Kostenloses eBook

LERNEN acumatica

Free unaffiliated eBook created from
Stack Overflow contributors.

#acumatica

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: acumatica

It is an unofficial and free acumatica ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official acumatica.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit acumatica

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick darüber, was acumatica ist und warum ein Entwickler es verwenden möchte.

Es sollte auch alle großen Themen innerhalb von acumatica erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für acumatica neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Examples

Installation oder Setup

Detaillierte Anweisungen zum Einrichten oder Installieren von acumatica.

Erste Schritte mit acumatica online lesen: https://riptutorial.com/de/acumatica/topic/7152/erste-schritte-mit-acumatica

Kapitel 2: Acumatica BQL-Referenz

Examples

BQL Parse und Verify

Jeder Acumatica-Anwendungsentwickler verbringt viel Zeit damit, BQL-Code zu schreiben. Gleichzeitig kennen nicht alle die zugrunde liegenden Details der Funktionsweise von BQL-Typen unter der Haube.

Das Kernstück von BQL sind zwei Schlüsselmethoden: `Parse()` und `Verify()` , die von der `IBqlCreator` Schnittstelle `IBqlCreator` werden. Die meisten der häufig verwendeten BQL-Typen, wie z. B. `Where<>` , `And<>` , `Or<>` usw., stammen von dieser Schnittstelle.

Man muss zugeben, dass die Namen, mit denen diese Methoden historisch verbunden sind, nicht sehr beschreibend sind. Möglicherweise bessere alternative Namen für sie wären `PrepareCommandText` und `Evaluate` .

Parse

```
public void Parse(
    PXGraph graph,
    List<IBqlParameter> pars,
    List<Type> tables,
    List<Type> fields,
    List<IBqlSortColumn> sortColumns,
    StringBuilder text,
    BqlCommand.Selection selection)
```

Der einzige Zweck von `Parse()` besteht darin, BQL in einen SQL-Befehl zu übersetzen, der an DBMS gesendet wird. Daher akzeptiert diese Methode einen `StringBuilder` Parameter, der den gerade erstellten SQL-Befehl darstellt, an den der BQL-Ersteller die SQL-Textdarstellung seiner selbst anhängt.

Zum Beispiel hängt die `Parse()` Methode des Prädikats `And<>` an den Befehlstext `" AND "` an und fordert rekursiv die Übersetzung aller geschachtelten BQL-Ersteller an.

Insbesondere und `And<ARRegister.docType, Equal<ARDocType.invoice>>` wird in etwas übersetzt wie `"AND "ARRegister.DocType = 'AR'"` .

Überprüfen

```
public void Verify(
    PXCache cache,
    object item,
```

```
    List<object> pars,
    ref bool? result,
    ref object value)
```

Im Gegensatz zu `Parse()` arbeitet `Verify()` ausschließlich auf Anwendungsebene.

Bei einem Datensatz (z. B. einem `ARRegister` Objekt) können mit ihm Ausdrücke berechnet werden, einschließlich Formeln berechnen und Bedingungen auswerten.

Der `result` wird zum Speichern des Ergebnisses der booleschen Zustandsbewertung verwendet. Es wird hauptsächlich von *Prädikat*- BQL-Erstellern wie `Where<>` .

Der `value` Parameter wird zum Speichern des Ausdrucksberechnungsergebnisses verwendet. Der `value` einer BQL- `Constant<string>` ist beispielsweise die Zeichenfolgendarstellung dieser Konstante.

Meistens beeinflussen BQL-Ersteller entweder das Ergebnis oder den Wert, selten jedoch beide.

Eine bemerkenswerte Verwendung der `Verify()` -Methode ist die statische `BqlCommand.Meet()` -Methode, die von `PXCache` verwendet wird, um festzustellen, ob ein bestimmtes Element den BQL-Befehl erfüllt:

```
public bool Meet(PXCache cache, object item, params object[] parameters)
{
    List<object> pars = new List<object>(parameters);
    bool? result = null;
    object value = null;
    try {
        Verify(cache, item, pars, ref result, ref value);
    }
    catch (SystemException ex) {
        throw new PXException(String.Format("BQL verification failed! {0}", this.ToString()),
ex);
    }
    return result == null || result == true;
}
```

Fazit

Die wahre Stärke und Schönheit von BQL-Erstellern liegt darin, dass die meisten von ihnen sowohl auf Datenbank- als auch auf Anwendungsebene verwendet werden können. Dadurch wird der Mechanismus zur Zusammenführung von Cumatica-Caches ermöglicht und eine großartige Möglichkeit für die Wiederverwendbarkeit von Code geschaffen.

Wenn Sie beispielsweise Datensätze aus der Datenbank auswählen, wird die `Where<>` Klausel des BQL-Befehls verwendet:

- Wird `Parse()` bereitstellen, um sich während der Befehlsvorbereitung in SQL-Text zu übersetzen.
- Liefert `Verify()` während der Cache - Fusion zu bestimmen , welche bereits Elemente im

Cache mit Wohnsitz `Meet()` die `Where<>` Bedingungen Klausel , um eine solche zwischengespeicherten Elemente in der Ergebnismenge aufzunehmen.

Acumatica BQL-Referenz online lesen: https://riptutorial.com/de/acumatica/topic/9690/acumatica-bql-referenz

Kapitel 3: Acumatica-Plattformattribute

Examples

PXFormula-Attribut

Allgemeine Beschreibung

Eine Formel in Acumatica ist ein DAC-Feld, das basierend auf den Werten anderer Objektfelder berechnet wird.

Um eine Formel zu berechnen, stellt Acumatica Framework eine Reihe verschiedener Operationen und Funktionen bereit (z. B. arithmetische, logische Operationen, Vergleichsoperationen und Funktionen zur Verarbeitung von Zeichenfolgen; siehe *Liste der integrierten allgemeinen Formeln*). Zusätzlich zu den Feldwerten kann eine Formel verschiedene Konstanten verwenden, die sowohl vom Kern von Acumatica als auch von den Anwendungslösungen bereitgestellt werden. Außerdem kann eine Formel Werte für die Berechnung nicht nur aus dem aktuellen Datensatz erhalten, sondern auch aus anderen Quellen (siehe *Formelkontext und seine Modifikatoren*).

Das Schöne an den Formeln ist, dass sie den Wert zum richtigen Zeitpunkt automatisch neu berechnen:

- On Field Defaulting (Einfügen einer neuen Zeile; `FieldDefaulting` Ereignishandler des `FieldDefaulting`)
- Beim Aktualisieren abhängiger Felder (`FieldUpdated` Ereignishandler jedes abhängigen Felds)
- Bei Datenbankauswahl (nur für ungebundene Felder; `RowSelecting` Ereignishandler)
- Wenn die Datenbank bei Bedarf persistiert (der Entwickler sollte dies explizit angeben; `RowPersisted` Ereignishandler)

Die Neuberechnung eines `FieldUpdated` bei der Aktualisierung eines abhängigen Felds löst ein `FieldUpdated` Ereignis für das `FieldUpdated` aus. Auf diese Weise können Sie eine Kette abhängiger Formeln erstellen (siehe direkte und vermittelte Zirkelreferenzen in Formeln).

Anwendungsentwickler können eigene anwendungsseitige Formeln schreiben.

Nutzungsarten

Eine Formel kann in drei Hauptmodi verwendet werden:

- Berechnen Sie einfach den Wert und weisen Sie ihn dem Formelfeld zu (siehe *Grundlegende Verwendung*)
- Berechnen des Aggregatwerts aus vorhandenen Werten von Formelfeldern und Zuweisen

zu einem angegebenen Feld im übergeordneten Objekt (siehe *Aggregatverwendung*)
- Mischmodus: Berechnen des Formelwerts, Zuordnen zu dem Formelfeld, Berechnen des Aggregatwerts und Zuweisen zu dem Feld im übergeordneten Objekt (siehe *Kombinierte Verwendung*)

Es gibt einen weiteren Hilfsmodus, ungebundene Formeln, der dem gemischten Modus sehr ähnlich ist, aber die berechneten Werte der Formel werden nicht dem Formelfeld zugewiesen. Der aggregierte Wert wird sofort berechnet und dem Feld des übergeordneten Objekts zugewiesen. Weitere Informationen finden Sie unter *Verwendung von nicht gebundenen Formeln* .

`PXFormulaAttribute` Eigenschaften und Konstruktorparameter

Die `PXFormulaAttribute` wird von `PXFormulaAttribute` implementiert. Der Konstruktor von PXFormulaAttribute hat folgende Signaturen:

```
public PXFormulaAttribute(Type formulaType)
{
    // ...
}
```

Der Einzelparameter `formulaType` ist ein Typ von Formelausdruck, der den Feldwert aus anderen Feldern des gleichen Datensatzes berechnet. Dieser Parameter muss eine der folgenden Bedingungen erfüllen:

- Muss die IBqlField-Schnittstelle implementieren
- Muss eine BQL-Konstante sein
- Muss die IBqlCreator-Schnittstelle implementieren (siehe *Liste der integrierten allgemeinen Formeln*).

```
public PXFormulaAttribute(Type formulaType, Type aggregateType)
{
    // ...
}
```

Der erste Parameter, `formulaType` , ist derselbe wie im ersten Konstruktor. Der zweite Parameter, `aggregateType` , ist eine Art Aggregationsformel zur Berechnung des Felds des übergeordneten Datensatzes aus den Feldern des untergeordneten Datensatzes. Eine Aggregationsfunktion kann verwendet werden, z. B. SumCalc, CountCalc, MinCalc und MaxCalc. Anwendungsentwickler können ihre eigenen Aggregationsformeln erstellen.

Ein Aggregatformeltyp muss ein generischer Typ sein und die `IBqlAggregateCalculator` Schnittstelle implementieren. Der erste generische Parameter des Aggregatformeltyps muss die `IBqlField` Schnittstelle implementieren und muss den `IBqlField` des übergeordneten Objekts haben.

```
public virtual bool Persistent { get; set; }
```

Die Eigenschaft `PXFormulaAttribute.Persistent` gibt an, ob das Attribut die Formel neu berechnet, nachdem Änderungen in der Datenbank gespeichert wurden. Möglicherweise müssen Sie eine Neuberechnung durchführen, wenn die Felder, von denen die Formel abhängig ist, im `RowPersisting` Ereignis aktualisiert werden. Die Eigenschaft entspricht standardmäßig " `false` .

Verwendungszweck

In den meisten Fällen werden Formeln zur direkten Berechnung des Wertes des Formelfelds aus anderen Feldern des gleichen Datensatzes verwendet.

Das einfachste Beispiel für die Verwendung von Formeln:

```
[PXDBDate]
[PXFormula(typeof(FADetails.receiptDate))]
[PXDefault]
[PXUIField(DisplayName = Messages.PlacedInServiceDate)]
public virtual DateTime? DepreciateFromDate { get; set; }
```

In diesem Beispiel wird der Wert des ReceiptDate-Felds beim Einfügen eines neuen Datensatzes und bei der Aktualisierung des ReceiptDate-Felds dem DepreciateFromDate-Feld zugewiesen.

Ein etwas komplexeres Beispiel:

```
[PXCurrency(typeof(APPayment.curyInfoID), typeof(APPayment.unappliedBal))]
[PXUIField(DisplayName = "Unapplied Balance", Visibility = PXUIVisibility.Visible, Enabled =
false)]
[PXFormula(typeof(Sub<APPayment.curyDocBal, APPayment.curyApplAmt>))]
public virtual Decimal? CuryUnappliedBal { get; set; }
```

Hier wird der nicht belegte Restbetrag des Belegs als Differenz zwischen dem Restbetrag des Belegs und dem verwendeten Betrag berechnet.

Beispiel für Multiple Choice mit einem Standardwert:

```
[PXUIField(DisplayName = "Class Icon", IsReadOnly = true)]
[PXImage]
[PXFormula(typeof(Switch<
    Case<Where<EPActivity.classID, Equal<CRActivityClass.task>>, EPActivity.classIcon.task,
    Case<Where<EPActivity.classID, Equal<CRActivityClass.events>>,
EPActivity.classIcon.events,
    Case<Where<EPActivity.classID, Equal<CRActivityClass.email>,
        And<EPActivity.isIncome, NotEqual<True>>>, EPActivity.classIcon.email,
    Case<Where<EPActivity.classID, Equal<CRActivityClass.email>,
        And<EPActivity.isIncome, Equal<True>>>, EPActivity.classIcon.emailResponse,
    Case<Where<EPActivity.classID, Equal<CRActivityClass.history>>,
EPActivity.classIcon.history>>>>,
    Selector<Current2<EPActivity.type>, EPActivityType.imageUrl>>))]
public virtual string ClassIcon { get; set; }
```

Reihenfolge der Felder

Die Reihenfolge der Felder im DAC ist wichtig, um die Formelberechnung zu korrigieren. Alle Quellfelder (aus denen die Formel berechnet wird) einschließlich anderer Formeln müssen vor dem Formelfeld im DAC definiert werden. Andernfalls kann das Feld falsch berechnet werden oder einen Laufzeitfehler verursachen.

Formelkontext und seine Modifikatoren

Standardmäßig ist der Kontext der Formelberechnung durch das aktuelle Objekt (Datensatz) der Klasse eingeschränkt, die die Formeldeklaration enthält. Es ist auch erlaubt, Konstanten (Nachkommen der `Constant<>` -Klasse) zu verwenden.

Eine Formel, die nur die Felder ihres Objekts verwendet:

```
public partial class Contract : IBqlTable, IAttributeSupport
{
    //...
    [PXDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0", PersistingCheck = PXPersistingCheck.Nothing)]
    [PXFormula(typeof(Add<Contract.pendingRecurring, Add<Contract.pendingRenewal,
Contract.pendingSetup>>))]
    [PXUIField(DisplayName = "Total Pending", Enabled=false)]
    public virtual decimal? TotalPending { get; set; }
    //...
}
```

Es ist jedoch möglich, Eingabewerte für die Formelberechnung aus anderen Quellen zu erhalten:

- Ein aktueller Datensatz eines Cache im BLC (falls zugewiesen).
- Ein von `PXSelectorAttribute` angegebener `PXSelectorAttribute` .
- Ein durch `PXParentAttribute` angegebener übergeordneter Datensatz.

Die Formel unterstützt die folgenden Kontextmodifizierer.

`Current<TRecord.field>` und `Current2<TRecord.field>`

Ruft den Feldwert aus dem Datensatz ab, der in der `Current` Eigenschaft des TRecord-Caches gespeichert ist.

Wenn die `Current` Eigenschaft des Caches **oder das Feld selbst** null enthält:

- Current <> erzwingt die Standardeinstellung des Felds und gibt den Standardfeldwert zurück.
- Current2 <> gibt null zurück.

Beispiel:

```
[PXFormula(typeof(Switch<
    Case<Where<
        ARAdjust.adjgDocType, Equal<Current<ARPayment.docType>>,
        And<ARAdjust.adjgRefNbr, Equal<Current<ARPayment.refNbr>>>>,
        ARAdjust.classIcon.outgoing>,
    ARAdjust.classIcon.incoming>))]
protected virtual void ARAdjust_ClassIcon_CacheAttached(PXCache sender)
```

`Parent<TParent.field>`

Ruft den Feldwert aus dem übergeordneten Datensatz ab, wie von PXParentAttribute definiert, das sich auf dem aktuellen DAC befindet.

```
public class INTran : IBqlTable
{
    [PXParent(typeof(Select<
        INRegister,
        Where<
            INRegister.docType, Equal<Current<INTran.docType>>,
            And<INRegister.refNbr,Equal<Current<INTran.refNbr>>>>>))]
    public virtual String RefNbr { ... }

    [PXFormula(typeof(Parent<INRegister.origModule>))]
    public virtual String OrigModule { ... }
}
```

`IsTableEmpty<TRecord>`

Gibt " `true` wenn die dem angegebenen DAC entsprechende DB-Tabelle keine Datensätze enthält, andernfalls " `false` .

```
public class APRegister : IBqlTable
{
    [PXFormula(typeof(Switch<
        Case<Where<
            IsTableEmpty<APSetupApproval>, Equal<True>>,
            True,
        Case<Where<
            APRegister.requestApproval, Equal<True>>,
            False>>,
        True>))]
    public virtual bool? DontApprove { get; set; }
}
```

`Selector<KeyField, ForeignOperand>`

Ruft ein PXSelectorAttribute ab, das für das Fremdschlüsselfeld (KeyField) des aktuellen DAC definiert ist.

Ruft den aktuell vom Selektor referenzierten Fremddatensatz ab.

Berechnet einen Ausdruck für diesen Datensatz und gibt diesen zurück, wie von ForeignOperand definiert.

```
public class APVendorPrice : IBqlTable
{
    // Note: inventory attribute is an
    // aggregate containing a PXSelectorAttribute
    // inside, which is also valid for Selector<>.
    // -
    [Inventory(DisplayName = "Inventory ID")]
    public virtual int? InventoryID

    [PXFormula(typeof(Selector<
        APVendorPrice.inventoryID,
        InventoryItem.purchaseUnit>))]
    public virtual string UOM { get; set; }
}
```

Verwenden von Formeln für nicht gebundene Felder

Wenn das `PXFieldAttribute` ein ungebundenes Feld ist, das mit einer der Nachkommen von `PXFieldAttribute` (z. B. `PXIntAttribute` oder `PXStringAttribute`) `PXStringAttribute` , wird die Berechnung zusätzlich während des `RowSelecting` Ereignisses ausgelöst.

Liste der integrierten allgemeinen Formeln

TBD

Direkte und vermittelte Zirkelreferenzen in Formeln

TBD

Kontrollfluss in bedingten Formeln

TBD

Mehrere Formeln in einem Feld verwenden

TBD

PXRestrictor-Attribut

Einführung

Das PXSelectorAttribute-Attribut (auch als Selektor bezeichnet) ist zwar wichtig und häufig verwendet, hat jedoch zwei Hauptnachteile:

- Wenn keine Elemente gefunden werden, die die Selektor-Bedingung erfüllen, wird eine nicht `"<object_name> cannot be found in the system"` Nachricht `"<object_name> cannot be found in the system".`
- Die gleiche Fehlermeldung wird angezeigt, wenn Sie *andere* Felder des Datensatzes aktualisieren, das vom Selector referenzierte Objekt jedoch bereits geändert wurde und seine Bedingung nicht mehr erfüllt. Dieses Verhalten ist eindeutig falsch, da das Gesetz nicht rückwirkend sein darf.

Das `PXRestrictorAttribute` (auch als Beschränkung bezeichnet) kann verwendet werden, um diese Probleme zu lösen.

Einzelheiten

`PXRestrictorAttribute` funktioniert nicht alleine. Es sollte immer mit einem `PXSelectorAttribute` gepaart werden. Die Verwendung der Drossel ohne den Wahlschalter hat keine Wirkung.

Der Begrenzer findet den Selektor auf demselben Feld und fügt ihm eine zusätzliche Bedingung und die entsprechende Fehlermeldung ein. Die Beschränkungsbedingung wird über eine boolesche UND-Verknüpfung an die Auswahlbedingung angehängt, und eine entsprechende Fehlermeldung wird generiert, wenn das referenzierte Objekt die Beschränkungsbeschränkung verletzt. Wenn das referenzierte Objekt geändert wurde und die Einschränkungsbedingung nicht mehr erfüllt, werden keine Fehlermeldungen ausgegeben, wenn Sie **andere** Felder des referenzierenden Objekts ändern.

Allgemeine Verwendung:

```
[PXDBInt]
[PXSelector(typeof(Search<FAClass.assetID, Where<FAClass.recordType,
```

```
Equal<FARecordType.classType>>>),
    typeof(FAClass.assetCD), typeof(FAClass.assetTypeID), typeof(FAClass.description),
typeof(FAClass.usefulLife),
    SubstituteKey = typeof(FAClass.assetCD),
    DescriptionField = typeof(FAClass.description), CacheGlobal = true)]
[PXRestrictor(typeof(Where<FAClass.active, Equal<True>>), Messages.InactiveFAClass,
typeof(FAClass.assetCD))]
[PXUIField(DisplayName = "Asset Class", Visibility = PXUIVisibility.Visible)]
public virtual int? ClassID { get; set; }
```

Mit einem Selektorattribut können mehrere Restriktoren verwendet werden. In diesem Fall werden alle zusätzlichen Begrenzungsbedingungen in einer nicht bestimmten Reihenfolge angewendet. Sobald eine Bedingung verletzt wird, wird die entsprechende Fehlermeldung generiert.

Die `Where<>` Bedingung des Wahlschalters selbst wird **nach** allen Begrenzerbedingungen angewendet.

```
[PXDefault]
//  An aggregate attribute containing the selector inside.
// -
[ContractTemplate(Required = true)]
[PXRestrictor(typeof(Where<ContractTemplate.status, Equal<Contract.status.active>>),
Messages.TemplateIsNotActivated, typeof(ContractTemplate.contractCD))]
[PXRestrictor(typeof(Where<ContractTemplate.effectiveFrom,
LessEqual<Current<AccessInfo.businessDate>>,
    Or<ContractTemplate.effectiveFrom, IsNull>>), Messages.TemplateIsNotStarted)]
[PXRestrictor(typeof(Where<ContractTemplate.discontinueAfter,
GreaterEqual<Current<AccessInfo.businessDate>>,
    Or<ContractTemplate.discontinueAfter, IsNull>>), Messages.TemplateIsExpired)]
public virtual int? TemplateID { get; set; }
```

Optionen

Der Konstruktor von PXRestrictorAttribute benötigt drei Parameter:

1. Der zusätzliche Zustand der Drossel. Dieser BQL-Typ muss die `IBqlWhere` Schnittstelle implementieren.
2. Die entsprechende Fehlermeldung Die Nachricht kann Formatelemente (geschweifte Klammern) enthalten, um den Kontext anzuzeigen. Die Nachricht muss eine Zeichenfolgekonstante sein, die in einer lokalisierbaren statischen Klasse (z. B. `PX.Objects.GL.Messages`) definiert ist.
3. Ein Array von Feldtypen. Diese Felder müssen zum aktuellen Objekt gehören und die `IBqlField` Schnittstelle implementieren. Die Werte der Felder werden für die Formatierung von Fehlernachrichten verwendet.

Außerdem gibt es mehrere Optionen, die das Verhalten der Drossel festlegen.

Vererbte Restriktoren überschreiben

Die `ReplaceInherited` Eigenschaft gibt an, ob die aktuelle Beschränkung die geerbten

Einschränkungen einschränken soll. Wenn diese Eigenschaft auf true festgelegt ist, werden alle geerbten Restriktoren (in Aggregatattributen oder Basisattributen platziert) ersetzt.

Vererbte Restriktoren ersetzen:

```
 [CustomerActive(Visibility = PXUIVisibility.SelectorVisible, Filterable = true, TabOrder =
2)]
 [PXRestrictor(typeof(Where<Customer.status, Equal<CR.BAccount.status.active>,
    Or<Customer.status, Equal<CR.BAccount.status.oneTime>,
    Or<Customer.status, Equal<CR.BAccount.status.hold>,
    Or<Customer.status, Equal<CR.BAccount.status.creditHold>>>>>),
Messages.CustomerIsInStatus, typeof(Customer.status),
    ReplaceInherited = true)] // Replaced all restrictors from CustomerActiveAttribute
[PXUIField(DisplayName = "Customer")]
[PXDefault()]
public override int? CustomerID { get; set; }
```

Beachten Sie, dass wir nicht empfehlen, die `ReplaceInherited` Eigenschaft im Anwendungscode zu verwenden, wenn sinnvolle Alternativen vorhanden sind. Diese Eigenschaft soll in erster Linie in Anpassungen verwendet werden.

Globales Caching

`CacheGlobal` unterstützt globale Wörterbuchfunktionen auf dieselbe Weise wie in `PXSelectorAttribute` .

Empfehlungen zur Verwendung

Verwenden Sie nur Restriktorbedingungen

Wenn Restriktoren und ein Selektor zusammen verwendet werden, sollte dieser nicht die `IBqlWhere` Klausel enthalten. Im Idealfall sollten alle Bedingungen in Restriktoren verschoben werden. Dieser Ansatz bietet benutzerfreundlichere Fehlermeldungen und beseitigt unnötige rückwirkende Fehler.

Ein ideales Beispiel:

```
[PXDBString(5, IsFixed = true, IsUnicode = false)]
[PXUIField(DisplayName = "Type", Required = true)]
[PXSelector(typeof(EPActivityType.type), DescriptionField =
typeof(EPActivityType.description))]
[PXRestrictor(typeof(Where<EPActivityType.active, Equal<True>>),
Messages.InactiveActivityType, typeof(EPActivityType.type))]
[PXRestrictor(typeof(Where<EPActivityType.isInternal, Equal<True>>),
Messages.ExternalActivityType, typeof(EPActivityType.type))]
public virtual string Type { get; set; }
```

Mögliche rückwirkende Fehler:

```
[PXDBInt]
[PXUIField(DisplayName = "Contract")]
[PXSelector(typeof(Search2<Contract.contractID,
    LeftJoin<ContractBillingSchedule, On<Contract.contractID,
Equal<ContractBillingSchedule.contractID>>>,
    Where<Contract.isTemplate, NotEqual<True>,
        And<Contract.baseType, Equal<Contract.ContractBaseType>,
        And<Where<Current<CRCase.customerID>, IsNull,
            Or2<Where<Contract.customerID, Equal<Current<CRCase.customerID>>,
                And<Current<CRCase.locationID>, IsNull>>,
            Or2<Where<ContractBillingSchedule.accountID, Equal<Current<CRCase.customerID>>,
                And<Current<CRCase.locationID>, IsNull>>,
            Or2<Where<Contract.customerID, Equal<Current<CRCase.customerID>>,
                And<Contract.locationID, Equal<Current<CRCase.locationID>>>>,
            Or<Where<ContractBillingSchedule.accountID, Equal<Current<CRCase.customerID>>,
                And<ContractBillingSchedule.locationID,
Equal<Current<CRCase.locationID>>>>>>>>>>>,
    OrderBy<Desc<Contract.contractCD>>>),
    DescriptionField = typeof(Contract.description),
    SubstituteKey = typeof(Contract.contractCD), Filterable = true)]
[PXRestrictor(typeof(Where<Contract.status, Equal<Contract.status.active>>),
Messages.ContractIsNotActive)]
[PXRestrictor(typeof(Where<Current<AccessInfo.businessDate>, LessEqual<Contract.graceDate>,
Or<Contract.expireDate, IsNull>>), Messages.ContractExpired)]
[PXRestrictor(typeof(Where<Current<AccessInfo.businessDate>,
GreaterEqual<Contract.startDate>>), Messages.ContractActivationDateInFuture,
typeof(Contract.startDate))]
[PXFormula(typeof(Default<CRCase.customerID>))]
[PXDefault(PersistingCheck = PXPersistingCheck.Nothing)]
public virtual int? ContractID { get; set; }
```

Acumatica-Plattformattribute online lesen:
https://riptutorial.com/de/acumatica/topic/8853/acumatica-plattformattribute

Kapitel 4: Ändern der Beschriftung mithilfe von Readonly-DAC-Feldern.

Einführung

In diesem Beispiel wird gezeigt, wie das Feld Beschriftung / Bezeichnung des Kundennamens in Customer ScreenID AR303000 in Acumatica ERP dynamisch geändert wird, abhängig von der aktuell im selben Formular ausgewählten Kunden-ID. Wir können:

Examples

Wie man

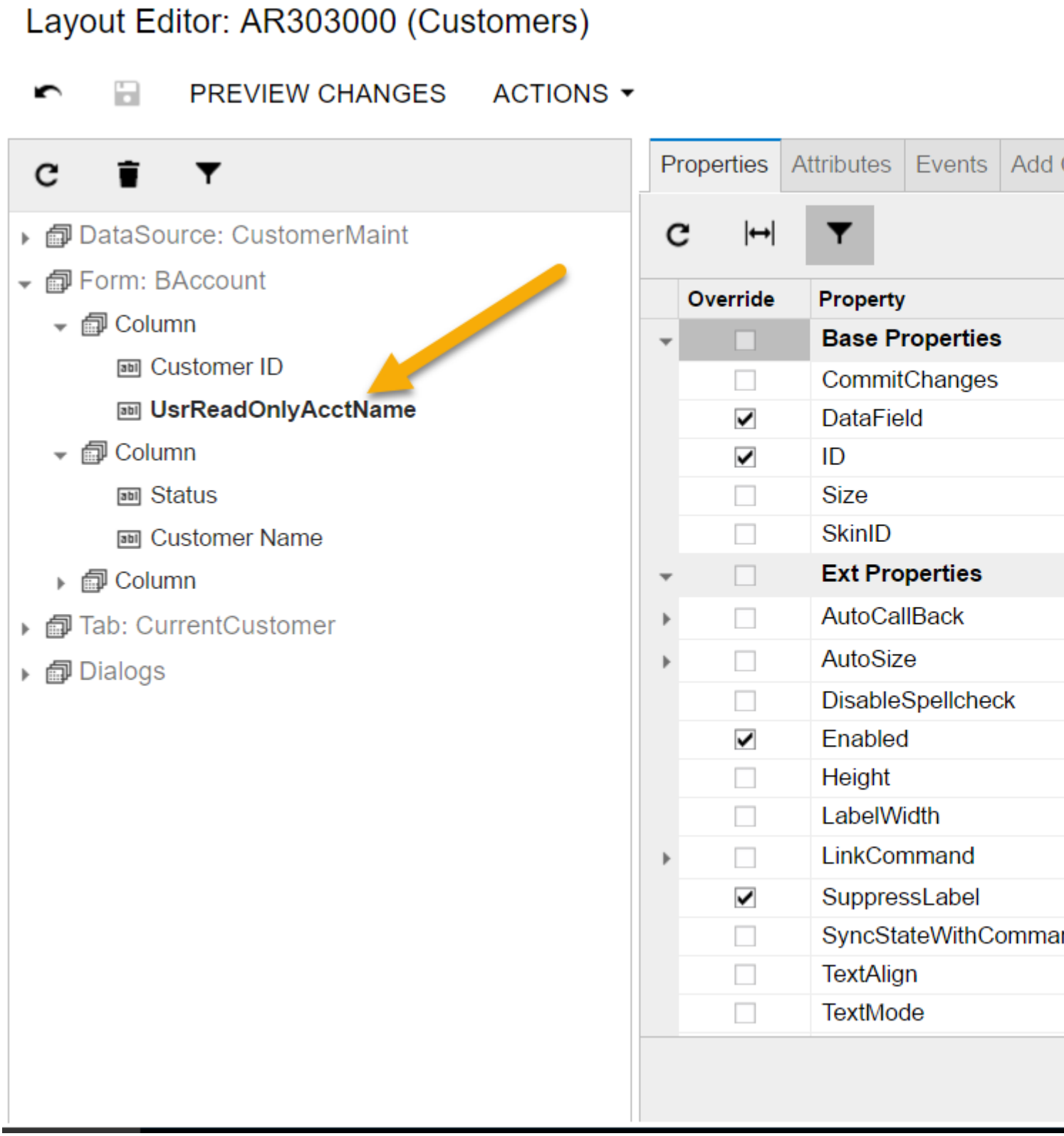
Fügen Sie dem DAC ein neues ungebundenes Feld hinzu. (wie Readonly)

```
  [PXString(60, IsUnicode = true)]
  [PXUIField(Enabled = false, IsReadOnly = true)]
  public virtual string UsrReadOnlyAcctName{get;set;}
  public abstract class usrReadOnlyAcctName : IBqlField{}
```

Ändern Sie den Wert in Abhängigkeit von den Bedingungen mithilfe von Handlern. (Nach Kundenzyklus-ID ausgewählt)

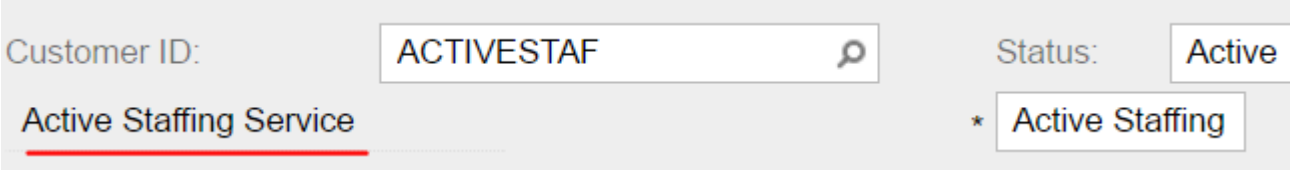
```
public class CustomerMaint_Extension:PXGraphExtension<CustomerMaint>
  {
    protected void Customer_RowSelected(PXCache sender, PXRowSelectedEventArgs e)
      {
          var customer = (BAccount)e.Row;
          var customerExt = customer.GetExtension<BAccountExt>();
          if (customerExt != null)
          {
              customerExt.UsrReadOnlyAcctName = customer.AcctName;
          }
      }
  }
```

SuppressLabel (true) sowohl für neue ungebundene Felder als auch für vorhandene Felder, deren Bezeichnung ersetzt wird.



Platzieren Sie das hinzugefügte ungebundene Feld vor dem vorhandenen Feld.

Ergebnisse:



Ändern der Beschriftung mithilfe von Readonly-DAC-Feldern. online lesen: https://riptutorial.com/de/acumatica/topic/8858/andern-der-beschriftung-mithilfe-von-readonly-dac-feldern-

Kapitel 5: Änderungen an Basisdatenansichten

Einführung

In diesem Thema werden verschiedene Muster und Vorgehensweisen veranschaulicht, die zum Ändern der Basisdatensichten in Acumatica verfügbar sind.

