



T200 Maintenance Forms

Vidhyalakshmi Hariharasubramanian

Technical Account Manager

Timing and Agenda

June 27, 2022 -10 AM -11 AM

Day 1

Lesson 1.1: Prepare a Customization Project

Lesson 1.2: Create a Form

Lesson 1.3: Make the New Form Visible in the UI

June 28, 2022 -10 AM -11 AM

Day 2

Lesson 1.4: Configure the Data Access Class

Lesson 1.5: Configure the Form

Lesson 1.6: Add an Event Handler to the Walk-In Service Check Box

Lesson 1.7: Debug the Customization Code

Timing and Agenda

June 29, 2022 -10 AM -11 AM

Day 3

Lesson 1.8: Move the Customization Code to an Extension Library

Lesson 1.9: Add an Event Handler In Visual Studio

Lesson 2.1: Create a Graph and a DAC in Visual Studio

June 30, 2022 -10 AM -11 AM

Day 4

Lesson 2.2: Create an ASPX Page in Visual Studio

Lesson 2.3: Configure a Form in Visual Studio

Lesson 2.4: Add the Form to the Site Map and Workspace

Lesson 2.5: Create a Substitute Form



Day 1

Part 1: Repair Services Maintenance Form

Lesson 1.1: Prepare a Customization Project

Learning Objectives

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

- What customization project is
- How to create a customization project
- How to add a database script for a database table to the customization project

Lesson Summary

In this lesson, you have learned how to create customization projects and add database scripts to the project.

As you have completed the lesson, you have created the *PhoneRepairShop* customization project on the Customization Projects (SM204505) form. You have used the Customization Project Editor to add database scripts for custom tables to the customization project.

Lesson 1.2: Create a Form

Learning Objectives

In this lesson, you will learn how to create a form of the application by generating the needed items with the New Screen wizard.

Lesson 1.2: Create a Form

Column Name	Data Type	Description
Service ID	String	The identifier of the service
Description	String	The description of the service
Active	Boolean	An indicator of whether a service is currently provided by the shop
Walk-In Service	Boolean	An indicator of whether this is a walk-in service
Requires Preliminary Check	Boolean	An indicator of whether the service requires diagnostic checks
Requires Prepayment	Boolean	An indicator of whether this service should be prepaid

Figure: The Customized Screen page of the Customization Project Editor

Customization Project Editor [Back](#) [Reload](#)

File Publish Extension Library Source Control

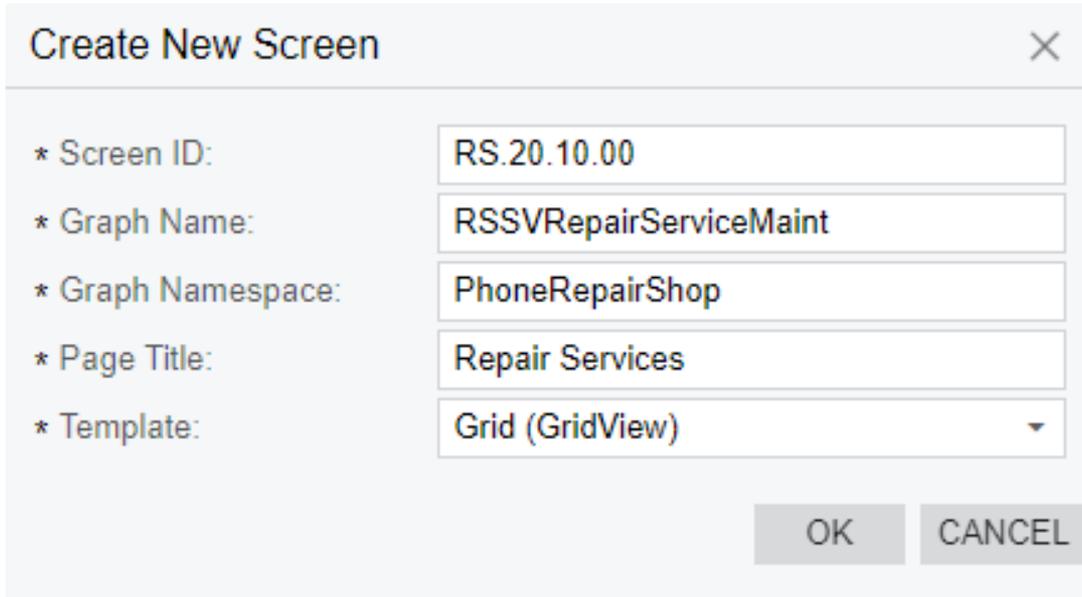
PhoneRepairShop **Customized Screens**

🔄 📄 ↶ ✕ + ✎ CREATE NEW SCREEN CUSTOMIZE EXISTING SCREEN

Screen ID	Title	Is New	Last Modified By	Last Modified On
No records found.				

Screens
Data Access
Code
Files
Generic Inquiries
Reports
Dashboards
Site Map
Database Scripts
System Locales
Import/Export Scenarios
Shared Filters
Access Rights
Wikis
Web Service Endpoints
Analytical Reports
Push Notifications
Business Events
Mobile Application
User-Defined Fields
Webhooks
Connected Applications

Figure: The Create New Screen dialog box



Create New Screen

* Screen ID: RS.20.10.00

* Graph Name: RSSVRepairServiceMaint

* Graph Namespace: PhoneRepairShop

* Page Title: Repair Services

* Template: Grid (GridView)

OK CANCEL

Lesson 1.2: Create a Form

Item	Description
RS201000	This <i>Screen</i> item contains the new page content.
RSSVRepairService-Maint	This <i>Code</i> item contains the code template of the graph for the new form. This item is saved in the database. When you publish the project, the platform creates a copy of the code in the <code>RSSVRepairServiceMaint.cs</code> file in the <code>App_RuntimeCode</code> folder of the Acumatica ERP application instance.
Pages\RS \RS201000.aspx Pages\RS \RS201000.aspx.cs	These <i>File</i> items contain ASPX page code for the new form. When you publish the project for the first time, the platform creates the files in the <code>Pages\RS</code> folder of the Acumatica ERP application instance, and the platform creates copies of these files in the <code>pages_RS</code> subfolder of the <code>CstPublished</code> folder of the instance.
Repair Services	This <i>SiteMapNode</i> item contains the site map object of the new form.

Lesson Summary

In this lesson, you've learned how to create a new form by using the New Screen wizard. From the example of the Repair Services form you created, you have learned about the following basic components of a form:

- The ASPX page
- The graph
- The site map node

Lesson 1.3: Make the New Form Visible in the UI

Learning Objectives

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

- Create a workspace
- Add a link to a custom form to the workspace
- Update the *SiteMapNode* item in the customization project

Figure: The Phone Repair Shop workspace in Menu Editing mode

The screenshot displays the 'Phone Repair Shop' workspace in 'Menu Editing mode'. The interface is divided into three main sections:

- Left Sidebar:** A vertical list of menu items with icons. The 'Phone Repair Shop' item is highlighted with a red border. Other items include Data Views, Bills of Material, Product Configurator, Time and Expenses, Finance, Banking, Project Management, Compliance, Currency Manage..., and Taxes.
- Central Workspace:** The title 'Phone Repair Shop' is at the top, followed by a red-bordered pin icon. Below the title, the text reads: 'Select the items to be added to the quick menu.'
- Right Panel:** Contains 'MENU SETTINGS' at the top, followed by 'CUSTOMIZATION' and 'TOOLS'. Below this is a table with one row: 'Last Modified On' with the value '2/20/2021'. At the bottom of the panel are navigation arrows: '< < > >'.

