Education

InsuranceSuite Fundamentals: Kickstart

Student Workbook

Table of contents

Introduction

Welcome to the Guidewire InsuranceSuite™ Fundamentals Kickstart course.

The Student Workbook you will lead you through various course module exercises and additional information. The module numbers correspond to the module numbers in your training. As time allows, complete the assigned exercises to the best of your ability.

Lesson 1 Introduction to Guidewire Configuration

As a new configuration developer, you will explore the development tools and learn about the basic functionality of TrainingApp so that you can easily extend the functionality later.

In this lab, you will start Guidewire Studio, start a development instance of TrainingApp from Studio, and then log in to TrainingApp. In TrainingApp, you will search for contacts and create new contacts. You will end this lab by exiting Guidewire Studio and stopping the application.

1.1 Prerequisites

For this exercise, 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.**

1.2 Lab: Working with Guidewire Studio

In this exercise, you will open Guidewire Studio. You will then start and stop the TrainingApp server from Guidewire Studio.

1. **Open Guidewire Studio**

   a) Open a command window from the **C:\GW10\TrainingApp** folder.

   b) In the command window, start Guidewire Studio for TrainingApp. Note: Alternatively, use the Start TrainingApp Studio shortcut.

2. **Start TrainingApp in debug mode**

   a) Use the main menu, toolbar, or keystroke to start TrainingApp in debug mode.

3. **Verify that you are running TrainingApp**

   a) Verify that the Debug console is open.

   b) Verify that the Console tab shows `***** ContactManager ready *****`

Hint

Opening a command from the command window

To open a new command window from a folder, hold down the Shift key on the keyboard and use your mouse to right-click on the TrainingApp folder. From the context menu select the Open command window here option.

Hint

Running tasks from the command window

If you cannot recall the exact name of the task you want to execute you can always run the following command:

```
C:\GW10\TrainingApp\gwb tasks
```

This will display the documentation of all the available tasks.

Write it down

What are the two ways that you can run TrainingApp from Guidewire Studio?

1.3 Lab: Explore TrainingApp

In this exercise, you will log in to TrainingApp as Alice Applegate. As Alice Applegate, you will search for contacts and create new contacts. Then, you will log out of TrainingApp. Finally, you will stop TrainingApp from Guidewire Studio.

1. **Log in to TrainingApp as Alice Applegate.**
2. **Search for contacts with the last name Smith and review the search results.**

Write it down

How many contacts have the last name Smith?

Describe the contact types with the last name Smith.

1.4 Lab: Create contacts

In this part of the exercise, you will create new contacts in TrainingApp.

Important!

Select the right contact type

From the Actions menu you can create many different types of contacts in TrainingApp. When creating the contacts in the next part, check all the available sub-menus and select the most specific type.

1. Create a contact of the type Law Firm

a) Specify the following details:

Field Name	Value
Name	Celebrity Law Office
Tax ID (EIN)	11-1234567
City	Hollywood
State	California

2. Create a contact of the type Autobody Repair Shop

a) Specify the following details:

Field Name	Value
Name	Celebrity Collision Repair
Tax ID (EIN)	22-1234567
City	Los Angeles
State	California

3. **Search for contacts with the name Celebrity**

   a) Review the search results.

   b) View the details for each contact in the results.

Best practice

Regenerate the data dictionary

Regenerating the data dictionary after making data model changes is a best practice because it creates current documentation for the data model. In addition, the build process for the data dictionary can identify issues beyond schema validation such as referential integrity in the data model.

Write it down

On the Details screen for the Law Firm, in the Additional Info section, review the available fields. Next, review the available fields in the Additional Info section for the Autobody Repair Shop. Which fields differ between the Law Firm and the Autobody Repair Shop?

1. **Log out of TrainingApp**

   a) In the Settings menu, select Log Out Alice Applegate.

2. **Stop TrainingApp from Guidewire Studio**

   a) Stop the Debug ‘Server’ process.

Write it down

After stopping the server from Guidewire Studio, are you able to navigate in the TrainingApp web application? What is the browser error?

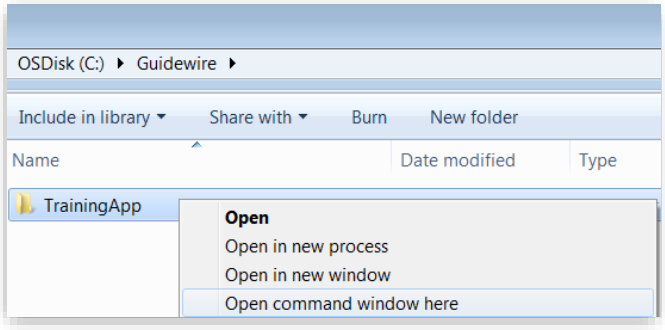
Stop

1.5 Lab Solution: Working with Guidewire Studio

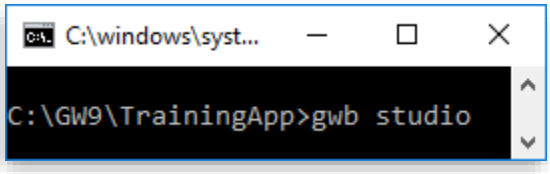
Solution

1. Open Guidewire Studio

a) To open Guidewire Studio, **SHIFT** + Right click on **C:\GW10\TrainingApp** and select **Open command window here** from the context menu.

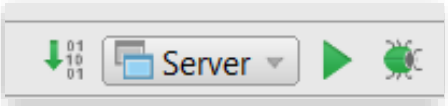


b) Execute the following Gradle task: `gwb studio`

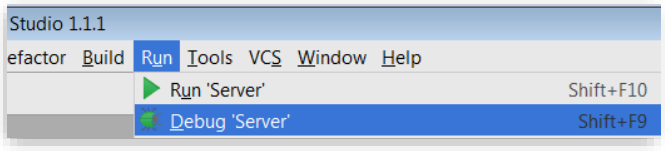


2. Start TrainingApp in debug mode

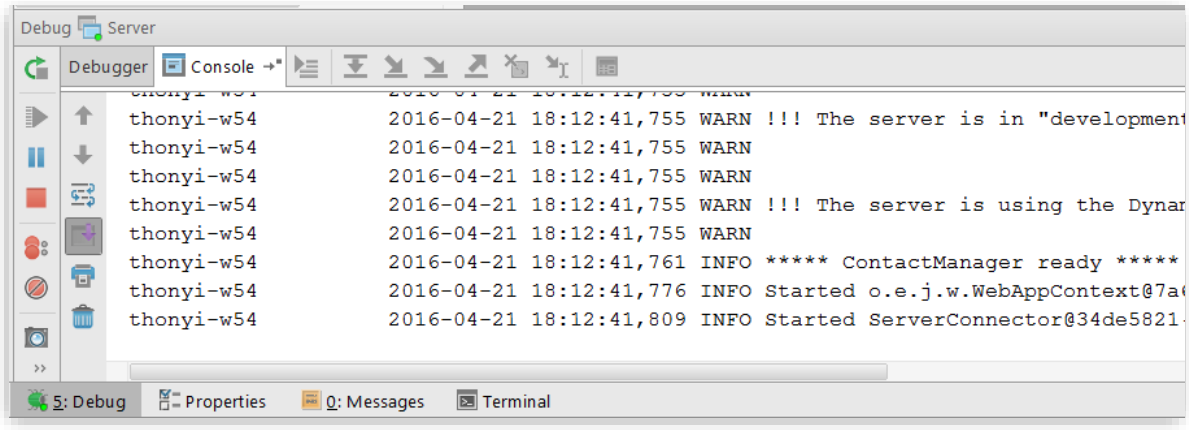
a) To start TrainingApp in debug mode, either (1) click on the Debug icon  or (2) from the menu select Run → Debug 'Server'



OR



3. **To verify that you are running TrainingApp, verify that the Debug Console tab shows `***** ContactManager ready *****`**



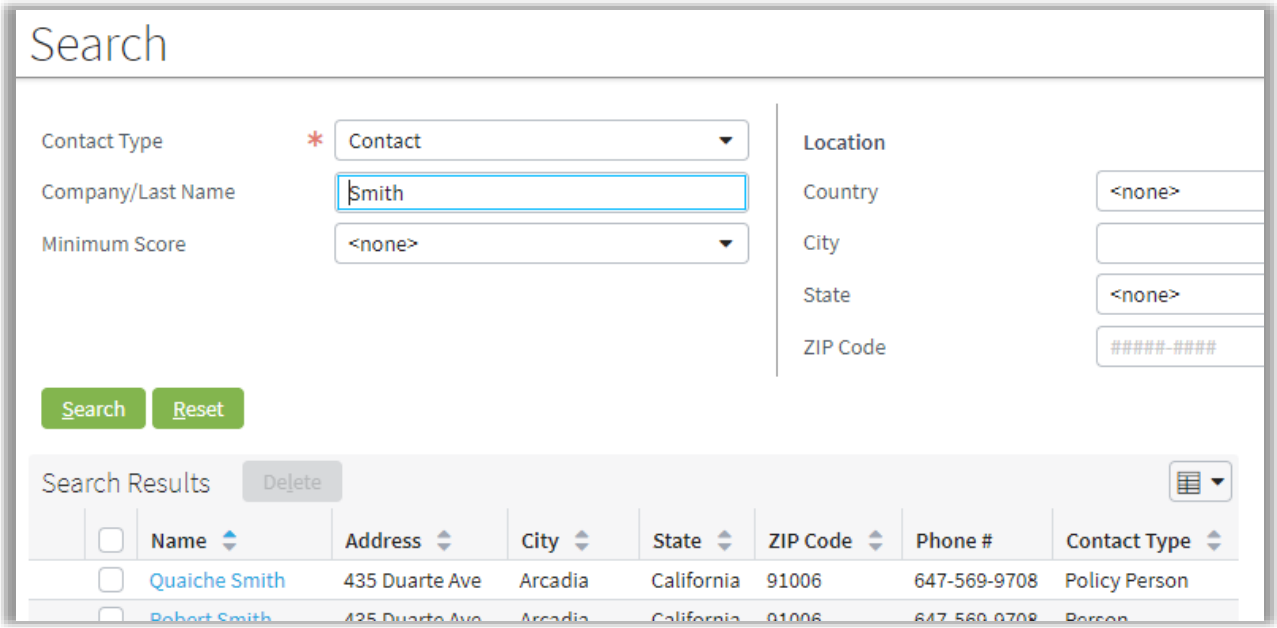
What are the two ways that you can run TrainingApp from Guidewire Studio?

From the Run menu, select Run 'Server' or select Debug 'Server'. Alternatively click on the Play or Debug buttons.

1.6 Lab Solution: Explore TrainingApp

Solution

1. **Log in to TrainingApp**
   a) Open the browser and go to http://localhost:8880/ab
   b) Log in as Alice Applegate. (username: aapplegate password: gw)
2. **Search for contacts with the last name Smith**



Write it down

1. **How many contacts have the last name Smith?**

2

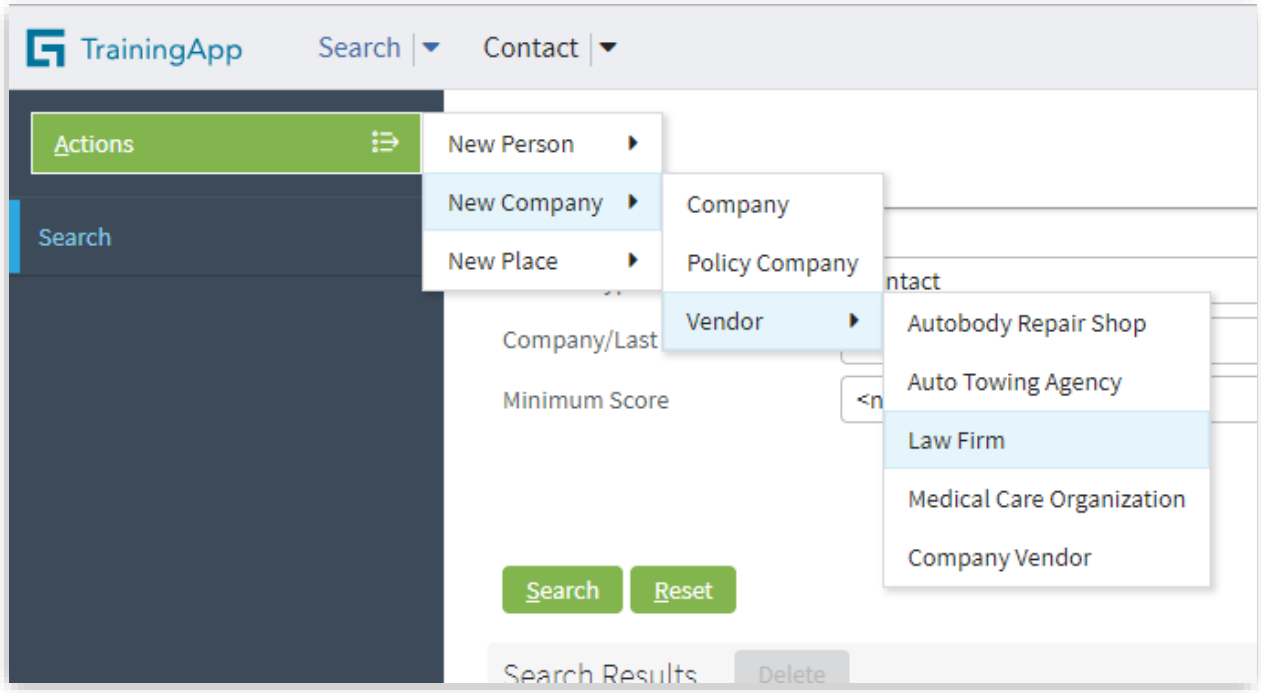
2. **Describe the contact types with the last name Smith.**

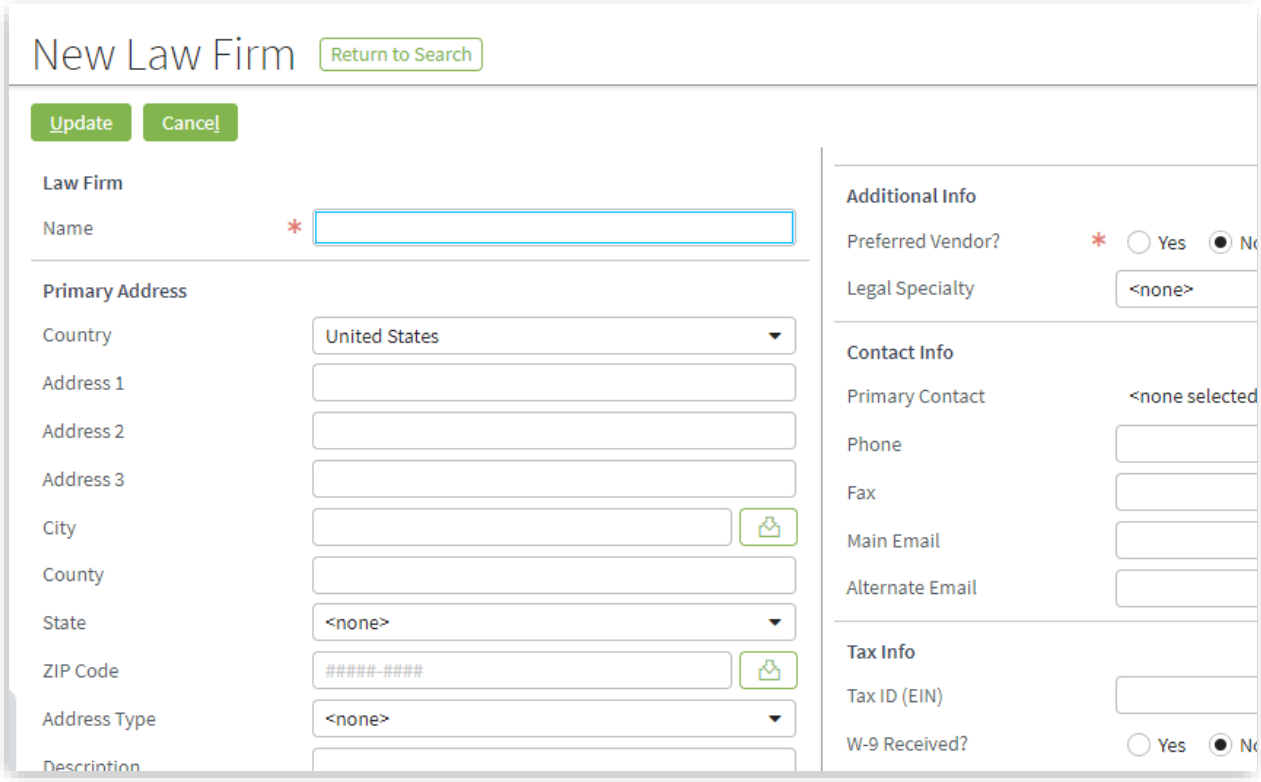
Robert Smith is a Person whereas Quaiche Smith is a Policy Person.

1.7 Lab Solution: Create contacts

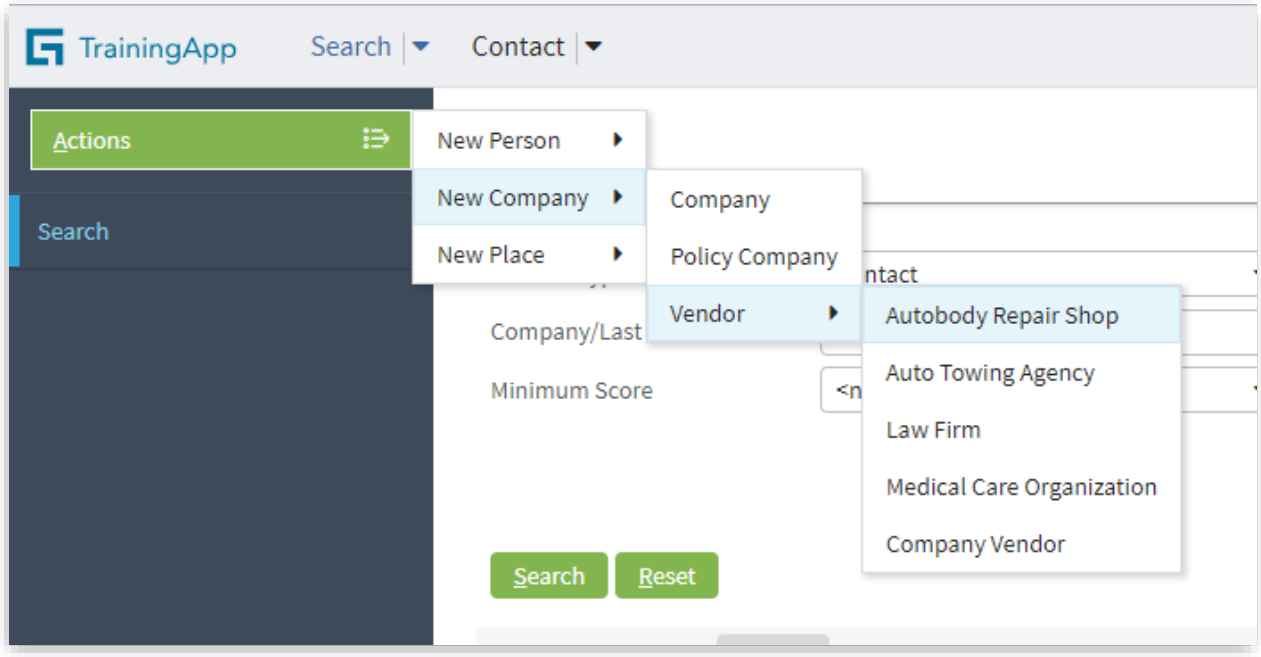
Solution

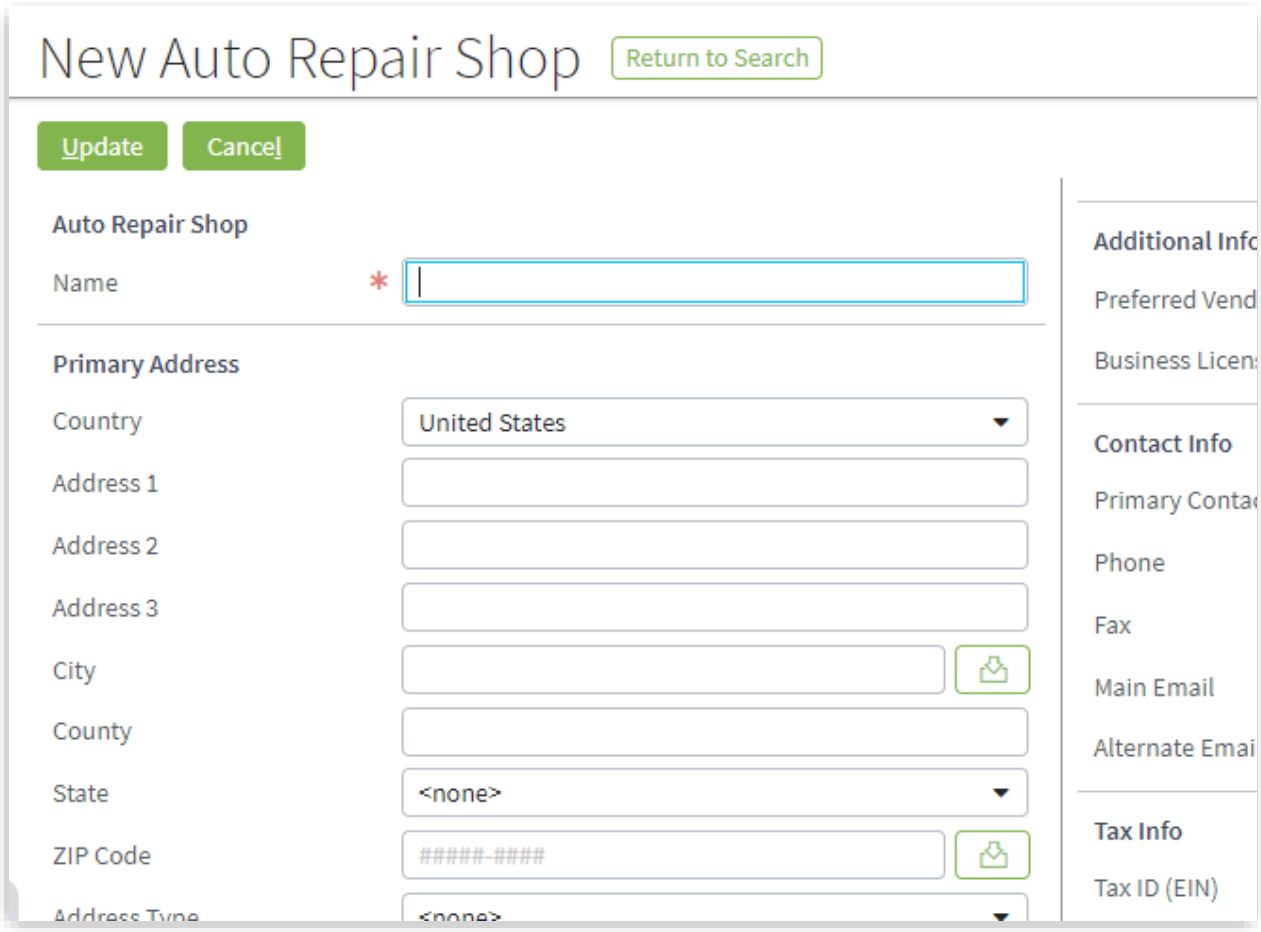
1. **Create a contact of the type Law Firm**



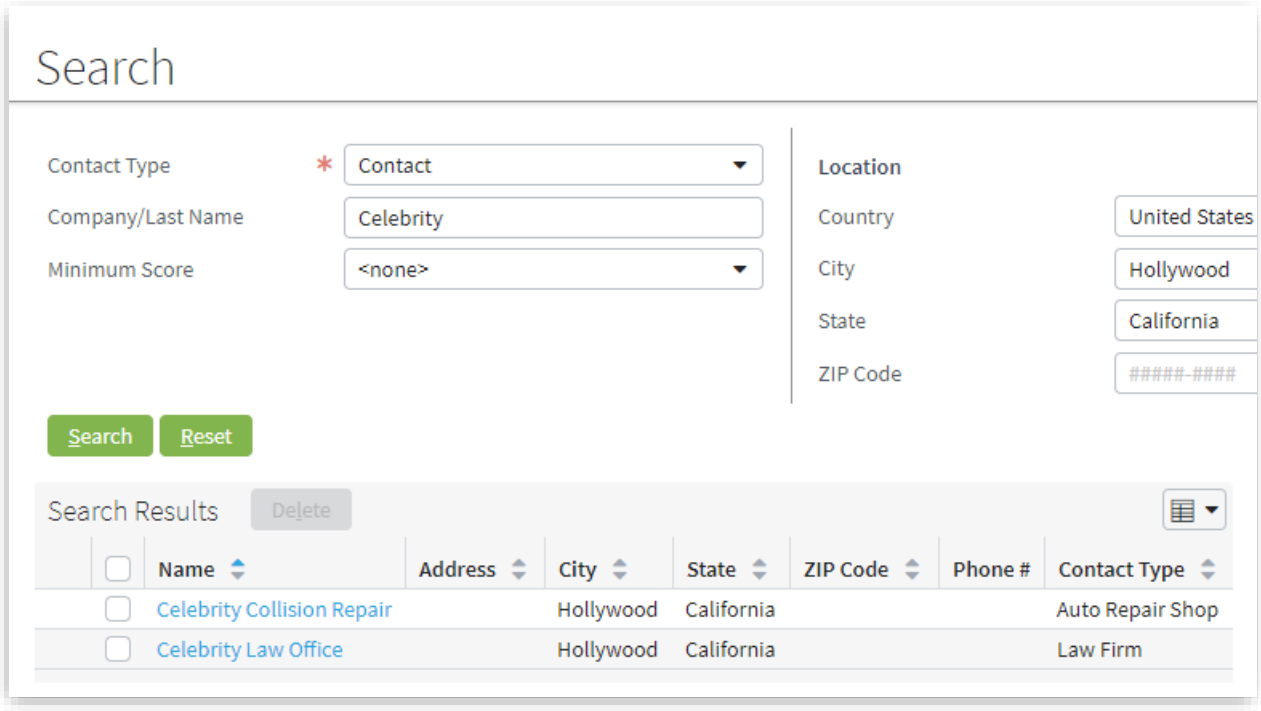


2. Create a contact of the type Autobody Repair Shop


TrainingApp
Search
Contact
Actions
Search
New Person
New Company
New Place
Company
Policy Company
Vendor
Autobody Repair Shop
Auto Towing Agency
Law Firm
Medical Care Organization
Company Vendor
Company/Last
Minimum Score
Search
Reset


New Auto Repair Shop
Return to Search
Update
Cancel
Auto Repair Shop
Name
Primary Address
Country
United States
Address 1
Address 2
Address 3
City
County
State
<none>
ZIP Code
#####-####
Additional Info
Preferred Vend
Business Licen
Contact Info
Primary Contac
Phone
Fax
Main Email
Alternate Emai
Tax Info
Tax ID (EIN)

3. **Search for contacts with the name Celebrity**

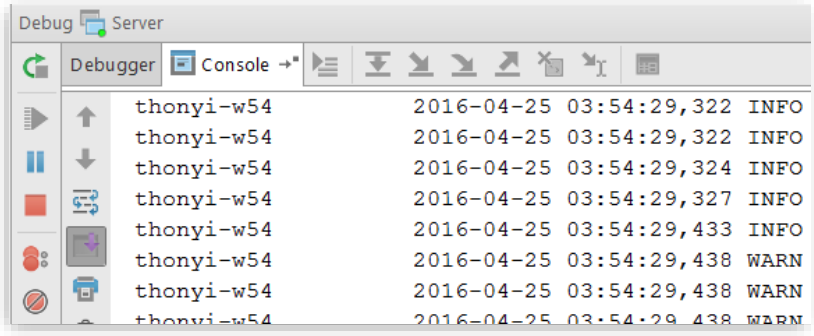


Write it down

1. **On the Details screen for the Law Firm, in the Additional Info section, review the available fields. Next, review the available fields in the Additional Info section for the Autobody Repair Shop. Which fields differ between the Law Firm and the Autobody Repair Shop?**

The Law Firm has a Specialty field whereas the Autobody Repair Shop has a License field.