Examples

APInvoiceEntry BLC: Fügen Sie der Datenansicht poReceiptLinesSelection eine zusätzliche Einschränkung hinzu

Um der **Datenansicht poReceiptLinesSelection** eine zusätzliche Einschränkung hinzuzufügen, müssen Sie die `Select` Methode in der Basisansicht aufrufen und jedes Element in dem zurückgegebenen `PXResultSet` unabhängig voneinander untersuchen, um zu entscheiden, ob es zusätzliche Einschränkungen erfüllt:

```
public class APInvoiceEntryExt : PXGraphExtension<APInvoiceEntry>
{
    [PXCopyPasteHiddenView]
    public PXSelect<APInvoiceEntry.POReceiptLineAdd> poReceiptLinesSelection;

    public virtual IEnumerable POReceiptLinesSelection()
    {
        foreach (var record in Base.poReceiptLinesSelection.Select())
        {
            // Additional restriction goes here
            if (true == true)
            {
                yield return record;
            }
        }
    }
}
```

Dieser Ansatz funktioniert perfekt mit der **Datensicht poReceiptLinesSelection** , da bei der Implementierung der **Basissicht** Paging und Aggregation fehlen. Um die Ergebnismenge zu **erstellen** , **fordert die** Ansicht **poReceiptLinesSelection die** erforderlichen Daten aus der Datenbank an und führt alle Berechnungen auf der Anwendungsseite aus.

```
public class APInvoiceEntry : APDataEntryGraph<APInvoiceEntry, APInvoice>,
PXImportAttribute.IPXPrepareItems
{
    ...

    [PXCopyPasteHiddenView]
    public PXSelect<POReceiptLineAdd> poReceiptLinesSelection;
```

```
    public virtual IEnumerable POReceiptLinesSelection()
    {
        APInvoice doc = this.Document.Current;
        if (doc == null || doc.VendorID == null || doc.VendorLocationID == null) yield break;
        if (doc.DocType != APDocType.Invoice && doc.DocType != APDocType.DebitAdj)
            yield break;

        string poReceiptType = (doc.DocType == APDocType.Invoice) ? POReceiptType.POReceipt :
POReceiptType.POReturn;

        HashSet<APTran> usedRecceiptLine = new HashSet<APTran>(new POReceiptLineComparer());
        HashSet<APTran> unusedReceiptLine = new HashSet<APTran>(new POReceiptLineComparer());

        foreach (APTran aPTran in Transactions.Cache.Inserted)
        {
            if (aPTran.ReceiptNbr != null && aPTran.ReceiptType != null &&
aPTran.ReceiptLineNbr != null)
                usedRecceiptLine.Add(aPTran);
        }

        foreach (APTran aPTran in Transactions.Cache.Deleted)
        {
            if (aPTran.ReceiptNbr != null && aPTran.ReceiptType != null &&
aPTran.ReceiptLineNbr != null && Transactions.Cache.GetStatus(aPTran) !=
PXEntryStatus.InsertedDeleted)
                if (!usedRecceiptLine.Remove(aPTran))
                    unusedReceiptLine.Add(aPTran);
        }

        foreach (APTran aPTran in Transactions.Cache.Updated)
        {
            APTran originAPTran = new APTran();
            originAPTran.ReceiptNbr =
(String)Transactions.Cache.GetValueOriginal<APTran.receiptNbr>(aPTran);
            originAPTran.ReceiptType =
(String)Transactions.Cache.GetValueOriginal<APTran.receiptType>(aPTran);
            originAPTran.ReceiptLineNbr =
(Int32?)Transactions.Cache.GetValueOriginal<APTran.receiptLineNbr>(aPTran);

            if (originAPTran.ReceiptNbr != null && originAPTran.ReceiptType != null &&
originAPTran.ReceiptLineNbr != null)
            {
                if (!usedRecceiptLine.Remove(originAPTran))
                    unusedReceiptLine.Add(originAPTran);
            }

            if (aPTran.ReceiptNbr != null && aPTran.ReceiptType != null &&
aPTran.ReceiptLineNbr != null)
            {
                if (!unusedReceiptLine.Remove(aPTran))
                    usedRecceiptLine.Add(aPTran);
            }
        }

        foreach (LinkLineReceipt item in PXSelect<LinkLineReceipt,
            Where<LinkLineReceipt.vendorID, Equal<Current<APInvoice.vendorID>>,
            And<LinkLineReceipt.vendorLocationID, Equal<Current<APInvoice.vendorLocationID>>,
            And<LinkLineReceipt.receiptCuryID, Equal<Current<APInvoice.curyID>>,
            And<LinkLineReceipt.receiptType, Equal<Required<POReceipt.receiptType>>,
            And<Where<LinkLineReceipt.orderNbr, Equal<Current<POReceiptFilter.orderNbr>>,
Or<Current<POReceiptFilter.orderNbr>, IsNull>>>
```

```
            >>>>>.SelectMultiBound(this, new object[] { doc }, poReceiptType))
        {
            APTran aPTran = new APTran();
            aPTran.ReceiptType = item.ReceiptType;
            aPTran.ReceiptNbr = item.ReceiptNbr;
            aPTran.ReceiptLineNbr = item.ReceiptLineNbr;
            if (!usedRecceiptLine.Contains(aPTran))
                yield return (PXResult<POReceiptLineAdd,
POReceipt>)ReceipLineAdd.Select(item.ReceiptNbr, item.ReceiptType, item.ReceiptLineNbr);
        }

        foreach (APTran item in unusedReceiptLine)
        {
            yield return (PXResult<POReceiptLineAdd,
POReceipt>)ReceipLineAdd.Select(item.ReceiptNbr, item.ReceiptType, item.ReceiptLineNbr);
        }

    }

    ...
}
```

Änderungen an Basisdatenansichten online lesen:
https://riptutorial.com/de/acumatica/topic/9101/anderungen-an-basisdatenansichten

Kapitel 6: Änderungen an Kontakt- und Adressinformationen durch Code

Einführung

In diesem Thema erfahren Sie, wie Sie Kontakt- und Adressinformationen durch Code auf verschiedenen Bildschirmen in Acumatica ändern.

Examples

Geben Sie Kontakt- und Adressinformationen für einen neuen Mitarbeiter an

Um Kontakt- und Adressinformationen für einen Mitarbeiter anzugeben, sollten Sie vor dem Zuweisen von Feldwerten immer die `Select()` Methode in den **Kontakt-** und **Adressdatensichten** aufrufen. Es ist auch das Ergebnis der empfohlenen zuweisen `Select()` Methode , um die **Kontakt-** und **Adressdatenansichten *Current*** - Eigenschaft zu gewährleisten , dass der Code den aktuellen Datensatz in **Kontakt** und **Adresse** PXCache modifiziert ist.

```
EmployeeMaint employeeMaintGraph = PXGraph.CreateInstance<EmployeeMaint>();
EPEmployee epEmployeeRow = new EPEmployee();
epEmployeeRow.AcctCD = "EMPLOYEE1";
epEmployeeRow = employeeMaintGraph.Employee.Insert(epEmployeeRow);

Contact contactRow = employeeMaintGraph.Contact.Current = employeeMaintGraph.Contact.Select();
contactRow.FirstName = "John";
contactRow.LastName = "Green";
employeeMaintGraph.Contact.Update(contactRow);

Address addressRow = employeeMaintGraph.Address.Current = employeeMaintGraph.Address.Select();
addressRow.CountryID = "US";
addressRow = employeeMaintGraph.Address.Update(addressRow);
addressRow.State = "DC";
employeeMaintGraph.Address.Update(addressRow);

epEmployeeRow.VendorClassID = "EMPSTAND";
epEmployeeRow.DepartmentID = "FINANCE";
employeeMaintGraph.Employee.Update(epEmployeeRow);

employeeMaintGraph.Actions.PressSave();
```

Beim Einfügen eines neuen Mitarbeiters gibt `employeeMaintGraph.Contact.Current` immer den Hauptkontaktdatensatz zurück, da der Kontaktdatensatz automatisch in den Cache eingefügt wird und daher über die ***Current-*** Eigenschaft von PXCache / Data View verfügbar ist. Die Verwendung der `Select()` Methode ist etwas generischer, da sie in allen möglichen Szenarien funktioniert, unabhängig davon, ob Sie einen neuen Mitarbeiter einfügen oder einen vorhandenen aktualisieren müssen.

Überschreiben Sie Rechnungskontakt- und

Rechnungsadresseninformationen für einen Kunden

Wenn Sie die Informationen für Rechnungskontakt und Rechnungsadresse für einen Kunden überschreiben müssen, müssen Sie zunächst die korrekten Werte für die Eigenschaften ***IsBillContSameAsMain*** und ***IsBillSameAsMain*** des **Customer-** DAC ***festlegen*** . Vergessen Sie nicht, die `Update()` Methode direkt im **Customer-** Cache aufzurufen, nachdem Sie die ***IsBillContSameAsMain-*** oder ***IsBillSameAsMain-*** Eigenschaft aktualisiert ***haben*** , um den aktuellen Wert des Feldes ***Same as Main*** in den Cache zu übernehmen.

Im nächsten Schritt müssen Sie die `Select()` Methode in den **Datensichten BillContact** und **BillAddress aufrufen** , bevor Sie **Feldwerte zuweisen** . Es ist auch das Ergebnis der empfohlenen zuweisen `Select()` Methode , um die ***aktuellen*** Eigenschaft **BillContact** und **BillAddress** Datenansichten zu gewährleisten , dass der Code den aktuellen Datensatz in **Kontakt** und **Adresse** PXCache modifiziert ist.

```
public class CustomerMaintExt : PXGraphExtension<CustomerMaint>
{
    public PXAction<Customer> UpdateBillingAddress;
    [PXButton(CommitChanges = true)]
    [PXUIField(DisplayName = "Update Bill-To Info")]
    protected void updateBillingAddress()
    {
        Customer currentCustomer = Base.BAccount.Current;

        if (currentCustomer.IsBillContSameAsMain != true)
        {
            currentCustomer.IsBillContSameAsMain = true;
            Base.BAccount.Update(currentCustomer);
        }
        else
        {
            currentCustomer.IsBillContSameAsMain = false;
            Base.BAccount.Update(currentCustomer);

            Contact billContact = Base.BillContact.Current = Base.BillContact.Select();
            billContact.FullName = "ABC Holdings Inc";
            billContact.Phone1 = "+1 (212) 532-9574";
            Base.BillContact.Update(billContact);
        }

        if (currentCustomer.IsBillSameAsMain != true)
        {
            currentCustomer.IsBillSameAsMain = true;
            Base.CurrentCustomer.Update(currentCustomer);
        }
        else
        {
            currentCustomer.IsBillSameAsMain = false;
            Base.CurrentCustomer.Update(currentCustomer);

            Address billAddress = Base.BillAddress.Current = Base.BillAddress.Select();
            billAddress.AddressLine1 = "65 Broadway";
            billAddress.AddressLine2 = "Office Suite 187";
            billAddress.City = "New York";
            billAddress.CountryID = "US";
            billAddress = Base.BillAddress.Update(billAddress);
```

```
            billAddress.State = "NY";
            billAddress.PostalCode = "10004";
            Base.BillAddress.Update(billAddress);
        }

        Base.Actions.PressSave();
    }
}
```

Überschreiben von Rechnungskontakt- und Rechnungsadresseninformationen für einen Kundenauftrag

Um Informationen zu Rechnungskontakt und Rechnungsadresse für einen **Kundenauftrag anzugeben** , sollten Sie immer zuerst die `Select()` Methode in den **Datenansichten "Billing_Contact"** und " **Billing_Address" aufrufen** , bevor Sie **Feldwerte zuweisen** . Es ist auch das Ergebnis der empfohlenen zuweisen `Select()` Methode , um die ***aktuellen*** Eigenschaft **Billing_Contact** und **Billing_Address** Datenansichten zu gewährleisten , dass der Code den aktuellen Datensatz in **SOBillingContact** und **SOBillingAddress** PXCache modifiziert ist.

Wenn Sie **Rechnungskontaktinformationen** und **Adressinformationen** für einen **Kundenauftrag überschreiben müssen** , legen Sie korrekte Werte für die Eigenschaften ***OverrideContact*** und ***OverrideAddress*** auf der DAC **SOBillingContact** und der DAC **SOBillingAddress fest** . Vergessen Sie nicht, die `Update()` Methode für die **SOBillingContact-** und **SOBillingAddress-** Caches direkt nach der Aktualisierung der ***OverrideContact-*** und ***OverrideAddress-*** Eigenschaften **aufzurufen** , um die Änderungen zu übernehmen. Sobald dies erledigt ist, können Sie die Angaben für Rechnungskontakt und Adresse für einen Kundenauftrag angeben.

```
public class SOOrderEntryExt : PXGraphExtension<SOOrderEntry>
{
    public PXAction<SOOrder> UpdateBillingAddress;
    [PXButton(CommitChanges = true)]
    [PXUIField(DisplayName = "Update Bill-To Info")]
    protected void updateBillingAddress()
    {
        SOBillingContact contact = Base.Billing_Contact.Current =
Base.Billing_Contact.Select();
        if (contact.OverrideContact == true)
        {
            contact.OverrideContact = false;
            Base.Billing_Contact.Update(contact);
        }
        else
        {
            contact.OverrideContact = true;
            contact = Base.Billing_Contact.Update(contact);
            if (contact == null)
            {
                contact = Base.Billing_Contact.Current;
            }

            contact.Phone1 = "+1 (908) 643-0281";
            contact.Email = "sales@usabartend.con";
            Base.Billing_Contact.Update(contact);
```

```
        }

        SOBillingAddress address = Base.Billing_Address.Current =
Base.Billing_Address.Select();
        if (address.OverrideAddress == true)
        {
            address.OverrideAddress = false;
            Base.Billing_Address.Update(address);
        }
        else
        {
            address.OverrideAddress = true;
            address = Base.Billing_Address.Update(address);
            if (address == null)
            {
                address = Base.Billing_Address.Current;
            }

            address.AddressLine1 = "201 Lower Notch Rd";
            address.AddressLine2 = "Office Suite 1936";
            address.City = "Little Falls";
            address.CountryID = "US";
            address = Base.Billing_Address.Update(address);
            address.State = "NJ";
            address.PostalCode = "07425";
            Base.Billing_Address.Update(address);
        }

        Base.Actions.PressSave();
    }
}
```

Änderungen an Kontakt- und Adressinformationen durch Code online lesen: https://riptutorial.com/de/acumatica/topic/10617/anderungen-an-kontakt--und-adressinformationen-durch-code

Kapitel 7: Anpassungsmechanismen

Examples

Verwenden von CacheAttached zum Überschreiben von DAC-Attributen im Diagramm

Manchmal müssen Sie ein oder mehrere Attribute eines bestimmten DAC-Felds (Data Access Class) nur für einen bestimmten Bildschirm überschreiben, ohne das vorhandene Verhalten für andere Bildschirme zu ändern.

Alle Attribute ersetzen

Angenommen, die ursprünglichen DAC-Feldattribute werden wie folgt deklariert:

```
public class ARInvoice : IBqlTable
{
    [PXDBDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0")]
    [PXUIField(DisplayName = "Commission Amount")]
    public virtual Decimal? CommnAmt
    {
        get;
        set;
    }
}
```

Die grundlegende Möglichkeit, die Attribute des Felds im Diagramm zu überschreiben, besteht darin, einen `CacheAttached` Ereignishandler in dem Diagramm zu deklarieren, das der Standardkonvention für die Benennung von Diagrammereignissen folgt (beachten Sie das Fehlen des Arguments EventArgs). Der Body des Event-Handlers wird nicht ausgeführt, aber alle *Attribute* , die Sie im Handler platzieren, ersetzen die Attribute im entsprechenden DAC-Feld:

```
[PXDBDecimal(4)]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXUIField(DisplayName = "Commission Amount")]
[PXAdditionalAttribute(NecessaryProperty = true)]
protected virtual void ARInvoice_CommnAmt_CacheAttached(PXCache sender) { }
```

Anhängen eines neuen Attributs an das DAC-Feld

Die im CacheAttached-Handler platzierten Attribute werden **die gesamte Attributgruppe** des DAC neu definiert. Das ist fast immer übertrieben. Beachten Sie, wie Sie im vorherigen Beispiel

alle anderen Attributdeklarationen aus dem DAC kopieren mussten, um dem Feld nur ein einzelnes Attribut hinzuzufügen. Dies führt zu unerwünschter Code-Duplizierung sowie der Möglichkeit, dass DAC und der Graph aus dem Takt geraten. Es ist sehr leicht, sich eine Situation vorzustellen, in der jemand die `PXDefaultAttribute` von beispielsweise `PXDefaultAttribute` im DAC `PXDefaultAttribute` , aber vergisst, alle entsprechenden Attribute zu aktualisieren, die in den verschiedenen `CacheAttached` in verschiedenen Diagrammen platziert sind.

Um dieses Problem zu beheben, bietet das Acumatica Framework ein spezielles Attribut namens `PXMergeAttributesAttribute` . Wenn dieses Attribut in einem `CacheAttached` Handler `CacheAttached` , können Sie die vorhandenen Attribute, die im DAC definiert sind, wiederverwenden.

Anhängen eines Attributs mit `PXMergeAttributesAttribute` :

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXAdditionalAttribute(NecessaryProperty = true)]
protected virtual void ARInvoice_CommnAmt_CacheAttached(PXCache sender) { }
```

Im obigen Beispiel wird der gesamte Satz von Attributen aus dem ursprünglichen DAC wiederverwendet und von allen Attributen angehängt, die Sie für den `CacheAttached` Ereignishandler angegeben haben.

`PXMergeAttributesAttribute` verfügt gemäß den folgenden möglichen Werten für die Method-Eigenschaft über anderes Zusammenführungsverhalten:

- `MergeMethod.Replace` ersetzt die Attribute des DAC vollständig (entspricht dem Fehlen von `PXMergeAttributesAttribute`).
- `MergeMethod.Append` hängt die Attribute des `CacheAttached` an die ursprünglichen DAC-Attribute an.
- `MergeMethod.Merge` ähnelt `Append` ; Er prüft jedoch auch, ob widersprüchliche Attribute zwischen den Handlerattributen und den DAC-Feldattributen vorliegen. Wenn ein Konflikt `CacheAttached` , hat das `CacheAttached` Attribut Vorrang und das entsprechende DAC-Attribut wird verworfen.

Überschreiben einer einzelnen Eigenschaft eines Attributs

Ein sehr häufiges Anwendungsentwicklungsszenario tritt auf, wenn Sie nur eine einzelne Eigenschaft eines DAC-Attributs für einen bestimmten Bildschirm neu definieren müssen. Berücksichtigen Sie die Situation, wenn Sie die `DisplayName` Eigenschaft des PXUIFieldAttribute definieren müssen.

Zu diesem Zweck können Sie ein weiteres spezielles Attribut verwenden, das vom Acumatica Framework bereitgestellt wird: `PXCustomizeBaseAttributeAttribute` . Sein Konstruktor akzeptiert drei Werte:

- Der Typ des DAC-Attributs, dessen Eigenschaft überschrieben werden muss

- Der Name der zu überschreibenden Eigenschaft des Attributs (verwenden Sie den Operator `nameof` in C # 6.0, um die `nameof`)
- Der neue Wert für die angegebene Eigenschaft.

Nehmen wir an, dass es eine Anforderung ist die UI - Anzeigename von *Kommission Betrag* an *Basiswährung Kommission* für nur einen Bildschirm zu ändern. Das folgende Codebeispiel veranschaulicht, wie das gewünschte Verhalten implementiert wird.

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXCustomizeBaseAttribute(typeof(PXUIFieldAttribute), nameof(PXUIFieldAttribute.DisplayName),
"Base Currency Commission")]
protected virtual void ARInvoice_CommnAmt_CacheAttached(PXCache sender) { }
```

In diesem Beispiel stellt `PXMergeAttributes` sicher, dass die ursprünglichen DAC-Attribute beibehalten werden. Mit `PXCustomizeBaseAttribute` kann der Software-Ingenieur den Anzeigenamen des UI-Felds für das betreffende Diagramm überschreiben.

Ein Attribut durch ein anderes Attribut ersetzen

Angenommen, es besteht die Notwendigkeit, das `PXDefaultAttribute` eines DAC-`PXDefaultAttribute` durch `PXDBDefaultAttribute` für nur einen Bildschirm zu ersetzen.

Dies kann auf folgende Weise erreicht werden:

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXRemoveBaseAttribute(typeof(PXDefaultAttribute))]
[PXDBDefault(typeof(SOShipment.siteID), PersistingCheck = PXPersistingCheck.Nothing)]
protected void SOOrderShipment_SiteID_CacheAttached(PXCache sender) { }
```

Anwendungsreihenfolge der Attribute-Customizing-Attribute

1. `PXCustomizeBaseAttribute`
2. `PXRemoveBaseAttribute`
3. `PXMergeAttributes`

Anpassungsmechanismen online lesen:
https://riptutorial.com/de/acumatica/topic/9751/anpassungsmechanismen

Kapitel 8: Anzeigen eines Fehlers, der die Eingabe von Entitätsdaten erfordert

Examples

Anzeigen eines Fehlers, der den Benutzer zur Eingabe von Entitätsdaten auffordert

Benutzer erscheinen häufig in einer Situation, in der ein Geschäftsprozess nicht abgeschlossen werden kann, weil der Benutzer nicht alle erforderlichen Informationen eingegeben hat.

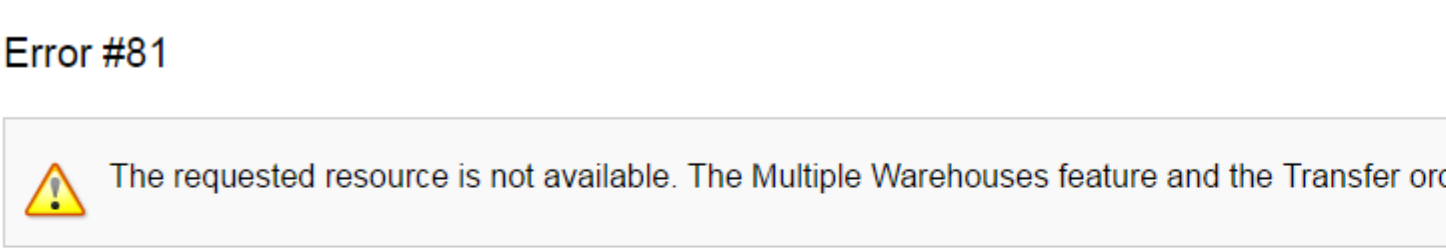
Ein Beispiel für diese Situation ist, wenn ein Benutzer versucht, einen Drop-Ship-Auftrag mit fehlender Kundenadresse zu erstellen.

Gemäß den Best Practices von UX sollte das System für den Benutzer freundlich sein und den Benutzer nicht nur über die Situation informieren, sondern ihn auch zur Lösung seines Problems führen. Wie wir wissen, verfügt das System bereits über einen ähnlichen Mechanismus, der durch `PXSetup<TSetup>.Current` aktiviert wird, wenn keine Datensätze in der `TSetup` Tabelle vorhanden sind. Es wird intern implementiert, indem eine `PXSetupNotEnteredException` .

Vor kurzem wurde dieser Ausnahme eine neue Funktionalität hinzugefügt, die es einem Anwendungsentwickler ermöglicht, einen Fehler mit einer Verknüpfung zu der Entität auszulösen, die neu konfiguriert werden muss:

```
INSite erroneousSite = PXSelect<
    INSite,
    Where<
        INSite.siteID, Equal<Current<SOCreateFilter.siteID>>,
        And<INSite.active, Equal<True>,
        And<Where<INSite.addressID, IsNull, Or<INSite.contactID, IsNull>>>>>>
    .SelectSingleBound(this, new object[] { e.Row });

if (erroneousSite != null)
{
    throw new PXSetupNotEnteredException<INSite, INSite.siteCD>(
        Messages.WarehouseWithoutAddressAndContact,
        erroneousSite.SiteCDlnk,
        erroneousSite.SiteCDinf);
}
```

Das Ergebnis wird dem Benutzer folgendermaßen angezeigt:



- Als erster `PXSetupNotEnteredException` akzeptiert `PXSetupNotEnteredException` den Typ der Entität, zu der die Standardgraphenverknüpfung generiert wird.
- Der zweite Typparameter gibt das Schlüsselfeld des Datensatzes an, das zur Erzeugung der Verknüpfung verwendet werden soll. Im obigen Beispiel wird die Navigation zur Lagerentität über den CD-Schlüssel vorgenommen.
- Das erste Konstruktorargument ist die Formatzeichenfolge für die Fehlernachricht. Die Nummerierung der internen Platzhalter sollte mit 1 beginnen: dh `The Multiple Warehouses feature and the Transfer order type are activated in the system, in this case an address and a contact must be configured for the '{1}' warehouse.`
- Das zweite Konstruktorargument ist der Wert des Schlüsselfelds, das als zweiter generischer Parameter angegeben ist. In diesem Beispiel würde der Link `/IN204000.aspx?siteCD=erroneousSite.SiteCDlnk` generiert.
- Das dritte Konstruktorargument ist der in der Fehlernachricht anzuzeigende, vom Menschen lesbare Wert: `...in this case an address and a contact must be configured for the 'erroneousSite.SiteCDinf' warehouse.`

Anzeigen eines Fehlers, der die Eingabe von Entitätsdaten erfordert online lesen: https://riptutorial.com/de/acumatica/topic/9274/anzeigen-eines-fehlers--der-die-eingabe-von-entitatsdaten-erfordert

Kapitel 9: Bedingt Tabs ausblenden

Einführung

In diesem Thema werden zwei Ansätze zum bedingten Ausblenden von Registerkarten auf den Dateneingabebildschirmen in Acumatica erläutert.

Examples

VisibleExp-Eigenschaft des PXTab-Steuerelements in Aspx

Die **VisibleExp-** Eigenschaft ist ein boolescher Ausdruck, der bestimmt, ob die angegebene Registerkarte sichtbar ist (wenn der logische Ausdruck TRUE ist) oder ausgeblendet ist. Sie geben die **VisibleExp-** Eigenschaft für PXTab-Steuerelemente auf der Aspx-Seite an:

```
<px:PXTabItem Text="Credit Card Processing Info" BindingContext="form"
    VisibleExp="DataControls[&quot;chkIsCCPayment&quot;].Value = 1">
```

VisibleExp besteht aus Eingabesteuerelementen, die innerhalb des Containers platziert werden und deren ID in der **BindingContext-** Eigenschaft des PXTab-Steuerelements angegeben ist. Sie dürfen Eingabesteuerelemente nicht aus mehr als einem Container verwenden. Der Zugriff auf eine bestimmte Eingabesteuerung erfolgt über das `DataControls` Wörterbuch über seine ID und nicht über den Namen eines DAC-Felds.

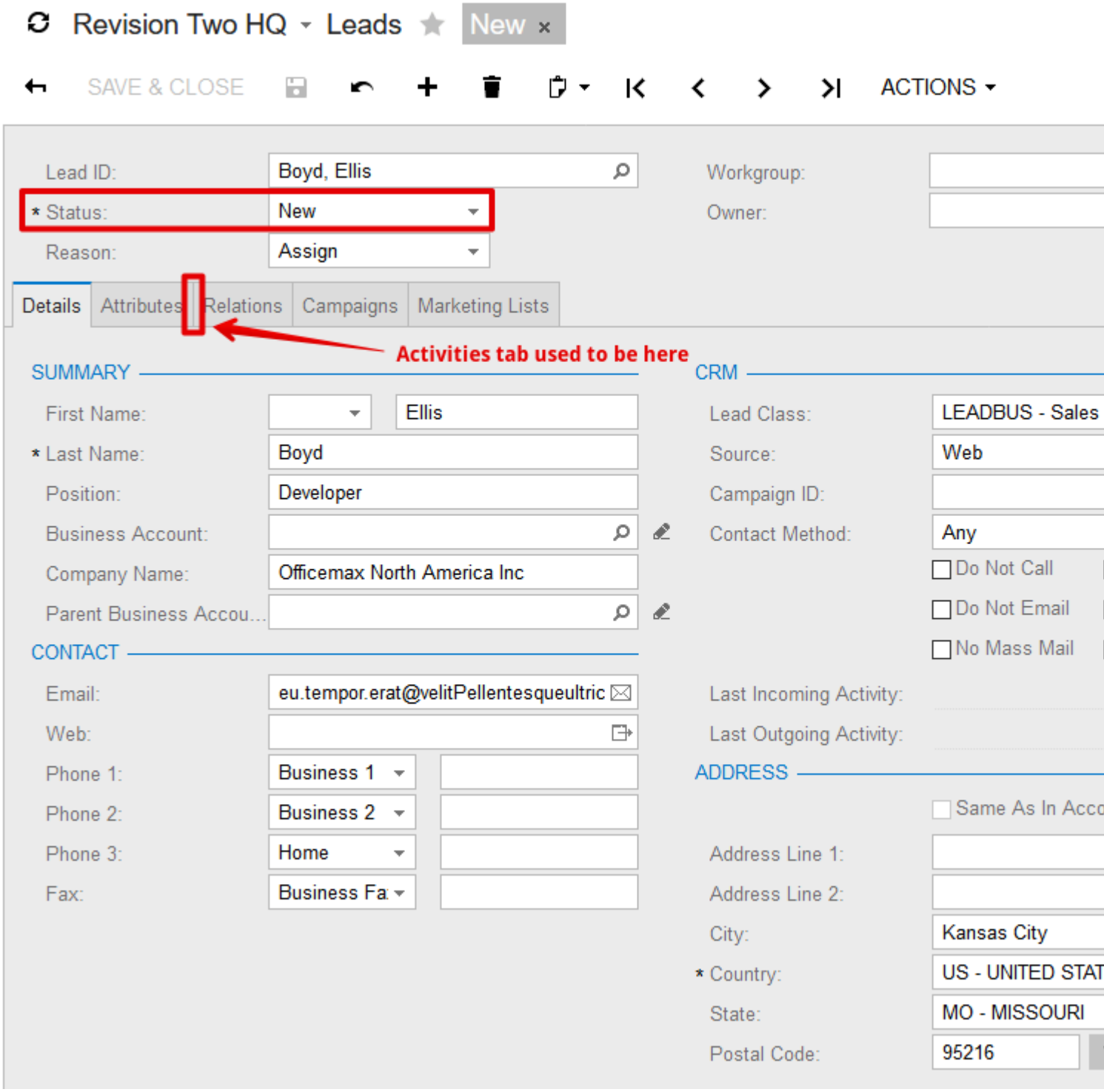
Normalerweise wird die **VisibleExp-** Eigenschaft verwendet, um relativ einfache boolesche Ausdrücke mit fest codierten Eingabesteuerungswerten zu **erstellen** , die sich mit der Zeit unwahrscheinlich ändern. Zum Beispiel wird der folgende Ausdruck auf den **Kundenaufträge** Bildschirm (SO.30.10.00) verwendet , um **Zahlungs** Registerkarte **Einstellung** für Aufträge der ***Übertragungsart*** zu verstecken:

```
<px:PXTabItem Text="Payment Settings"
    VisibleExp="DataControls[&quot;edOrderType&quot;].Value!=TR" BindingContext="form">
```

So blenden Sie die Registerkarte Aktivitäten für Status mit Leads mit Neu aus

So blenden Registerkarte **Aktivitäten** aus der **Leads** Bildschirm (CR.30.10.00), stellen ***Binding*** Eigenschaft ***bilden*** (Top-Level - *Blei Zusammenfassung* Formular ***Formular*** - ID hält) und definieren **VisibleExp** FALSE zurück , wenn Lead - Status Offen ist *(Stand* Dropdown hält ***edStatus*** ID) :

```
<px:PXTabItem Text="Activities" LoadOnDemand="True"
    BindingContext="form" VisibleExp="DataControls[&quot;edStatus&quot;].Value != H">
```



AllowSelect-Eigenschaft in Datenansichten

Im Gegensatz zu der in **Aspx** definierten Eigenschaft VisibleExp bearbeiten Sie die **AllowSelect-** Eigenschaft einer Datenansicht mit **BLC-** oder BLC-Erweiterungscode. Die **AllowSelect-** Eigenschaft ermöglicht die Verwendung komplexerer boolescher Ausdrücke (im Vergleich zur **VisibleExp-** Eigenschaft) und **ruft** gegebenenfalls zusätzliche Informationen aus der Datenbank oder anderen Quellen ab, die auf einer Webseite nicht verfügbar sind.

Im Folgenden sind die drei häufigsten Szenarien aufgeführt, in denen Sie mit der **AllowSelect-** Eigenschaft arbeiten können:

- **RowSelected** Ereignishandler für Top-Level - Einheit Registerkarte **Anwendungen** für Rechnungen von ***Barverkauf*** und ***Cash*** - ***Return*** - Typ zu verstecken:

```
 public class SOInvoiceEntry : ARInvoiceEntry
 {
     ...
     protected override void ARInvoice_RowSelected(PXCache cache, PXRowSelectedEventArgs
e)
     {
         ...

         Adjustments.AllowSelect =
             doc.DocType != ARDocType.CashSale &&
             doc.DocType != ARDocType.CashReturn;
     }
     ...
 }
```

- BLC Konstruktor **Unterposten Replenishment Info** auf der Registerkarte **Artikellager Detailansicht** nur den Bildschirm zu übertragen, wenn beide *Bestandsauffüllung* und *Bestandsunteroptionen* Funktionen aktiviert sind:

```
 public class INItemSiteMaint : PXGraph<INItemSiteMaint, INItemSite>
 {
     ...
     public INItemSiteMaint()
     {
         ...

         bool enableSubItemReplenishment =
PXAccess.FeatureInstalled<FeaturesSet.replenishment>() &&
PXAccess.FeatureInstalled<FeaturesSet.subItem>();
         subitemrecords.AllowSelect = enableSubItemReplenishment;
     }
     ...
 }
```

- **RowSelected-** Handler für eine Entität auf oberster Ebene, um die Registerkarte " **Abschreibungsverlauf"** auszublenden, sofern das aktuelle Asset nicht abschreibbar ist und die **Ansicht "Abschreibungsverlauf"** in den **Voreinstellungen** für festgelegte **Anlagen** auf " ***Nebeneinander" gesetzt ist*** :

```
 public class AssetMaint : PXGraph<AssetMaint, FixedAsset>
 {
     ...
     protected virtual void FixedAsset_RowSelected(PXCache sender, PXRowSelectedEventArgs
e)
     {
         ...