At the bottom of the interface, there is a black bar with 'EXIT MENU EDITING' and a blue banner with the message: 'Your product is in trial mode. Only two concurrent users are allowed.'

Figure: The Select Forms dialog box

The screenshot displays the Acumatica user interface with a 'Select Forms' dialog box open. The background shows a menu editor for 'Phone Repair Shop' with options like 'Data Views', 'Bills of Material', and 'Product Configurator'. The dialog box has a search bar containing 'Repair' and a list of form categories. Under the 'Other' category, the 'Repair Services' form is selected with a checked checkbox. At the bottom of the dialog, there are three buttons: 'ADD', 'ADD & CLOSE', and 'CLOSE'. A blue banner at the bottom of the screen reads: 'Your product is in trial mode. Only two concurrent users are allowed.'

Figure: The Edit button

The screenshot displays the Acumatica menu editor interface. The top navigation bar includes options for '+ ADD WORKSPACE', '+ ADD MENU ITEM', '+ ADD TILE', and 'MENU SETTINGS'. The left sidebar lists various menu items, with 'Phone Repair Shop' selected. The main content area shows the configuration for 'Phone Repair Shop', including a selection prompt and a list of items under the 'Other' category. The 'Repair Services' item is highlighted, and a blue callout box points to the 'Edit link parameters' button. The right sidebar shows 'CUSTOMIZATION' and 'TOOLS' options, along with a 'Last Modified On' date of 2/20/2021. A blue banner at the bottom contains the text 'Product is in trial mode. Only two concurrent users are allowed.'

+ ADD WORKSPACE + ADD MENU ITEM + ADD TILE MENU SETTINGS

Data Views

Phone Repair Shop

Bills of Material

Product Configurator

Time and Expenses

Finance

Banking

Project Management

Compliance

Currency Manage...

Taxes

EXIT MENU EDITING

Phone Repair Shop

Select the items to be added to the quick menu.

Other

Repair Services [Edit link parameters](#)

CUSTOMIZATION TOOLS

Last Modified On

2/20/2021

Product is in trial mode. Only two concurrent users are allowed.

Figure: The Phone Repair Shop workspace

The screenshot displays the Acumatica user interface for the 'Phone Repair Shop' workspace. The top navigation bar includes the Acumatica logo, a search bar, a refresh icon, the user name 'Yogifon', the date '2/20/2021 4:30 AM', a help icon, and the user profile 'admin, admin'. The left sidebar lists various modules: Favorites, Data Views, Phone Repair Shop (selected), Time and Expenses, Finance, Banking, Payables, Receivables, Sales Orders, Purchases, and Inventory. The main content area is titled 'Phone Repair Shop' and contains a 'Configuration' section with a link for 'Repair Services'. The right sidebar shows 'CUSTOMIZATION' and 'TOOLS' options, with a table listing 'Last Modified On' as '2/20/2021'. A blue banner at the bottom of the workspace states: 'Your product is in trial mode. Only two concurrent users are allowed.' and includes an 'ACTIVATE' button.

Figure: The Site Map page

The screenshot shows the 'Customization Project Editor' interface. The top navigation bar includes 'File', 'Publish', 'Extension Library', and 'Source Control'. The main header displays 'PhoneRepairShop' and 'Site Map'. Below the header, there are icons for refresh, save, undo, and close, along with buttons for 'RELOAD FROM DATABASE' and 'MANAGE SITE MAP'. A table lists site map objects with columns for 'Object Name', 'Description', 'Last Modified By', and 'Last Modified On'. The table contains one entry: 'Repair Services' with 'admin admin' as the last modified by and '9/20/2021' as the last modified on. A left-hand sidebar lists various project categories, with 'Site Map (1)' currently selected.

Object Name	Description	Last Modified By	Last Modified On
Repair Services		admin admin	9/20/2021

Lesson Summary

In this lesson, you have learned how to add a new workspace to the Acumatica ERP UI, add links to a workspace, and update the *SiteMapNode* item of the customization project.



Day 2

Lesson 1.4: Configure the Data Access Class

Learning Objectives

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

- Generate and configure a DAC
- Configure a view in a graph

The connection between DAC fields and database fields

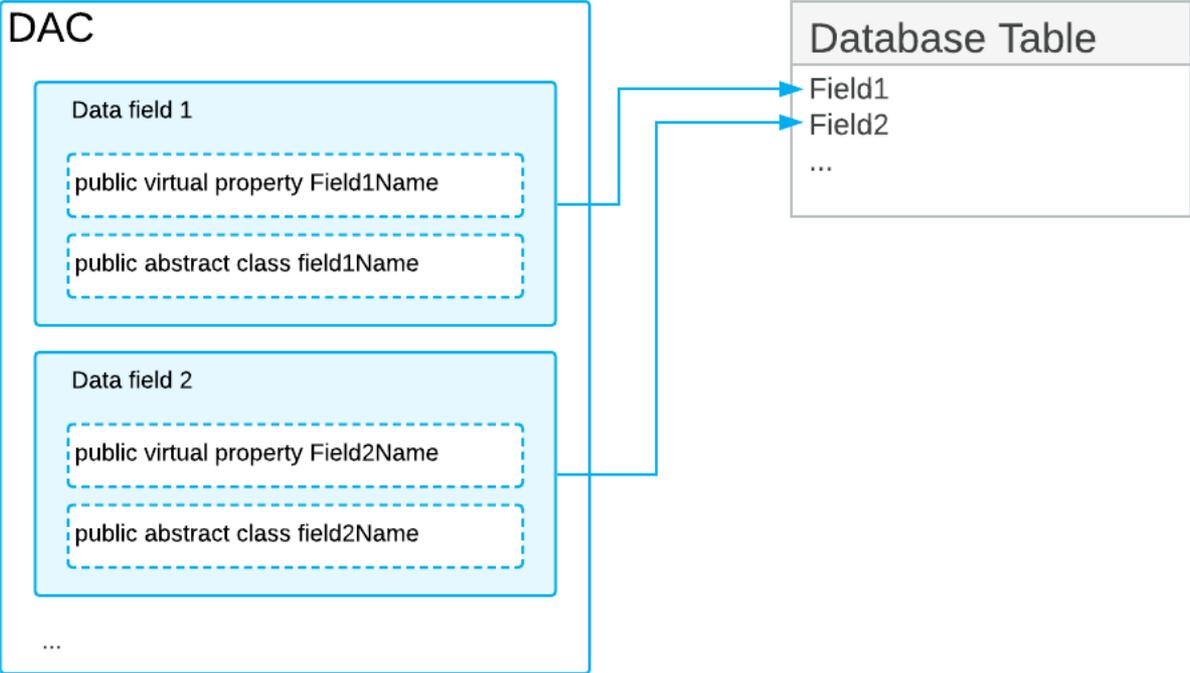
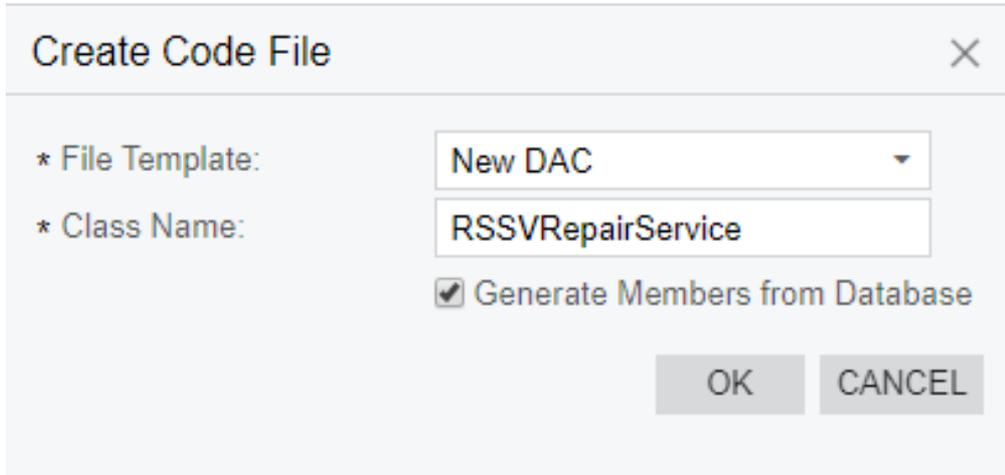