2. **Log out of TrainingApp**
   a) Settings → Log out as Alice Applegate
3. **Stop TrainingApp from Guidewire Studio**
   a) Click on the Stop button in the sidebar of the Debug Tool window



4. **After stopping the server from Guidewire Studio, are you able to navigate in the TrainingApp web application? What is the browser error?**

It is not possible to continue navigating the TrainingApp web application. The browser shows an error that says the webpage is not available.

Lesson 2 Introduction to the Data Model

2.1 Prerequisites

For this exercise, use TrainingApp.

2.2 Lab: Explore the Data Dictionary

As a new configuration developer, you will explore the Data Dictionary and identify information about a given application's data model.

In this lab, you will review basic concepts introduced in the *Introduction to Data Model* lesson by using the Data Dictionary.

1. **Build the Data Dictionary from the command window**
   a) Open a Command window from the C:\GW10\TrainingApp folder.
   b) In the Command window, enter the command to build the Data Dictionary.
2. **Open the Data Dictionary**
   a) In Windows Explorer, navigate to the Data Dictionary.
   b) Open the Data Dictionary using a web browser.
   c) Create a bookmark in your browser so you can easily come back later.

Tools

genDataDictionary

Open a command window from the `root` directory of the Guidewire application. In the command window, enter `gwb genDataDictionary`

Find

Open the Data Dictionary

The Data Dictionary documents the entities and typelists in the Guidewire application including both base application and customer extensions. To open the Data Dictionary, open `…\build\dictionary\data\index.html` in a recommended web browser.

Write it down

Hint

You can make use of the browser's text search functionality to assist in finding the answers for many of these questions

1. **Describe the purpose of the data dictionary.**

2. **For the ABContact entity, describe the datatype for the EmailAddress1 field.**

3. **ABContact has a Tax ID field. Is the Tax IDs field used to store a social security number (SSN), an employer ID number (EIN), or both?**

4. **Name one typekey field in ABContact where the field name is identical to the name of the related typelist.**

5. **Name one typekey field in ABContact where the field name is NOT identical to the name of the related typelist.**

6. **Name one field on ABContact which is not stored in the database. What type of field is this?**

7. **Which has more arrays: the Group entity or the Role entity?**

8. **How many entities have a field named Organization?**

Hint

Use the browser's text search functionality

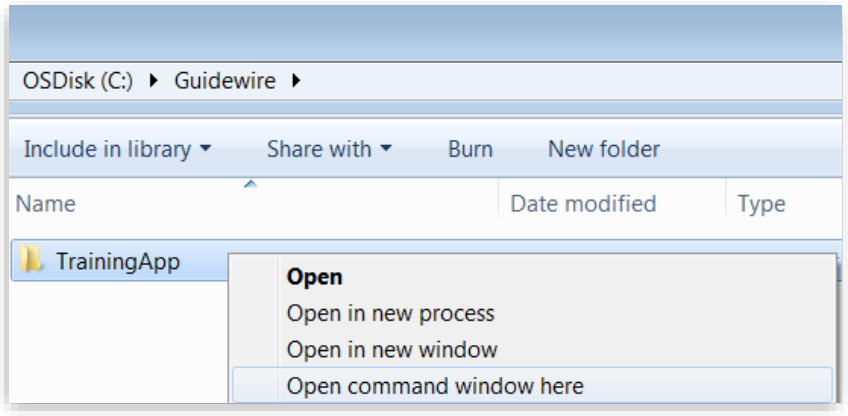
The easiest way to answer questions 4, 5 and 6 is by using the browser's Find functionality. Use the **CTRL + F** keystroke in the browser and enter **typekey** or **virtual property** or **derived property** as your search keyword and all the matches will be highlighted in the page. This will help you identify possible answers to some of the following questions.

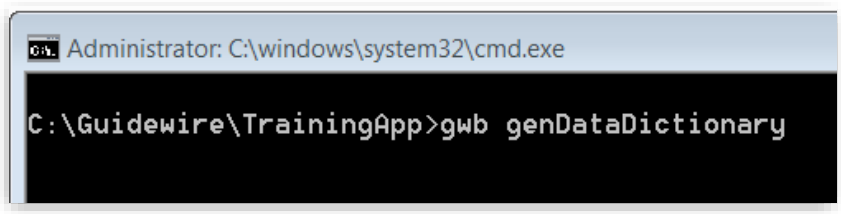
Stop

2.3 Lab Solution: Explore the Data Dictionary

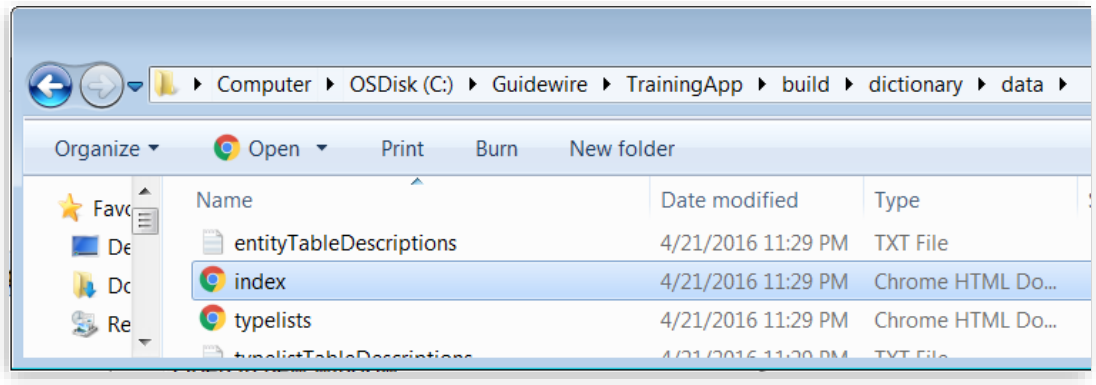
Solution

1. Build the Data Dictionary from the command window





2. Open the Data Dictionary



Write it down

1. Describe the purpose of the data dictionary.

The Data Dictionary is a set of linked documentation in HTML. It describes all the data entities and typelists that make up the data model. The Data Dictionary also lists all attributes and fields for the data entities and extension entities.

2. For the ABContact entity, describe the datatype for the EmailAddress1 field.

varchar(60)

3. ABContact has a Tax ID field. Is the Tax IDs field used to store a social security number (SSN), an employer ID number (EIN), or both?

Both – based on the description: Tax ID for the contact (SSN or EIN).

4. Name one typekey field in ABContact where the field name is identical to the name of the related typelist.

Possible answers: TaxStatus, ValidationLevel, VendorType, etc.

5. Name one typekey field in ABContact where the field name is NOT identical to the name of the related typelist.

Possible answers: PreferredCurrency (which points to Currency), PrimaryPhone (which points to PrimaryPhoneType), Subtype (which points to ABContact subtypes), etc.

Req

6. Name one field on ABContact which is not stored in the database. What type of field is this?

Possible answers: AddressOwner, CollectionAgency, HasUnverifiedEvaluations

7. Which has more arrays: the Group entity or the Role entity?

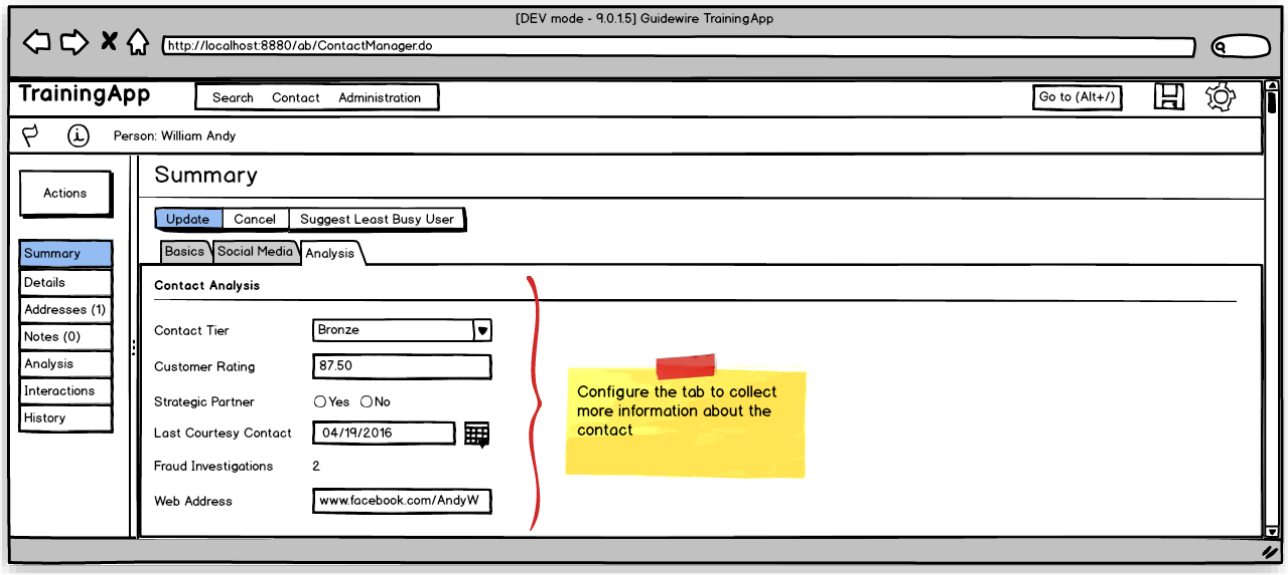
Group – Group has 10 arrays and Role has 5 arrays

8. How many entities have a field named Organization?

10 – From the main page of the Data Dictionary click the All Fields link. Scroll down to the Organization attribute (green color) and count the entities below it.

Lesson 3 Extending the Data Model

An insurance company wants to extend TrainingApp functionality by capturing more details about each contact. The complete requirement will be implemented over multiple modules.



In this exercise your job is to make the necessary **data model changes** to meet customer requirements. As a configuration developer, you will use the Entity Editor in Guidewire Studio to modify the TrainingApp data model. You will implement the **user interface changes** later in a later lesson.

3.1 Prerequisites

For this exercise, 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`.

3.2 Lab: Modify an existing Entity Extension

You will use the Entity Editor to modify the existing ABContact.etx Entity Extension and use the column element to add new data fields to the ABContact base application entity.

Review the field definitions received from the business analysts. This table below summarizes the requirements for each field. Note: The Contact Tier field is missing from this list, because it has already been added by one of our fellow configuration developers.

Field name	Datatype	Null ok?
WebAddress_Ext	String of up to 40 characters	true
FraudInvestigationNum_Ext	Integer	true
LastCourtesyContact_Ext	Date (a date and time value)	true
CustomerRating_Ext	A decimal in the format XXX.Y (Values will range from 0.0 to 999.9)	true
IsStrategicPartner_Ext	Bit (a Boolean value)	true

Important!

Read carefully.

For each new entity element, remember to set the nullok attribute to true if not specifically defined to be false. When required, add an element description and specify column parameters.

Best practice

Use _Ext in entity field name

Notice that every field in the table above has an _Ext suffix. For fields that are added to a base application entity, Guidewire recommends that the field name should end with _Ext. This is to prevent potential conflicts during the upgrade to the next version of the Guidewire application.

1. **Open Guidewire Studio**
   a) From the TrainingApp folder, open a command window and start Guidewire Studio for TrainingApp.
   b) Note: Alternatively, use the Start TrainingApp Studio shortcut.
2. **Navigate to ABContact.etx**

3. **Add fields to ABContact.etx to capture additional details**

   a) Add the elements defined in the table below.

Field name	Datatype	Null ok?
WebAddress_Ext	String of up to 40 characters	True
FraudInvestigationNum_Ext	Integer	True
LastCourtesyContact_Ext	Date (a date and time value)	True
CustomerRating_Ext	A decimal in the format XXX.Y (Values will range from 0.0 to 999.9)	True
IsStrategicPartner_Ext	Bit (aBoolean value)	True

4. **Validate your changes in Guidewire Studio and generate the Java class**

   a) Either click on the validate icon in the Entity Editor or use CTRL + S to save your changes

   b) If the code generation was successful (no errors shown in the Codegen tool window) then open the generated Java class and verify you can find your newly added fields

5. **Deploy your changes**

   a) From studio, restart the server using Debug 'Server'.

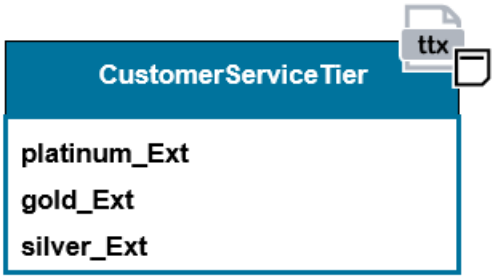
   b) If open, review the Messages Make window for compilation errors.

   c) Review the Debug console for errors.

   d) Verify that the application is running in the Debug console.

3.3 Lab: Extend a base application Typelist

"We also defined new customer service tiers, so we can properly prioritize customer request. Please see the diagram below and configure the Data Model to meet the requirement." – Insurance company business analysts



As a configuration developer, you want to be able to extend base application Typelists so you can add new Typecodes as needed. In this exercise, you will create an extension for the CustomerServiceTier Typelist and use the Typelist Editor to define three new Typecodes.

3.3.1 Configuration

1. **Extend the CustomerServiceTier Typelist**
   a) Create a new TTX file
   b) Define the following Typecodes to display in the order listed using recommended naming conventions:
      - Platinum
      - Gold
      - Silver
2. **Validate your changes in Guidewire Studio and generate the Java class**
   a) Either click on the validate icon in the Typelist Editor or use CTRL + S to save your changes
   b) If the code generation was successful (no errors shown in the Codegen tool window) then open the generated Java class and verify you can find your newly added Typecodes
3. **Deploy your changes**
   a) From studio, restart the server using Debug 'Server'.
   b) If open, review the Messages Make window for compilation errors.
   c) Review the Debug console for errors.
   d) Verify that the application is running in the Debug console.

Best practice

Use intervals when defining priorities

Guidewire recommends using intervals when defining priorities. For example.: 10, 20, 30 This enables you to insert a new value between two existing values without renumbering the other priorities. E.g. you can insert a new Typecode in a later release with the priority 15

Best Practice

Naming typecodes

For typecodes that are added to a base application typelist, Guidewire recommends that the typecode's code should end with _Ext. This is to prevent potential conflicts during the upgrade to the next version of the Guidewire application.

3.3.2 Verification

Activity

Verify the work you have done

Regenerating the data dictionary is not required. Regenerating the data dictionary is a best practice because it creates current documentation for the data model. In addition, the build process for the data dictionary can identify issues beyond schema validation such as referential integrity in the data model.

1. **Build the data dictionary from the command window**
   a) From the bin folder, open a command window.
   b) In the command window, enter the command to build the data dictionary.
2. **Open the data dictionary**
   a) In Windows Explorer, navigate to the data dictionary.
   b) Open the data dictionary using your preferred browser.
3. **View the ABContact entity**
   a) Verify each new field and associated datatype.
4. **View the CustomerServiceTier Typelist**
   a) Verify each new Typecodes.

Best practice

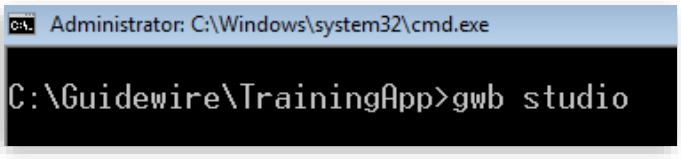
Regenerate the Data Dictionary

Regenerating the data dictionary is not required. Regenerating the data dictionary is a best practice because it creates current documentation for the data model. In addition, the build process for the data dictionary can identify issues beyond schema validation such as referential integrity in the data model.

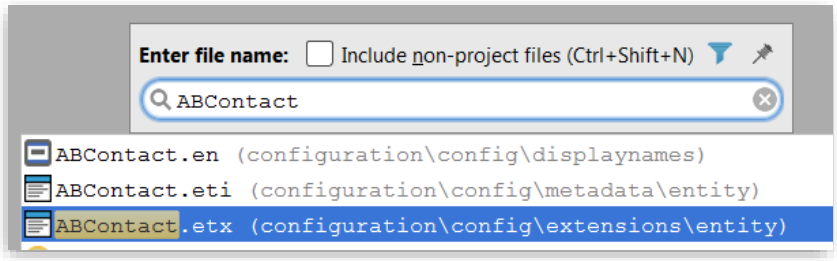
3.4 Lab Solution: Modify an existing Entity Extension

Solution

1. **Open Guidewire Studio**

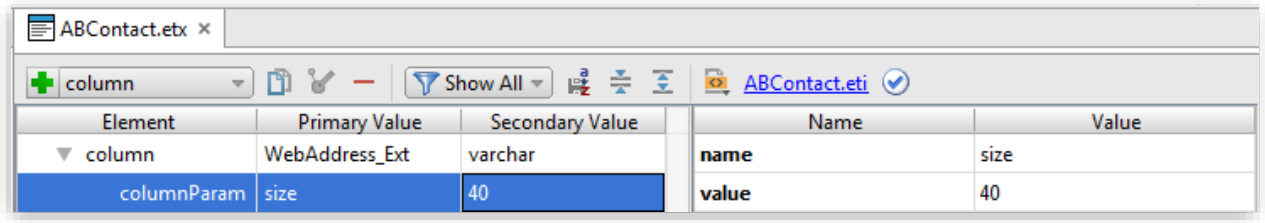


2. **Use CTRL + SHIFT + N to navigate to the existing Entity Extension**

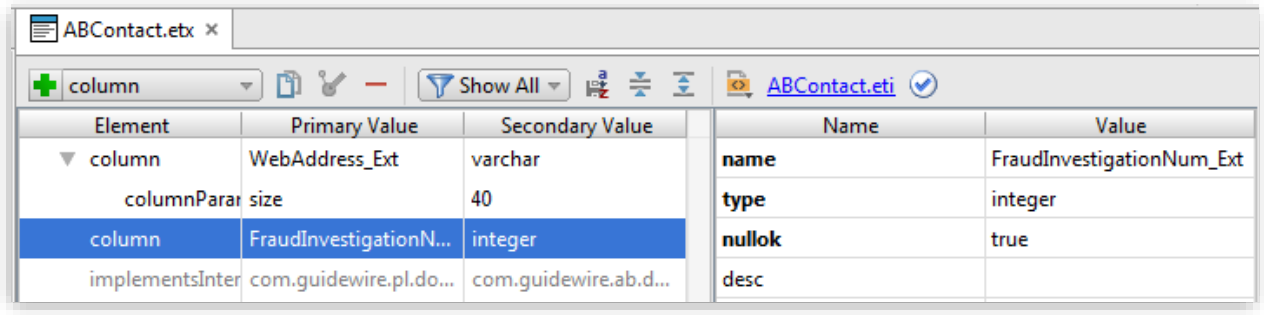


3. **Add the new elements (and subelements) to ABContact.etx**
4. **Add the new WebAddress_Ext column**

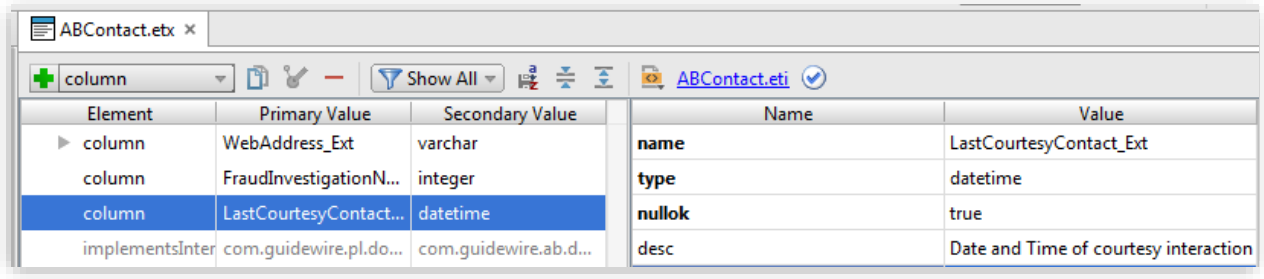
   a) Right click on the WebAddress_Ext column and add select Add New… columnParam to add a columnParam with the following details:



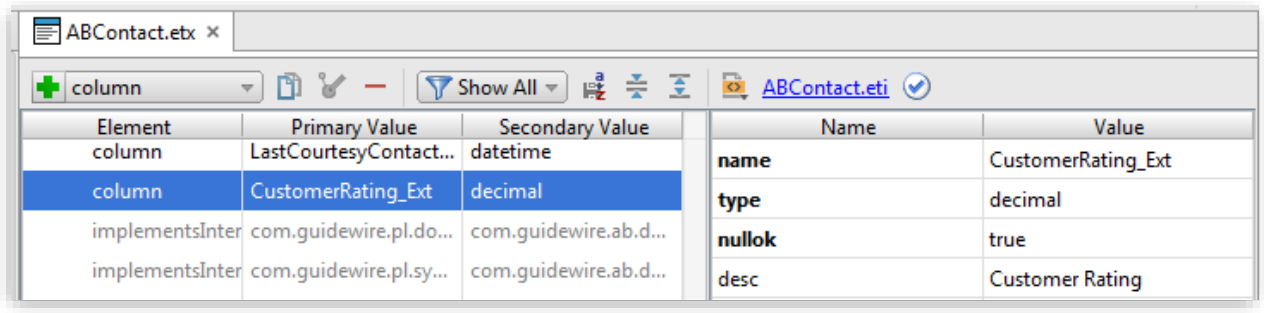
5. **Add the new FraudInvestigationNum_Ext integer column:**



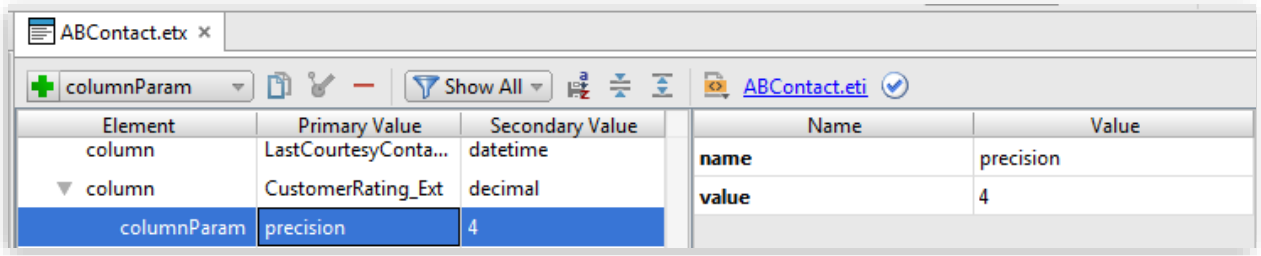
6. **Add a new LastCourtesyContact_Ext column:**



7. **Add a new decimal column with the following details:**

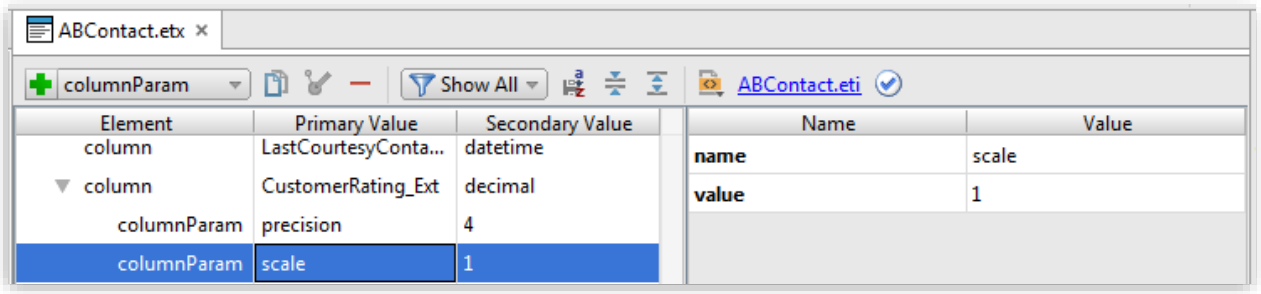


8. **Right click on the CustomerRating_Ext column and add select Add New… columnParam to add a columnParam with the following details:**

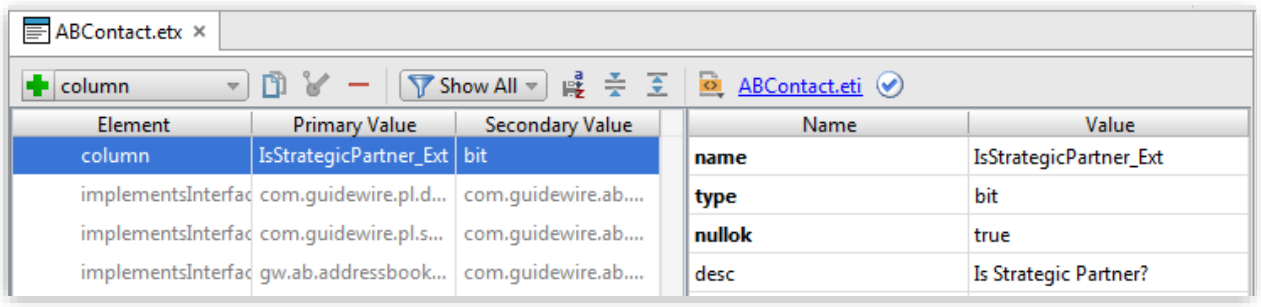


9. **Right click on the CustomerRating_Ext column again and add select Add New… columnParam to add a second columnParam with the following details:**

   Screenshot of the ABContact.etx extension, with the new column

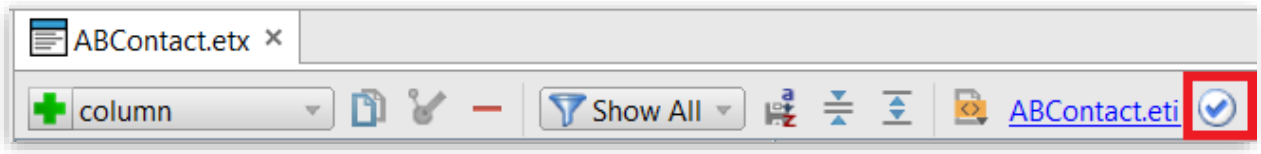


10. **Add the new IsStrategicPartner_Ext column with the following details:**



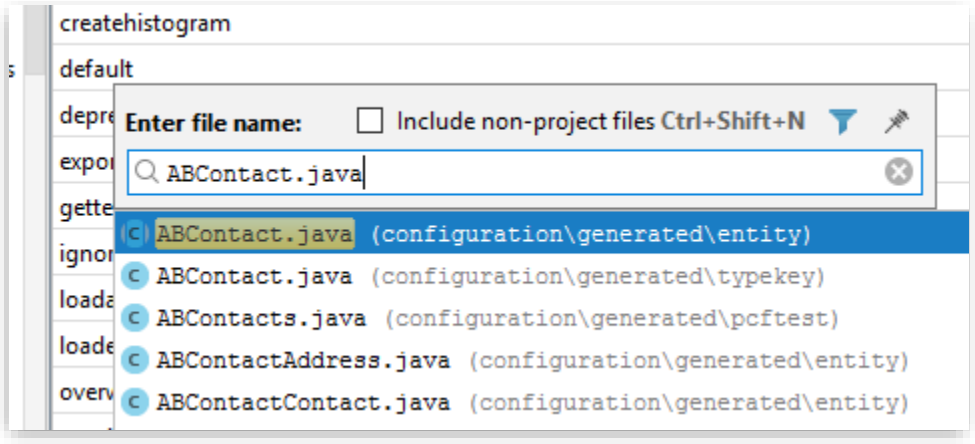
11. **Validate changes in Studio and generate the Java class**

    a) Either click on the validate icon in the Entity Editor or use CTRL + S to save your changes

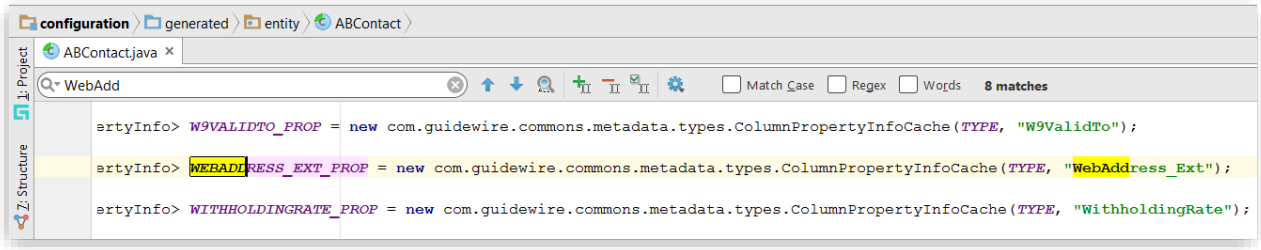


12. If the code generation was successful (no errors shown in the codegen tool window) then open the generated Java class and verify you can find your newly added fields.