         AssetHistory.AllowSelect = asset.Depreciable == true &&
fasetup.Current.DeprHistoryView == FASetup.deprHistoryView.SideBySide;
     }
     ...
 }
```

Jedes Mal, **wenn die AllowSelect-** Eigenschaft verwendet wird, um die Sichtbarkeit der Registerkarten durch BLC- oder BLC-Erweiterungscode bedingt zu ändern, müssen Sie die **RepaintOnDemand-** Eigenschaft in Aspx für den entsprechenden PXTab-Container auf ***false*** setzen:

```
<px:PXTabItem Text="Depreciation History" RepaintOnDemand="false">
```

Die **RepaintOnDemand-** Eigenschaft ist standardmäßig ***true*** . Diese Eigenschaft steuert die Initialisierung des PXTab-Containers: Wenn diese Eigenschaft auf ***true festgelegt ist*** , wird PXTab erst initialisiert, wenn es von einem Benutzer ausgewählt wurde. Offensichtlich muss **RepaintOnDemand** auf ***false gesetzt sein*** , um das korrekte Verhalten des angegebenen PXTab-Containers zu gewährleisten, auch wenn er ausgewählt wurde oder nicht.

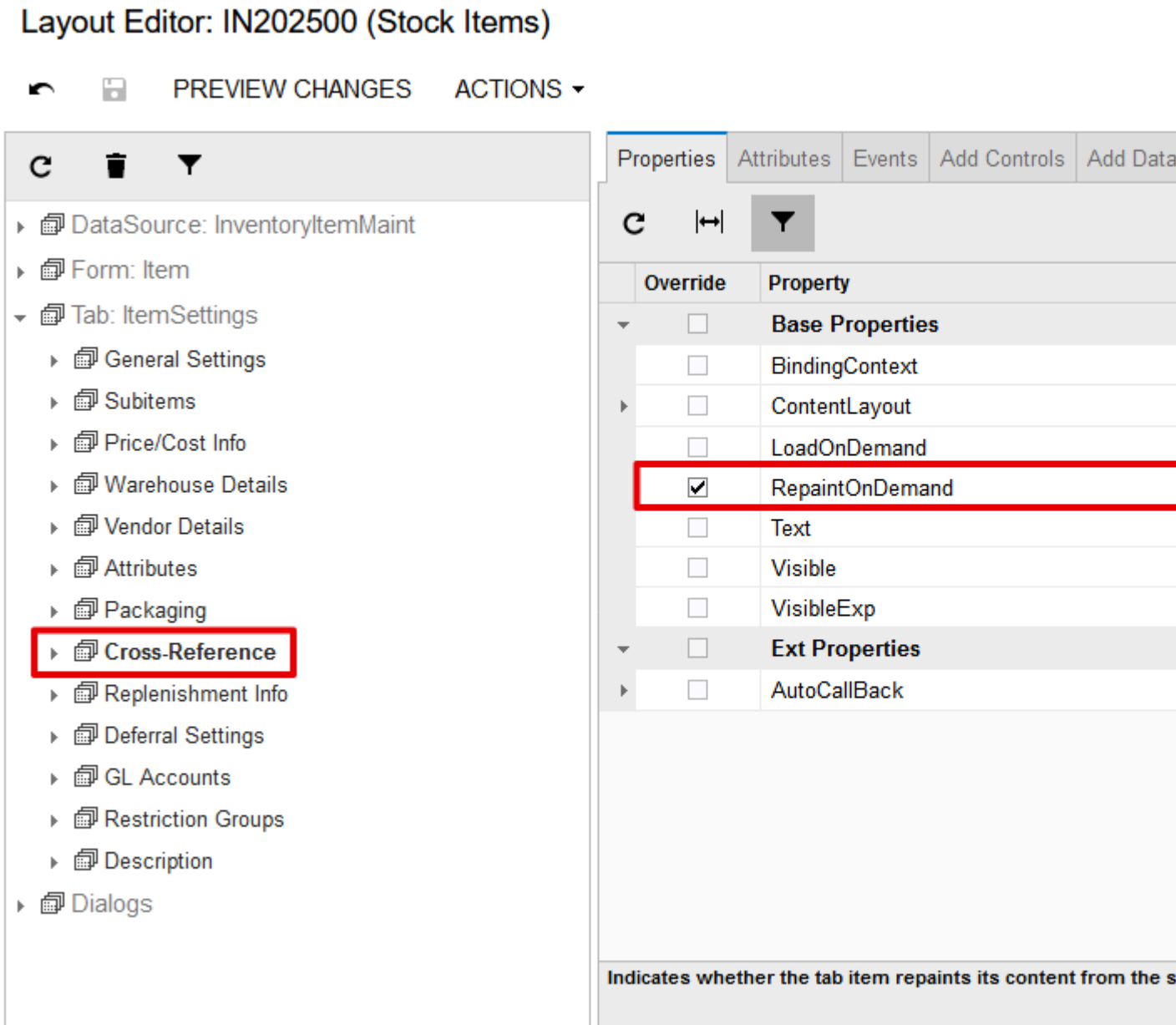
So blenden Sie die Registerkarte "Querverweis" für Lagerartikel aus, die nicht verkauft werden können

So blenden **Querverweis** Registerkarte aus dem **Stock Items** Bildschirm (IN.20.25.00) für Artikel ***ohne Verkaufsstand,*** gehen Sie wie folgt vor :

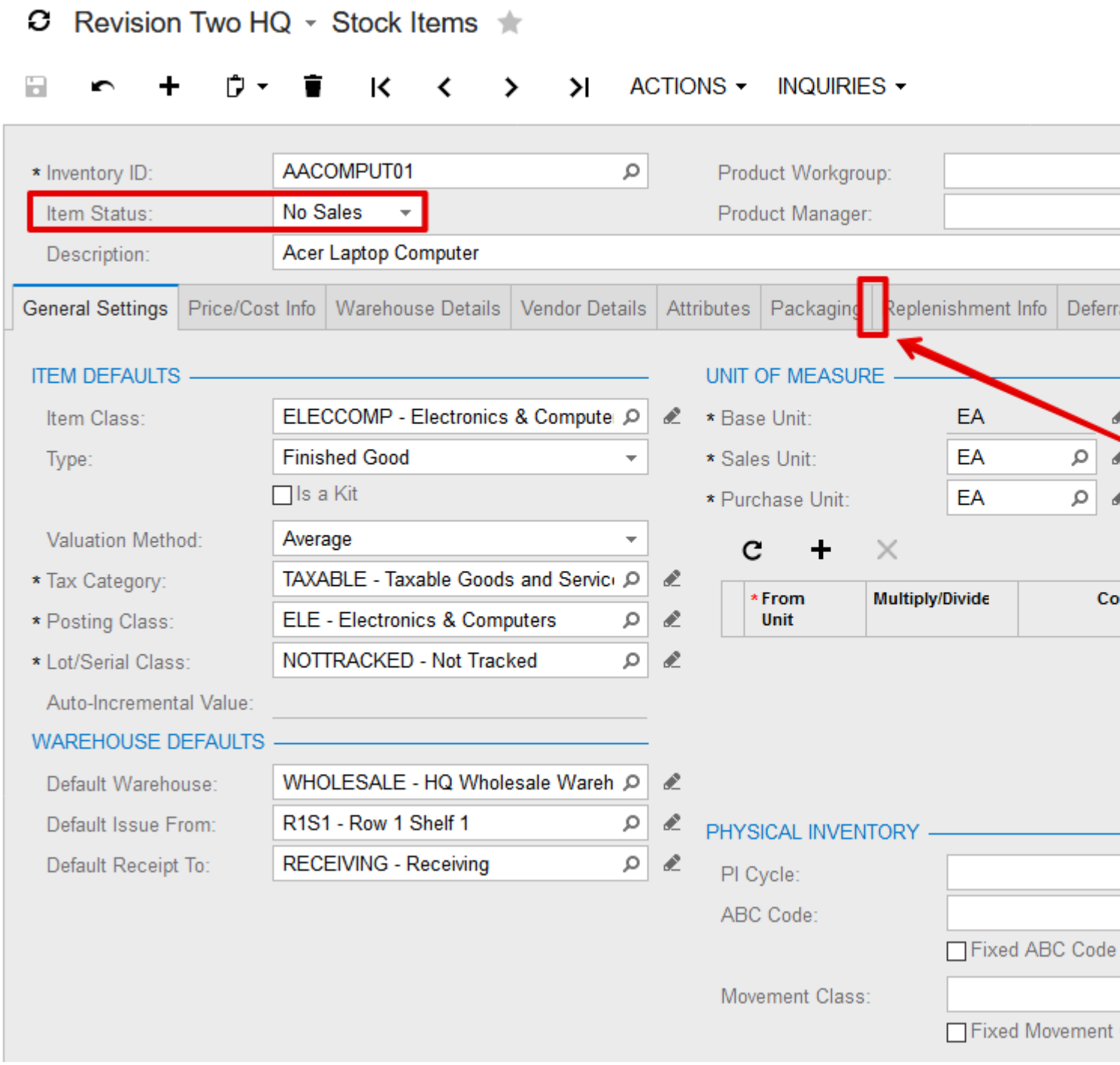
1. Implementieren Sie **InventoryItem_RowSelected-** Handler in der BLC-Erweiterung InventoryItemMaint, um die **AllowSelect-** Eigenschaft für die `itemxrefrecords` **itemxrefrecords** auf ***false*** zu `itemxrefrecords` , wenn für **Artikelstatus *No Sales festgelegt wurde*** .

   ```
   public class InventoryItemMaintExt : PXGraphExtension<InventoryItemMaint>
   {
       protected void InventoryItem_RowSelected(PXCache sender, PXRowSelectedEventArgs e)
       {
           InventoryItem item = (InventoryItem)e.Row;
           if (item == null) return;

           Base.itemxrefrecords.AllowSelect = (item.ItemStatus !=
   InventoryItemStatus.NoSales);
       }
   }
   ```

2. in Customization Managern, **RepaintOnDemand** Eigenschaft auf ***False*** für den **Querverweis** Registerkarte und veröffentlicht personalisierbar:



Nachdem Sie zwei ganz einfache Schritte oben abgeschlossen ist , sollte der **Querverweis** Registerkarte nicht für Lagerware ***ohne Verkaufsstand*** zugänglich sein:



Registerkarte "Attribute" für inaktive Bestandsartikel ausblenden

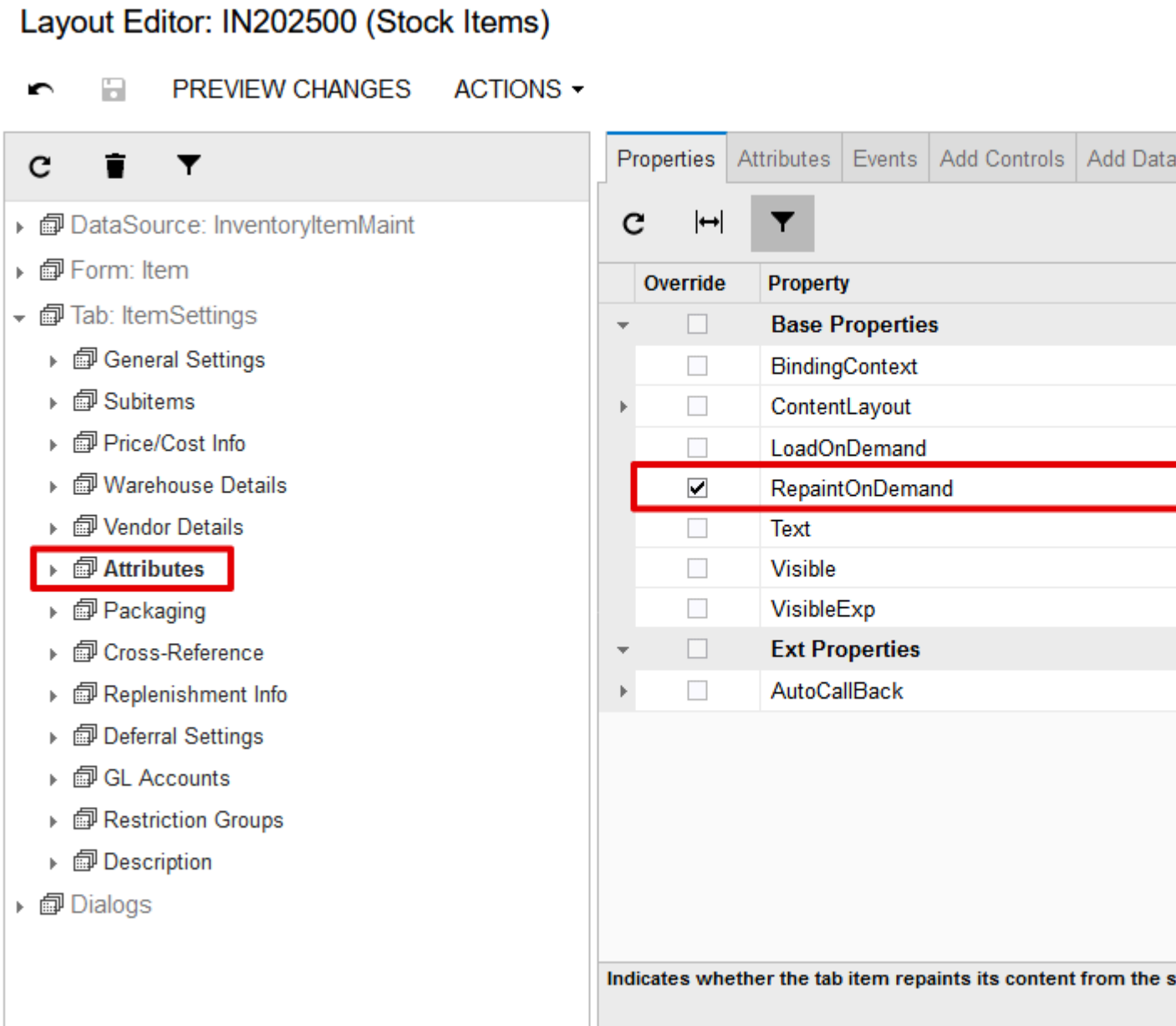
Gehen Sie wie folgt vor, um die Registerkarte ** Attribute ** im Bildschirm mit den **Bestandspositionen** (IN.20.25.00) bedingt auszublenden:

1. **InventoryItem_RowSelected** Handler in der Verlängerung InventoryItemMaint BLC implementieren **AllowSelect** Eigenschaft auf ***False*** für die festlegen `Answers` und `Category` Datenansichten , wenn **Einzelteil** - **Status** auf ***Inaktiv*** gesetzt wurde. `PXUIFieldAttribute` auch, dass die Eigenschaft **Visible** für `PXUIFieldAttribute` auf ***false*** gesetzt ist, die vom

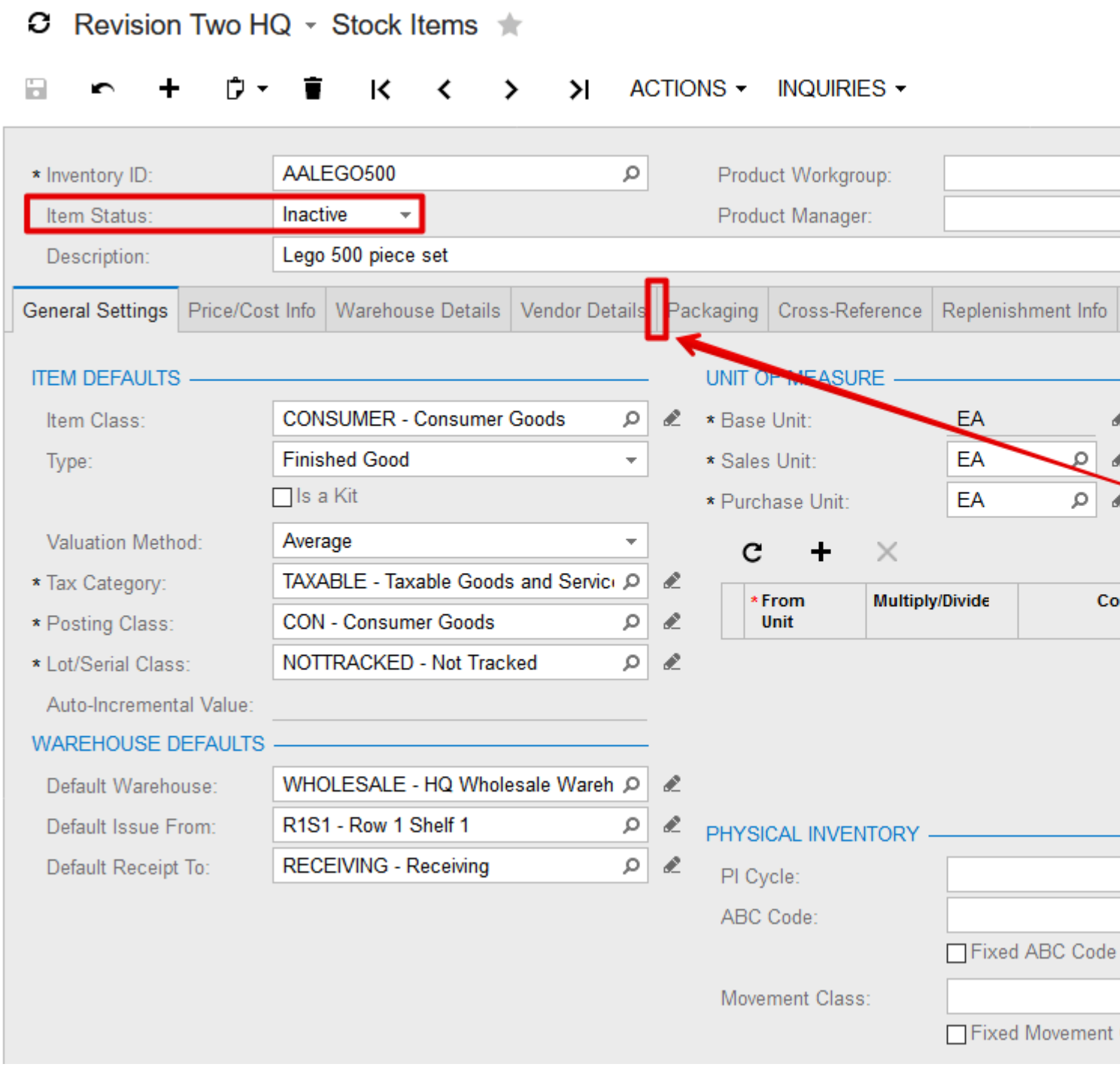
CacheAttached- Handler im Feld `InventoryItem.ImageUrl` **hinzugefügt wurde** :

```
public class InventoryItemMaintExt : PXGraphExtension<InventoryItemMaint>
{
    protected void InventoryItem_RowSelected(PXCache sender, PXRowSelectedEventArgs e)
    {
        InventoryItem item = (InventoryItem)e.Row;
        if (item == null) return;

        bool showAttributesTab = item.ItemStatus != InventoryItemStatus.Inactive;
        Base.Answers.AllowSelect = Base.Category.AllowSelect = showAttributesTab;
        PXUIFieldAttribute.SetVisible<InventoryItem.imageUrl>(sender, item,
showAttributesTab);
    }

    [PXMergeAttributes(Method = MergeMethod.Append)]
    [PXUIField(DisplayName = "Image")]
    protected void InventoryItem_ImageURL_CacheAttached(PXCache sender)
    { }
}
```

2. Setzen **Sie** im Anpassungsmanager die **RepaintOnDemand-** Eigenschaft für die Registerkarte **Attribute** auf ***false*** und veröffentlichen Sie die Anpassung:



Nachdem Sie die beiden obigen Schritte ausgeführt haben, sollte die Registerkarte " **Attribute"** für Bestandsartikel mit ***inaktivem*** Status nicht ***verfügbar sein*** :



Bedingt Tabs ausblenden online lesen: https://riptutorial.com/de/acumatica/topic/9506/bedingt-tabs-ausblenden

Kapitel 10: Bei der Veröffentlichung wurde der bereits angewendete Anpassungsinhalt nicht berücksichtigt

Einführung

Beim Veröffentlichen eines Anpassungsprojekts wird möglicherweise ein Element übersprungen, weil es bereits angewendet wurde. Ex:

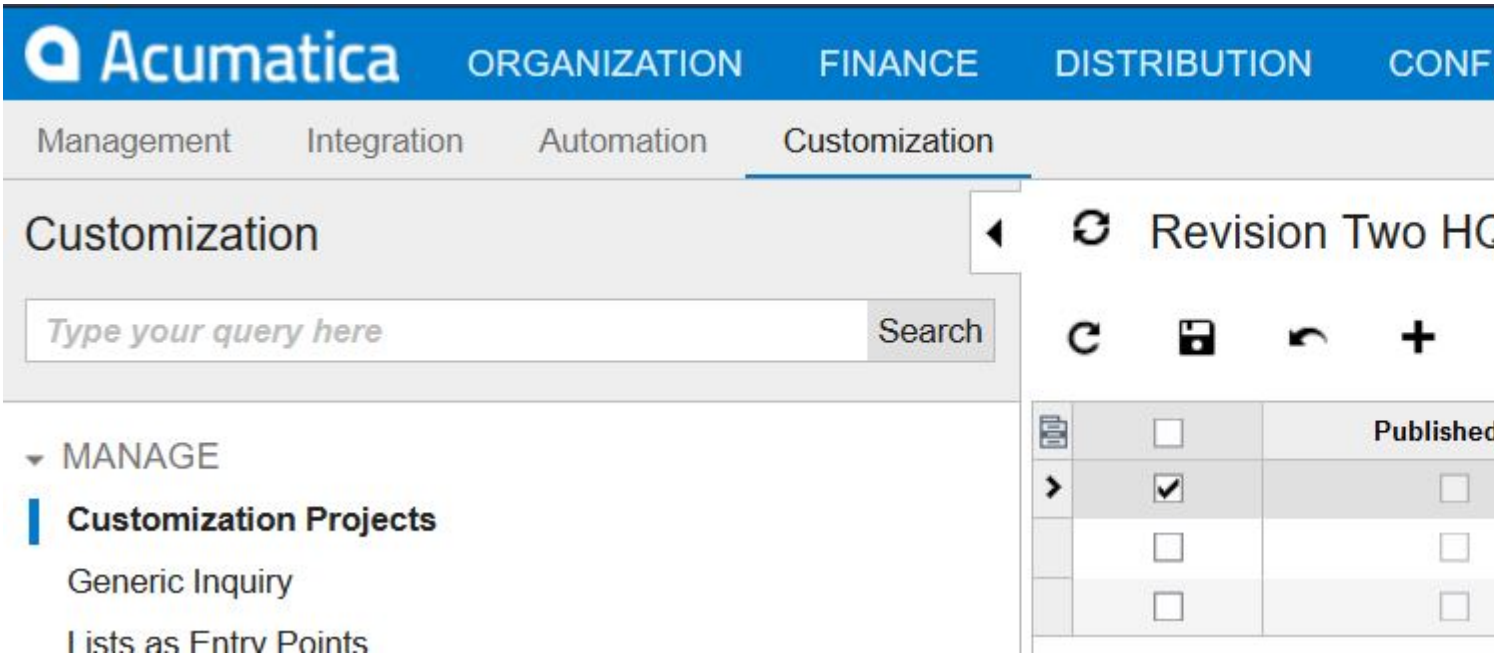
EntityEndpoint EntityEndpoint # 6.00.001§DefaultPlus (übersprungen, bereits angewendet)

Dies kann für alle in der Datenbank gespeicherten Elemente passieren. Beispiel: Generische Abfragen, Berichte, Site-Map-Knoten, DB-Skripts, Systemumgebungen, Import- / Exportszenarien, gemeinsame Filter, Zugriffsrechte, Wikis, Web-Service-Endpunkte und Analyseberichte.

Examples

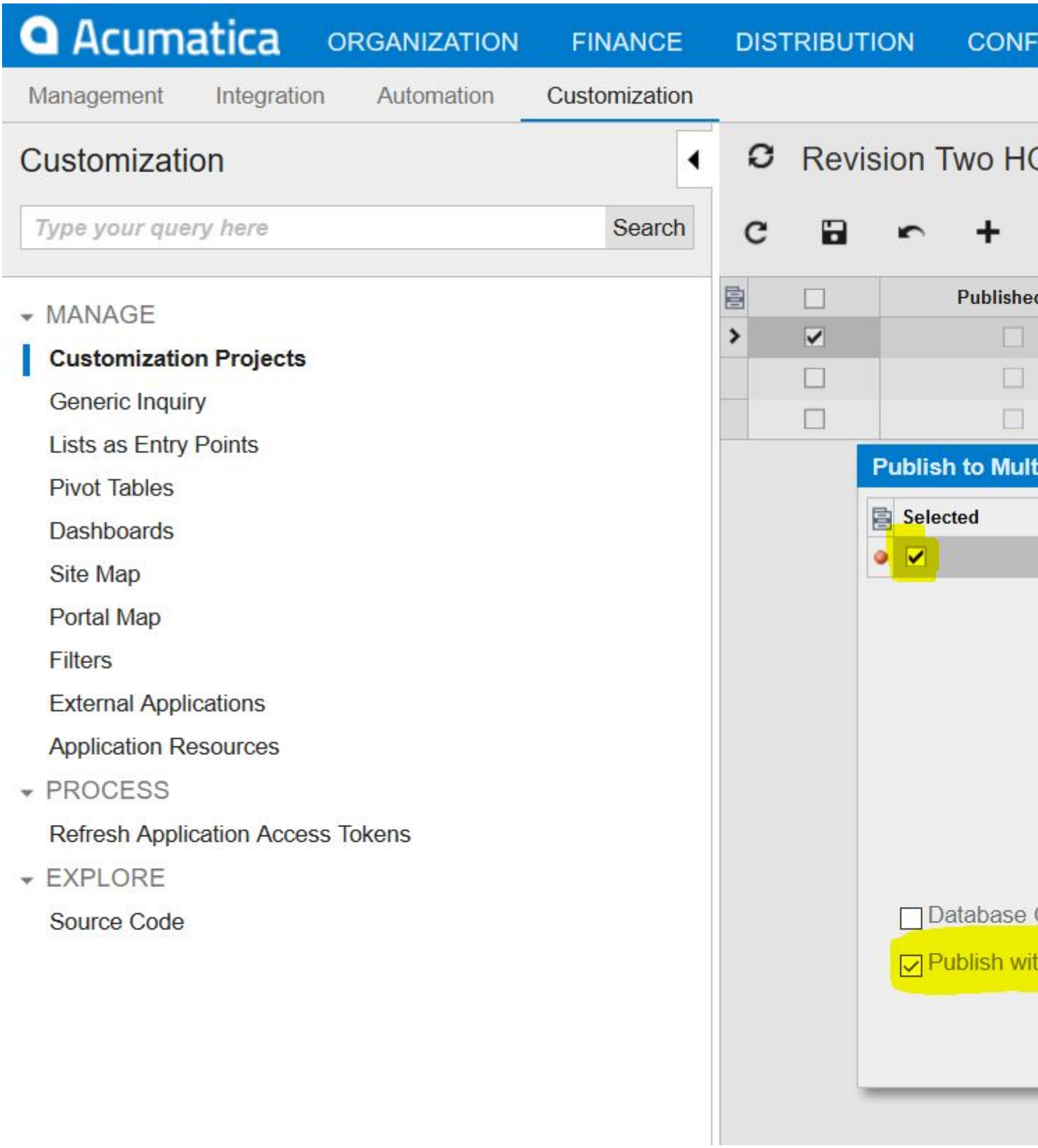
Veröffentlichen Sie mit Bereinigung im Anpassungsbildschirm

1. Sie müssen natürlich das Projekt auswählen, das Sie veröffentlichen möchten.
2. Sie müssen auf den kleinen Pfeil rechts neben der Schaltfläche "Veröffentlichen" klicken.
3. Sie müssen auf die Option "In mehreren Unternehmen veröffentlichen" klicken.



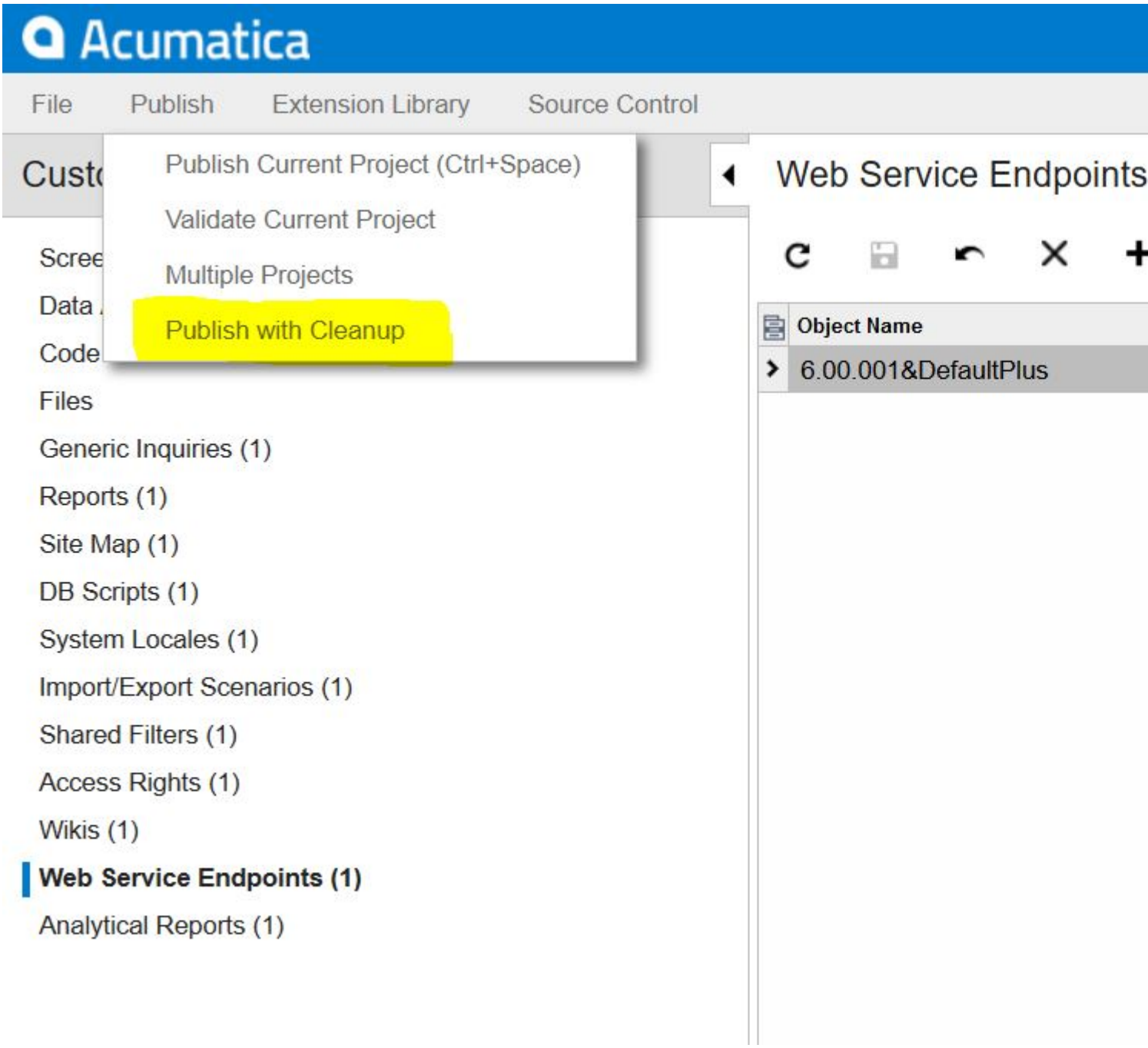
4. Auf dem Smart-Panel, das angezeigt wird, müssen Sie die Unternehmen auswählen, deren Projekte Sie veröffentlichen möchten. Es ist auch nur eine Firma möglich.
5. Aktivieren Sie das Kontrollkästchen "Mit Bereinigung veröffentlichen", um sicherzustellen, dass alle im Anpassungsprojekt vorhandenen Elemente erneut angewendet werden, wobei

das bereits vorhandene durch die neuere Version ersetzt wird.



Veröffentlichen Sie mit einem Cleanup in einem Anpassungsprojekt

1. Öffnen Sie das Anpassungsprojekt, das Sie mit dieser Methode veröffentlichen möchten.
2. Öffnen Sie oben das Veröffentlichungsmenü und wählen Sie die Option "Mit Bereinigung veröffentlichen".



* Beachten Sie, dass alle Anpassungsprojekte, die auf dem Anpassungsbildschirm ausgewählt werden, auch dann erneut veröffentlicht werden, wenn Sie sich nur in einem einzigen Projekt befinden.

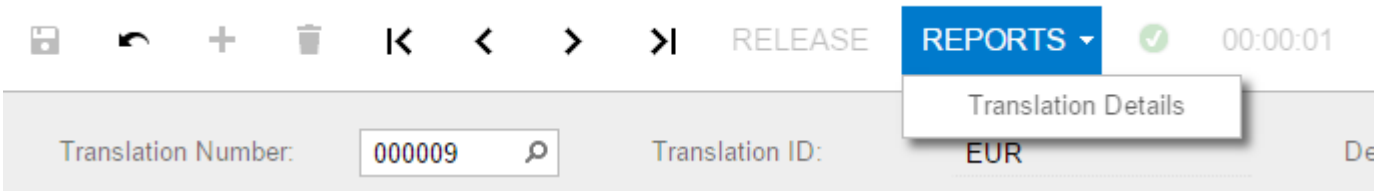
Bei der Veröffentlichung wurde der bereits angewendete Anpassungsinhalt nicht berücksichtigt online lesen: https://riptutorial.com/de/acumatica/topic/10155/bei-der-veroffentlichung-wurde-der-bereits-angewendete-anpassungsinhalt-nicht-berucksichtigt

Kapitel 11: Benutzeroberflächen-Techniken

Examples

Erstellen eines Dropdown-Menüs für einen Bildschirm

Angenommen, Sie müssen ein Dropdown-Menü für einen bestimmten Acumatica-Bildschirm definieren, z. B. das Menü Berichte in der folgenden Abbildung.



Dies kann auf drei verschiedene Arten erreicht werden:

- Durch Hinzufügen einer Symbolleiste mit einem Menüelement zum ASPX des Bildschirms
- Durch Deklarieren einer speziellen "Ordner" -Aktion in der Grafik und Hinzufügen von Menüelementen im Code
- Verwendung des Automatisierungs-Subsystems des Acumatica Framework (nicht in diesem Beispiel enthalten)

Option 1: Erstellen eines Dropdown-Menüs in ASPX

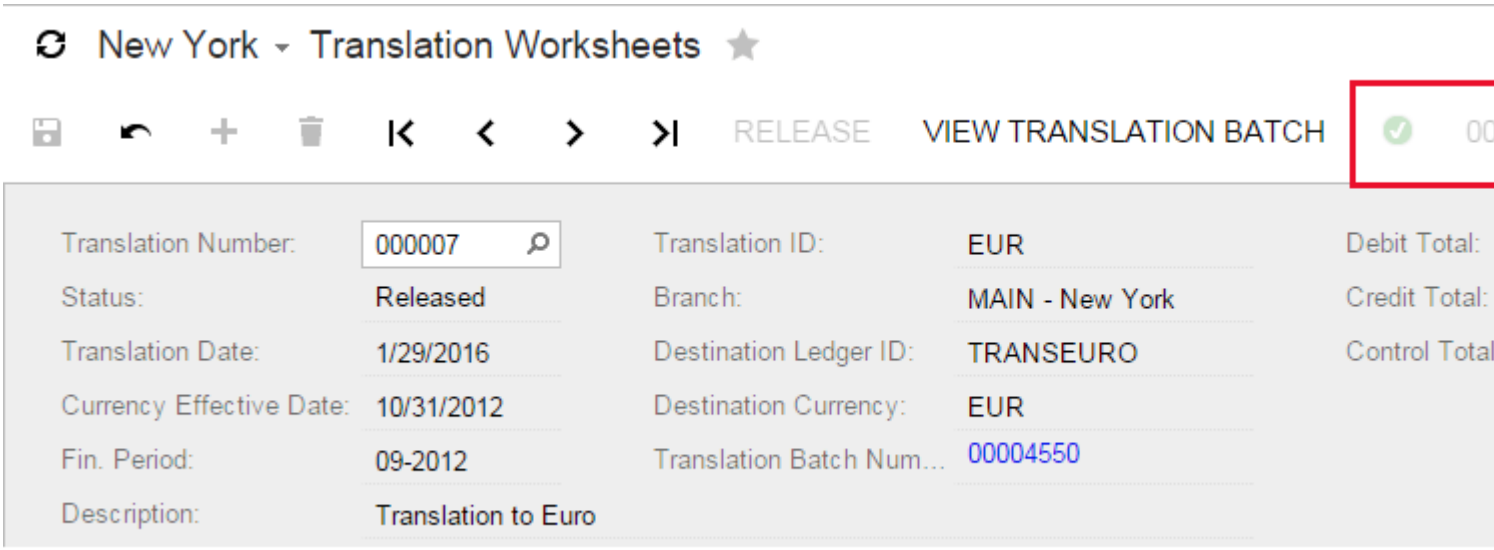
Stellen Sie zunächst sicher, dass das PXDataSource-Element der ASPX-Seite alle erforderlichen Befehle enthält, die den Diagrammaktionen entsprechen, die Sie ausführen möchten, wenn Sie auf ein Menüelement klicken.

