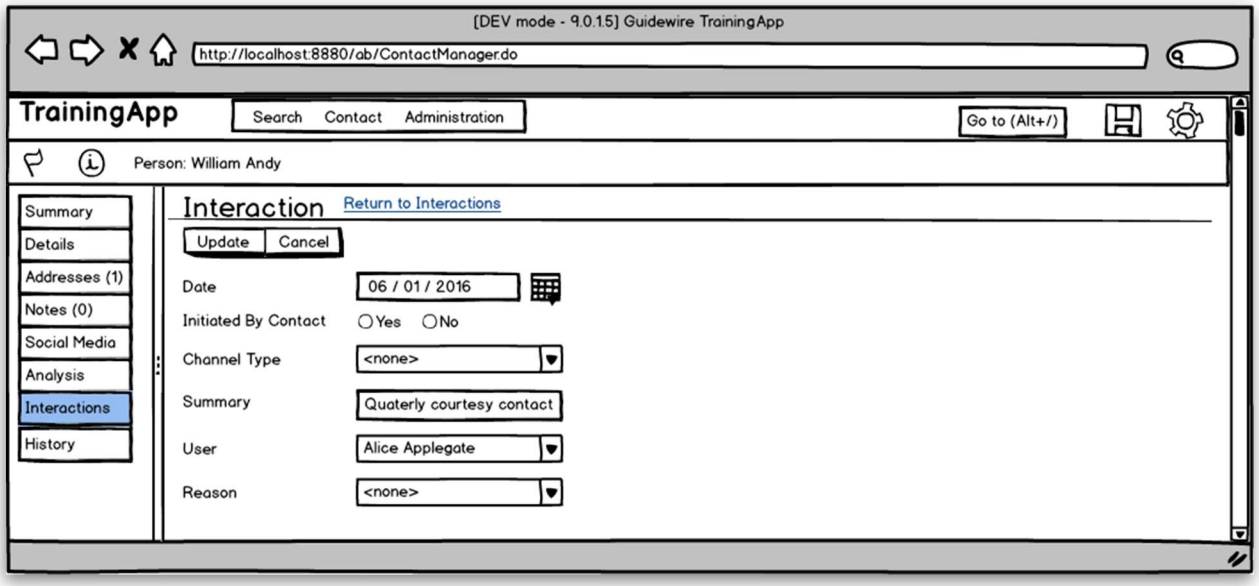
Lesson 5 Popups: View and Edit

"Currently, users can add, remove, and edit contact interactions on the Interactions page in the Interactions list view panel. However, the editable list view panel on the page only exposes some of the Interaction entity fields. Configure a popup that will allow users to view and edit all the fields of a contact interaction." – Insurance company business analysts



As a configuration developer, you want to be able to create, configure and open popup locations so that the users can see and/or edit more information about an object. In this lab, you will first create the Interaction Popup. The popup will allow users to edit and view an interaction. Next, you will modify the InteractionsLV. You will configure a cell widget in the list view panel to navigate to the popup.

5.1 Prerequisites

You must first complete the following previous lesson(s):

1. **Creating New Entities**
2. **List Views**
3. **Editable List Views**
4. **Typelists**

For this lab, use TrainingApp, Guidewire Studio, and a supported web browser. `http://localhost:8880/ab/ContactManager.do` is the default URL for TrainingApp. To view, edit, and delete various contacts, log in to TrainingApp as Alice Applegate. The login/password for Alice Applegate is `aapplegate/gw`.

5.2 Lab: Popup locations

1. **In Project View, create the InteractionPopup in the traininglabs PCF folder**

   a) Create the traininglabs PCF folder if it doesn't exist

   b) For the Popup, set the startInEditMode and canEdit properties to true

   c) To the Popup, add an inline Screen and an inline Detail View

   d) To the inline Detail View, add editable atomic widgets for the following Interaction_Ext fields:

   - Interaction Date
   - Initiated By Contact
   - Channel Type
   - Summary
   - Associated User
   - Interaction Reason

2. **Modify the InteractionsLV.pcf**

   a) Modify the cell widget for the Summary field to be a navigable link to the editable Interaction Popup. The Popup should display the entity from the row that the user clicked on.

Note: Use your best judgment for field ordering, display keys, and input widget selection. Review the Atomic Widgets lesson for more details if needed.

Guidewire API

The `startInEditMode` property

If this property is set to true then the location will open in edit mode allowing the user to modify the data immediately. If the property is set to false then the location will open in read-only mode and the user will need to click on the edit button to edit the details. This property could be also set dynamically, e.g. based on an expression, result of a function call or a parameter received from the parent location in case of Popups.

