Acumatica
The Cloud ERP

T240 Processing Forms

Evgeny Afanasiev

Technical Account Manager

Timing and Agenda

September 20, 2022 - 9:00-10:30 am PST

Day 1

Lesson 1.1: Creating a Simple Processing Form

September 21, 2022 - 9:00-10:30 am PST

Day 2

Lesson 1.2: Adding Filtering Parameters to the Processing Form

Lesson 2.1: Implementing a Custom PXAccumulator Attribute

Timing and Agenda

September 22, 2022 - 9:00-10:30 am PST

Day 3

Lesson 2.2: Modifying the Processing Form to Use the Field Updated by PXAccumulator

Lesson 3.1: Adding Redirection to a Report at the End of Processing

Day 1

xByte Technologies
Acumatica Customer Since 2014

Acumatica
The Cloud ERP

Introduction and Company story

Introduction

In this course we will learn how to create processing forms by using Acumatica Framework and the customization tools of Acumatica ERP.

A processing form is a form on which users can invoke an operation on multiple selected records at once.

Company Story

Let’s continue the development to support the cell phone repair shop of the Smart Fix company

Done so far:

- The Repair Services Screen(RS201000)
- The Serviced Devices (RS202000)
- The Services and Prices (RS203000)
- The Repair Work Orders (RS301000)
- The Repair Work Order Preferences (RS101000)

To do:

Assign Work Orders (RS501000) custom processing form

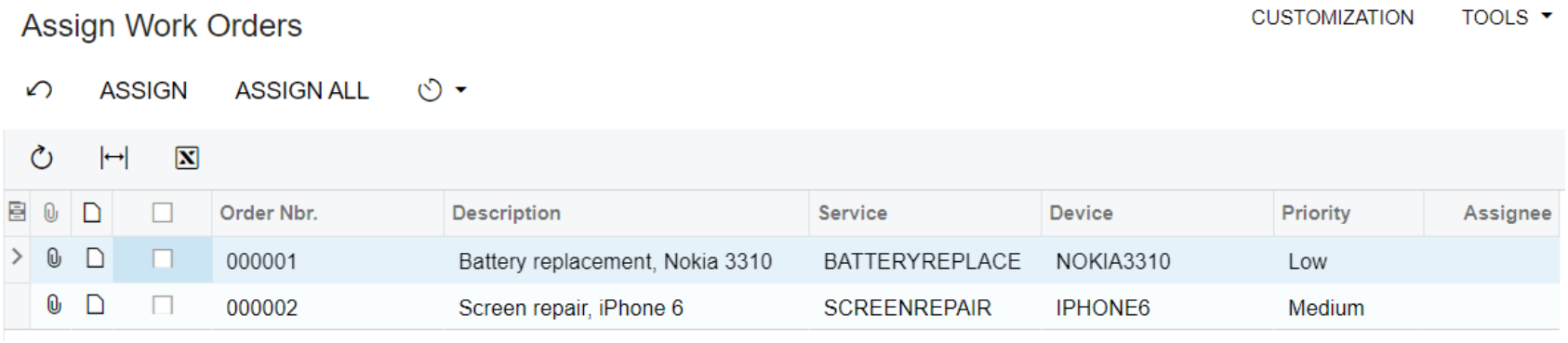
This will allow users to assign multiple repair work orders at the same time.

Part 1: Processing Form (Assign Work Orders)

Lesson 1.1: Creating a Simple Processing Form

Learning Objectives

In this lesson, you will learn how to create a simple processing form (that is, one that does not have any filtering parameters defined).



