Acumatica
The Cloud ERP

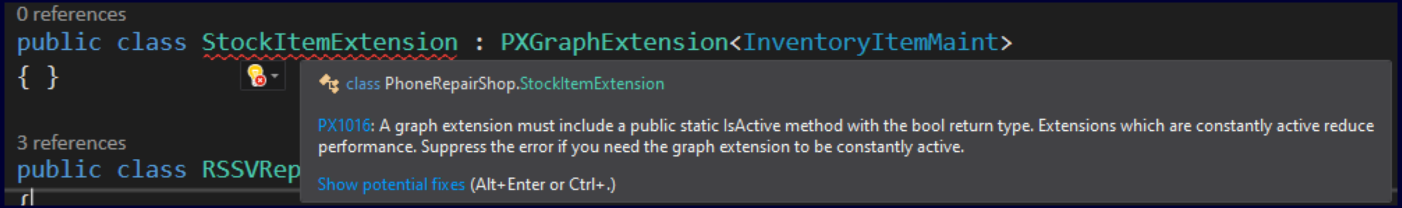
T210 Customized Forms and Master-Detail Relationship

Dmitii Naumov

Solutions Architect

Possible Issues

- Ignore Acuminator error PX1016



```
0 references
public class StockItemExtension : PXGraphExtension<InventoryItemMaint>
{ }

3 references
public class RSSVRep
{|
```

class PhoneRepairShop.StockItemExtension

PX1016: A graph extension must include a public static IsActive method with the bool return type. Extensions which are constantly active reduce performance. Suppress the error if you need the graph extension to be constantly active.

Show potential fixes (Alt+Enter or Ctrl+.)

- OpenSolution.bat does not work in AWS virtual machines that you were provided. Please use the following path to open the solution:

C:\PhoneRepairShop\App_Data\Projects\PhoneRepairShop_Code

Timing and Agenda

Aug 17, 2020 – 10 AM-12 PM PST

Day 1

- **Part 1 – Custom Fields (Stock Items Form)**

Aug 18, 2020 – 10 AM-12 PM

Day 2

- **Part 2 – Master-Detail Relationship and Business Logic (Services and Prices Form)**

Aug 19, 2020 – 10 AM-12 PM

Day 3

- **Part 3 – Custom Tab (Stock Items Form)**

- **Part 4 – Calculations and Insertion of a Default Record (Services and Prices Form)**

T210: Part 1 – Custom Fields (Stock Items Form)

Company Story and Customization Description

- Cell Phone Repair Shop
- Development Started in T200 Training
  - Repair Services Form
  - Serviced Devices Form
- During T210 Training
  - Add Services and Prices Form
  - Customize Stock Items Form

1.1: Adding Custom Fields

Objectives:

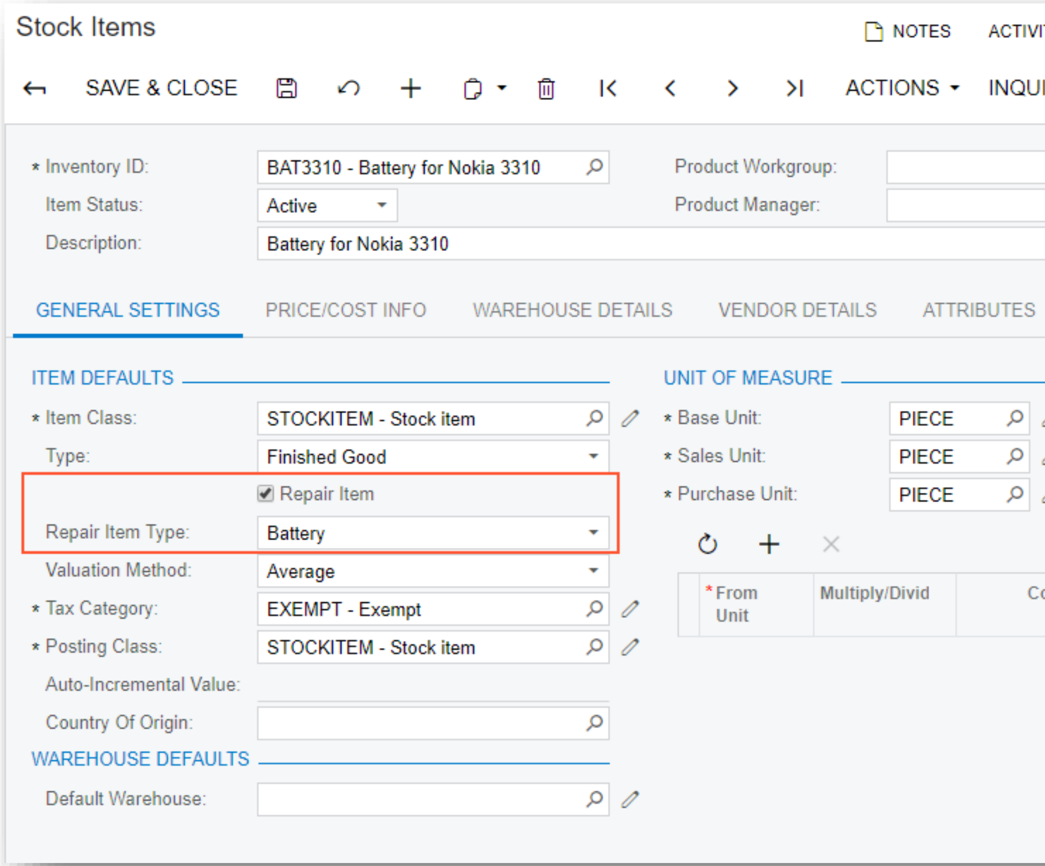
- Add a custom column to an Acumatica ERP database table
- Add a custom field to an Acumatica ERP data access class
- Add the control for the custom field to the form

Acumatica The Cloud ERP

Stock Items Form

- The **Repair Item** check box will be used to define whether the selected stock item is a repair item.
- The **Repair Item Type** box will hold the repair item type to which the repair item belongs

  Battery, *Screen*, *Screen Cover*, *Back Cover*, or *Motherboard*.



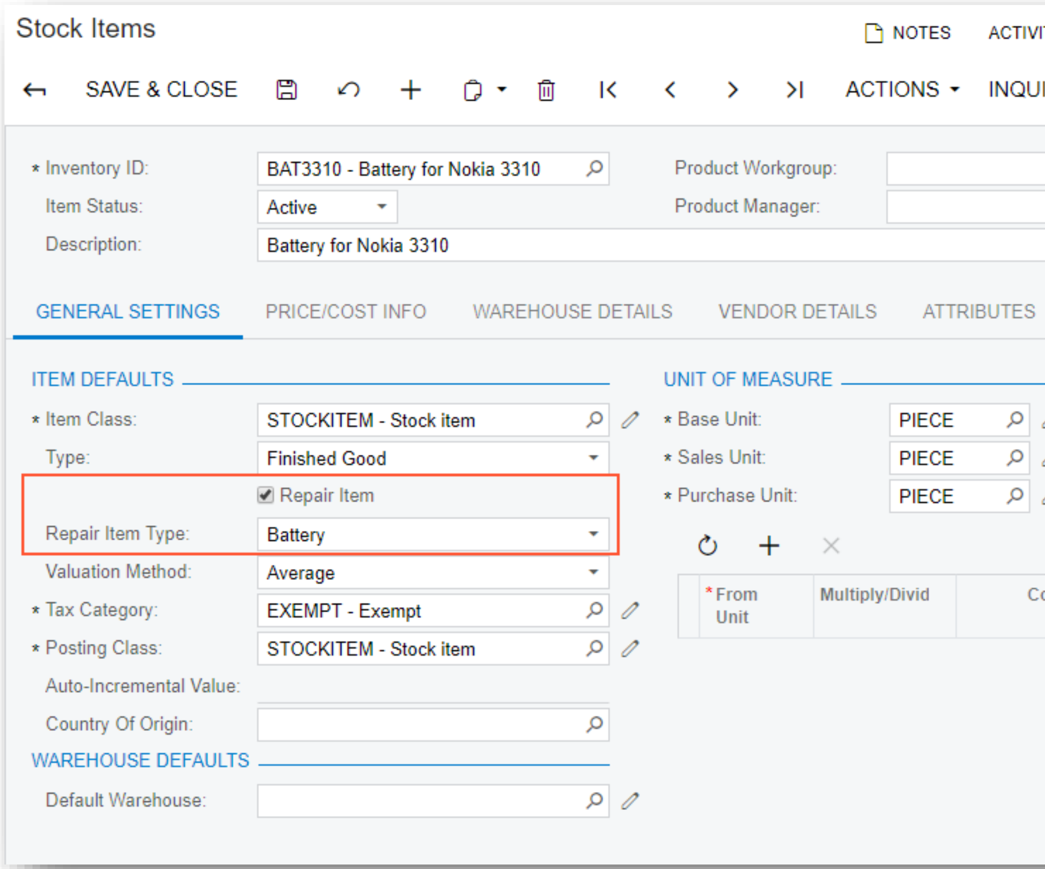
* Inventory ID:	BAT3310 - Battery for Nokia 3310
Item Status:	Active
Description:	Battery for Nokia 3310
Product Workgroup:	
Product Manager:	
* Item Class:	STOCKITEM - Stock item
Type:	Finished Good
	☑ Repair Item
Repair Item Type:	Battery
Valuation Method:	Average
* Tax Category:	EXEMPT - Exempt
* Posting Class:	STOCKITEM - Stock item
Auto-Incremental Value:	
Country Of Origin:	
Default Warehouse:	
* Base Unit:	PIECE
* Sales Unit:	PIECE
* Purchase Unit:	PIECE

Creating New Repair Item Checkbox

1. Create new fields in the Database & Acumatica Data Access Layer
2. Move DAC extensions to Extension Library
3. Create new controls in UI