Guidewire API

The `canEdit` property

This property determines if the location (and any containers inside it) can or cannot be in edit mode. If the startInEditMode property is set to true or the location has a Toolbar and Edit Buttons widget then the canEdit property must be set to true. The default value is false.

Hint

Referencing the object associated with a row

When you click on a cell in a row, you want to open the popup and pass in the object that is associated with that specific row. Remember, the variable defined in the elementName attribute of the Row Iterator is always the variable that will store that associated object. As a configuration developer, you don’t have to worry about indexes. You can just simply reference the variable and the PCF framework will determine which object from the array is associated with that row the user clicked on.

Review

Navigating to a location

Review the Introduction to Locations lesson if you need tips on how to define navigation in Guidewire applications.

5.2.1 Verification

Activity

Verify the work you have done

1. **Log in to TrainingApp as Alice Applegate**

   a) Use the appropriate shortcut to reload the PCF changes

   b) Edit the Interactions page for William Andy

   c) Open the William Andy contact

   d) In the sidebar menu, click Interactions

   e) If you previously did not create an interaction, create at least one interaction now in the list view panel

   f) Click **Edit** then click **Add**

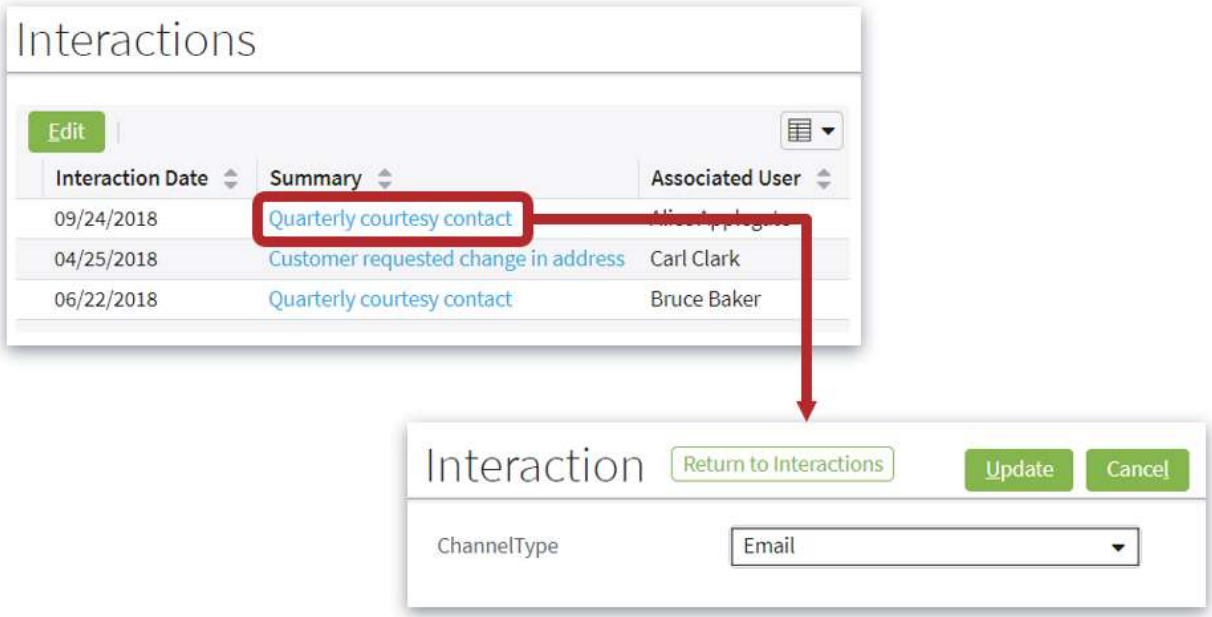
   g) Enter a date, a reason, and select a user

   h) Click **Update**

   i) For an interaction row, click an interaction reason link

   j) Edit all editable fields in the Interaction Popup

   k) Click **Update**

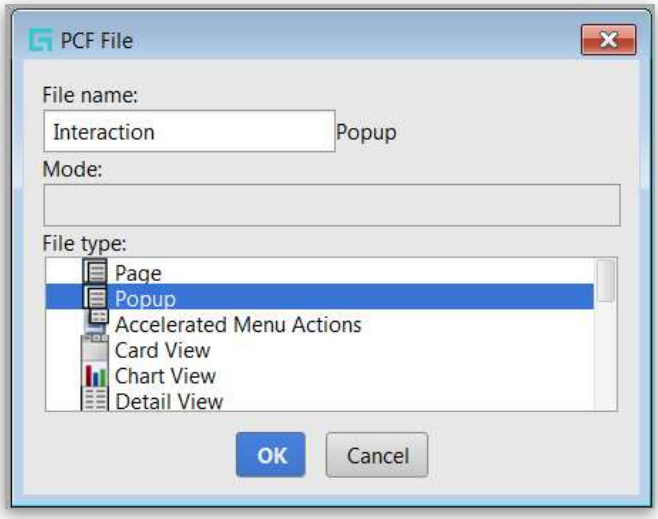

Interactions
Edit
Interaction Date
Summary
Associated User
09/24/2018
Quarterly courtesy contact
04/25/2018
Customer requested change in address
Carl Clark
06/22/2018
Quarterly courtesy contact
Bruce Baker
Interaction
Return to Interactions
Update
Cancel
ChannelType
Email