Step 1.1.1: Creating the Form

Let’s create a from with following parameters:

- **Screen ID: RS.50.10.00**
- **Graph Name: RSSVAssignProcess**
- **Graph Namespace: PhoneRepairShop**
- **Page Title: Assign Work Orders**
- **Template: Grid (GridView)**

Step 1.1.2: Changing the Processing Action

Let’s modify the Assign action of the Repair Work Order form:

- **Move the action delegate to the separate static method.**
- **Change signature of the method (return type void -> IEnumerable)**
- **Change Attribute (PXButton->PXProcessButton)**
- **Use PXLongOperation**
- **Set isMassProcess flag**
- **Use PXProcessing.SetInfo() / PXProcessing<T>.SetError() Methods**
- **Test modifications**

Step 1.1.3: Configuring the Processing Graph and Data View

Let’s configure the Processing Graph:

- **Define the dataview of PXProcessing type**
- **Specify captions for processing actions**
- **Define Workflow action in RowSelected event handler**

Important Note:

- **For forms that using WorkFlow set the processing action in RowSelected method using SetProcessWorkflowAction(). Never call this method in the constructor of the BLC.**
- **For forms that not using the WorkFlow the processing method should be set using SetProcessDelegate() in BLC constructor.**

Step 1.1.4: Creating Controls for the Processing Form

Let’s add the Grid to the Processing Screen:

- **Add unbound “Selected” field to the Work Order DAC**
- **Link the Grid control to the PXProcessing dataview**
- **Add needed fields (columns) in the Grid**
- **Configure the Grid and its Fields layout parameters**

Lesson Summary

In this lesson, you have learned how to create a simple processing form that displays data to be processed and provides processing actions. For the processing form, you have defined:

- In the processing graph, the specific PXProcessing (derived from PXProcessingBase) data view type to provide data records for the form.
- In the graph constructor, the names of the processing buttons.
- In the RowSelected event handler, the workflow action to be used for processing.
- In the DAC, the unbound Selected data field, which is used to indicate the records to be processed.
- In the ASPX page, the column in the grid for the Selected data field.

Day 2

Cherrylake, Inc.
Acumatica Customer Since 2013

Acumatica
The Cloud ERP

Timing and Agenda

Day 2

Lesson 1.2: Adding Filtering Parameters to the Processing Form

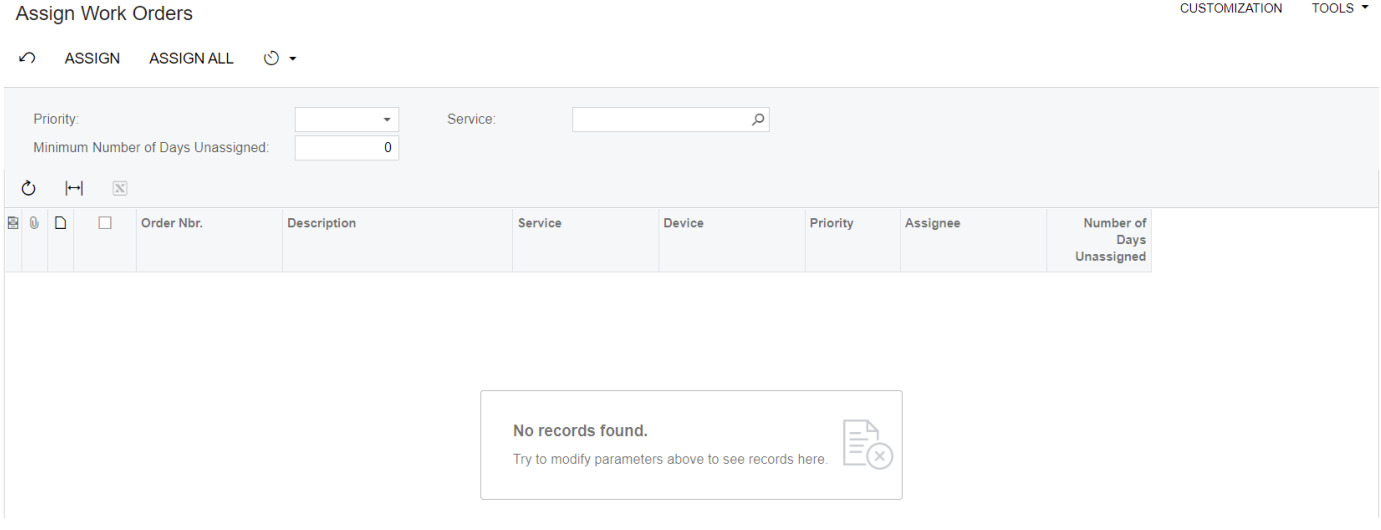
Lesson 2.1: Implementing a Custom PXAccumulator Attribute

