

2. Die Klasse

VonmaehlenOrderCreationClass

Die Hauptklasse, die in diesem Skript verwendet wird, ist `VonmaehlenOrderCreationClass`. Diese Klasse enthält Methoden zur Erstellung von Bestellungen und unterstützende Hilfsfunktionen.

2.1. Methoden

`createOrder()`: Diese Methode initiiert den Prozess zur automatisierten Auftragsanlage. Sie überprüft vorhandene Bestellungen für eine bestimmte Verkaufschance, speichert die Verkaufschance, ruft erforderliche Daten ab, erstellt Bestellungen basierend auf den Verkaufschancen und öffnet die erstellten Bestellungen in Business Central.

```
async createOrder() {  
    let parentAccountId = this.getParentAccountId();  
    let opportunityId = getCurrentEntityId();  
    let createdOrders = [];  
  
    // Check if there are any orders already  
    await showProgressIndicator("Voraussetzungen prüfen...");  
    let orders = await retrieveMultipleRecords("salesorder", `?$filter=_opportunityid_value eq ${opportunityId}`);  
  
    if (orders.entities.length > 0) {
```

```

let orderBaseData = {
  "name": `${opportunity["nps_id"]} | ${i} | ${productNumber} | ${opportunity[`${cre56_logo_name_${i}}`]}`,
  "customerid_account@odata.bind": "/accounts(" + account["accountid"] + ")",
  "opportunityid@odata.bind": "/opportunities(" + opportunityId + ")",
  "pricelevelid@odata.bind": "/pricelevels(" + priceListId + ")",
  "ownerid@odata.bind": "/systemusers(" + opportunity["_ownerid_value"] + ")",
  "cre56_scm_owner@odata.bind": "/systemusers(" + opportunity["_cre56_scm_owner_value"] + ")",
  "cre56_customer_logo": opportunity[`${cre56_logo_name_${i}}`],
  "billto_contactname": account["address1_primarycontactname"],
  "billto_telephone": account["address1_telephone1"],
  "billto_fax": account["address1_fax"],
  "billto_name": account["address1_name"],
  "emailaddress": account["emailaddress1"],
  "billto_line1": account["address1_line1"],
  "billto_line2": account["address1_line2"],
  "billto_line3": account["address1_line3"],
  "billto_city": account["address1_city"],
  "billto_stateorprovince": account["address1_stateorprovince"],
  "billto_postalcode": account["address1_postalcode"],
  "billto_country": account["address1_country"],
  "transactioncurrencyid": account["transactioncurrencyid"],
  "shippingmethodcode": account["address1_shippingmethodcode"],
  "paymenttermscode": account["paymenttermscode"],
  "freighttermscode": account["address1_freighttermscode"],
  "cre56_order_date": opportunity["cre56_order_date"],
};

let mainSalesOrderData = Object.assign({}, orderBaseData);

if (opportunity["cre56_ship_to_different_address"]) {
  mainSalesOrderData["shipto_contactname"] = opportunity["cre56_ship_to_contact_name"]
  mainSalesOrderData["shipto_name"] = opportunity["cre56_ship_to_address_name"]
  mainSalesOrderData["shipto_line1"] = opportunity["cre56_ship_to_address_line_1"]
  mainSalesOrderData["shipto_line2"] = opportunity["cre56_ship_to_address_line_2"]
  mainSalesOrderData["shipto_city"] = opportunity["cre56_ship_to_city"]
  mainSalesOrderData["shipto_country"] = opportunity["cre56_ship_to_country"]
  mainSalesOrderData["shipto_postalcode"] = opportunity["cre56_ship_to_post_code"]
  mainSalesOrderData["cre56_different_ship_to_address"] = opportunity["cre56_ship_to_different_address"]
}

if (opportunity["cre56_different_bill_to_address"]) {
  mainSalesOrderData["billto_name"] = opportunity["cre56_bill_to_name_1"];
  mainSalesOrderData["billto_contactname"] = opportunity["cre56_bill_to_name_2"];
  mainSalesOrderData["billto_line1"] = opportunity["cre56_bill_to_address_line_1"];
  mainSalesOrderData["billto_line2"] = opportunity["cre56_bill_to_address_line_2"];
  mainSalesOrderData["billto_city"] = opportunity["cre56_bill_to_city"];
  mainSalesOrderData["billto_country"] = opportunity["cre56_bill_to_country"];
  mainSalesOrderData["billto_postalcode"] = opportunity["cre56_bill_to_post_code"];
}