Figure: The Create Code File dialog box



Create Code File

* File Template: New DAC

* Class Name: RSSVRepairService

Generate Members from Database

OK CANCEL

Figure: The RSSVRepairService DAC code

The screenshot displays the Customization Project Editor interface. The top bar is blue with the text 'Customization Project Editor' on the left and 'Back' and 'Reload' on the right. Below the bar is a menu with 'File', 'Publish', 'Extension Library', and 'Source Control'. The main area is titled 'PhoneRepairShop' and contains a tree view on the left with 'CODE' expanded to show 'RSSVRepairService'. The main editor window shows the following C# code:

```
1 using System;
2 using PX.Data;
3
4 namespace PhoneRepairShop
5 {
6     [Serializable]
7     [PXCacheName("RSSVRepairService")]
8     public class RSSVRepairService : IBqlTable
9     {
10         #region ServiceID
11         [PXDBIdentity(IsKey = true)]
12         public virtual int? ServiceID { get; set; }
13         public abstract class serviceID : PX.Data.BQL.BqlInt.Field<serviceID> { }
14         #endregion
15
16         #region ServiceCD
17         [PXDBString(15, IsKey = true, IsUnicode = true, InputMask = "")]
18         [PXUIField(DisplayName = "Service CD")]
19         public virtual string ServiceCD { get; set; }
20         public abstract class serviceCD : PX.Data.BQL.BqlString.Field<serviceCD> { }
21         #endregion
22
23         #region Description
24         [PXDBString(50, IsUnicode = true, InputMask = "")]
25         [PXUIField(DisplayName = "Description")]
26         public virtual string Description { get; set; }
27         public abstract class description : PX.Data.BQL.BqlString.Field<description> { }
28         #endregion
29
30         #region Active
31         [PXDBBool()]
32         [PXUIField(DisplayName = "Active")]
33         public virtual bool? Active { get; set; }
34         public abstract class active : PX.Data.BQL.BqlBool.Field<active> { }
35         #endregion
36
37         #region WalkInService
38
```

Field	Attribute
CreatedDateTime	[PXDBCreatedDateTime ()]
CreatedByID	[PXDBCreatedByID ()]
CreatedByScreenID	[PXDBCreatedByScreenID ()]
LastModifiedDateTime	[PXDBLastModifiedDateTime ()]
LastModifiedByID	[PXDBLastModifiedByID ()]
LastModifiedByScreenID	[PXDBLastModifiedByScreenID ()]
Tstamp	[PXDBTimestamp ()]
Noteid	[PXNote ()]

Figure: Specifying the PrimaryView property in the Screen Editor

The screenshot displays the Customization Project Editor interface for the 'PhoneRepairShop' project. The main window is titled 'Screen Editor: RS201000 (Repair Services)'. The left sidebar shows a tree view of the project structure, with 'RS201000' and 'DataSource: RSSVRepairServiceMaint' highlighted. The main area shows a table of properties for the selected DataSource. The 'PrimaryView' property is set to 'RepairService'.

Customization Project Editor Back Reload

File Publish Extension Library Source Control

PhoneRepairShop Screen Editor: RS201000 (Repair Services)

SCREENS
▶ **RS201000**
Data Access

CODE
▶ **DataSource: RSSVRepairServiceMaint**
Grid: RepairService
Dialogs

LAYOUT PROPERTIES ATTRIBUTES EVENTS ADD CONTROLS ADD DATA FIELDS

Override	Property	Value
▼	Base Properties	
<input type="checkbox"/>	ID	ds
<input type="checkbox"/>	PageLoadBehavior	
<input type="checkbox"/>	PrimaryView	RepairService
<input type="checkbox"/>	TypeName	PhoneRepairShop.RSSV...
<input type="checkbox"/>	Visible	True
▼	Ext Properties	
<input type="checkbox"/>	AttributesFound	
▶ <input type="checkbox"/>	ClientEvents	
<input type="checkbox"/>	EnableAttributes	
<input type="checkbox"/>	HeaderDescriptionField	
<input type="checkbox"/>	Height	
<input type="checkbox"/>	KeySeparatorChar	
<input type="checkbox"/>	OSType	

Figure: Specifying the DataMember property in the Screen Editor

The screenshot shows the Customization Project Editor interface. The top navigation bar includes 'File', 'Publish', 'Extension Library', and 'Source Control'. The main title is 'PhoneRepairShop' and the current screen is 'Screen Editor: RS201000 (Repair Services)'. The left sidebar shows a tree view of the project structure, with 'Grid: DetailsView' selected under 'CODE' > 'RSSVRepairServiceMaint'. The main area displays the 'LAYOUT PROPERTIES' tab for the selected grid. The table below shows the properties of the grid, with the 'DataMember' property highlighted in red and set to 'RepairService'.

Override	Property	Value
<input type="checkbox"/>	AllowPaging	
<input type="checkbox"/>	AutoAdjustColumns	
<input type="checkbox"/>	AutoSize	
<input type="checkbox"/>	BatchUpdate	
<input type="checkbox"/>	Caption	
<input type="checkbox"/>	CaptionVisible	
<input type="checkbox"/>	ContentLayout	
<input type="checkbox"/>	DataMember	RepairService
<input type="checkbox"/>	Height	150px
<input type="checkbox"/>	ID	grid
<input type="checkbox"/>	Layout	
<input type="checkbox"/>	MatrixMode	
<input type="checkbox"/>	Mode	
<input type="checkbox"/>	SkinID	Primary

The table or view used for binding against.

Lesson Summary

In this lesson, you have learned the concept of a data access class (DAC), and have generated and configured a DAC.

You have learned about the attributes that are required for DAC fields and have configured a view by using a fluent BQL query.

Lesson 1.5: Configure the Form

Learning Objectives

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

- Add columns to a form grid
- Configure the appearance of the columns in the grid

Figure: Columns to be added

The screenshot shows the Customization Project Editor interface. The left sidebar displays a tree view for 'PhoneRepairShop' with categories like SCREENS, CODE, and Files. The main area shows the 'Screen Editor: RS201000 (Repair Services)' with a 'Grid: RepairService' selected. The 'ADD DATA FIELDS' tab is active, showing a table of data fields for the 'Repair Service(RepairService)' data view. A red box highlights the 'CREATE CONTROLS' button and the first column of the table.