5.3 Lab Solution: Popup locations

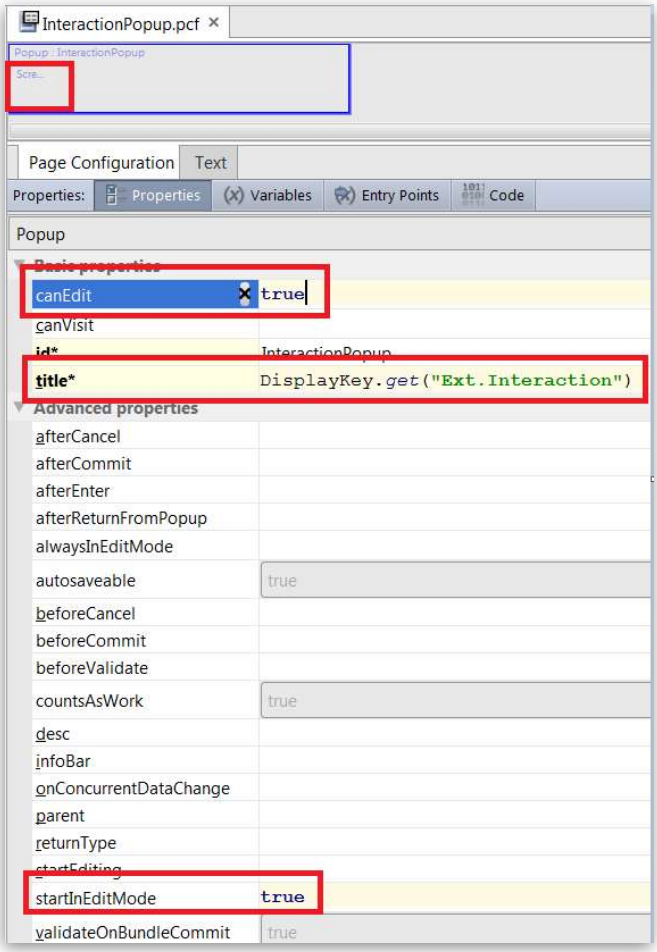
Solution

Exact details on how to complete the lab.

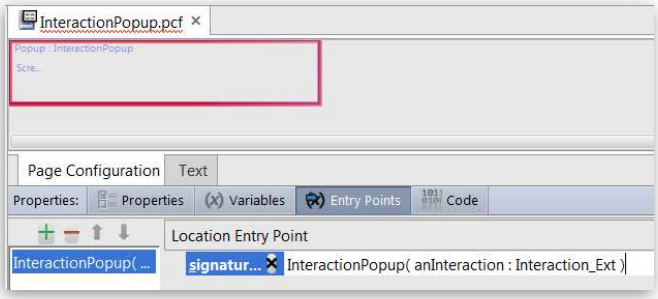
1. **Create the traininglabs PCF folder if it doesn't exist**
2. **Create the InteractionPopup**



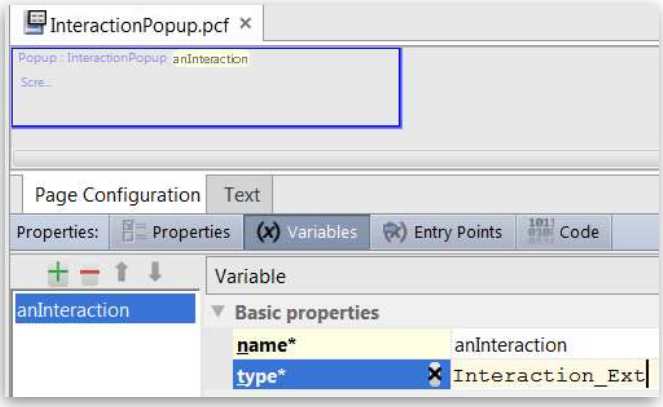
3. **Add an inline Screen container and set the following properties: `startInEditMode`, `canEdit`, `title`**



4. **Specify an entry point with a single Interaction_Ext parameter**



5. **Specify a variable for the parameter (same name and same type)**



6. **Add a toolbar and edit buttons**

   a) From the toolbox, drag and drop a **Toolbar** into the screen you just added.

   b) Drag and drop an **Edit Buttons** widget into the toolbar.