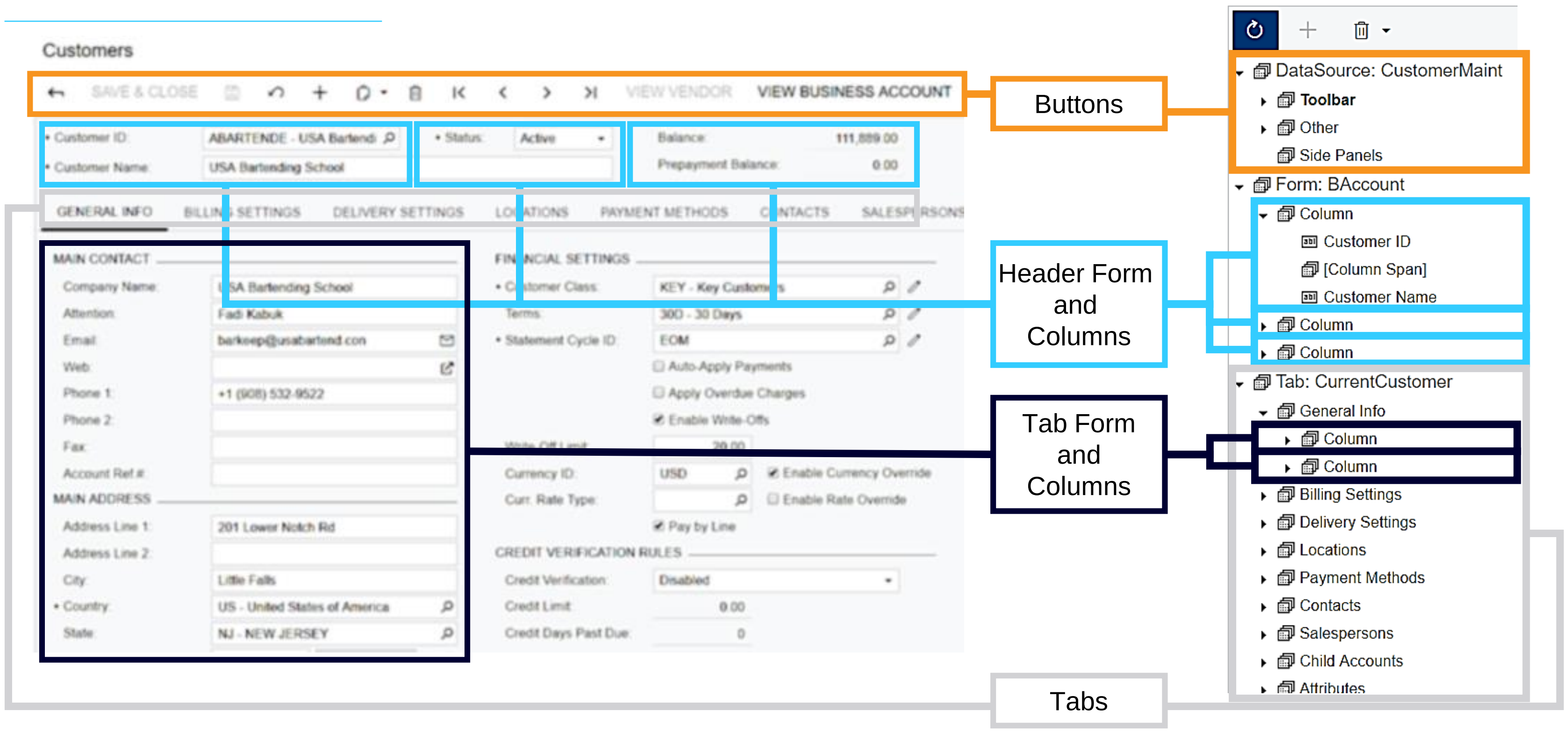
Stock Items Form

1. Click Customization -> Inspect Element
2. Click on the Form
3. Click Customize
4. Select PhoneRepairShop Customization Project -> Ok
5. Add new fields
6. Modify attributes
7. Publish project
8. Create controls
9. Verify result



* Inventory ID:	BAT3310 - Battery for Nokia 3310
Item Status:	Active
Description:	Battery for Nokia 3310
Product Workgroup:	
Product Manager:	
* Item Class:	STOCKITEM - Stock item
Type:	Finished Good
	✓ Repair Item
Repair Item Type:	Battery
Valuation Method:	Average
* Tax Category:	EXEMPT - Exempt
* Posting Class:	STOCKITEM - Stock item
Auto-Incremental Value:	
Country Of Origin:	
Default Warehouse:	
* Base Unit:	PIECE
* Sales Unit:	PIECE
* Purchase Unit:	PIECE

The Anatomy of Customization Project



Customers

SAVE & CLOSE | VIEW VENDOR | VIEW BUSINESS ACCOUNT

• Customer ID:	ABARTENDE - USA Bartend	• Status:	Active	Balance:	111,889.00
• Customer Name:	USA Bartending School			Prepayment Balance:	0.00

GENERAL INFO | BILLING SETTINGS | DELIVERY SETTINGS | LOCATIONS | PAYMENT METHODS | CONTACTS | SALESPERSONS

MAIN CONTACT

Company Name:	USA Bartending School
Attention:	Fadi Kabuk
Email:	barkeep@usabartend.con
Web:	
Phone 1:	+1 (908) 532-9522
Phone 2:	
Fax:	
Account Ref #:	

MAIN ADDRESS

Address Line 1:	201 Lower Notch Rd
Address Line 2:	
City:	Little Falls
• Country:	US - United States of America
State:	NJ - NEW JERSEY

FINANCIAL SETTINGS

• Customer Class:	KEY - Key Customers
Terms:	30D - 30 Days
• Statement Cycle ID:	EOM

☐ Auto-Apply Payments
☐ Apply Overdue Charges
☑ Enable Write-Offs

Write-Off Limit:	20.00	
Currency ID:	USD	☑ Enable Currency Override
Curr. Rate Type:		☐ Enable Rate Override

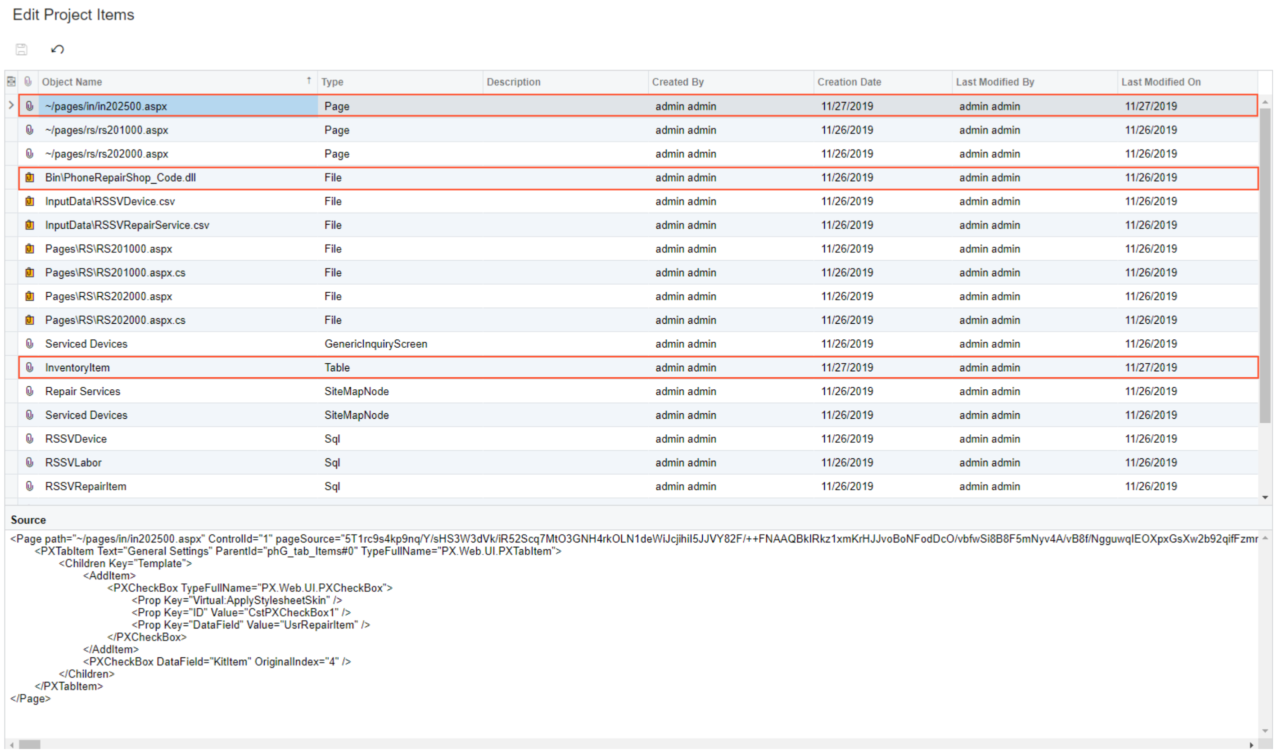
☑ Pay by Line

CREDIT VERIFICATION RULES

Credit Verification:	Disabled
Credit Limit:	0.00
Credit Days Past Due:	0

- DataSource: CustomerMaint
  - Toolbar
  - Other
  - Side Panels
- Form: BAccount
  - Column
    - Customer ID
    - [Column Span]
    - Customer Name
  - Column
  - Column
- Tab: CurrentCustomer
  - General Info
    - Column
    - Column
  - Billing Settings
  - Delivery Settings
  - Locations
  - Payment Methods
  - Contacts
  - Salespersons
  - Child Accounts
  - Attributes