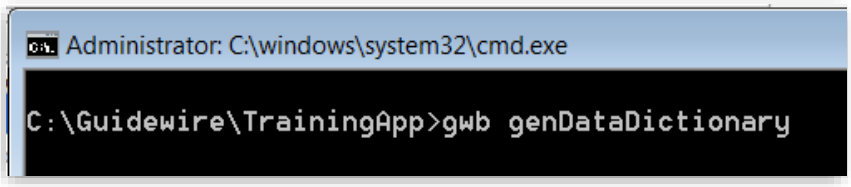
a) Use `CTRL + SHIFT + N` to search for the Java class. Note that the Java class is in the `configuration\generated\entity` package.



The simplest way to locate the new properties is searching for them using the **CTRL + F** shortcut. For example, we are searching for WebAddress_Ext in the following screenshot:

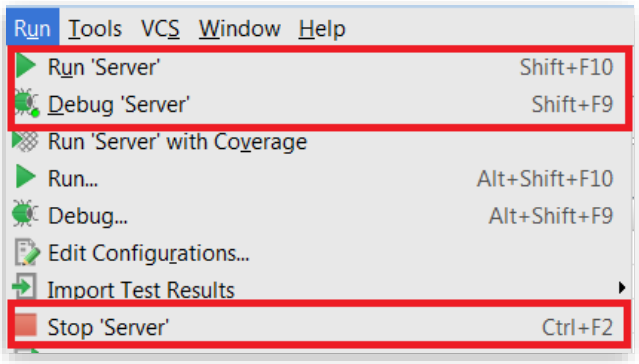


13. Build the Data Dictionary



14. Deploy changes

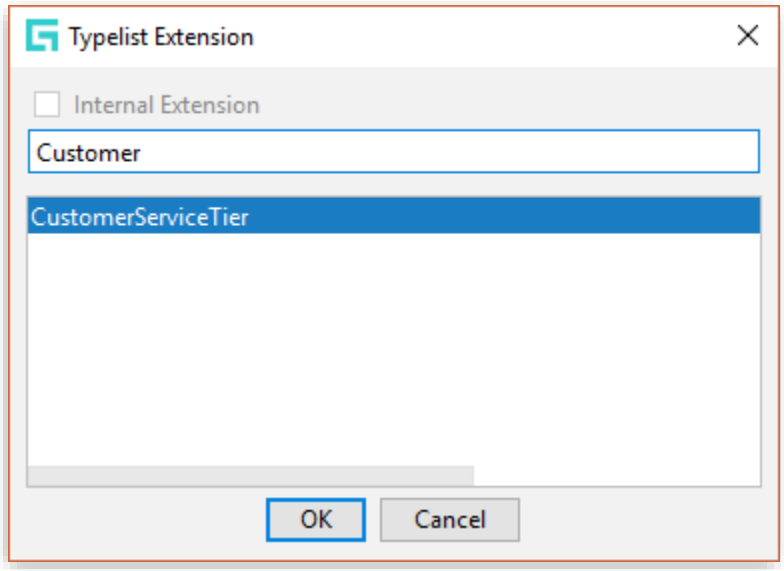
a) To restart the server, first select Main Menu → Run → Stop, then select Main Menu → Run → Run 'Server' or Debug 'Server'.



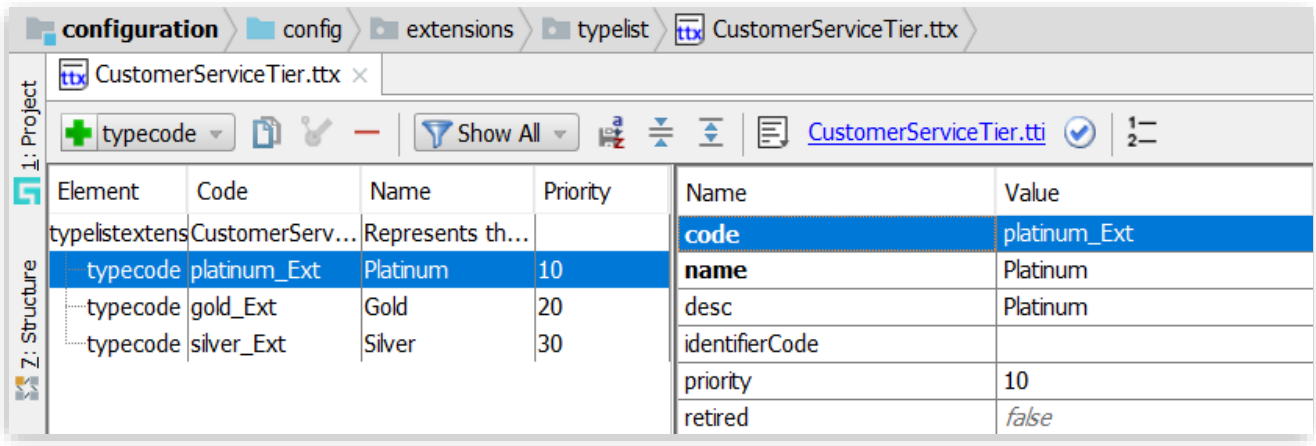
3.5 Lab Solution: Extend a base application Typelist

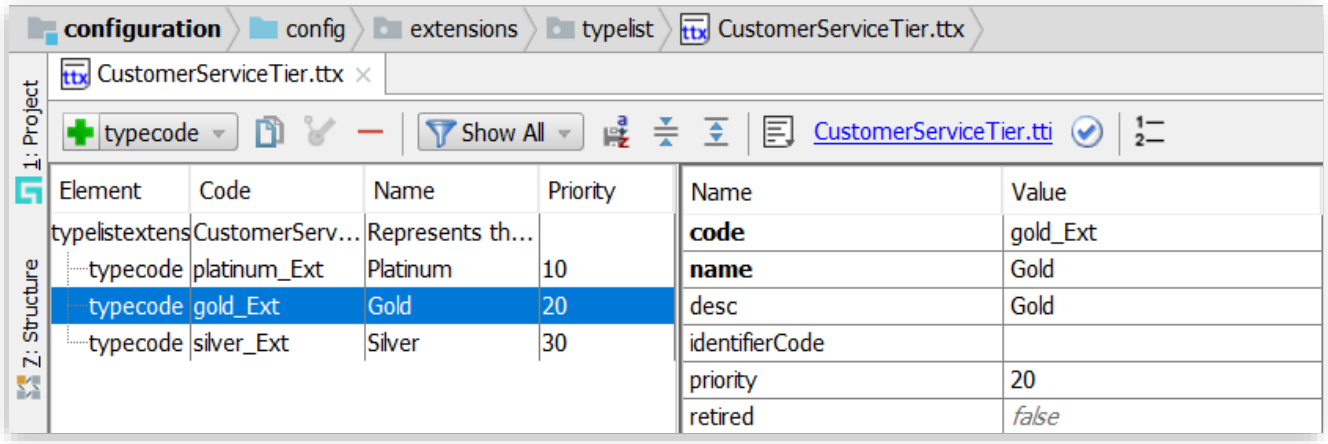
Solution

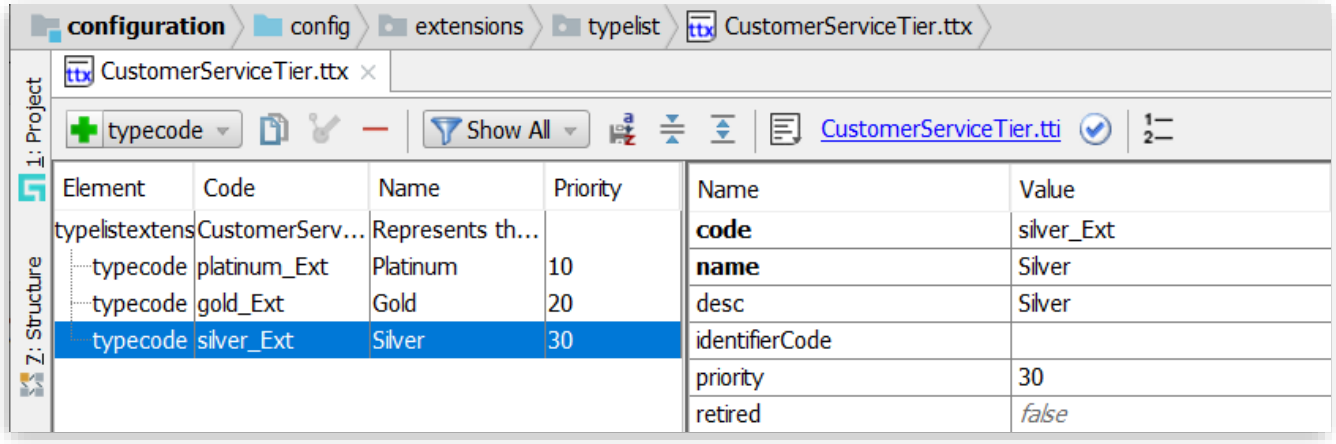
1. Create the Typelist Extension



2. Add Typecodes by clicking the typecode button (the icon with the green plus icon)

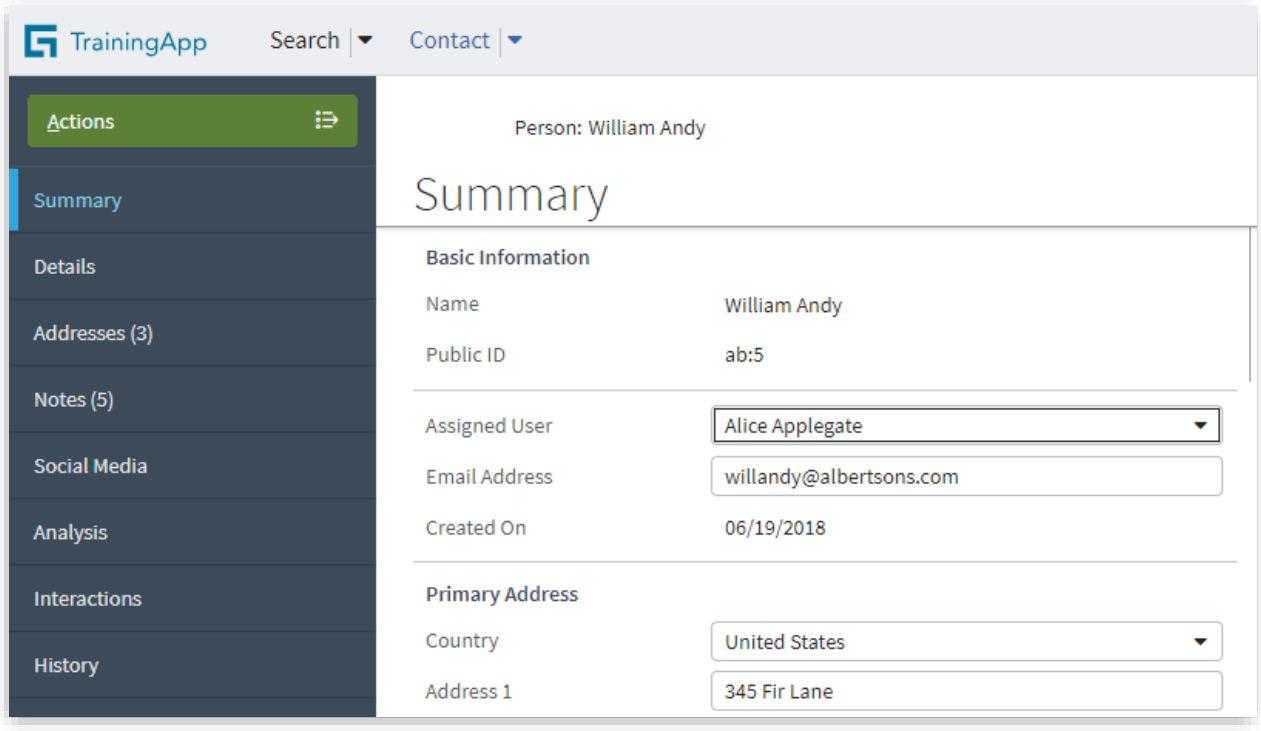






Lesson 4 The User Interface Architecture

After reviewing the out-of-the-box (OOTB) UI layout of your insurance company customer, they communicate that they want to implement a few minor changes on the Summary screen's Basic tab.



In this lab, you will first explore the tools available for investigating the architecture of the user interface. Then, you will use the PCF Editor in Guidewire Studio to modify the order of widgets in a PCF file. Finally, you will deploy and verify your changes.

4.1 Prerequisites

For this exercise, 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`.

4.1.1 Investigation: Explore the page structure

As a configuration developer, you need to understand how to explore the PCF structure, so you can easily find and navigate to the widgets you need to modify.

In this exercise, you will explore the page structure of the summary page using the internal tools and the PCF Editor.

1. **Open Guidewire Studio**
   a) From the TrainingApp folder, open a command window and start Guidewire Studio for TrainingApp.
   b) Alternatively, use the Start TrainingApp Studio shortcut.
2. **Run the TrainingApp project**
   a) Use the main menu, toolbar, or keystroke to run the server.
   b) Verify that the Run Console tab shows `***** ContactManager ready *****`.
3. **Log in to TrainingApp**
   a) Log in as Alice Applegate.
4. **View the William Andy Summary**
   a) Search for William Andy.
   b) In the search results list view, navigate to the William Andy contact.

4.2 Lab: Write it down

Action: Use `ALT + SHIFT + I` to open the Location Info window.

1. **Question: Review the file structure in the dialog. What is the parent PCF of the FlagEntriesLV.pcf?**

2. **Action: Use `ALT + SHIFT + W` to open the Widget Inspector window.**
   Question: Review the Page Include Structure. What is the value of anABContact variable?

3. **Question: Review the Page Widget Structure. Compare the id and renderId values of the Screen widget. How does the renderId value differ from the id value?**

Action: Use `ALT + SHIFT + E` to open the page in Guidewire Studio.

4. **Question: Review the ABContactSummaryPage.pcf structure in the Structure tab. What is the parent widget of the Toolbar widget?**

5. **Action: Navigate to the ABContactSummaryDV.pcf – Double click on the blue area**
   Question: Review the ABContactSummaryDV.pcf in the canvas. How are you able to distinguish included sections from the widgets specified in the file itself?

Keyboard shortcut

Open Location Information

If you are running an application project with internal tools enabled, you can view the location information with `ALT + SHIFT + I` in a separate browser window. The window details the file structure and details the hierarchy of the location, screen, and any child container widgets. For each location and container widget, the name of the file in which it is referenced is listed. Location Info is also useful when Studio is not running.

Keyboard shortcut

Open Widget Inspector

If you are running an application project with internal tools enabled, you can view the widget information with `ALT + SHIFT + W` in a separate browser window. The Widget Inspector shows all the PCF files and widgets referenced in the active application browser window except for the workspace area. The Widget Inspector also shows the available variables and their current values.

Keyboard shortcut

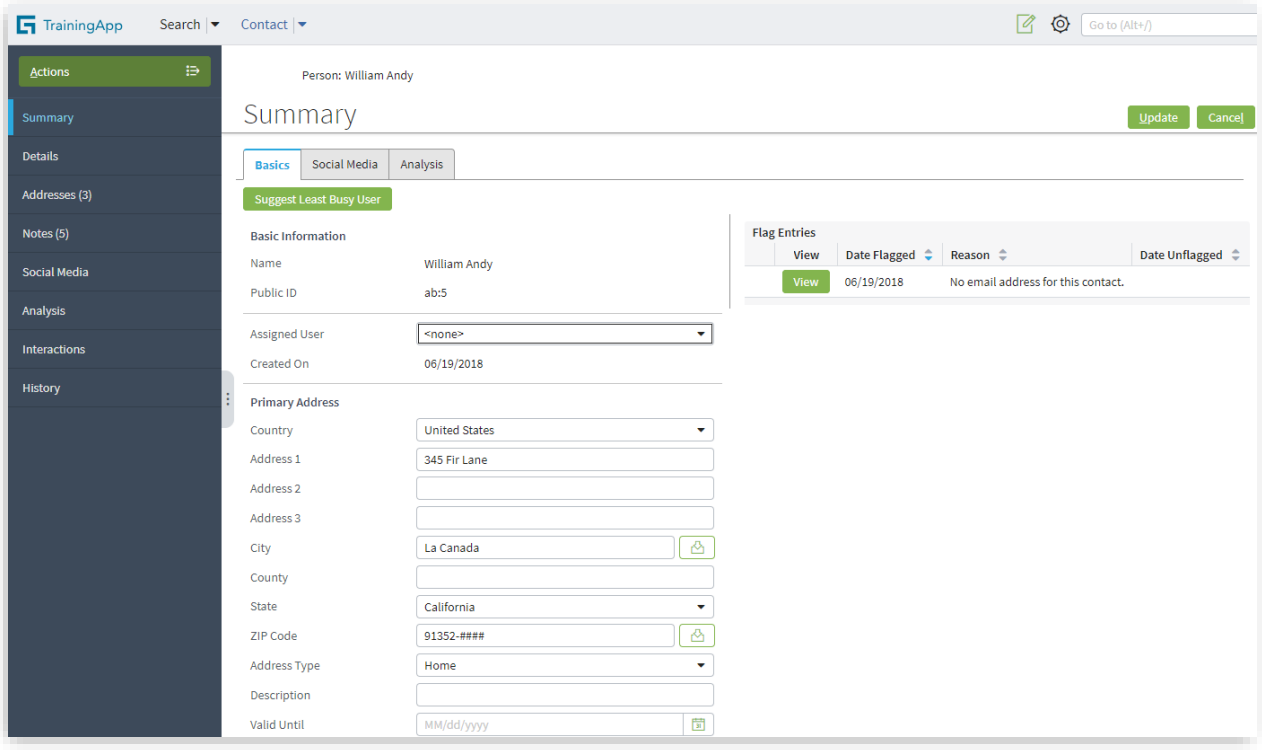
Open a PCF file in Studio from the Browser

If you are running an open application project in Guidewire Studio and if internal tools are enabled, you can automatically open the location being viewed in the application. Use the `ALT + SHIFT + E` keystroke in the browser.

4.3 Lab: Modify the Basics card on the Summary page

As a configuration developer, you want to be able to open and modify existing PCF files.

In this exercise, you will modify the ABContactSummaryDV.pcf file that you have already open in the PCF Editor in Guidewire Studio.



4.3.1 Configuration

1. **Modify the order of the widgets in Basic Information**
   a) Move the Created On widget below the Assigned User widget.
   b) From the Toolbox, add an input divider between Assigned User and Public ID.

4.3.2 Verification

Activity

Verify the work you have done

1. **Reload the PCF changes**
   a) In TrainingApp, reload the changes to the PCF file(s).
   b) Verify the changed order of the widgets and the addition of the input divider.

Keyboard shortcut

Reload PCF changes without restarting the Server

If you are running an open application project in Guidewire Studio and if internal tools are enabled, you can reload all the page configuration files and display keys for the server.

Important: If you reload PCF files while in edit mode, you may experience unpredictable results. For the current location, where there is a data modification in progress, the new PCFs may not be reloaded. Therefore, Guidewire recommends reloading PCF files while in read-only mode as it provides for more predictable results.

Use the `ALT + SHIFT + L` keystroke in the browser.

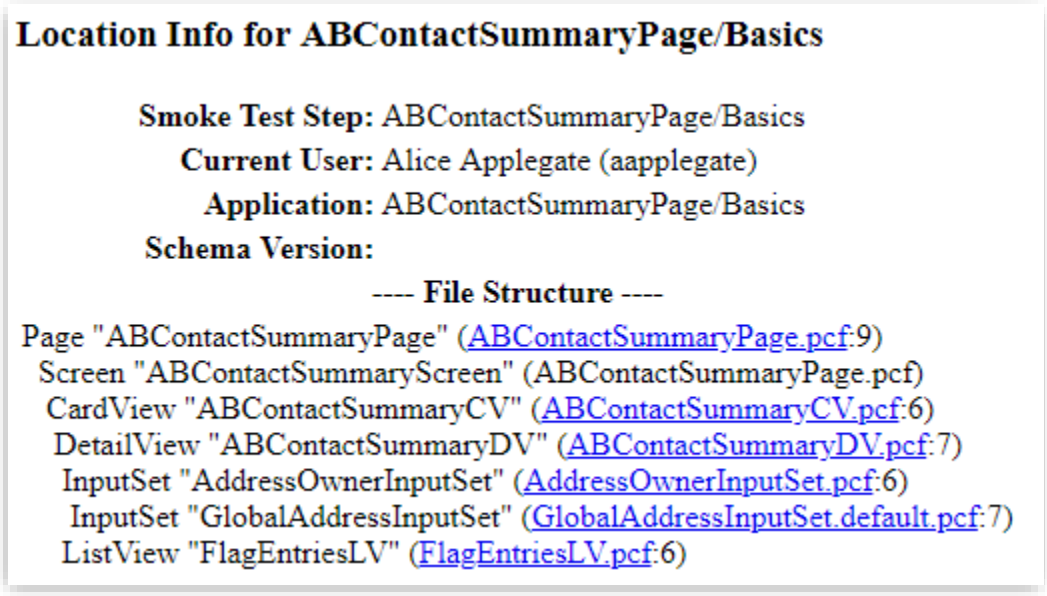
Stop

4.4 Lab Solution: Write it down

Solution

Exact details on how to complete the exercise

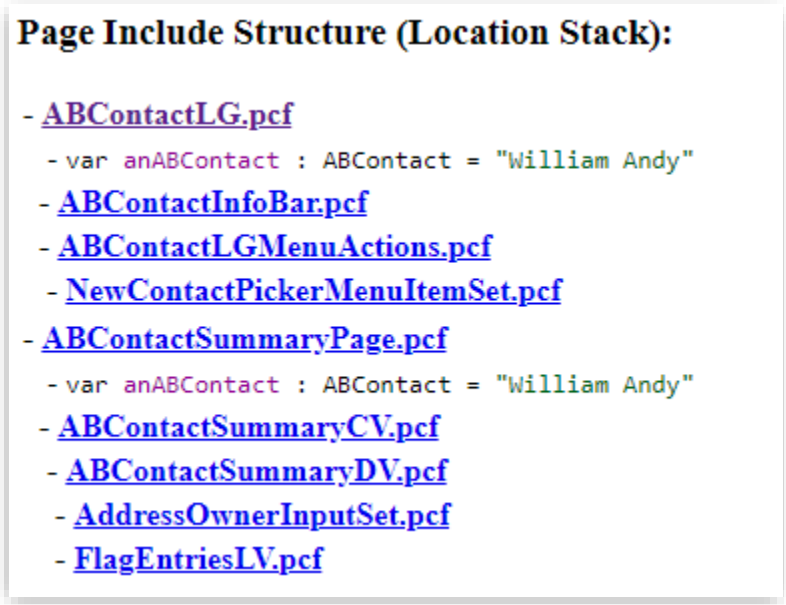
1. **Action: Use `ALT + SHIFT + I` to open the Location Info window.**



2. **Question: Review the file structure in the dialog. What is the parent PCF of the FlagEntriesLV.pcf?**

ABContactSummaryDV – you can see that based on the indent

3. **Action: Use `ALT + SHIFT + W` to open the Widget Inspector window.**



4. **Question: Review the Page Include Structure. What is the value of anABContact variable?**

"William Andy"

5. **Question: Review the Page Widget Structure. Compare the id and renderId values of the Screen widget. How does the renderId value differ from the id value?**

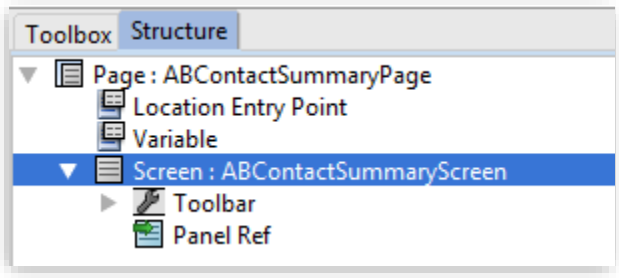
- Collapse

Widget	ID	Render ID	File	Label
Page	ABContactSummaryPage		ABContactSummaryPage.pcf:9	
Screen	ABContactSummaryScreen	ABContactSummaryPage-ABContactSummaryScreen	ABContactSummaryPage.pcf:16	
TitleBar	ttlBar	ABContactSummaryPage-ABContactSummaryScreen-ttlBar	ABContactSummaryPage.pcf:16	
Messages	_msgs	ABContactSummaryPage-ABContactSummaryScreen-_msgs	ABContactSummaryPage.pcf:16	

renderId includes the id of the page in which to render the screen, whereas id simply gives the id of the screen Generally speaking the id simply gives the id of the widget, whereas renderId includes the id of the widget and the concatenation of all the parent ids.

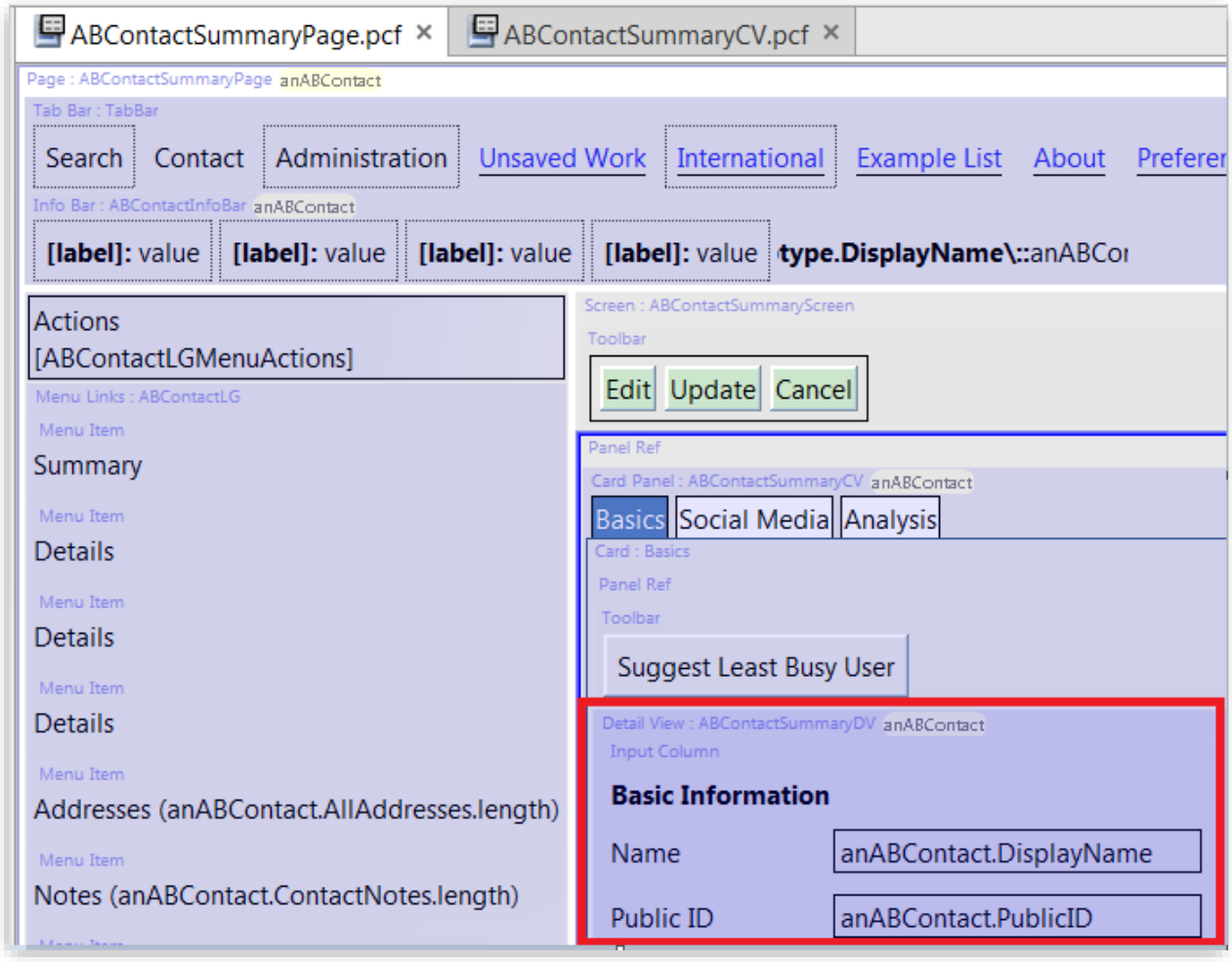
Action: Use `ALT + SHIFT + E` to open the page in Guidewire Studio.

