

phamos Code

Templates

In diesem Buch finden wir diverse Lösungen die von phamos für ERPNext entwickelt wurden. Dazu gehören Vorlagen für Custom Fields, Client Scripts und Server Scripts.

Alle Lösungen sind mit Einleitung beschrieben und ausreichend bebildert. Sie sollen als übertragbare Lösungen dienen und für interne und externe als Nachschlagewerk und Ideen für zukünftige Implementierungen genutzt werden.

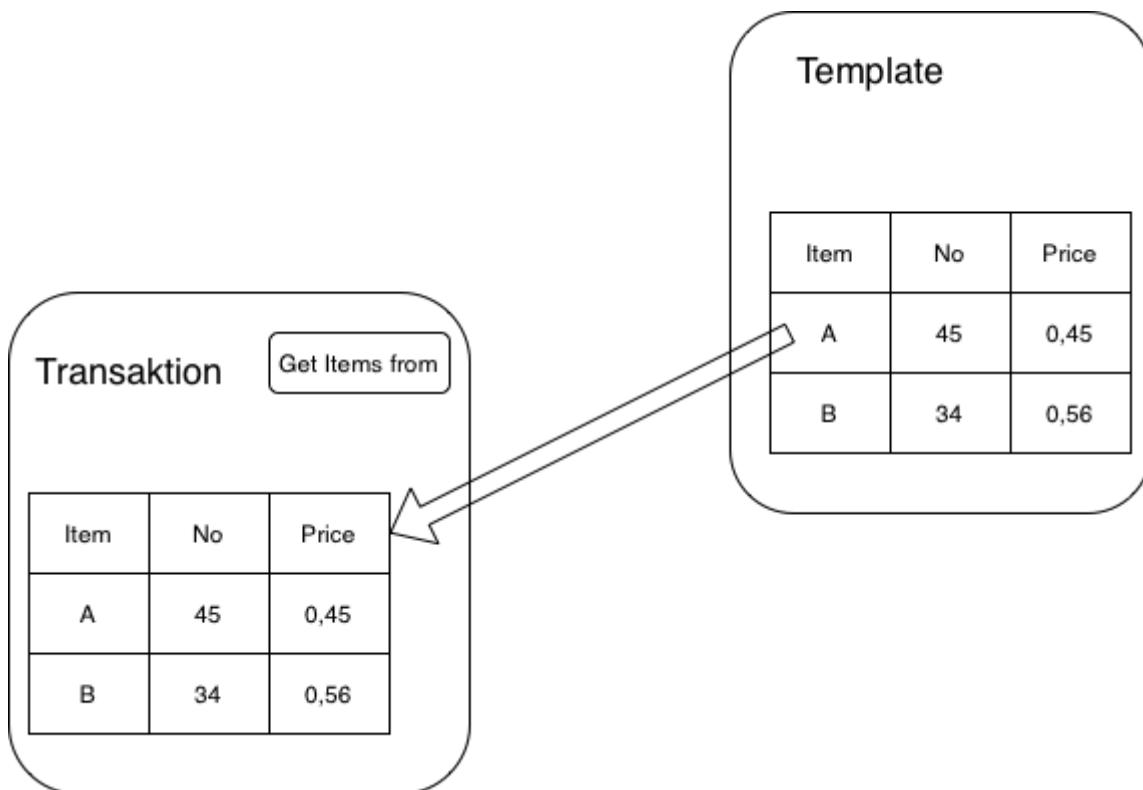
Aller hier gezeigter code wird unter der GPLv3 öffentlich gestellt.

- Tabelleninhalte aus einer Tabelle befüllen
- Datumsfelder mit Zeitspanne auf Beleg und Positionen
- Daten auf Mahnung automatisch berechnen
- Felder in Childtable addieren und Ergebnis auf Feld in Parentdocument / Sum column in child table and show total in parent field
- DocType Feld mit Wert aus einer Tabelle füllen
- Angemeldeten User in eine entsprechendes Feld setzen
- Buttons auf DocType um unterschiedliche Listen zu öffnen

Tabelleninhalte aus einer Tabelle befüllen

Einleitung

In dieser Anleitung finden wir ein Beispiel wie wir eine Tabelle in einer Transaktion mit Inhalten aus einer Tabelle einer Vorlage füllen kann.



DocType Template erstellen

Zunächst wollen wir ein Dokument erstellen in welchem wir unser Template festhalten, damit wir diese als Vorlage für unsere Transaktion verwenden können. Dazu erstellen wir hier Transaction Template.

Das Transaction Template nehmen wir in diesem Beispiel als Platzhalter. Es könnte z.B. auch eine Angebotsvorlage sein in welcher wir eine bestimmte Sammlung von Artikeln halten welche wir immer wieder zum Einsatz bringen wollen.

Im gezeigten Beispiel haben wir ein Template mit dem Namen Colors erstellen und mit den werden Red und Green.

Hier die wesentlichen Inhalte des DocTypes

The screenshot shows the configuration interface for a Transaction Template named 'Colors'. The top navigation bar includes links for 'DocType' and 'Transaction Template', a search bar, and a help icon. The main area is titled 'Transaction Template' and contains two sections: 'Felder' (Fields) and 'Bezeichnung' (Label).

Felder (Fields):

| No. | Bezeichnung | Typ | Name | Zwingen... | Optionen |
|-----|-------------------|---------|-------------------|--------------------------|---|
| 1 | Title | Daten | title | <input type="checkbox"/> | <input type="checkbox"/> Bearbeiten |
| 2 | Transaction Items | Tabelle | transaction_items | <input type="checkbox"/> | <input type="checkbox"/> Transaction Items <input type="checkbox"/> Bearbeiten |

A button 'Zeile hinzufügen' (Add Row) is located below the table.