Acumatica Extension Library vs. Acumatica Customization Project



Edit Project Items

Object Name	Type	Description	Created By	Creation Date	Last Modified By	Last Modified On
~/pages/in/in202500.aspx	Page		admin admin	11/27/2019	admin admin	11/27/2019
~/pages/rs/rs201000.aspx	Page		admin admin	11/26/2019	admin admin	11/26/2019
~/pages/rs/rs202000.aspx	Page		admin admin	11/26/2019	admin admin	11/26/2019
Bin\PhoneRepairShop_Code.dll	File		admin admin	11/26/2019	admin admin	11/26/2019
InputData\RSSVDevice.csv	File		admin admin	11/26/2019	admin admin	11/26/2019
InputData\RSSVRepairService.csv	File		admin admin	11/26/2019	admin admin	11/26/2019
Pages\RS\RS201000.aspx	File		admin admin	11/26/2019	admin admin	11/26/2019
Pages\RS\RS201000.aspx.cs	File		admin admin	11/26/2019	admin admin	11/26/2019
Pages\RS\RS202000.aspx	File		admin admin	11/26/2019	admin admin	11/26/2019
Pages\RS\RS202000.aspx.cs	File		admin admin	11/26/2019	admin admin	11/26/2019
Serviced Devices	GenericInquiryScreen		admin admin	11/26/2019	admin admin	11/26/2019
InventoryItem	Table		admin admin	11/27/2019	admin admin	11/27/2019
Repair Services	SiteMapNode		admin admin	11/26/2019	admin admin	11/26/2019
Serviced Devices	SiteMapNode		admin admin	11/26/2019	admin admin	11/26/2019
RSSVDevice	Sql		admin admin	11/26/2019	admin admin	11/26/2019
RSSVLabor	Sql		admin admin	11/26/2019	admin admin	11/26/2019
RSSVRepairItem	Sql		admin admin	11/26/2019	admin admin	11/26/2019

Source

```
<Page path="~/pages/in/in202500.aspx" ControlId="1" pageSource="5T1rc9s4kp9nq/Y/sHS3W3dVk/iR52Scq7MtO3GNH4rkOLN1deWiJcjihil5JJVY82F/++FNAAQBkIRkz1xmKrHJJvoBoNFodDcO/vbfwSi8B8F5mNyv4A/vB8f/NgguwqIEOXpxGsXw2b92qifFzmr
    <PXTabItem Text="General Settings" ParentId="phG_tab_Items#0" TypeFullName="PX.Web.UI.PXTabItem">
        <Children Key="Template">
            <AddItem>
                <PXCheckBox TypeFullName="PX.Web.UI.PXCheckBox">
                    <Prop Key="Virtual:ApplyStylesheetSkin" />
                    <Prop Key="ID" Value="CstPXCheckBox1" />
                    <Prop Key="DataField" Value="UsrRepairItem" />
                </PXCheckBox>
            </AddItem>
            <PXCheckBox DataField="KitItem" OriginalIndex="4" />
        </Children>
    </PXTabItem>
</Page>
```

Acumatica Extensibility Framework

```
public class DACExtension : PXCacheExtension<BaseDAC>
{
       //Put new fields definition here
       //Customize existing attributes and fields
}
public class BLCExtension : PXGraphExtension<BaseBLC>
{
       //Put new event handlers, actions, data views or methods here
       //Customize existing logic with defining new one with the same name
}
```

Why Extensibility Framework

No difference at all when you declare Event, DataView or Action :

1. In Graph or Cache Extension
2. In regular BLC or DAC class

- Separate Acumatica code and Customization code
- Allow debug for System and Customized code separately
- Allow customization development to the Visual Studio
- Allows source control usage

Acumatica Extensibility Framework

- Multiple projects that extend the same DAC or BLC
- Advanced level of control over the business logic
- Multilevel extension model
- Straightforward deployment and upgrade process
- Extensions are precompiled
    - A measure of protection for your source code
    - Creates your own intellectual property

T210: Part 1.1 – Q&A

1.2: Adding Business Logic to Control the Fields Behavior

Objectives:

- Make availability of a field depend on value of another field
- Make visibility of a field depend on value of another field
- Make value available in a dropdown depend on value of another field

Creating New Repair Item Type Dropdown

1. Create new fields in the Database
2. Create new fields in the Acumatica Data Access Layer in Extension Library
3. Create new controls in UI

Configuring Drop-Downs

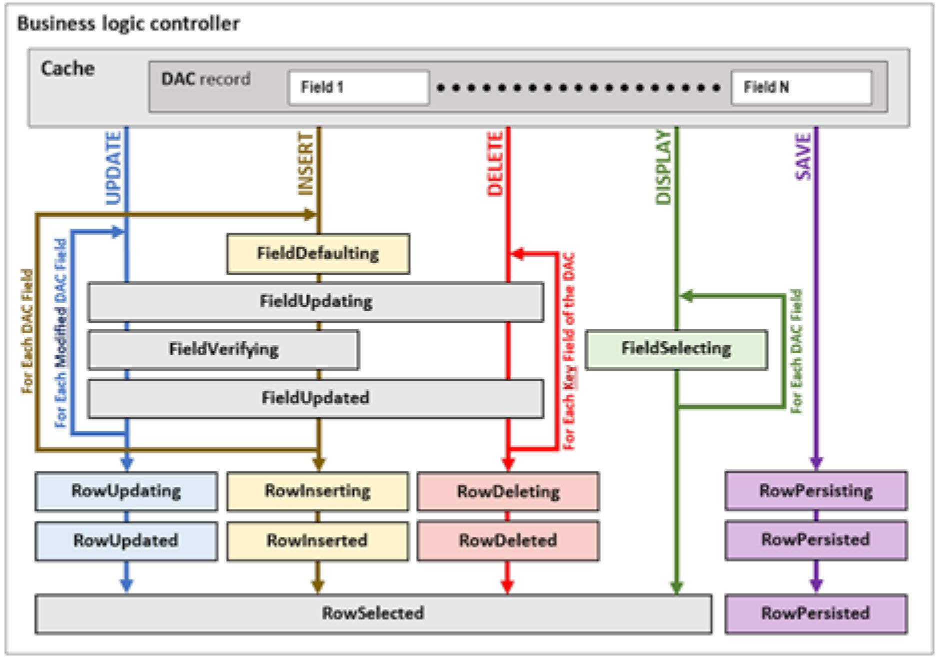
What can you do to the drop-downs?

- Chose between `int` and `string` values
- Set the list of values that the *system* will see
- Set the list of values that a *user* will see

You will likely use these values elsewhere, so store them in some constants!

```
[PXStringList(
    new string[]
    {
        Status.OnHold,      // "H"
        Status.Shipping,    // "S"
        Status.Delivered    // "D"
    },
    new string[]
    {
        "Hold",
        "In progress",
        "Done"
    })]
public string Status { get; set; }
```

Event Model


Business logic controller
Cache
DAC record
Field 1
Field N
UPDATE
INSERT
DELETE
DISPLAY
SAVE
For Each DAC Field
For Each Modified DAC Field
For Each Key Field of the DAC
For Each DAC Field
FieldDefaulting
FieldUpdating
FieldVerifying
FieldUpdated
FieldSelecting
RowUpdating
RowUpdated
RowInserting
RowInserted
RowDeleting
RowDeleted
RowPersisting
RowPersisted
RowSelected
RowPersisted

RowSelected event

`RowSelected` occurs each time a data record is displayed in the UI

+ when one sets the `Current` property of a `PXCache` object

This is the *best* place to configure the UI based on the values of data fields!

However, note that `RowSelected` is fired several times for a record during each round trip and it is the *worst* place to read data from the database

Adjusting UI Dynamically

You can show and hide fields based on various conditions:

```
PXUIFieldAttribute.SetEnabled<Shipment.deliveryDate>(cache, row,
        row.ShipmentType == ShipmentTypes.Single);
```

Similarly fields can be enabled/disabled:

```
PXUIFieldAttribute.SetVisible<Shipment.pendingQty>(cache, row,
        row.ShipmentType != ShipmentTypes.Single);
```

These things are better done in the `RowSelected` event handler.

Commit Changes

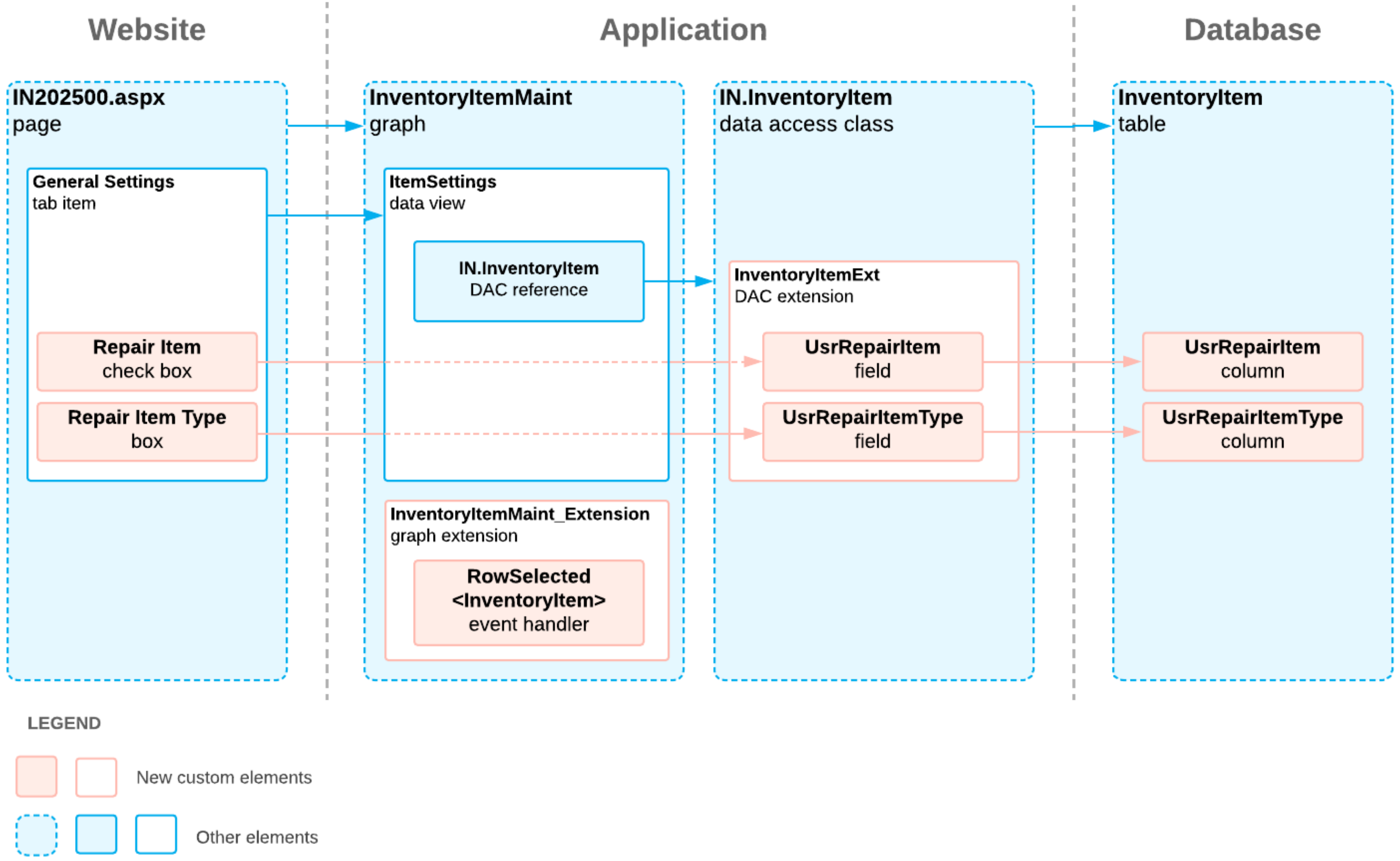
Normally server doesn’t get notified of every change to data made by user
You can make certain controls post changes to server once they occur:

```
<px:PXFormView ID="form" ... >
        <Template>
                <px:PXCheckBox ID="chkHold"
                               runat="server"
                               DataField="Hold"
                               CommitChanges="True" />
```