6. **Question: Review the ABContactSummaryPage.pcf structure in the Structure tab. What is the parent widget of the Toolbar widget?**



ABContactSummaryScreen

7. **Action: Navigate to the ABContactSummaryDV.pcf – Double click on the blue area**



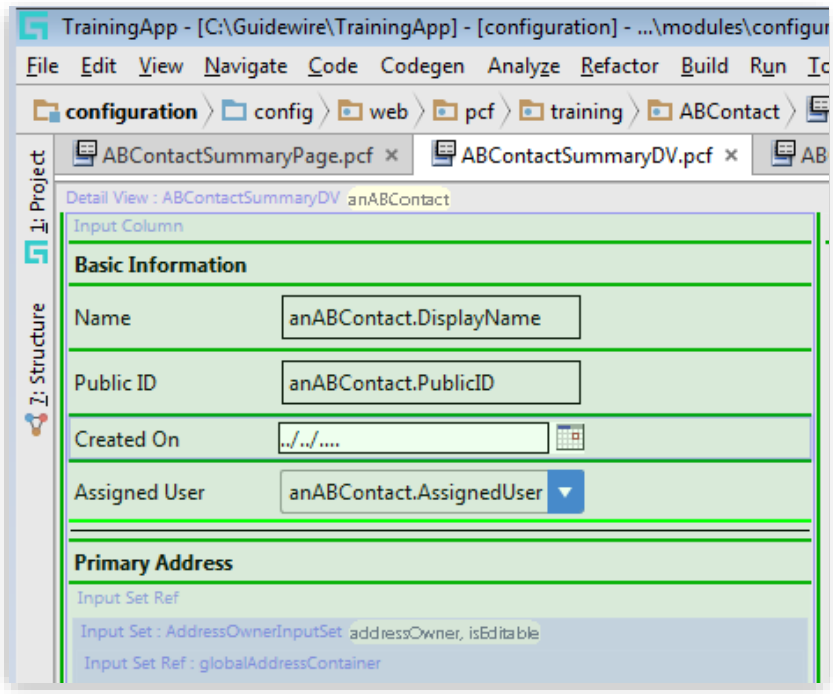
8. **Question: Review the ABContactSummaryDV.pcf in the canvas. How are you able to distinguish included sections from the widgets specified in the file itself?**

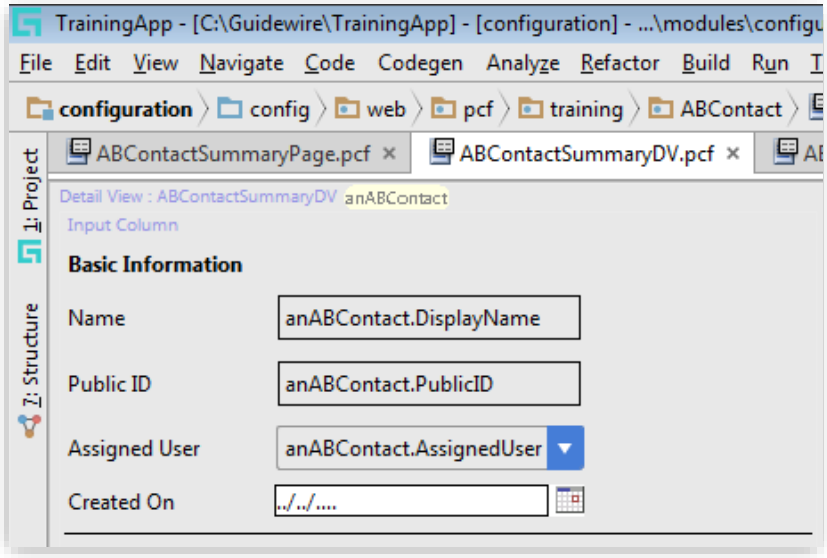
Included sections are colored purple (the color varies depending on how deep the files are nested), whereas widgets that are in the file have a gray background.

4.5 Lab Solution: Modify the Basics card on the Summary page

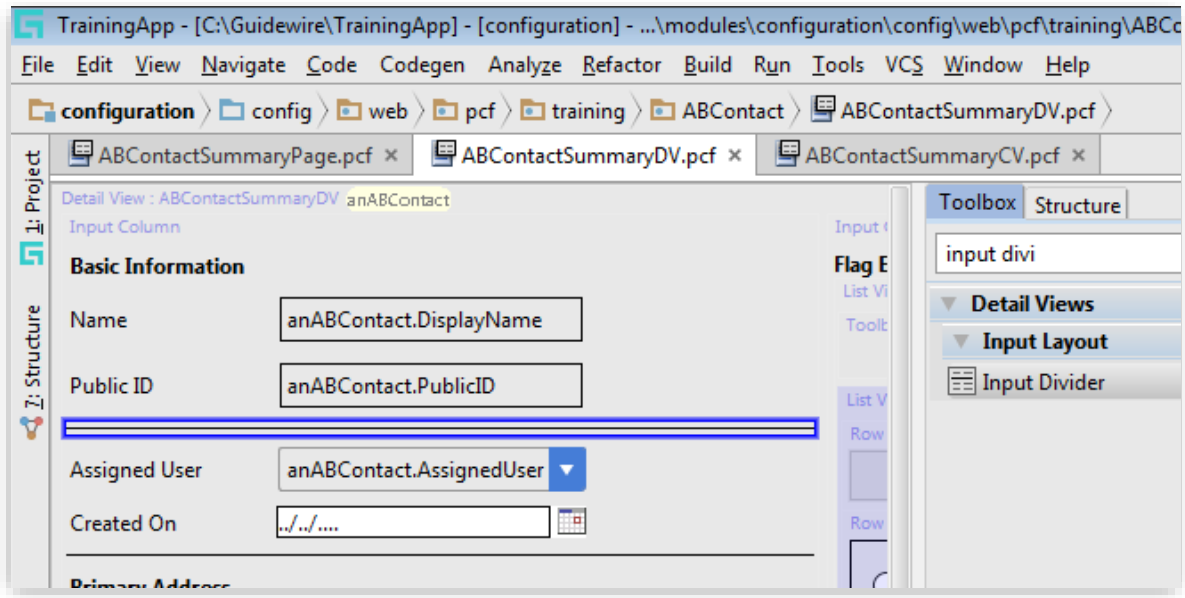
1. **Move Created On widget below Assigned User widget. You can drag the Created On to its new position.**

Remember: Dark green lines are places where the widget could be placed; and the light green line is where the widget will be placed.

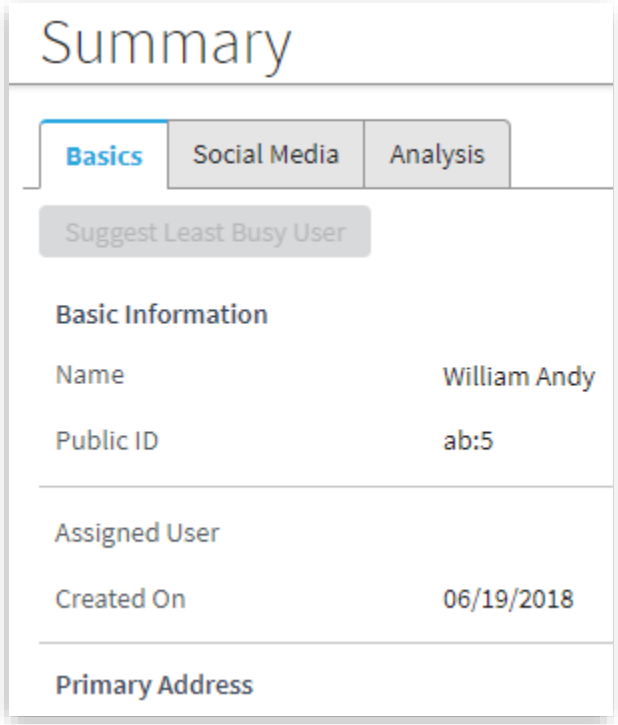




a) From the Toolbox, add an input divider between Assigned User and Public ID

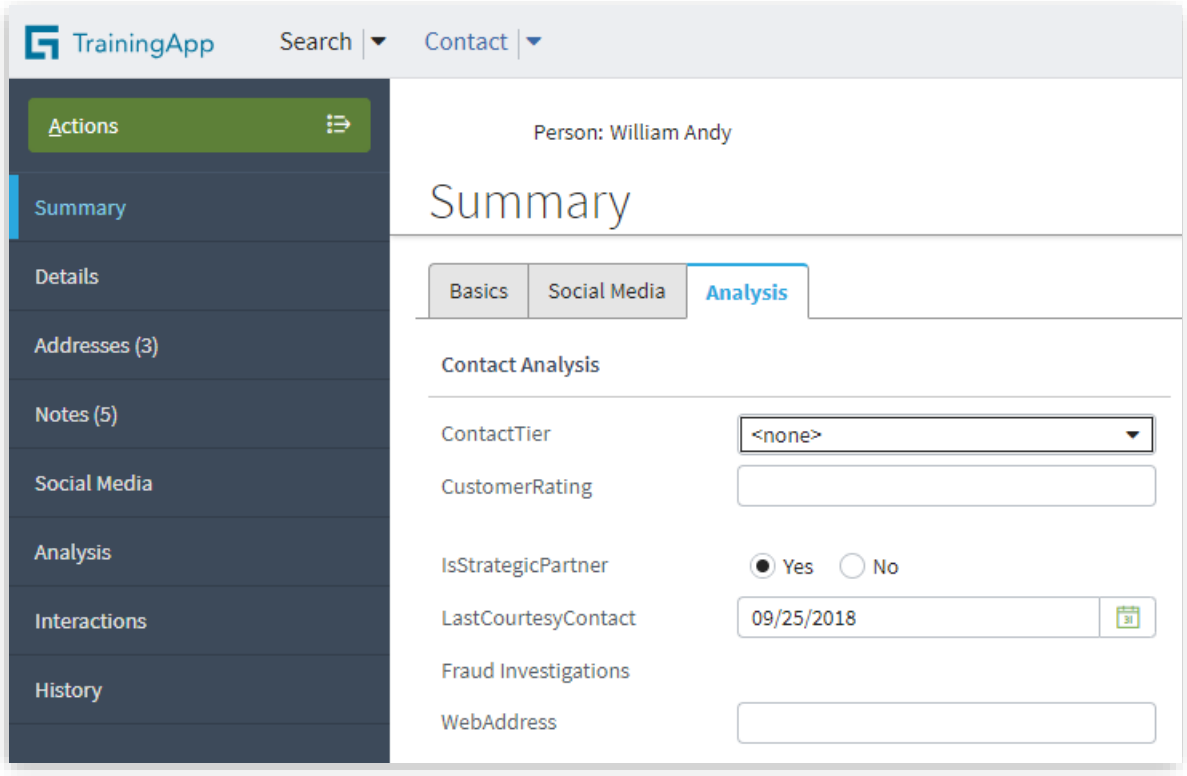


b) Reload the PCF changes using `ALT + SHIFT + L` in the browser.



Lesson 5 Atomic widgets

An insurance company wants to extend TrainingApp functionality by capturing more details about each contact. The complete requirement will be implemented over multiple modules. Recall that you have already implemented the required **data model changes** in the previous *Extending Base Application Entities* lesson.



In this exercise your job is to make the necessary **user interface changes** to meet customer requirements. As a configuration developer you will use the PCF Editor in Guidewire Studio to modify the TrainingApp UI.

In this module, you will first practice the DOT notation. Next, you will review the requirements. Finally, you will configure the appropriate PCF file to display the fields and deploy your changes.

5.1 Prerequisites

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

1. **Extending Base Application Entities**

For this exercise, 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: Practice the Dot notation

As a configuration developer, you need to understand how to work with the dot notation. In this exercise, you will use the data dictionary to determine how to reference entities, fields, and properties using dot notation.

1. Open the data dictionary

a) Open the data dictionary in your browser.

5.2.1 Write it down

For an object named "anABContact" that is of the type ABContact, identify the correct dot notation syntax for the following:

1. The creation date and time for anABContact.

2. The tax status for anABContact.

3. The city for the primary address of anABContact.

4. All the buildings associated with anABContact.

5. Assuming that anABContact is a company, the number of employees for anABContact.

6. Assuming that anABContact is a legal venue, the type of court for anABContact.

Stop

5.3 Lab: Determining Widgets and their requirements

In this exercise, you will create new atomic widgets to capture and display additional contact details for an existing details view. Review the widget requirements received from the business analysts. This table summarizes the requirements for each widget. Refer to the wireframe at the beginning of the exercise to see how the fields will be displayed in the UI.

As a configuration developer, you want to be able to find fields in the data dictionary and determine their data model type. As a configuration developer, you want to be able to determine the appropriate widget and Gosu data type based on the data model type of a field.

1. Open the Data Dictionary

a) Open the data dictionary in your browser.

2. Review the Widget reference table

Data type (Data Model)	Recommended widget(s) in a Detail View	Recommended widget(s) in a List View	Data type (Gosu)
varchar	Text Input	Text Cell	java.lang.String
integer	Text Input	Text Cell	java.lang.Integer
decimal	Text Input	Text Cell	java.math.BigDecimal
text	Text Input Text Area Input	Text Cell Text Area Cell	java.lang.String
date, datetime	Date Input	Date Cell	java.util.Date

Data type (Data Model)	Recommended widget(s) in a Detail View	Recommended widget(s) in a List View	Data type (Gosu)
bit	Boolean Dropdown Input Boolean Radio Button Input Check Box Input	Boolean Radio Cell Checkbox Cell	java.lang.Boolean
foreignkey	Range Input Range Radio Button Input	Range Cell	OtherEntityType
typekey	TypeKey Input TypeKey Radio Button Input	Typekey Cell	typekey.TypelistName

Table 1 Widget reference table

3. Use the Data Dictionary and the Widget reference table to complete the following table Note: All the fields are defined on ABContact entity

	Label/Info	Editable	Field	Datatype (Data Model)	Widget	valueType (Gosu)
1	Contact Analysis	n/a	n/a	n/a		n/a
2	We need a horizontal divider	n/a	n/a	n/a		n/a
3	Contact Tier	Yes				
4	Customer Rating	Yes				
5	Strategic Partner	Yes				

	Label/Info	Editable	Field	Datatype (Data Model)	Widget	valueType (Gosu)
6	Last Courtesy Contact	Yes				
7	Fraud Investigation Number	No				
8	Web Address	Yes				

Stop

5.4 Lab: Add new Atomic Widgets

As a configuration developer, you want to create atomic widgets; bind some of those atomic widgets to entity fields to display data from the database; and define user-friendly labels.

1. **Open Guidewire Studio**
   a) From the TrainingApp folder, open a command window and start Guidewire Studio for TrainingApp.
   b) Alternatively, use the Start TrainingApp Studio shortcut.
2. **Run or Debug the TrainingApp project**
   a) Use the main menu, toolbar, or keystroke to run or Debug the server.
   b) Verify that the Run or Debug Console tab shows ***** ContactManager ready *****.
3. **Log in to TrainingApp**
   a) Log in as Alice Applegate.
4. **View the William Andy Summary**
   a) Search for William Andy.

b) In the search results list view, navigate to the William Andy contact.

5. Open the Summary Page in Guidewire Studio

a) Use the appropriate internal tools shortcut to open the PCF file in Guidewire Studio.

Keyboard shortcut

Open a PCF file in Studio from the Browser

If you are running an open application project in Guidewire Studio and if internal tools are enabled, you can automatically open the location being viewed in the application. Use the `ALT + SHIFT + E` keystroke in the browser.

6. Open the ABContactSummaryCV

a) In the PCF Editor, open the ABContactSummaryCV.pcf file.

7. Add widgets to the Detail View Panel on the Analysis card

a) Use the Toolbox to add widgets.

b) Use the Properties window to define the necessary widget properties when such as id*, editable, and value*.

c) When defining labels, create display keys.

d) Use your completed table and the wireframe as guidelines

Best practices

Adhere to Best practices

Guidewire recommends PCF file should not contain "hard coded" display text. Benefits of display keys:

- Easily localize text and accommodate users from various locales
- Allows for labels with dynamic content
- Reusability – you must make modifications only at one place even if you decide to change a certain text that is shown at multiple places in the User Interface

The recommended naming convention for customer display keys, is to use a standard prefix such as "Ext". Use the following syntax to reference a display key:

```
DisplayKey.get("NameOfTheKey")
```

Or if the key expects parameters:

```
DisplayKey.get("NameOfTheKey", Parameter1, Parameter2,… )
```

Hint

View or modify a Display Key value

To view the value, just simply hold down the CRTL key and hover over the Display Key with your mouse. To modify the value, just simply hold down the CTRL key and left click on the Display Key with your mouse. In general, the same methods work the same way for other elements in Studio, such as entity fields, Gosu variables, PCF file references, etc.

CTRL + Hover: Tooltip
CTRL + Click: Open definition file

Hint

Hover over for tooltips

Simply hover over the PCF element (or property) in the toolbox (or property window) to see the PCF documentation about what that element (or property) does.

5.4.1 Verification

Activity

Verify the work you have done

1. **Reload the PCF changes**
   a) In TrainingApp, reload the changes to the PCF file(s) and display keys.
   b) Verify your changes on the Summary page for William Andy.

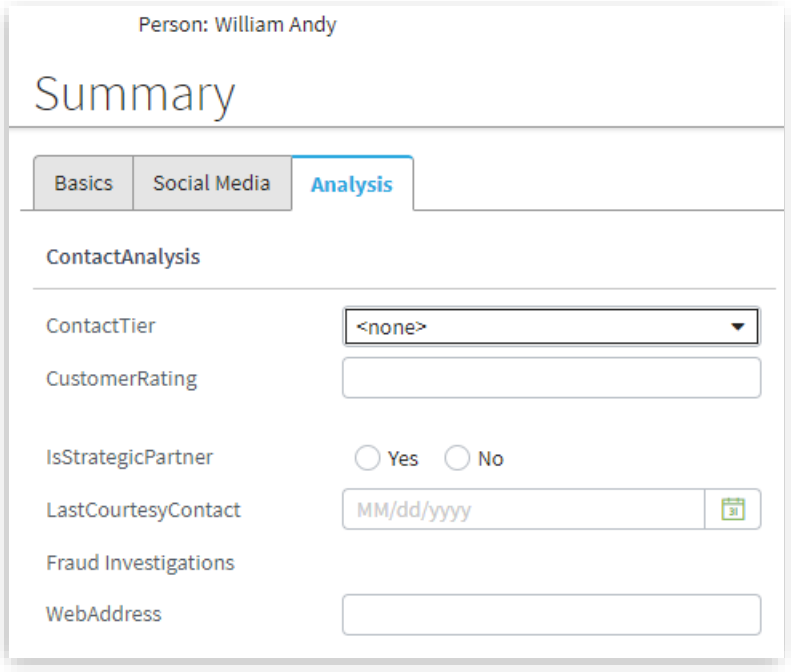
Keyboard shortcut

Reload PCF changes without restarting the Server

If you are running an open application project in Guidewire Studio and if internal tools are enabled, you can reload all the page configuration files and display keys for the server.

Important: If you reload PCF files while in edit mode, you may experience unpredictable results. For the current location, where there is a data modification in progress, the new PCFs may not be reloaded. Therefore, Guidewire recommends reloading PCF files while in read-only mode as it provides for more predictable results.

Use the `ALT + SHIFT + L` keystroke in the browser.



5.5 Bonus Lab: Investigate optional widget properties

In this exercise, you will use the PCF Format Reference to investigate various widget properties.

1. **Open the PCF Format Reference**
   a) In Chrome, open the `pcf.html` file from the `C:\<installDirectory/>\TrainingApp\modules\pcf.html`
2. **Review the BooleanRadioInput widget properties**
   a) Use the Quick Jump drop-down list to select the BooleanRadioInput widget.
   b) Review the widget properties.
3. **Review the DateInput widget properties**
   a) Use the Quick Jump drop-down list to select the DateInput widget.
   b) Review the widget properties.
4. **Review the TextInput widget properties**

a) Use the Quick Jump drop-down list to select the TextInput widget.

b) Review the widget properties.

5. Review the TypeKeyInput widget properties

a) Use the Quick Jump drop-down list to select the TypeKeyInput widget.

b) Review the widget properties.

5.5.1 Write it down

Requirement: When a user attempts to modify the Is Verified widget value, a dialog box opens. The dialog box asks the user to confirm the change.

1. Question: What is the name of the attribute (property) for a BooleanRadioInput that displays a confirmation message in a dialog box?

Requirement: The label for the Evaluation Date must be in bold in the user interface.

2. **Question: What is the name of the attribute (property) for a DateInput that allows a configurator to implement a bold style for the label?**

Requirement: When a user hovers over a Legacy Code text input widget with their mouse cursor, the user interface displays, "The code must come from the legacy claim system."

3. Question: What is the name of the attribute (property) for a TextInput that displays this message?

Stop

Solution

5.6 Lab Solution: Practice Dot notation

1. **The creation date and time for anABContact.**

```
anABContact.CreateTime
```

2. **The tax status for anABContact.**

```
anABContact.TaxStatus
```

3. **The city for the primary address of anABContact.**

```
anABContact.PrimaryAddress.City
```

4. **All the buildings associated with anABContact.**

```
anABContact.Buildings_Ext
```

5. **Assuming that anABContact is a company, the number of employees for anABContact.**

```
(anABContact as ABCompany).NumberOfEmployees
```

6. **Assuming that anABContact is a legal venue, the type of court for anABContact.**

```
(anABContact as ABLegalVenue).VenueType
```

5.7 Lab Solution: Determining widgets and their requirements

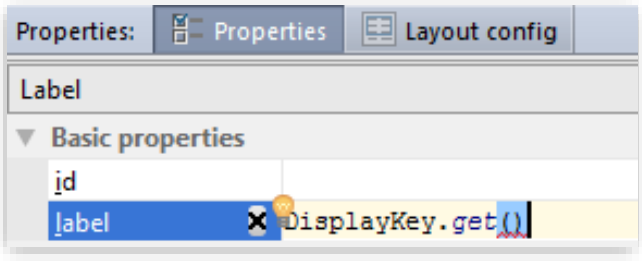
Solution

	Label/Info	Editable	Field	Datatype (Data Model)	Widget	valueType (Gosu)
1	Contact Analysis	n/a	n/a	n/a	Label	n/a
2	We need a horizontal divider	n/a	n/a	n/a	Input divider	n/a
3	Contact Tier	Yes	ContactTier	typekey.ContactTier	TypeKey Input	typekey.ContactTier
4	Customer Rating	Yes	CustomerRating_Ext	decimal	Text Input	java.math.BigDecimal
5	Strategic Partner	Yes	IsStrategicPartner_Ext	bit	Boolean Radio Button Input	java.lang.Boolean
6	Last Courtesy Contact	Yes	LastCourtesyContact_Ext	datetime	Date Input	java.util.Date
7	Fraud Investigation Number	No	FraudInvestigationNum_Ext	integer	Text Input	java.lang.Integer
8	Web Address	Yes	WebAddress_Ext	varchar(60)	Text Input	java.lang.String

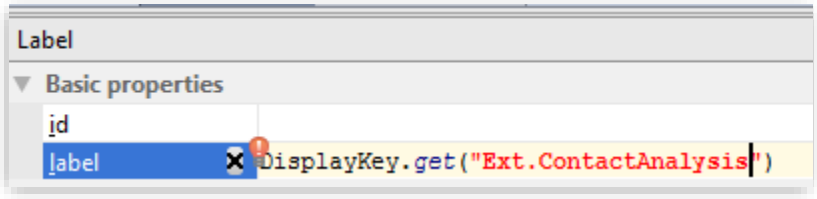
5.8 Lab Solution: Add new atomic widgets

Solution

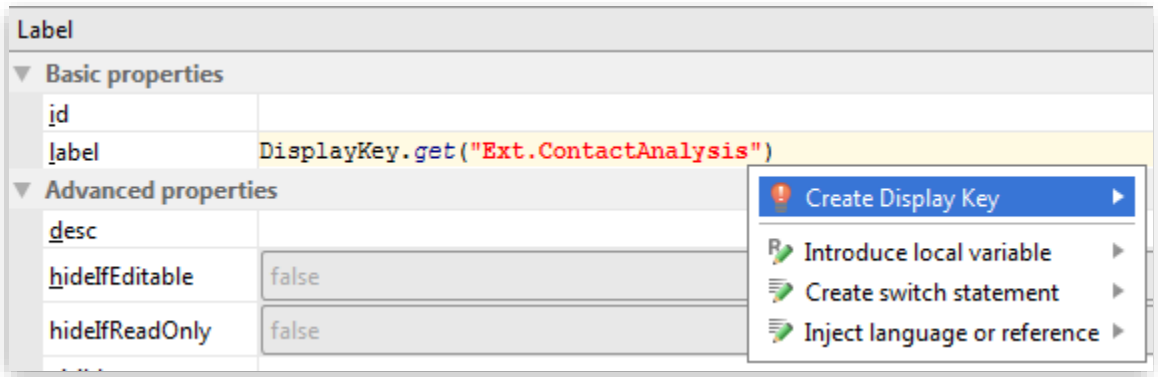
1. **Call the get() function of the DisplayKey class in the label property.**



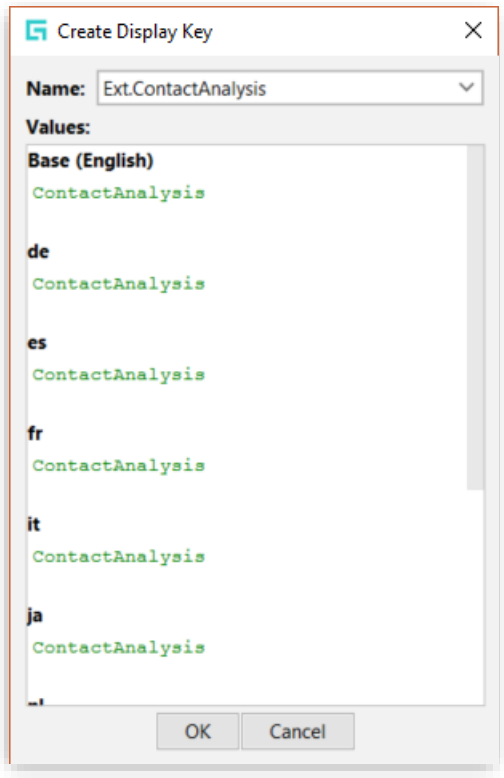
2. **Enter the name for the display key in the get() function as a String parameter (between double quotes).**



3. **Press Alt + Enter, and click Create Display Key:**

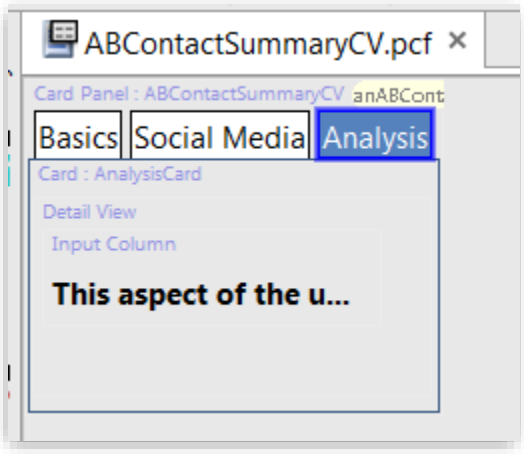


4. **Then enter the text value for the locale and click OK**



5. **Open the PCF file:**

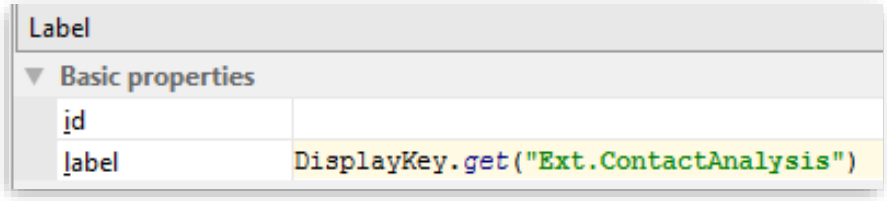
   a) Open ABContactSummaryCV.pcf and select the Analysis card.



