition is optional. The syntax is the same as used in the WHERE part of a query.

Examples:

We would like to know how many customers we can find in the customer database who live in Berlin.
Number = DBQ ("Customer", "COUNT", "Customer_ID", "masterdata", "" Town` like 'Berlin' ")

We are looking for the place of residence of the customer with the customer ID 45872.
Place of residence = DBQ ("customer", "", "Town", "masterdata", "" customer_ID` = 45872")

1.4. programming

You can insert your own JS code into forms and reports. This can be done in several ways. Additional JS and CSS files can be entered in the document's settings dialog. JS code can also be entered there, which is executed immediately after loading the document.

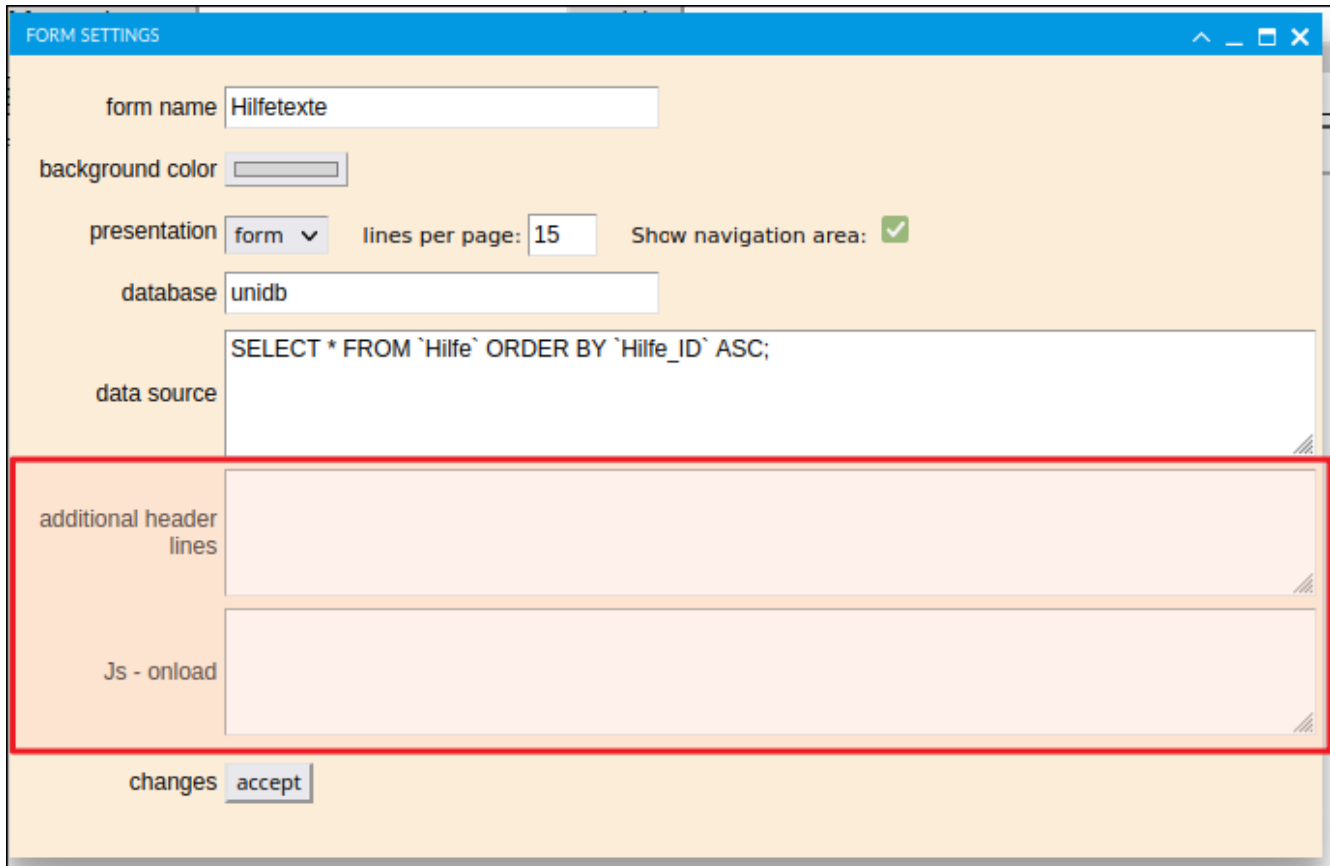


Fig. 71: Dialog form settings, the fields for the integration of own code are marked.

Another option is to react to events of the elements in the document. Figure 72 shows a section of a form. On it you can see a switch and a graphic that is supposed to represent a control lamp. The switch is selected on the pane and the settings dialog for that item is also visible. If you click on the **Scripts** button in the dialog, you can enter a function name for the *onclick* event. In our example this is `switch_Lamp();`.

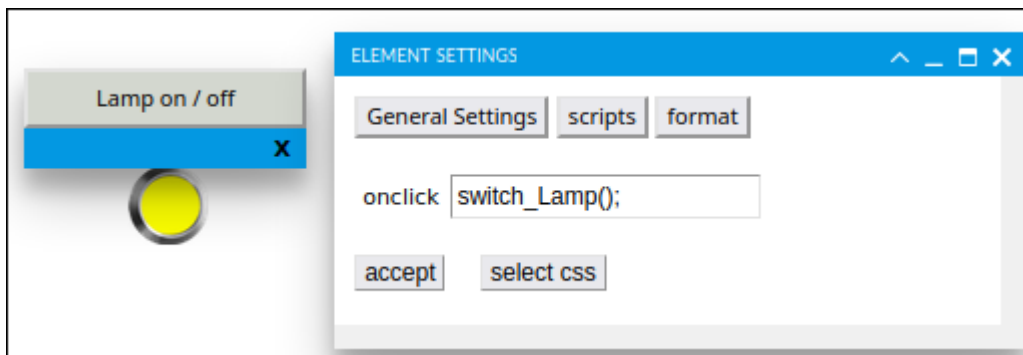


Fig. 72: Settings dialog for the onclick event of the button.

The `switch_Lamp()` function can be entered in the dialog that opens when you press the Code button in the header. Alternatively, the function can also be in a separate JS script file, which is entered as an additional header in the dialog for the document settings, as shown in Figure 71.

You can make life a little easier if you give the elements their own element names. Figure 73 shows the settings dialog for the second element. The element is named *Lamp*. This makes it easier to address the element later in the `switch_Lamp()` function.