Lesson 1.2: Adding Filtering Parameters to the Processing Form

Learning Objectives

In this lesson, you will learn how to do the following:

- Create processing pages with filtering parameters
- Use the PXDBCalced attribute



Step 1.2.1: Extending the DAC with a New Field (Using PXDBCalced)

Let’s extend the RSSVWorkOrder DAC

- **Add TimeWithoutAction integer DAC Field**
- **Decorate TimeWithoutAction with PXDBCalcedAttribute**

Step 1.2.2: Defining the Filter DAC

Let’s specify the DAC to hold filter values

- **Define DAC with following UNBOUND Non-Key Fields:**
  - ServiceID
  - TimeWithoutAction
  - Priority
- **Decorating DAC with PXHiddenAttribute**

Step 1.2.3: Defining the Data Views (with PXFilter and PXFilteredProcessing)

Let’s configure the Filter

- **Define PXFilter view type for Form header**
- **Replace PXProcessing with PXFilteredProcessing view type for the Grid**
- **Enable Assignee selector on the Grid**
- **Override isDirty graph property to always false.**

Step 1.2.4: Adjusting the ASPX Page (with SyncPosition and AutoRefresh)

Let’s adjust ASPX page

- **Redefine the PrimaryView to PXFilter view**
- **Add form control for Filter**
- **Add controls for Filter fields**
- **Configure layout parameters**

Lesson Summary

In this lesson, you have learned how to define processing forms with filtering parameters. Because you have already implemented the processing method in the previous lesson, to add a filter to the processing form, you have completed the following steps:

1. Prepared the DAC that provides records for processing.

2. Defined the DAC that provides the filtering parameters for the processing form.

3. In the graph, defined the following data views:

• The data view of the PXFilter type, which provides data for the filter

• The data view of the PXFilteredProcessing type, which retrieves records for possible processing

4. In the graph, modified the RowSelected event handler so that it uses the primary DAC of the primary data view, which is the RSSVWorkOrderToAssignFilter DAC.

You have also used the PXUIFieldAttribute.SetEnabled<>() method in the graph constructor to enable editing for the Assignee data field.

Part 2: Update of Data with a Custom Accumulator Attribute

Lesson 2.1: Implementing a Custom PXAccumulator Attribute

Learning Objectives

In this lesson, you will learn how to implement a custom attribute derived from the PXAccumulator attribute.

Step 2.1.1: Preparing the Data

Let’s complete all the open Repair Work Orders

Step 2.1.2: Creating a DAC

Let’s create DAC to accumulate the number of assigned orders for each employee

- **Generate RSSVEmployeeWorkOrderQty DAC from the database**
- **Decorate the DAC with PXHiddenAttribute**
- **Mark UserId as Key field**

Step 2.1.3: Implementing the Accumulator Attribute

Let’s define the Custom PXAccumulator attribute

- **Derive the attribute from the PXAccumulatorAttribute**
- **Enable single-record update mode by setting _SingleRecord field in constructor**
- **Implement restriction not to assign more than 10 orders per employee in PrepareInsert() method**

Lesson Summary

In this lesson, you have learned how to create a custom accumulator attribute to summarize the numbers of assigned work orders for each employee during the assignment or completion of work orders.