Bezeichnung (Label):

Automatische Benennung: field:title

Beschreibung:

Naming Options:

1. field:[fieldname] - By Field
2. naming_series: - By Naming Series (field called naming_series must be present)
3. Prompt - Prompt user for a name
4. [series] - Series by prefix (separated by a dot); for example PRE.#####
5. format:EXAMPLE-(MM)morewords(fieldname1)-(fieldname2)-##### - Replace all braced words (fieldnames, date words (DD, MM, YY), series) with their value. Outside braces, any characters can be used.

Großschreibung:

Umbenennen zulassen

Und die dazu gehörige Untertabelle "Transaction Items"

DocType > Transaction Template Items

Suchen oder Befehl eingeben (Strg + G) | Hilfe | WS

Transaction Template Items

dies vor 3 Tag(en)

Administrator erstellte dies vor 3 Tag(en)

Modul *: Custom

Benutzerdefiniert?

Beta

Is Virtual

Ist Untertabelle
Untergeordnete Tabellen werden in anderen DocTypes als Raster angezeigt

Ist Baum
Baumstrukturen werden mit Nested Set implementiert

Editierbares Raster

Felder

| No. | Bezeichnung | Typ | Name | Zwing... | Optionen | Aktionen |
|-----|-------------|-------|---------|--------------------------|----------|--|
| 1 | Value 1 | Daten | value_1 | <input type="checkbox"/> | | <input checked="" type="checkbox"/> Bearbeiten |
| 2 | Value 2 | Daten | value_2 | <input type="checkbox"/> | | <input checked="" type="checkbox"/> Bearbeiten |

Zeile hinzufügen

Bezeichnung

Automatische Benennung

Beschreibung

Ist der DocType erstellt, erstellen wir direkt eine Instanz.

Suchen oder Befehl eingeben (Strg + G) | Hilfe | WS

Transaction Template

Filtern nach

- Zugewiesen zu
- Erstellt von
- Edit Filters
- Schlagworte
- Tags anzeigen
- Filter speichern

Name des Filters

Name

Name

6da55d2c0c

Filter | Zuletzt geändert am

1 von 1

- 3 d 0

List View | ... | + Hinzufügen Transaction Template

DocType Transaktion erstellen

Nun erstellen wir einen weiteren DocType der die eigentliche Transaktion repräsentiert. Diese kann z.B. ein Angebot sein in welches wir Artikel aus unserer Angebotsvorlage importieren. In diesem Beispiel importieren wir lediglich zwei Felder mit Werten vom Typ Data. Dem Beispiel folgend werden es der Wert Red und Green.

The screenshot shows a software interface for managing 'Transaction Document' entries. The top navigation bar includes icons for back, forward, search ('Suchen oder Befehl eingeben (Strg + G)'), help ('Hilfe'), and workspace ('WS'). The main title is 'Transaction Document'. On the right, there are buttons for 'List View' (grid icon), 'Search' (magnifying glass icon), 'More' (ellipsis icon), and a blue button '+ Hinzufügen Transaction Document' (Add Transaction Document). Below the title, there's a search bar and filter buttons for 'Filter' and 'Zuletzt geändert am' (Last changed). The left sidebar contains filtering options: 'Zugewiesen zu', 'Erstellt von', 'Edit Filters', 'Schlagworte', and 'Tags anzeigen'. A 'Filter speichern' (Save Filter) button is also present. The main content area displays a list of documents. The first item in the list is 'Name' (checkbox selected, heart icon, link to '302cf50b1f'), with a status '1 von 1' and a timestamp '16 m' ago. The bottom of the screen shows a navigation bar with icons for back, forward, and search, and a page number indicator '100 / 500'.

Der DocType ist wie folgt erstellt

Transaction Document

Gehen Sie zur Liste Transaction Document



speichern

 Schnelleingabe

Öffnen Sie ein Dialogfeld mit Pflichtfeldern, um schnell einen neuen Datensatz zu erstellen.

Felder

Felder

| No. | Bezeichnung | Typ | Name | Zwingend no... | Optionen | |
|-----|-------------------|---------|-------------------|--------------------------|-------------------|--|
| 1 | Title | Daten | title | <input type="checkbox"/> | | |
| 2 | Transaction Items | Tabelle | transaction_items | <input type="checkbox"/> | Transaction Items | |

Zeile hinzufügen

Bezeichnung

Automatische Benennung

field:title

Naming Options:

1. field:[fieldname] - By Field
2. naming_series: - By Naming Series (field called naming_series must be present)
3. Prompt - Prompt user for a name
4. [series] - Series by prefix (separated by a dot); for example PRE.#####
5. format:EXAMPLE-{MM}morewords{fieldname1}-{fieldname2}-{#####} - Replace all braced words (fieldnames, date words (DD, MM, YY), series) with their value. Outside braces, any characters can be used.