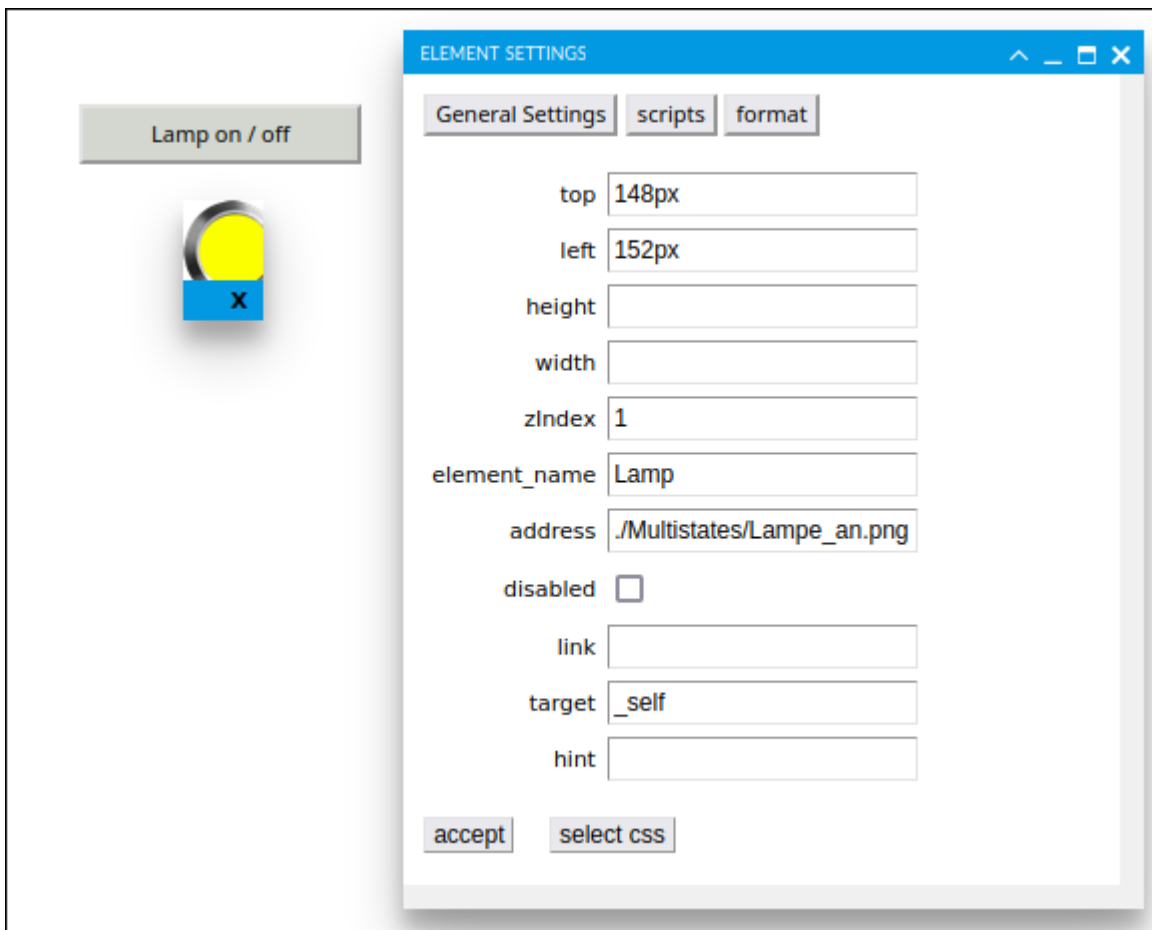


Fig. 73: Settings dialog for the graphic.

In the next figure, the code represents the action to take when the button is pressed. When the button is pressed, the `src` attribute of the graphic element is exchanged for another graphic. In this way, the control lamp appears to be switched on or off.

In the first line, an object variable named *Lamp* is created. This is done using the *getElementById* function, which expects the id of the object. Instead of specifying the id directly, the *unldb* function *Elem()* is used here, which returns the id for the element name specified in brackets.

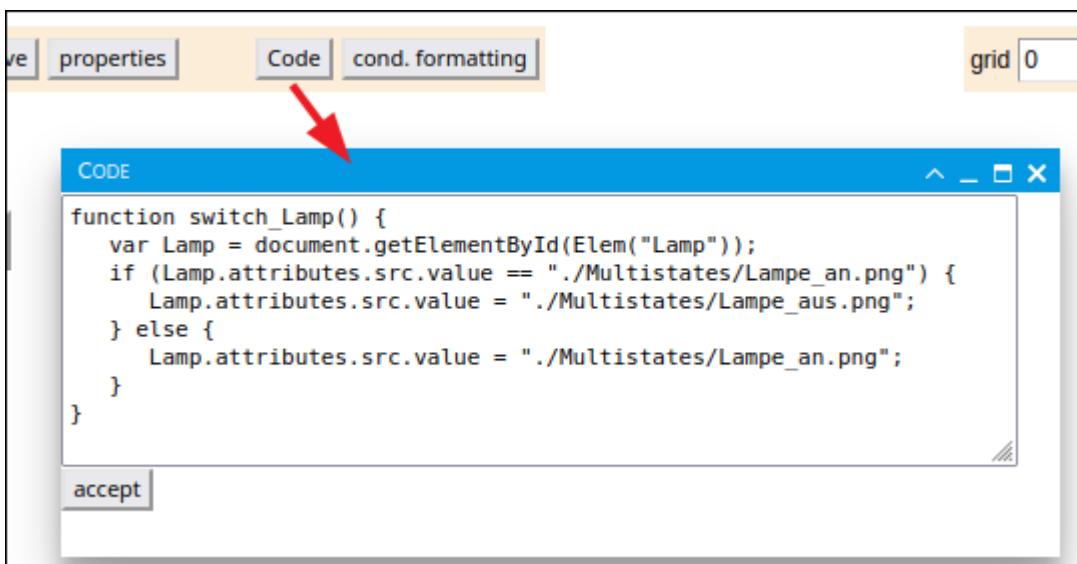


Fig. 74: JS code for the function *switch_Lamp()* in the code dialog.

An action that doesn't particularly make sense in practice, but is always good enough as an example, as you can see in Figure 75.

When the form is created, the source of the graphic element is set to the *Lampe_an.png* file. In order to always present the user with a switched-off lamp, two lines of code are executed immediately after the document has been loaded.

Fig. 75: Code that is executed immediately after loading the document.

2. DH

2.1. Collect, process and provide data

Before we turn to the processing of images and groups, it is good to know how the data gets into the DataHistorian and how it is processed there. Don't worry, it won't get too theoretical.

The DH itself does not read any data. The data is written to the DH by many small programs. These programs are called **interfaces**.

The interfaces write the data, i.e. *Point_ID*, *time stamp* and *value*, into a table with the name **act**.

Another program works as something like a compressor. This compressor reads the data from the table *act* line by line and checks whether the respective value is discarded or whether it is written to the **archive**. The compressor deletes discarded and archived values from the table *act*.

This means that the latest value of a point is always in the table *akt*. The compressor also calculates the statistical values (hMW, hMIN, hMAX, dMW, ...), if a point has been configured for this.

The values of many points are calculated by an interface that is usually called **calc**. These calculated values are also written to the table *act*. The DH specific functions are described in section 2.4. When using these functions, a distinction must be made as to whether they are reading a value from the table *akt* or from the *archive*.

2.2. Edit DH documents

In the case of the trend group and spreadsheet document types, no distinction is made between design mode and view mode. Therefore, these document types have already been described in the user manual.

Groups are very simple documents, but you can save yourself a lot of work here if you organize and use the paths correctly.

In terms of their complexity, images are comparable to forms or reports. We shall deal with this in a little more detail.

2.2.1. Groups

A group is really just a simple table. In each line there is a day, a time stamp and a value matching the time stamp. You can add or remove tags from a group. The order of the tags can also be edited. And that's basically it.