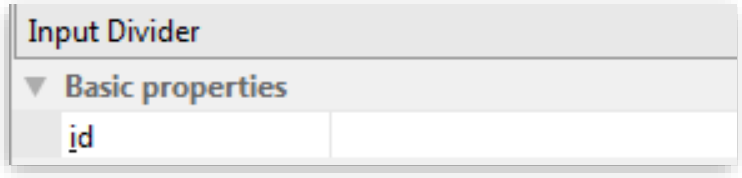
   b) To delete the existing label, right click on it and select Delete from the context menu.

6. **Drag and drop widgets inside the existing Input Column**

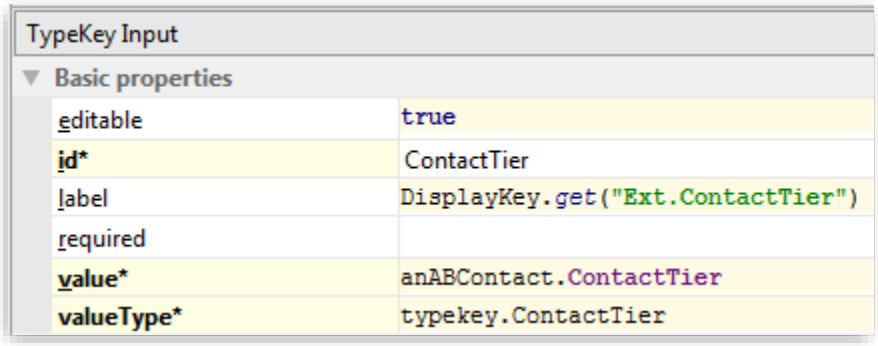
   a) Label widget for Contact Analysis label:



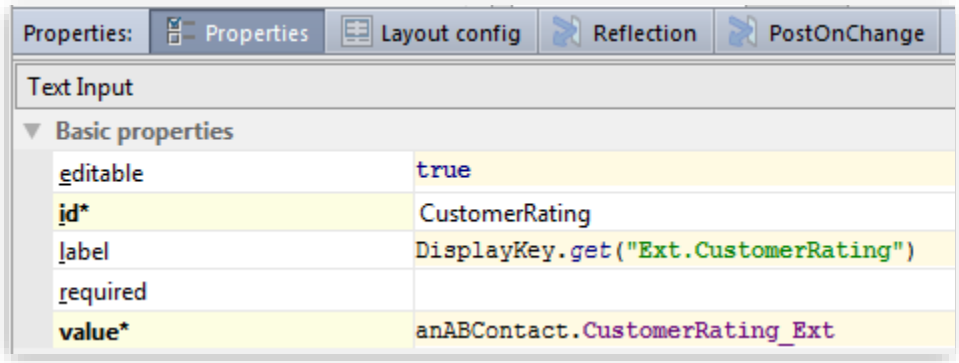
b) Input Divider widget to display a horizontal line:



c) TypeKey Input widget for Contact Tier



7. Text Input widget for Customer Rating:



…and set the valueType to java.math.BigDecimal

validationExpression	
valueType	java.math.BigDecimal
visible	

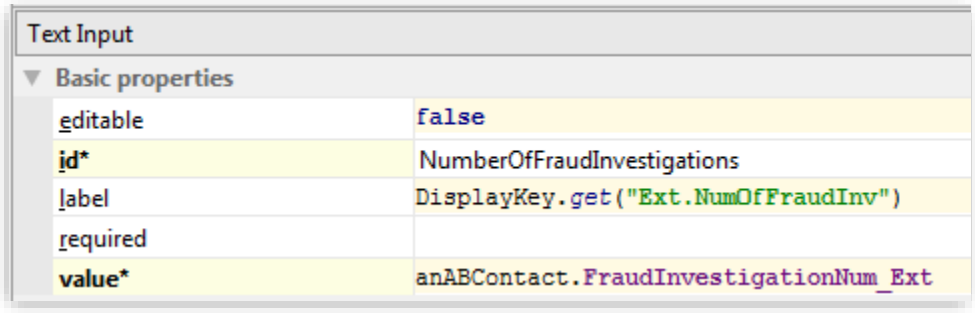
8. Boolean Radio Button Input widget for Strategic Partner:

Boolean Radio Button Input	
▼ Basic properties	
editable	true
falseLabel	
id*	IsStrategicPartner
label	DisplayKey.get("Ext.IsStrategicPartner")
required	
trueLabel	
value*	anABContact.IsStrategicPartner_Ext

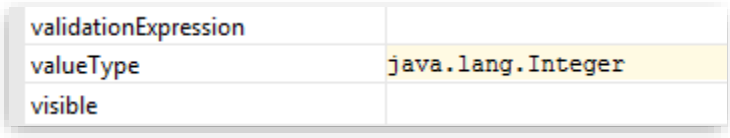
9. Date Input widget for Last Courtesy Contact:

Date Input	
▼ Basic properties	
dateFormat	<none selected>
editable	true
id*	LastCourtesyContact
label	DisplayKey.get("Ext.LastCourtesyContact")
required	
timeFormat	<none selected>
value*	anABContact.LastCourtesyContact_Ext

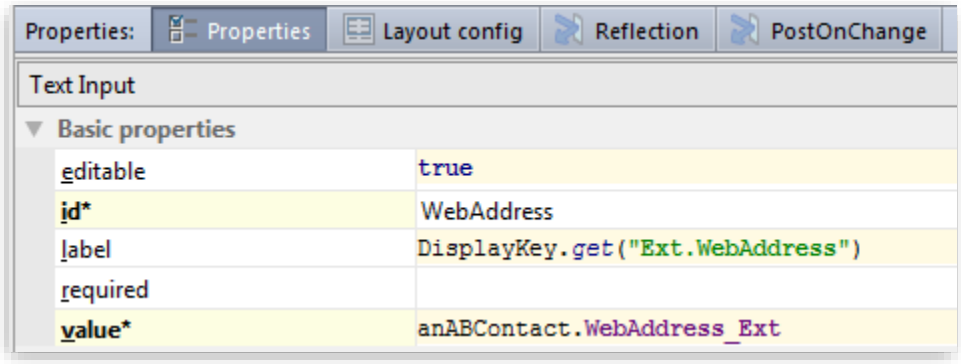
10. Text Input widget for Number Of Fraud Investigations…



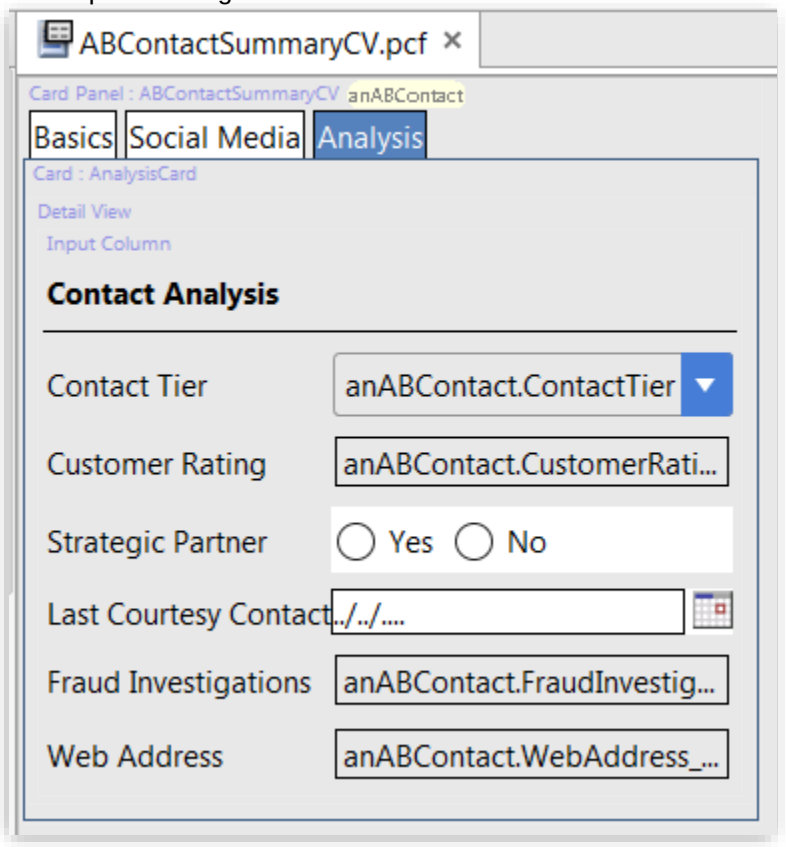
…and set the valueType to java.lang.Integer



11. Text Input widget for Web Address:



The completed design:



5.9 Bonus Lab Solution: Investigate optional widget properties

Solution

Requirement: When a user attempts to modify the Is Verified widget value, a dialog box opens. The dialog box asks the user to confirm the change.

1. **Question:** What is the name of the attribute (property) for a BooleanRadioInput that displays a confirmation message in a dialog box?

showConfirmMessage

Requirement: The label for the Evaluation Date must be in bold in the user interface.

2. **Question:** What is the name of the attribute (property) for a DateInput that allows a configurator to implement a bold style for the label?

boldLabel

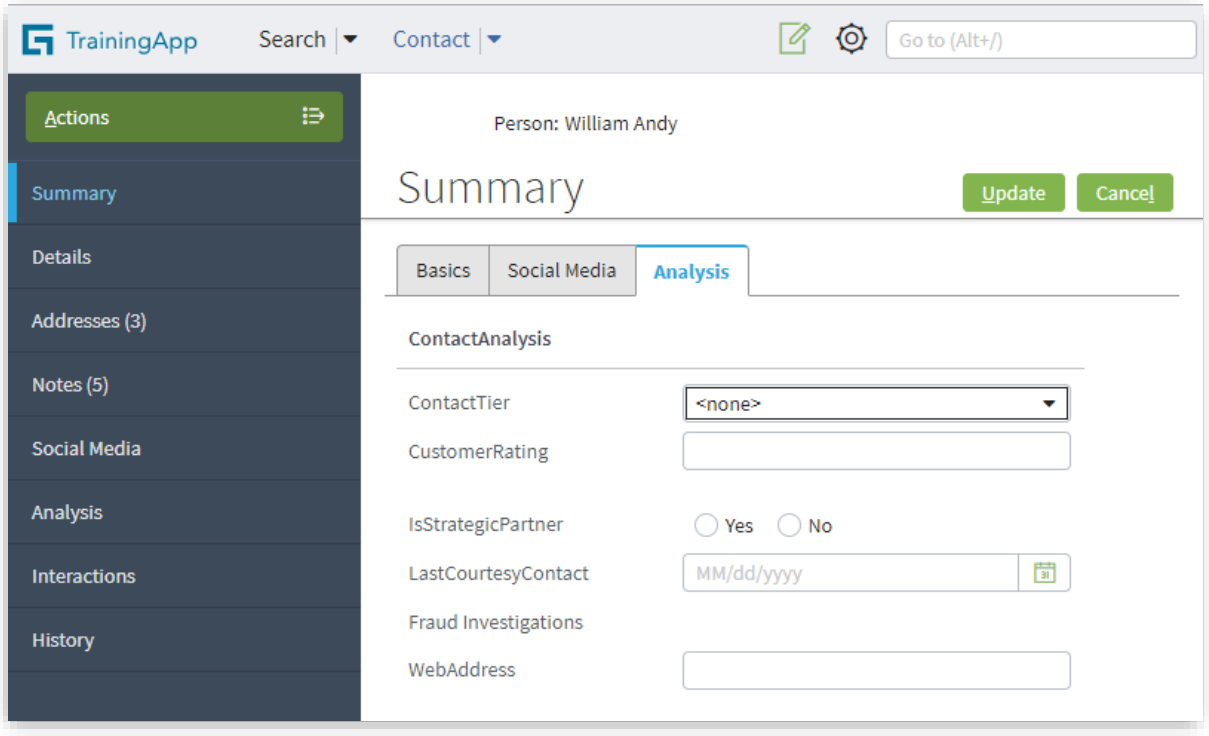
Requirement: When a user hovers over a Legacy Code text input widget with their mouse cursor, the user interface displays, "The code must come from the legacy claim system."

3. **Question:** What is the name of the attribute (property) for a TextInput that displays this message?

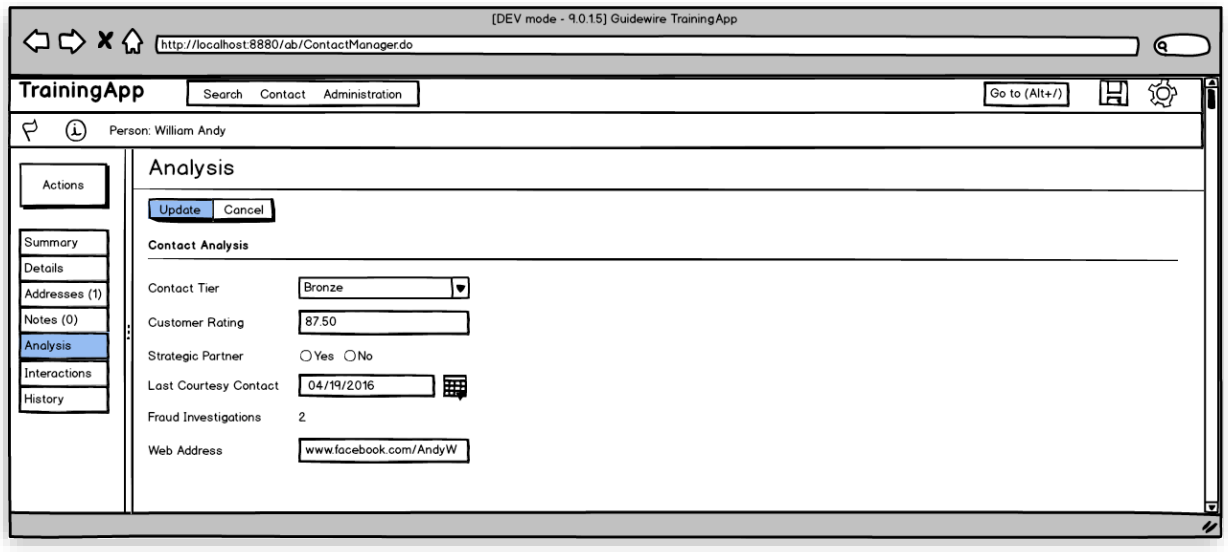
helpText

Lesson 6 Detail Views

In the *Extending Base Application Entities* lesson, we made certain data model changes to collect additional analytical information about the contact; and in the *Atomic Widgets* lessons we configured the Analysis tab to display those fields, so the users can view and edit them.



We received the following message from the Business Analysts: *"The Analysis tab on the Summary screen looks great. Nice job! We did some further analysis and concluded that we want to see the same UI screen layout when we navigate to the Analysis link in the sidebar too. (See wireframe below.) Please note that this functionality will be most likely extended in the future. If that happens, the UI layout must change at both places the same way."*



In this module, you will create a reusable Detail View Panel. Next, you will move widgets from an existing PCF file that you modified in a previous lab to your new file. You will then reference your Detail View Panel in various PCF files. Last, you will add a Toolbar with Edit Buttons to make the Analysis page editable.

6.1 Prerequisites

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

- Extending Base Application Entities
- Atomic Widgets

For this exercise, 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`.

6.2 Lab: Reusable Detail View Panel

As a configuration developer, you want to create a reusable Detail View Panel and reference it from different PCF files.

6.2.1 Create the reusable Detail View Panel

1. Open Guidewire Studio

a) From the TrainingApp folder, open a command window and start Guidewire Studio for TrainingApp. Note: Alternatively, use the Start TrainingApp Studio shortcut.

2. **Run or Debug the TrainingApp project**

a) Use the main menu, toolbar, or keystroke to run or Debug the server.

b) Verify that the Run or Debug Console tab shows ***** ContactManager ready *****.

3. **Create the traininglabs PCF Folder if it doesn't exist already**

a) In the Project View, select the pcf folder.

b) Create a new PCF Folder named `traininglabs` (one word, no spaces).

4. **Create the ABContactAnalysisDV PCF file**

a) Create the detail view panel PCF in the traininglabs PCF folder. Remember: You should enter ABContactAnalysis only as the name, because Studio automatically adds the DV suffix if you select Detail View Panel as the PCF file type.

b) Specify a root object of the type ABContact. Follow the recommended practice to name your variable. Remember: Use the Required variables tab. (No need to use the Variables tab for this exercise)

5. **Navigate to the Analysis tab in ABContactSummaryCV.pcf**
Note: This is the PCF file that you modified in the previous exercise.

a) Cut the whole Input Column - containing all the widgets - from the Analysis tab and paste it in the ABContactAnalysisDV that you created in the previous step.

Hint

Required variables vs variables

The required variables tab defines the input parameters for a container, while the variables tab defines local variables to store values temporarily. (E.g. storing the result of an expensive function call to use it at multiple places in the PCF.)

6.2.2 Reference the new Detail View Panel on the Summary screen's Analysis Card

1. **Go back to ABContactSummaryCV.pcf**

a) Delete the existing inline Detail Panel Widget.

2. **Add a PanelRef widget**

a) In the Toolbox, select the PanelRef widget.

b) Add the Panel Ref widget to the AnalysisCard in the ABContactSummaryCV.

3. **Configure the PanelRef widget**
   a) Reference the ABContactAnalysisDV in the Panel Ref.

Hint

Panel Ref

Include a panel (such as a DetailView, ListView, PanelSet or CardView, etc.) and (optionally) supply it with title, toolbar or instructional text. The `def` property must be configured using the following format:

```
NameOfThePanelToBeIncluded( argument1, argument2, … )
```

The number of arguments depends on the number of required variables the included panel has.

6.2.3 Reference the new Detail View Panel on the Analysis page

1. **Navigate to ABContactAnalysisPage.pcf**
   a) Delete the existing inline Detail Panel Widget.
2. **Add a PanelRef widget**
   a) In the Toolbox, select the PanelRef widget.
   b) Add the Panel Ref widget to the ABContactAnalysisPage.
3. **Configure the PanelRef widget**
   a) Reference the ABContactAnalysisDV in the Panel Ref.

6.2.4 Verification

Activity

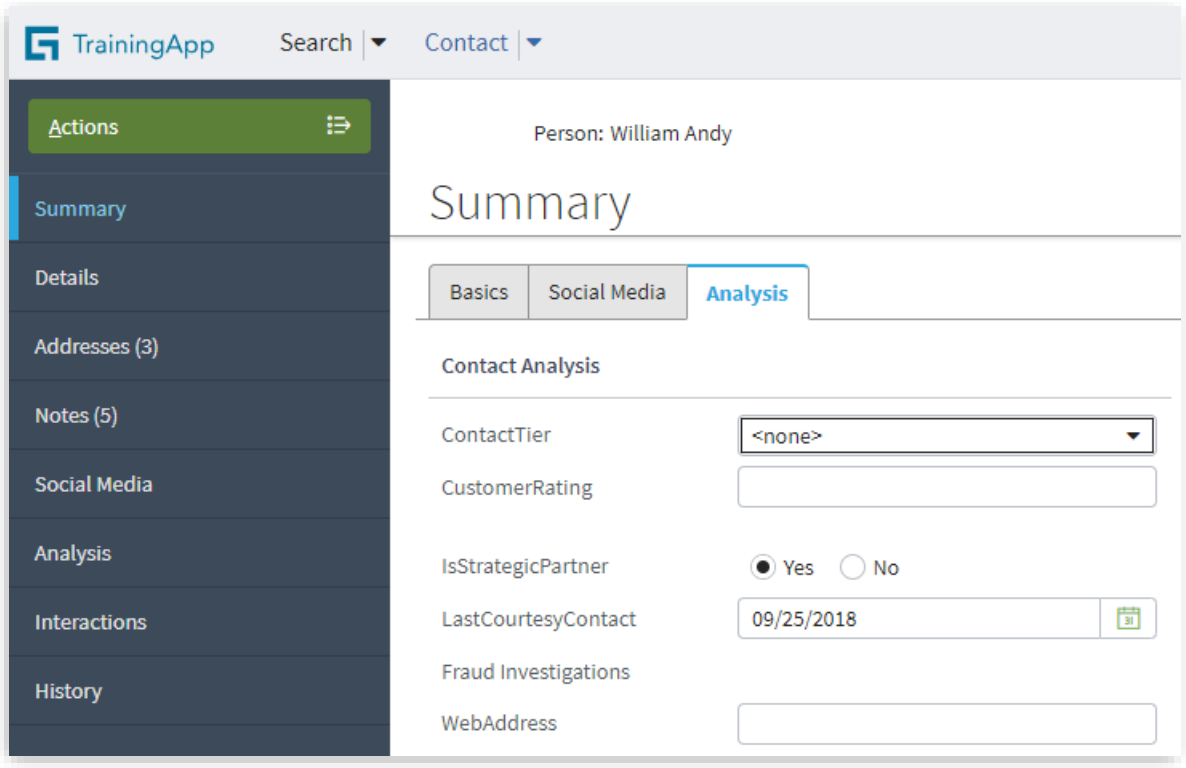
Verify the work you have done

1. **Log in to TrainingApp**
   a) Log in as Alice Applegate.
2. **Reload the PCF changes**
   a) In TrainingApp, reload the changes to the PCF file(s).
3. **View the Analysis card for William Andy**
   a) Search for William Andy.

b) In the search results list view, navigate to the William Andy contact.

c) Click the Analysis card tab.

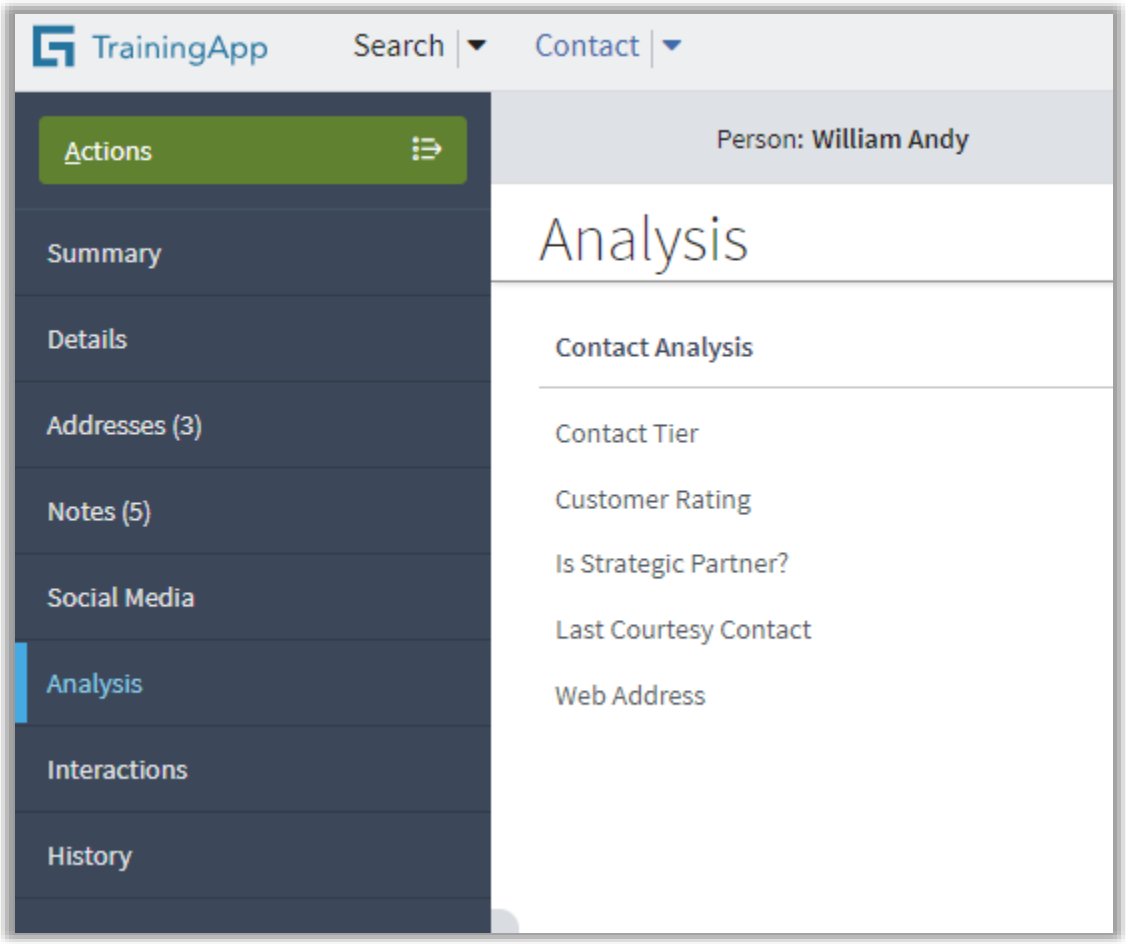
d) Verify that you see the Contact Analysis detail view panel.



4. View the Analysis page for William Andy

a) In the sidebar menu, click Analysis.

b) Verify that you see the Contact Analysis detail view panel.



Icon for stop, consisting of two crossed flags marking the end of a lab.

Stop

6.3 Lab: Toolbar and edit buttons

As a configuration developer, you want to create a toolbar with edit buttons so that end users can view, update, and create object data in the application UI.

1. **Navigate to ABContactAnalysisPage**
2. **Add a Toolbar with Edit Buttons**
   a) In the canvas, add a Toolbar with Edit Buttons (Edit|Update|Cancel) to the top-level container widget.

6.3.1 Verification

Activity

Verify the work you have done

1. **Log in to TrainingApp**
   a) Log in as Alice Applegate.
2. **Reload the PCF changes**
   a) In TrainingApp, reload the PCF file changes.
3. **Edit the Analysis page for William Andy**
   a) Search for William Andy and select the search result.
   b) In the sidebar menu, click Analysis.
   c) Edit the Contact Analysis details for William Andy. Fraud Investigations should not be editable.
   d) Click Update.

6.4 Lab: Write it down

1. **Question**: In the sidebar menu, click Summary for the William Andy contact. Next, select the Analysis card. Click Edit. Why is it possible to edit the Contact Analysis details on the Summary page?

Stop

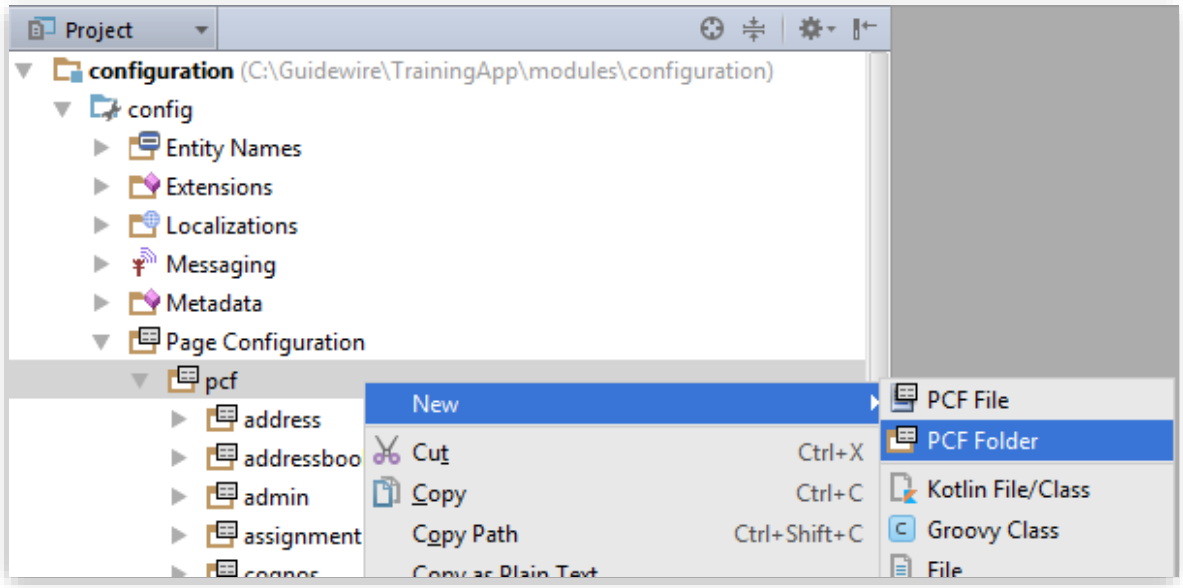
6.5 Lab Solution: Reusable Detail View Panel

Solution

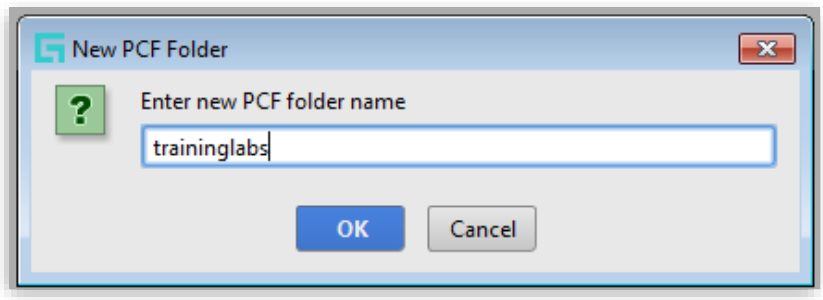
1. **Open Guidewire Studio**
   a) See the Introduction to Guidewire Configuration lesson, if needed

2. **Run or Debug the TrainingApp project**
   a) See the Introduction to Guidewire Configuration lesson, if needed.