The same for grid columns:

```
<px:PXGridLevel ... >
        <RowTemplate>
                <px:PXGridColumn DataField="ProductID"
                                 CommitChanges="True" />
```

Summary



Website
Application
Database
IN202500.aspx page
General Settings tab item
Repair Item check box
Repair Item Type box
InventoryItemMaint graph
ItemSettings data view
IN.InventoryItem DAC reference
InventoryItemMaint_Extension graph extension
RowSelected <InventoryItem> event handler
IN.InventoryItem data access class
InventoryItemExt DAC extension
UsrRepairItem field
UsrRepairItemType field
InventoryItem table
UsrRepairItem column
UsrRepairItemType column
LEGEND
New custom elements
Other elements

T210: Part 1.2 – Q&A

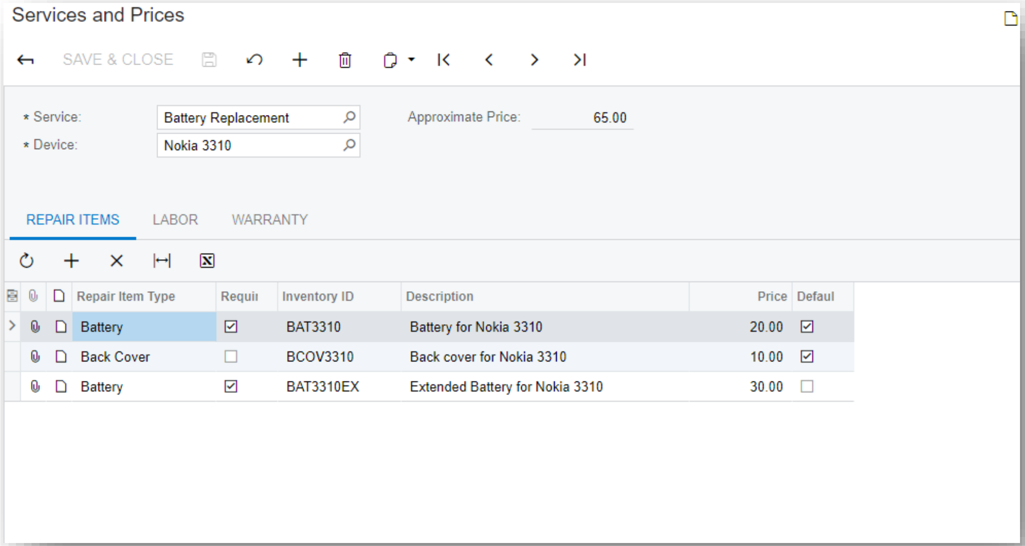
1.3: Configuring the Sitemap — Self-Guided Exercise

T210: The Services and Prices Form

The Services and Prices Form

The form will contain the following tabs:

- **Repair Items**: Will show the list of repair items (stock items) necessary for the repair service of the device.



* Service: Battery Replacement
* Device: Nokia 3310
Approximate Price: 65.00

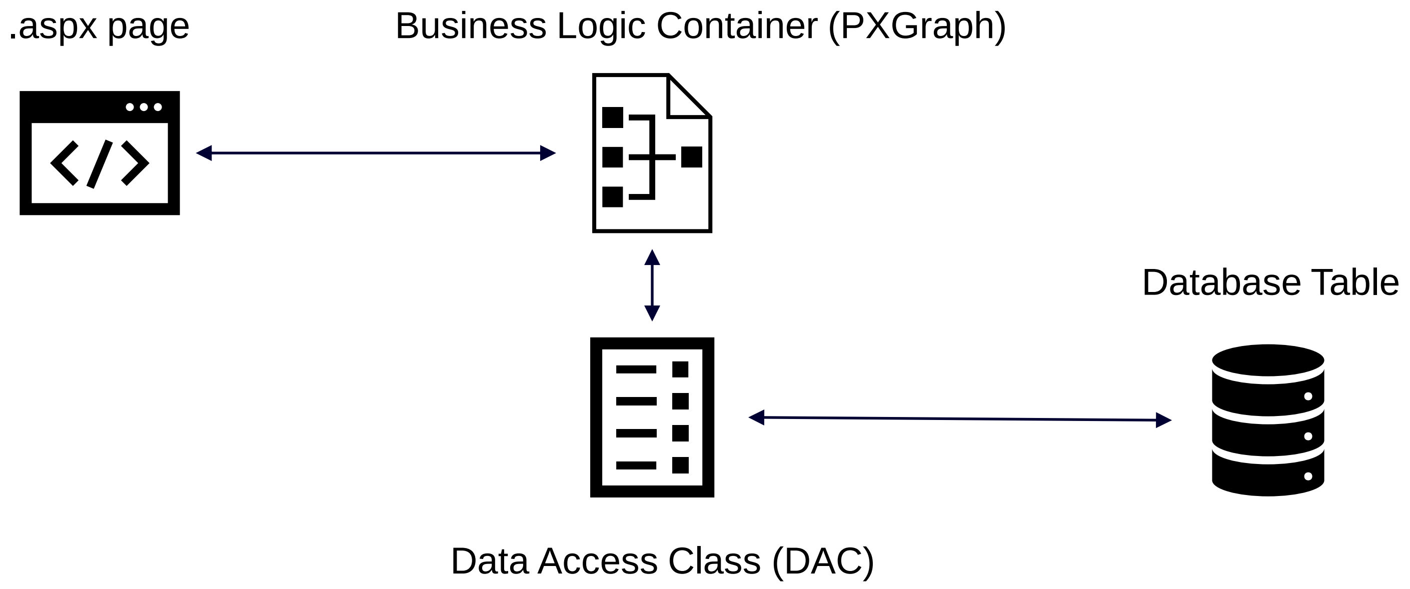
Repair Item Type	Requi	Inventory ID	Description	Price	Defaul
Battery	☑	BAT3310	Battery for Nokia 3310	20.00	☑
Back Cover	☐	BCOV3310	Back cover for Nokia 3310	10.00	☑
Battery	☑	BAT3310EX	Extended Battery for Nokia 3310	30.00	☐

2.1: Defining a Master-Detail Relationship

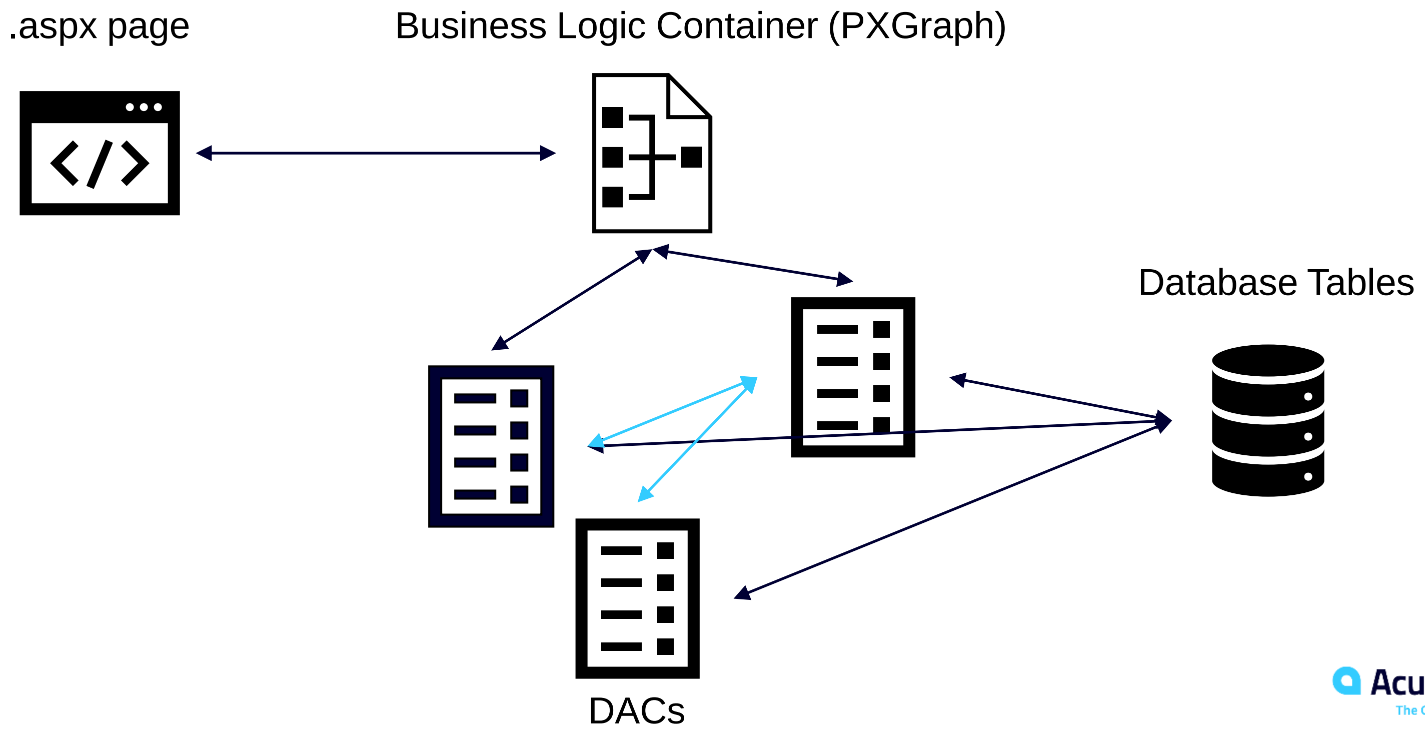
Objectives:

- Define the master-detail relationship between data
- Implement automatic numbering of detail records

Components of the Form



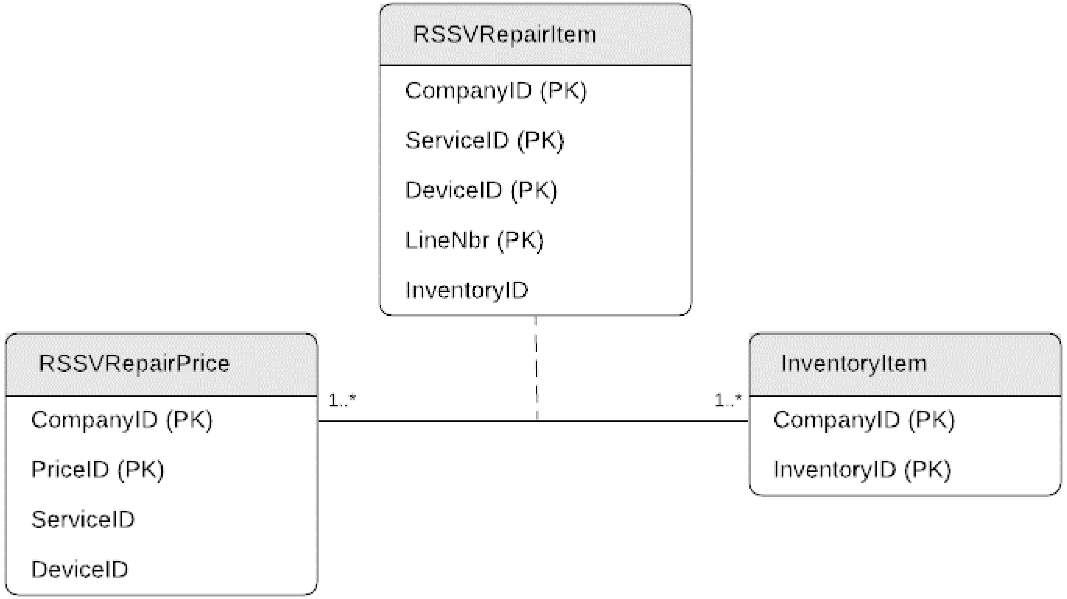
Components of the Form



Defining DACs

Many-To-Many Relationship

Tables for the Repair Items Tab of the Services and Prices Form



Master-Detail

To implement a 1:m relationship one has to:

1. Use `PXDBDefault` attribute in detail DAC, so that it inherits key fields from its parent
2. Use `PXParent` attribute in the detail DAC. This one manages deletion of dependent records and is used as a link by other attributes – e.g. `PXDBIdentity`
3. Set proper constraints in the detail data view. `Current` argument is used for that
4. The order of Data Views defines the order of insertion, update, and deletion operations.

PXSelector & PXRestrictor Attributes

`PXSelector` attribute is *attached to a field that refers other DAC*:

```
[PXSelector(typeof(Document.refNbr))]
public virtual string RefNbr { get; set; }
```

`PXRestrictor` attribute is *attached to a field that has a selector attribute to apply additional restriction to data selection*:

```
[PXSelector(typeof(Document.refNbr))]
[PXRestrictor(typeof(Where<Document.type, Equals<DocTypes.Invoice>>))]
public virtual string RefNbr { get; set; }
```

Business Query Language (BQL)

Current Record

1. Current property of PXCache returns the object of the main DAC type
2. The framework automatically assigns the Current property during Select/Insert/Update operations
3. Can be specified as BQL parameter

```
AccessInfo.businessDate.FromCurrent
Current<AccessInfo.countryCD>,
```

4. Can be set for redirection purpose.

Business Query Language (BQL)

Data View -
```
PXSelect<Product, Where<Product.active, Equals<True>>> Products;
```

BQL Query -
```
typeof(Select<Product, Where<Product.active, Equals<True>>>)
```

Converted

- Is a part of Acumatica Data Access Layer
- Is mapped to SQL queries
- Hides the underlying database engine
- Is checked at compile time
- Comes with a variety of clauses allowing to express most DB queries

BQL – Business Query Language

A simple query can look like:

```
public PXSelect<Product> Products;
```

Result SQL:

```
SELECT Product.ProductCD, Product.Active, ...

FROM Product Product

ORDER BY Produc.ProductCD
```

BQL – Business Query Language

```
public PXSelectJoin<SupplierProduct,
        InnerJoin<Supplier,
                On<Supplier.supplierID, Equal<SupplierProduct.supplierID>>>,
        Where2<
                Where<Current<SupplierFilter.countryCD>, IsNull,
                        Or<Supplier.countryCD, Equal<Current<SupplierFilter.countryCD>>>>,
                And<Where<Current<SupplierFilter.minOrderQty>, IsNull,
                        Or<SupplierProduct.minOrderQty,
                                GreaterEqual<Current<SupplierFilter.minOrderQty>>>>>>,
        OrderBy<Asc<SupplierProduct.productID,
                Asc<SupplierProduct.supplierPrice,
                Desc<SupplierProduct.lastPurchaseDate>>>>> Products;
```

SELECT … FROM …

JOIN … ON …

WHERE (… OR …) AND (… OR …)

(…)

… > "VALUE"

ORDER BY … ASC, … DESC

Acumatica
The Cloud ERP

BQL Putting it all together…

So much loved clauses:

- `Where`
- `Join`
- `OrderBy`
- `GroupBy`

And `PXSelectBase`-d Data View types to combine them:

- `PXSelect<>`
- `PXSelectOrderBy<>`
- `PXSelectJoin<>`
- `PXSelectJoinGroupBy<>`
- `PXSelectReadonly<>`
- and many more

BQL – Querying Data

Passing Parameters from the code – Required:

```
foreach(Product record in PXSelect<Product,
  Where<Product.isActive,
        Equal<Required<Product.isActive>>>>
  >.Select(this, true));
```

Parameter value from context - Current:

```
Product record = PXSelect<Product,
  Where<Product.productID,
        Equal<Current<Tran.productID>>>>
  >.Select(this));
```

BQL - LINQ 2 BQL

Inline mode:

```
var results = graph
    .Select<CRCase>()
    .FirstOrDefault(c
       => c.CaseCD == “000123”);
```

```
SELECT TOP 1 *
FROM CRCase
WHERE CaseCD = ‘000123’
```

Parameter mode:

```
var results = graph
    .Select<CRCase>()
    .FirstOrDefault(c
       => c.CaseCD == “000123”.AsParam());
```

```
DECLARE @P0 NVARCHAR(6)
DECLARE @P0 = ‘000123’
SELECT TOP 1 *
FROM CRCase
WHERE CaseCD = @P0
```

BQL – Fluent BQL

```
SelectFrom<Detail>.
InnerJoin<Document>.
       On<Detail.docType.IsEqual<Document.docType>.
       And<Detail.docNbr.IsEqual<Document.docNbr>>>.
Where<Document.bAccountID.IsEqual<@P.AsInt>.
       And<AccessInfo.businessDate.FromCurrent.
       Diff<Document.date>.Days.IsGreater<TwentyEight>>>.
AggregateTo<GroupBy<Detail.itemID>, Sum<Detail.amount>>.
             Having<Detail.amount.Summarized.IsGreater<Zero>>.
OrderBy<Detail.itemID.Asc, Document.date.Desc>
```