7. **Add a detail view**

   a) Drag and drop a **Detail View** below the **Toolbar** widget

   b) The widget is colored red because the Detail View must contain an Input Column or a Table Layout. You'll fix this in the next step.

8. **Add an Input Column**

   a) Drag and drop an **Input Column** widget into the **Detail View**.

   b) Notice that the Detail View is no longer red.

9. **Add an input for the Interaction Date field**

    a) Drag and drop a **Date Input** into the **Input Column**.

    b) Set the editable property to **true**.

    c) Set the id property to **InteractionDate**.

    d) Set the label property to **DisplayKey.get("Ext.InteractionDate")**. This display key was created an earlier lab lab. If it does not exist, create it at this point.

    e) Set the value property to **anInteraction.InteractionDate**.

10. **Add an input for the Initiated by Contact field**

    a) Drag and drop a **Boolean Radio Button** Input into the **Input Column** below the Interaction Date input.

    b) Set the editable property to **true**.

    c) Set the id property to **InitiatedByContact**.

    d) Set the label property to **DisplayKey.get("Ext. InitiatedByContact ")**. This is a new display key, so create it using Studio.

    e) Set the value property to **anInteraction.InitiatedByContact**.

11. **Add an input for the Channel Type field**

    a) Drag and drop a **TypeKey Input** into the **Input Column** below the Initiated By Contact input.

    b) Set the editable property to **true**.

    c) Set the id property to **ChannelType**.

    d) Set the label property to **DisplayKey.get("Ext. ChannelType")**.

    e) Set the value property to **anInteraction.Channel**.

    f) Notice that the valueRange and valueType fields were automatically entered.

12. **Add an input for the Summary field**

    a) Drag and drop a **Text Input** into the **Input Column** below the Channel Type input.

    b) Set the editable property to **true**.

    c) Set the id property to **Summary**.

    d) Set the label property to **DisplayKey.get("Ext. Summary")**.

    e) Set the value property to **anInteraction.Summary**.

13. **Add an input for the Associated User field**

    a) Drag and drop an **Alt User Input** into the **Input Column** below the Summary input.

    b) Set the editable property to **true**.

    c) Set the id property to **AssociatedUser**.

d) Set the label property to **DisplayKey.get("Ext.AssociatedUser")**.

e) Set the value property to **anInteraction.AssociatedUser**.

14. **Add an input for the Interaction Reason field**

a) Drag and drop a **Range Input** into the **Input Column** below the Associated User input.

b) Set the editable property to **true**.

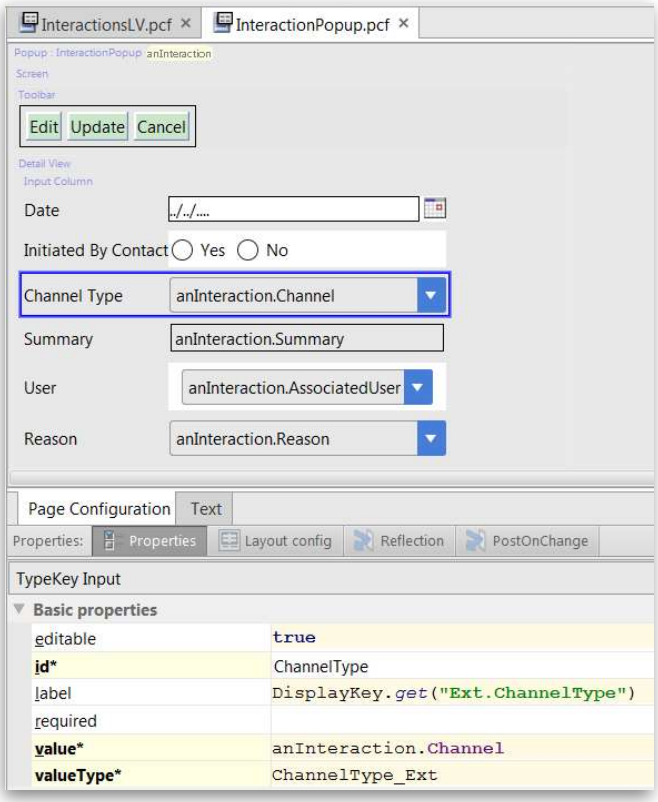
c) Set the id property to **ReasonType**.

d) Set the label property to DisplayKey.get("Ext.ReasonType").

e) Set the value property to **anInteraction.Reason**.

f) Notice that the valueRange and valueType fields were automatically entered.

15. **The completed popup should look like this screenshot:**



16. **Configure the action property of the Summary cell to open the InteractionPopup**