In the custom attribute, you have defined the following elements:

• The constructor, in which you have specified the update mode for the records

• The PrepareInsert() method, in which you have defined the updating policy for the particular field (the values of this field are summarized) and specified the restriction for the values of this field You have also assigned the custom accumulator attribute to the DAC that stores the field to be updated by the accumulator attribute.

Day 3

Curran Young Construction Company
Acumatica Customer Since 2018

Timing and Agenda

Day 3

Lesson 2.2: Modifying the Processing Form to Use the Field Updated by PXAccumulator

Lesson 3.1: Adding Redirection to a Report at the End of Processing

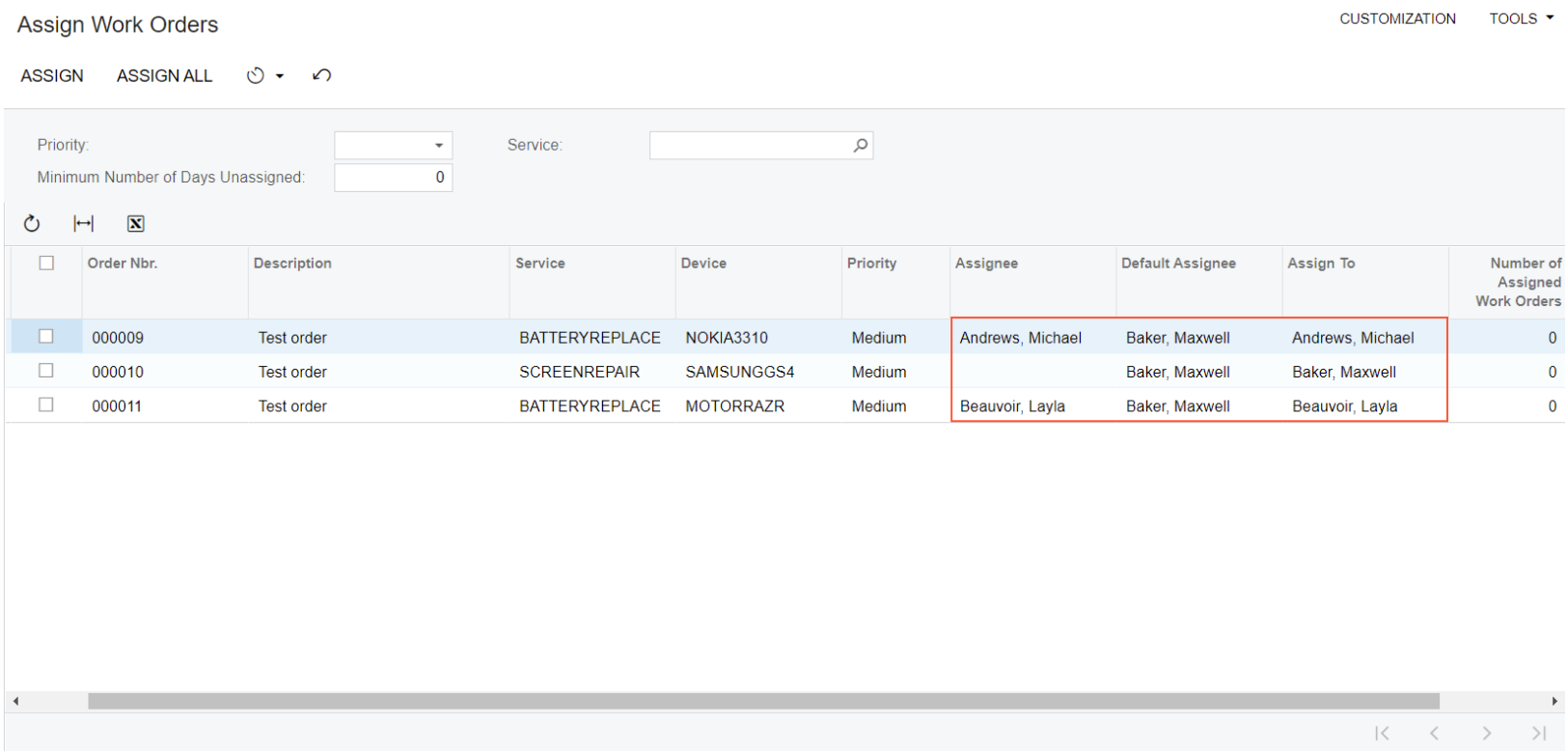
Lesson 2.2: Modifying the Processing Form to Use the Field Updated by PXAccumulator