Großschreibung

 Umbenennen zulassen

Beschreibung

Dokumentationslink

URL für Dokumentation oder Hilfe

Dazu die Untertabelle Transaction Items**Transaction Items** Custom Beta Ist Untertabelle Is Virtual

Untergeordnete Tabellen werden in anderen DocTypes als Raster angezeigt

 Ist Baum

Baumstrukturen werden mit Nested Set implementiert

 Editierbares Raster**Felder**

Felder

| No. | Bezeichnung | Typ | Name | Zwing... | Optionen | |
|-----|-------------|-------|---------|--------------------------|----------|--|
| 1 | Value 1 | Daten | value_1 | <input type="checkbox"/> | | |
| 2 | Value 2 | Daten | value_2 | <input type="checkbox"/> | | |

Zeile hinzufügen

Bezeichnung

Automatische Benennung

Beschreibung

Naming Options:

Button in Transaktion erstellen

Nun wollen wir über einen Button die Möglichkeit bereit stellen, dass wir die Daten aus dem Template in die Transaction übertragen können. Dazu erstellen wir ein Client Script welches einen Button auf unser DocType Transaction bereitstellt über welchen wir per Dialog aus einem Template wählen können.

Nun haben wir alles eingebaut um eine neue Transaktion zu erstellen und dort Werte aus einem Template zu holen.

Dazu erstellen wir ein Custom Script

The screenshot shows the Odoo Client Script list view. The top navigation bar includes a search bar ('Suchen oder Befehl eingeben (Strg + G)'), a help icon ('Hilfe'), and a workspace switcher ('WS'). The main title is 'Clientskript'. On the left, there's a sidebar with filtering options: 'Zugewiesen zu', 'Erstellt von', 'Edit Filters', 'Schlagworte', and 'Tags anzeigen'. Below the sidebar, there's a 'Filter speichern' button with a placeholder 'Name des Filters'. The main area displays a table with two columns: 'Name' and 'Transaction Docur'. One record is listed:

| Name | Transaction Docur |
|--|---|
| <input type="checkbox"/> Name | Status • Aktiviert DocType Transaction Document 1 von 1 |
| <input type="checkbox"/> Transaction Document-Form | - 36 m 0 |

mit dem folgenden Code

```
// The fetch-from fields
var fields = [
    "value_1",
    "value_2"];

frappe.ui.form.on('Transaction Document', {
    refresh(frm) {
        var cur_frm = frm;
        console.log("Add button");
        frm.add_custom_button('Transaction Template', function () { frm.trigger('get_items') }, __("Get Items From"));
    },
    get_items(frm){
        start_dialog(frm);
    }
});
```

```

function start_dialog(frm) {
    let dialog = new frappe.ui.form.MultiSelectDialog({
        // Read carefully and adjust parameters
        doctype: "Transaction Template", // Doctype we want to pick up
        target: cur_frm,
        setters: {},
        },
        date_field: "creation", // "modified", "creation", ...
        get_query() {
            // MultiDialog Listfilter
            return {
                filters: { }
            };
        },
        action(selections) {
            var name = selections[0];
            frappe.db.get_doc("Transaction Template", name) // Again, the Doctype we want to pick up
            .then(doc => {
                // Copy the items from the template and paste them into the cur_frm
                for(var n = 0; n < doc.transaction_items.length; n++){
                    var item=doc.transaction_items[n];

                    // Copy-Paste Operation
                    var child = {};
                    for(var m = 0; m < fields.length; m++){
                        child[fields[m]] = item[fields[m]];
                    }
                    cur_frm.add_child("transaction_items",child);
                    cur_frm.refresh_fields("transaction_items"); // Refresh Tabelle
                }
            });
        }
    });
}

```

Demo

Hier sehen wir wie wir eine neue Transaktion erstellen und diese gleich mit den Werten einer Vorlage füttern.

The screenshot shows a software interface for managing transaction documents. At the top, there's a navigation bar with a blue 'E' icon, the text 'Transaction Document', a search bar ('Suchen oder Befehl eingeben (Strg + G)'), a help button ('Hilfe'), and a workspace switch ('WS'). Below the navigation is a title bar with 'Transaction Document' and buttons for 'List View', 'Filter', 'Zuletzt geändert am', and '+ Hinzufügen Transaction Document'.

On the left, there's a sidebar titled 'Filtern nach' with dropdown menus for 'Zugewiesen zu', 'Erstellt von', 'Edit Filters', 'Schlagworte', and 'Tags anzeigen'. Below this is a 'Filter speichern' section with a text input field 'Name des Filters'.

The main area is a table titled 'Name' with two rows of data:

| <input type="checkbox"/> | >Name | 2 von 2 | |
|--------------------------|-------------------------|---------|---|
| <input type="checkbox"/> | Transaction with Colors | - 8 m | 0 |
| <input type="checkbox"/> | 302cf50b1f | - 24 m | 0 |

At the bottom of the table are buttons for '20', '100', and '500' items per page.

Hilfe Artikel

- DocType erstellen
- Client Script erstellen

Datumsfelder mit Zeitspanne auf Beleg und Positionen

Einleitung

Auf dieser Seite sehen wir, wie ein Datums Feld "Ende" in Abhängigkeit zu einem Datumsfeld "Anfang" befüllt wird und diese Daten auf die Artikelpositionen vererbt werden.