It becomes interesting when there are several or even many tags with the same name but different path. Instead of creating a group over and over again, it is sufficient to copy an existing group and simply

change the path for the tags it contains.

This becomes clearer with an example:

We imagine that we operate a wastewater treatment facility. All the rainwater from the plant flows into a collecting basin there. From there it is pumped into the sewer if it is clean. Each production area operates its production facilities outdoors. It goes without saying that these production systems are each located in a concrete tub. The rainwater flows from the concrete tub into a collecting basin and is pumped from there into the large collecting basin for waste water treatment. The rainwater from the streets also flows into the large collecting basin.

What is noticeable here is that we have several constructions that are always set up the same way. Water flows into a basin and is pumped away from there. So it stands to reason that you always give the same tag name to the fill level of the basin. The tags differ in their path, which is e.g. based on the department name.

In a drawing you could represent a rainwater basin like this:

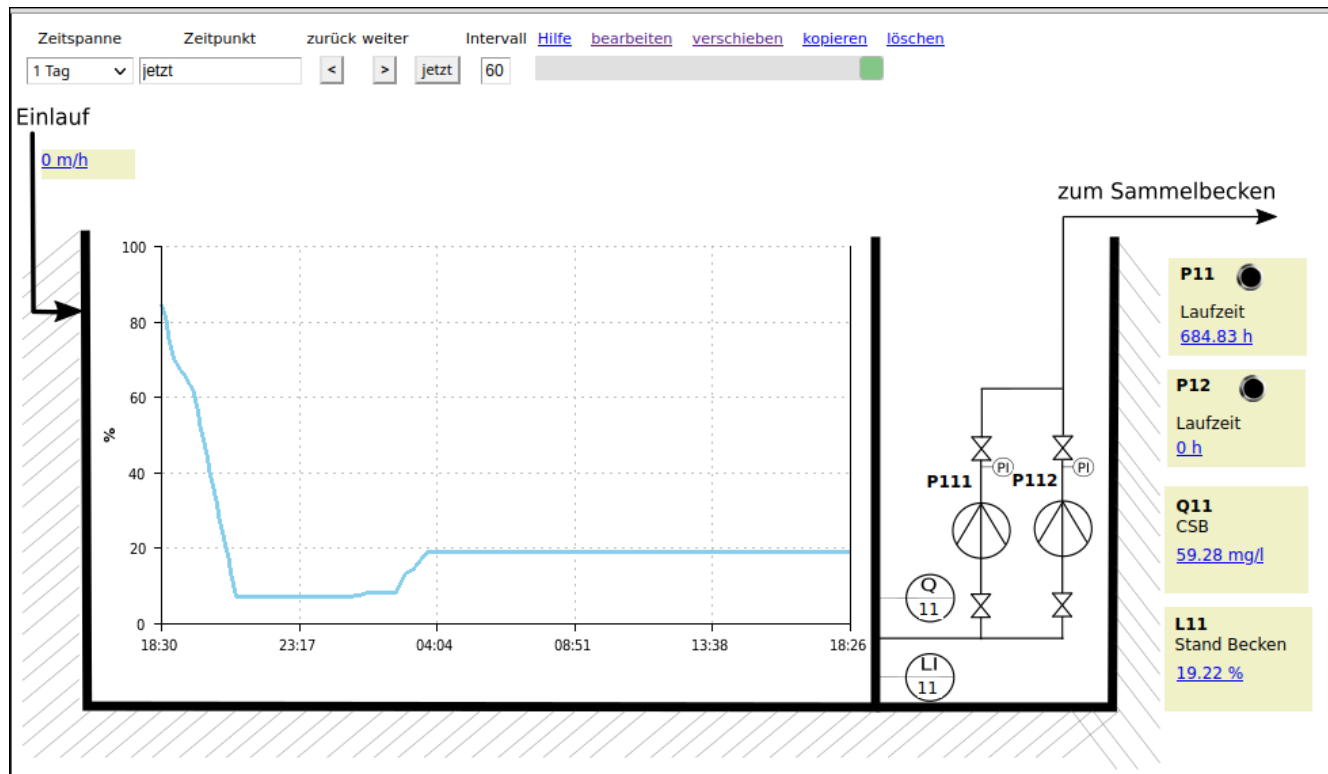


Fig. 76: Schematic representation of a rainwater basin

In every basin, we are always dealing with a level measurement, a COD value, an inflow quantity and two running displays of the pumps.

If we now build a group that displays these measurements, it looks something like this:

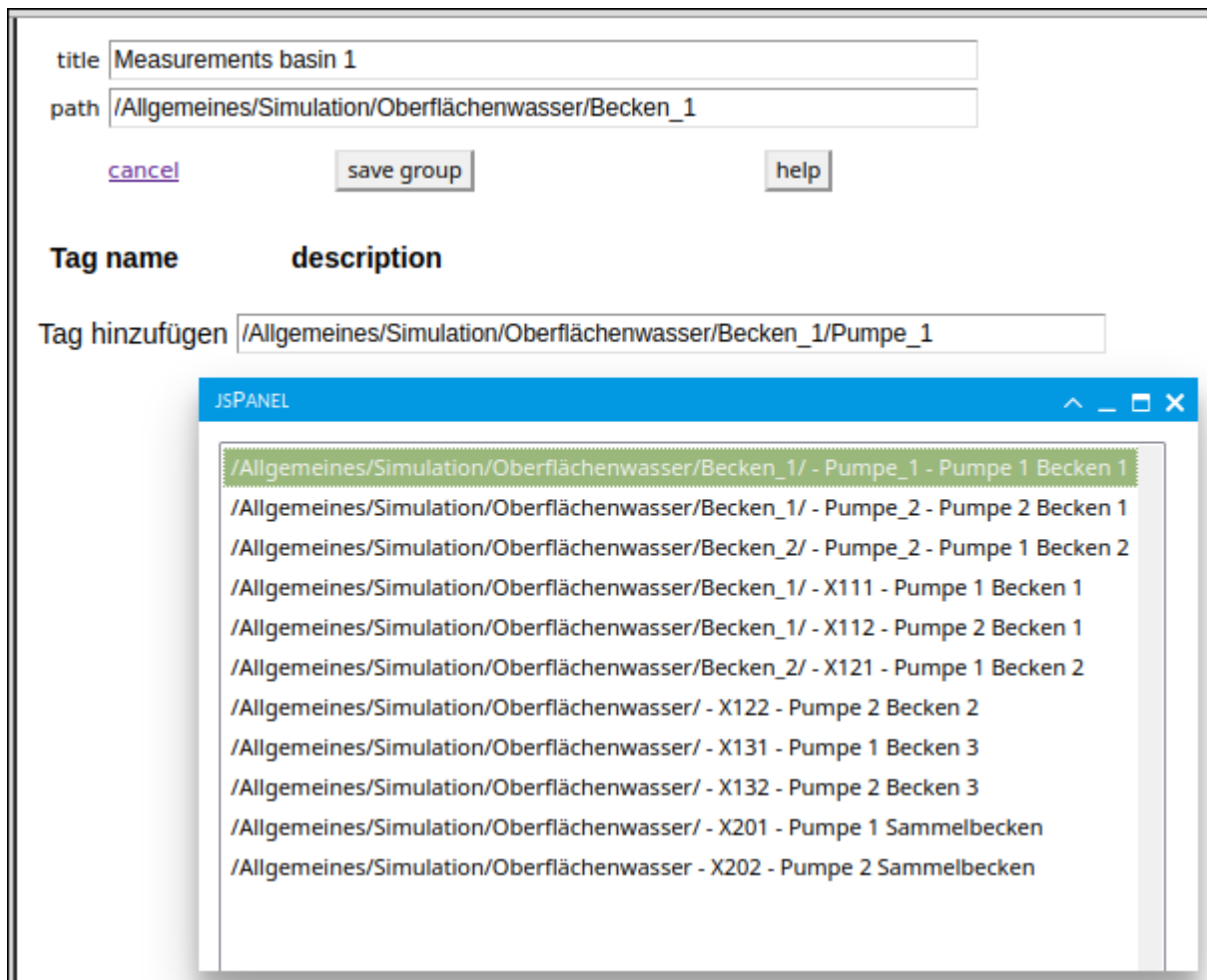


Fig. 77: Searching for a tag and adding it to the group.

Our group should be called Measurements Pool 1. Later we will create another group for the measurements from pool 2.

In the Add Tag field, we only enter Pump%. Then we display all tags whose names begin with Pump. We choose the first entry on the list. The Add Tag field now contains the complete tag name including the path.

We press the switch **accept**:

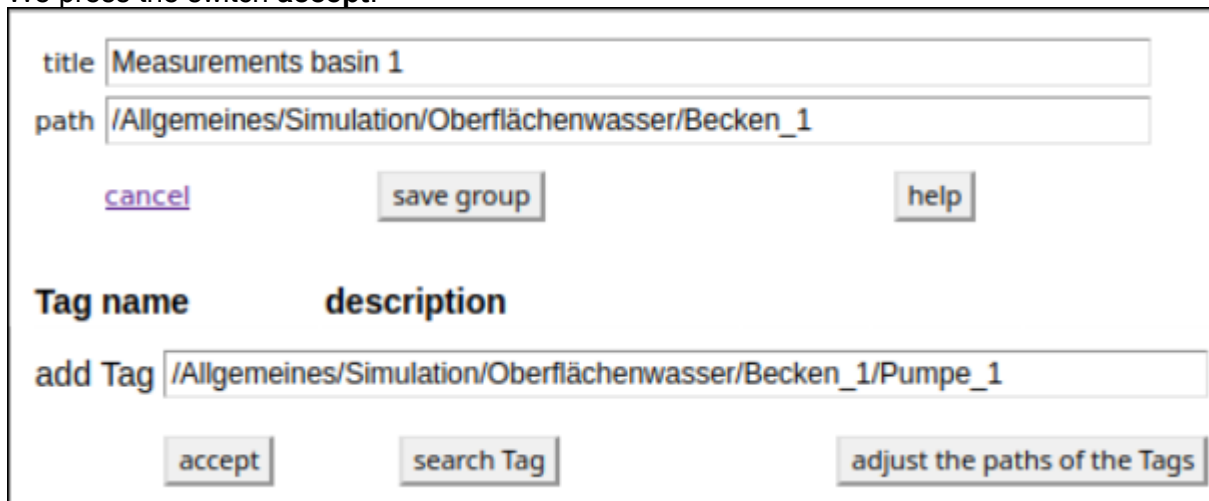


Fig. 78: Transferring the day to the group

The tag has been adopted:

title

path

[cancel](#)

Tag name	description			
Pumpe_1	Pumpe 1 Becken 1	up	down	remove

add Tag

Fig. 79: first tag in the new group

After all tags have been entered, we write the path of these tags in the second field from the top with the name Path.

The fully configured group looks like this:

title

path

[cancel](#)

Tag name	description			
Pumpe_1	Pumpe 1 Becken 1	up	down	remove
Laufzeit_Pumpe_1	Laufzeit Pumpe 1 Becken 1	up	down	remove
Pumpe_2	Pumpe 2 Becken 1	up	down	remove
Laufzeit_Pumpe_2	Laufzeit Pumpe 2 Becken 1	up	down	remove
Einlauf	Wasser zum Becken	up	down	remove
CSB	online CSB Wert Becken 1	up	down	remove
Wasserstand	Stand Becken 1	up	down	remove

add Tag

Fig. 80: fully configured group

And now as usual:

Just don't forget to press the Save group button!

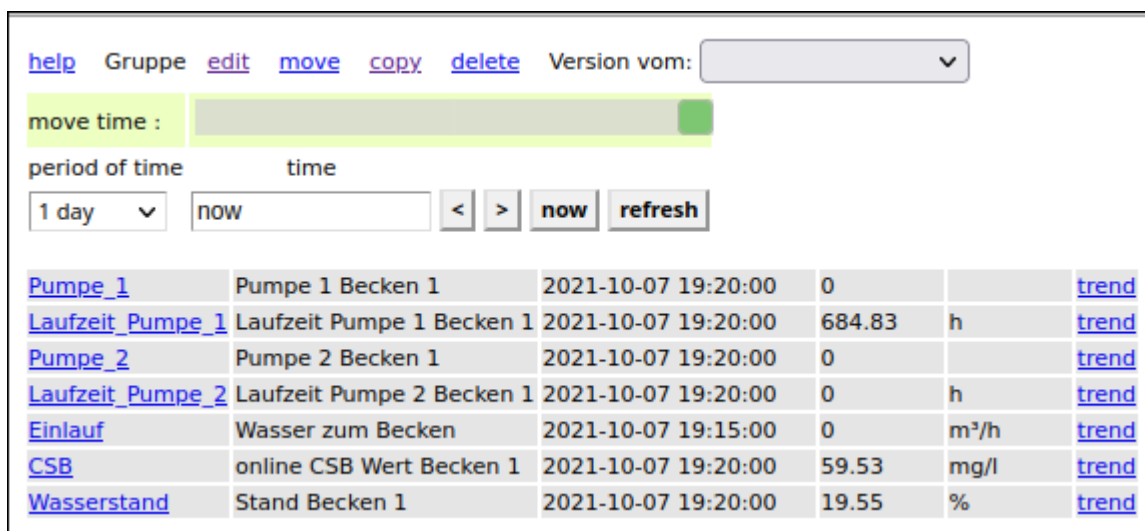


Fig. 81: our group in viewing mode

Now we have a good template for Basin 2, Basin 3, ... and the collecting basin. We will now create the group *Measurements Basin 2* very quickly. To do this, we copy the group *Measurements Basin 1* via the link Copy in the header to *Measurements Basin 2*.

The group *measurements Basin 2* is first and foremost an exact copy of the group *measurements Basin 1*. We change this by changing the path for the group and then pressing the button **adjust paths of the tags** at the very bottom. A small dialog appears, which ideally shows that for each tag name from the group it has found a tag with the path specified in the **path** field.