<input type="checkbox"/>	Used	Field Name	Control
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Active	CheckBox
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID (Created By)	Selector
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID_Creator_displayName (Created By)	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID_Creator_Username (Created By)	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID_description (Created By)	TextEdit
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Description	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID (Last Modified By)	Selector
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID_description (Last Modified By)	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID_Modifier_displayName (Last Mo	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID_Modifier_Username (Last Modif	TextEdit
<input checked="" type="checkbox"/>	<input type="checkbox"/>	PreliminaryCheck (Requires Preliminary Check)	CheckBox
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Prepayment (Requires Prepayment)	CheckBox
<input checked="" type="checkbox"/>	<input type="checkbox"/>	ServiceCD (Service ID)	MaskEdit
<input checked="" type="checkbox"/>	<input type="checkbox"/>	WalkInService (Walk-In Service)	CheckBox

Figure: Control tree with new columns

The screenshot displays the Customization Project Editor interface. The top navigation bar includes 'File', 'Publish', 'Extension Library', and 'Source Control'. The main window is titled 'PhoneRepairShop' and shows a 'Screen Editor: RS201000 (Repair Services)'. The left sidebar contains a tree view with categories like 'SCREENS', 'CODE', 'Files (2)', 'Reports', 'Dashboards', etc. The 'Grid: RepairService' is selected, and its fields are listed in a table. A red box highlights the 'Service ID', 'Description', 'Active', 'Walk-In Service', 'Requires Prepayment', and 'Requires Preliminary Check' fields. The right pane shows the 'ADD DATA FIELDS' tab with a table of data fields and their corresponding controls.

<input type="checkbox"/>	Used	Field Name	Control
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Active	CheckBox
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID (Created By)	Selector
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID_Creator_displayName (Created By)	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID_Creator_Username (Created By)	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	CreatedByID_description (Created By)	TextEdit
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Description	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID (Last Modified By)	Selector
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID_description (Last Modified By)	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID_Modifier_displayName (Last Mo	TextEdit
<input type="checkbox"/>	<input type="checkbox"/>	LastModifiedByID_Modifier_Username (Last Modif	TextEdit
<input type="checkbox"/>	<input checked="" type="checkbox"/>	PreliminaryCheck (Requires Preliminary Check)	CheckBox
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Prepayment (Requires Prepayment)	CheckBox
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ServiceCD (Service ID)	MaskEdit
<input type="checkbox"/>	<input checked="" type="checkbox"/>	WalkInService (Walk-In Service)	CheckBox

Figure: Property to define the Active column as a check box

Screen Editor: RS201000 (Repair Services)

EDIT ASPX PREVIEW CHANGES ...

The screenshot displays the 'LAYOUT PROPERTIES' tab in the Screen Editor. On the left, a tree view shows the hierarchy: DataSource: RSSVRepairServiceMaint > Grid: RepairService > Active (highlighted with a red box). The main area shows a table of properties for the selected 'Active' property.

Override	Property	Value
<input type="checkbox"/>	Base Properties	
<input type="checkbox"/>	CommitChanges	
<input type="checkbox"/>	DataField	Active
<input type="checkbox"/>	DisplayMode	
<input type="checkbox"/>	Type	CheckBox
<input type="checkbox"/>	Width	60
<input type="checkbox"/>	Ext Properties	
<input type="checkbox"/>	AllowCheckAll	
<input type="checkbox"/>	AllowFilter	
<input type="checkbox"/>	AllowMove	
<input type="checkbox"/>	AllowResize	
<input type="checkbox"/>	AllowShowHide	
<input type="checkbox"/>	AllowSort	
<input type="checkbox"/>	AutoGenerateOption	

The 'Type' property is highlighted with a red box, and its value is 'CheckBox'. A mouse cursor is visible over the 'Type' row. At the bottom of the editor, a note reads: 'The type of column display.'

Figure: The Repair Services form in the preview mode

CUSTOMIZATION TOOLS ▾

  	* Service ID	* Description	Active	Walk-In Service	Requires Prepayment	Requires Preliminary Check

⏪ ⏩ ⏴ ⏵

Figure: The Repair Services form

The screenshot displays the Acumatica interface for the 'Repair Services' form. The top navigation bar includes the Acumatica logo, a search bar, a refresh icon, the user 'Yogifon', the date '9/21/2021 7:52 AM', a help icon, and a user profile icon. The left sidebar contains navigation options: Favorites, Data Views, Phone Repair Shop (highlighted), Time and Expenses, Finance, Banking, Payables, Receivables, and Sales Orders. The main content area is titled 'Repair Services' and includes 'CUSTOMIZATION' and 'TOOLS' options. Below the title is a toolbar with icons for refresh, save, undo, add, delete, and print. A table with the following columns is shown: Service ID, Description, Active, Walk-In Service, Requires Prepayment, and Requires Preliminary Check. The table is currently empty, displaying a 'No records found.' message with a document icon and a close button. The bottom of the interface features a pagination bar with navigation arrows.

Service ID	Description	Active	Walk-In Service	Requires Prepayment	Requires Preliminary Check
No records found.					

Figure: The new row on the Repair Services form

Repair Services ☆

CUSTOMIZATION TOOLS ▾

↻ 📄 ↶ + × ⏪ ⏩

   * Service ID	* Description	Active	Walk-In Service	Requires Prepayment	Requires Preliminary Check
*   TESTID	Test Description	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Add the following rows to the table

Service ID	Description	Active	Walk-In Service	Requires Pre-payment	Requires Preliminary Check
BatteryReplace	Battery Replace-ment	Selected	Selected	Cleared	Cleared
LiquidDamage	Liquid Damage	Selected	Cleared	Selected	Selected
ScreenRepair	Screen Repair	Selected	Selected	Cleared	Cleared

Figure: New rows on the Repair Services form

Repair Services CUSTOMIZATION TOOLS ▾

🔄 📁 ↶ + × ⏪ ☒

		* Service ID	* Description	Active	Walk-In Service	Requires Prepayment	Requires Preliminary Check
	📁	BATTERYREPLACE	Battery Replacement	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	📁	LIQUIDDAMAGE	Liquid Damage	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
>	📁	SCREENREPAIR	Screen Repair	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Lesson Summary

In this lesson, you have learned how to configure the elements on a form by using the Screen Editor; you have also tested the created form.

Lesson 1.6: Add an Event Handler to the Walk-In Service Check Box

Learning Objectives

In this lesson, you will learn how to add an event handler for a check box.

Figure: The FieldUpdated event

Screen Editor: RS201000 (Repair Services)

EDIT ASPX PREVIEW CHANGES ...

Screen Editor: RS201000 (Repair Services)

EDIT ASPX PREVIEW CHANGES ...

LAYOUT PROPERTIES ATTRIBUTES **EVENTS** ADD CONTROLS ADD DATA FIELDS VIEW ASPX

DataSource: RSSVRepairServiceMaint

Grid: RepairService

- Service ID
- Description
- Active
- Walk-In Service**
- Requires Prepayment
- Requires Preliminary Check
- Levels
- Dialogs

Data Class: PhoneRepairShop.RSSVRepairService

Field Name: WalkInService

Business Logic: PhoneRepairShop.RSSVRepairServiceMaint.RepairServic

ADD HANDLER VIEW SOURCE

Event	Handled in Source	Customized
FieldSelecting	<input type="checkbox"/>	<input type="checkbox"/>
RowInserting	<input type="checkbox"/>	<input type="checkbox"/>
RowInserted	<input type="checkbox"/>	<input type="checkbox"/>
RowUpdating	<input type="checkbox"/>	<input type="checkbox"/>
RowUpdated	<input type="checkbox"/>	<input type="checkbox"/>
RowDeleting	<input type="checkbox"/>	<input type="checkbox"/>
RowDeleted	<input type="checkbox"/>	<input type="checkbox"/>
FieldDefaulting	<input type="checkbox"/>	<input type="checkbox"/>
FieldUpdating	<input type="checkbox"/>	<input type="checkbox"/>
FieldVerifying	<input type="checkbox"/>	<input type="checkbox"/>
ExceptionHandling	<input type="checkbox"/>	<input type="checkbox"/>
FieldUpdated	<input type="checkbox"/>	<input type="checkbox"/>
RowPersisting	<input type="checkbox"/>	<input type="checkbox"/>
RowPersisted	<input type="checkbox"/>	<input type="checkbox"/>

Figure: Clearing the Walk-In Service check box

Repair Services CUSTOMIZATION TOOLS ▾

🔄 📄 ↶ + × ⏪ 🗑️

			* Service ID	* Description	Active	Walk-In Service	Requires Prepayme	Requires Prelimina Check
	🔗	📄	BATTERYREPLACE	Battery Replacement	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	🔗	📄	LIQUIDDAMAGE	Liquid Damage	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
>	🔗	📄	SCREENREPAIR	Screen Repair	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Lesson Summary

In this lesson, you learned how to add and enable an event handler by using the Customization Project Editor. You have added code to the generated event handler and tested it on the Repair Services (RS201000) form to be sure it works as intended.