Datumsfeld "Ende" ein Jahr nach "Beginn"

```
frappe.ui.form.on('Sales Order', {
    performance_period_start: function(frm, cdt,cdn){
        if (frm.doc.performance_period_start) {
            var end_date = frappe.datetime.add_days(frm.doc.performance_period_start, 364);
            frappe.model.set_value(cdt,cdn,"performance_period_end",end_date);
            for (var i =0; i < frm.doc.items.length; i++){
                frm.doc.items[i].start_date = frm.doc.performance_period_start;
                frm.doc.items[i].end_date = frm.doc.performance_period_end;
            }
        }
    });
});
```

Daten auf Positionstabelle übertragen

```
frappe.ui.form.on('Sales Order Item', {
    start_date: function(frm, cdt,cdn){
```

```
var d = locals[cdt][cdn];
if (d.start_date) {
    var end_date = frappe.datetime.add_days(d.start_date, 364);
    frappe.model.set_value(cdt,cdn,"end_date",end_date);
}
});

});
```

Daten auf Mahnung automatisch berechnen

```
frappe.ui.form.on('Dunning', {
    validate(frm) {
        // your code here
        frm.set_value('sub_sum',frm.doc.outstanding_amount+frm.doc.interest_amount)
    }
},
refresh(frm) {
    if (frm.doc.docstatus === 1 && !frm.doc.sub_sum) {
        frm.set_value('sub_sum',frm.doc.outstanding_amount+frm.doc.interest_amount)
    }
    let due_date = new Date(frm.doc.posting_date)
    due_date.setDate(due_date.getDate() + 14);
    frm.set_value('dunning_due_date',due_date)
}
})
```

Felder in Childtable addieren und Ergebnis auf Feld in Parentdocument / Sum column in child table and show total in parent field

Um die Felder in einem Childtable zu addieren und das Ergebnis auf ein Feld im Parenttable zu übertragen kann folgendes Script benutzt werden:

```
frappe.ui.form.on("Name des Childtables", {  
    zu addierendes Feld auf CT:function(frm, cdt,cdn){  
        var d = locals[cdt][cdn];  
        var total = 0;  
        frm.doc.Childtablename auf Parentdocument.forEach(function(d) { total += d.hours; });  
        frm.set_value("Feld auf Parentdoc für Ergebnis", total);  
        refresh_field("Feld auf Parentdoc für Ergebnis");  
    },  
    Childtablename auf Parentdocument_remove:function(frm, cdt,cdn){  
        var d = locals[cdt][cdn];  
        var total = 0;  
        frm.doc.Childtablename auf Parentdocument.forEach(function(d) { total += d.zu addierendes Feld auf CT; });  
        frm.set_value("Feld auf Parentdoc für Ergebnis", total);  
        refresh_field("Feld auf Parentdoc für Ergebnis");  
    }  
});
```

Beispiel:

Name Childtable: Stunden Monteure Projekte

zu addierende Felder auf CT : hours

Childtablename auf Parentdoctype: time_logs

Feld auf Parentdoc für Ergebnis: arbeitszeit_gesamt

```
“ frappe.ui.form.on("Stunden Monteure Projekte", {  
    hours:function(frm, cdt,cdn){  
        var d = locals[cdt][cdn];  
        var total = 0;  
        frm.doc.time_logs.forEach(function(d) { total += d.hours; });  
        frm.set_value("arbeitszeit_gesamt", total);  
        refresh_field("arbeitszeit_gesamt");  
    },  
    time_logs_remove:function(frm, cdt,cdn){  
        var d = locals[cdt][cdn];  
        var total = 0;  
        frm.doc.time_logs.forEach(function(d) { total += d.hours; });  
        frm.set_value("arbeitszeit_gesamt", total);  
        refresh_field("arbeitszeit_gesamt");  
    }  
});
```

Zeile bearbeiten #1



Unterhalb einfügen Oberhalb einfügen Duplizieren Verschieben ▾

Projekt

Std

☒ Tastenkombinationen: Strg + Auf . Strg + Ab . ESC

Unterhalb einfügen

Zeile hinzufügen

Arbeitszeit gesamt

0

DocType Feld mit Wert aus einer Tabelle füllen

In diesem Beispiel wollen wir ein Feld auf DocType-Ebene mit dem Namen "anzahl_sm4" mit einem Anzahl füllen. Das Feld ist ein Feld vom Typ Integer und soll mit eben einer solchen Zahl gefüllt werden.

Die Zahl befindet sich in einer Tabelle. In der Tabelle kann es aber mehrere Einträge geben, was es erforderlich macht, dass wir zusätzlich noch bestimmen bei welchem Referenz-Wert die Anzahl übernommen werden soll. In unserem Beispiel ist der Referenzwert der Artikelname (item_code) "Batteriemodul sM4".

Hier das Script dazu

```
frappe.ui.form.on('Sales Order', {
    refresh: function(frm) {
        var anzahl_sm4 = 0;
        $.each(frm.doc.items || [], function(i, item) {
            if (item.item_code === "Batteriemodul sM4") {
                anzahl_sm4 += item.qty;
            }
        });
        frm.set_value("anzahl_sm4", anzahl_sm4);
    }
});
```

Dieses Script wurde mit der Hilfe von ChatGPT erstellt!



I need a new script on Sales Order. This script shall go through the table "items" and put the value of "qty" of that table into a custom field "anzahl_sm4" of the docType Sales Order. It should only consider the qty from the row where the value of the field item_code is "Batteriemodul sM4"



Certainly, here is a sample custom script that will iterate through the "items" table of the Sales Order doctype, check if the "item_code" field is "Batteriemodul sM4", and if so, set the value of a custom field named "anzahl_sm4" in the Sales Order header to the value of the "qty" field in the corresponding row:



```
js
frappe.ui.form.on('Sales Order', {
    refresh: function(frm) {
        var anzahl_sm4 = 0;
        $.each(frm.doc.items || [], function(i, item) {
            if (item.item_code === "Batteriemodul sM4") {
                anzahl_sm4 += item.qty;
            }
        });
        frm.set_value("anzahl_sm4", anzahl_sm4);
    }
});
```

Copy code

To use this script, you can paste it into the "Custom Script" field on the Sales Order's customization page.