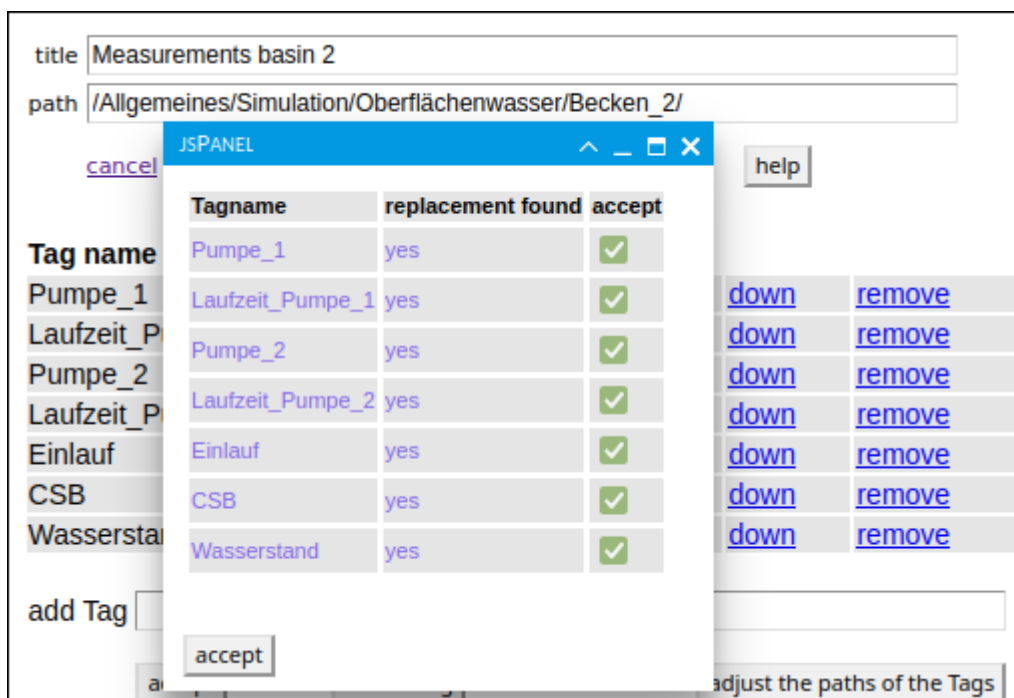


Fig. 82: Path changed, **adjust the paths of the tags** pressed, tags found with the path

Now that we have pressed the Apply button in this dialog, the tags in the group are exchanged. Nothing changes optically. But if we save the group now and look at the group in viewing mode, then we can see the difference.

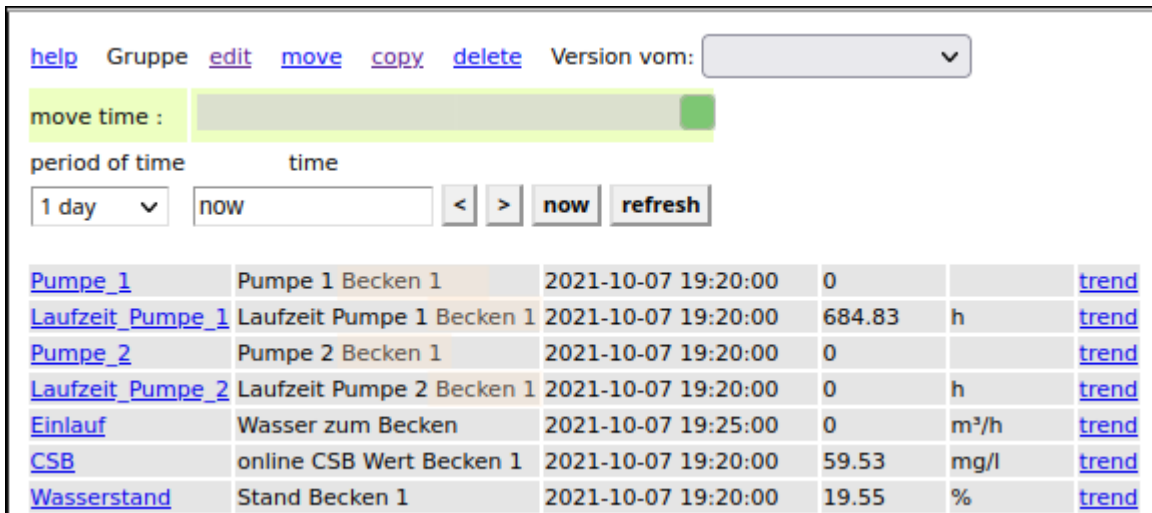


Fig. 83: Group measurements pool 2

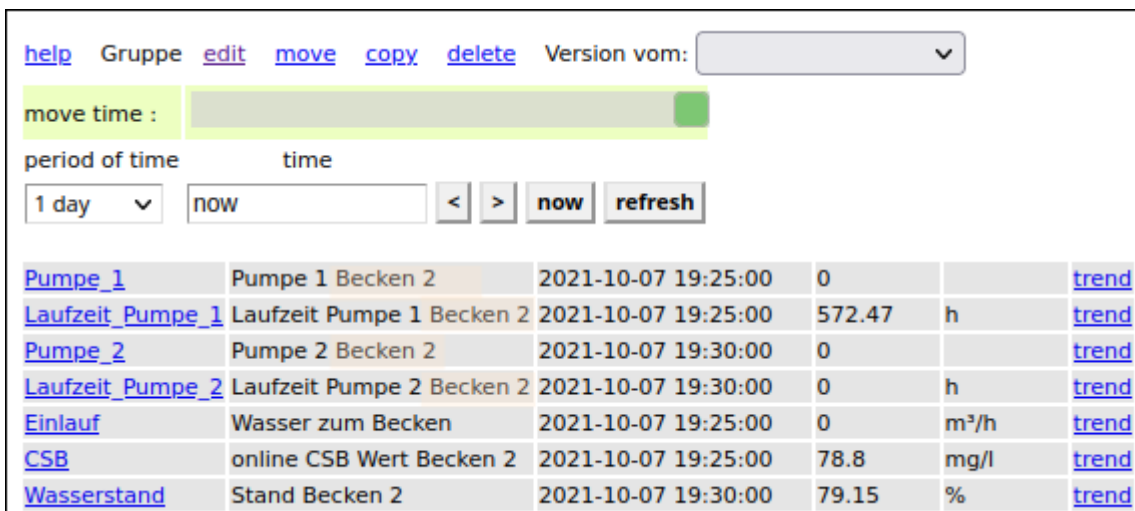


Fig 84: Group measurements pool 1

2.2.2. drawings

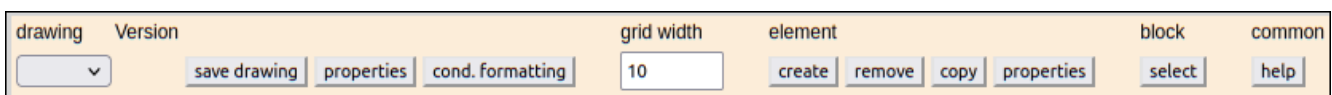


Fig. 85: Header in edit mode

The header is divided into areas. The Image area is responsible for everything that affects the entire document. The Element area always refers only to the currently selected element.