```
<px:PXDataSource
    ID="ds" runat="server" Visible="True" PrimaryView="TranslHistRecords"
TypeName="PX.Objects.CM.TranslationHistoryMaint">
    <CallbackCommands>
        ...
        <px:PXDSCallbackCommand Name="TranslationDetailsReport" Visible="False"/>
        ...
    </CallbackCommands>
</px:PXDataSource>
```

Fügen Sie anschließend ein benutzerdefiniertes Symbolleistenelement direkt nach dem PXDataSource-Element hinzu. Definieren Sie darin einen PXToolbarButton mit den gewünschten Dropdown-Menüelementen, die auf die entsprechenden Datenquellenbefehle verweisen, wie im folgenden Code gezeigt.

```
<px:PXToolBar ID="toolbar1" runat="server" SkinID="Navigation" BackColor="Transparent"
CommandSourceID="ds">
    <Items>
        <px:PXToolBarButton Text="Reports">
            <MenuItems>
                <px:PXMenuItem Text="Translation Details" CommandSourceID="ds"
CommandName="TranslationDetailsReport"/>
            </MenuItems>
        </px:PXToolBarButton>
    </Items>
    <Layout ItemsAlign="Left" />
</px:PXToolBar>
```

Diese Option kann aufgrund ihrer Einfachheit verführerisch erscheinen. Es gibt jedoch einen **wichtigen Nachteil** . Wenn Sie eine solche Dropdown-Liste auf einem Bildschirm mit einem Verarbeitungskennzeichen implementieren (z. B. ein Dokumentfreigabebildschirm oder ein Massenverarbeitungsbildschirm), wird das Symbol links neben Ihrem Dropdown-Menü angezeigt (siehe unten).



Wenn dies nicht erwünscht ist, können Sie ein Dropdown-Menü im Code definieren, wie im Abschnitt Option 2 beschrieben.

Option 2: Erstellen eines Menüs in der Grafik

Zuerst deklarieren Sie in der Seitengrafik eine "Ordner" -Aktion, die der Dropdown-Menüschaltfläche entspricht.

```
public PXAction<TranslationHistory> reportsFolder;
[PXUIField(DisplayName = "Reports", MapEnableRights = PXCacheRights.Select)]
[PXButton(SpecialType = PXSpecialButtonType.Report)]
protected virtual IEnumerable Reportsfolder(PXAdapter adapter)
{
    return adapter.Get();
}
```

Geben Sie anschließend im Konstruktor des Diagramms an, dass die Aktion tatsächlich ein Dropdown-Menü ist, und fügen Sie alle Aktionen hinzu, die als Menüelemente angezeigt werden müssen (siehe unten).

```
public TranslationHistoryMaint()
{
    this.reportsFolder.MenuAutoOpen = true;
    this.reportsFolder.AddMenuAction(this.translationDetailsReport);
}
```

Wenn Sie diesen Ansatz wählen, wird das Verarbeitungskennzeichen immer rechts von Ihrem Menü angezeigt, was wahrscheinlich eine bessere UX ist.

Benutzeroberflächen-Techniken online lesen:
https://riptutorial.com/de/acumatica/topic/10150/benutzeroberflachen-techniken

Kapitel 12: Bericht mit Daten durch Code füllen

Examples

Dieser Artikel beschreibt ein Beispiel, das zeigt, wie ein Bericht mithilfe von Speicherdatensätzen erstellt wird:

In diesem Beispiel wird gezeigt, wie der Bericht mit Daten gefüllt wird, die von einem Datensichtedelegierten zurückgegeben werden. Während der Übung entwickeln wir ein Anfragedisplay mit einer Liste von Verkaufsaufträgen zwischen zwei Terminen. Der Datenansichtsdelegierte wird zum Auffüllen der Verkaufsauftragsinformationen verwendet.

Voraussetzungen:

1. Wir beginnen mit der Deklaration des SOOrderFilter DAC:

```
[Serializable]
public class SOOrderFilter : IBqlTable
{
    public abstract class dateFrom : IBqlField
    {
    }
    [PXDate()]
    [PXUIField(DisplayName = "Date From")]
    public DateTime? DateFrom { get; set; }

    public abstract class dateTo : IBqlField
    {
    }
    [PXDate()]
    [PXUIField(DisplayName = "Date To")]
    public DateTime? DateTo { get; set; }
}
```

2. Fahren Sie mit der Deklaration des SOOrderData-DAC fort:

```
[Serializable]
public class SOOrderData : IBqlTable
{
    #region OrderType
    public abstract class orderType : PX.Data.IBqlField
    {
    }
    [PXString(2, IsKey = true, IsFixed = true)]
    [PXUIField(DisplayName = "Type")]
    public virtual string OrderType { get; set; }
    #endregion
    #region OrderNbr
    public abstract class orderNbr : PX.Data.IBqlField
    {
```

```
    }
    [PXString(15, IsKey = true, IsUnicode = true, InputMask = ">CCCCCCCCCCCCCCC")]
    [PXUIField(DisplayName = "Order Nbr.")]
    public virtual string OrderNbr { get; set; }
    #endregion
    #region OrderDate
    public abstract class orderDate : PX.Data.IBqlField
    {
    }
    [PXDate]
    [PXUIField(DisplayName = "Date")]
    public virtual DateTime? OrderDate { get; set; }
    #endregion
    #region Status
    public abstract class status : PX.Data.IBqlField
    {
    }
    [PXString(1, IsFixed = true)]
    [PXUIField(DisplayName = "Status")]
    [SOOrderStatus.List()]
    public virtual string Status { get; set; }
    #endregion
    #region OrderDesc
    public abstract class orderDesc : PX.Data.IBqlField
    {
    }
    [PXString(60, IsUnicode = true)]
    [PXUIField(DisplayName = "Description", Visibility = PXUIVisibility.SelectorVisible)]
    public virtual string OrderDesc { get; set; }
    #endregion
    #region OrderTotal
    public abstract class orderTotal : PX.Data.IBqlField
    {
    }
    [PXDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0")]
    public virtual decimal? OrderTotal { get; set; }
    #endregion
    #region DueDate
    public abstract class dueDate : PX.Data.IBqlField
    {
    }
    [PXDate]
    [PXUIField(DisplayName = "Due Date")]
    public virtual DateTime? DueDate { get; set; }
    #endregion
}
```

3. Erstellen Sie im PX.Documentation-Namespace Ihren SOOrderInq-BLC mit dem folgenden Code-Snippet, um die Delegierung der Ergebnisdatenansicht zu deklarieren, die später zum Auffüllen des Berichts mit Daten verwendet wird:

```
public class SOOrderInq : PXGraph<SOOrderInq>
{
    public PXCancel<SOOrderFilter> Cancel;
    public PXFilter<SOOrderFilter> Filter;

    [PXFilterable]
    public PXSelectOrderBy<SOOrderData,
```

```
        OrderBy<Desc<SOOrderData.orderNbr>>> Result;
    protected virtual IEnumerable result()
    {
        BqlCommand cmd = PXSelect<SOOrder,
            Where<SOOrder.orderDate,
                Between<Current<SOOrderFilter.dateFrom>,
                    Current<SOOrderFilter.dateTo>>>>.GetCommand();
        PXView inView = new PXView(this, true, cmd);
        int startRow = PXView.StartRow;
        int totalRows = 0;
        foreach (SOOrder order in inView.Select(PXView.Currents, PXView.Parameters,
            PXView.Searches, PXView.SortColumns, PXView.Descendings, PXView.Filters,
            ref startRow, PXView.MaximumRows, ref totalRows))
        {
            yield return new SOOrderData
            {
                OrderType = order.OrderType,
                OrderNbr = order.OrderNbr,
                OrderDate = order.OrderDate,
                Status = order.Status,
                OrderDesc = order.OrderDesc,
                OrderTotal = order.OrderTotal,
                DueDate = order.DueDate,
            };
        }
        PXView.StartRow = 0;
    }

    public SOOrderInq()
    {
        Result.Cache.AllowInsert = false;
        Result.Cache.AllowUpdate = false;
        Result.Cache.AllowDelete = false;
    }

    public PXAction<SOOrderFilter> Report;
    [PXButton]
    [PXUIField(DisplayName = "View As Report", MapEnableRights = PXCacheRights.Select,
MapViewRights = PXCacheRights.Select)]
    protected virtual void report()
    {
        PXReportResultset reportData = new PXReportResultset(typeof(SOOrderData));
        foreach (SOOrderData row in Result.Select())
        {
            reportData.Add(row);
        }
        throw new PXReportRequiredException(reportData, "SO610501",
PXBaseRedirectException.WindowMode.NewWindow, "Report");
    }
}
```

4. Erstellen Sie die Seite SO401090.aspx, indem Sie die FormDetail-Vorlage auswählen, und legen Sie die folgenden Eigenschaften für PXDataSource fest:

- PrimaryView: Filter
- Typname: PX.Documentation.SOOrderInq

Fügen Sie anschließend das Eingabesteuerelement im Filter-Header-Formular hinzu:

```
 <px:PXFormView ID="form" runat="server" DataSourceID="ds" Style="z-index: 100"
     Width="100%" DataMember="Filter">
     <Template>
         <px:PXLayoutRule runat="server" StartRow="True" Merge="True" LabelsWidth="XS"
ControlSize="S" />
         <px:PXDateTimeEdit ID="edDateFrom" runat="server" CommitChanges="True"
DataField="DateFrom" />
         <px:PXDateTimeEdit ID="edDateTo" runat="server" CommitChanges="True"
DataField="DateTo" />
         <px:PXLayoutRule runat="server" />
     </Template>
 </px:PXFormView>
```

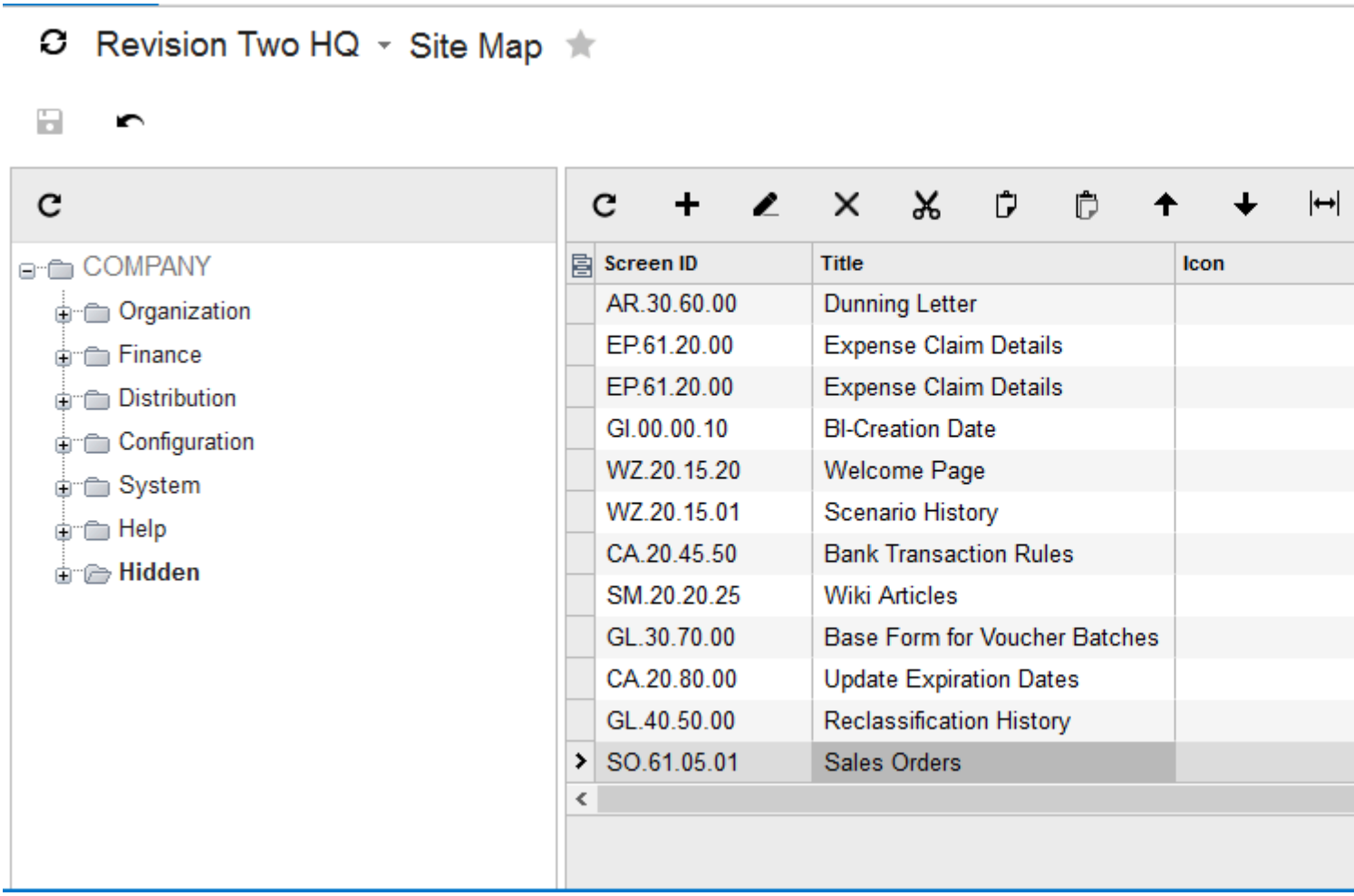
Erstellen Sie die folgenden Spalten für das Detailraster:

```
 <px:PXGrid ID="grid" runat="server" DataSourceID="ds" Style="z-index: 100"
     Width="100%" Height="150px" SkinID="Inquire" AllowPaging="True"
AdjustPageSize="Auto">
     <Levels>
         <px:PXGridLevel DataMember="Result">
             <Columns>
                 <px:PXGridColumn DataField="OrderType" />
                 <px:PXGridColumn DataField="OrderNbr" Width="90px" />
                 <px:PXGridColumn DataField="OrderDate" Width="90px" />
                 <px:PXGridColumn DataField="Status" />
                 <px:PXGridColumn DataField="OrderDesc" Width="200px" />
                 <px:PXGridColumn DataField="DueDate" Width="90px" />
             </Columns>
         </px:PXGridLevel>
     </Levels>
     <AutoSize Container="Window" Enabled="True" MinHeight="150" />
 </px:PXGrid>
```

5. Hinzufügen des erstellten Bildschirms zur Sitemap

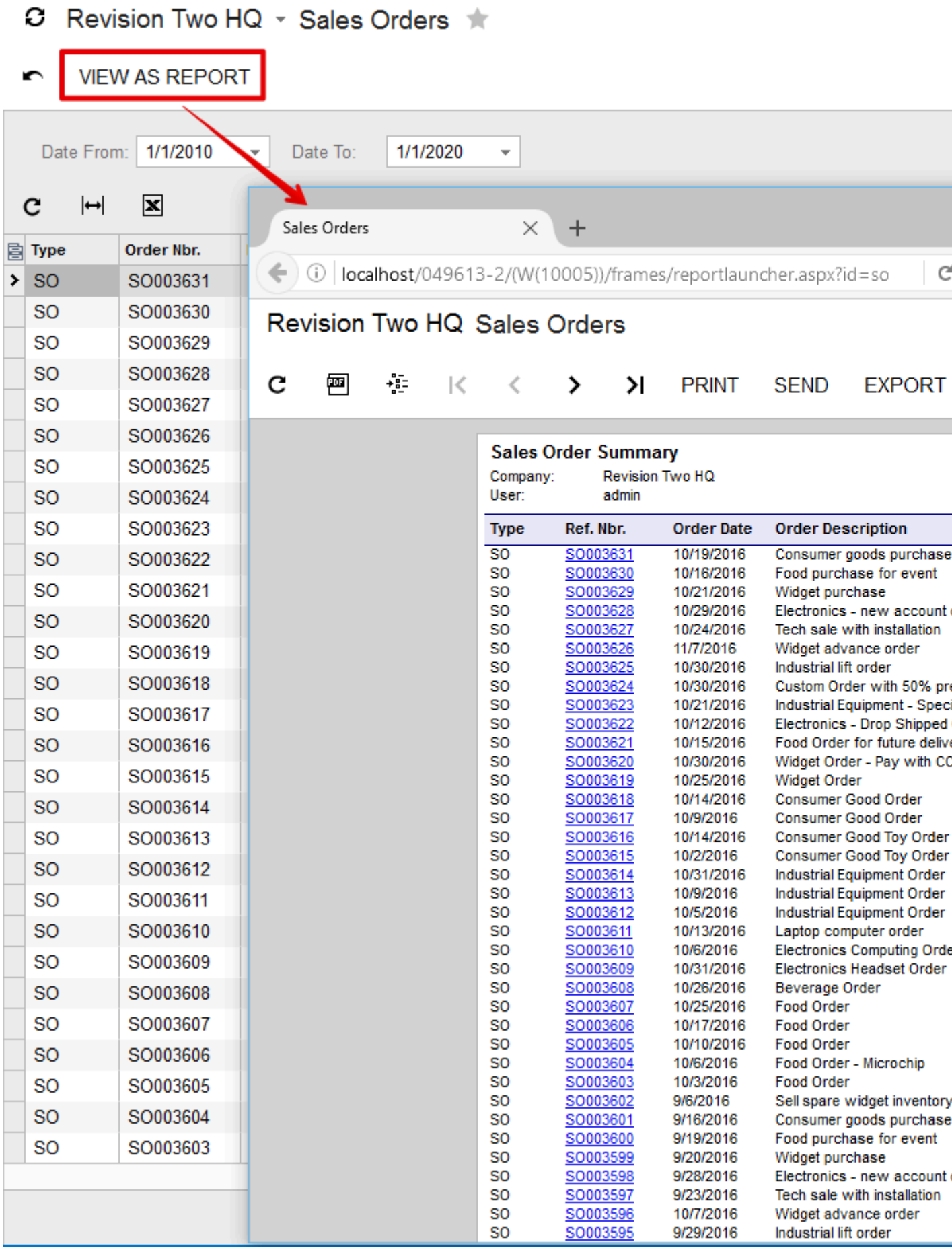
So füllen Sie den Bericht mit Daten, die von einem Datensichtedelegierten zurückgegeben werden:

1. Fügen Sie die SO610501.rpx- Berichtsdatei in den Ordner ReportsCustomized Ihrer Acumatica-Website ein, und fügen Sie dann den Bericht Verkaufsaufträge im Ordner Site Map **Hidden** hinzu



2. Deklarieren Sie die **Ansicht als** Berichtaktion in der SOOrderInq-BLC, um einen Kundenauftragsbericht zu generieren und anzuzeigen. Die PXReportRequiredException akzeptiert PXReportResultset, das innerhalb der Aktion vorbereitet wurde, um den Bericht mit Daten zu füllen, die vom **Ergebnisdatensichtdelegaten zurückgegeben** werden:

```
public class SOOrderInq : PXGraph<SOOrderInq>
{
    ...

    public PXAction<SOOrderFilter> Report;
    [PXButton]
    [PXUIField(DisplayName = "View as Report", MapEnableRights = PXCacheRights.Select,
MapViewRights = PXCacheRights.Select)]
    protected virtual void report()
    {
        PXReportResultset reportData = new PXReportResultset(typeof(SOOrderData));
        foreach (SOOrderData row in Result.Select())
        {
            reportData.Add(row);
        }
        throw new PXReportRequiredException(reportData, "SO610501",
PXBaseRedirectException.WindowMode.NewWindow, "Report");
    }
}
```



Bericht mit Daten durch Code füllen online lesen:
https://riptutorial.com/de/acumatica/topic/7700/bericht-mit-daten-durch-code-fullen

Kapitel 13: Bilder auf der Anmeldeseite ersetzen

Einführung

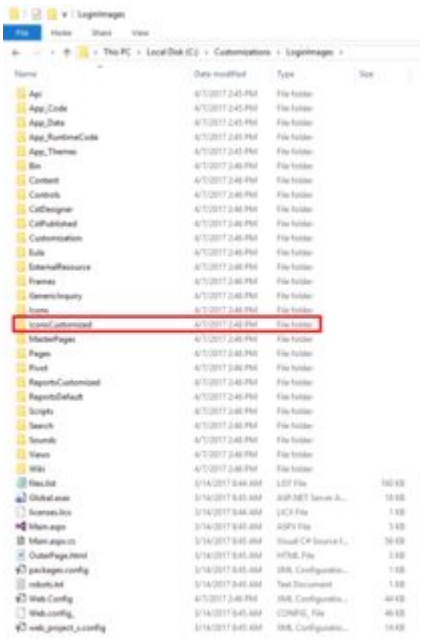
In diesem Thema erfahren Sie, wie Sie Standard-Acumatica-Bilder auf der Anmeldeseite ersetzen. Der aufgezeigte Ansatz stellt sicher, dass Ihre benutzerdefinierten Bilder nach dem Upgrade auf eine neuere Version auf der Anmeldeseite verbleiben und die von Acumatica bereitgestellten Originalbilder wiederherstellen, wenn Ihre Anpassung irgendwann unveröffentlicht erscheint.

Examples

Verwenden der Anpassung zum Ersetzen von Bildern auf der Anmeldeseite

Um ein Anpassungspaket zu erstellen, das Bilder auf der Anmeldeseite ersetzt, führen Sie die folgenden Schritte in Ihrer **lokalen Acumatica-Instanz aus** :

1. Erstellen Sie einen neuen Ordner im Acumatica-Instanzordner. Für dieses Beispiel habe ich in meiner lokalen **LoginImages-** Instanz einen Ordner namens ***IconsCustomized*** **hinzugefügt** :

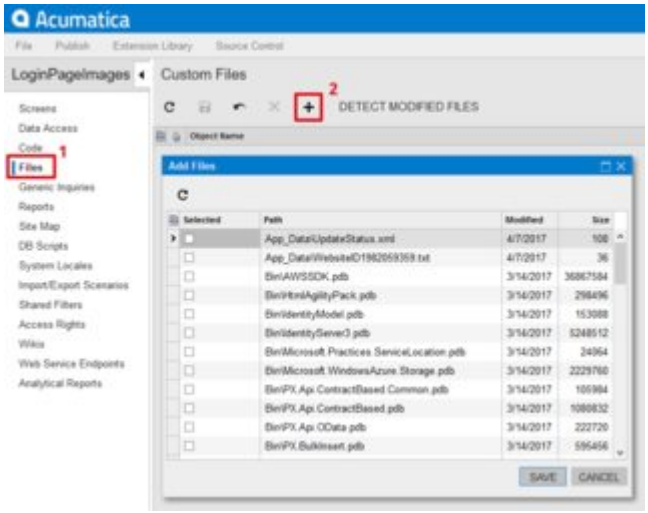


2. Fügen Sie in diesem Ordner Ihre benutzerdefinierten Bilder hinzu. In diesem Beispiel habe ich Bilder von der Acumatica 4.2-Anmeldeseite verwendet:

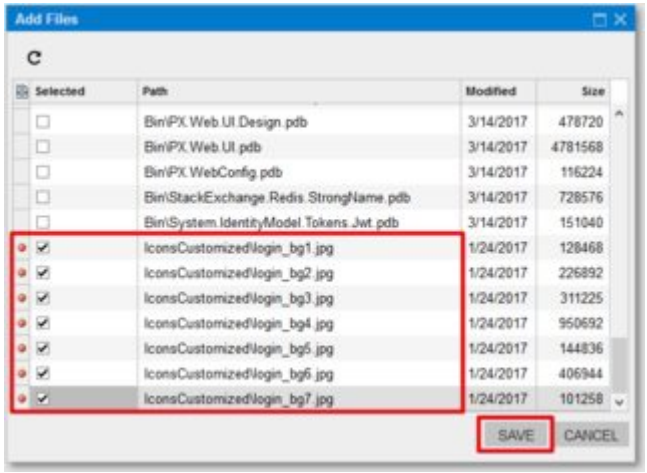
Um alle Bilder auf der Anmeldeseite zu ersetzen , müssen Sie mindestens so viele benutzerdefinierte Bilder in Ihrem **IconsCustomized-** Ordner **hinzufügen** wie die Anzahl der `login_bg*.*` -Dateien, die sich ursprünglich im **Icons-** Ordner Ihrer Acumatica-Website befinden. Es ist vollkommen in Ordnung, dasselbe Bild oder mehrere Bilder mehrmals zu verwenden (durch unterschiedliche Benennung der Dateien), wenn die Anzahl Ihrer benutzerdefinierten Bilder unter der von Acumatica ursprünglich angegebenen Anzahl liegt.

3. Melden Sie sich jetzt bei Ihrer Acumatica-Anwendung an, erstellen Sie ein neues Anpassungsprojekt mit dem Namen ***LoginPageImages*** und öffnen Sie es in Customization Manager.

4. Navigieren Sie im Anpassungs-Manager zum Abschnitt " **Dateien** " und klicken Sie auf die Schaltfläche **Neuen Datensatz** hinzufügen, um das Dialogfeld " **Dateien** hinzufügen" zu öffnen:

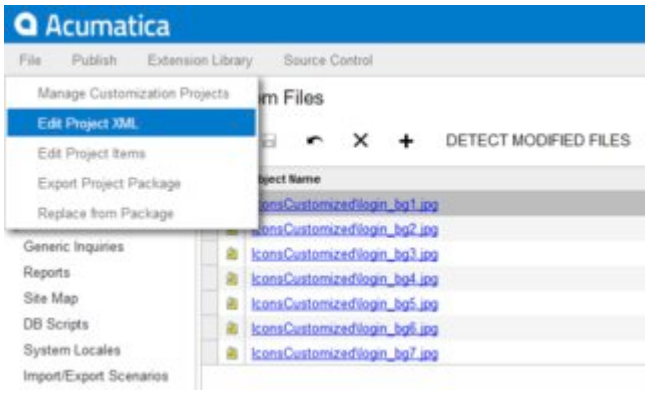


5. **Wählen** Sie im Dialogfeld "Dateien **hinzufügen** " alle Dateien aus Ihrem **IconsCustomized-** Ordner aus und klicken Sie auf " **Speichern"** .

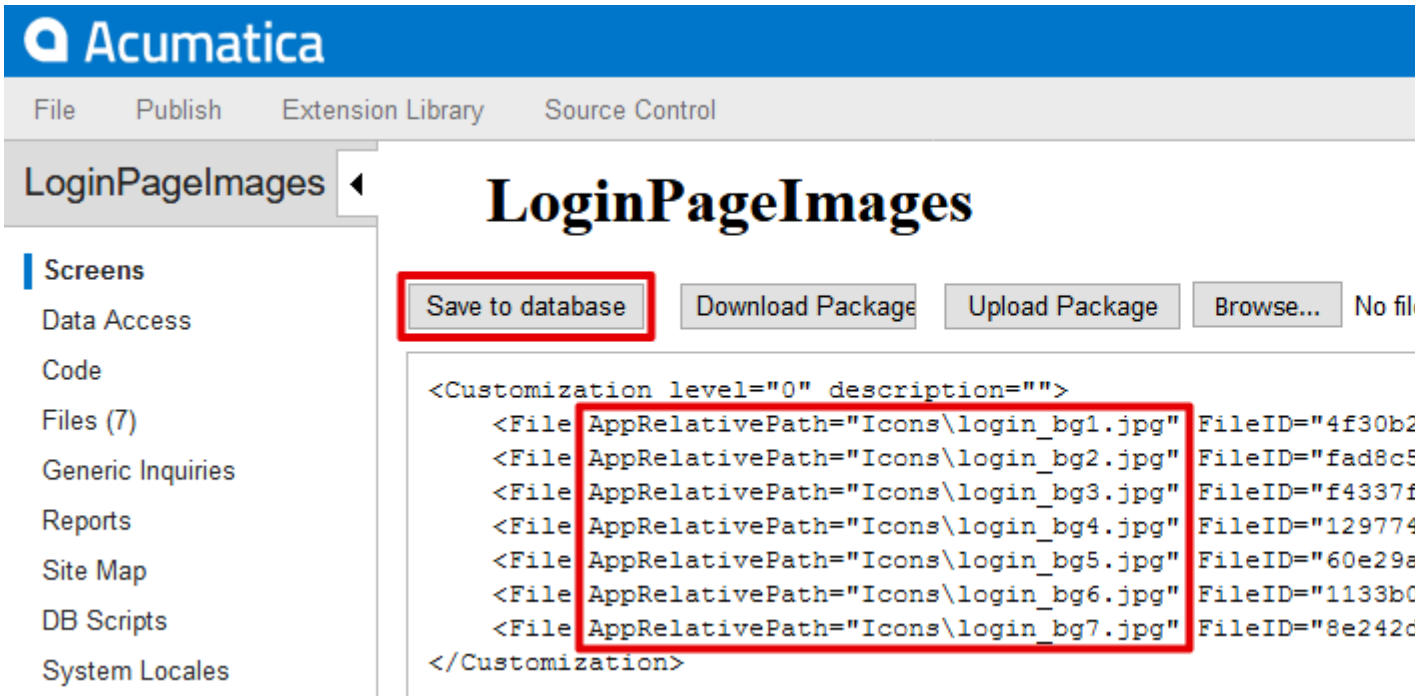


Jetzt haben Sie die benutzerdefinierten Anmeldeseitenbilder im Anpassungsprojekt. Sie müssen den Pfad jedoch noch bearbeiten, damit sie die Standardbilder korrekt ersetzen.

6. Wählen Sie im Anpassungs-Manager im Menü **Datei** die **Option Projekt-XML bearbeiten** :

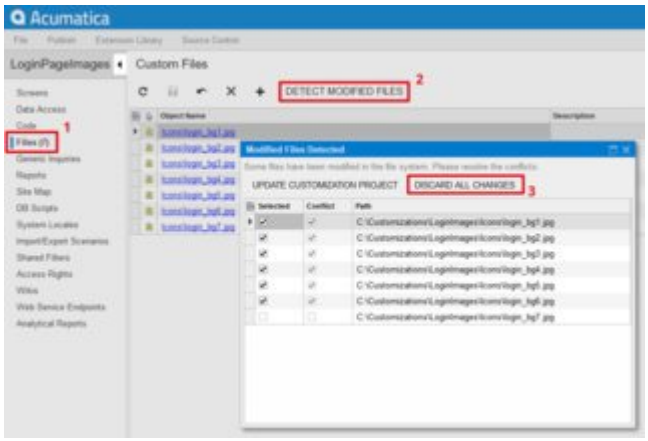


7. Für alle **Datei** - Tags, für die benutzerdefinierten Bilder erzeugt, laden Sie das **AppRelativePath** Attribut ***AppRelativePath = „Icons ...“*** und das **Systemfile-** Attribut auf ***True*** für die Bilder festgelegt, die derzeit in dem **Icons** Ordner, dann das **Speichern** klicken Sie **auf Datenbank** Taste, wenn fertig:



Bei der Veröffentlichung der Anpassung werden von Acumatica automatisch Dateien gesichert, die sich aktuell im Websiteordner befinden. Diese Dateien werden durch Dateien aus der Anpassung ersetzt, wobei **das** Attribut " **SystemFile"** auf " ***True" gesetzt ist*** .

8. Wenn Sie nun mit der Veröffentlichung der Anpassung fortfahren, ist es sehr wahrscheinlich, dass **Einige Dateien im Dateisystem geändert wurden.** Fehlermeldung, die angezeigt wird. Um dies zu verhindern ziemlich erschreckend Nachricht erscheinen, öffnen Sie in Customization Manager - Projekt, navigieren Sie zu dem Abschnitt **Dateien** und klicken Sie auf **Detect Geänderte Dateien** die **geänderten Dateien erkannt** Dialog zu öffnen, klicken Sie auf die **Discard All** Schaltfläche **Änderungen:**



9. Jetzt können Sie die Anpassung veröffentlichen, um Ihre benutzerdefinierten Bilder auf der Anmeldeseite anzuzeigen:

Bilder auf der Anmeldeseite ersetzen online lesen:
https://riptutorial.com/de/acumatica/topic/9657/bilder-auf-der-anmeldeseite-ersetzen

Kapitel 14: Datensätze über REST-vertragsbasierte API exportieren

Einführung

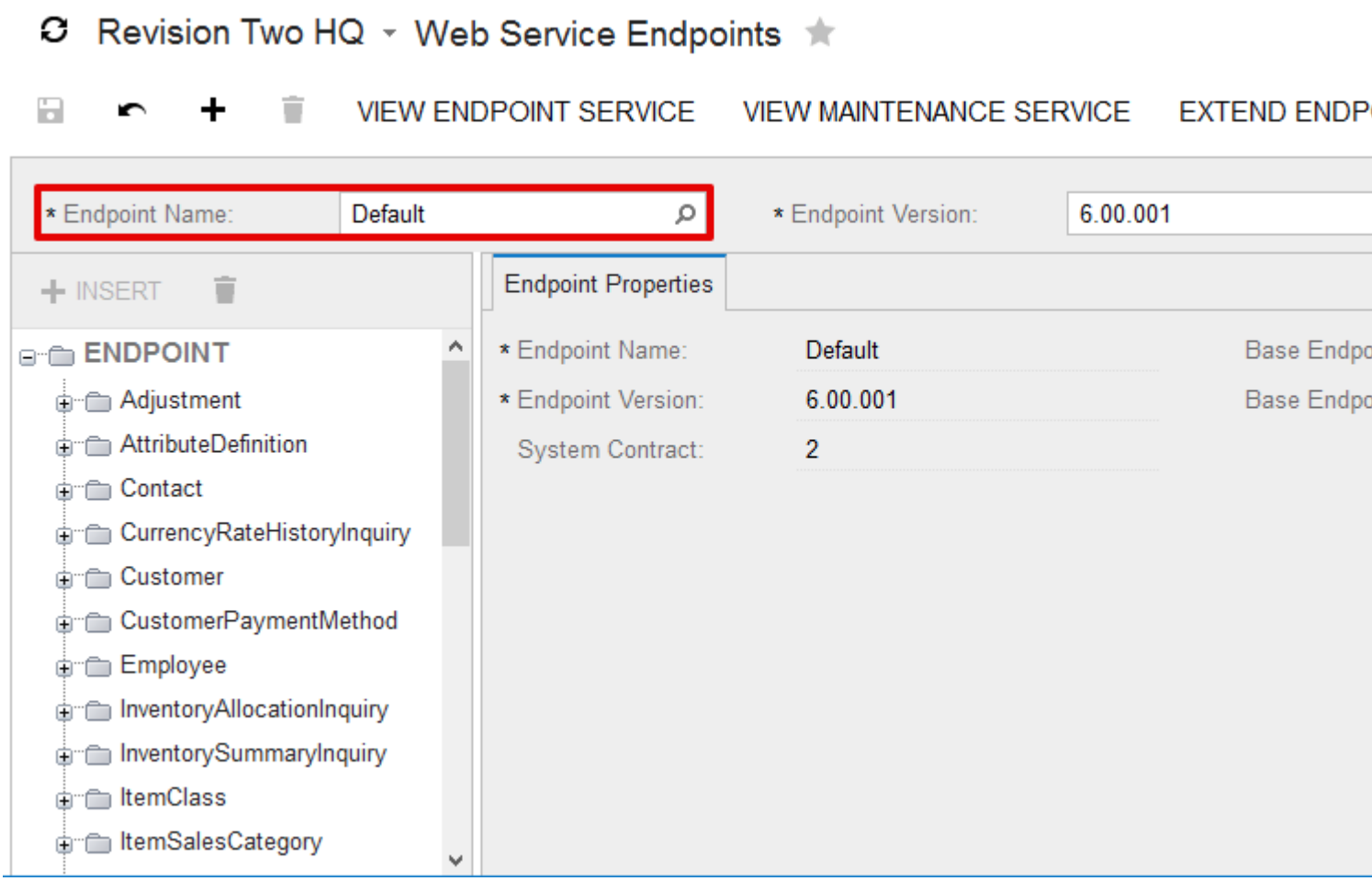
In diesem Thema wird veranschaulicht, wie Datensätze aus Acumatica ERP über die vertragsbasierte REST-API exportiert werden. Im Gegensatz zu der screen-based API von Acumatica ERP bietet die vertragsbasierte API sowohl SOAP- als auch REST-Schnittstellen. Weitere Informationen zur vertragsbasierten API finden Sie in der Acumatica ERP-Dokumentation

Bemerkungen

Um mit der vertragsbasierten REST-API von Acumatica ERP zu kommunizieren, muss Ihre Clientanwendung immer die folgenden 3 Schritte ausführen:

1. Melden Sie sich bei der Acumatica ERP-Instanz an und erhalten Sie ein Cookie mit Informationen zur Benutzersitzung
2. Interaktion mit einem der vertragsbasierten API-Endpunkte, die in der Acumatica ERP-Instanz verfügbar sind
3. Melden Sie sich von Acumatica ERP ab, um die Benutzersitzung zu schließen

Alle in diesem Thema bereitgestellten Beispiele wurden mit dem ***Standardendpunkt erstellt*** und immer im Rahmen des Standardinstallationsprozesses von Acumatica ERP bereitgestellt. Auf dem Bildschirm **Web-Service-Endpunkte** (SM.20.70.60) können Sie die Details vorhandener Endpunkte anzeigen oder Ihre benutzerdefinierten Endpunkte der vertraglichen Web-Services von Acumatica ERP konfigurieren:



Nachfolgend finden Sie die Implementierung der **RestService-** Klasse, die in allen obigen Beispielen für die Interaktion mit dem vertragsbasierten Webservice von Acumatica ERP verwendet wird:

```
public class RestService : IDisposable
{
    private readonly HttpClient _httpClient;
    private readonly string _acumaticaBaseUrl;
    private readonly string _acumaticaEndpointUrl;

    public RestService(string acumaticaBaseUrl, string endpoint,
        string userName, string password,
        string company, string branch)
    {
        _acumaticaBaseUrl = acumaticaBaseUrl;
        _acumaticaEndpointUrl = _acumaticaBaseUrl + "/entity/" + endpoint + "/";
        _httpClient = new HttpClient(
            new HttpClientHandler
            {
                UseCookies = true,
                CookieContainer = new CookieContainer()
            })
        {
            BaseAddress = new Uri(_acumaticaEndpointUrl),
            DefaultRequestHeaders =
            {
                Accept = {MediaTypeWithQualityHeaderValue.Parse("text/json")}
            }
        };
```

```
        var str = new StringContent(
            new JavaScriptSerializer()
                .Serialize(
                    new
                    {
                        name = userName,
                        password = password,
                        company = company,
                        branch = branch
                    }),
                    Encoding.UTF8, "application/json");

        _httpClient.PostAsync(acumaticaBaseUrl + "/entity/auth/login", str)
            .Result.EnsureSuccessStatusCode();
    }

    void IDisposable.Dispose()
    {
        _httpClient.PostAsync(_acumaticaBaseUrl + "/entity/auth/logout",
            new ByteArrayContent(new byte[0])).Wait();
        _httpClient.Dispose();
    }

    public string GetList(string entityName)
    {
        var res = _httpClient.GetAsync(_acumaticaEndpointUrl + entityName)
            .Result.EnsureSuccessStatusCode();

        return res.Content.ReadAsStringAsync().Result;
    }

    public string GetList(string entityName, string parameters)
    {
        var res = _httpClient.GetAsync(_acumaticaEndpointUrl + entityName + "?" + parameters)
            .Result.EnsureSuccessStatusCode();

        return res.Content.ReadAsStringAsync().Result;
    }
}
```

Examples

Datenexport in einem einzigen REST-Aufruf

In diesem Beispiel erfahren Sie, wie Sie die folgenden Daten aus Acumatica ERP in einem einzigen Aufruf über die vertragsbasierte REST-API exportieren:

- alle in der Anwendung vorhandenen Lagerartikel
- Alle Kundenaufträge des Typs IN

Wenn Sie Datensätze aus Acumatica ERP exportieren müssen, verwenden Sie die folgende URL:
`http://<Acumatica ERP instance URL>/entity/<Endpoint name>/<Endpoint version>/<Top-level entity>`

`<Top-level entity>` ist der Name der Entität, die Sie exportieren möchten

So exportieren Sie alle Lagerartikel in einem einzigen REST-Aufruf:

Um Bestandsartikeldatensätze aus einer lokalen `AcumaticaERP` Instanz mithilfe des ***Standardendpunkts*** der Version ***6.00.001 zu exportieren*** , sollten Sie die folgende URL verwenden: `http://localhost/AcumaticaERP/entity/Default/6.00.001/StockItem`

Nachfolgend finden Sie den in C # geschriebenen Beispielcode zum Exportieren aller Lagerbestände durch Senden eines einzelnen REST-Aufrufs an den ***Standardendpunkt*** der Version ***6.00.001*** :

```
using (RestService rs = new RestService(
    @"http://localhost/AcumaticaERP/", "Default/6.00.001",
    username, password, company, branch))
{
    string stockItems = rs.GetList("StockItem");
}
```

So exportieren Sie alle Kundenaufträge des Typs IN in einem einzigen REST-Aufruf:

Um Verkaufsaufträge des Typs ***IN*** aus einer lokalen `AcumaticaERP` Instanz mithilfe des ***Standardendpunkts*** der Version ***6.00.001 zu exportieren*** , sollten Sie die folgende URL verwenden: `http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$filter=OrderType eq 'IN'`

Nachfolgend finden Sie den in C # geschriebenen Beispielcode zum Exportieren aller Verkaufsaufträge des Typs ***IN,*** indem ein einzelner REST-Aufruf an den ***Standardendpunkt*** der Version ***6.00.001 gesendet wird*** :

```
using (RestService rs = new RestService(
    @"http://localhost/StackOverflow/", "Default/6.00.001",
    username, password, company, branch))
{
    var parameters = "$filter=OrderType eq 'IN'";
    string inSalesOrders = rs.GetList("SalesOrder", parameters);
}
```

Paginierung bei mehreren REST-Anforderungen implementieren

In diesem Beispiel erfahren Sie, wie Sie folgende Daten stapelweise aus Acumatica ERP über die vertragsbasierte REST-API exportieren:

- Bestandsartikel, die in der Anwendung in Chargen von 10 Datensätzen vorhanden sind
- alle Kundenaufträge in Chargen von 100 Datensätzen

So exportieren Sie Lagerartikel in Chargen von 10 Datensätzen mit mehreren REST-Aufrufen:

Um die ersten 10 vorrätigen Artikel aus einer lokalen `AcumaticaERP` Instanz mithilfe des ***Standardendpunkts*** der Version ***6.00.001 zu exportieren*** , sollten Sie die folgende URL verwenden: `http://localhost/AcumaticaERP/entity/Default/6.00.001/StockItem?$top=10`

`http://localhost/AcumaticaERP/entity/Default/6.00.001/StockItem?$top=10&$filter=InventoryID gt '<InventoryID>'` Bestandsartikel von 10 auf 20 anzufordern, erweitern Sie einfach die obige URL um den ***Filterparameter*** :
`http://localhost/AcumaticaERP/entity/Default/6.00.001/StockItem?$top=10&$filter=InventoryID gt '<InventoryID>'`

`<InventoryID>` ist die ID der letzten Lagerposition, die mit einem vorherigen REST-Aufruf exportiert wurde

Nachfolgend finden Sie den Beispielcode, der in C # geschrieben ist, um alle ***Lagerartikel in Stapeln*** von 10 Datensätzen zu exportieren, indem Sie mehrere REST-Aufrufe an den ***Standardendpunkt*** von Version ***6.00.001 senden*** :

```
using (RestService rs = new RestService(
    @"http://localhost/StackOverflow/", "Default/6.00.001",
    username, password, company, branch))
{
    var json = new JavaScriptSerializer();
    string parameters = "$top=10";
    string items = rs.GetList("StockItem", parameters);
    var records = json.Deserialize<List<Dictionary<string, object>>>(items);

    while (records.Count == 10)
    {
        var inventoryID = records[records.Count - 1]["InventoryID"] as Dictionary<string,
object>;
        var inventoryIDValue = inventoryID.Values.First();
        string nextParameters = parameters + "&" +
            "$filter=" + string.Format("InventoryID gt '{0}'", inventoryIDValue);
        items = rs.GetList("StockItem", nextParameters);
        records = json.Deserialize<List<Dictionary<string, object>>>(items);
    }
}
```

So exportieren Sie alle Kundenaufträge in Chargen von 100 Datensätzen mit mehreren REST-Aufrufen:

Um die ersten 100 Verkaufsaufträge von einer lokalen `AcumaticaERP` Instanz mithilfe des ***Standardendpunkts*** der Version ***6.00.001 zu exportieren*** , sollten Sie die folgende URL verwenden: `http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$top=100`

Da der Primärschlüssel der **Verkaufsauftragseinheit** aus ***Auftragstyp*** und ***Auftragsnummer besteht*** , verwenden Sie in diesem Beispiel eine Kombination von **Filterparametern** für die Felder ***Auftragstyp*** und ***Auftragsnummer*** :

- `http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$top=100&$filter=OrderType eq 'SO' and OrderNbr gt '<OrderNbr>'` Verkaufsaufträge von 100 bis 200 des ***SO-*** Typs anzufordern, sollten Sie die folgende URL verwenden:
  `http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$top=100&$filter=OrderType eq 'SO' and OrderNbr gt '<OrderNbr>'`

`<OrderNbr>` ist die Nummer des letzten `<OrderNbr>` der mit einem vorherigen REST-Aufruf exportiert wurde

- `http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$top=100&$filter=OrderType gt 'SO' and OrderNbr gt ''` ersten 100 Verkaufsaufträge für den nächsten ***SO-*** Typ anzufordern, sollten Sie die folgende URL verwenden:
  `http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$top=100&$filter=OrderType gt 'SO' and OrderNbr gt ''`

Nachfolgend finden Sie den in C # geschriebenen Beispielcode, um alle Verkaufsaufträge in Stapeln von 100 Datensätzen mit mehreren REST-Aufrufen an den ***Standardendpunkt*** der Version ***6.00.001 zu exportieren*** :

```
using (RestService rs = new RestService(
    @"http://localhost/StackOverflow/", "Default/6.00.001",
    username, password, company, branch))
{
    var json = new JavaScriptSerializer();
    string parameters = "$top=100";
    string items = rs.GetList("SalesOrder", parameters);
    var records = json.Deserialize<List<Dictionary<string, object>>>(items);

    bool sameOrderType = true;
    while (records.Count > 0 && (records.Count == 100 || !sameOrderType))
    {
        var orderType = records[records.Count - 1]["OrderType"] as Dictionary<string, object>;
        var orderTypeValue = orderType.Values.First();
        var orderNbr = records[records.Count - 1]["OrderNbr"] as Dictionary<string, object>;
        var orderNbrValue = orderNbr.Values.First();

        string nextParameters = parameters + "&" + "$filter=" +
            string.Format("OrderType {0} '{1}'", sameOrderType ? "eq" : "gt", orderTypeValue)
+ " and " +
            string.Format("OrderNbr gt '{0}'", sameOrderType ? orderNbrValue : "''" );
        items = rs.GetList("SalesOrder", nextParameters);
        records = json.Deserialize<List<Dictionary<string, object>>>(items);
        sameOrderType = records.Count == 100;
    }
}
```

Datensätze über REST-vertragsbasierte API exportieren online lesen:
https://riptutorial.com/de/acumatica/topic/9298/datensatze-uber-rest-vertragsbasierte-api-

exportieren

Kapitel 15: Datensätze über screen-based API exportieren

Einführung

In diesem Thema wird veranschaulicht, wie Datensätze aus Acumatica ERP über die screen-based API exportiert werden. Die screen-based API von Acumatica ERP bietet nur die SOAP-Schnittstelle. Wenn Ihre Entwicklungsplattform nur eine begrenzte Unterstützung für SOAP-Webdienste bietet, sollten Sie die vertragsbasierte API in Betracht ziehen, die sowohl SOAP- als auch REST-Schnittstellen bietet. Weitere Informationen zur screen -based API finden Sie in der Acumatica ERP-Dokumentation

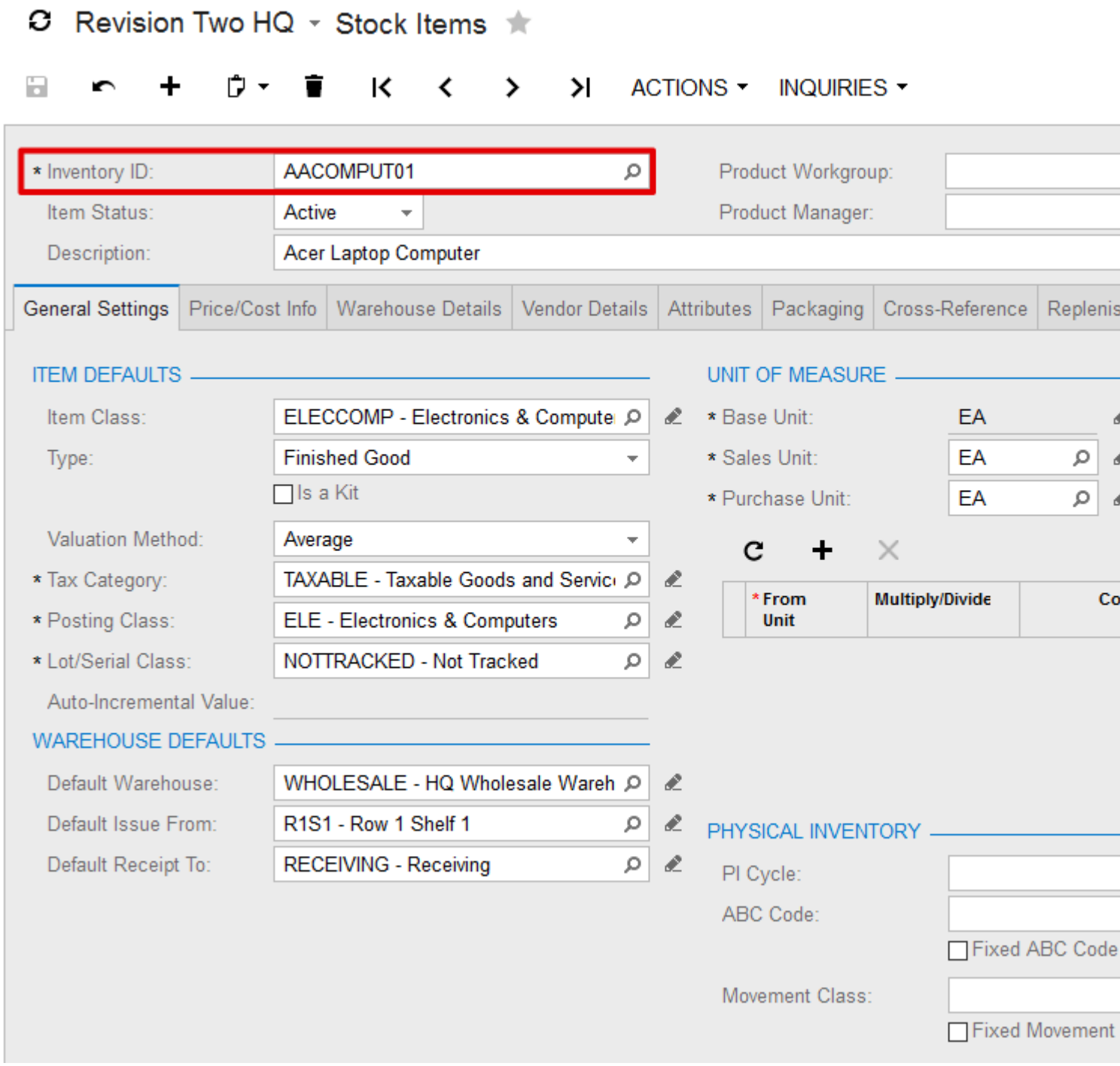
Bemerkungen

Alle in diesem Thema bereitgestellten Beispiele wurden mit dem screen-based API Wrapper erstellt. Wenn Sie möchten, dass Ihre Clientanwendung nicht von den Änderungen der Benutzeroberfläche in der Acumatica ERP-Anwendung abhängt, sollten Sie den bildschirmbasierten API-Wrapper verwenden, der in der Acumatica ERP-Dokumentation beschrieben wird

Examples

Datenexport aus einem Erfassungsformular mit einem einzigen Primärschlüssel

Der Bildschirm **Stock Items** (IN.20.25.00) ist eines der am häufigsten verwendeten Datenerfassungsformulare von Acumatica ERP, um Daten zu exportieren. ***Die Bestands-ID*** ist der einzige Primärschlüssel auf dem Bildschirm mit den **Bestandsposten** :



Um Datensätze aus einem Dateneingabeformular zu exportieren, muss Ihre SOAP-Anforderung immer mit dem Befehl `ServiceCommands.Every[Key]` , wobei `[Key]` durch den Namen des Primärschlüssels ersetzt werden soll.

So exportieren Sie alle Lagerartikel in einem einzigen Web-Service-Aufruf:

```
Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/IN202500.asmx";
```

```
context.Login(username, password);
try
{
    Content stockItemsSchema = PX.Soap.Helper.GetSchema<Content>(context);
    Field lastModifiedField = new Field
    {
        ObjectName = stockItemsSchema.StockItemSummary.InventoryID.ObjectName,
        FieldName = "LastModifiedDateTime"
    };
    var commands = new Command[]
    {
        stockItemsSchema.StockItemSummary.ServiceCommands.EveryInventoryID,
        stockItemsSchema.StockItemSummary.InventoryID,
        stockItemsSchema.StockItemSummary.Description,
        stockItemsSchema.GeneralSettingsItemDefaults.ItemClass,
        stockItemsSchema.GeneralSettingsUnitOfMeasureBaseUnit.BaseUnit,
        lastModifiedField
    };
    var items = context.Export(commands, null, 0, false, false);
}
finally
{
    context.Logout();
}
```

Mit der Zeit wächst die Datenmenge in jeder ERP-Anwendung. Wenn Sie alle Datensätze aus Ihrer Acumatica ERP-Instanz in einem einzigen Web-Service-Aufruf exportieren, können Sie sehr bald Timeout-Fehler feststellen. Das Erhöhen des Timeouts ist eine mögliche einmalige, aber nicht sehr gute langfristige Lösung. Die beste Option zur Bewältigung dieser Herausforderung besteht darin, Lagerartikel in mehreren Datensätzen zu exportieren.

So exportieren Sie Lagerbestände in Chargen von 10 Datensätzen:

```
Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/IN202500.asmx";
context.Login(username, password);
try
{
    Content stockItemsSchema = PX.Soap.Helper.GetSchema<Content>(context);
    Field lastModifiedField = new Field
    {
        ObjectName = stockItemsSchema.StockItemSummary.InventoryID.ObjectName,
        FieldName = "LastModifiedDateTime"
    };
    var commands = new Command[]
    {
        stockItemsSchema.StockItemSummary.ServiceCommands.EveryInventoryID,
        stockItemsSchema.StockItemSummary.InventoryID,
        stockItemsSchema.StockItemSummary.Description,
        stockItemsSchema.GeneralSettingsItemDefaults.ItemClass,
        stockItemsSchema.GeneralSettingsUnitOfMeasureBaseUnit.BaseUnit,
        lastModifiedField
    };
```

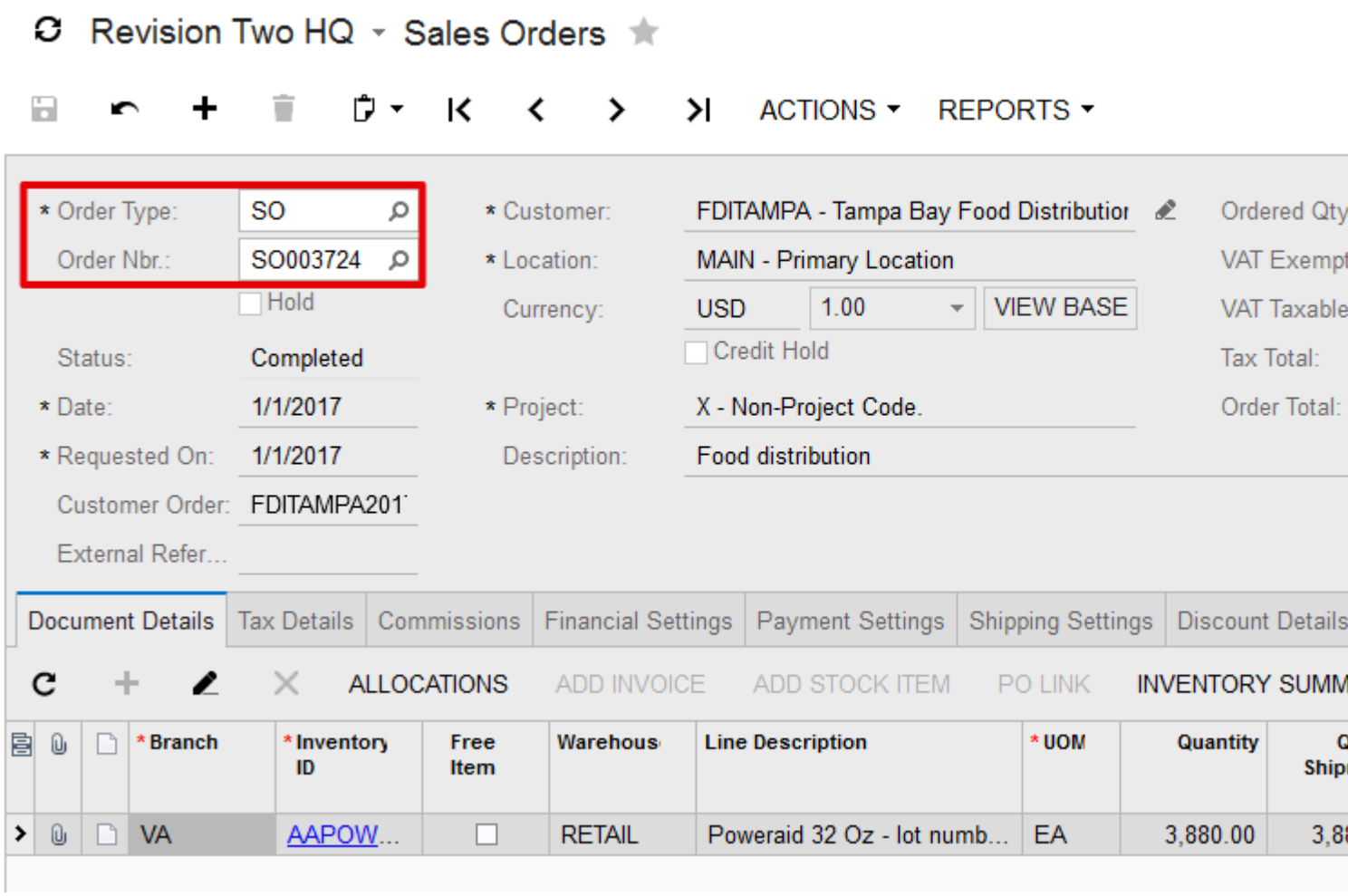
```
    var items = context.Export(commands, null, 10, false, false);

    while (items.Length == 10)
    {
        var filters = new Filter[]
        {
            new Filter
            {
                Field = stockItemsSchema.StockItemSummary.InventoryID,
                Condition = FilterCondition.Greater,
                Value = items[items.Length - 1][0]
            }
        };
        items = context.Export(commands, filters, 10, false, false);
    }
}
finally
{
    context.Logout();
}
```

Es gibt zwei Hauptunterschiede zwischen dem Einzelaufruf-Ansatz und dem Export in Batches:

- **Der** Parameter **topCount** des **Exportbefehls** wurde bei der **Methode** des **Einzelaufrufs** immer auf `0` gesetzt
- Beim Exportieren von Datensätzen in Batches wird die Größe eines **Stapels mit dem** Parameter **topCount** konfiguriert, der durch das **Filter-** Array ergänzt wird, um die nächste Ergebnismenge anzufordern

Datenexport aus einem Erfassungsformular mit einem zusammengesetzten Primärschlüssel

Die **Kundenaufträge** Bildschirm (SO.30.10.00) ist ein perfektes Beispiel für ein Dateneingabeformular mit einem zusammengesetzten Primärschlüssel. Der Primärschlüssel im Bildschirm **Kundenaufträge** setzt sich aus der ***Auftragsart*** und der ***Bestellnummer zusammen*** :



Die empfohlene 2-Schritt-Strategie zum Exportieren von Daten aus dem Bildschirm " **Kundenaufträge"** oder einem anderen Dateneingabeformular mit einem zusammengesetzten Primärschlüssel über die bildschirmbasierte API:

- In Schritt 1 fordern Sie alle Arten von Bestellungen an, die zuvor in Ihrer Acumatica ERP-Anwendung erstellt wurden
- Der zweite Schritt besteht darin, Aufträge jeder Art unabhängig voneinander entweder in einem einzigen Aufruf oder in Batches zu exportieren

Um alle Arten von bestehenden Bestellungen anzufordern:

```
Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/SO301000.asmx";
context.Login(username, password);
try
{
    Content orderSchema = PX.Soap.Helper.GetSchema<Content>(context);
    var commands = new Command[]
    {
        orderSchema.OrderSummary.ServiceCommands.EveryOrderType,
```

```
        orderSchema.OrderSummary.OrderType,
    };

    var types = context.Export(commands, null, 1, false, false);
}
finally
{
    context.Logout();
}
```

Beachten Sie beim SOAP-Aufruf oben, **dass der** Parameter **topCount** des **Exportbefehls** auf `1` . Diese Anforderung dient nur dazu, alle zuvor in Ihrer Acumatica ERP-Anwendung erstellten Auftragsarten abzurufen, nicht um Daten zu exportieren.

So exportieren Sie Datensätze jedes Typs unabhängig in Batches:

```
Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/SO301000.asmx";
context.Login(username, password);
try
{
    Content orderSchema = PX.Soap.Helper.GetSchema<Content>(context);
    var commands = new Command[]
    {
        orderSchema.OrderSummary.ServiceCommands.EveryOrderType,
        orderSchema.OrderSummary.OrderType,
    };
    var types = context.Export(commands, null, 1, false, false);

    for (int i = 0; i < types.Length; i++)
    {
        commands = new Command[]
        {
            new Value
            {
                LinkedCommand = orderSchema.OrderSummary.OrderType,
                Value = types[i][0]
            },
            orderSchema.OrderSummary.ServiceCommands.EveryOrderNbr,
            orderSchema.OrderSummary.OrderType,
            orderSchema.OrderSummary.OrderNbr,
            orderSchema.OrderSummary.Customer,
            orderSchema.OrderSummary.CustomerOrder,
            orderSchema.OrderSummary.Date,
            orderSchema.OrderSummary.OrderedQty,
            orderSchema.OrderSummary.OrderTotal
        };
        var orders = context.Export(commands, null, 100, false, false);
        while (orders.Length == 100)
        {
            var filters = new Filter[]
            {
                new Filter
```

```
                {
                    Field = orderSchema.OrderSummary.OrderNbr,
                    Condition = FilterCondition.Greater,
                    Value = orders[orders.Length - 1][1]
                }
            };
            orders = context.Export(commands, filters, 100, false, false);
        }
    }
}
finally
{
    context.Logout();
}
```

Das obige Beispiel veranschaulicht, wie alle Kundenaufträge aus Acumatica ERP in Chargen von 100 Datensätzen exportiert werden. Um den Kundenauftrag jeder Art unabhängig voneinander zu exportieren, muss Ihre SOAP-Anforderung immer mit dem Befehl `Value` beginnen, der den Typ der zu exportierenden Aufträge bestimmt. Nach dem Befehl Value, der zum Festlegen des ersten Schlüsselwerts verwendet wird, wird der Befehl `ServiceCommands.Every[Key]` aufgerufen, wobei `[Key]` durch den Namen des zweiten Schlüssels ersetzt werden soll.

So exportieren Sie Datensätze eines bestimmten Typs:

Wenn Sie Verkaufsaufträge eines bestimmten Typs exportieren müssen, können Sie die Art der Aufträge mit dem Befehl `Value` am Anfang Ihrer SOAP-Anforderung explizit definieren, gefolgt von der Einzelaufrufmethode oder dem Export in Batches.

So exportieren Sie alle Kundenaufträge des Typs ***IN*** in einem Anruf:

```
Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/SO301000.asmx";
context.Login(username, password);
try
{
    Content orderSchema = PX.Soap.Helper.GetSchema<Content>(context);
    var commands = new Command[]
    {
        new Value
        {
            LinkedCommand = orderSchema.OrderSummary.OrderType,
            Value = "IN"
        },
        orderSchema.OrderSummary.ServiceCommands.EveryOrderNbr,
        orderSchema.OrderSummary.OrderType,
        orderSchema.OrderSummary.OrderNbr,
        orderSchema.OrderSummary.Customer,
        orderSchema.OrderSummary.CustomerOrder,
        orderSchema.OrderSummary.Date,
        orderSchema.OrderSummary.OrderedQty,
        orderSchema.OrderSummary.OrderTotal
```