Less brackets in declaration

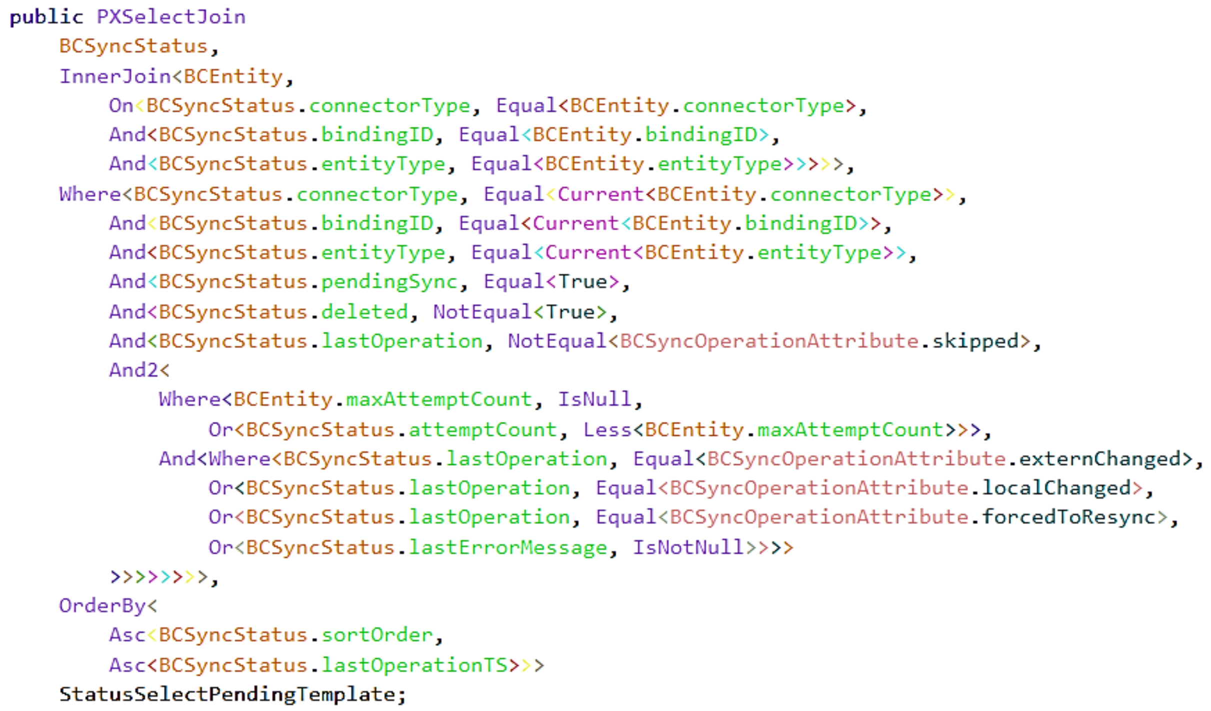
Type validations on comparisons

Easier access to functions and parameters

Having clause

Acuminator – Your Magic Wand

- Validation
- Formatting
- Colorizing



```
public PXSelectJoin
    BCSyncStatus,
    InnerJoin<BCEntity,
        On<BCSyncStatus.connectorType, Equal<BCEntity.connectorType>,
        And<BCSyncStatus.bindingID, Equal<BCEntity.bindingID>,
        And<BCSyncStatus.entityType, Equal<BCEntity.entityType>>>>>,
    Where<BCSyncStatus.connectorType, Equal<Current<BCEntity.connectorType>>,
        And<BCSyncStatus.bindingID, Equal<Current<BCEntity.bindingID>>,
        And<BCSyncStatus.entityType, Equal<Current<BCEntity.entityType>>,
        And<BCSyncStatus.pendingSync, Equal<True>,
        And<BCSyncStatus.deleted, NotEqual<True>,
        And<BCSyncStatus.lastOperation, NotEqual<BCSyncOperationAttribute.skipped>,
        And2<
            Where<BCEntity.maxAttemptCount, IsNull,
                Or<BCSyncStatus.attemptCount, Less<BCEntity.maxAttemptCount>>>,
            And<Where<BCSyncStatus.lastOperation, Equal<BCSyncOperationAttribute.externChanged>,
                Or<BCSyncStatus.lastOperation, Equal<BCSyncOperationAttribute.localChanged>,
                Or<BCSyncStatus.lastOperation, Equal<BCSyncOperationAttribute.forcedToResync>,
                Or<BCSyncStatus.lastErrorMessage, IsNotNull>>>>
        >>>>>>>>,
    OrderBy<
        Asc<BCSyncStatus.sortOrder,
        Asc<BCSyncStatus.lastOperationTS>>>
    StatusSelectPendingTemplate;
```

PXLineNbr – Auto-Numbering Details

`PXLineNbr` attribute is *attached to detail* field and *refers master* field:

```
[PXDBInt(IsKey = true)]
[PXLineNbr(typeof(Document.lastLineNbr))]
public virtual Int32? LineNbr { get; set; }
```

The last detail number is stored in master field (`Document.lastLineNbr`).

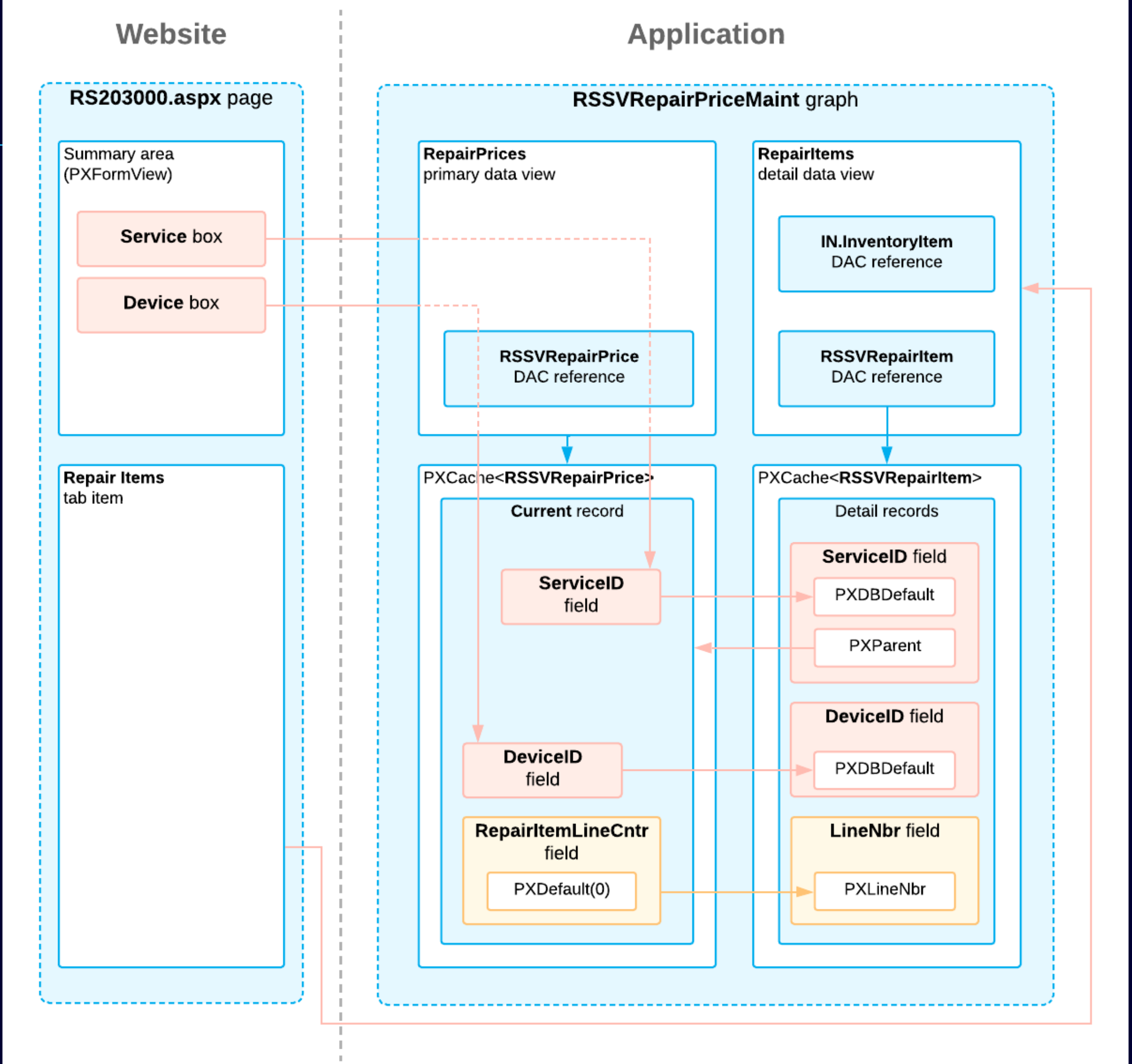
On detail insert `PXLineNbrAttribute` will fetch and increment it, assign the result to the field in the new detail line and store it back in the master.

Note: `PXLineNbr` relies on the `PXParent` – make sure you have one!

Defining PXGraph

Creating aspx Page and Sitemap

Summary



Website
Application
RS203000.aspx page
Summary area (PXFormView)
Service box
Device box
Repair Items tab item
RSSVRepairPriceMaint graph
RepairPrices primary data view
RSSVRepairPrice DAC reference
RepairItems detail data view
IN.InventoryItem DAC reference
RSSVRepairItem DAC reference
PXCache<RSSVRepairPrice>
Current record
ServiceID field
DeviceID field
RepairItemLineCntr field
PXDefault(0)
PXCache<RSSVRepairItem>
Detail records
ServiceID field
PXDBDefault
PXParent
DeviceID field
PXDBDefault
LineNbr field
PXLineNbr

2.2: Defining the Business Logic

Objectives:

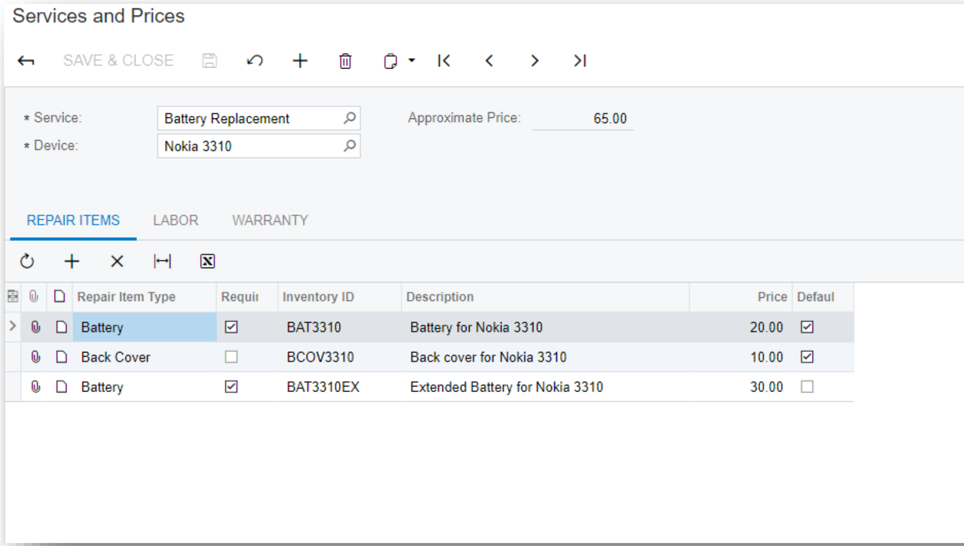
- Restrict the possible values of a field by using the PXRestrictor attribute
- Update the fields of the same data record on update of a field of this record
- Update the fields of other records on update of a field

The Services and Prices Form

• For a particular row, if a value is selected in the Repair Item Type column, the Inventory ID column will display only the stock items that are repair items and have the selected repair item type. If no value is selected in the Repair Item Type column, the Inventory ID column will display all stock items that are repair items.