When a user opens or saves a Sales Order document, this script will iterate through the items table and sum up the quantity of any rows where the item_code field is "Batteriemodul sM4". It will then set the value of the "anzahl_sm4" field in the Sales Order header to this sum.

Regenerate response

Angemeldeten User in eine entsprechendes Feld setzen

Mit dem folgenden Script können wir beim erstellen eines neuen Dokuments immer den angemeldeten User in das Feld `user` setzen. Ändern wir es nach dem speichern wird es nicht beim nächsten laden überschrieben. Dieser code hilft bei einer schnellen Dateneingabe.

```
frappe.ui.form.on('Session Standard User', {
  onload: function(frm) {
    if (!frm.doc.user) {
      frm.set_value('user', frappe.session.user);
    }
  }
});
```

Buttons auf DocType um unterschiedliche Listen zu öffnen

Einleitung

In diesem Beispiel versuchen wir Buttons auf unserem DocType Sales Opportunity zu erstellen hinter welchen jeweils die gefilterte Listenansicht einer bestimmten Verknüpfung, hier Quoation, zu finden ist. Die Buttons sollen für die Userin des Systems ansprechend und verständlich sein. Hier ein paar Beispiele welche mit Hilfe von ChatGPT (immer wieder beeindruckend!) erstellt wurden.

Feel free to copy and use GPLv3 ❤

Teilweise ist der Code noch überladen, er könnte weiter optimiert werden.

Tabelle mit blauen Buttons

The screenshot shows a Sales Opportunity record with ID 85262402b2. The left sidebar contains links for Assigned To, Attachments (with an 'Attach File' button), Shared With, Tags (with an 'Add a tag ...' button), and social sharing icons for 0 likes and 0 follows. The top navigation bar includes a back arrow, a search bar ('Search or type a command (Ctrl + G)'), a help icon, and a save button. The main content area has tabs for Sales, Delivery, and Accounting. Under Sales, there are three Quotations (one Draft, one Open) with an 'Open in List View' button. Under Delivery, there is one Delivery Note with a '+' button. Under Accounting, there is one Sales Invoice with a '+' button. A table below lists the quotations with columns for Type, Status, Amount, and Action (the 'Open in List View' button).

| Type | Status | Amount | Action |
|-----------|--------|--------|-----------------------------------|
| Quotation | Draft | 1 | Open in List View |
| Quotation | Open | 2 | Open in List View |

The Script

```
frappe.ui.form.on('Sales Opportunity', {
    refresh: function(frm) {
        // Get the linked Quotations
        frappe.call({
            method: 'frappe.client.get_list',
            args: {
                doctype: 'Quotation',
                filters: {
                    sales_opportunity: frm.doc.name
                },
                fields: ['status']
            },
            callback: function(response) {
                var data = response && response.message;

                // Count the Quotations by status
                var counts = {};
                if (data && data.length > 0) {
                    data.forEach(function(row) {
                        if (row.status) {
                            if (counts[row.status]) {
                                counts[row.status]++;
                            } else {
                                counts[row.status] = 1;
                            }
                        }
                    });
                }

                // Create or update the visual section with the table
                var section = frm.dashboard.add_section__('Quotations');
                var html = `<style>
                    .sales-opportunity-table {
                        font-size: 70%;
                    }
                </style>

```

```

<table class="table table-bordered sales-opportunity-table">
  <thead>
    <tr>
      <th>Type</th>
      <th>Status</th>
      <th>Amount</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>`;

if (Object.keys(counts).length > 0) {
  Object.keys(counts).forEach(function(status) {
    html += `<tr>
      <td>Quotation</td>
      <td>${status}</td>
      <td>${counts[status]}</td>
      <td><button class="btn btn-primary btn-sm view-opportunity-btn" data-
status="${status}">Open in List View</button></td>
    </tr>`;
  });
} else {
  html += `<tr>
    <td>Opportunity</td>
    <td colspan="3" align="center">No Quotations</td>
  </tr>`;
}

html += `</tbody>
</table>`;

section.html(html);

// Add click event to the button
section.find('.view-opportunity-btn').on('click', function() {
  var status = $(this).data('status');
  viewQuotationList(status, frm.doc.name);
});
}

```

```

        });
    }
});

function viewQuotationList(status, salesOpportunity) {
    // Redirect to the Quotation List view with the status filter applied
    frappe.set_route('List', 'Quotation', { 'status': status, 'sales_opportunity': salesOpportunity });
}

```

Tabelle mit Button "List View" aus der Listensicht

The screenshot shows a Frappe application interface for a Sales Opportunity record. The record ID is 85262402b2. The left sidebar contains links for 'Assigned To', 'Attachments' (with an 'Attach File' button), 'Shared With', and 'Tags'. The main area has tabs for 'Details' and 'Quotations'. Under 'Connections', there are two entries:

| Type | Status | Amount | Action |
|-----------|--------|--------|--------------------------------------|
| Quotation | Draft | 1 | <button>View Quotation List</button> |
| Quotation | Open | 2 | <button>View Quotation List</button> |

The Script

```

frappe.ui.form.on('Sales Opportunity', {
    refresh: function(frm) {
        // Get the linked Quotations
        frappe.call({
            method: 'frappe.client.get_list',
            args: {
                doctype: 'Quotation',
                filters: {
                    sales_opportunity: frm.doc.name
                },
                fields: ['status']
            }
        })
    }
})