```
    };
    var orders = context.Export(commands, null, 0, false, false);
}
finally
{
    context.Logout();
}
```

Datensätze über screen-based API exportieren online lesen:
https://riptutorial.com/de/acumatica/topic/9288/datensatze-uber-screen-based-api-exportieren

Kapitel 16: Datums- und Zeitfelder in Acumatica erstellen

Einführung

In diesem Thema werden die verschiedenen in Acumatica Framework verfügbaren Optionen zum Erstellen von Datums- und Uhrzeitfeldern in einer Datenzugriffsklasse (DAC) erläutert.

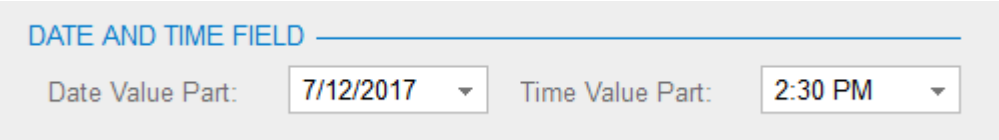
Examples

Das PX (DB) DateAndTime-Attribut

Das **PXDBDateAndTime-** Attribut und das **PXDateAndTime-** Attribut arbeiten mit einem DAC-Feld des `Nullable<DateTime>` (`DateTime?`) Zusammen und speichern sowohl Datums- als auch Zeitwertteile in einem einzigen Feld:

```
#region UsrDateAndTime
public abstract class usrDateAndTimeAttribute : IBqlField
{ }

[PXDBDateAndTime(
    DisplayNameDate = "Date Value Part",
    DisplayNameTime = "Time Value Part")]
public DateTime? UsrDateAndTime { get; set; }
#endregion
```

Aus der Sicht der Benutzeroberfläche wird erwartet, dass für ein Feld, das mit **PXDBDateAndTimeAttribute** oder **PXDateAndTimeAttribute** dekoriert ist, entweder separate Eingabesteuerelemente für Datums- und Zeitwertteile erstellt werden:




```
<px:PXDateTimeEdit runat="server" ID="edUsrDate" DataField="UsrDateAndTime_Date" />
<px:PXDateTimeEdit runat="server" ID="edUsrTime" DataField="UsrDateAndTime_Time"
TimeMode="True" />
```

oder separate Rasterspalten, um Datums- und Uhrzeitwerte einzugeben und anzuzeigen:

Date Value Part	Time Value Part
7/14/2017	9:30 AM

```
<Columns>
    ...
    <px:PXGridColumn DataField="UsrDateAndTime_Date" Width="90px" />
```

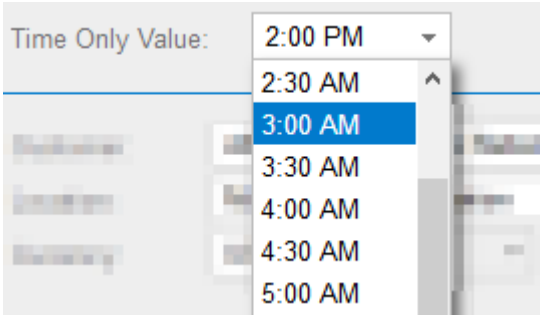
```
    <px:PXGridColumn DataField="UsrDateAndTime_Time" Width="90px" TimeMode="True" />
    ...
</Columns>
```

Das PXDBTime-Attribut

Das **PXDBTime-** Attribut kann mit einem DAC-Feld des `Nullable<DateTime>` (`DateTime?`) `DateTime?` und speichert nur den Zeitabschnitt ohne Datum in einem DAC-Feld:

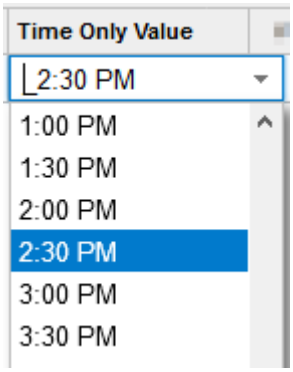
```
#region UsrTime
public abstract class usrTime : IBqlField
{ }

[PXDBTime(DisplayMask = "t", InputMask = "t")]
[PXUIField(DisplayName = "Time Only Value")]
public DateTime? UsrTime { get; set; }
#endregion
```

In der Benutzeroberfläche **erstellt** das System für ein mit **PXDBTimeAttribute** dekoriertes Feld ein Eingabesteuerelement, das nur Zeitwerte in einem Formular akzeptiert:




```
<px:PXDateTimeEdit runat="server" ID="edUsrTime" DataField="UsrTime" TimeMode="True" />
```

und innerhalb einer Gitterzelle:



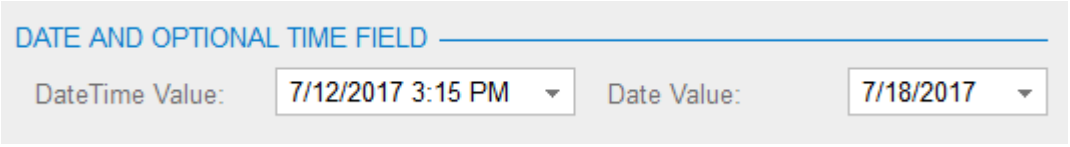

```
<Columns>
    ...
    <px:PXGridColumn DataField="UsrTime" Width="120px" TimeMode="True" />
    ...
</Columns>
```

Das PX (DB) DateAttribute-Attribut

Das **PXDBDate-** Attribut und das **PXDate-** Attribut arbeiten mit einem DAC-Feld vom Typ `Nullable<DateTime>` (`DateTime?`) Zusammen und speichern den Datumswert mit einem optionalen `DateTime?` in einem einzelnen Feld. Wenn **PX (DB) DateAttribute** zusätzlich zum Datum Zeit in einem DAC-Feld sparen soll, wird dies durch die **PreserveTime-** Eigenschaft definiert: Wenn **PreserveTime** auf ***True gesetzt ist*** , wird der **Zeitteil eines Feldwerts beibehalten** DAC-Feld:

```
#region UsrDateTime
public abstract class usrDateTime : IBqlField
{ }

[PXDBDate(PreserveTime = true, InputMask = "g")]
[PXUIField(DisplayName = "DateTime Value")]
public DateTime? UsrDateTime { get; set; }
#endregion

#region UsrDate
public abstract class usrDate : IBqlField
{ }

[PXDBDate]
[PXUIField(DisplayName = "Date Value")]
public DateTime? UsrDate { get; set; }
#endregion
```

In der Benutzeroberfläche **erstellt** das System für ein mit **PXDBDateAttribute** oder **PXDateAttribute** dekoriertes Feld ein Eingabesteuerelement, das abhängig vom Wert der **PreserveTime-** Eigenschaft entweder nur Datumswerte oder sowohl Datums- als auch **Uhrzeitwerte** akzeptiert. Dieses Konzept funktioniert in einem Formular genauso:




```
<px:PXDateTimeEdit runat="server" ID="edUsrDateTime" DataField="UsrDateTime" Size="SM" />
<px:PXDateTimeEdit runat="server" ID="edUsrDate" DataField="UsrDate" />
```

und innerhalb einer Gitterzelle:

DateTime Value	Date Value
7/11/2017 2:45 PM	7/19/2017

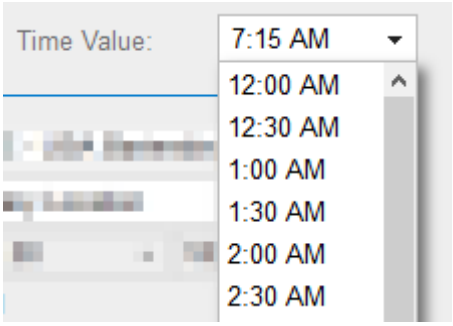
```
<Columns>
    ...
    <px:PXGridColumn DataField="UsrDateTime" Width="130px" />
    <px:PXGridColumn DataField="UsrDate" Width="90px" />
    ...
</Columns>
```

Das PXDBTimeSpan-Attribut

Das Attribut **PXDBTimeSpan** kann mit einem DAC-Feld vom Typ `Nullable<int>` (`int?`) Arbeiten und speichert den Zeitwert in einem DAC-Feld als Anzahl der seit Mitternacht verstrichenen Minuten:

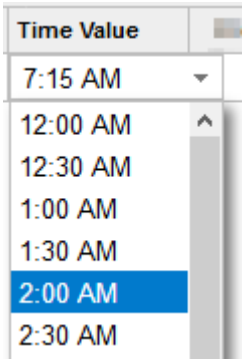
```
#region UsrTimeInt
public abstract class usrTimeInt : IBqlField
{ }

[PXDBTimeSpan(DisplayMask = "t", InputMask = "t")]
[PXUIField(DisplayName = "Time Value")]
public int? UsrTimeInt { get; set; }
#endregion
```

In der Benutzeroberfläche **erstellt** das System für ein Feld, das mit **PXDBTimeSpanAttribute** dekoriert ist, eine Dropdown- **Liste** mit Intervallen von jeweils einer halben Stunde in einem Formular:



und innerhalb einer Gitterzelle:

```
<px:PXDateTimeEdit runat="server" ID="edUsrTimeInt" DataField="UsrTimeInt" TimeMode="true" />
```




```
<px:PXGridColumn DataField="UsrTimeInt" Width="90px" TimeMode="true" />
```

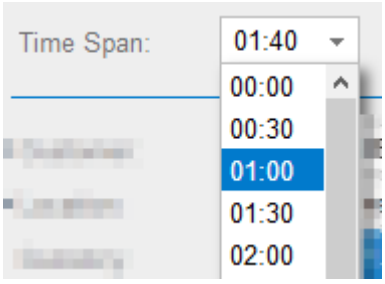
Das PXTimeList-Attribut

Das **PXTimeList-** Attribut kann mit einem DAC-Feld des `Nullable<int>` (`int?`) Arbeiten und speichert den Wert der Zeitspanne in einem DAC-Feld in Minuten.

```
#region UsrTimeSpan
public abstract class usrTimeSpan : IBqlField
{ }
```

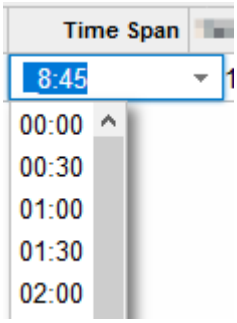
```
[PXDBInt]
[PXTimeList]
[PXUIField(DisplayName = "Time Span")]
public int? UsrTimeSpan { get; set; }
#endregion
```

In der Benutzeroberfläche **erstellt** das System für ein Feld, das mit **PXTimeListAttribute** dekoriert ist, eine Dropdown- **Liste** mit 30-Minuten-Intervallwerten in einem Formular:




```
<px:PXTimeSpan ID="edUsrTimeSpan" runat="server" DataField="UsrTimeSpan" InputMask="hh:mm" />
```

und innerhalb einer Gitterzelle:




```
<RowTemplate>
    ...
    <px:PXTimeSpan ID="edgUsrTimeSpan" runat="server" DataField="UsrTimeSpan"
InputMask="hh:mm" />
    ...
</RowTemplate>
<Columns>
...
    <px:PXGridColumn DataField="UsrTimeSpan" Width="90px" RenderEditorText="True" />
...
</Columns>
```

Datums- und Zeitfelder in Acumatica erstellen online lesen:
https://riptutorial.com/de/acumatica/topic/10783/datums--und-zeitfelder-in-acumatica-erstellen

Kapitel 17: Elemente in einer Dropdown-Liste ändern

Einführung

In diesem Thema erfahren Sie, wie Sie Feldattribute ändern, die von den Attributen PXStringList oder PXIntList geerbt wurden. Der aufgezeigte Ansatz stellt sicher, dass die Funktionalität des Basisprodukts von Acumatica ERP nicht beeinträchtigt wird, und erfordert, wenn überhaupt, nur eine minimale Wartung, während Sie Ihre Anpassungen auf eine neuere Version von Acumatica aktualisieren.

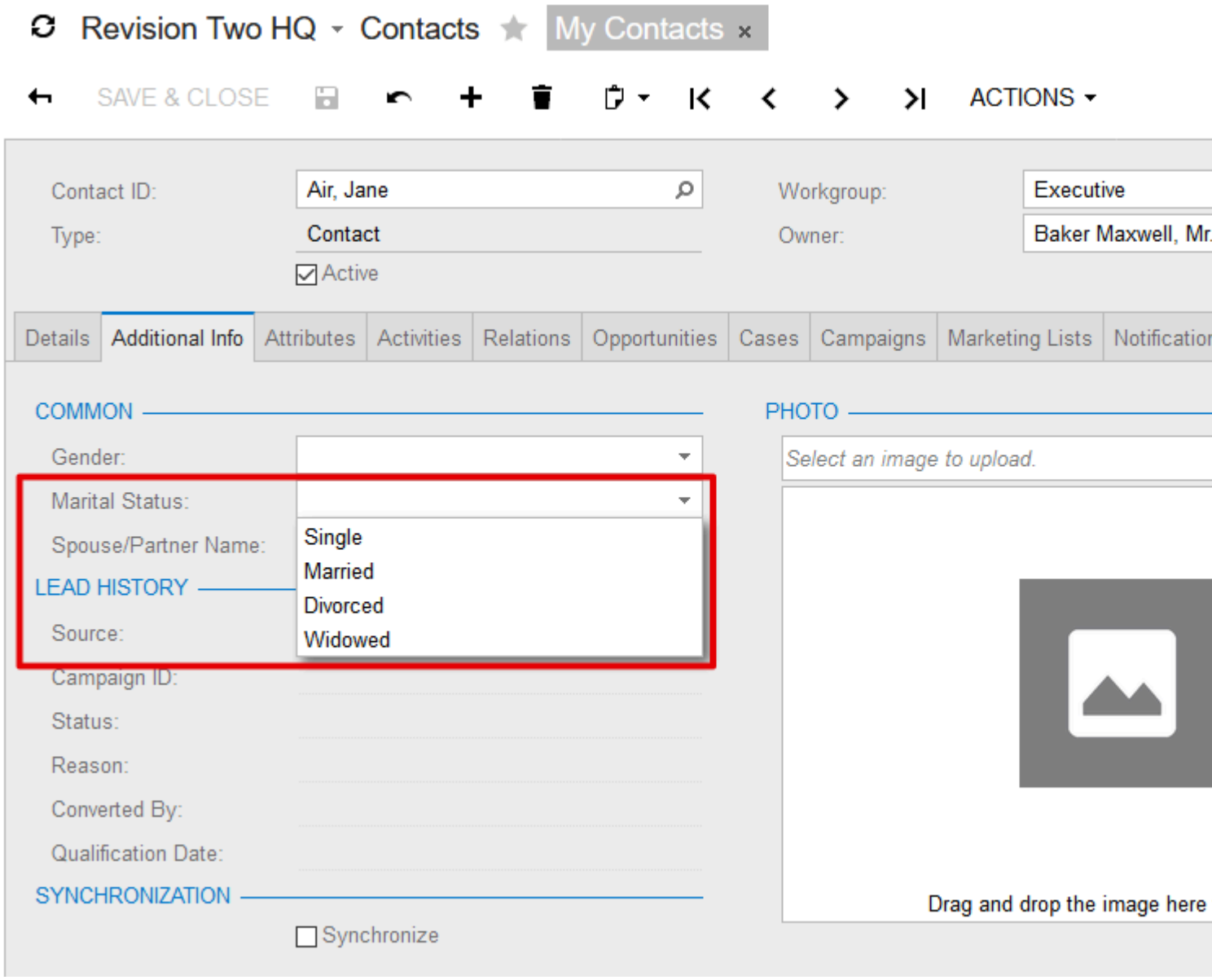
Bemerkungen

In allen obigen Beispielen haben Sie Änderungen an den Arrays `_AllowedValues` und `_AllowedLabels` . Wenn Sie lediglich die Bezeichnung (externer Wert) eines Dropdown-Elements ändern möchten, sollten Sie Übersetzungswörterbücher verwenden. Weitere Informationen zu Übersetzungswörterbüchern finden Sie in der Acumatica ERP-Dokumentation

Examples

Familienstand ändern

In diesem Beispiel werden Sie die **Familienstand** Dropdown-Liste auf der **Kontaktform** (CR302000) zu finden ändern:



So fügen Sie dem PXStringListAttribute-Nachfolger neue Elemente hinzu

Dropdown-Elemente, die innerhalb eines Attributs, das von den Attributen PXStringList oder PXIntList geerbt wurde, hartcodiert werden, können Sie am besten erweitern, indem Sie die Größe der Arrays `_AllowedValues` und `_AllowedLabels` im Konstruktor Ihres benutzerdefinierten `_AllowedValues` `_AllowedLabels` :

```
[PXLocalizable(Messages.Prefix)]
public static class MaritalStatusesMessages
{
    public const string CommonLaw = "Living common law";
    public const string Separated = "Separated (not living common law)";
    public const string DivorcedNoCommonLaw = "Divorced (not living common law)";
    public const string NeverMarried = "Never Married";
}

public class MaritalStatusesCst1Attribute : MaritalStatusesAttribute
{
```

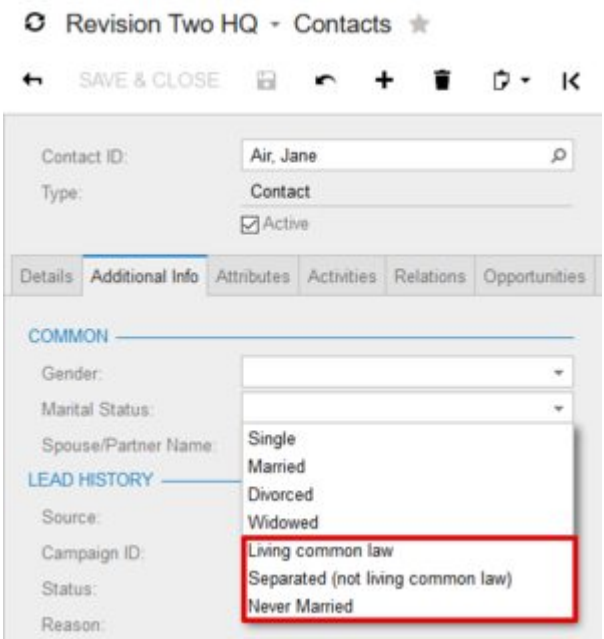
```
    public const string CommonLaw = "L";
    public const string Separated = "P";
    public const string NeverMarried = "N";

    public MaritalStatusesCst1Attribute()
    {
        Array.Resize(ref _AllowedValues, _AllowedValues.Length + 3);
        _AllowedValues[_AllowedValues.Length - 3] = CommonLaw;
        _AllowedValues[_AllowedValues.Length - 2] = Separated;
        _AllowedValues[_AllowedValues.Length - 1] = NeverMarried;
        Array.Resize(ref _AllowedLabels, _AllowedLabels.Length + 3);
        _AllowedLabels[_AllowedLabels.Length - 3] = MaritalStatusesMessages.CommonLaw;
        _AllowedLabels[_AllowedLabels.Length - 2] = MaritalStatusesMessages.Separated;
        _AllowedLabels[_AllowedLabels.Length - 1] = MaritalStatusesMessages.NeverMarried;
    }
}
```

In dem obigen Beispiel haben Sie die Größe der Arrays `_AllowedValues` und `_AllowedLabels` erhöht, um der Dropdown-Liste **Familienstatus** 3 zusätzliche Elemente hinzuzufügen. Interne Werte, die im Array `_AllowedValues` gespeichert sind, werden DAC-Feldern zugewiesen und in der Datenbank gespeichert. Externe Werte aus dem Array `_AllowedValues` repräsentieren interne Werte in der Benutzeroberfläche.

Um die Ergebnisse zu testen, dekorieren **Sie** in der Erweiterung Contact DAC **das** Feld **MaritalStatus** mit dem `MaritalStatusesCst1Attribute` :

```
public class ContactExt : PXCacheExtension<Contact>
{
    [PXRemoveBaseAttribute(typeof(MaritalStatusesAttribute))]
    [PXMergeAttributes(Method = MergeMethod.Append)]
    [MaritalStatusesCst1]
    public string MaritalStatus { get; set; }
}
```

Nun gibt es 7 Elemente in der Dropdown-Liste **Familienstand** :



So entfernen Sie Elemente, die im Nachfolger PXStringListAttribute deklariert sind

Um ein bestimmtes Dropdown-Element zu entfernen, das in einem vom PXStringList- oder PXIntList-Attribut geerbten Attribut hartcodiert wurde, müssen Sie die Größe der `_AllowedValues` und `_AllowedLabels` Arrays im Konstruktor des benutzerdefinierten `_AllowedValues` `_AllowedLabels` :

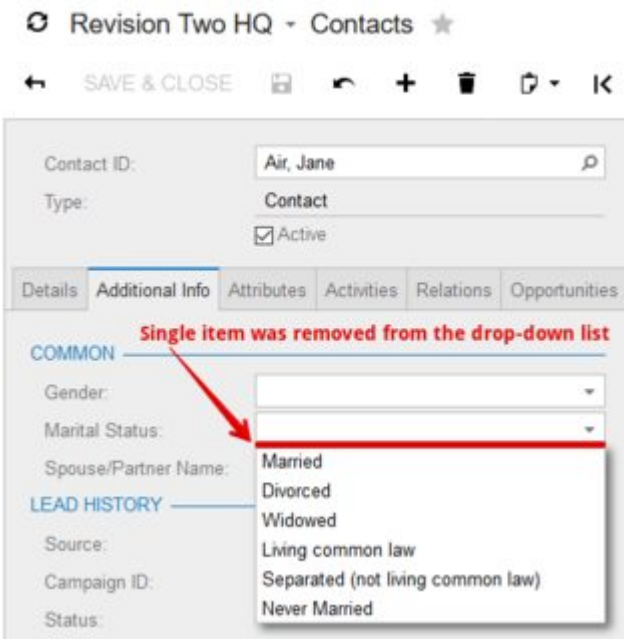
```
public class MaritalStatusesCst2Attribute : MaritalStatusesCst1Attribute
{
    public MaritalStatusesCst2Attribute()
    {
        string[] allowedValues = new string[_AllowedValues.Length - 1];
        string[] allowedLabels = new string[_AllowedLabels.Length - 1];
        Array.Copy(_AllowedValues, 1, allowedValues, 0, _AllowedValues.Length - 1);
        Array.Copy(_AllowedLabels, 1, allowedLabels, 0, _AllowedValues.Length - 1);
        _AllowedValues = allowedValues;
        _AllowedLabels = allowedLabels;
    }
}
```

In der Probe über verringert Sie Größe des `_AllowedValues` und `_AllowedLabels` Arrays ***einzelnes*** Element aus dem **Familienstand** der Dropdown-Liste zu entfernen.

Um die Ergebnisse zu testen, dekorieren **Sie** in der Erweiterung Contact DAC **das** Feld **MaritalStatus** mit dem `MaritalStatusesCst2Attribute` :

```
public class ContactExt : PXCacheExtension<Contact>
{
    [PXRemoveBaseAttribute(typeof(MaritalStatusesAttribute))]
    [PXMergeAttributes(Method = MergeMethod.Append)]
    [MaritalStatusesCst2]
    public string MaritalStatus { get; set; }
}
```

Jetzt gibt es nur noch 6 Elemente: 3 Original und 3 Brauch - in der Dropdown-Liste **Familienstand** :



Ersetzen von Elementen, die im Nachfolger PXStringListAttribute deklariert sind

Um ein bestimmtes Dropdown-Element zu ersetzen, das ursprünglich in einem vom PXStringList- oder PXIntList-Attribut geerbten Attribut hartcodiert ist, müssen Sie den entsprechenden Wert in den Arrays `_AllowedValues` und `_AllowedLabels` im Konstruktor Ihres benutzerdefinierten `_AllowedValues` `_AllowedLabels` :

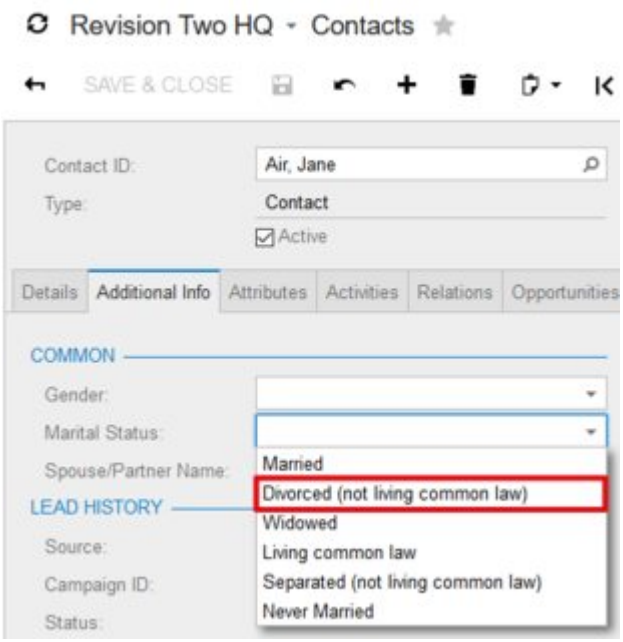
```
public class MaritalStatusesCst3Attribute : MaritalStatusesCst2Attribute
{
    public const string DivorcedNoCommonLaw = "V";

    public MaritalStatusesCst3Attribute()
    {
        _AllowedValues[Array.IndexOf(_AllowedValues, Divorced)] = DivorcedNoCommonLaw;
        _AllowedLabels[Array.IndexOf(_AllowedLabels, Messages.Divorced)] =
MaritalStatusesMessages.DivorcedNoCommonLaw;
    }
}
```

In dem obigen Beispiel haben Sie ***D*** - ***Divorced*** Item durch ***V*** - ***Divorced (nicht geläufiges Common Law)*** in den Arrays `_AllowedValues` und `_AllowedLabels` ersetzt. Auf diese Weise ersetzen wir sowohl interne als auch externe Werte eines Dropdown-Objekts.

Um die Ergebnisse zu testen, dekorieren **Sie** in der Erweiterung Contact DAC **das** Feld **MaritalStatus** mit dem `MaritalStatusesCst3Attribute` :

```
public class ContactExt : PXCacheExtension<Contact>
{
    [PXRemoveBaseAttribute(typeof(MaritalStatusesAttribute))]
    [PXMergeAttributes(Method = MergeMethod.Append)]
    [MaritalStatusesCst3]
    public string MaritalStatus { get; set; }
}
```

Jetzt gibt es nur noch 6 Elemente: 2 Original, 3 Brauch und 1 ersetzt - in der Dropdown-Liste **Familienstand** :



Elemente in einer Dropdown-Liste ändern online lesen:
https://riptutorial.com/de/acumatica/topic/9392/elemente-in-einer-dropdown-liste-andern

Kapitel 18: Filtern mit mehreren Werten mit nur einem Selektor

Einführung

Hier haben Sie die Möglichkeit, mehrere Werte innerhalb eines Selektors zu haben, um ein Gitter zu filtern.

Examples

Kundenauftrag für Mehrfachkunden abrufen

Beim Versuch, einen Datensatz mit mehreren Werten in einem Selektor zu filtern. Zuerst müssen Sie den px: PXMultiSelector auf der aspx-Seite anstelle des normalen px: PXSelector verwenden. Danach müssen Sie sich ein Diagramm erstellen, das mindestens drei Ansichten und einen Ansichtsdelegierten enthält. Sie benötigen außerdem mindestens einen einfachen ungebundenen DAC.

Hier ist eine Beispielseite mit dem px: PXMultiSelector:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPages/FormDetail.master" AutoEventWireup="true"
ValidateRequest="false" CodeFile="TT000000.aspx.cs" Inherits="Page_TT000000" Title="Untitled
Page" %>

<%@ MasterType VirtualPath="~/MasterPages/FormDetail.master" %>

<asp:Content ID="cont1" ContentPlaceHolderID="phDS" runat="Server">
<px:PXDataSource ID="ds" runat="server" Visible="True" Width="100%"
    TypeName="MultiSelector.MultiInquiry"
    PrimaryView="MasterView">
    <CallbackCommands>
    </CallbackCommands>
</px:PXDataSource>
</asp:Content>
<asp:Content ID="cont2" ContentPlaceHolderID="phF" runat="Server">
<px:PXFormView ID="form" runat="server" DataSourceID="ds" DataMember="MasterView" Width="100%"
Height="100px" AllowAutoHide="false">
    <Template>
        <px:PXMultiSelector ID="edInventoryID" runat="server" Width="100%" DataSourceID="ds"
DataField="Customer" CommitChanges="True"></px:PXMultiSelector>
    </Template>
</px:PXFormView>
</asp:Content>
<asp:Content ID="cont3" ContentPlaceHolderID="phG" runat="Server">
<px:PXGrid ID="grid" runat="server" DataSourceID="ds" Width="100%" Height="150px"
SkinID="Details" AllowAutoHide="false">
    <Levels>
        <px:PXGridLevel DataMember="DetailsView">
            <Columns>
                <px:PXGridColumn DataField="OrderType" Width="70"></px:PXGridColumn>
                <px:PXGridColumn DataField="OrderNbr" Width="200"></px:PXGridColumn>
```

```
                <px:PXGridColumn DataField="OrderDesc" Width="100"></px:PXGridColumn>
                <px:PXGridColumn DataField="CustomerOrderNbr" Width="100"></px:PXGridColumn>
                <px:PXGridColumn DataField="Status" Width="100"></px:PXGridColumn>
                <px:PXGridColumn DataField="RequestDate" Width="100"></px:PXGridColumn>
                <px:PXGridColumn DataField="ShipDate" Width="100"></px:PXGridColumn>
                <px:PXGridColumn DataField="CustomerID" Width="100"></px:PXGridColumn>
            </Columns>
        </px:PXGridLevel>
    </Levels>
    <AutoSize Container="Window" Enabled="True" MinHeight="150" />
    <ActionBar>
    </ActionBar>
</px:PXGrid>
</asp:Content>
```

Hier ist das Beispieldiagramm mit den Ansichten und dem Delegierten.

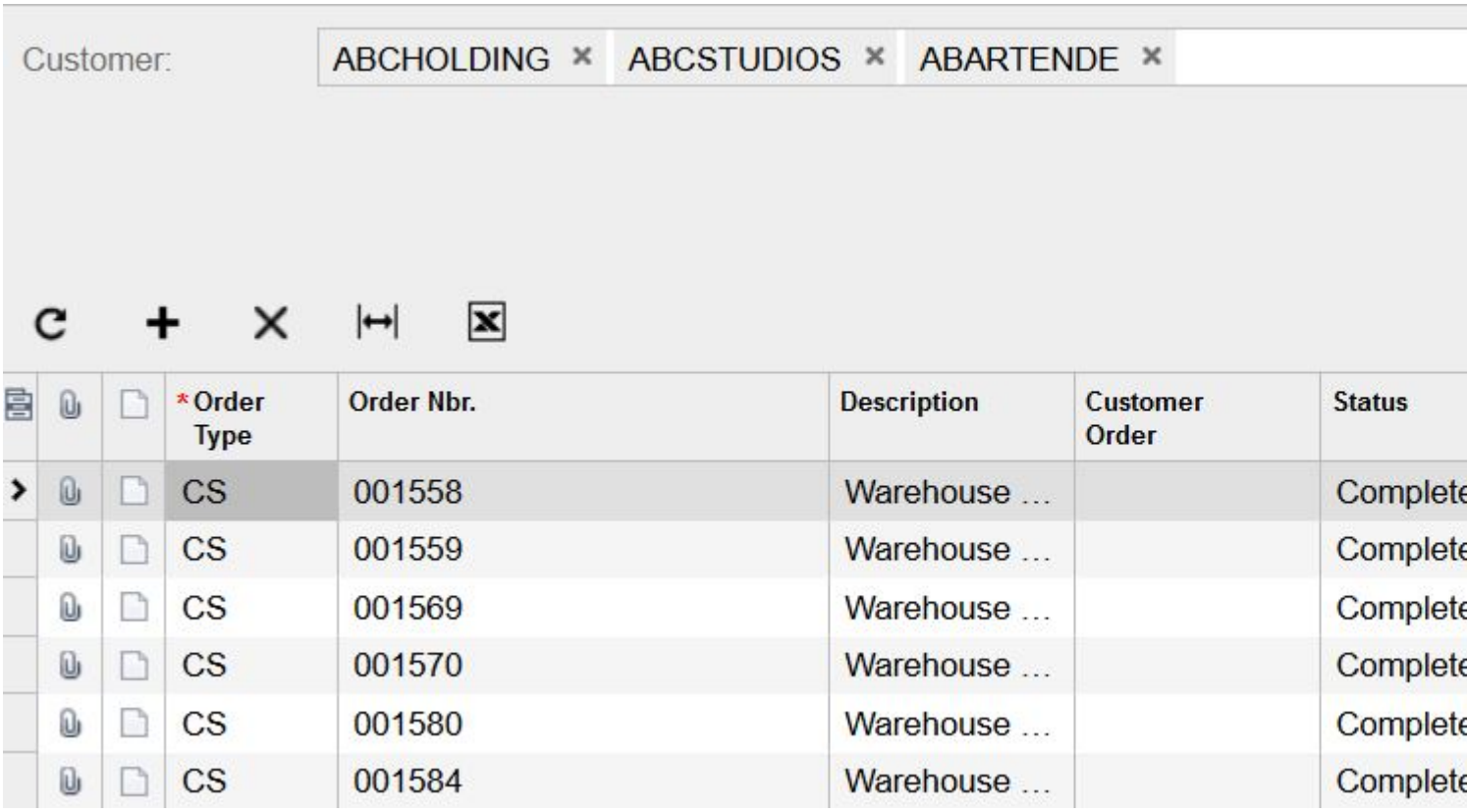
```
public class MultiInquiry : PXGraph<MultiInquiry>
{
    public PXCancel<MasterTable> Cancel;
    public PXFilter<MasterTable> MasterView;
    public PXSelect<SOOrder> DetailsView;

    public PXSelectJoin<SOOrder, LeftJoin<BAccount, On<SOOrder.customerID,
Equal<BAccount.bAccountID>>>, Where<BAccount.acctCD, In<Required<BAccount.acctCD>>>> Orders2;

    protected virtual IEnumerable detailsView()
    {
        var list = new List<SOOrder>();
        var customers = MasterView.Current.Customer;
        if (customers != null)
        {
            List<string> customerList = new List<string>();
            customerList.AddRange(customers.Split(new string[] { "; " },
StringSplitOptions.None));
            object[] val = new object[] { customerList.ToArray() };

            foreach (PXResult<SOOrder> res in Orders2.Select(val))
            {
                SOOrder order = res;
                list.Add(order);
            }
        }
        return list;
    }
}
```

Dazu fügen wir den DAC hinzu, der die Definition für das Feld enthält, das im MultiSelector verwendet wird, und die Konstante, um nur Kundenkonten auszuwählen.

```
    [Serializable]
    public class MasterTable : IBqlTable
    {
        #region InventoryID
        public abstract class customer : IBqlField { }
        [PXString()]
        [PXUIField(DisplayName = "Customer")]
        [PXSelector(typeof(Search<BAccount.acctCD, Where<BAccount.type,
```

```
Equal<CustomerType>>>), ValidateValue = false)]
        public virtual string Customer { get; set; }
        #endregion

    }

    public class CustomerType : Constant<string> { public CustomerType() : base("CU") { } }
```

Und das Ergebnis für dieses Beispiel könnte so aussehen:



Filtern mit mehreren Werten mit nur einem Selektor online lesen: https://riptutorial.com/de/acumatica/topic/10707/filtern-mit-mehreren-werten-mit-nur-einem-selektor

Kapitel 19: Frachtberechnung

Einführung

Mit Acumatica ERP können Sie die Fracht verwalten, um zusätzliche Kosten und Einnahmen aus Verkaufstransaktionen besser zu kontrollieren. Der Frachtbetrag, den Sie Ihren Kunden in Rechnung stellen, umfasst möglicherweise nicht nur die Fracht, die Ihr Unternehmen von den Spediteuren berechnet, sondern auch die durch Ihre Versandbedingungen und Prämienfracht bestimmten Versicherungs-, Bearbeitungs- und Verpackungsgebühren.