Learning Objectives

In this lesson, you will learn how to do the following:

- Specify the values of the fields updated by a PXAccumulator attribute
- Use the PXDBScalar attribute
- Append and replace attributes on a certain DAC field within a particular graph
- Define the external presentation of field values

Figure: The assignees on the Assign Work Orders form



Step 2.2.1: Extending the DAC with New Fields

Let’s and some more fields to the DAC, that stores WorkOrders

Add to the RSSVWorkOrder following fields:

- **DefaultAssignee**
- **AssignTo**
- **NbrOfAssignedOrders**

Step 2.2.2: Replacing Field Attributes

Let’s append attributes to the DAC fields using CacheAttached Event

Modify decoration of

- **DefaultAssignee DAC field with adding**
  - **PXMergeAttribute**
  - **OwnerAttribute**
  - **PXDBScalarAttribute**
- **AssignTo DAC field with adding**
  - **PXMergeAttribute**
  - **OwnerAttribute**
  - **PXUnboundDefaultAttribute**

Step 2.2.3: Modifying the Assignment and Completion Operations

Let’s dynamically count the number of orders assigned

Change value of NbrOfAssignedOrders by

- **Adding “1” in the AssignOrders() method**
- **Subtracting “1” in the complete() method**

Step 2.2.4: Defining the External Presentation of Field Values

Let’s use FieldSelecting event handler to Field Value presented in Web Interface.

Return “0” in e.ReturnValue if the value in Database is “NULL”

Important Note:

- **For internal presentation of the Value use return e.NewValue in FieldUpdating event hander**
- **For external presentation of the Value use return e.ReturnValue in FieldSelecting event handler**

Step 2.2.5: Adjusting the ASPX Page

Let’s extend the processing grid

Add columns:

- **Default Assignee**
- **Assigned To**
- **Number of Assigned Work Orders**

Configure fields layout properties

Lesson Summary

In this lesson, you have learned how to implement a processing operation by using a static method and how to change the values of the fields that are updated by a PXAccumulator attribute.

You have modified the implementation of the AssignOrders() method and the complete() action handler of the RSSVWorkOrderEntry graph so that 1 is added to or subtracted from the number of assigned work orders. The value that is specified for the number of assigned work orders in the AssignOrders() method and the complete() action handler is added to the value stored in the database by the custom PXAccumulator attribute.

You have learned how to use the PXDBScalar and PXUnboundDefault attributes and how to define the external presentation of a field value.

You have also learned how to replace attributes of a DAC field using the CacheAttached event handler.