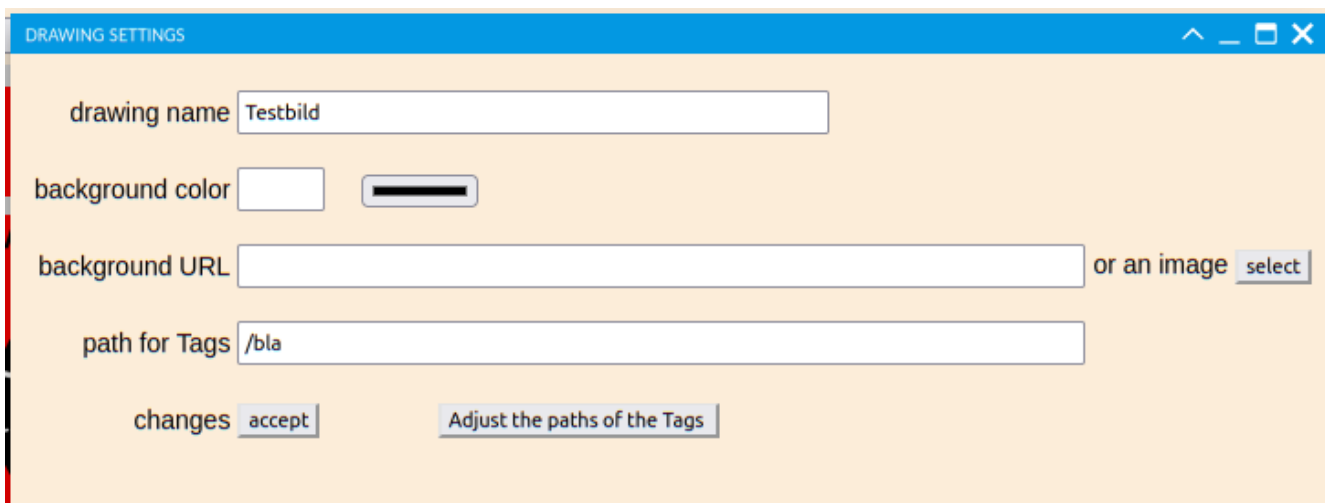


Fig. 86: Settings dialog for the image

The **Image Name** field contains the document name as it appears in the tree structure. The address of the graphic that serves as the background for the image can be stored as the **background URL**. This can be a schematic representation of a machine, a workflow, a P&ID scheme or simply a photo of the object to which the data in the image refers. You have two options to choose from here. Either you enter the URL for the graphic in the field or you select a graphic that is already stored using the adjacent button. If you choose the second option, you will see the following dialog:

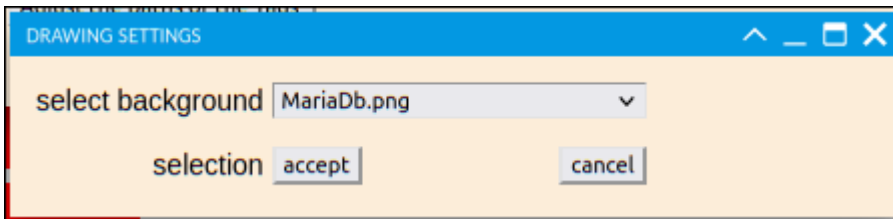


Fig. 87: Selection dialog for the background graphic

Only graphics that you have previously uploaded to the server are displayed in this dialog. The graphics are uploaded via the Settings link above the tree structure.

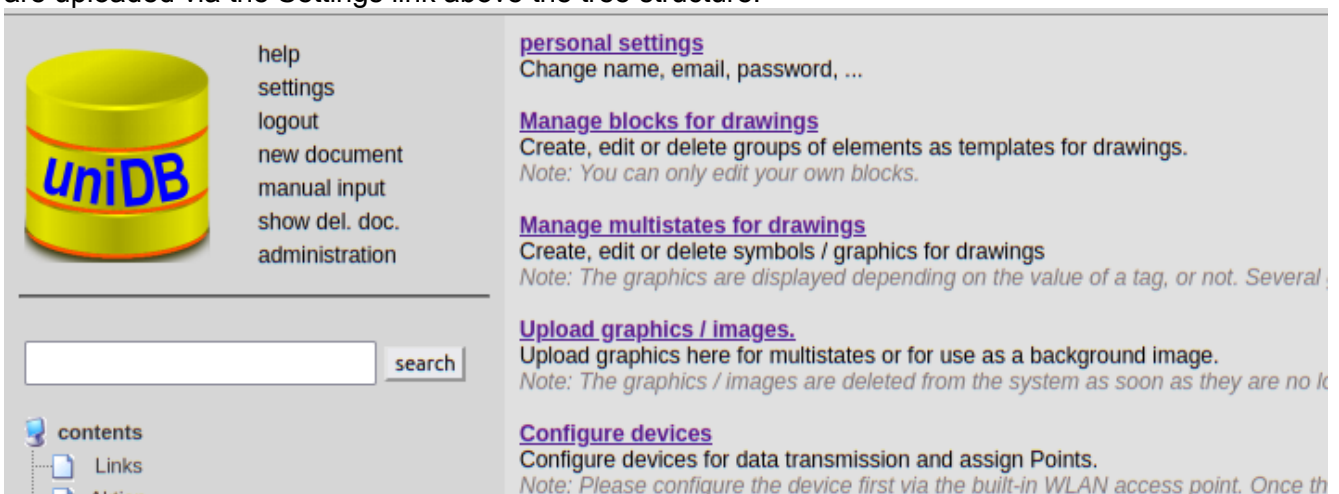


Fig. 88: Settings page with the link Upload graphics / images.

The link Upload graphics / images opens the following dialog:

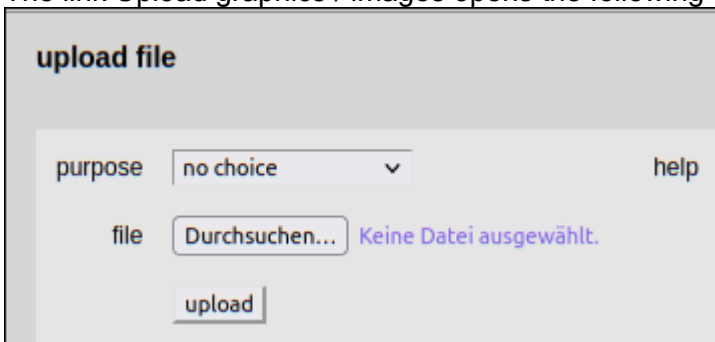


Fig. 89: Upload images

Determination determines what the graphic should be available for selection. Here you can choose between the options Multistate and Background. No further explanation should be necessary for the rest.

We already know the third field in the dialog from Fig. 81 from the design mode of the groups. Again, it makes it easier to adjust a copied image for tags of the same name from another path.

The switch bed. Formatting opens the dialog for conditional formatting. This topic has already been described in chapter 1.1.2.

2.3. settings

Under Settings, you'll find five links, two of which we've already discussed. The links for the multistates, graphics / images and the device configuration remain.

2.3.1. Manage building blocks for images.

A building block is a prefabricated group of elements. The use of building blocks can make the work much easier, since areas that are used again and again do not have to be built in from scratch in each image. After inserting a block, only the tags have to be inserted into the elements. The rest, such as B. the arrangement of the elements, their size and their style, are usually adopted unchanged. When opening the settings page, we first see a header in the already familiar style. The only thing that really stands out in this header is the first field, **Selection**.