Examples

Überschreibung des Frachtbetrags in Versand und Rechnung

Acumatica ermöglicht das Erstellen und Verwalten der Liste der Versandbedingungen im System. Versandbedingungen werden verwendet, um die Versand-, Verpackungs- und Bearbeitungskosten in Abhängigkeit von der Versandmenge festzulegen.

In diesem Beispiel werde ich zeigen, wie der Frachtbetrag für eine Sendung anhand des Verkaufsauftragsbetrags berechnet wird. Dadurch können Benutzer mehrere Sendungen pro Verkaufsauftrag erstellen, wobei dieselben Versandbedingungen automatisch für alle Sendungen gelten.

FreightCalculator

Die `FreightCalculator` Klasse ist für die Berechnung der Frachtkosten und Frachtbedingungen verantwortlich. In diesem Beispiel liegt unser Interesse nur in der `GetFreightTerms` Methode:

```
public class FreightCalculator
{
    ...

    protected virtual ShipTermsDetail GetFreightTerms(string shipTermsID, decimal? lineTotal)
    {
        return PXSelect<ShipTermsDetail,
            Where<ShipTermsDetail.shipTermsID, Equal<Required<SOOrder.shipTermsID>>,
            And<ShipTermsDetail.breakAmount, LessEqual<Required<SOOrder.lineTotal>>>>,
            OrderBy<Desc<ShipTermsDetail.breakAmount>>>.Select(graph, shipTermsID, lineTotal);
    }

    ...
}
```

Sowohl die **Kundenaufträge** und die **Sendungen** Bildschirme nutzen `FreightCalculator` Klasse Frachtmenge zu berechnen , basierend auf Kundenauftrag des und Versand der Betrag jeweils:

Verkaufsaufträge

```
public class SOOrderEntry : PXGraph<SOOrderEntry, SOOrder>, PXImportAttribute.IPXPrepareItems
{
    ...

    public virtual FreightCalculator CreateFreightCalculator()
    {
        return new FreightCalculator(this);
    }

    ...

    protected virtual void SOOrder_RowUpdated(PXCache sender, PXRowUpdatedEventArgs e)
    {
        ...

        PXResultset<SOLine> res = Transactions.Select();
        FreightCalculator fc = CreateFreightCalculator();
        fc.CalcFreight<SOOrder, SOOrder.curyFreightCost, SOOrder.curyFreightAmt>(sender,
(SOOrder)e.Row, res.Count);

        ...
    }

    ...
}
```

Sendungen

```
public class SOShipmentEntry : PXGraph<SOShipmentEntry, SOShipment>
{
    ...

    protected virtual FreightCalculator CreateFreightCalculator()
    {
        return new FreightCalculator(this);
    }

    ...

    protected virtual void SOShipment_RowUpdated(PXCache sender, PXRowUpdatedEventArgs e)
    {
        ...

        PXResultset<SOShipLine> res = Transactions.Select();
        ...
        FreightCalculator fc = CreateFreightCalculator();
        fc.CalcFreight<SOShipment, SOShipment.curyFreightCost,
SOShipment.curyFreightAmt>(sender, (SOShipment)e.Row, res.Count);

        ...
    }

    ...
}
```

Frachtbetrag überschreiben

Um anzupassen , wie Acumatica Frachtbetrag berechnet auf die **Sendungen** Bildschirm werde ich erklären `FreightCalculatorCst` Klasse geerbt von `FreightCalculator` und außer Kraft setzen `GetFreightTerms` Methode:

```
public class FreightCalculatorCst : FreightCalculator
{
    public FreightCalculatorCst(PXGraph graph)
        : base(graph)
    {
    }

    protected override ShipTermsDetail GetFreightTerms(string shipTermsID, decimal? lineTotal)
    {
        if (graph is SOShipmentEntry)
        {
            var shipmentEntry = graph as SOShipmentEntry;
            int orderCount = 0;
            decimal? lineTotalTemp = null;

            foreach (PXResult<SOOrderShipment, SOOrder, CurrencyInfo, SOAddress, SOContact,
SOOrderType> orderRec in
                shipmentEntry.OrderList.SelectWindowed(0, 2))
            {
                orderCount++;
                SOOrder order = (SOOrder)orderRec;
                if (orderCount == 1)
                    lineTotalTemp = order.LineTotal;
                else
                    break;
            }

            if (orderCount == 1)
            {
                lineTotal = lineTotalTemp;
            }
        }

        return base.GetFreightTerms(shipTermsID, lineTotal);
    }
}
```

Danach werde ich eine Verlängerung für die Umsetzung `SOShipmentEntry` BLC und außer Kraft setzen `CreateFreightCalculator` Methode zu ersetzen `FreightCalculator` mit meiner benutzerdefinierten `FreightCalculatorCst` Klasse auf dem Bildschirm **Sendungen:**

```
public class SOShipmentEntryExt : PXGraphExtension<SOShipmentEntry>
{
    [PXOverride]
    public FreightCalculator CreateFreightCalculator()
    {
        return new FreightCalculatorCst(Base);
    }
}
```

Informationen zur Implementierung der FreightCalculatorCst-Klasse im obigen Beispiel

In der überschriebenen Methode `GetFreightTerms` werde ich den Betrag aus dem Verkaufsauftrag anstelle des `GetFreightTerms` Methode `GetFreightTerms` aufzurufen und die Versandbedingungen zu erhalten:

```
foreach (PXResult<SOOrderShipment, SOOrder, CurrencyInfo, SOAddress, SOContact, SOOrderType>
orderRec in
    shipmentEntry.OrderList.SelectWindowed(0, 2))
{
    orderCount++;
    SOOrder order = (SOOrder)orderRec;
    if (orderCount == 1)
        lineTotalTemp = order.LineTotal;
    else
        break;
}

if (orderCount == 1)
{
    lineTotal = lineTotalTemp;
}
```

Offensichtlich kann der Kundenauftragsbetrag nur zur Berechnung des Frachtbetrags für Sendungen verwendet werden, die nur eine Bestellung erfüllen. Wenn eine Sendung mehrere Bestellungen erfüllt, müssen wir das Verhalten des Basisprodukts verfolgen und den Frachtbetrag basierend auf dem Versandbetrag berechnen. Um die Anzahl der Bestellungen zu prüfen, die die Lieferung erfüllt, habe ich die `SelectWindowed` Methode in der `OrderList` verwendet und die ersten 2 Bestellungen angefordert, die von der aktuellen Lieferung erfüllt wurden. Ich hätte alle Bestellungen anfordern können, die von der Sendung erfüllt wurden. Dies würde jedoch erheblich mehr Zeit in Anspruch nehmen und zu vielen Datensätzen zurückkehren als erforderlich, um zu überprüfen, ob der Auftragsbetrag anstelle des Sendebetrags zur Berechnung der Fracht verwendet werden kann.

Frachtberechnung online lesen: https://riptutorial.com/de/acumatica/topic/9044/frachtberechnung

Kapitel 20: Größe des Auswahlfensters ändern

Einführung

In diesem Thema erfahren Sie, wie Sie die Größe des Auswahlfelds ändern. Jeder Wahlschalter in Acumatica hat eine Schaltfläche, die mit einem Lupensymbol gekennzeichnet ist. Durch Klicken auf diese Schaltfläche können Benutzer ein Dropdown-Fenster mit einer Liste der zur Auswahl stehenden Objekte öffnen.

Examples

Ändern der Standardgrößenbereiche für das Auswahlfeld

Die folgenden vier Eigenschaften sind für die **Eingabesteuerelemente PXSelector** und **PXSegmentMask** verfügbar, um den Größenbereich für ein Dropdown-Fenster zu definieren:

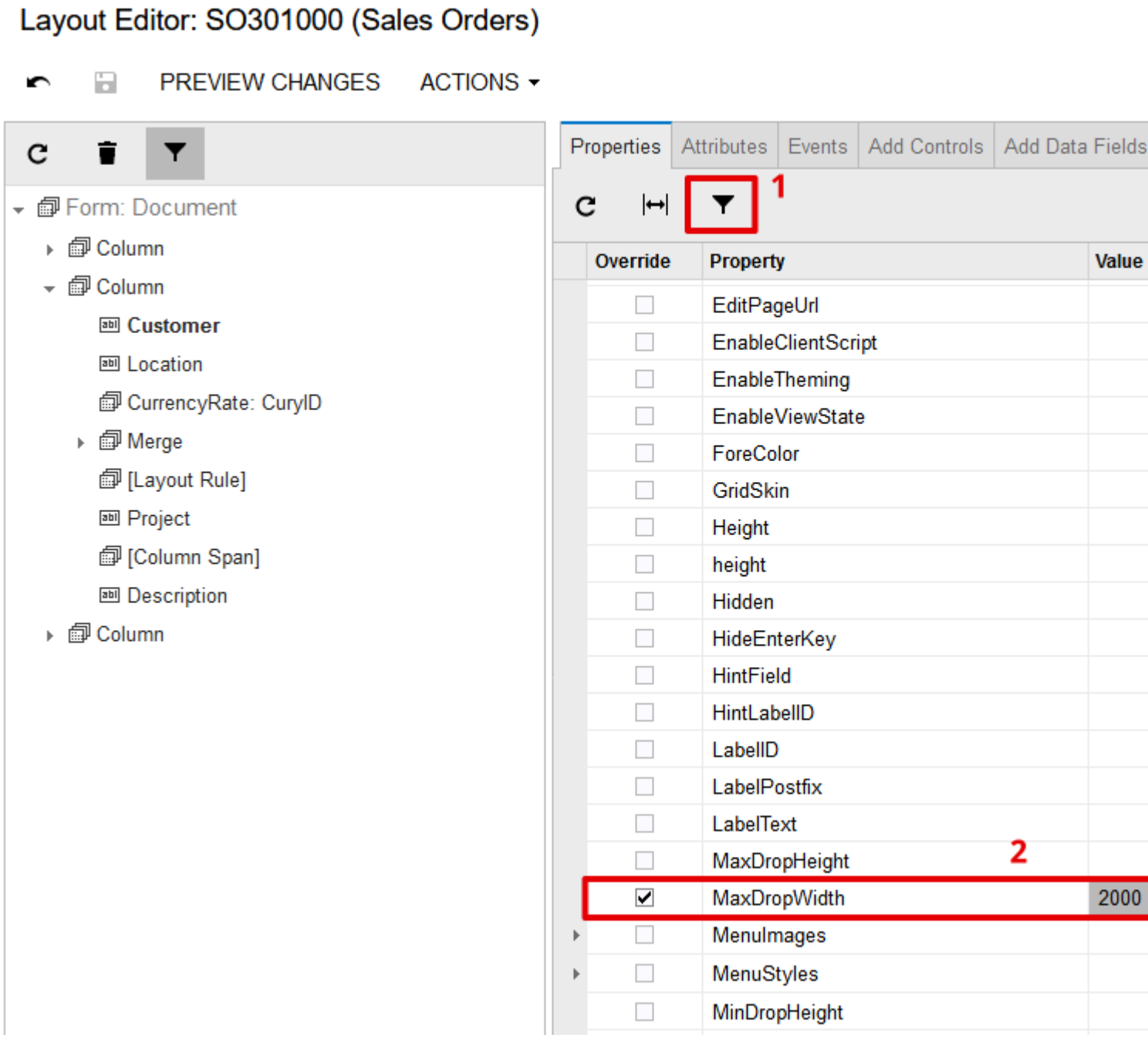
- **MinDropWidth** : **Ruft** die minimale Dropdown-Kontrollbreite ab oder legt diese fest
- **MinDropHeight** : **Ermittelt** oder legt die minimale Dropdown- **Steuerhöhe fest**
- **MaxDropWidth** : **Ruft** die maximale Dropdown-Kontrollbreite ab oder legt diese fest
- **MaxDropHeight** : **Ruft** die maximale Dropdown-Kontrollhöhe ab oder legt diese fest

Bitte beachten Sie, dass die vier oben aufgelisteten Eigenschaften im Eigenschaftenfenster ausgeblendet sind und von IntelliSense beim Bearbeiten von Aspx-Seiten in Visual Studio nicht vorgeschlagen werden.

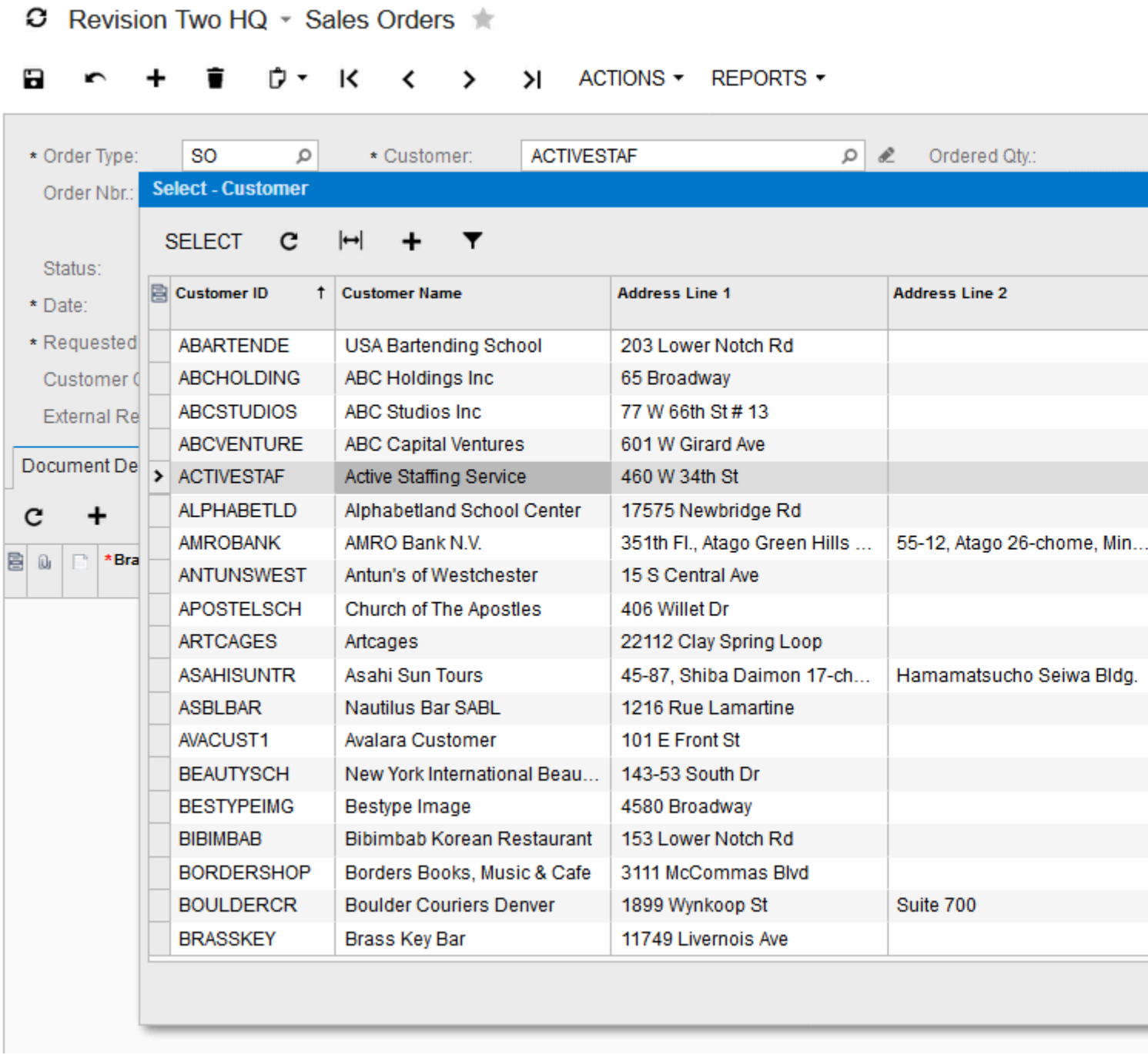
Um die Dropdown-Fensterbreite des Customer-Selektors zu erweitern

Das 13-Spalten-Standardlayout, das für den **Customer** Selector auf dem Bildschirm **Sales Orders** (SO.30.10.00) definiert wurde, passt nicht ganz zum Standardgrößenbereich, der für das Dropdown-Fenster des Selektors angegeben ist. Damit Benutzer möglichst viele Informationen ausloten und beim horizontalen Scrollen Zeit sparen, um alle Spalten **anzuzeigen** , müssen Sie die maximale Dropdown-Kontrollbreite vergrößern, indem Sie der **MaxDropWidth-** Eigenschaft von für den **Customer-** Selector einen größeren **Wert zuweisen** .

Um den Wert für die **MaxDropWidth-** Eigenschaft im Layout-Editor festzulegen, deaktivieren Sie das Optionsfeld **Erweiterte Eigenschaften ausblenden,** wie in der Abbildung unten gezeigt:



Nach der Veröffentlichung der Anpassung können Benutzer das neue Layout von **Customer** Selector genießen, das nun auf den gesamten Arbeitsbereich erweitert wird:



Größe des Auswahlfensters ändern online lesen:
https://riptutorial.com/de/acumatica/topic/9524/gro-e-des-auswahlfensters-andern

Kapitel 21: Herunterladen von Dateien, die an eine Detailentität angehängt sind, mithilfe einer vertragsbasierten API

Einführung

In diesem Thema wird beschrieben, wie Sie Dateien herunterladen, die an eine Detailentität in Acumatica ERP mithilfe der vertragsbasierten API angefügt sind.

Bemerkungen

Das obige Code-Snippet wurde mit dem Json.NET-Framework (**Newtonsoft.Json.dll**) erstellt.

Um einen HTTP-Cookie-Header aus einer SOAP-Antwort zu erhalten, fügen Sie einen Verweis auf die **Assemblys System.ServiceModel** und **System.ServiceModel.**

```
using System.ServiceModel;
using System.ServiceModel.Web;
```

Examples

HTTP-Cookie-Header aus einer von SOAP- und REST-Clients gemeinsam genutzten SOAP-Antwort

In der vertragsbasierten SOAP-API von Acumatica gibt es eine Einschränkung, die das Herunterladen von Anhängen nur für eine Entität der obersten Ebene ermöglicht. Jeder Versuch, die GetFiles () -Methode zu verwenden, um die Anhänge einer Detailentität abzurufen , führt leider zu dem Fehler " **Entität ohne Bildschirmbindung kann nicht als Entität der obersten Ebene verwendet werden.** " Entität der Ebene, die im Web-Service-Endpunkt definiert ist.

Eine weitere Einschränkung bei der GetFiles () - Methode besteht darin, dass immer der Inhalt aller an eine Entität angehängten Dateien zurückgegeben wird. Es gibt keine Option, nur die Dateinamen abzurufen und dann zu entscheiden, welche Datei (en) von Acumatica heruntergeladen werden sollen.

Glücklicherweise gibt es eine bessere und besser kontrollierbare Möglichkeit, mit Anhängen zu arbeiten, die mit der vertragsbasierten REST-API bereitgestellt werden. Die `files` Array zurückgegeben als Teil eines jeden von dem vertragsbasierte REST API exportierte Einheit enthält nur:

- Dateinamen (die Eigenschaft **Dateiname**)
- Dateikennungen (die **ID-** Eigenschaft)
- Hypertextverweise (die Eigenschaft **href**), die später zum Herunterladen von Dateiinhalten

verwendet werden können

Ein Beispiel für das Abrufen einer Liste von Dateien, die an eine Entität angehängt sind, vom Web-Service-Endpunkt und das Abrufen bestimmter Dateiinhalte über die vertragsbasierte REST-API finden Sie in der Acumatica-Produkthilfe

Wie kann man die an eine Detaileinheit angehängten Dateien herunterladen, wenn das gesamte Integrationsprojekt mit der vertragsbasierten SOAP-API entwickelt wurde? Wie im folgenden Codeausschnitt gezeigt, ist es möglich, den HTTP-Cookie-Header von einer SOAP-Antwort an den REST-API-Client zu übergeben, der ausschließlich für die Arbeit mit den Anhängen verwendet wird:

```
using (var soapClient = new DefaultSoapClient())
{
    var address = new Uri("http://localhost/AcumaticaERP/entity/Default/6.00.001/");
    CookieContainer cookieContainer;
    using (new OperationContextScope(soapClient.InnerChannel))
    {
        soapClient.Login(login, password, null, null, null);
        string sharedCookie = WebOperationContext.Current.IncomingResponse.Headers["Set-
Cookie"];
        cookieContainer = new CookieContainer();
        cookieContainer.SetCookies(address, sharedCookie);
    }
    try
    {
        var shipment = new Shipment()
        {
            ShipmentNbr = new StringSearch { Value = "001301" },
            ReturnBehavior = ReturnBehavior.OnlySpecified
        };
        shipment = soapClient.Get(shipment) as Shipment;

        var restClient = new HttpClient(
            new HttpClientHandler
            {
                UseCookies = true,
                CookieContainer = cookieContainer
            });
        restClient.BaseAddress = address;// new
Uri("http://localhost/059678/entity/Default/6.00.001/");

        var res = restClient.GetAsync("Shipment/" + shipment.ID + "?$expand=Packages")
            .Result.EnsureSuccessStatusCode();
        var shipmentWithPackages = res.Content.ReadAsStringAsync().Result;

        JObject jShipment = JObject.Parse(shipmentWithPackages);
        JArray jPackages = jShipment.Value<JArray>("Packages");
        foreach (var jPackage in jPackages)
        {
            JArray jFiles = jPackage.Value<JArray>("files");
            string outputDirectory = ".\\Output\\";
            if (!Directory.Exists(outputDirectory))
            {
                Directory.CreateDirectory(outputDirectory);
            }

            foreach (var jFile in jFiles)
```

```
            {
                string fullFileName = jFile.Value<string>("filename");
                string fileName = Path.GetFileName(fullFileName);
                string href = jFile.Value<string>("href");

                res = restClient.GetAsync(href).Result.EnsureSuccessStatusCode();
                byte[] file = res.Content.ReadAsByteArrayAsync().Result;
                System.IO.File.WriteAllBytes(outputDirectory + fileName, file);
            }
        }
    }
    finally
    {
        soapClient.Logout();
    }
}
```

Herunterladen von Dateien, die an eine Detailentität angehängt sind, mithilfe einer vertragsbasierten API online lesen: https://riptutorial.com/de/acumatica/topic/10692/herunterladen-von-dateien--die-an-eine-detailentitat-angehangt-sind--mithilfe-einer-vertragsbasierten-api

Kapitel 22: Hinzufügen einer Attributunterstützung zu einer Out-of-Box-Bestellungseinheit

Einführung

Mit Acumatica ERP können Sie Attribute für eine flexible, aussagekräftige Klassifizierung eines Unternehmens (Lead, Stock / Non Stock-Produkte usw.) gemäß den Anforderungen Ihres Unternehmens definieren. Ein Attribut ist eine Eigenschaft, mit der Sie zusätzliche Informationen für Objekte im System angeben können. Attribute werden im Kontext einer Klasse definiert, bei der es sich um eine Gruppierung der Geschäftskonten (einschließlich Leads, Opportunities, Kunden und Kunden), Bestand und Nicht-Lagerartikel nach einer oder mehreren Eigenschaften handelt.

Bemerkungen

Dieses Beispiel gilt für die Acumatica 6.0-Serie

Examples

In diesem Artikel finden Sie eine Anleitung zum Hinzufügen von Acumatica ERP-Attributunterstützung zur werkseitigen Verkaufsauftragsentität

Im Kern muss der Entity-Haupt-DAC über eine GUID-Spalte (`NoteID`) verfügen, um auf die `CSAnswers` Tabelle zu `CSAnswers` , und über ein Feld verfügen, das die Klasse der Entität `CSAnswers` .

Wir werden den `Order Type` , um eine Liste von Attributen zu definieren, um bestimmte auftragstypspezifische Informationen zu sammeln.

Erstellen Sie eine Diagrammerweiterung für `SOOrderTypeMaint` Graph und deklarieren Sie die Datenansicht, um eine Liste von Attributen für einen bestimmten Auftragstyp zu definieren. Wir werden `CSAttributeGroupList<TEntityClass, TEntity>`

```
public class SOOrderTypeMaintPXExt : PXGraphExtension<SOOrderTypeMaint>
{
    [PXViewName(PX.Objects.CR.Messages.Attributes)]
    public CSAttributeGroupList<SOOrderType, SOOrder> Mapping;
}
```

Erstellen Sie eine Diagrammerweiterung für `SOOrderEntry` Diagramm, und deklarieren Sie die Datenansicht für Attribute, die für den aktuellen Auftragstyp spezifisch sind.

```
public class SOOrderEntryPXExt : PXGraphExtension<SOOrderEntry>
```

```
{
    public CRAttributeList<SOOrder> Answers;
}
```

Erstellen DAC - Erweiterung für `SOOrder` DAC und erklären benutzerdefiniertes Feld mit verziertem `CRAttributesField` Attribute und geben Sie die `ClassID` Feld - in unserem Fall ist es `OrderType` .

```
public class SOOrderPXExt : PXCacheExtension<SOOrder>
{
    #region UsrAttributes

    public abstract class usrAttributes : IBqlField { }

    [CRAttributesField(typeof(SOOrder.orderType))]
    public virtual string[] UsrAttributes { get; set; }

    #endregion
}
```

Ändern `Order Types` Seite (`SO201000`) , wie unten Customization Engine

```
<px:PXTabItem Text="Attributes">
  <Template>
    <px:PXGrid runat="server" BorderWidth="0px" Height="150px" SkinID="Details" Width="100%"
ID="AttributesGrid"
                MatrixMode="True" DataSourceID="ds">
        <AutoSize Enabled="True" Container="Window" MinHeight="150" />
        <Levels>
            <px:PXGridLevel DataMember="Mapping">
                <RowTemplate>
                    <px:PXSelector runat="server" DataField="AttributeID"
FilterByAllFields="True" AllowEdit="True"
                                   CommitChanges="True" ID="edAttributeID" /></RowTemplate>
                <Columns>
                    <px:PXGridColumn DataField="AttributeID" Width="81px" AutoCallBack="True"
LinkCommand="ShowDetails" />
                    <px:PXGridColumn DataField="Description" Width="351px" AllowNull="False"
/>
                    <px:PXGridColumn DataField="SortOrder" TextAlign="Right" Width="81px" />
                    <px:PXGridColumn DataField="Required" Type="CheckBox" TextAlign="Center"
AllowNull="False" />
                    <px:PXGridColumn DataField="CSAttribute__IsInternal" Type="CheckBox"
TextAlign="Center" AllowNull="True" />
                    <px:PXGridColumn DataField="ControlType" Type="DropDownList" Width="81px"
AllowNull="False" />
                    <px:PXGridColumn DataField="DefaultValue" RenderEditorText="False"
Width="100px" AllowNull="True" />
                </Columns>
            </px:PXGridLevel>
        </Levels>
    </px:PXGrid>
  </Template>
</px:PXTabItem>
```

Ändern `Sales Orders` Seite (`SO301000`) , wie unten Customization Engine

```
<px:PXTabItem Text="Attributes">
```

```
  <Template>
    <px:PXGrid runat="server" ID="PXGridAnswers" Height="200px" SkinID="Inquire"
                Width="100%" MatrixMode="True" DataSourceID="ds">
        <AutoSize Enabled="True" MinHeight="200" />
        <ActionBar>
            <Actions>
                <Search Enabled="False" />
            </Actions>
        </ActionBar>
        <Mode AllowAddNew="False" AllowDelete="False" AllowColMoving="False" />
        <Levels>
            <px:PXGridLevel DataMember="Answers">
                <Columns>
                    <px:PXGridColumn TextAlign="Left" DataField="AttributeID"
TextField="AttributeID_description"
                                                Width="250px" AllowShowHide="False" />
                    <px:PXGridColumn Type="CheckBox" TextAlign="Center" DataField="isRequired"
Width="80px" />
                    <px:PXGridColumn DataField="Value" Width="300px" AllowSort="False"
AllowShowHide="False" />
                </Columns>
            </px:PXGridLevel>
        </Levels>
    </px:PXGrid>
  </Template>
</px:PXTabItem>
```

Laden Sie das Bereitstellungspaket herunter

Hinzufügen einer Attributunterstützung zu einer Out-of-Box-Bestellungseinheit online lesen: https://riptutorial.com/de/acumatica/topic/8666/hinzufugen-einer-attributunterstutzung-zu-einer-out-of-box-bestellungseinheit

Kapitel 23: Liste der Entitäten erweitern, die durch Aufgaben, Ereignisse und Aktivitäten unterstützt werden

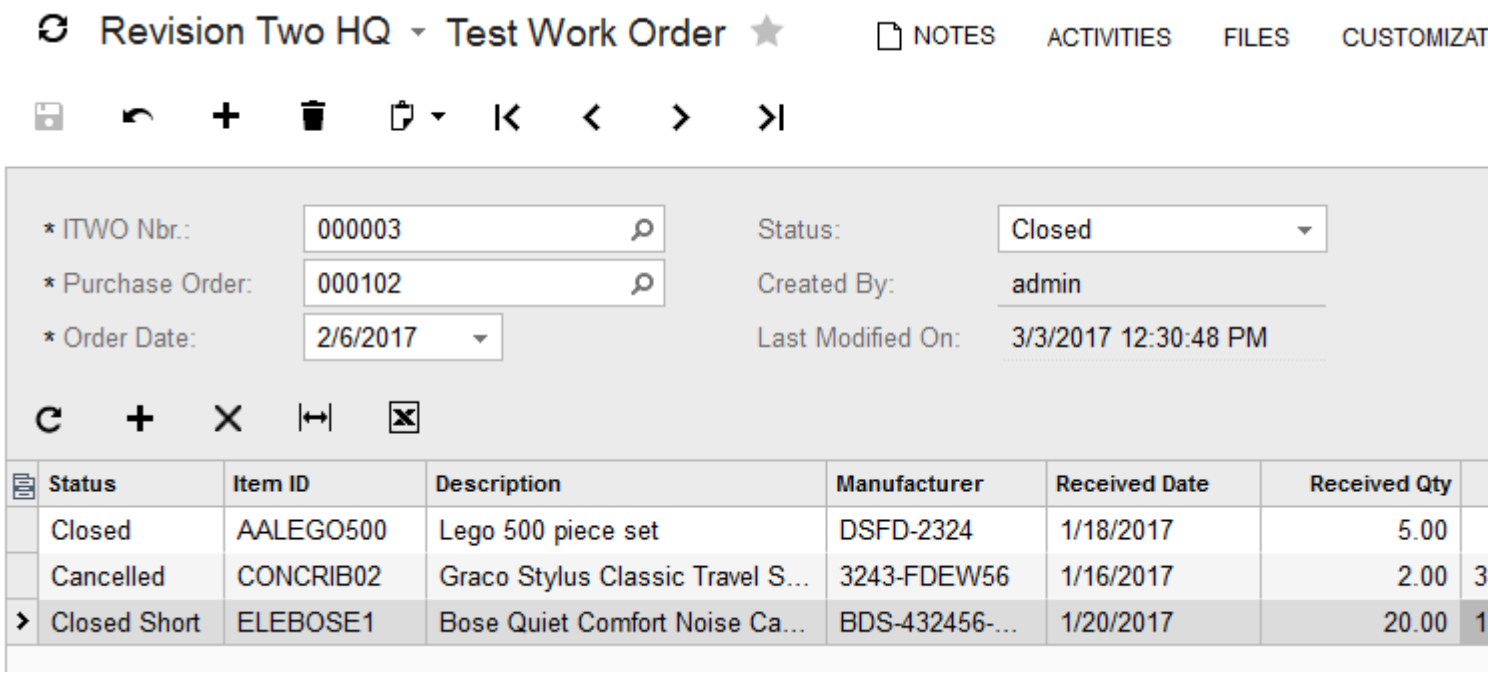
Einführung

In diesem Thema erfahren Sie, wie Sie das Feld Beschreibung der zugehörigen Entität um eine benutzerdefinierte Entität für Aufgaben, Ereignisse und Aktivitäten erweitern.