3. **Create the traininglabs PCF Folder if it doesn't exist already**
   a) Right click on the pcf node and select `New` → `PCF Folder` from the context menu.

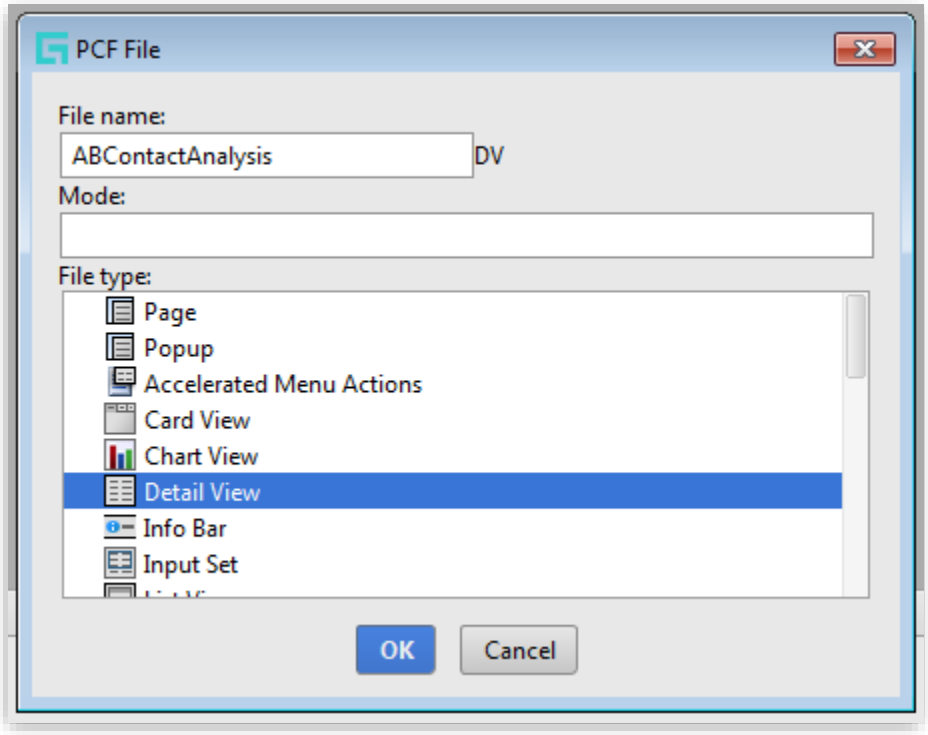


4. **Enter the traininglabs name in the popup – no spaces**

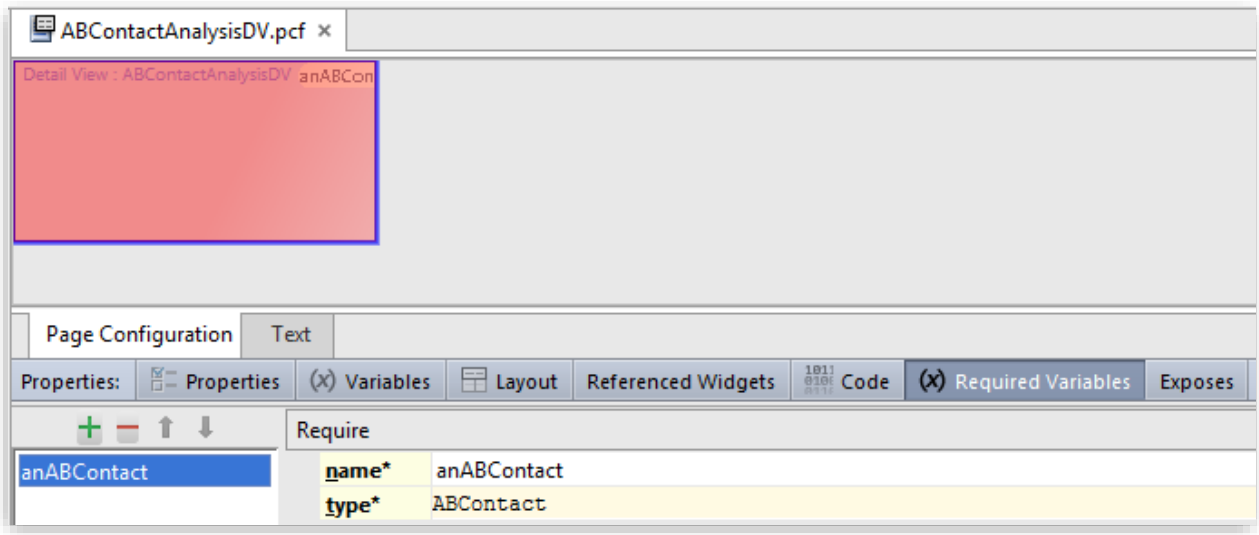


5. Create the ABContactAnalysisDV PCF file

a) Right click on the training labs folder and select `New` → `PCF File` from the context menu. Specify the file name and select the file type.

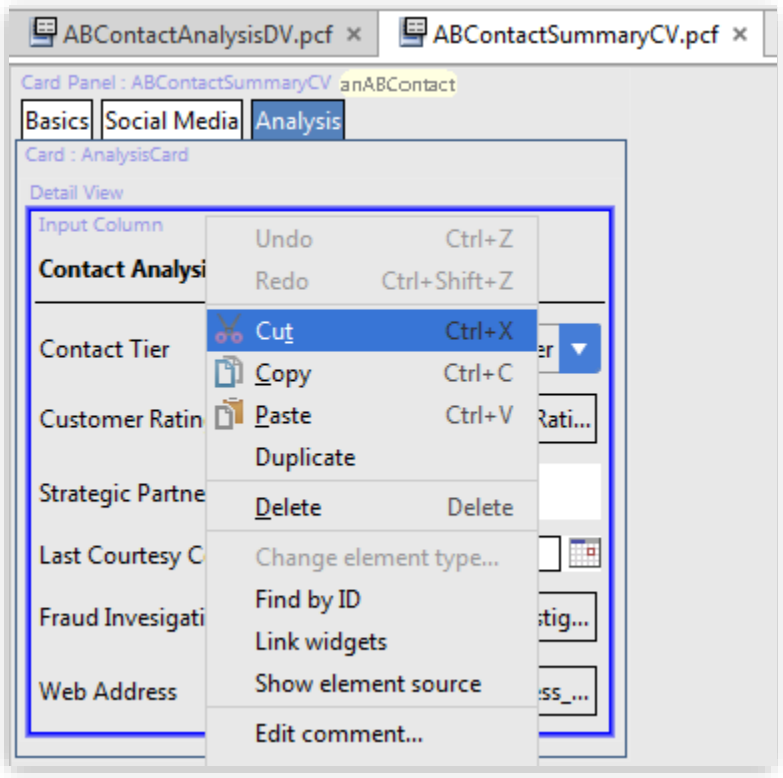


b) Specify a root object of the type ABContact.

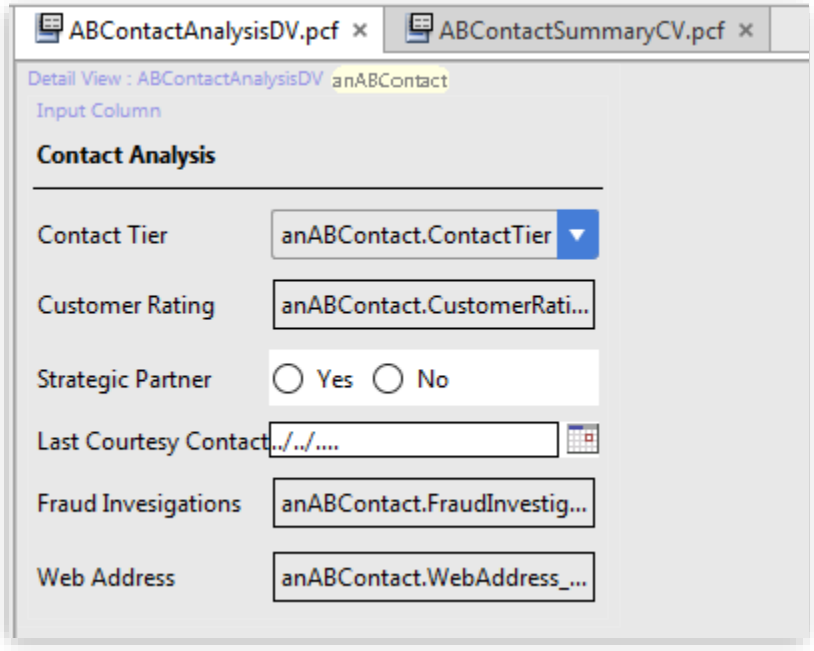


6. Navigate to the Analysis tab in ABContactSummaryCV.pcf

a) Cut the whole Input Column - containing all the widgets - from the Analysis tab…



b) …and paste it in the ABContactAnalysisDV that you created in the previous step.

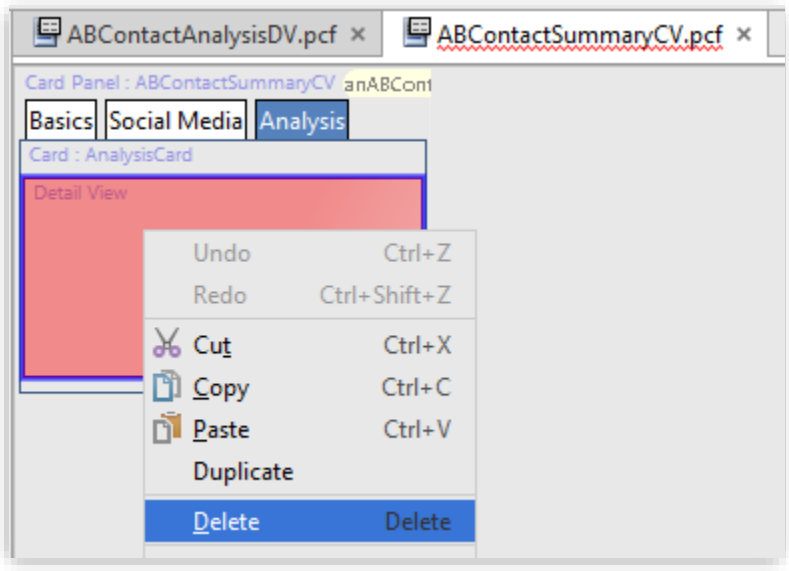


6.6 Lab Solution: Reference the new Detail View Panel on the Summary screen's Analysis Card

Solution

1. Go back to ABContactSummaryCV.pcf

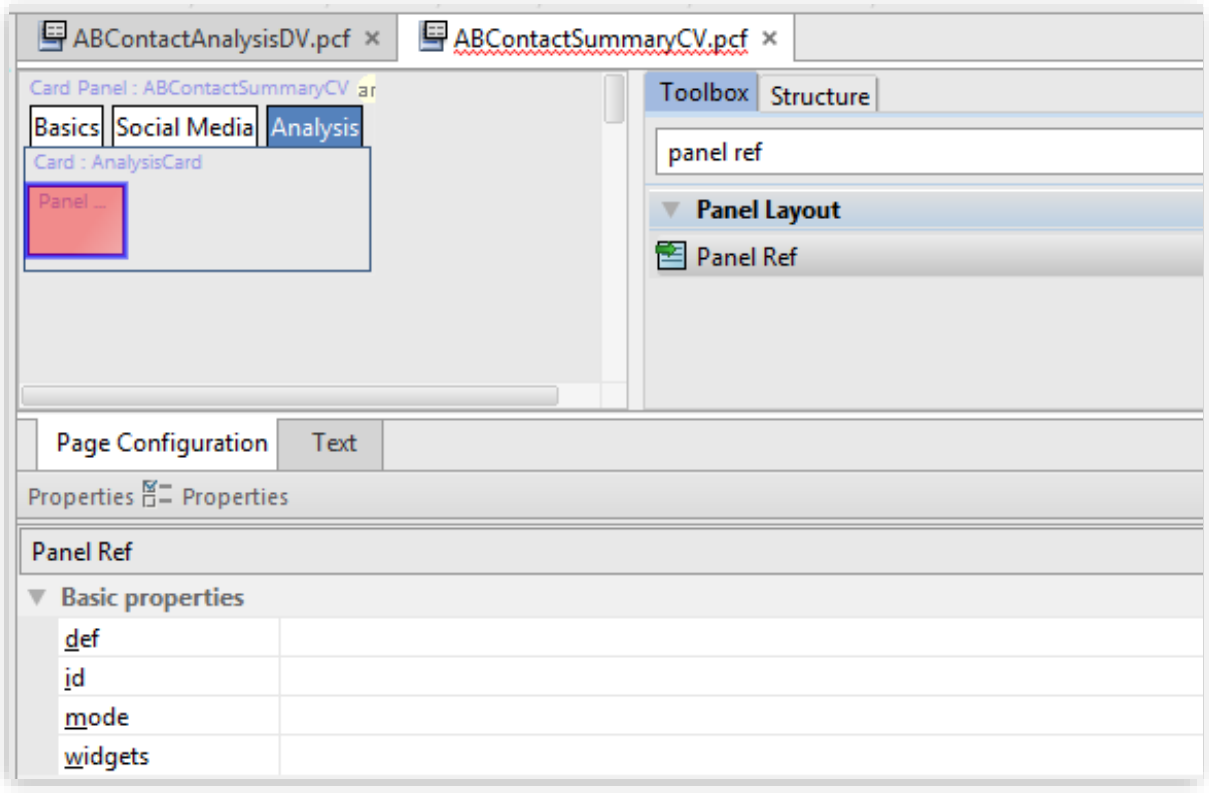
a) Delete the existing inline Detail Panel Widget



2. Add a PanelRef widget

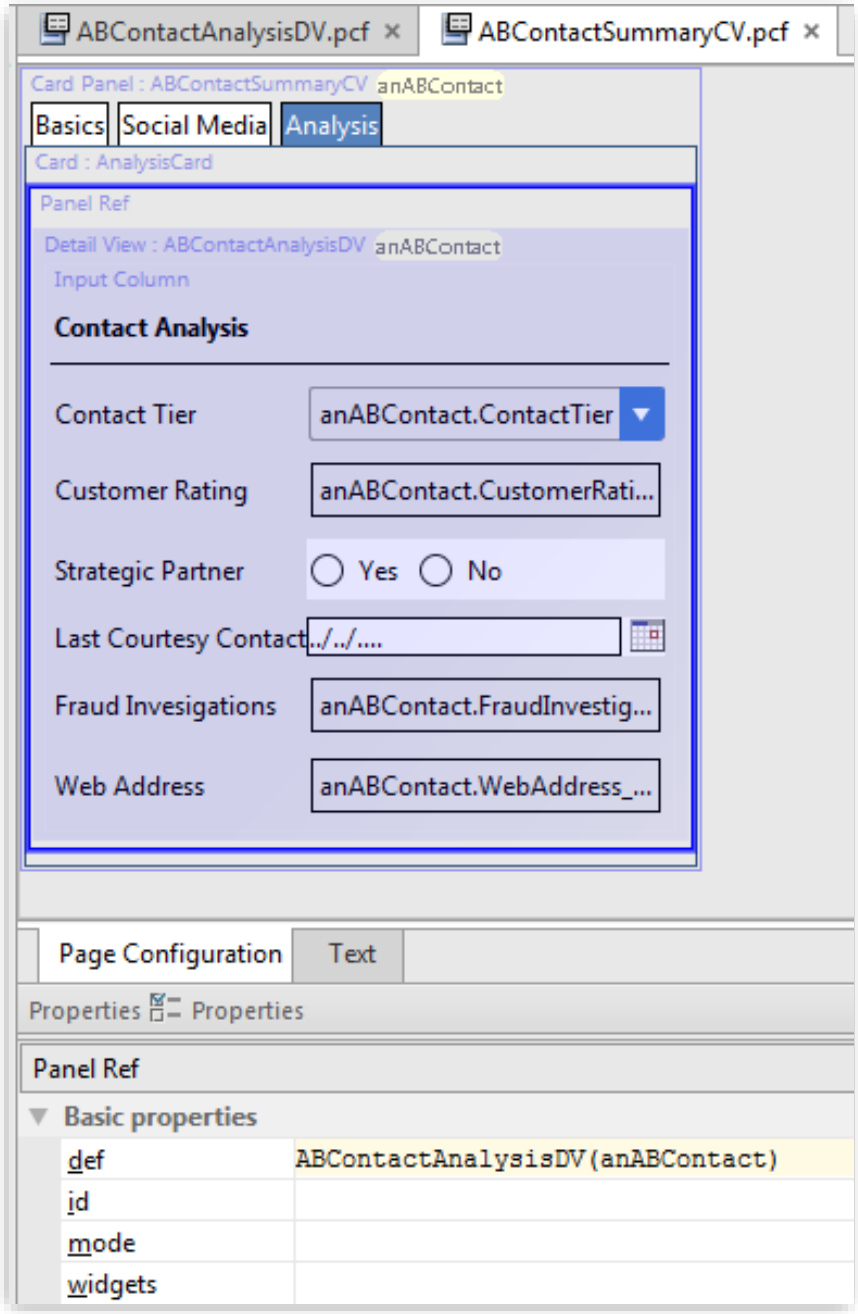
a) In the Toolbox, select the PanelRef widget.

b) Add the Panel Ref widget to the AnalysisCard in the ABContactSummaryCV.



3. Configure the PanelRef widget

a) Reference the ABContactAnalysisDV in the Panel Ref.


ABContactAnalysisDV.pcf
ABContactSummaryCV.pcf
Card Panel : ABContactSummaryCV anABContact
Basics
Social Media
Analysis
Card : AnalysisCard
Panel Ref
Detail View : ABContactAnalysisDV anABContact
Input Column
Contact Analysis
Contact Tier
anABContact.ContactTier
Customer Rating
anABContact.CustomerRati...
Strategic Partner
Yes
No
Last Courtesy Contact
../../....
Fraud Invesigations
anABContact.FraudInvestig...
Web Address
anABContact.WebAddress_...
Page Configuration
Text
Properties
Properties
Panel Ref
Basic properties
def
ABContactAnalysisDV(anABContact)
id
mode
widgets

6.6.1 Solution: Reference the new Detail View Panel on the Analysis page

Solution

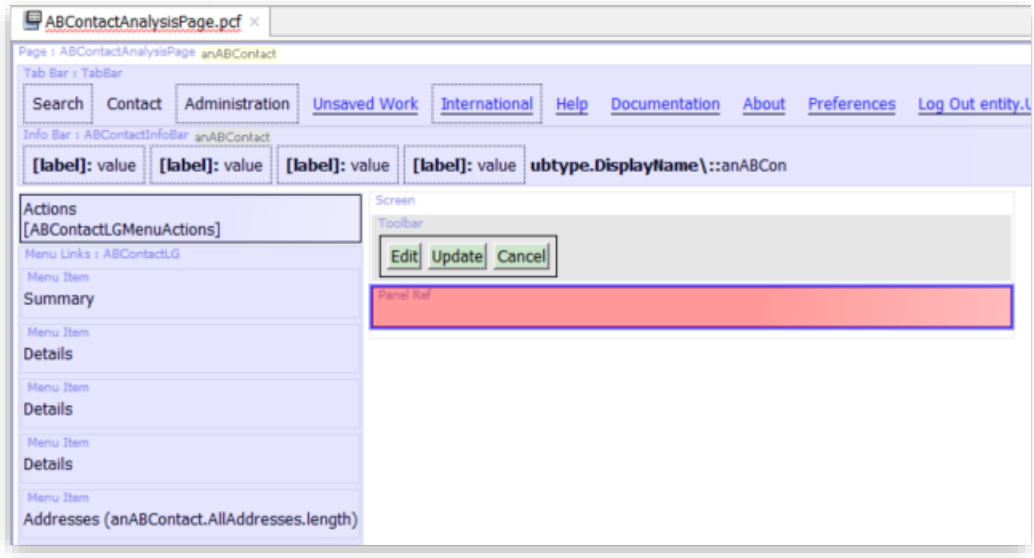
1. Navigate to ABContactAnalysisPage.pcf

a) Delete the existing inline Detail Panel Widget.

2. Add a PanelRef widget

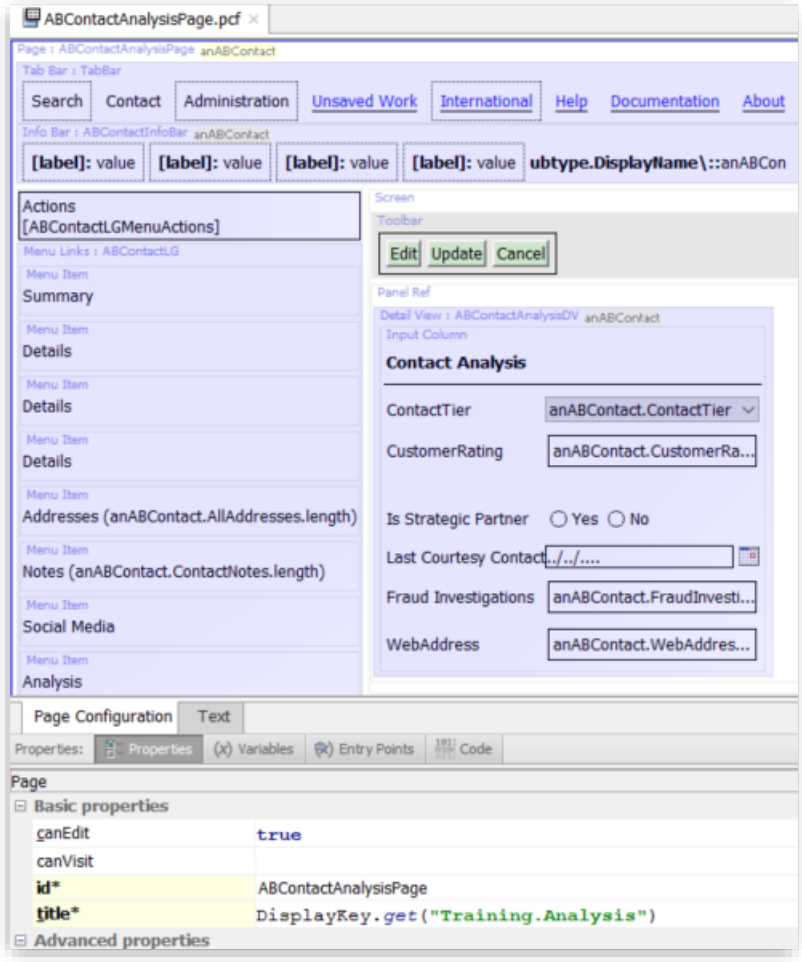
a) In the Toolbox, select the PanelRef widget.

b) Add the Panel Ref widget to the ABContactAnalysisPage.



3. Configure the PanelRef widget

a) Reference the ABContactAnalysisDV in the Panel Ref.

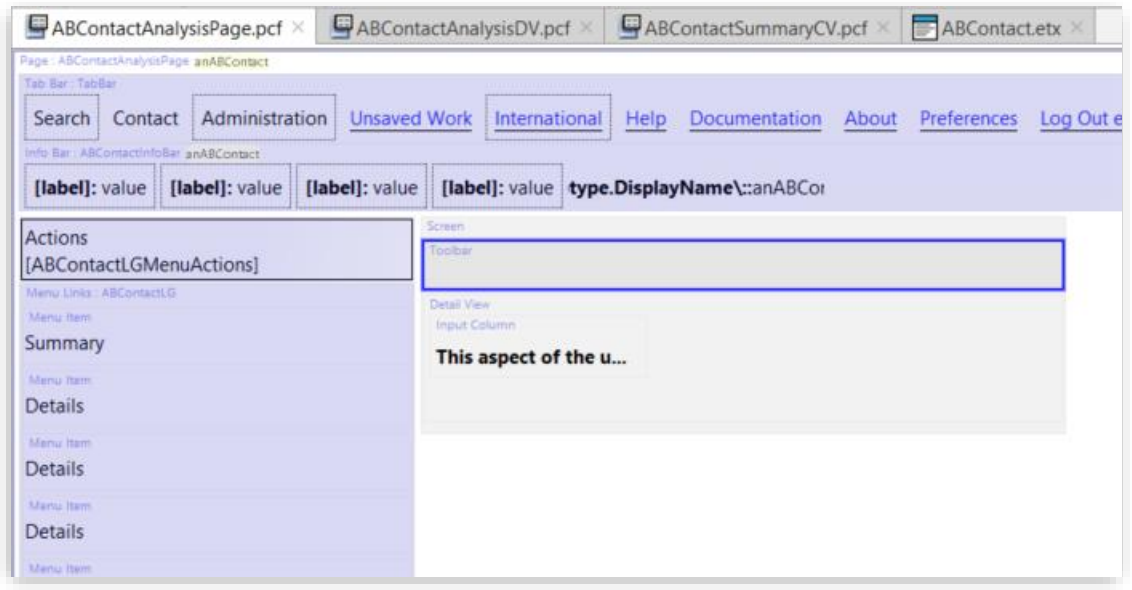


6.7 Lab Solution: Toolbar and edit buttons

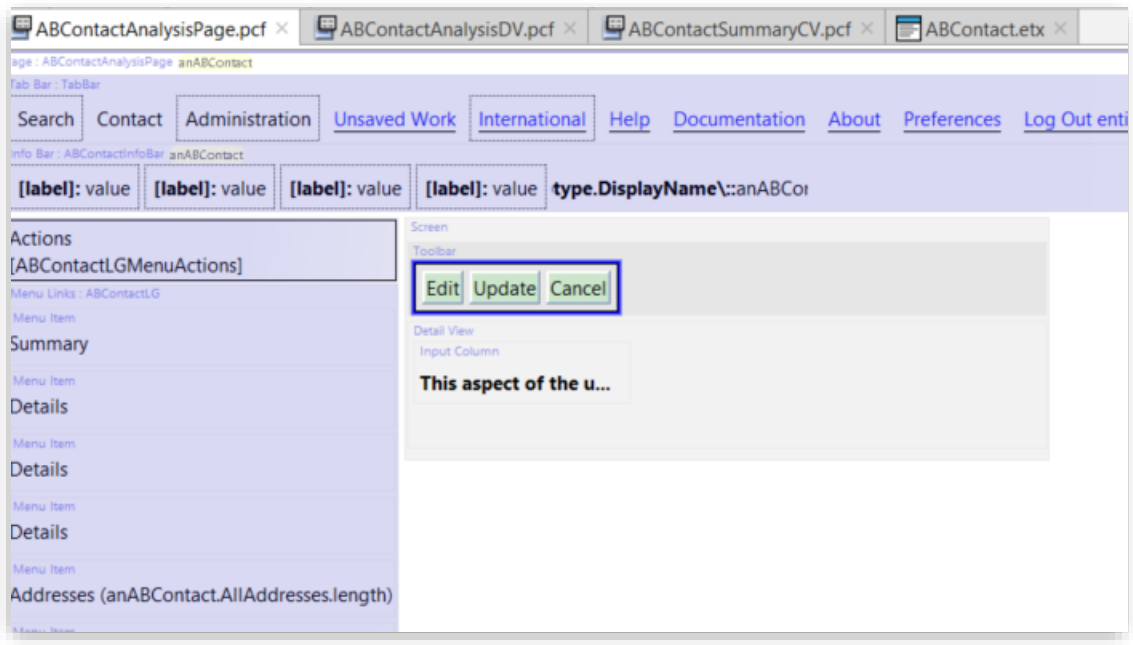
Solution

1. **Navigate to ABContactAnalysisPage**
2. **Add a Toolbar with Edit Buttons**

   a) In the canvas, add a Toolbar



b) Add Edit Buttons



6.8 Lab Solution: Write it down

Solution

1. **Question: In the sidebar menu, click Summary for the William Andy contact. Next, select the Analysis card.**

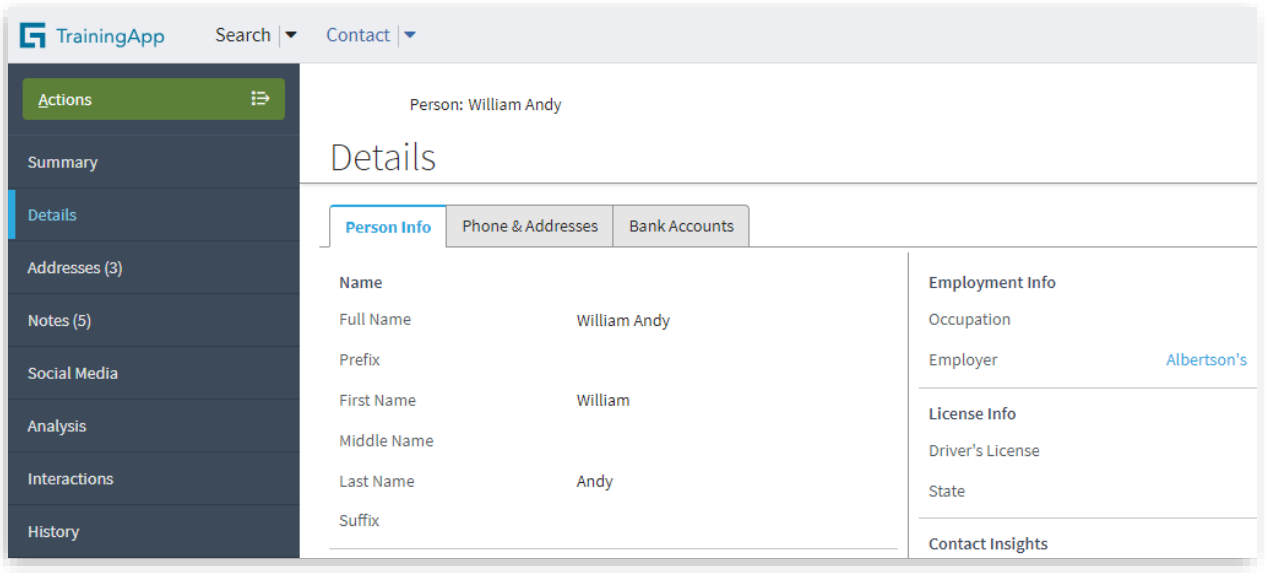
   a) Click Edit. Why is it possible to edit the Contact Analysis details on the Summary page?