• For a particular row, if a value is selected in the Inventory ID column, the values in the Repair Item Type and Price columns will be changed to the repair item type and base price (respectively) of the selected stock item as specified on the Stock Items (IN202500) form.

• If the Default check box is selected for a repair item listed in the grid, this check box must be cleared for all other repair items of the same repair item type.



* Service: Battery Replacement
* Device: Nokia 3310
Approximate Price: 65.00

Repair Item Type	Requi	Inventory ID	Description	Price	Defaul
Battery	✓	BAT3310	Battery for Nokia 3310	20.00	✓
Back Cover	☐	BCOV3310	Back cover for Nokia 3310	10.00	✓
Battery	✓	BAT3310EX	Extended Battery for Nokia 3310	30.00	☐

Modifying Data in Cache

Modifying Data in Cache

`var row = cache.Insert()` – simply inserts a new record into cache.

`var row = cache.Insert(object)` – does nothing and returns `null` if a record with the same keys already exists in the cache. Inserts the record and returns the *inserted* one otherwise.

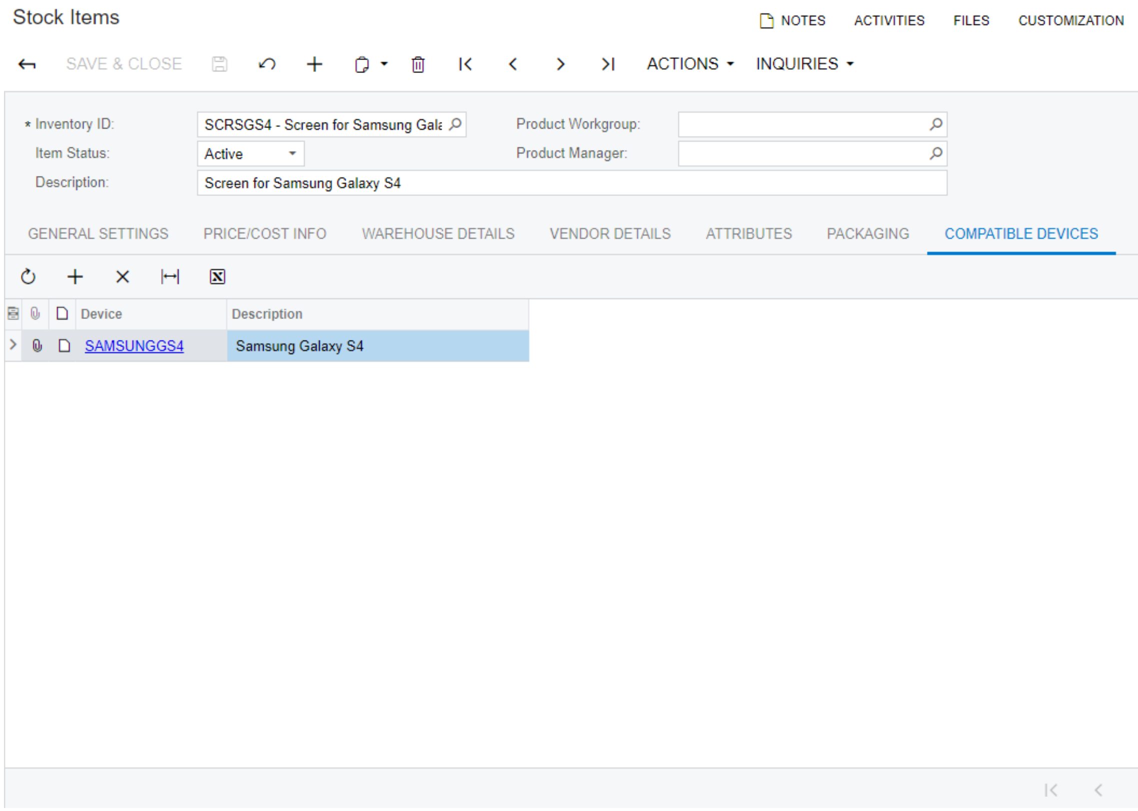
`var row = cache.Update(object)` – updates the record if it is already in the cache. If there is no such record in the cache, gets it from the DB, puts into the cache and updates. If there is no suitable record in the DB, it is inserted with `Insert()`.

`var row = cache.Delete(object)` – sets `Deleted` status for the record. Similarly to the `Update(..)` will go to DB if fails to find the record in the cache. No, the record is not removed from the cache or the database. Feels crazy?

T210: Customization of the Stock Items Form

Stock Items Form

- The **Compatible Devices** tab, which will be used to define and maintain a list of compatible serviced devices for the selected repair item.

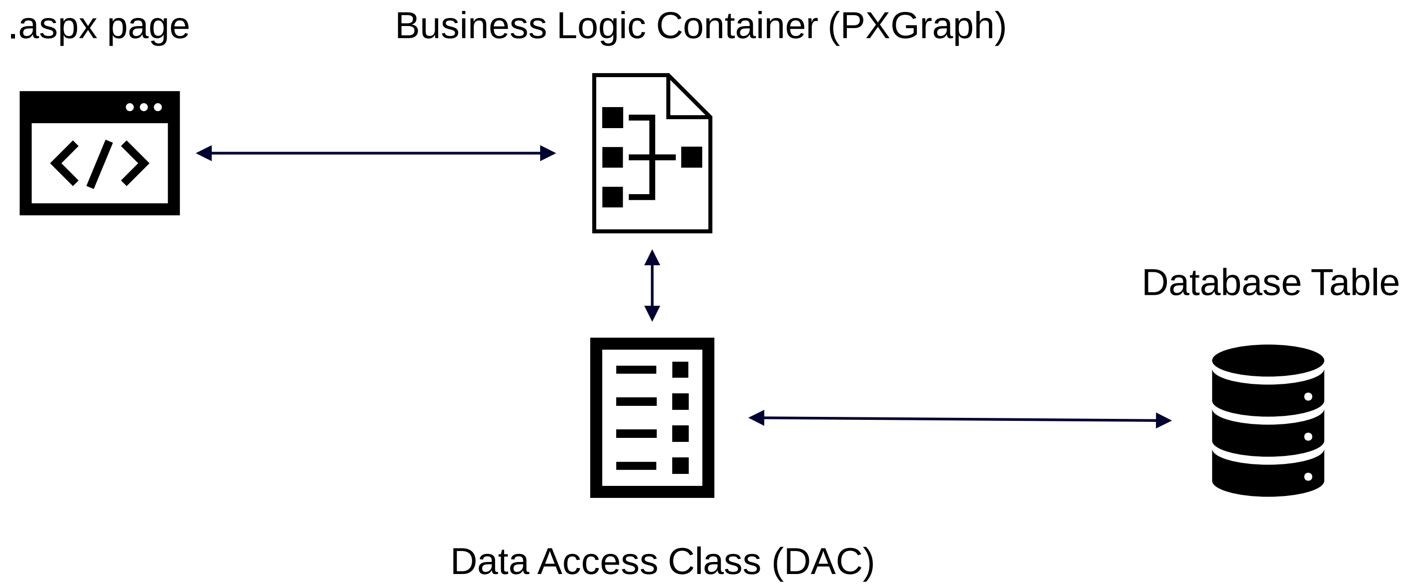


* Inventory ID: SCRSGS4 - Screen for Samsung Gala
Item Status: Active
Description: Screen for Samsung Galaxy S4
Product Workgroup:
Product Manager:

Device	Description
SAMSUNGGS4	Samsung Galaxy S4

Acumatica
The Cloud ERP

Components of the Form



3.1: Adding a New Tab to Stock Items

Objectives:

- Add a custom data view for an Acumatica ERP form
- Create a custom tab on an Acumatica ERP form
- Add a link to a field in an Acumatica ERP grid

T210: Part 3.1 – Q&A

Lesson 4.1: Calculating Field Values using Formulas

Objectives:

- Use the PXFormula attribute for calculations

Formulas

```
[PXDBDecimal(2)]
[PXUIField(DisplayName = “Line Qty")]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXFormula(null, typeof(SumCalc<Document.totalQty>))]
public virtual decimal? LineQty { get; set; }
```

Formulas

Quantities are too simple, you say?

OK, why don’t we do the same with total prices?