To add and enable an event handler, you have done the following:

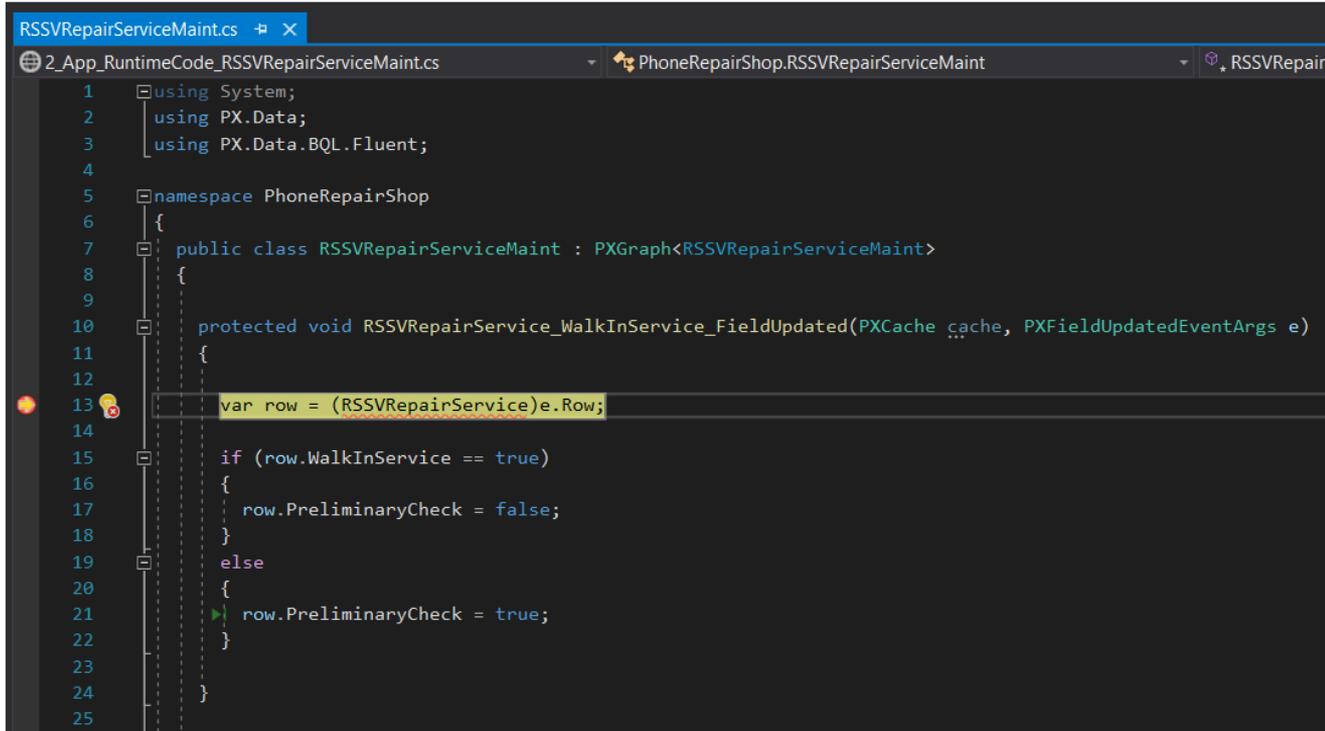
1. In the Screen Editor, configured the CommitChanges property of the control for which an event handler should work.
2. Generated code for the event handler by using the Screen Editor. You have added the FieldUpdated event, which is raised for each field of a record that is currently updated or inserted.
3. Added code to the event handler by using the Code Editor.

Lesson 1.7: Debug the Customization Code

Learning Objectives

In this lesson, you will learn how to debug the code of a customization project by using Visual Studio.

Figure: Breakpoint hit



```
1  using System;
2  using PX.Data;
3  using PX.Data.BQL.Fluent;
4
5  namespace PhoneRepairShop
6  {
7  public class RSSVRepairServiceMaint : PXGraph<RSSVRepairServiceMaint>
8  {
9
10     protected void RSSVRepairService_WalkInService_FieldUpdated(PXCache cache, PXFieldUpdatedEventArgs e)
11     {
12
13     var row = (RSSVRepairService)e.Row;
14
15     if (row.WalkInService == true)
16     {
17         row.PreliminaryCheck = false;
18     }
19     else
20     {
21     row.PreliminaryCheck = true;
22     }
23
24     }
25 }
```

Lesson Summary

In this lesson, you have learned how to debug customization code by using Visual Studio.

To debug customization code, you have completed the following steps:

1. Configured the web.config file of the Acumatica ERP instance.
2. In Visual Studio, added a breakpoint.
3. Attached the Visual Studio debugger to the w3wp.exe process.
4. Performed the debugging of the instance in Visual Studio.

A woman with blonde hair in a ponytail, wearing glasses and a grey t-shirt with an American flag patch, is working at a metalworking machine. She is using a tool to work on a piece of metal. The machine has a "DANGER" warning sign that says "KEEP FINGERS CLEAR OF MACHINE WHILE IN MOTION". There are several bins of metal shavings and tools around the machine. The scene is set in a factory or workshop.

Day 3

Lesson 1.8: Move the Customization Code to an Extension Library

Learning Objectives

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

- Create an extension library
- Open the created extension library in Visual Studio
- Build the created extension library in Visual Studio

Figure: Creation of the PhoneRepairShop_Code extension library

The screenshot displays the 'Customization Project Editor' interface. The top navigation bar includes 'File', 'Publish', 'Extension Library' (highlighted with a red box), and 'Source Control'. Below the navigation bar, the left sidebar shows a tree view with 'PhoneRepairShop' selected, containing sub-items like 'SCREENS', 'Data Access', and 'CODE'. A context menu is open over the 'Extension Library' tab, listing options: 'Create New', 'Bind to Existing', 'Open in Visual Studio', 'Show Active Extensions', and 'Analyze Referenced Assemblies'. The main workspace shows a table with columns 'Is New', 'Last Modified By', and 'Last Modified On'. A table row is visible with the following data:

Is New	Last Modified By	Last Modified On
<input checked="" type="checkbox"/>	admin admin	9/22/2021

Figure: Button to move the Code item to the extension library

The screenshot shows the Customization Project Editor interface. The title bar reads "Customization Project Editor" with "Back" and "Reload" buttons. Below the title bar is a menu bar with "File", "Publish", "Extension Library", and "Source Control". The main area is divided into a left sidebar and a right code editor. The sidebar shows a tree view for "PhoneRepairShop" with a sub-item "RSSVRepairServiceMaint" selected. The code editor displays the following C# code:

```
1 using System;
2 using PX.Data;
3 using PX.Data.BQL.Fluent;
4 |
5 namespace PhoneRepairShop
6 {
7     public class RSSVRepairServiceMaint : PXGraph<RSSVRepairServiceMaint>
8     {
9
10         protected void RSSVRepairService_WalkInService_FieldUpdated(PXCache cache, PXFieldUpdatedEventArgs e)
11         {
12             var row = (RSSVRepairService)e.Row;
13
14             if (row.WalkInService == true)
15             {
16                 row.PreliminaryCheck = false;
17             }
18             else
19             {
20                 row.PreliminaryCheck = true;
21             }
22         }
23     }
24
25     public SelectFrom<RSSVRepairService>.View RepairService;
26
27     public PXSave<RSSVRepairService> Save;
28     public PXCancel<RSSVRepairService> Cancel;
29 }
30
31
32
33
34
35
36
37
38
39
```

At the top of the code editor, there is a toolbar with buttons: "OPEN SCREEN", "VIEW SOURCE", "OVERRIDE METHOD", "NEW ACTION", and "MOVE TO EXTENSION LIB". The "MOVE TO EXTENSION LIB" button is highlighted with a red rectangular box.