Part 3: Redirection to a Report at the End of Processing

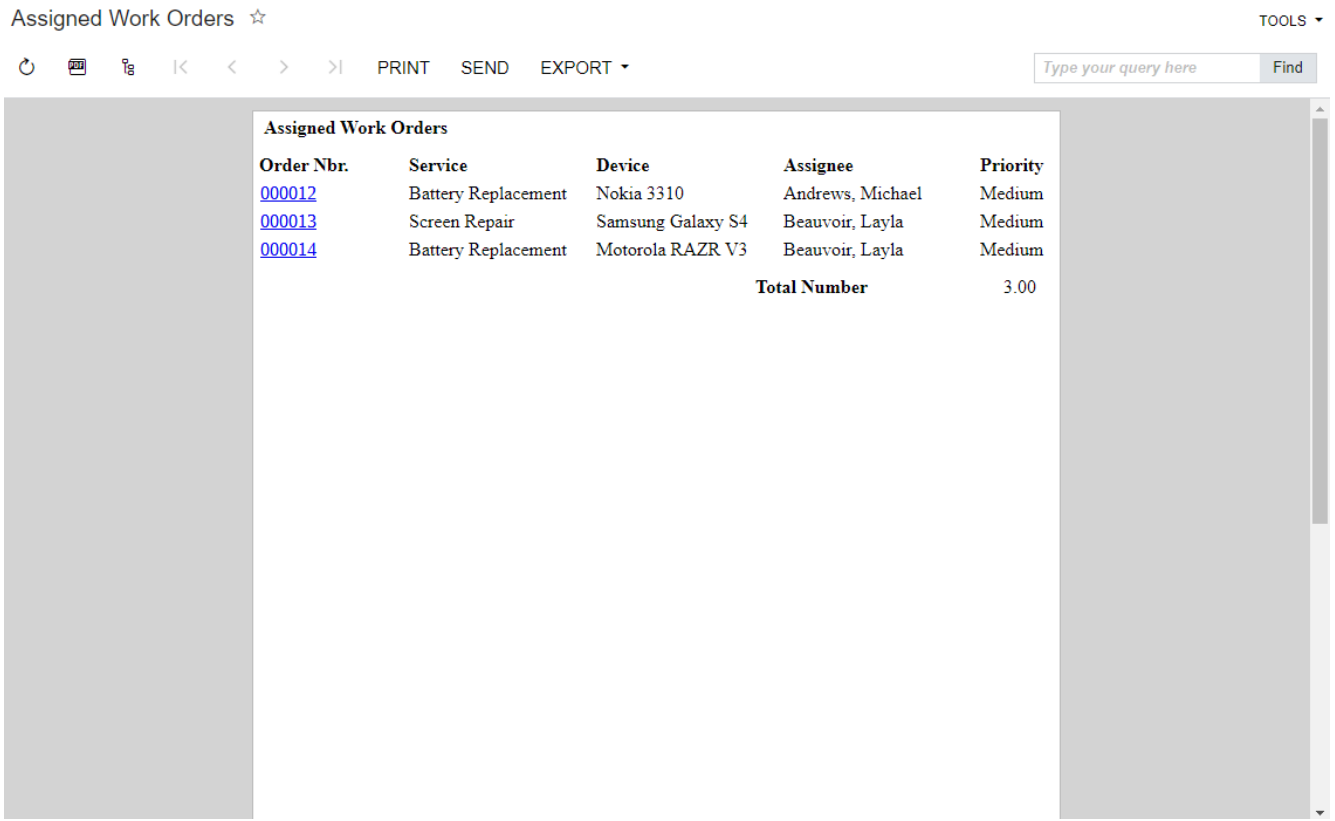
Lesson 3.1: Adding Redirection to a Report at the End of Processing

Learning Objectives

In this lesson, you will learn how to do the following:

- Redirect to a report at the end of the processing delegate
- Include a report in a customization project

Figure: The report



Step 3.1.1: Including a Report in the Customization Project

Let’s add the report and test it

- **Add RS601000.rpx to the customization project as file**
- **Configure Site Map node as following:**
  - **Screen ID: RS.60.10.00**
  - **Title: Assigned Work Orders**
  - **URL: ~/frames/reportlauncher.aspx?id=RS601000.rpx**
  - **Graph Type: Empty**
  - **Workspaces: Empty**
  - **Category: Empty**
- **Test that report is working**