```

- **postOrder(salesOrderData, salesLinesData)**: Diese Methode erstellt eine Bestellung und die dazugehörigen Auftragspositionen basierend auf den übergebenen Daten.

```

02     if (opportunity['cre56_approval_sample_required']) {
03         let approvalSampleSalesOrderData = Object.assign({}, orderBaseData);
04         approvalSampleSalesOrderData["name"] += ' | Freigabemuster';
05
06         if (opportunity['cre56_different_approval_sample_address']) {
07             approvalSampleSalesOrderData["shipto_name"] = opportunity["cre56_approval_sample_to_name_1"]
08             approvalSampleSalesOrderData["shipto_contactname"] = opportunity["cre56_approval_sample_to_name_2"]
09             approvalSampleSalesOrderData["shipto_line1"] = opportunity["cre56_approval_sample_to_address_line_1"]
10             approvalSampleSalesOrderData["shipto_line2"] = opportunity["cre56_approval_sample_to_address_line_2"]
11             approvalSampleSalesOrderData["shipto_city"] = opportunity["cre56_approval_sample_to_city"]
12             approvalSampleSalesOrderData["shipto_country"] = opportunity["cre56_approval_sample_to_country"]
13             approvalSampleSalesOrderData["shipto_postalcode"] = opportunity["cre56_approval_sample_to_post_code"]
14             approvalSampleSalesOrderData["cre56_different_ship_to_address"] = true;
15         }
16
17         createdOrders.push(await this.postOrder(approvalSampleSalesOrderData, [
18             this.makeSalesLine(account, null, productNumber, 1, pricePerUnit, 100)
19         ]));
20     }
21
22     if (opportunity['cre56_different_archive_sample_to_address'] && archiveSampleQuantity > 0) {
23         let archiveSampleSalesOrderData = Object.assign({}, orderBaseData);
24
25         archiveSampleSalesOrderData["name"] += ' | Belegmuster';
26         archiveSampleSalesOrderData["shipto_name"] = opportunity["cre56_archive_sample_to_name_1"]
27         archiveSampleSalesOrderData["shipto_contactname"] = opportunity["cre56_archive_sample_to_name_2"]
28         archiveSampleSalesOrderData["shipto_line1"] = opportunity["cre56_archive_sample_to_address_line_1"]
29         archiveSampleSalesOrderData["shipto_line2"] = opportunity["cre56_archive_sample_to_address_line_2"]
30         archiveSampleSalesOrderData["shipto_city"] = opportunity["cre56_archive_sample_to_city"]
31         archiveSampleSalesOrderData["shipto_country"] = opportunity["cre56_archive_sample_to_country"]
32         archiveSampleSalesOrderData["shipto_postalcode"] = opportunity["cre56_archive_sample_to_post_code"]
33         archiveSampleSalesOrderData["cre56_different_ship_to_address"] = true;
34
35         createdOrders.push(await this.postOrder(archiveSampleSalesOrderData, [
36             this.makeSalesLine(account, null, productNumber, archiveSampleQuantity, pricePerUnit, discountPercentage)
37         ]));
38     }
39 }

```

- **makeSalesLine(account, salesOrder, product, quantity, pricePerUnit, discountPercentage, discountAbsolute, customName)**: Diese Methode erstellt eine Auftragsposition mit den angegebenen Parametern wie Konto, Produkt, Menge, Preis pro Einheit und Rabatt.

```

217     createdOrders.push(await this.postOrder(approvalSampleSalesOrderData, [
218         this.makeSalesLine(account, null, productNumber, 1, pricePerUnit, 100)
219     ]));
220 }
221 You, 7 hours ago • change line ...
222 if (opportunity['cre56_different_archive_sample_to_address'] && archiveSampleQuantity > 0) {
223     let archiveSampleSalesOrderData = Object.assign({}, orderBaseData);
224
225     archiveSampleSalesOrderData["name"] += ' | Belegmuster';
226     archiveSampleSalesOrderData["shipto_name"] = opportunity["cre56_archive_sample_to_name_1"]
227     archiveSampleSalesOrderData["shipto_contactname"] = opportunity["cre56_archive_sample_to_name_2"]
228     archiveSampleSalesOrderData["shipto_line1"] = opportunity["cre56_archive_sample_to_address_line_1"]
229     archiveSampleSalesOrderData["shipto_line2"] = opportunity["cre56_archive_sample_to_address_line_2"]
230     archiveSampleSalesOrderData["shipto_city"] = opportunity["cre56_archive_sample_to_city"]
231     archiveSampleSalesOrderData["shipto_country"] = opportunity["cre56_archive_sample_to_country"]
232     archiveSampleSalesOrderData["shipto_postalcode"] = opportunity["cre56_archive_sample_to_post_code"]
233     archiveSampleSalesOrderData["cre56_different_ship_to_address"] = true;
234
235     createdOrders.push(await this.postOrder(archiveSampleSalesOrderData, [
236         this.makeSalesLine(account, null, productNumber, archiveSampleQuantity, pricePerUnit, discountPercentage)
237     ]));
238 }
239 }
240