Tell the prices of the lines to *calculate themselves and sum up*!

```
[PXDBDecimal(2)]
[PXUIField(DisplayName = “Price")]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXFormula(
        typeof(Mult<Line.lineQty, Line.unitPrice>),
        typeof(SumCalc<Document.totalPrice>))]
public virtual decimal? LinePrice { get; set; }
```

How to **calculate** the field

How to **aggregate** the field

Right, you don’t need any handlers in addition to the formula – it just works

Formulas

Calculate the value of its field:

- `Add<,>`
- `Sub<,>`
- `Mult<,>`
- `Div<,>`
- `Minus<>`
- `Switch<Case<>,>`

This one enables complex conditional calculations

Aggregate the resulting values into a *parent's* field with:

`SumCalc<>`

`CountCalc<>`

`MinCalc<>`

`MaxCalc<>`

Formulas

There is also `PXUnboundFormula` – unlike `PXFormula` it doesn't assign the calculated value to the field, which it is attached to

It calculates what it is told to and aggregates it in the parent, without changing anything in the detail record.

```
[PXDBDecimal(2)]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXUnboundFormula(
        typeof(Switch<Case<Where<Current<ShipmentLine.cancelled>, Equal<False>>,
                             ShipmentLine.lineQty>,
                         decimal_0>),
        typeof(SumCalc<Shipment.totalQty>))]
public virtual decimal? LineQty { get; set; }
```

Note: unbound formula supports only `SumCalc` and `MaxCalc` aggregates.

Formulas

```
[PXFormula(typeof(Validate<…>))]
```

- formula will raise dependentField's FieldVerifying event each time the RelatedField is updated.

```
[PXFormula(typeof(Current<…>))]
```

- formula fetches the field value from the record stored in the Current property of the DAC’s cache.

```
[PXFormula(typeof(Parent<…>))]
```

- formula fetches the field value from the parent data record as defined by PXParentAttribute.

```
[PXFormula(typeof(IsTableEmpty<…>))]
```

- formula returns true if the DB table corresponding to the specified DAC contains no records.

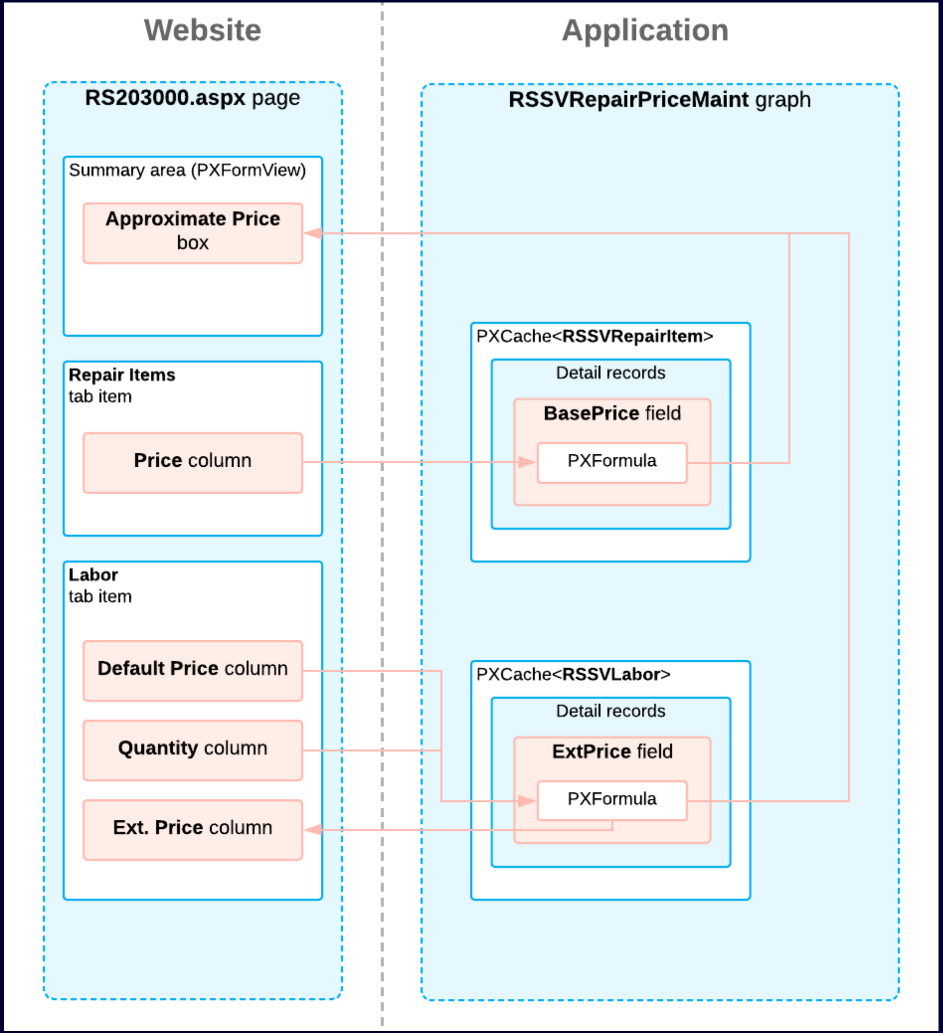
```
[PXFormula(typeof(Selector<…>))]
```

- formula can evaluate and update value from foreign record referenced by selector.

```
[PXFormula(typeof(Default<…>))]
```

- Raises the FieldDefaulting for the field to which the formula is attached once the specified field changes.

Summary



Acumatica The Cloud ERP

Lesson 4.2: Inserting a Default Detail Record

Objectives:

- Add a default detail record to the grid

Event Handlers in Attributes and Graphs

-*ing* events (e.g. `FieldDefaulting, RowUpdating`):

- *Graph* handlers first, then *attribute* handlers
- Graph handlers can cancel execution of attribute handlers by setting `e.Cancel= true`

-*ed* events (e.g. `FieldUpdated, RowUpdated`):

- *Attribute* handlers first, then *graph* handlers
- One typically can’t cancel anything in graph

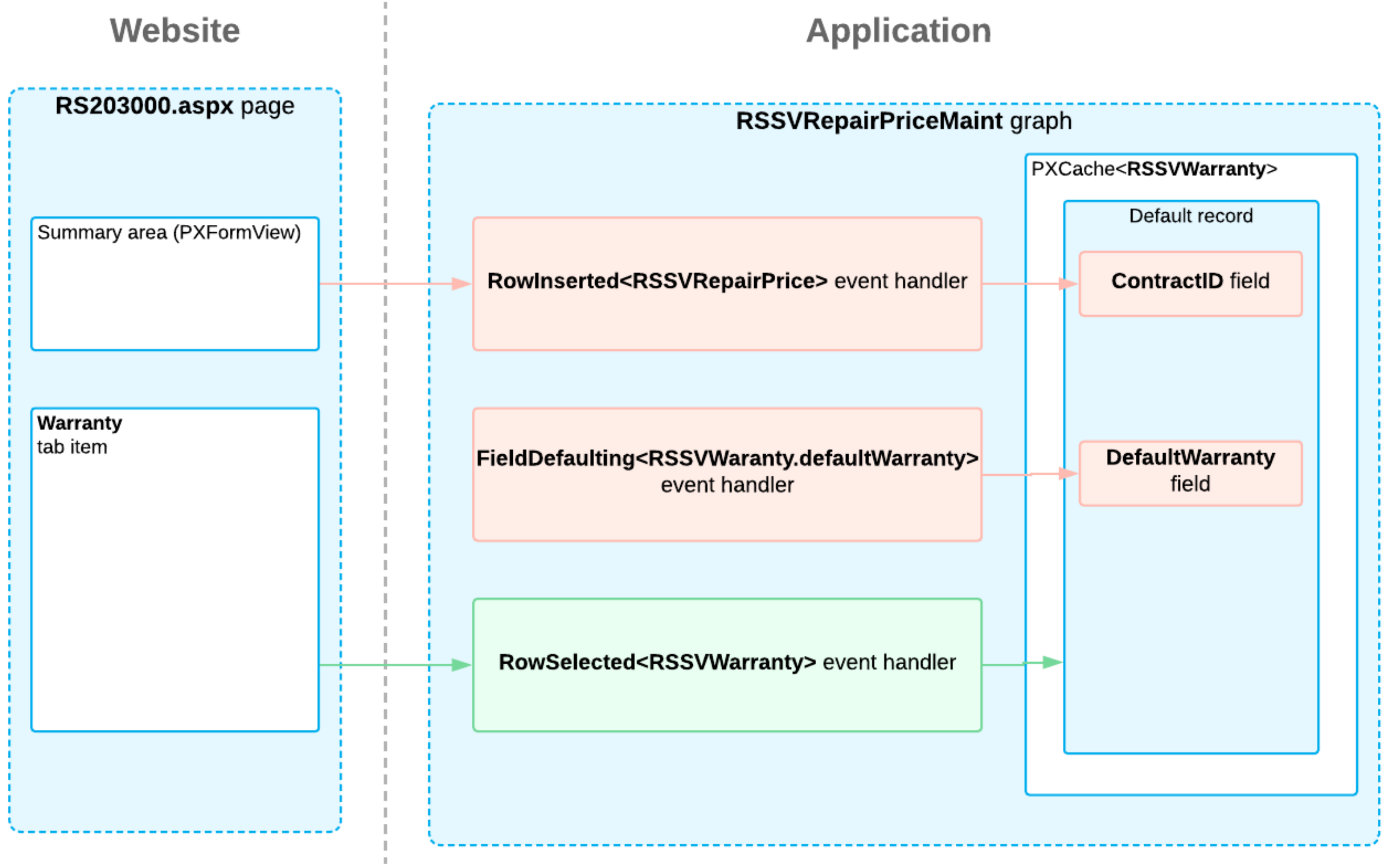
Inserting a Record

You just type

```
RSSVWarranty line = new RSSVWarranty();
line.ContractID = defaultWarranty.ContractID;
Warranty.Insert(line);
```

And …

Summary



Website
Application
RS203000.aspx page
Summary area (PXFormView)
Warranty tab item
RSSVRepairPriceMaint graph
RowInserted<RSSVRepairPrice> event handler
FieldDefaulting<RSSVWaranty.defaultWarranty> event handler
RowSelected<RSSVWarranty> event handler
PXCache<RSSVWarranty>
Default record
ContractID field
DefaultWarranty field

Acumatica
The Cloud ERP

Thank You

Dmitrii Naumov