```

```

    },
    callback: function(response) {
        var data = response && response.message;

        // Count the Quotations by status
        var counts = {};
        if (data && data.length > 0) {
            data.forEach(function(row) {
                if (row.status) {
                    if (counts[row.status]) {
                        counts[row.status]++;
                    } else {
                        counts[row.status] = 1;
                    }
                }
            });
        }

        // Create or update the visual section with the table
        var section = frm.dashboard.add_section(__('Quotations'));
        var html = `<style>
.sales-opportunity-table {
    font-size: 70%;
}
</style>
<table class="table table-bordered sales-opportunity-table">
    <thead>
        <tr>
            <th>Type</th>
            <th>Status</th>
            <th>Amount</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>`;

        if (Object.keys(counts).length > 0) {
            Object.keys(counts).forEach(function(status) {
                html += `<tr>
                    <td>${status}</td>
                    <td>${counts[status]}</td>
                    <td>${(counts[status] * 100).toFixed(2)}%</td>
                    <td>${button}</td>
                </tr>`;
            });
        }
    }
}

```

```

        <td>Quotation</td>
        <td>${status}</td>
        <td>${counts[status]}</td>
        <td><button class="btn btn-secondary btn-sm view-opportunity-btn" data-
status="${status}" data-doctype="Quotation"><i class="fa fa-list"></i> View Quotation
List</button></td>
        </tr>`;
    });
} else {
    html += `<tr>
        <td>Opportunity</td>
        <td colspan="3" align="center">No Quotations</td>
    </tr>`;
}

html += `</tbody>
</table>`;

section.html(html);

// Add click event to the button
section.find('.view-opportunity-btn').on('click', function() {
    var status = $(this).data('status');
    var doctype = $(this).data('doctype');
    viewQuotationList(status, doctype, frm.doc.name);
});

}
});

});

function viewQuotationList(status, doctype, salesOpportunity) {
    // Redirect to the Quotation List view with the status and sales opportunity filter applied
    frappe.set_route('List', doctype, { 'status': status, 'sales_opportunity': salesOpportunity });
}

```

Tabelle mit farbigen Buttons

The Script

```

frappe.ui.form.on('Sales Opportunity', {
    refresh: function(frm) {
        // Get the linked Quotations
        frappe.call({
            method: 'frappe.client.get_list',
            args: {
                doctype: 'Quotation',
                filters: {
                    sales_opportunity: frm.doc.name
                },
                fields: ['status']
            },
            callback: function(response) {
                var data = response && response.message;

                // Count the Quotations by status
                var counts = {};
                if (data && data.length > 0) {
                    data.forEach(function(row) {
                        if (row.status) {
                            if (counts[row.status]) {
                                counts[row.status]++;
                            } else {
                                counts[row.status] = 1;
                            }
                        }
                    })
                }
            }
        })
    }
})

```

```

        }
    });
}

// Create or update the visual section with the table
var section = frm.dashboard.add_section__('Quotations');
var html = `<style>
    .sales-opportunity-table {
        font-size: 100%;
    }
    .status-draft {
        background-color: #f2f2f2;
        color: #495057;
    }
    .status-open {
        background-color: #28a745;
        color: #fff;
    }
    .status-submitted {
        background-color: #ffc107;
        color: #fff;
    }
    .status-accepted {
        background-color: #17a2b8;
        color: #fff;
    }
    .status-rejected {
        background-color: #dc3545;
        color: #fff;
    }
</style>
<table class="table table-bordered sales-opportunity-table">
    <thead>
        <tr>
            <th>Quotations</th>
        </tr>
    </thead>
    <tbody>
        <tr>`;

```

```

// Add buttons for each status

var statusOrder = ['Draft', 'Open', 'Submitted', 'Accepted', 'Rejected'];

statusOrder.forEach(function(status) {
    var statusClass = getStatusClass(status);
    var amount = counts[status] || 0;
    var buttonText = `Open List View with <b>${amount}</b> Quotations in <i>${status}</i>`;

    html += `<td><button class="btn btn-sm view-opportunity-btn ${statusClass}" data-
status="${status}" data-doctype="Quotation">${buttonText}</button></td>`;
});

html += `</tr>
</tbody>
</table>`;

section.html(html);

// Add click event to the buttons

section.find('.view-opportunity-btn').on('click', function() {
    var status = $(this).data('status');
    var doctype = $(this).data('doctype');
    viewQuotationList(status, doctype, frm.doc.name);
});

}

});

}

function viewQuotationList(status, doctype, salesOpportunity) {
// Redirect to the Quotation List view with the status and sales opportunity filter applied
frappe.set_route('List', doctype, { 'status': status, 'sales_opportunity': salesOpportunity });
}

function getStatusClass(status) {
// Define the class for each status
var statusClass = {
'Draft': 'status-draft',
'Open': 'status-open',

```

```

        'Submitted': 'status-submitted',
        'Accepted': 'status-accepted',
        'Rejected': 'status-rejected'
    };

    return statusClass[status] || "";
}

```

Using Badges instead of buttons to save space!