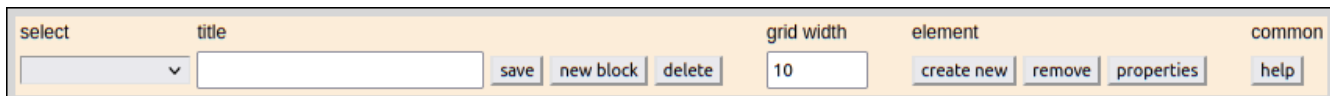


Fig. 90: Header of the settings page for blocks.

All the building blocks you have already created are listed in this field. If you select a module from the list, it will appear below the header for further processing.

If you would like to create a new building block, simply press the **new building block** button.

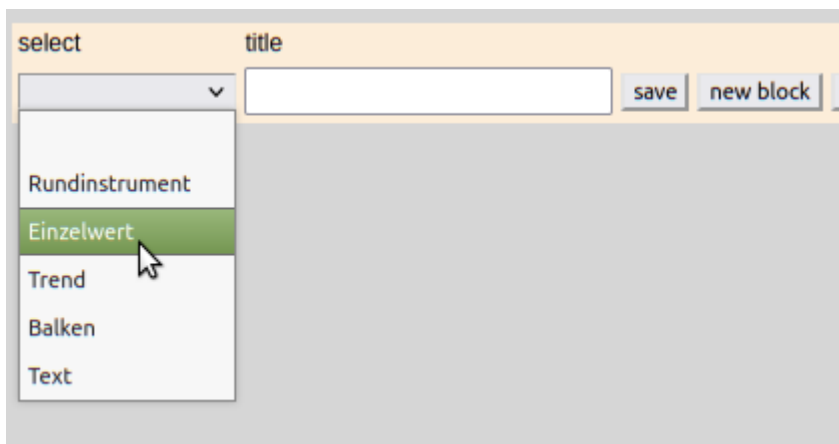


Fig. 91: The Single value block has been selected for editing.

2.3.2. Manage multistates for drawings.

A multistate element is a collection of graphics that are displayed depending on the value of a tag. Example: A multistate consists of two graphics. One graphic shows an indicator light that is off and the other graphic shows the same indicator light when it is on. A tag that represents either the status 0 for switched off or 1 for switched on can be represented very clearly with this multistate. Let's just build a multistate like this.

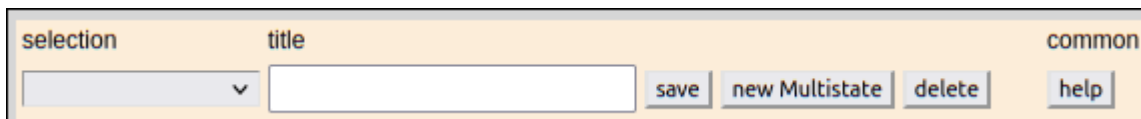


Fig. 92: Empty settings page for multistates

We'll start by clicking the **new multistate** button. You are then expected to enter a designation for the multistate. In our example, the multistate should be given the name control lamp. The screen now looks like this:

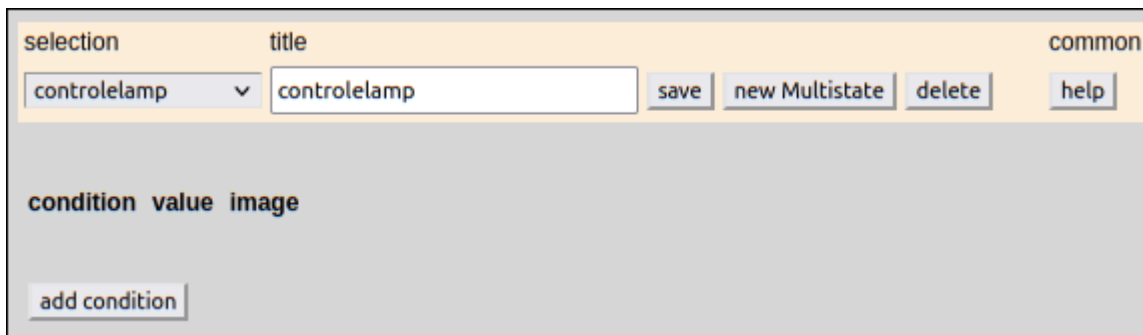


Fig. 93: The multistate control lamp was created. The graphics and conditions are still missing. Since the multistate should display a graphic depending on a condition, we now have to define the conditions and assign the graphics. The graphics are uploaded to the server using the Upload Graphics / Images setting. This process has already been described at the end of Chapter 2.2.2 Images (Upload Graphics / Images).

A click on the **Add condition** button creates a new, empty condition:

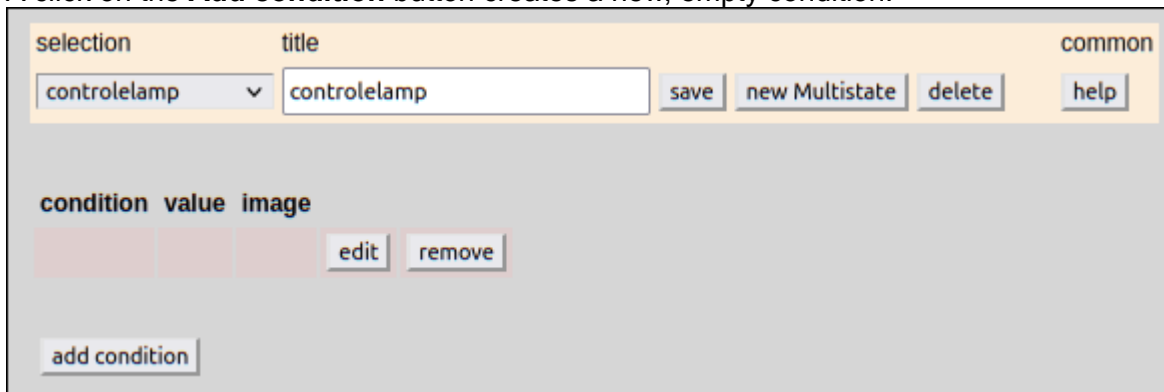


Fig. 94: new, empty condition

We set the condition using the **edit** button:

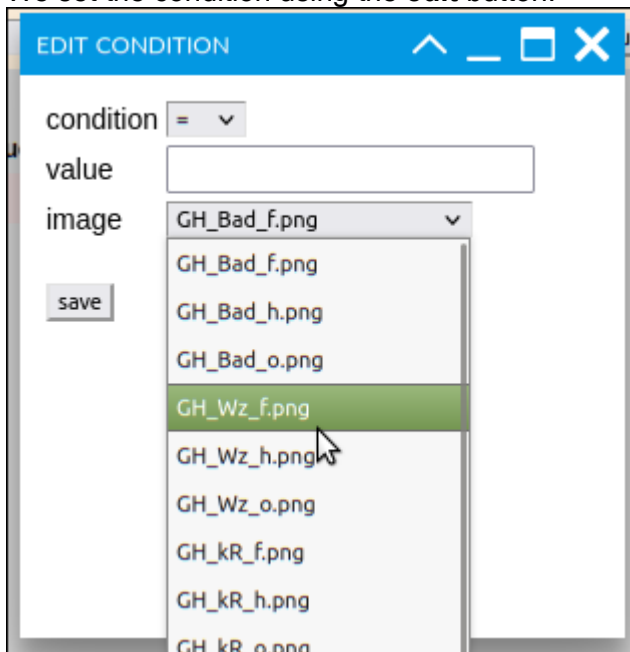


Fig. 95: Set condition

After selecting the graphic, it is displayed directly in the dialog.