Because ABContactSummaryScreen in ABContactSummaryPage already has a Toolbar and Edit Buttons.

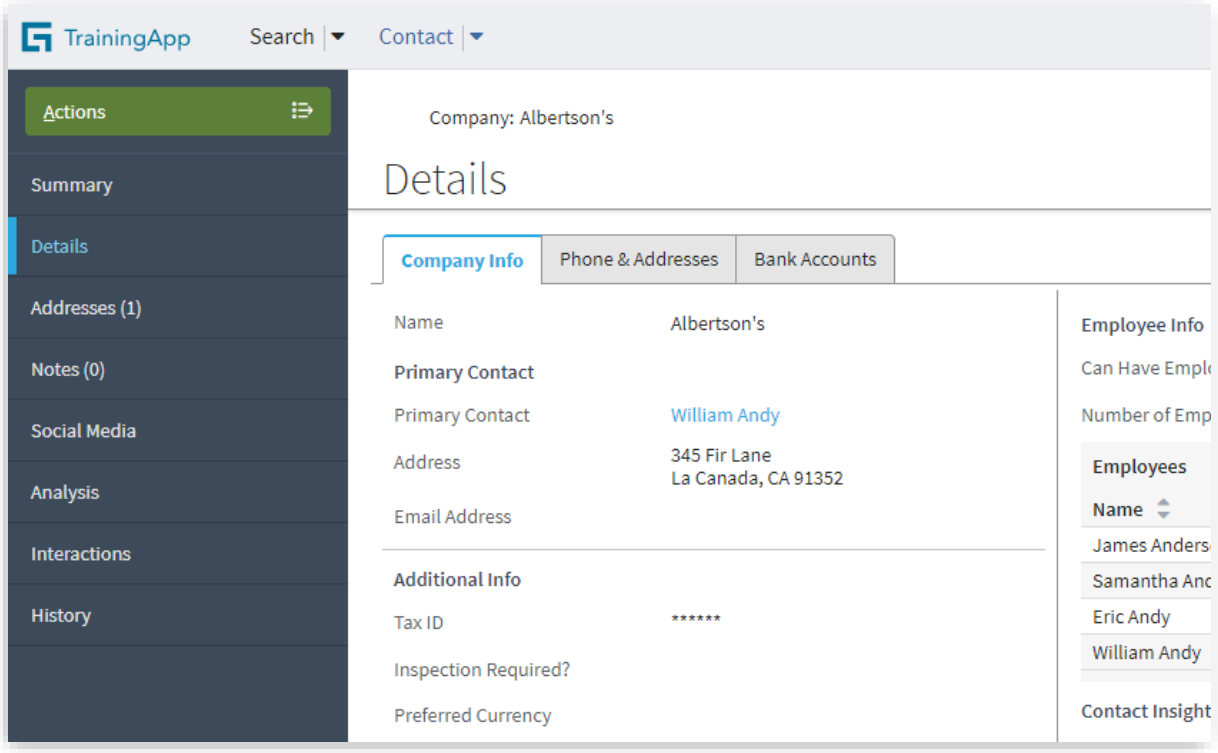
Stop

Lesson 7 Introduction to Locations

The business analysts of an insurance company sent the following user stories:



Location User Story #1: *"Let's say we look at a person, e.g. William Andy, and we want to know more details about his employer. Currently the only way to navigate to the employer is searching for it based on the company name. We want to make the employer widget clickable on the person's Person Info card on the Details page. When the user clicks on it, it should navigate to the Company Info card on the Details page of the company."* – Insurance company business analysts



Location User Story #2: *"When we look at a company, e.g. Albertson's, we can see that it has a Primary Contact. However, currently the only way to know more about that contact is searching for him/her based on the name. We want to make the name of the primary contact navigable, so if the user clicks on it the system will display the Person Info card on the Details page of the person."* **–** Insurance company business analysts

7.1 Prerequisites

For this exercise, 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.

7.2 Lab: Location user story #1

"We want to make the employer widget clickable on the person's Person Info card on the Details page. When the user clicks on it, it should navigate to the Company Info card on the Details page of the company." – Insurance company business analyst

7.2.1 Write it down

1. **Question: What is the name of the destination PCF file? What is the entry point signature?**

2. **Question: What is the name of the source PCF file?**

Stop

7.2.2 Configuration

1. **Open Guidewire Studio**
   a) From the TrainingApp folder, open a command window and start Guidewire Studio for TrainingApp.
   b) Alternatively, use the Start TrainingApp Studio shortcut.
2. **Run or Debug the TrainingApp project**
   a) Use the main menu, toolbar, or keystroke to run or Debug the server.
   b) Verify that the Run or Debug Console tab shows `***** ContactManager ready *****`.
3. **Modify the Employer widget to link to the Employer's Details page**

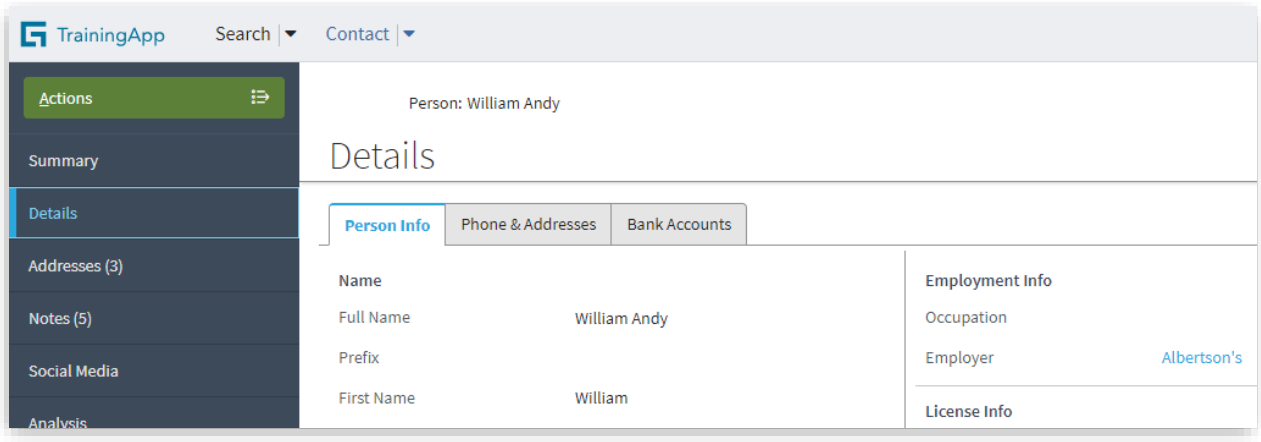


Figure 1 Source

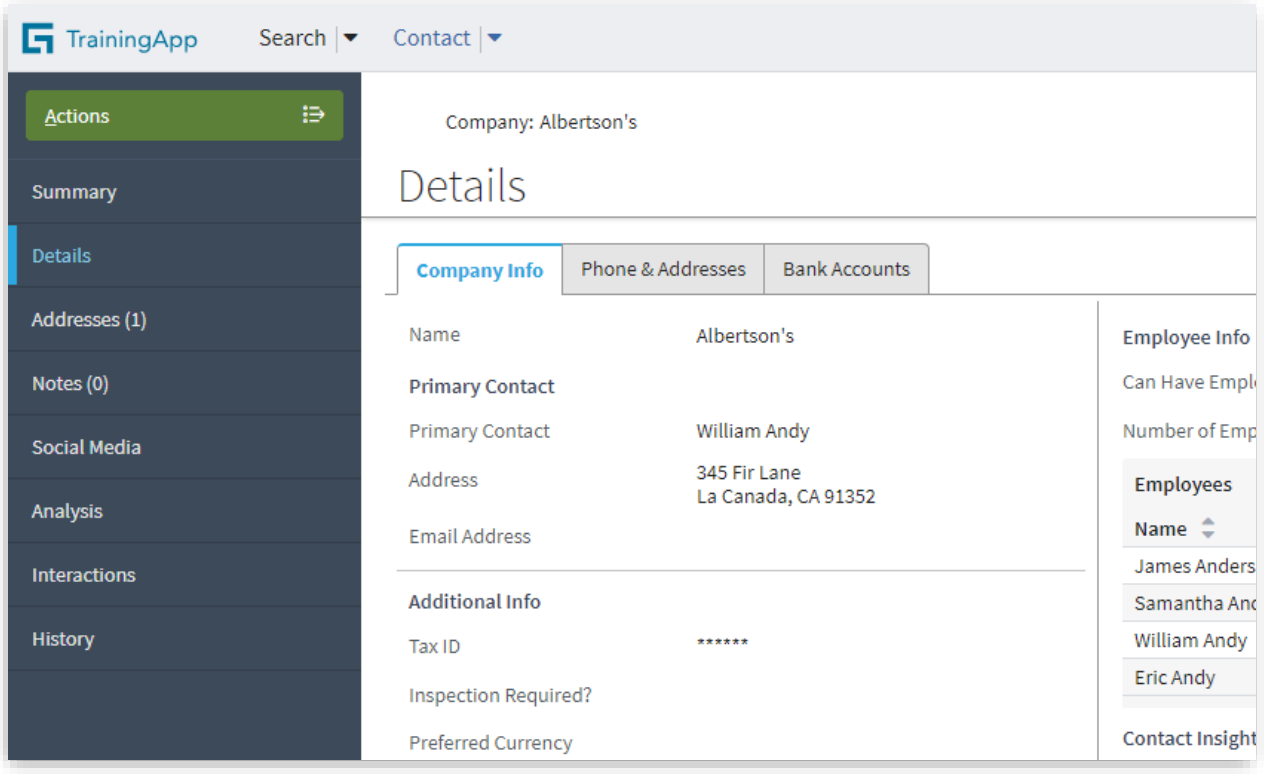


Figure 2 Destination

Hint

Troubleshooting “Widget is not clickable after configuration”

If you configured the source widget’s `action` property and reloaded the PCF changes, but the widget is still not clickable: verify that you are passing in the right data to the destination page. For example, if you wrote this:

```
ABCompanyDetailsPage.go(anABContact)
```

in the widget’s action property, then use the `ALT + SHIFT + W` shortcut in the browser to verify what is data in the anABContact variable. If it is not the employer, then use the Data Dictionary to find out how you can get access to an ABPerson’s Employer.

Guidewire API

The `action` property

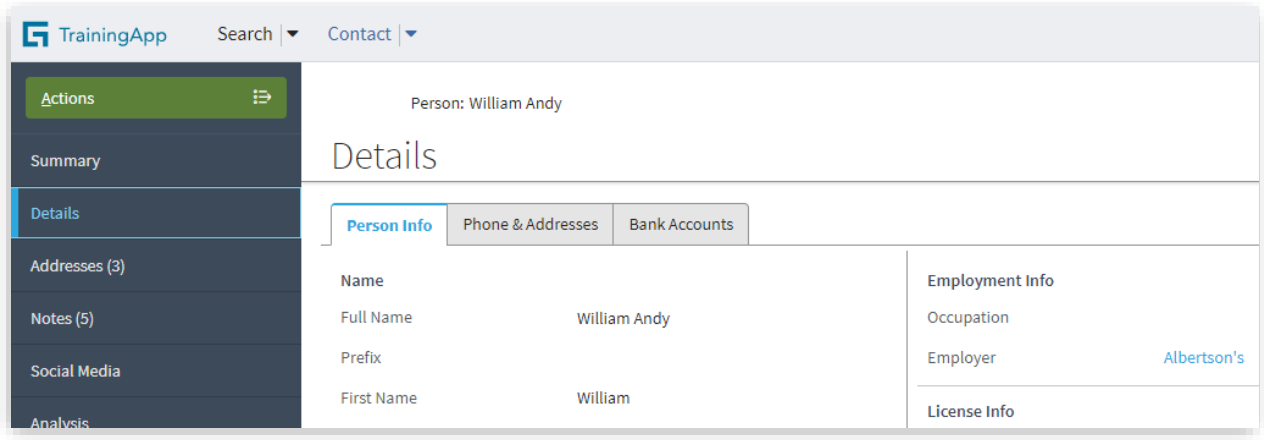
Most atomic widgets have the `action` property. This can be set to a Gosu statement. When set, the atomic widget becomes clickable and when clicked, the Gosu statement is executed.

7.2.3 Verification

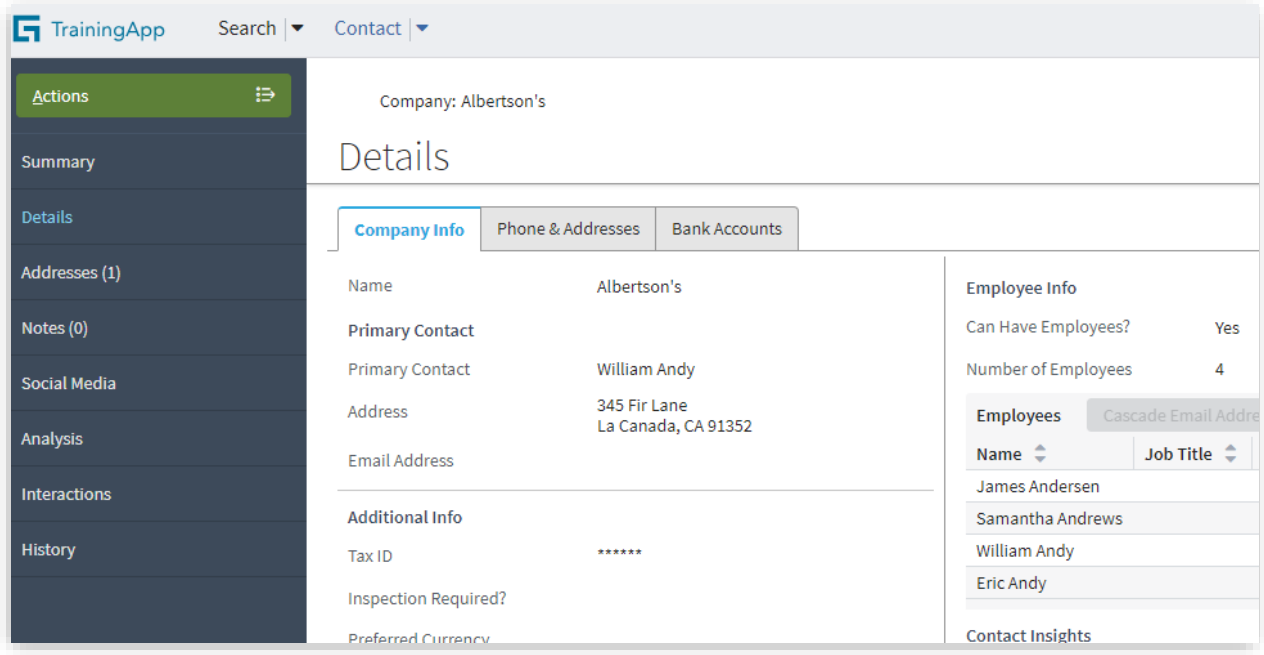
Activity

Verify the work you have done

1. **Reload the PCF changes**
   a) In TrainingApp, reload the changes to the PCF file(s) and display keys.
2. **Go to the Details page for William Andy**
   a) Click on edit and select Albertson's as the employer
   b) Click on update
   c) Verify that Albertson's is displayed as a link



   d) Verify if you click on the Albertson's link the application displays the Details page of Albertson's



7.3 Lab: Location user story #2

7.3.1 Write it down

"We want to make the name of the primary contact navigable, so if the user clicks on it the system will display the Person Info card on the Details page of the person." – Insurance company business analyst

1. **Question: What is the name of the destination PCF file? What is the entry point signature?**

2. **Question: What is the name of the source PCF file?**

Stop

7.3.2 Configuration

1. **Modify the Primary Contact widget so that it links to the person's Details page.**

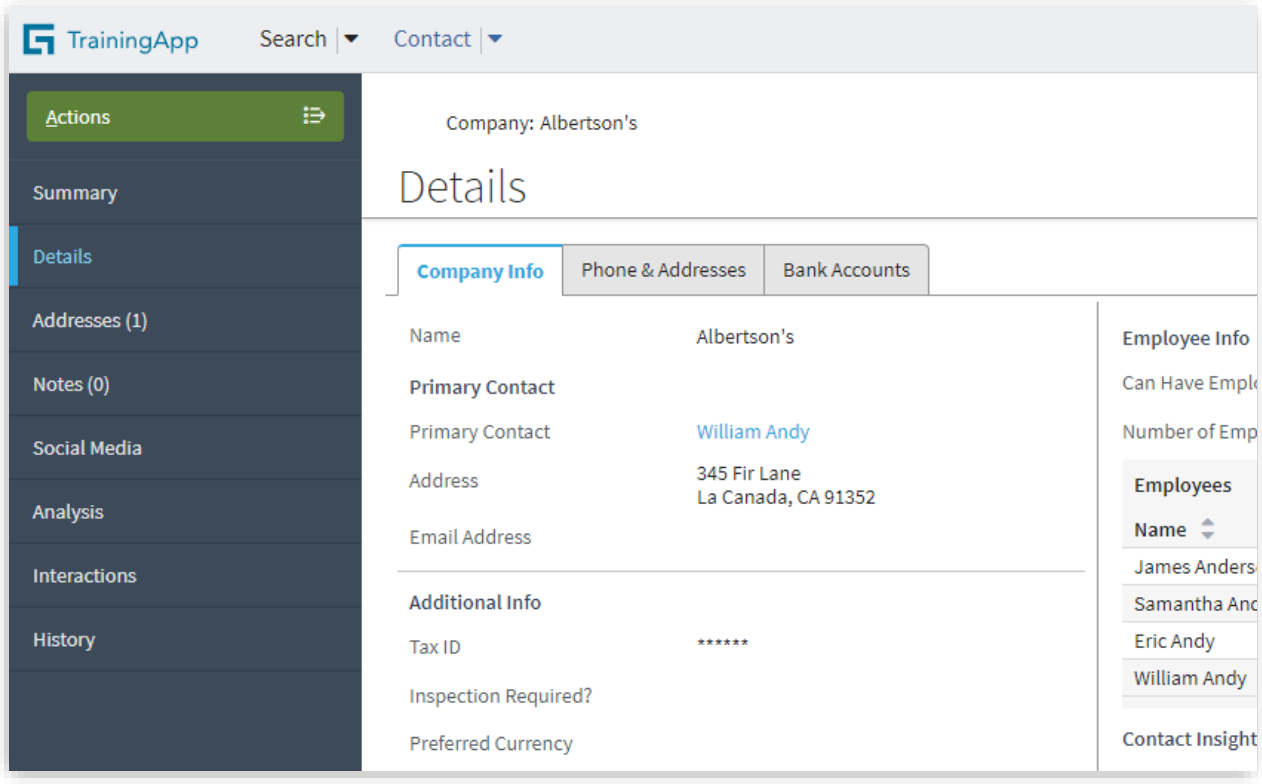


Figure 1 Source

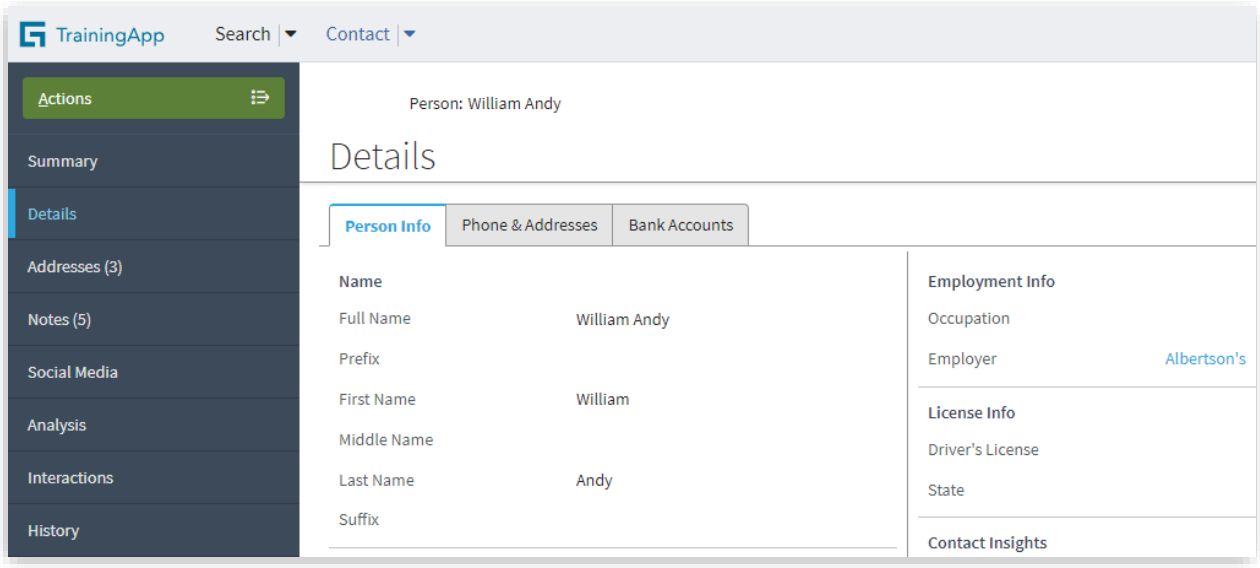


Figure 2 Destination

Hint

Troubleshooting “You do not have the permission required to perform this action: ABCompanyDetailsPage”

Hint

In InsuranceSuite 9 and earlier, if the company does not have a primary contact, there is no visible link, but the widget is still clickable. If you click on the empty link, TrainingApp will throw an error. If you are working in one of these versions, you must configure the `available` property to fix this:

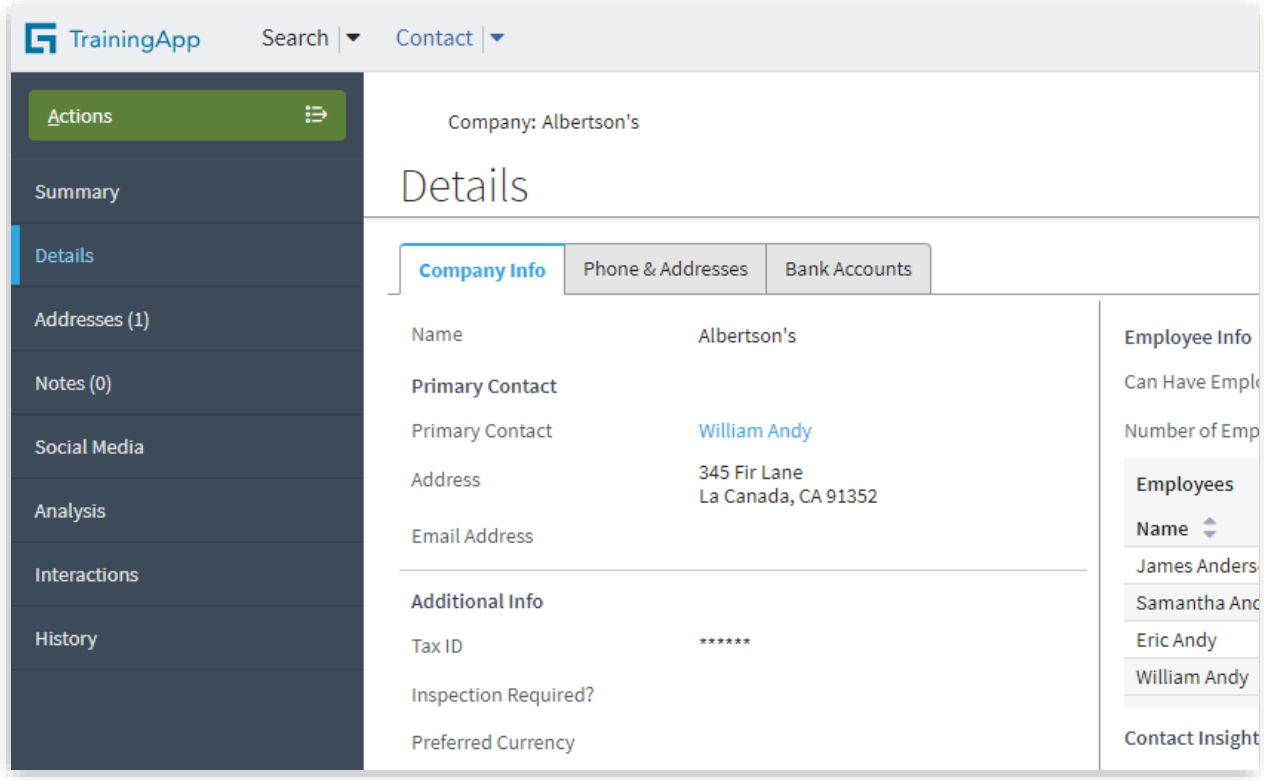
```
(anABContact as ABCompany).PrimaryContact != null or CurrentLocation.InEditMode
```