The screenshot shows a Sales Opportunity record with the name "85262402b2". On the left, there are sections for "Assigned To", "Attachments" (with an "Attach File" button), and "Shared With". Below these are "Tags" and a "Follow" button. The main area displays "Connections" for Sales, Delivery, and Accounting. Under Sales, there are 3 Quotations and 3 Sales Orders. Under Delivery, there is 1 Delivery Note. Under Accounting, there is 1 Sales Invoice. To the right, a table lists "Document Type" and "Documents" with corresponding badge counts and links. For Quotations, there are 1 in Draft and 1 in Open. For Sales Orders, there are 1 in Completed, 1 in To Deliver and Bill, and 1 in Closed.

| Document Type | Documents |
|---------------|---|
| Quotation | Open (1) Quotations in Draft Open (1) Quotations in Open |
| Sales Order | Open (1) Sales Orders in Completed Open (1) Sales Orders in To Deliver and Bill Open (1) Sales Orders in Closed |

The Script

```

frappe.ui.form.on('Sales Opportunity', {
    refresh: function(frm) {
        // Get the linked Quotations
        frappe.call({
            method: 'frappe.client.get_list',
            args: {
                doctype: 'Quotation',
                filters: {
                    sales_opportunity: frm.doc.name
                }
            }
        })
    }
})

```

```
},
fields: ['status']
},
callback: function(response) {
var data = response && response.message;

// Count the Quotations by status
var quotationCounts = {};
if (data && data.length > 0) {
data.forEach(function(row) {
if (row.status) {
if (quotationCounts[row.status]) {
quotationCounts[row.status]++;
} else {
quotationCounts[row.status] = 1;
}
}
});
}

// Get the linked Sales Orders
frappe.call({
method: 'frappe.client.get_list',
args: {
doctype: 'Sales Order',
filters: {
sales_opportunity: frm.doc.name
},
fields: ['status']
},
callback: function(response) {
var data = response && response.message;

// Count the Sales Orders by status
var orderCounts = {};
if (data && data.length > 0) {
data.forEach(function(row) {
if (row.status) {
if (orderCounts[row.status]) {
```

```
        orderCounts[row.status]++;
    } else {
        orderCounts[row.status] = 1;
    }
}
});

}

// Create or update the visual section with the table
var section = frm.dashboard.add_section(__('Sales Documents'));
var html = `<style>
    .sales-documents-table {
        font-size: 100%;
    }
    .badge-closed {
        background-color: green;
        color: #fff;
    }
    .badge-draft {
        background-color: red;
        color: #495057;
    }
    .badge-overdue {
        background-color: red;
        color: #495057;
    }
    .badge-open {
        background-color: orange;
        color: #fff;
    }
    .status-submitted {
        background-color: #ffc107;
        color: #fff;
    }
    .status-accepted {
        background-color: #17a2b8;
        color: #fff;
    }
    .status-rejected {
```

```

        background-color: #dc3545;
        color: #fff;
    }

    .badge-to_deliver_and_bill {
        background-color: orange;
        color: #fff;
    }

}

</style>

<table class="table table-bordered sales-documents-table">

    <thead>
        <tr>
            <th>Document Type</th>
            <th>Documents</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Quotation</td>
            <td>`;

// Add badges for Quotations
var quotationStatusOrder = ['Draft', 'Open', 'Submitted', 'Accepted', 'Rejected'];
quotationStatusOrder.forEach(function(status) {
    var amount = quotationCounts[status] || 0;
    if (amount > 0) {
        var badgeClass = getStatusBadgeClass(status);
        var badgeText = `Open (${amount}) Quotations in ${status}`;
        var linkUrl = getQuotationListUrl(status, frm.doc.name);
        html += `<a class="${badgeClass}" href="${linkUrl}">${badgeText}</a>`;
    }
});

html += `</td>
</tr>
<tr>
    <td>Sales Order</td>
    <td>`;

// Add badges for Sales Orders

```

```

var orderStatusOrder = ['Draft', 'Submitted', 'Completed', 'To Deliver and Bill', 'Overdue',
'Closed'];

orderStatusOrder.forEach(function(status) {
    var amount = orderCounts[status] || 0;
    if (amount > 0) {
        var badgeClass = getStatusBadgeClass(status);
        var badgeText = `Open (${amount}) Sales Orders in ${status}`;
        var linkUrl = getSalesOrderListUrl(status, frm.doc.name);
        html += `<a class="badge ${badgeClass}" href="${linkUrl}">${badgeText}</a>`;
    }
});

html += `
</tr>
</tbody>
</table>`;

section.html(html);
}

});

}

});

function getStatusBadgeClass(status) {
// Define the class for each status badge
var statusBadgeClass = {
    'Closed': 'badge-closed',
    'Draft': 'badge-draft',
    'Open': 'badge-open',
    'Overdue': 'badge-overdue',
    'Submitted': 'badge-warning',
    'Accepted': 'badge-info',
    'Rejected': 'badge-danger',
    'Completed': 'badge-success',
    'To Deliver and Bill': 'badge-to_deliver_and_bill'
};

}

```

```
return statusBadgeClass[status] || 'badge-secondary';

}

function getQuotationListUrl(status, salesOpportunity) {
    // Generate the Quotation List URL with the status and sales opportunity filter
    var url = frappe.urllib.get_base_url();
    url +=
        `/app/list/Quotation?status=${encodeURIComponent(status)}&sales_opportunity=${encodeURIComponent(salesOpportunity)}`;
    return url;
}

function getSalesOrderListUrl(status, salesOpportunity) {
    // Generate the Sales Order List URL with the status and sales opportunity filter
    var url = frappe.urllib.get_base_url();
    url +=
        `/app/list/Sales%20Order?status=${encodeURIComponent(status)}&sales_opportunity=${encodeURIComponent(salesOpportunity)}`;
    return url;
}
```

Accordion

The screenshot shows the ERPNext interface for creating a new Sales Opportunity. The top navigation bar includes the title "Sales Opportunity" and reference number "85262402b2". A search bar at the top right contains the placeholder "Search or type a command (Ctrl + G)". On the far right, there are "Help" and "A" buttons.