Figure: The empty list of the Code page

The screenshot displays the 'Customization Project Editor' interface. At the top, there is a blue header bar with the title 'Customization Project Editor' and two links: 'Back' and 'Reload'. Below the header is a menu bar with 'File', 'Publish', 'Extension Library', and 'Source Control'. The main area is divided into a left sidebar and a central workspace. The sidebar shows a tree view with 'PhoneRepairShop' selected, and under it, 'CODE' is highlighted with a red box. The central workspace shows a table with the following columns: 'Object Name', 'Description', 'Last Modified By', and 'Last Modified On'. The table is empty, and a message box in the center states 'No records found.' with a document icon and a close button.

Customization Project Editor Back Reload

File Publish Extension Library Source Control

PhoneRepairShop CODE

SCREENS

- RS201000
- Data Access
- Code**
- Files (2)
- Generic Inquiries
- Reports
- Dashboards
- Site Map (1)
- Database Scripts (1)
- System Locales
- Import/Export Scenarios
- Shared Filters
- Access Rights
- Wikis
- Web Service Endpoints
- Analytical Reports
- Push Notifications
- Business Events
- Mobile Application
- User-Defined Fields
- Webhooks
- Connected Applications

Object Name	Description	Last Modified By	Last Modified On
-------------	-------------	------------------	------------------

No records found.

Figure: Opening of the solution from the Customization Project Editor

The screenshot displays the 'Customization Project Editor' interface. The top navigation bar includes 'Back' and 'Reload' buttons. Below the navigation bar, there are tabs for 'File', 'Publish', 'Extension Library', and 'Source Control'. The 'Extension Library' tab is active, and its dropdown menu is open, showing options: 'Create New', 'Bind to Existing', 'Open in Visual Studio' (highlighted), 'Show Active Extensions', and 'Analyze Referenced Assemblies'. The main content area shows a table with columns 'Description', 'Last Modified By', and 'Last Modified On'. The table is empty, displaying a message: 'No records found.' with a document icon and a close button. The left sidebar contains a tree view with categories like 'SCREENS', 'Data Access', 'Code', and various project components such as 'Files (2)', 'Generic Inquiries', 'Reports', 'Dashboards', 'Site Map (1)', 'Database Scripts (1)', 'System Locales', 'Import/Export Scenarios', 'Shared Filters', 'Access Rights', 'Wikis', 'Web Service Endpoints', 'Analytical Reports', 'Push Notifications', 'Business Events', 'Mobile Application', 'User-Defined Fields', 'Webhooks', and 'Connected Applications'.

Lesson Summary

In this lesson, you have learned how to create an extension library, move customization code to the extension library, and open the code items of a customization project in Visual Studio in order to start developing customization code in Visual Studio.

You have completed the following steps:

1. Created an extension library
2. Moved all code items from the customization project to the Visual Studio project
3. Opened the created project in Visual Studio
4. Built the project in Visual Studio

You have also included the extension library in the customization project as a File item.

Lesson 1.9: Add an Event Handler In Visual Studio

Learning Objectives

In this lesson, you will learn how to add an event handler in Visual Studio and use Acuminator to refactor existing code.

Figure: Refactoring of the event handler

```
16 0 references  
16 protected void RSSVRepairService_WalkInService_FieldUpdated(PXCache cache, PXFieldUpdatedEventArgs e)  
17  
18  
19  
20
```

Convert an event handler signature to the generic one. Please, be careful... ▸ Lines 9 to 11

Change signature... ▸

Wrap every parameter ▸

Unwrap and indent all parameters ▸

```
protected void RSSVRepairService_WalkInService_FieldUpdated(PXCache cache, PXFieldUpdatedEventArgs e)  
protected void _(Events.FieldUpdated<RSSVRepairService, RSSVRepairService.WalkInService> e)  
{  
}
```

Preview changes

Lesson Summary

In this lesson, you have added an event handler in Visual Studio and learned how to use Acuminator to refactor the existing code.

Part 2: Serviced Devices Maintenance Form

Lesson 2.1: Create a Graph and a DAC in Visual Studio

Learning Objectives

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

- Define a graph and DAC in Visual Studio
- Configure a selector for a field and a drop-down menu for a field

Lesson Summary

In this lesson, you have learned how to create a graph and a DAC in Visual Studio. Also, you learned how to configure a lookup control by using the PXSelector attribute and a drop-down list by using the PXStringList attribute.



Day 4

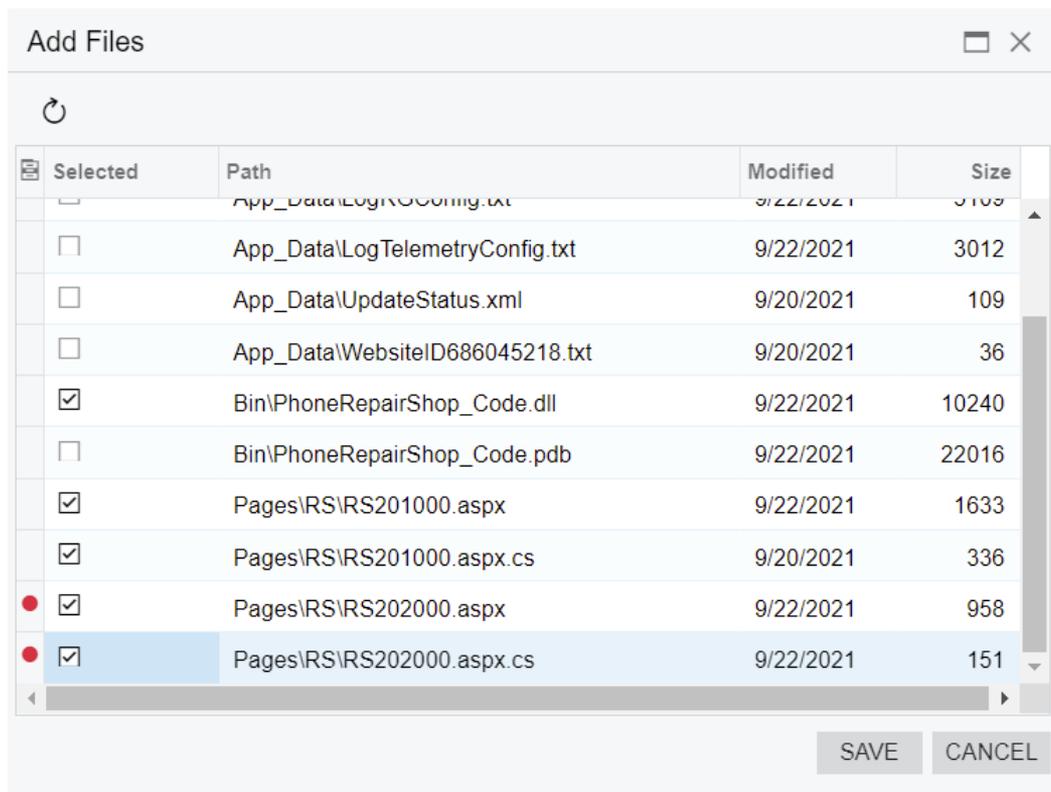
Lesson 2.2: Create an ASPX Page in Visual Studio

Learning Objectives

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

- Create an ASPX file
- Configure a data member for an Acumatica ERP form
- Add the ASPX and ASPX.CS files to the customization project

Figure: The Add Files dialog box



Lesson Summary

In this lesson, you have learned how to create an ASPX page in Visual Studio and configure a data member for it.