```

- **makeAnnotationLine(salesOrder, annotation)**: Diese Methode erstellt eine Anmerkung als Auftragsposition für zusätzliche Informationen.
- **getVatRate(customer)**: Diese Methode berechnet die Steuer anhand der Informationen aus dem Kunden.

```

/**
 * Make this async too, so it behaves the same as makeSalesLine
 *
 * @param salesOrder
 * @param annotation
 * @returns {{isproductoverridden: boolean, productdescription}}
 */
async makeAnnotationLine(salesOrder, annotation) {
    return {
        "isproductoverridden": true,
        "productdescription": annotation,
        "quantity": 1,
    };
}

/**
 * Get the VAT rate
 * @returns {number}
 */
getVatRate(customer) {
    if (this.includeVAT(customer)) {
        return 19;
    }

    return 0;
}

/**
 *
 * @param account
 * @param opportunity
 * @returns {Promise<*>}
 */
async getPriceListId(account, opportunity) {
    if (account['_defaultpricelevelid_value']) {
        return account['_defaultpricelevelid_value'];
    }
}

```

2.2. Hilfsfunktionen

- Verschiedene Hilfsfunktionen wie `getVatRate`, `getPriceListId`, `includeVAT`, etc., werden verwendet, um die automatisierte Auftragsanlage zu unterstützen.

```

    * @returns {Promise<*>}
    */
    async getPriceListId(account, opportunity) {
        if (account['_defaultpricelevelid_value']) {
            return account['_defaultpricelevelid_value'];
        }

        let priceListName = 'EUR_CO-B. IND 24';

        if (opportunity['cre56_ship_to_country'] === "CH") {
            priceListName = 'EUR_CO-B. IND CH 24';
        }

        let priceList = await this.getPriceListByName(priceListName);

        return priceList.pricelevelid;
    }

```

```

/**
 * Get the VAT rate
 * @returns {number}
 */
getVatRate(customer) {
    if (this.includeVAT(customer)) {
        return 19;
    }

    return 0;
}

/**

```

Version #1

Erstellt: 26 März 2024 13:49:02 von Kay Dietrich

Zuletzt aktualisiert: 26 März 2024 13:56:36 von Kay Dietrich

Schlüsselwörter und Funktionen

Operator	Beschreibung
<code>async</code>	Kennzeichnet eine Funktion als asynchron. Eine asynchrone Funktion gibt immer ein Promise zurück. Sie ermöglicht es, dass die Funktion im Hintergrund ausgeführt wird, während der restliche Code weiterläuft, und später zur Auflösung des Versprechens zurückkehrt, wenn sie abgeschlossen ist.
<code>await</code>	Wird innerhalb einer asynchronen Funktion verwendet, um auf die Auflösung eines Versprechens zu warten. Wenn eine Funktion mit <code>await</code> aufgerufen wird, wird die Ausführung der Funktion unterbrochen, bis das Versprechen erfüllt ist und ein Ergebnis zurückgegeben wird.

Funktion	Was es tut
<code>createOrder()</code>	Diese Methode erstellt Bestellungen, basierend auf einer Verkaufschance. Sie überprüft verschiedene Bedingungen, ruft Daten ab, erstellt Bestellungen und führt den Prozess weiter. Am Ende fragt sie den Benutzer, ob die Bestellungen direkt in Business Central übertragen und geöffnet werden sollen.
<code>postOrder(salesOrderData, salesLinesData)</code>	Diese Methode erstellt eine Bestellung und die dazugehörigen Positionen in Dynamics 365 Sales. Sie akzeptiert Daten für die Bestellung und die Positionen und gibt ein Objekt mit Informationen über die erstellte Bestellung und die Positionen zurück.
<code>makeSalesLine(account, salesOrder, product, quantity, pricePerUnit, discountPercentage, discountAbsolute, customName)</code>	Diese Methode erstellt eine Verkaufspostenzeile für eine Bestellung. Sie akzeptiert Daten wie Konto, Bestellung, Produkt, Menge, Preis pro Einheit und Rabattinformationen und gibt die Daten für die Verkaufspostenzeile zurück.
<code>makeAnnotationLine(salesOrder, annotation)</code>	Diese Methode erstellt eine annotierte Verkaufspostenzeile für eine Bestellung. Sie akzeptiert eine Bestellung und eine Anmerkung und gibt die Daten für die annotierte Verkaufspostenzeile zurück.
<code>getVatRate(customer)</code>	Diese Methode gibt die Mehrwertsteuer (VAT) Rate basierend auf dem Kunden zurück.
<code>getPriceListId(account, opportunity)</code>	Diese Methode gibt die Preislisten-ID basierend auf dem Konto und der Verkaufschance zurück.
<code>getPriceListByName(priceListName)</code>	Diese Methode gibt die Preislisten-Entität basierend auf dem Namen der Preisliste zurück.
<code>includeVAT(customer)</code>	Diese Methode überprüft, ob die Mehrwertsteuer (VAT) in die Bestellung des Kunden einbezogen werden soll.

Funktion	Was es tut
<code>getParentAccountId()</code>	Diese Methode gibt die ID des übergeordneten Kontos zurück.
<code>getProductByNumber(productNumber)</code>	Diese Methode gibt die Produktentität basierend auf der Produktnummer zurück.