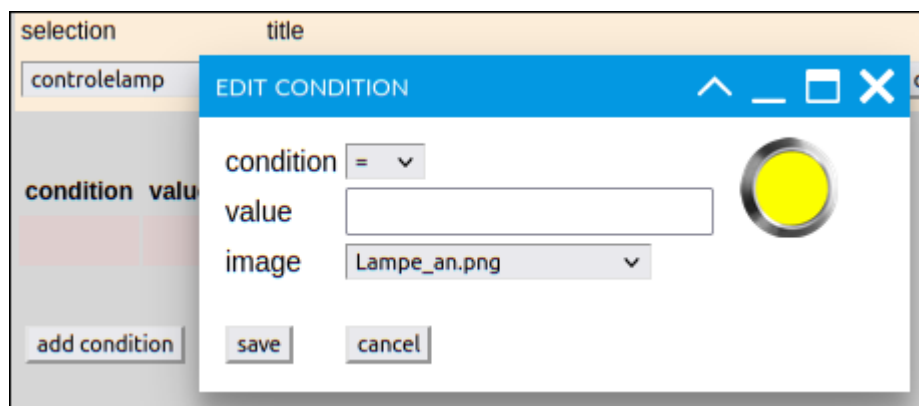


Fig. 96: Selected graphic in the dialog

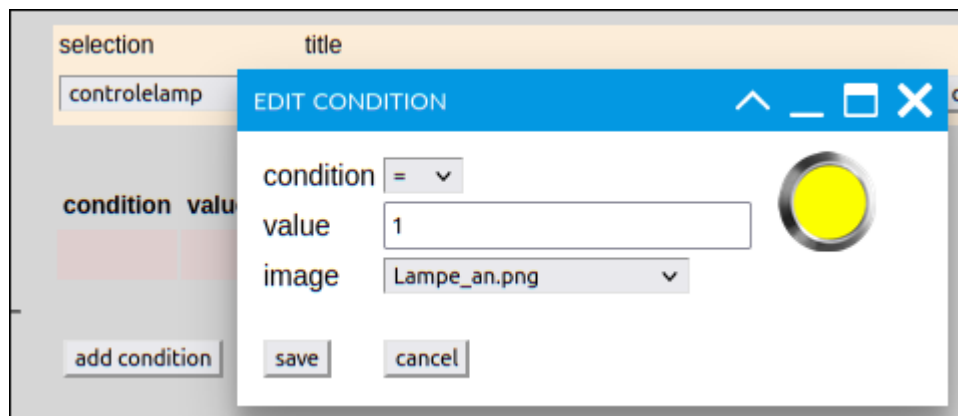


Fig. 97: The second condition

With that, our multistate would be complete. It can now be used in an image. If the tag assigned in the picture has the value 1, then the lit lamp is displayed, otherwise the switched-off lamp is displayed. Here is the full setup:

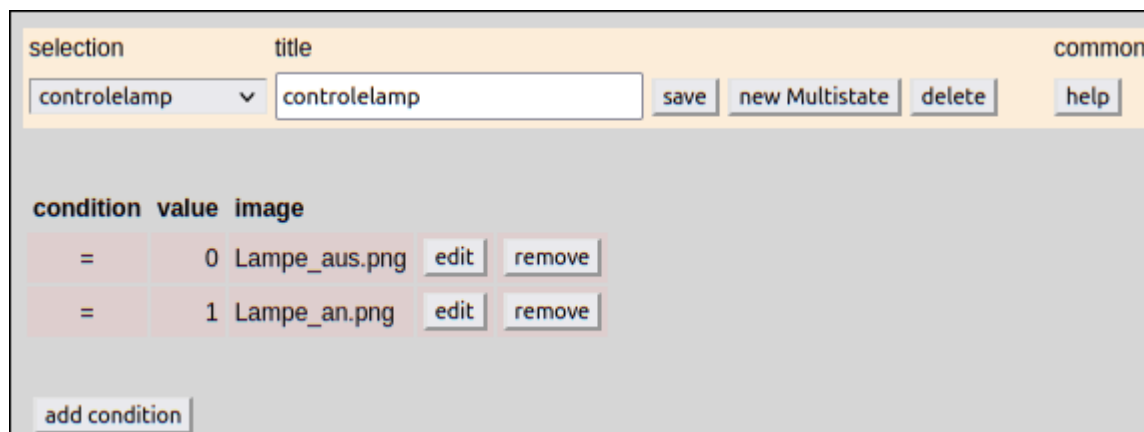


Fig. 98: Complete setting for the multistate control lamp.

2.3.3. Configure devices

This settings page is currently not being used (status 02/2022). This is the settings page for devices that read data from sensors and deliver it directly to the Datahistorian. The devices currently only exist as prototypes. Series production is still pending.

2.4. functions

Functions are available for both the DH and the unidb, which you can use in the settings dialog for elements and in your own JS code.

The DH spreadsheet functions have already been discussed in the user manual. Similar functions are also available for images. In images, you always have the choice of assigning a tag or an expression to

an element. If you want to display an expression, then a DH function is usually used in it. Here is an example of an expression using the DH function AW(Point_ID, timestamp):

The screenshot shows a window titled 'ELEMENT SETTINGS' with a blue header bar. Inside, there are several input fields: 'path', 'Tag_ID', 'units', and 'Tagname'. The 'expression' field is populated with the text '(AW(322,0s)*10) - 2147.5'. At the bottom of the dialog, there are two checkboxes: 'set link' and 'refresh', both of which are checked with green checkmarks.

Fig. 99: Printout in the Settings Element dialog.

Function	description
ZP(Point_ID, timestamp)	Returns the Unix timestamp of the value that was last written to the archive before the specified timestamp. The function accepts the time in the format YYYY-MM-DD hh:mm:ss as a time stamp. Alternatively, a relative time specification can also be made. Now always stands for the current time. If you want to use the current time minus one hour, then you now write - 1h as a relative time specification. Tags are specified as d, minutes as m, and seconds as s.
ZS(Point_ID, timestamp)	Returns the same result as before, but in a human-readable form in the format YYYY-MM-DD hh:mm:ss.
AW(Point_ID, timestamp)	Returns the last archive value whose timestamp is less than or equal to the specified timestamp.
intp(Point_ID,relativer Zeitpunkt)	Returns the most probable value of a point at the given time.
MW(Point_ID, start timestamp, end timestamp)	Returns the mean value of the point for the specified period, taking into account the temporal relationships.
akt(Point_ID)	Returns the latest value for the specified Point.