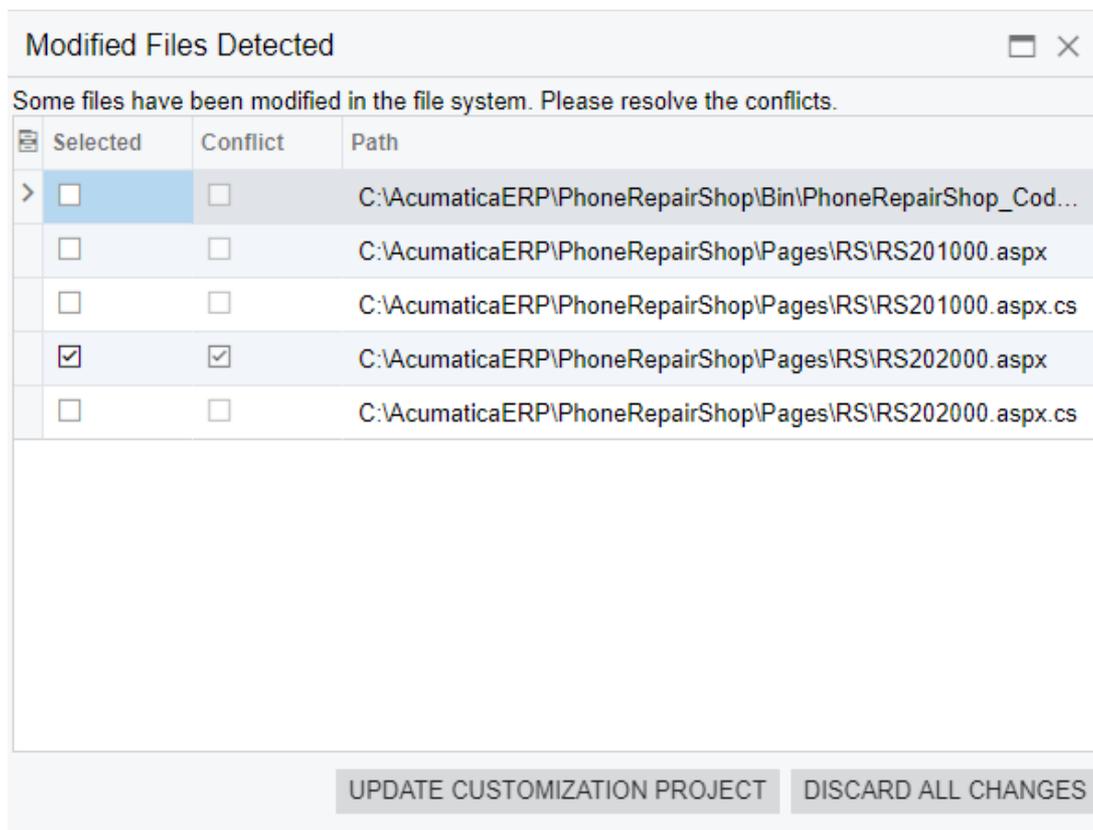
Also, you have added created files to the customization project.

Lesson 2.3: Configure a Form in Visual Studio

Learning Objectives

In this lesson, you will learn how to use Visual Studio to add controls and configure layout of a form.

Figure: The Modified Files Detected dialog box



Lesson Summary

In this lesson, you have learned how to configure input controls on a form in Visual Studio and organize the layout of the form.

To configure controls on the Serviced Devices (RS202000) form, you have used PXSelector, PXTextEdit, PXCheckBox, and PXDropDown tags in ASPX. To organize the layout of the form, you have added the PXLAYOUTRule tags. You have also updated the modified ASPX file in the customization project.

Lesson 2.4: Add the Form to the Site Map and Workspace

Learning Objectives

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

- Create a site map item for a custom form
- Save the created site map item to the customization project
- Add a form created in Visual Studio to the Screen Editor

Figure: The selected site map item

Site Map

RELOAD FROM DATABASE MANAGE SITE MAP

Object Name	Description	Last Modified By	Last Modified On
> Repair Services			

Add Site Map Item(s)

<input type="checkbox"/>	* Screen ID	Title	Last Modified On
<input checked="" type="checkbox"/>	RS.20.20.00	Serviced Devices	9/23/2021
<input type="checkbox"/>	IN.GI.00.16	Modified Stock Items	12/31/1899
<input type="checkbox"/>	IN.GI.00.01	Last Modified Date	12/31/1899
<input type="checkbox"/>	IN.GI.00.02	Item Availability Data	12/31/1899
<input type="checkbox"/>	AR.GI.00.15	Customer Contacts	12/31/1899
<input type="checkbox"/>	IN.GI.00.05	Stock Item Attachments	12/31/1899
<input type="checkbox"/>	RM.00.00.02	Profit and Loss Software & ...	12/31/1899
<input type="checkbox"/>	RM.00.00.01	Balance Sheet Software & ...	12/31/1899
<input type="checkbox"/>	RM.00.00.03	Profit and Loss Budget Perf...	12/31/1899
<input checked="" type="checkbox"/>	RS.20.10.00	Repair Services	9/20/2021

SAVE CANCEL

Figure: The Site Map page

The screenshot displays the 'Customization Project Editor' interface for the 'PhoneRepairShop' project. The main area is titled 'Site Map' and contains a table of site map objects. The table has four columns: 'Object Name', 'Description', 'Last Modified By', and 'Last Modified On'. Two objects are listed: 'Serviced Devices' and 'Repair Services'. The 'Serviced Devices' row is highlighted. Above the table, there are icons for refresh, save, undo, and a plus sign, along with the text 'RELOAD FROM DATABASE' and 'MANAGE SITE MAP'. On the left side, there is a navigation pane with a tree view showing various project components, with 'Site Map (2)' selected and highlighted in blue.

Object Name	Description	Last Modified By	Last Modified On
> Serviced Devices		admin admin	9/23/2021
Repair Services		admin admin	9/20/2021

Figure: Selecting the Serviced Devices form

The screenshot displays the Customization Project Editor interface. The main window is titled 'Customized Screens' and contains a table with the following data:

Screen ID	Title	Is New	Last Modified By	Last Modified On
RS.20.10.00	Repair Services	<input checked="" type="checkbox"/>	admin admin	9/22/2021

Below the table, a 'Customize Existing Screen' dialog box is open, showing a search field for 'Select Screen:'. A 'Select - Select Screen' dialog box is also open, displaying a search field with 'RS202000' entered. The 'Select - Select Screen' dialog box contains a table with the following data:

Title	Screen ID	Workspaces	Category
> Serviced Devices	RS 20.20.00	Phone Repa...	Configuration

Figure: Viewing the Serviced Devices form in the Screen Editor

Customization Project Editor [Back](#) [Reload](#)

File Publish Extension Library Source Control

PhoneRepairShop Screen Editor: RS202000 (Serviced Devices)

SCREENS

- RS201000
- RS202000**

Data Access

Code

- DataSource: **RSSVDeviceMaint**
- Form: ServDevices
- Dialogs

Files (5)

Generic Inquiries

Reports

Dashboards

Site Map (2)

Database Scripts (2)

System Locales

Import/Export Scenarios

Shared Filters

Access Rights

Wikis

Web Service Endpoints

Analytical Reports

Push Notifications

Business Events

Mobile Application

User-Defined Fields

Webhooks

Connected Applications

EDIT ASPX PREVIEW CHANGES ...