Examples

Hinzufügen von Testarbeitsaufträgen zum Feld Beschreibung der zugehörigen Entität

Angenommen, Sie haben bereits den benutzerdefinierten Bildschirm " **Test-Arbeitsaufträge" erstellt** , um Test-Arbeitsaufträge in Ihrer Acumatica ERP-Anwendung zu verwalten:



Es gibt bereits `NoteID` Feld in der erklärt `TestWorkOrder` DAC, auf dem **Test** - **Arbeitsaufträge** Bildschirm verwaltet:

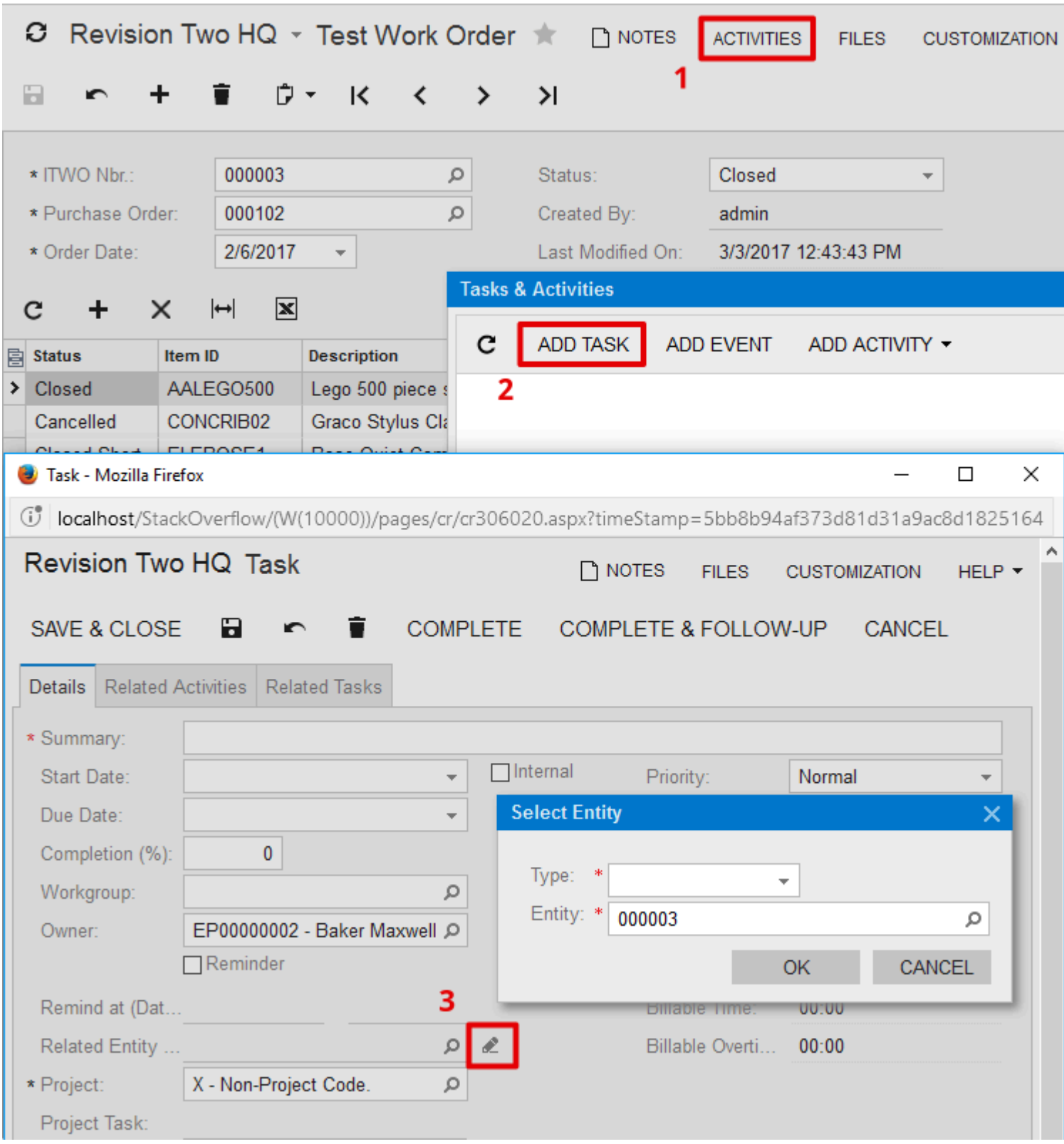
```
[Serializable]
public class TestWorkOrder : IBqlTable
{
    ...

    #region NoteID
    public abstract class noteID : IBqlField { }
```

```
    [PXNote]
    public virtual Guid? NoteID { get; set; }
    #endregion


    ...
}
```

Die `ActivityIndicator` Eigenschaft wird für den `PXForm` Container der obersten Ebene auf " ***True"*** `PXForm` :

```
<px:PXFormView ID="form" runat="server" ActivityIndicator="true" DataSourceID="ds" Style="z-
index: 100" DataMember="ITWO" Width="100%" >
```

Wenn jedoch eine neue Aufgabe, ein Ereignis oder eine neue Aktivität für einen Testarbeitsauftrag erstellt wird, ist das Steuerelement für die **zugehörige Entitätbeschreibung** immer leer:



Führen Sie die folgenden Schritte aus, um die Entität **Test Work Order** zum Selector **Related Entity Description** hinzuzufügen:

1. `PXNoteAttribute` für **das** Feld `PXNoteAttribute` on **TestWorkOrder.NoteID** die Eigenschaft `ShowInReferenceSelector` auf ***True,*** und definieren Sie den BQL-Ausdruck, um die in der **Entity-** Suche angezeigten Datensätze auszuwählen:

```
[PXNote(
```

```
    ShowInReferenceSelector = true,
    Selector = typeof(Search<TestWorkOrder.orderNbr>))]
public virtual Guid? NoteID { get; set; }
```

2. Dekorieren Sie den `TestWorkOrder` DAC mit dem `PXCacheNameAttribute` und dem `PXPrimaryGraphAttribute` :

```
[PXLocalizable]
public static class Messages
{
    public const string Opportunity = "Test Work Order";
}

[Serializable]
[PXCacheName(Messages.Opportunity)]
[PXPrimaryGraph(typeof(TestWorkOrderEntry))]
public class TestWorkOrder : IBqlTable
{
    ...
}
```

   Das `PXCacheName` Attribut definiert den benutzerfreundlichen Namen für den `TestWorkOrder` DAC (***Test Work Order*** in diesem Fall), der in der Dropdown-Liste **Typ** verfügbar ist. Das `PXPrimaryGraph` Attribut bestimmt die Eingabeseite , wo ein Benutzer zum Bearbeiten eines Testarbeitsauftrages umgeleitet wird, die der **Test Arbeitsaufträge** Bildschirm in dem gegebenen Beispiel ist.

3. Dekorieren Sie einige `TestWorkOrder` Felder mit dem `PXFieldDescriptionAttribute` . Diese Feldwerte werden zu einer einzelnen Textbeschriftung verkettet, die die referenzierte Testarbeitsreihenfolge im Feld **Beschreibung** der **zugehörigen Entität darstellt** :

```
...
[PXFieldDescription]
public virtual string OrderNbr { get; set; }

...
[PXFieldDescription]
public virtual String Status { get; set; }

...
[PXFieldDescription]
public virtual string POOrderNbr { get; set; }
```

4. Definieren Sie die Liste der Spalten, die in der **Entitätssuche** angezeigt werden, indem Sie eine der folgenden Methoden auswählen:

   ein. Verwenden Sie die `PXNoteAttribute.FieldList` Eigenschaft (erhält die höchste Priorität):

```
public abstract class noteID : IBqlField { }
[PXNote(
    ShowInReferenceSelector = true,
    Selector = typeof(Search<TestWorkOrder.orderNbr>),
    FieldList = new Type[]
```

```
    {
        typeof(TestWorkOrder.orderNbr),
        typeof(TestWorkOrder.orderDate),
        typeof(TestWorkOrder.status),
        typeof(TestWorkOrder.poOrderNbr)
    })]
public virtual Guid? NoteID { get; set; }
```

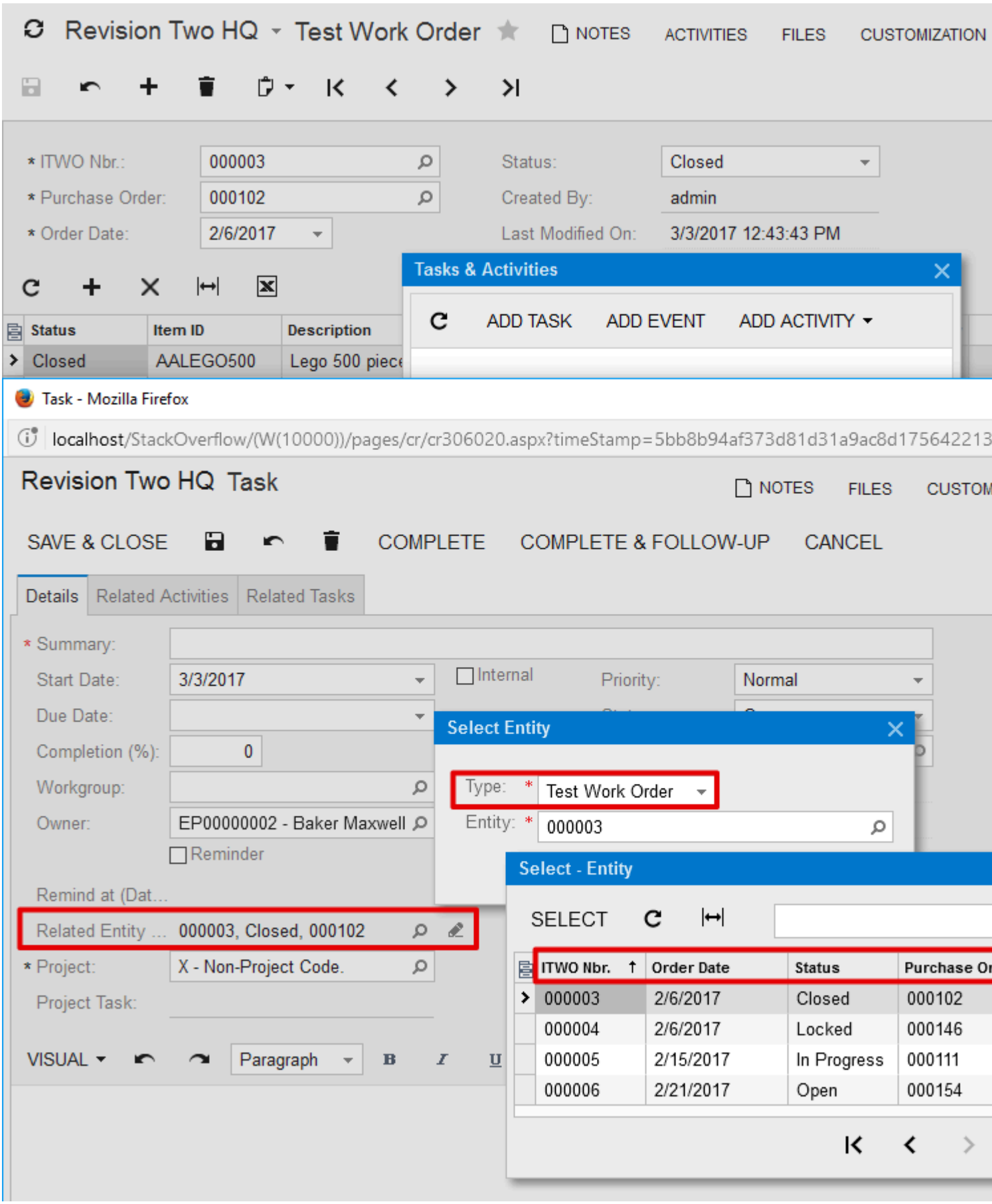
b. Leihen Sie sich die Liste der für die **OrderNbr-** Suche definierten **Spalten aus** :

```
public abstract class orderNbr : IBqlField { }
[PXDBString(15, IsKey = true, IsUnicode = true, InputMask = ">CCCCCCCCCCCCCCC")]
[PXDefault()]
[PXUIField(DisplayName = "ITWO Nbr.", Visibility = PXUIVisibility.SelectorVisible)]
[PXSelector(typeof(Search<TestWorkOrder.orderNbr>),
    typeof(TestWorkOrder.orderNbr),
    typeof(TestWorkOrder.orderDate),
    typeof(TestWorkOrder.status),
    typeof(TestWorkOrder.poOrderNbr))]
[PXFieldDescription]
public virtual string OrderNbr { get; set; }
```

c. Alle `TestWorkOrder` Felder `TestWorkOrder` , deren **Sichtbarkeit** auf `PXUIVisibility.SelectorVisible` :

```
...
[PXUIField(DisplayName = "ITWO Nbr.", Visibility = PXUIVisibility.SelectorVisible)]
public virtual string OrderNbr { get; set; }

...
[PXUIField(DisplayName = "Order Date", Visibility = PXUIVisibility.SelectorVisible)]
public virtual DateTime? OrderDate { get; set; }

...
[PXUIField(DisplayName = "Status", Visibility = PXUIVisibility.SelectorVisible)]
public virtual String Status { get; set; }

...
[PXUIField(DisplayName = "Purchase Order", Visibility = PXUIVisibility.SelectorVisible)]
public virtual string POOrderNbr { get; set; }
```

Nachdem Sie die oben genannten 4 Schritte ausgeführt haben, sollte **Test Work Orders** vollständig durch das Feld **Enthält Beschreibung** der **Entität** für Aufgaben, Ereignisse und Aktivitäten unterstützt werden



Liste der Entitäten erweitern, die durch Aufgaben, Ereignisse und Aktivitäten unterstützt werden online lesen: https://riptutorial.com/de/acumatica/topic/9342/liste-der-entitaten-erweitern--die-durch-aufgaben--ereignisse-und-aktivitaten-unterstutzt-werden

Kapitel 24: Verwenden des Anpassungs-Plug-Ins für Änderungen in mehreren Unternehmen

Einführung

Mit von **CustomizationPlug** abgeleiteten Klassen können Sie die Funktionen der Acumatica Customization Platform nutzen und nach Veröffentlichung des Anpassungsprojekts benutzerdefinierten Code ausführen. In diesem Thema erfahren Sie, wie Anpassungs-Plug-Ins für Änderungen in mehreren Unternehmen verwendet werden können.

Weitere Informationen zu Anpassungs-Plug-Ins finden Sie im Acumatica Customization Guide

Examples

Implementierung eines Anpassungs-Plug-Ins zum Aktualisieren mehrerer Unternehmen

Um ein Anpassungs-Plug-in zu erstellen, erstellen Sie einfach eine von **CustomizationPlug** abgeleitete Klasse und packen sie in die Anpassung. Während das System ein Anpassungsprojekt veröffentlicht, führt es die in Ihrem Anpassungs- **Plug-In** implementierten **OnPublished-** und **UpdateDatabase-** Methoden ***nur im aktuellen Unternehmensbereich aus*** .

Das heißt, das Anpassungs-Plug-In nimmt niemals Änderungen an einem anderen als dem aktuellen Unternehmen vor, es sei denn, es verwendet **PXLoginScope,** um sich **nacheinander** bei allen Unternehmen anzumelden, die der aktuellen Benutzerveröffentlichungsanpassung zur Verfügung stehen.

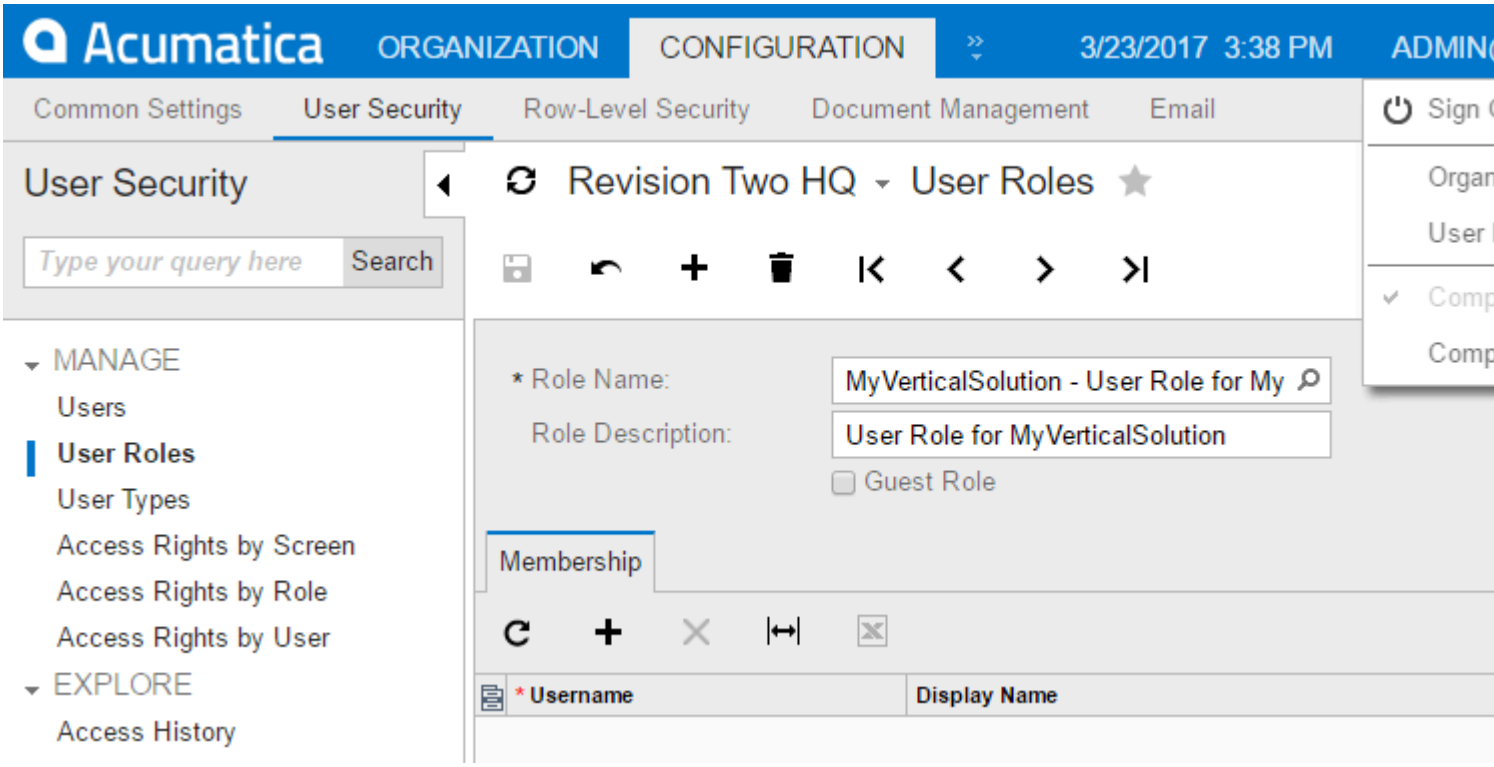
Im Folgenden finden Sie ein Beispiel für ein Anpassungs-Plugin, ***das die*** Benutzerrolle " ***MyVerticalSolution"*** in allen Unternehmen erstellt, die dem aktuellen Benutzer zur Verfügung stehen:

```
public class MyVerticalSolutionInit : CustomizationPlugin
{
    public override void UpdateDatabase()
    {
        var companies = PXAccess.GetCompanies();

        foreach (var company in companies)
        {
            using (var loginScope = new PXLoginScope(string.Format("{0}@{1}",
                PXAccess.GetUserLogin(), company)))
            {
                string roleName = "MyVerticalSolution";
                RoleAccess graph = PXGraph.CreateInstance<RoleAccess>();
```

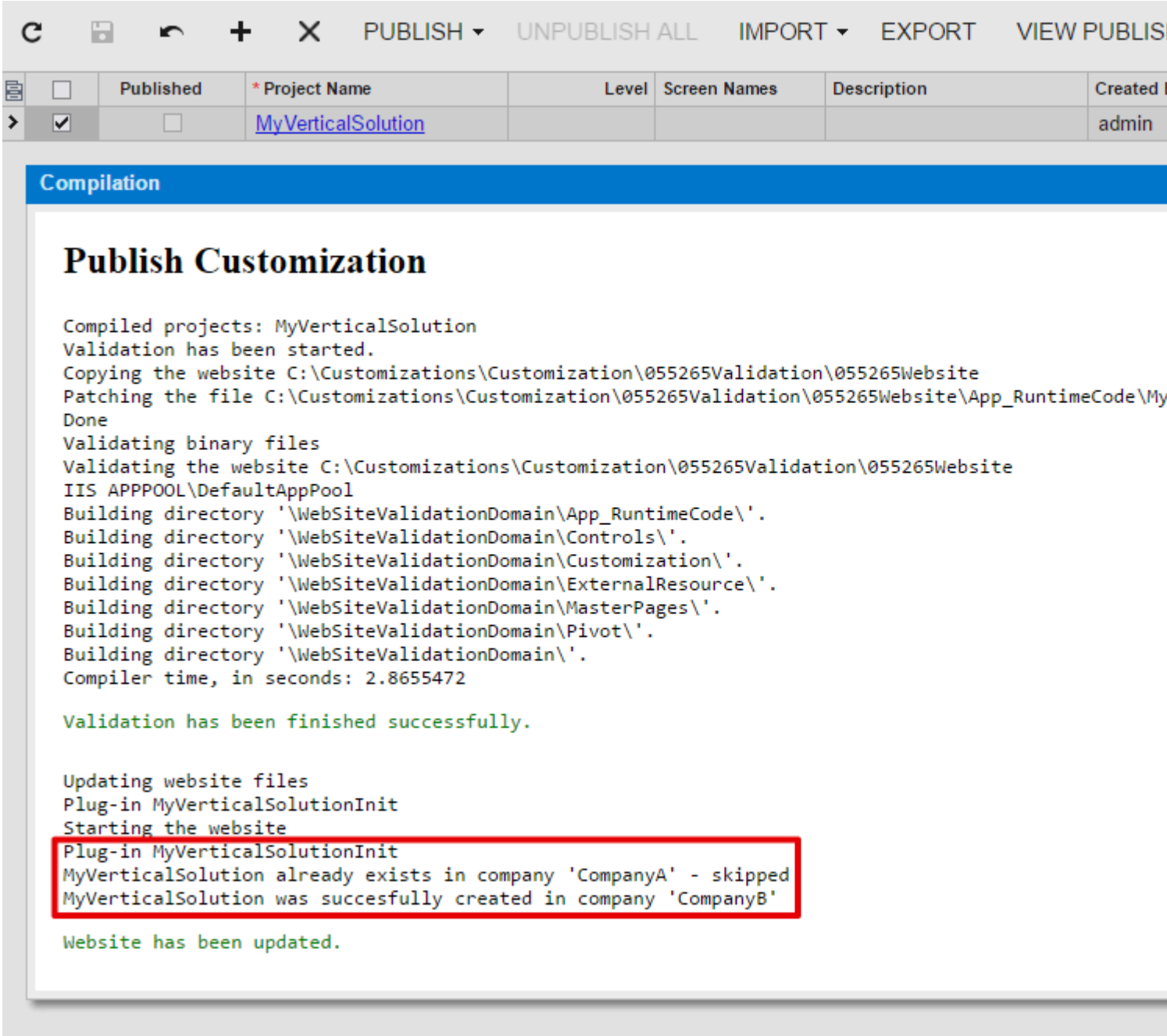
```
                Roles existingRole = graph.Roles.Search<Roles.rolename>(roleName);
                if (existingRole != null)
                {
                    WriteLog(string.Format("{0} already exists in company '{1}' - skipped",
roleName, company));
                    continue;
                }

                var wmsRole = new Roles();
                wmsRole.Rolename = roleName;
                wmsRole.Descr = "User Role for MyVerticalSolution";

                graph.Roles.Insert(wmsRole);
                graph.Save.Press();

                WriteLog(string.Format("{0} was succesfully created in company '{1}'",
roleName, company));
            }
        }
    }
}
```

Um eine Liste der Unternehmen zu erhalten, die dem aktuellen Benutzer zur Verfügung stehen, rufen Sie einfach die statische `PXAccess.GetCompanies()` Methode auf. Anschließend wird **PXLoginScope** verwendet, um sich bei jedem der verfügbaren Unternehmen ***anzumelden*** , um ***die*** Benutzerrolle ***MyVerticalSolution*** zu erstellen. Hinweis-Instanz des **RoleAccess-** BLC für jedes Unternehmen neu initialisiert - dies ist ein zwingender Schritt, um Änderungen an mehreren Unternehmen gleichzeitig vorzunehmen.

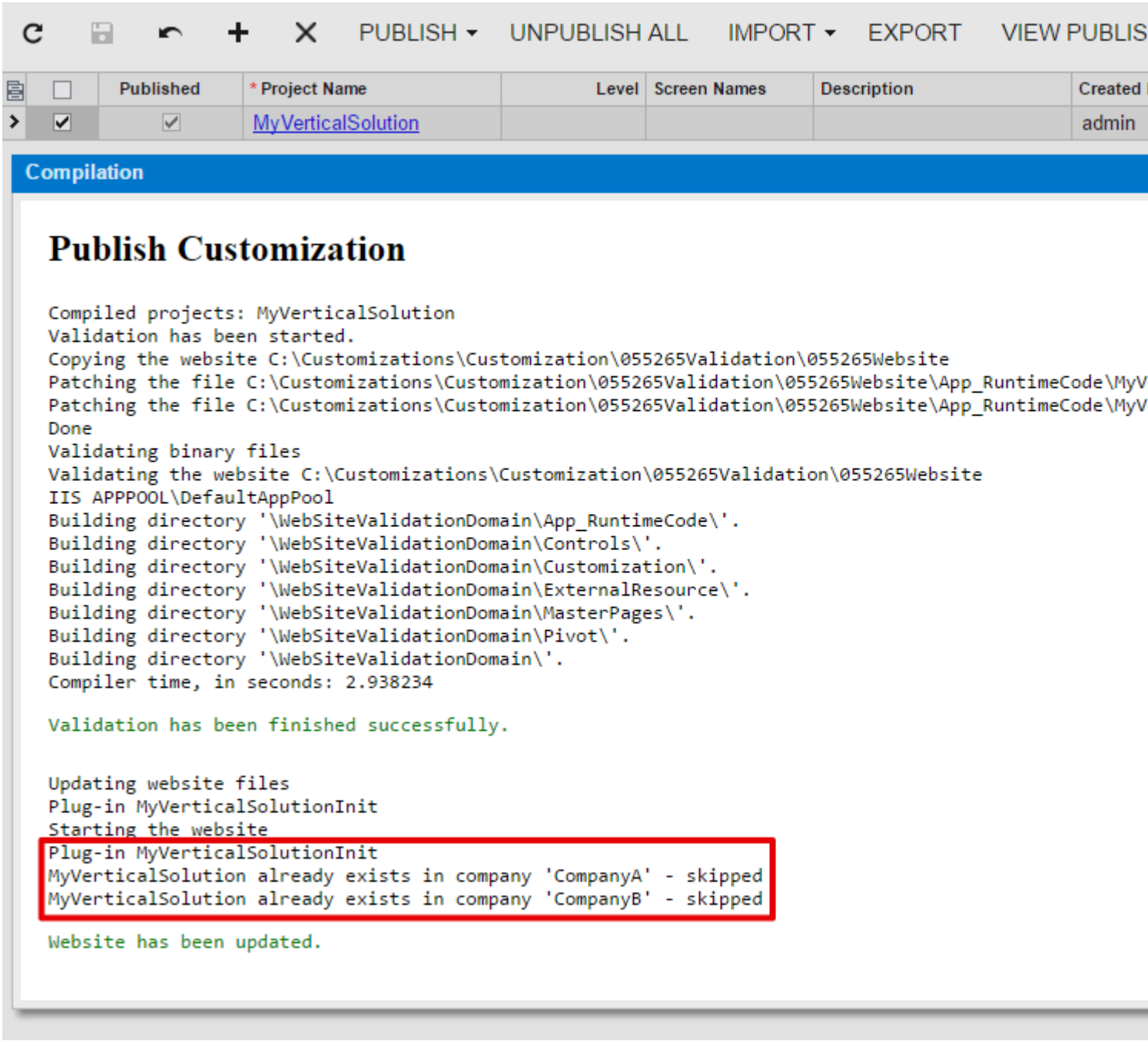
Nehmen wir an, es gibt zwei Unternehmen in Ihrer Acumatica-Instanz: CompanyA und CompanyB. Der **Administratorbenutzer** , den Sie für die Veröffentlichung der Anpassung verwenden werden, hat Zugriff auf beide Unternehmen und die ***MyVerticalSolution-*** Rolle, die durch das Anpassungs-Plug-in erstellt wurde, ist bereits in CompanyA vorhanden:



Nachdem Sie Anpassungen veröffentlicht haben (während Sie bei CompanyA oder CompanyB angemeldet sind), und zwar mit einem zuvor entwickelten Anpassungs-Plug-in zum Erstellen der ***MyVerticalSolution-*** Rolle in allen dem aktuellen Benutzer verfügbaren Unternehmen, beachten ***Sie*** , dass die ***MyVerticalSolution-*** Rolle für CompanyA übersprungen und erfolgreich für CompanyB erstellt wurde.



Wenn Sie diese Anpassung das nächste Mal veröffentlichen, wird die ***MyVerticalSolution-*** Rolle für beide Unternehmen in Ihrer Acumatica-Anwendung übersprungen:



Verwenden des Anpassungs-Plug-Ins für Änderungen in mehreren Unternehmen online lesen: https://riptutorial.com/de/acumatica/topic/9522/verwenden-des-anpassungs-plug-ins-fur-anderungen-in-mehreren-unternehmen

Kapitel 25: Wichtige API-Änderungen zwischen den Versionen

Examples

PXSelectGroupBy und Bitwerte in Acumatica 5.1 und 5.2+

Die Methode der SQL-Generierung aus BQL- `PXSelectGroupBy<>` -Sichten wurde in Acumatica Framework 5.2 geändert.

In den folgenden Abschnitten werden die Unterschiede am Beispiel von `PXSelectGroupBy<FinYear, Aggregate<GroupBy<FinYear.finPeriods>>>.Select(graph)` :

Acumatica Framework 5.2 und höher

```
SELECT Max([finyear].[year]),
       Max([finyear].[startdate]),
       Max([finyear].[enddate]),
       [finyear].[finperiods],
       -- Attention!
       CONVERT (BIT, Max([finyear].[customperiods] + 0)),
       --
       Max([finyear].[begfinyearhist]),
       Max([finyear].[periodsstartdatehist]),
       Max([finyear].[noteid]),
       ( NULL ),
       ( NULL ),
       ( NULL ),
       Max([finyear].[tstamp]),
       Max([finyear].[createdbyid]),
       Max([finyear].[createdbyscreenid]),
       Max([finyear].[createddatetime]),
       Max([finyear].[lastmodifiedbyid]),
       Max([finyear].[lastmodifiedbyscreenid]),
       Max([finyear].[lastmodifieddatetime])
FROM   finyear FinYear
WHERE  ( finyear.companyid = 2 )
GROUP  BY [finyear].[finperiods]
ORDER  BY Max([finyear].[year])
```

Acumatica Framework 5.1 und früher

```
SELECT Max([finyear].[year]),
       Max([finyear].[startdate]),
       Max([finyear].[enddate]),
       [finyear].[finperiods],
       -- Attention!
       ( NULL ),
```

```
       --
       Max([finyear].[begfinyearhist]),
       Max([finyear].[periodsstartdatehist]),
       ( NULL ),
       ( NULL ),
       ( NULL ),
       Max([finyear].[tstamp]),
       ( NULL ),
       Max([finyear].[createdbyscreenid]),
       Max([finyear].[createddatetime]),
       ( NULL ),
       Max([finyear].[lastmodifiedbyscreenid]),
       Max([finyear].[lastmodifieddatetime])
FROM   finyear FinYear
WHERE  ( finyear.companyid = 2 )
GROUP  BY [finyear].[finperiods]
ORDER  BY Max([finyear].[year])
```

Erläuterung

Standardmäßig wird das Aggregat `Max()` auf alle Felder angewendet, die nicht explizit in einer BQL-Anweisung angegeben sind.

Doch in Acumatica 5.1 und früher, schließt sie die `CreatedByID` , `LastModifiedByID` und `bool` Felder aus . Bei der Übersetzung in SQL sind diese Felder immer `null` wenn Sie nicht explizit nach gruppiert werden.

Ab Version 5.2 wird auch `Max()` standardmäßig für sie angewendet.

Wichtige API-Änderungen zwischen den Versionen online lesen:
https://riptutorial.com/de/acumatica/topic/9697/wichtige-api-anderungen-zwischen-den-versionen

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit acumatica	Community
2	Acumatica BQL-Referenz	wh1t3cat1k
3	Acumatica-Plattformattribute	wh1t3cat1k
4	Ändern der Beschriftung mithilfe von Readonly-DAC-Feldern.	cbetabeta, Simon ML
5	Änderungen an Basisdatenansichten	RuslanDev
6	Änderungen an Kontakt- und Adressinformationen durch Code	RuslanDev
7	Anpassungsmechanismen	wh1t3cat1k
8	Anzeigen eines Fehlers, der die Eingabe von Entitätsdaten erfordert	wh1t3cat1k
9	Bedingt Tabs ausblenden	RuslanDev
10	Bei der Veröffentlichung wurde der bereits angewendete Anpassungsinhalt nicht berücksichtigt	samol518
11	Benutzeroberflächen-Techniken	wh1t3cat1k
12	Bericht mit Daten durch Code füllen	Gabriel, RuslanDev
13	Bilder auf der Anmeldeseite ersetzen	RuslanDev
14	Datensätze über REST-	RuslanDev

	vertragsbasierte API exportieren	
15	Datensätze über screen-based API exportieren	RuslanDev
16	Datums- und Zeitfelder in Acumatica erstellen	RuslanDev
17	Elemente in einer Dropdown-Liste ändern	RuslanDev
18	Filtern mit mehreren Werten mit nur einem Selektor	samol518
19	Frachtberechnung	RuslanDev
20	Größe des Auswahlfensters ändern	Gabriel, RuslanDev
21	Herunterladen von Dateien, die an eine Detailentität angehängt sind, mithilfe einer vertragsbasierten API	RuslanDev
22	Hinzufügen einer Attributunterstützung zu einer Out-of-Box-Bestellungseinheit	DChhapgar
23	Liste der Entitäten erweitern, die durch Aufgaben, Ereignisse und Aktivitäten unterstützt werden	RuslanDev
24	Verwenden des Anpassungs-Plug-Ins für Änderungen in mehreren Unternehmen	RuslanDev
25	Wichtige API-Änderungen zwischen den Versionen	wh1t3cat1k