Step 3.1.2: Adding Redirection to a Report

Let’s open the report with processed items after the processing

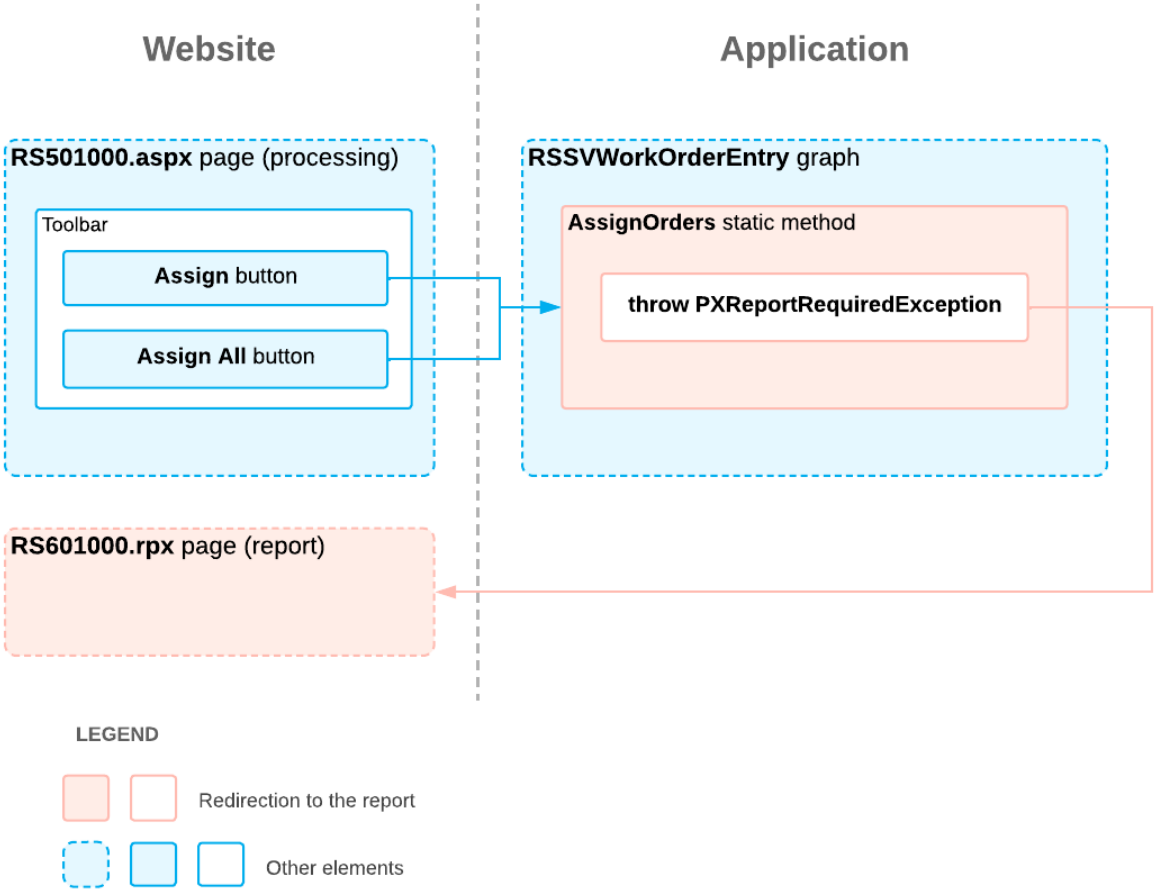
Implement PXReportRequiredException in the AssignedOrders method

Lesson Summary

In this lesson, you have learned how to implement the redirection to a report at the end of the processing delegate. You have used the PXReportRequiredException exception to perform the redirection.

You have passed the result set with the data of the repair work orders that have been assigned to the PXReportRequiredException constructor.

Redirection to the Report



No Reliance
This document is subject to change without notice. Acumatica cannot guarantee completion of any future products or program features/enhancements described in this document, and no reliance should be placed on their availability.

Confidentiality: This document, including any files contained herein, is confidential information of Acumatica and should not be disclosed to third parties.

Acumatica
The Cloud ERP

Thank you

Evgeny Afanasiev