LAYOUT PROPERTIES ATTRIBUTES EVENTS ADD CONTROLS ADD DATA FIELDS VIEW ASPX

Override	Property	Value
<input type="checkbox"/>	Base Properties	
<input type="checkbox"/>	ID	ds
<input type="checkbox"/>	PageLoadBehavior	
<input type="checkbox"/>	PrimaryView	ServDevices
<input type="checkbox"/>	TypeName	PhoneRepairShop.RSSV...
<input type="checkbox"/>	Visible	True
<input type="checkbox"/>	Ext Properties	
<input type="checkbox"/>	AttributesFound	
<input type="checkbox"/>	ClientEvents	
<input type="checkbox"/>	EnableAttributes	
<input type="checkbox"/>	HeaderDescriptionField	
<input type="checkbox"/>	Height	
<input type="checkbox"/>	KeySeparatorChar	
<input type="checkbox"/>	QPToolBar	
<input type="checkbox"/>	ShowProcessInfo	

Figure: The Serviced Devices form

Serviced Devices

 NOTES FILES CUSTOMIZATION TOOLS ▾

     ▾    

* Device Code:

Description:

Active

Complexity:

Medium ▾

Figure: The lookup table

Serviced Devices

NOTES FILES CUSTOMIZATION TOOLS ▾

📄 ↶ + 🗑️ 📄 ▾ ⏪ < > ⏩

* Device Code: IPHONE6 🔍 Active

Description:

Select - Device Code 🗑️ ✕

SELECT ↻ ⏪ 🔍

Device Code	Active	Complexity
> IPHONE6	<input checked="" type="checkbox"/>	High
MOTORRAZR	<input checked="" type="checkbox"/>	Low
NOKIA3310	<input checked="" type="checkbox"/>	Low
SAMSUNGGS4	<input checked="" type="checkbox"/>	Medium

⏪ < > ⏩

Figure: The device properties

Serviced Devices NOTES FILES CUSTOMIZATION TOOLS ▾

* Device Code: 

Description:

Complexity:

Active

Lesson Summary

In this lesson, you have added the Serviced Devices (RS202000) form to a workspace, saved a site map item to the customization project, added the form to the Screen Editor, and tested the form.

Lesson 2.5: Create a Substitute Form

Learning Objectives

In this lesson, you will learn how to create a substitute form for a custom form and save it to the customization project.

Figure: The Serviced Devices generic inquiry

Generic Inquiry ★

NOTES FILES CUSTOMIZATION TOOLS ▾

📄 ↶ + 🗑️ 📄 ▾ ⏪ < > ⏩ **VIEW INQUIRY** ⋮

* Inquiry Title:

Site Map Title:

Workspace:

Category:

Screen ID:

Make Visible on the UI

Show Deleted Records

Expose via OData

Expose to the Mobile Application

Arrange Parameters in: columns

Select Top: records

Records per Page:

Export Top: Records

Attach Notes To:

TABLES RELATIONS PARAMETERS CONDITIONS GROUPING **🔔 SORT ORDER** RESULTS GRID ENTRY POINT NAVIGATION

Row Style:

🔄 + × ⏪ ⏩

📄	🔍	📄	Active	Object	Data Field	Schema Field	Width (px)	Style	Visible	Default Navigation	Navigate To
>	🔍	📄	<input checked="" type="checkbox"/>	ServicedDevices	DeviceCD	ServicedDevices.Devi...			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	🔍	📄	<input checked="" type="checkbox"/>	ServicedDevices	Description	ServicedDevices.Des...			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	🔍	📄	<input checked="" type="checkbox"/>	ServicedDevices	Active	ServicedDevices.Active			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	🔍	📄	<input checked="" type="checkbox"/>	ServicedDevices	AvgComplexityOfRepair	ServicedDevices.Avg...			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figure: Selecting the Serviced Device form

Generic Inquiry ★

NOTES FILES CUSTOMIZATION TOOLS ▾

📄 ↶ + 🗑️ 📄 ▾ ⏪ < > ⏩ **VIEW INQUIRY** ⋮

* Inquiry Title: 🔍

Site Map Title:

Workspace: 🔍

Category: 🔍

Screen ID:

Make Visible on the UI

Show Deleted Records

Expose via OData

Expose to the Mobile Application

Arrange Parameters in: columns

Select Top: records

Records per Page: Records

Export Top: Records

Attach Notes To:

TABLES RELATIONS PARAMETERS CONDITIONS GROUPING ⚠️ SORT ORDER RESULTS GRID **ENTRY POINT** NAVIGATION

ENTRY SCREEN SETTINGS

Entry Screen:

Replace Entry Screen with this Inquiry in Menu

Select - Entry Screen

SELECT ↻ |←| 🔍

Title	Screen ID	Workspaces	Category
> Serviced Devices	RS.20.20.00	Phone Repa...	Configuration

⏪ < > ⏩

Figure: The Add Generic Inquiries dialog box

Add Generic Inquiries ☐ ✕



 <input type="checkbox"/>	* Inquiry Title	Last Modified By	LastModified
<input type="checkbox"/>	Customer Contacts	admin ad...	12/31/1899
<input type="checkbox"/>	Item Availability Data	admin ad...	12/31/1899
<input type="checkbox"/>	Modified Stock Items	admin ad...	12/31/1899
<input checked="" type="checkbox"/>	Serviced Devices	admin ad...	9/23/2021
<input type="checkbox"/>	Stock Item Attachments	admin ad...	12/31/1899
<input type="checkbox"/>	Stock Items: Last Modified Date	admin ad...	12/31/1899

SAVE CANCEL

Figure: The Serviced Devices substitute form

Serviced Devices CUSTOMIZATION ▾ TOOLS ▾

🔄 ↶ + ✎ ⏪ ☒

Drag column header here to configure filter

		Device Code	Description	Active	Complexity
>	🔗 ☒	IPHONE6	iPhone 6	<input checked="" type="checkbox"/>	High
	🔗 ☒	MOTORRAZR	Motorola RAZR V3	<input type="checkbox"/>	Low
	🔗 ☒	NOKIA3310	Nokia 3310	<input checked="" type="checkbox"/>	Low
	🔗 ☒	SAMSUNGGS4	Samsung Galaxy S4	<input checked="" type="checkbox"/>	Medium

1-4 of 4 records ⏪ < > ⏩

Lesson Summary

In this lesson, you have learned about substitute forms and have created one for the Serviced Devices (RS202000)

form.

To configure the substitute form, you have completed the following steps:

1. Loaded a predefined generic inquiry on the Generic Inquiry (SM208000) form.
2. Configured the properties of the generic inquiry on the **Entry Point** tab of this form.
3. Saved the generic inquiry to the customization project.

Course Summary

In this course, you have learned the basic concepts of developing a custom form in Acumatica ERP using the Customization Project Editor and Visual Studio. Now you know the following:

- How to create a maintenance form
- How to access and manipulate data by using DACs and graph members
- How to implement business logic on a form by using event handlers
- How to configure controls that display data on a form and adjust the layout of controls on the form

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.



Thank you

**Vidhyalakshmi
Hariharasubramanian**