The main content area displays the opportunity details:

- Connections:** Shows Sales, Delivery, and Accounting sections with counts of Quotation (3), Delivery Note (1), and Sales Invoice (1) respectively.
- Quotation:** The current active tab, showing 1 Quotation in Draft and 1 Quotation in Open.
- Sales Order:** Another tab in the Connections section.
- Title:** ERPNext Projekt Personalwesen
- Customer:** CUST-2023-00038
- Assignment:** wolfram.schmidt@phamos.eu
- Potential Date:** 01-06-2023

On the right side, there is a rich text editor toolbar with various styling options like bold, italic, underline, and alignment. The text area contains the note: "mit diesem Kunden könnten wir ein neues ERP Projekt machen".

On the left sidebar, there are several collapsed sections: Assigned To, Attachments (with an "Attach File" button), Shared With, Tags (with an "Add a tag ..." button), and a social-like section showing 0 likes and 0 comments, and a "FOLLOW" button.

At the bottom left, there are notifications: "You edited this 4 hours ago" and "You created this 4 hours ago".

The Script

```
frappe.ui.form.on('Sales Opportunity', {
    refresh: function(frm) {
        // Get the linked Quotations
        frappe.call({
            method: 'frappe.client.get_list',
            args: {
                doctype: 'Quotation',
                filters: {
                    sales_opportunity: frm.doc.name
                },
                fields: ['status']
            }
        });
    }
});
```

```

callback: function(response) {
    var data = response && response.message;

    // Count the Quotations by status
    var quotationCounts = {};
    if (data && data.length > 0) {
        data.forEach(function(row) {
            if (row.status) {
                if (quotationCounts[row.status]) {
                    quotationCounts[row.status]++;
                } else {
                    quotationCounts[row.status] = 1;
                }
            }
        });
    }

    // Get the linked Sales Orders
    frappe.call({
        method: 'frappe.client.get_list',
        args: {
            doctype: 'Sales Order',
            filters: {
                sales_opportunity: frm.doc.name
            },
            fields: ['status']
        },
        callback: function(response) {
            var data = response && response.message;

            // Count the Sales Orders by status
            var orderCounts = {};
            if (data && data.length > 0) {
                data.forEach(function(row) {
                    if (row.status) {
                        if (orderCounts[row.status]) {
                            orderCounts[row.status]++;
                        } else {
                            orderCounts[row.status] = 1;
                        }
                    }
                });
            }
        }
    });
}

```

```
        }

    }

});

}

// Create or update the visual section with the accordion
var section = frm.dashboard.add_section(__('Sales Documents'));

var accordionHtml = `<style>

.sales-accordion {
    font-size: 100%;
}

.sales-accordion .accordion-title {
    background-color: #f5f5f5;
    color: #333;
    cursor: pointer;
    padding: 10px;
    border: none;
    text-align: left;
    outline: none;
    font-weight: bold;
    transition: background-color 0.3s;
}

.sales-accordion .accordion-content {
    padding: 10px;
    display: none;
    overflow: hidden;
    background-color: #fff;
    border: 1px solid #e7e7e7;
}

.sales-accordion .accordion-content a {
    display: block;
    margin-bottom: 5px;
}

.sales-accordion .accordion-content a:hover {
    text-decoration: underline;
}

.sales-accordion .accordion-title.active {
    background-color: #ccc;
}
```

```

    .sales-accordion .accordion-content.active {
        display: block;
    }

```

```

</style>
<div class="sales-accordion">`;
```

```

// Add accordion for Quotations
var quotationStatusOrder = ['Draft', 'Open', 'Submitted', 'Accepted', 'Rejected'];
accordionHtml += `<button class="accordion-title">Quotation</button>
<div class="accordion-content">`;
```

```

quotationStatusOrder.forEach(function(status) {
    var amount = quotationCounts[status] || 0;
    if (amount > 0) {
        var badgeClass = getStatusBadgeClass(status);
        var badgeText = `Open (${amount}) Quotations in ${status}`;
        var linkUrl = getQuotationListUrl(status, frm.doc.name);
        accordionHtml += `<a class="badge ${badgeClass}" href="${linkUrl}">${badgeText}</a>`;
    }
});
```

```

accordionHtml += `</div>`;
```

```

// Add accordion for Sales Orders
var orderStatusOrder = ['Draft', 'Submitted', 'Completed', 'To Deliver and Bill', 'Overdue',
'Closed'];
accordionHtml += `<button class="accordion-title">Sales Order</button>
<div class="accordion-content">`;
```

```

orderStatusOrder.forEach(function(status) {
    var amount = orderCounts[status] || 0;
    if (amount > 0) {
        var badgeClass = getStatusBadgeClass(status);
        var badgeText = `Open (${amount}) Sales Orders in ${status}`;
        var linkUrl = getSalesOrderListUrl(status, frm.doc.name);
        accordionHtml += `<a class="badge ${badgeClass}" href="${linkUrl}">${badgeText}</a>`;
    }
});
```

```

    });

    accordionHtml += `
```

```
// Generate the Quotation List URL with the status and sales opportunity filter
var url = frappe.urllib.get_base_url();
url +=
`/app/list/Quotation?status=${encodeURIComponent(status)}&sales_opportunity=${encodeURIComponent(salesOpportunity)}`;
return url;
}

function getSalesOrderListUrl(status, salesOpportunity) {
// Generate the Sales Order List URL with the status and sales opportunity filter
var url = frappe.urllib.get_base_url();
url +=
`/app/list/Sales%20Order?status=${encodeURIComponent(status)}&sales_opportunity=${encodeURIComponent(salesOpportunity)}`;
return url;
}
```