7.3.3 Verification

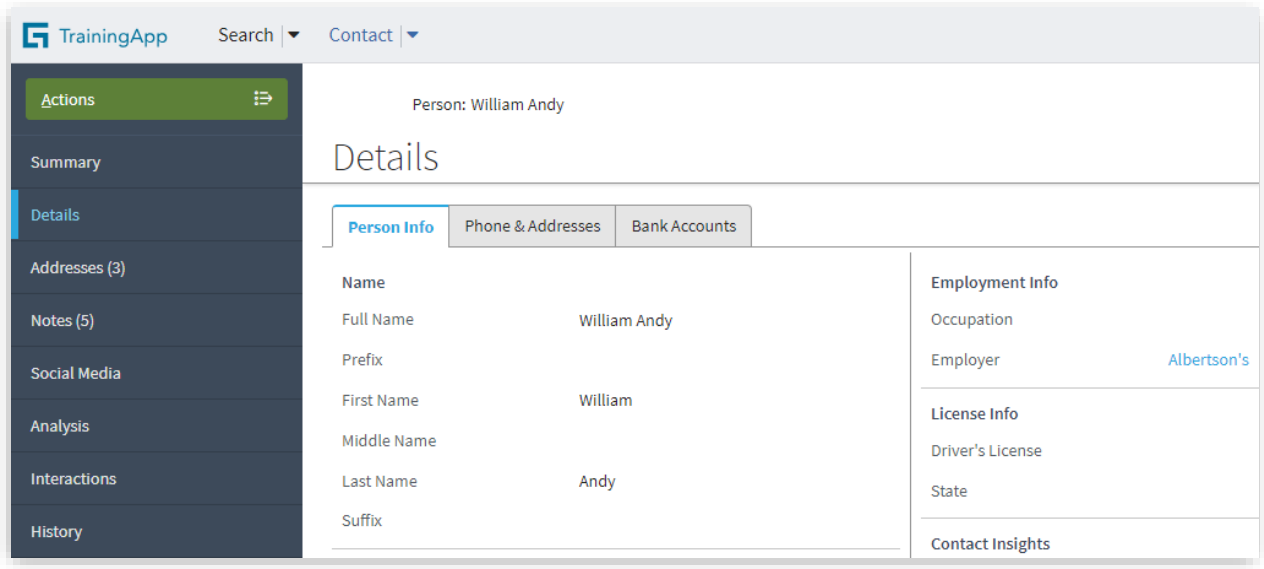
Activity

Verify the work you have done

1. **Reload the PCF changes**
   a) In TrainingApp, reload the changes to the PCF file(s) and display keys.
2. **Go to the Details page for Alberton’s**
   a) If William Andy is not already the primary contact than click on Edit, select William Andy from the drop down list and click on Update.
      - Note#1: You can select a Primary Contact only from the list of employees.
      - Note#2: If the list of employees is empty, then navigate to William Andy’s Details Page and set Albertson’s as his employer.
   b) Verify that William Andy is displayed for the Primary Contact as a link



c) Verify if you click the link on the Primary Contact the application displays the Details page of William Andy



Stop

7.4 Lab Solution: Location user story #1

Solution

7.4.1 Write it down

1. Question: What is the name of the destination PCF file? What is the entry point signature?

ABCompanyDetailsPage.pcf

ABCompanyDetailsPage(anABContact : ABContact)

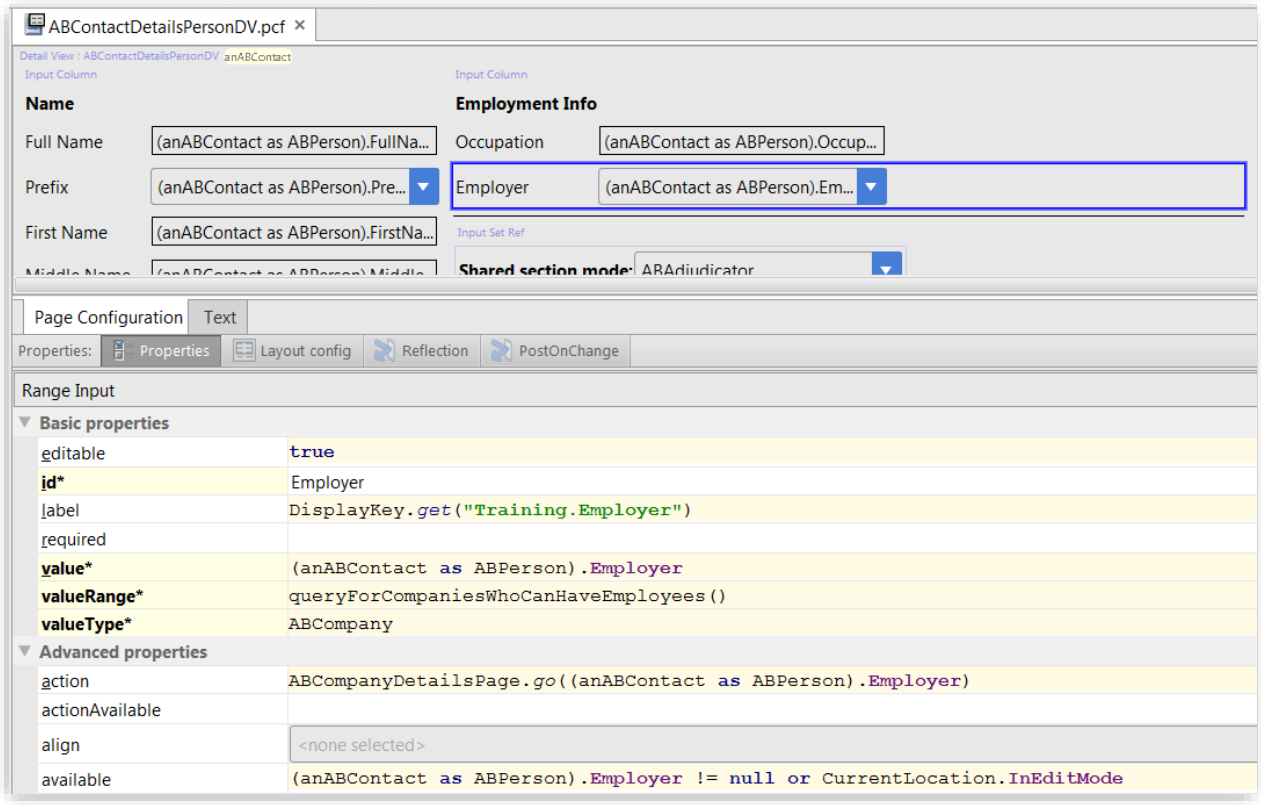
2. Question: What is the name of the source PCF file?

ABContactDetailsPersonDV.pcf

7.4.2 Configuration

1. Modify the Employer widget's action property in ABContactDetailsPersonDV.pcf:

```
ABCompanyDetailsPage.go((anABContact as ABPerson).Employer)
```



7.5 Lab Solution: Location user story #2

Solution

7.5.1 Write it down

1. **Question: What is the name of the destination PCF file? What is the entry point signature?**

Name: ABPersonDetailsPage.pcf, **Entry point:** ABPersonDetailsPage(anABContact : ABContact)

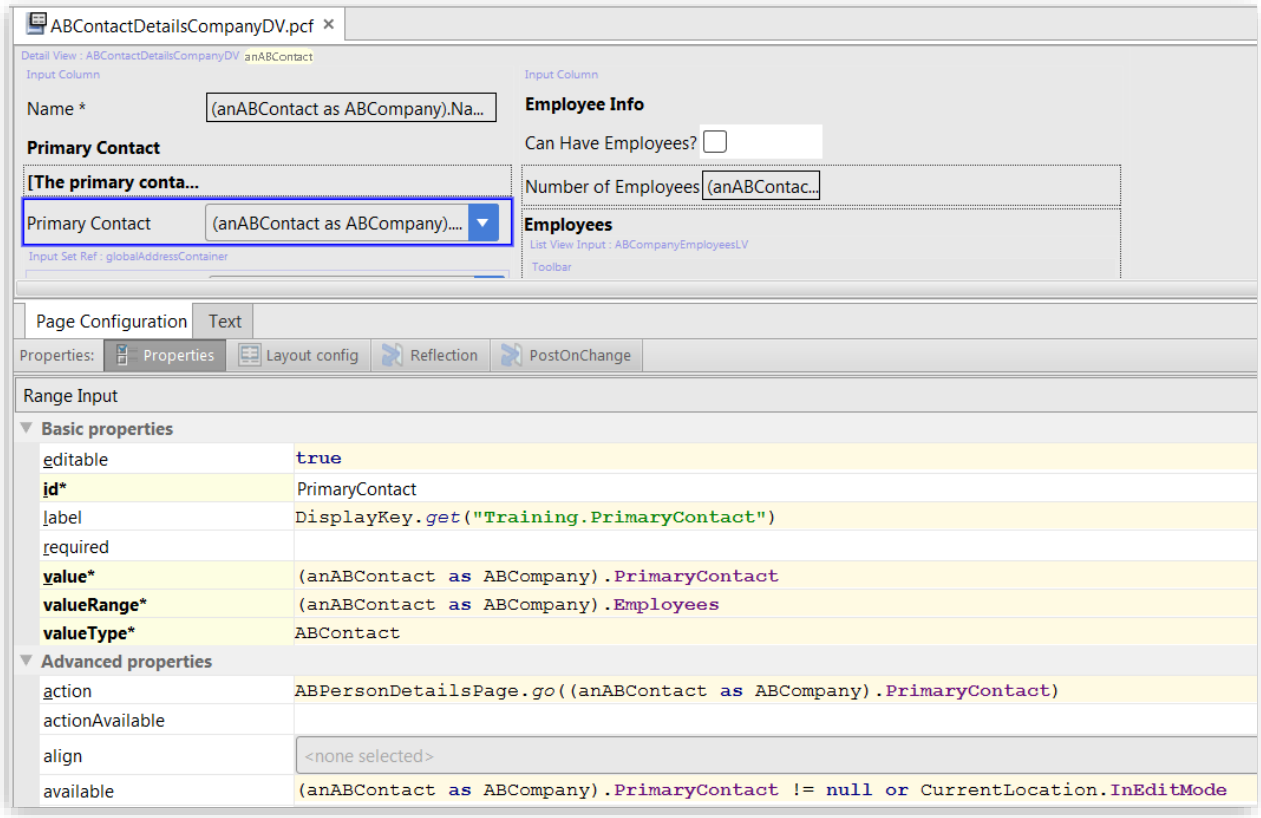
2. **Question: What is the name of the source PCF file?**

ABContactDetailsCompanyDV.pcf

7.5.2 Configuration

1. Modify the Primary Contact widget's action property in ABContactDetailsCompanyDV.pcf:

```
ABPersonDetailsPage.go((anABContact as ABCompany).PrimaryContact)
```



Lesson 8 Introduction to Gosu

As a configuration developer, you have to be familiar with the basic Gosu syntax and language features. In this lab, you will write Gosu code in Gosu Scratchpad to print to the debug console details about specific contacts in TrainingApp.

8.1 Prerequisites

For this exercise, 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`.

8.2 Lab: Coding with Gosu

In this exercise, you find the PublicIDs for various contacts. You will use these PublicIDs to retrieve contact objects in Gosu Scratchpad later in this exercise.

Guidewire API

PublicID

PublicID is a field on almost every Guidewire data model entity that stores a unique identifier value. Unlike the "ID" field, PublicID is a writeable string. It is usually used to identify business objects as they are known to external systems.

8.2.1 Write it down

1. **Question: Navigate to the Summary page of the following company: Burlingame Saab. On the Summary page, what is the Public ID for Burlingame Saab?**

2. **Question: Navigate to the Summary page of the following doctor: Rebecca Stevens. On the Summary page, what is the Public ID for Rebecca Stevens?**

8.2.2 Configuration

In the second part of the exercise, you will open Gosu Scratchpad. Then, you will write and run Gosu code to print to the debug console details about specific contacts in TrainingApp.

1. **Open Gosu Scratchpad in Studio**
   a) Verify that the *Run in Debug Process* icon is available in Gosu Scratchpad.
2. **Write Gosu code**
   a) Use the `trainingapp.base.QueryUtil.findContact()` method to retrieve the contact using the PublicID of the contact.
   b) Write Gosu code that prints to the debug console details about the contact based on the following logic:
      - If the contact found for the PublicID
        - The display name and create date of the contact

          Format: <contactname> was created <date>.
        - If the contact has a Primary Address, the State of the primary address

          Format: Primary address state is <state>.
        - The type of the contact and whether the contact is or is not a strategic partner

          Format: The contact is of the subtype <subtypename> and is [NOT] a strategic partner.
        - If the contact is of the type ABDoctor, print the doctor category and the doctor specialty.

          Format: Doctor category is <category> and doctor specialty is <specialty>.

          (If the doctor category or doctor specialty is null then the following: Doctor category is NOT set and doctor specialty is NOT set.)
        - Else print that the contact is NOT of the type ABDoctor

          Format: Contact is NOT of the type <ABDoctorTypeName>.
      - If the contact is NOT found for the PublicID
        - **Format**: No contact found for PublicID: <publicID>.

8.2.3 Verification

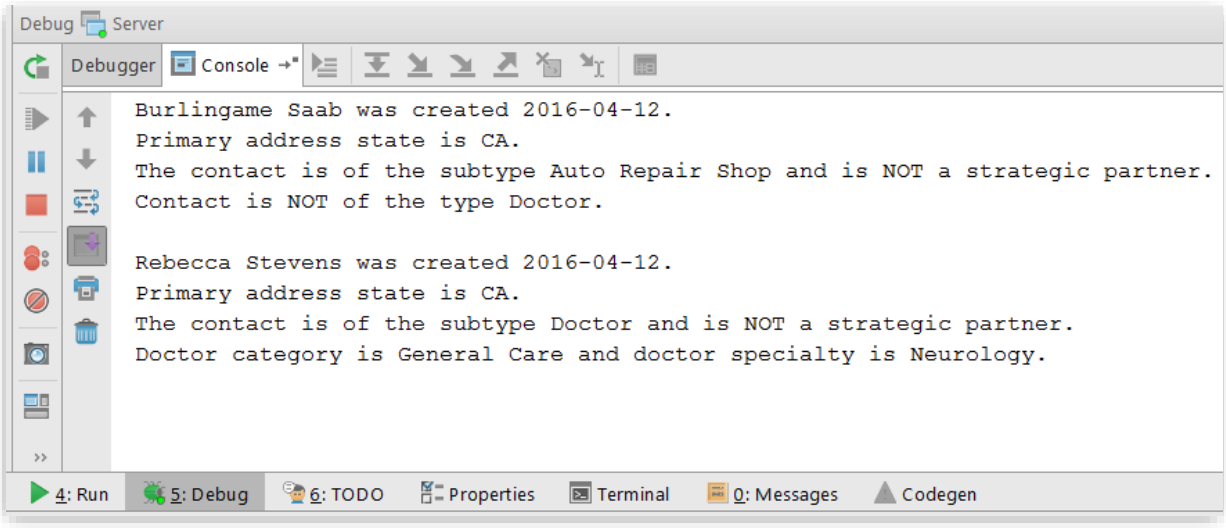
Activity

Verify the work you have done

1. **Execute your Gosu Code in Gosu Scratchpad**

   a) Using the PublicID for Burlingame Saab, verify the console output.

   b) Using the PublicID for Rebecca Stevens, verify the console output.



Stop

8.3 Lab: Working with arrays

In this exercise, you will write Gosu code in Gosu Scratchpad to retrieve the objects in an entity data model array. Finally, you will print different details of the bank accounts to the server log.

8.3.1 Write it down

1. **Question: Navigate to the Summary page of the following contact: Eric Andy. On the Summary page, what is the Public ID for Eric Andy? Go to the Bank accounts card on the Details page and verify that the contact has 2 bank accounts: ACME Credit Union – checking and savings accounts.**

2. **Question: Navigate to the Summary page of the following contact: William Andy. On the Summary page, what is the Public ID for William Andy? Go to the Bank accounts card on the Details page and verify that the contact has 2 bank accounts: SUCCEED Credit Union – checking and National Bank -**

savings accounts.

8.3.2 Configuration

In the second part of the exercise, you will open Gosu Scratchpad. Then, you will write and run Gosu code to print to the debug console details about specific contacts in TrainingApp.

1. **Open Gosu Scratchpad in Studio**
   a) Verify that the *Run in Debug Process* icon is available in Gosu Scratchpad.
2. **Write Gosu code**
   a) Use the trainingapp.base.QueryUtil.findPerson() method to retrieve each contact.
   b) Write Gosu code that prints bank account details to the server log per contact, based on the following requirements:
      - If the contact found for the PublicID
        - The number of bank accounts

          Format: <*contactname*> *has* <*numberofaccounts*> *bank accounts.*
        - If the contact has bank accounts, a numbered list of the bank accounts starting from 1

          Format: <*contactname*> has:

          <*bankname*> : <*accounttype*> -- <*accountnumber*>

          <*bankname*> : <*accounttype*> -- <*accountnumber*>
        - If the contact has at least one account with the bank name “National Bank”

          Format: This contact (<*contactname*>) has an account at <*bankname*>.

          Otherwise:

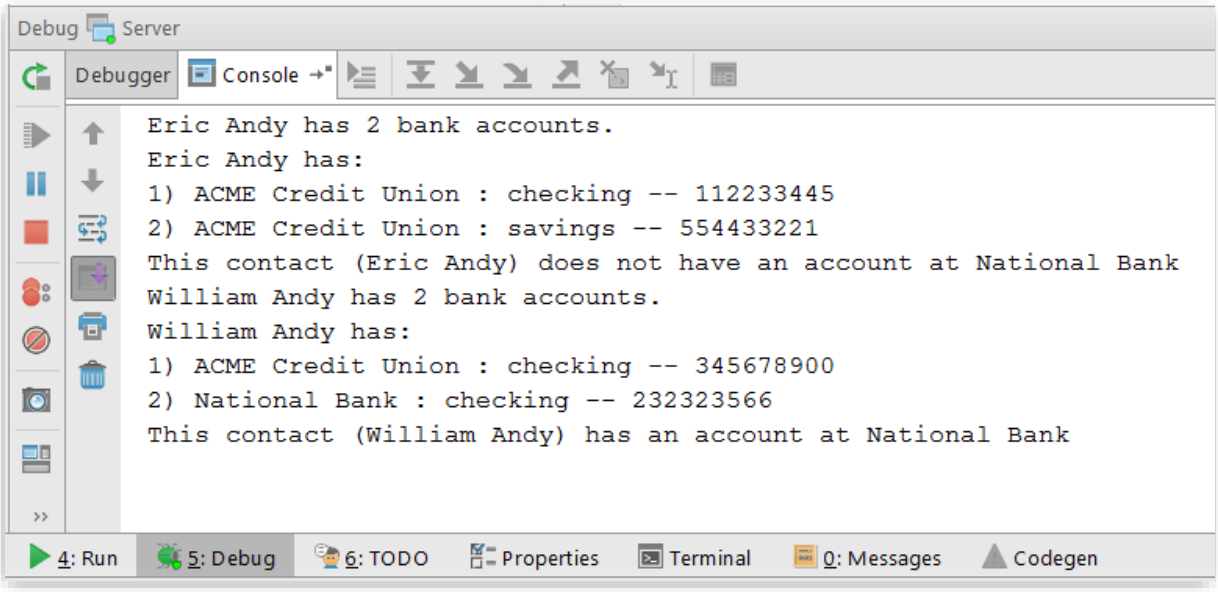
          Format: This contact (<*contactname*>) does not have an account at <*bankname*>.
      - If the contact is NOT found for the PublicID
        - **Format**: No person found for PublicID: <publicID>

8.3.3 Verification

Activity

Verify the work you have done

1. **Execute you Gosu Code in Gosu Scratchpad**
   a) Verify the console output.



Stop

8.4 Lab Solution: Coding with Gosu

Solution

8.4.1 Write it down

1. **Question: Navigate to the Summary page of the following company: Burlingame Saab. On the Summary page, what is the Public ID for Burlingame Saab?**

ab:78

2. **Question: Navigate to the Summary page of the following doctor: Rebecca Stevens. On the Summary page, what is the Public ID for Rebecca Stevens?**

ab:70

8.4.2 Configuration

Solution

Note: there are multiple correct solutions for this exercise.

1. **Open Gosu Scratchpad in Studio**
   a) Verify that the *Run in Debug Process* icon is available in Gosu Scratchpad.
2. **Write Gosu code**

```
uses trainingapp.base.QueryUtil

//**** START - my first Gosu program ****

//before running the code, replace this with the contact's PublicID
var publicID = "ReplaceThisWithThePublicIDOfTheContact"

var contact = QueryUtil.findContact(publicID)

if(contact != null) { //Requirement 1

  //Requirement 1/A
  print(contact.DisplayName + " was created " + contact.CreateTime + ".")

  //Requirement 1/B
  if(contact.PrimaryAddress != null) {
    print("Primary address state is " + contact.PrimaryAddress.State + ".")
  }

  //Requirement 1/C
  var isStrategicPartner = contact.IsStrategicPartner_Ext ? " and is a strategic partner" : " and is NOT
a strategic partner."
  print("The contact is of the subtype " + contact.Subtype.DisplayName + isStrategicPartner)

  //Requirement 1/D
  if(contact typeis ABDoctor) {
    var doctorCategory = contact.DoctorCategory != null ? contact.DoctorCategory.DisplayName : "NOT set"
    var doctorSpecialty = contact.DoctorSpecialty != null ? contact.DoctorSpecialty.DisplayName : "NOT
set"

    print("Doctor category is " + doctorCategory + " and doctor specialty is " + doctorSpecialty + ".")
  } else { //Requirement 1/E
    print("Contact is NOT of the type " + ABDoctor.Type.DisplayName + ".")
  }

} else { //Requirement 2/A
  print("No contact found for PublicID: " + publicID + ".")
}

//**** END - my first Gosu program ****
```

8.5 Lab Solution: Working with arrays

Solution

8.5.1 Write it down

1. **Question: Navigate to the Summary page of the following contact: Eric Andy. On the Summary page, what is the Public ID for Eric Andy? Go to the Bank accounts card on the Details page and verify that the contact has 2 bank accounts: ACME Credit Union – checking and savings accounts.**

ab:98

2. **Question: Navigate to the Summary page of the following contact: William Andy. On the Summary page, what is the Public ID for William Andy? Go to the Bank accounts card on the Details page and verify that the contact has 2 bank accounts: SUCCEED Credit Union – checking and National Bank - savings accounts.**

ab:5

8.5.2 Configuration

Solution

Note: there are multiple correct solutions for this exercise.

1. **Open Gosu Scratchpad in Studio**
   a) Verify that the *Run in Debug Process* icon is available in Gosu Scratchpad.
2. **Write Gosu code**

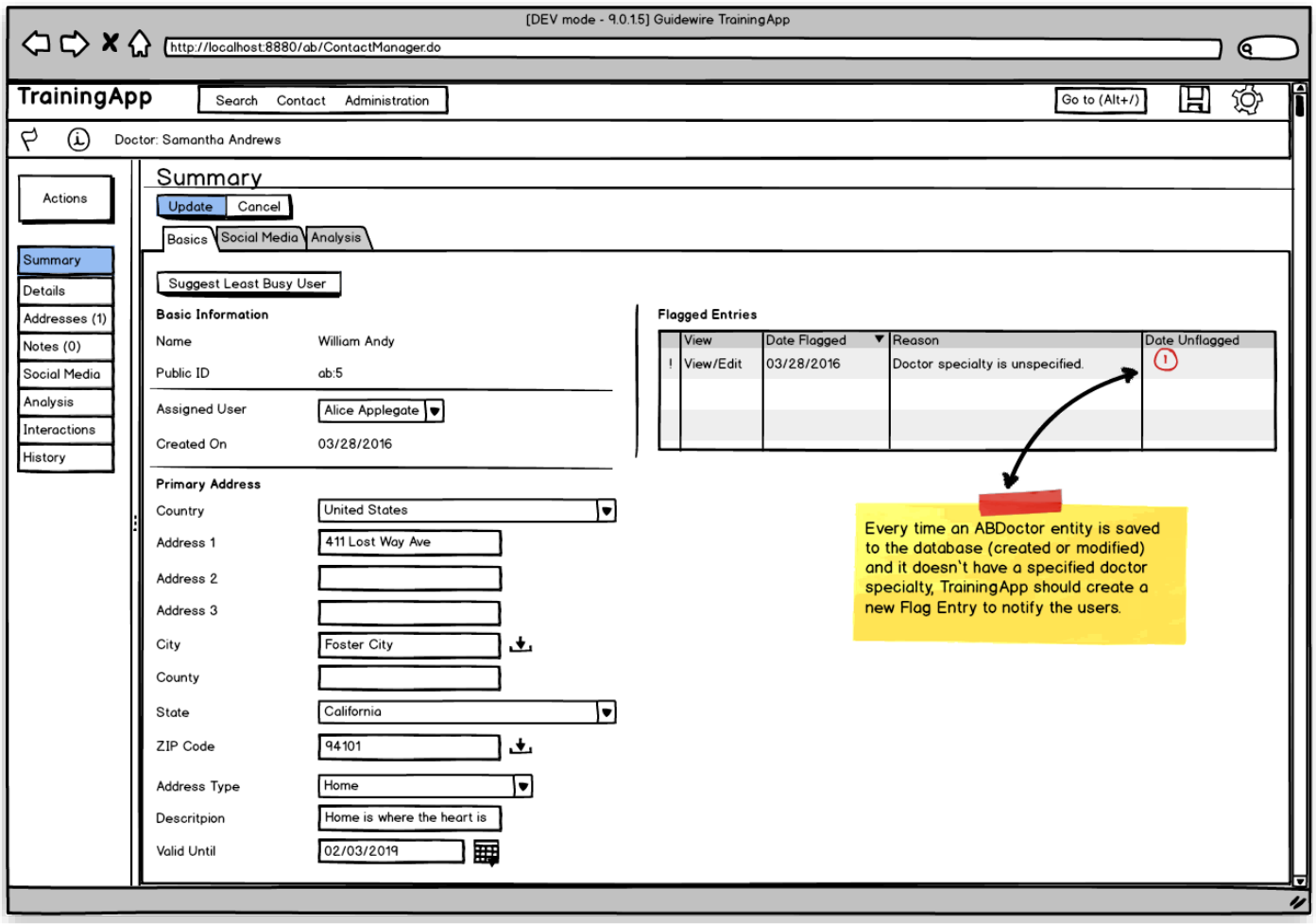
```
uses trainingapp.base.QueryUtil

var personIDs = new String[] {"98", "5"}

for (personID in personIDs) {
   var person = QueryUtil.findPerson(personID)

   if (person != null) {
     print(person.DisplayName + " has " + person.BankAccounts.length + " bank accounts.")

     if (person.BankAccounts.length > 0) {
       print(person.DisplayName + " has: ")
```

```
      for (bankAccount in person.BankAccounts index i) {
        print(i + 1 + ") " + bankAccount.BankName + " : " + bankAccount.AccountType + " -- " +
bankAccount.AccountNumber)
      }
    }

    var bankName = "National Bank"

    if (person.BankAccounts.hasMatch(\account -> account.BankName == bankName)) {
      print("This contact (" +person.DisplayName + ") has an account at " + bankName)
    } else {
      print("This contact (" +person.DisplayName + ") does not have an account at " + bankName)
    }

  } else {
     print("No person found for PublicID: " + personID)
  }
}
```

Lesson 9 Gosu Rules

An insurance company wants to ensure that every ABDoctor has a value specified for the doctor specialty field in the application. TrainingApp users can continue creating ABDoctors without specifying a doctor specialty, but the application should automatically create a reminder for the users that they will have to enter this information later.

The business analysts sent us the following wireframe and user story:



"Every time an ABDoctor entity is saved to the database (created or modified) and it doesn't have a specified doctor specialty, TrainingApp should create a new Flag Entry to notify the users." – Insurance company business analysts

In this exercise you will create a new Gosu Rule to implement this requirement. **No Data Model or UI changes are required to meet the requirement.**

9.1 Prerequisites

For this exercise, 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`.

9.2 Lab: Create a new Gosu Rule

As a configuration developer, you want to be able to outline Gosu Rules based on generic business requirements.

Generic business requirement:

"Every time an ABDoctor entity is saved to the database (created or modified) and it doesn't have a specified doctor specialty, TrainingApp should create a new Flag Entry to notify the users." – Insurance company business analysts

9.2.1 Write it down

1. **Question**: **Which one of the 3 Rule Set Categories in TrainingApp is the best to implement the requirement: EventMessage, Preupdate or Validation**?

Review

What is a Rule Set Category

A **Rule Set Category** is a collection of Rule Sets that have the following 2 things in common:

- High-level business purpose
- Trigger (Event)

2. Question: Which existing Rule Set in the Rule Set Category will be used? What is the Root entity type? Hint: It will be one of the ancestors of ABDoctor.

Review

What is a Rule Set?

A **Rule Set** combines many individual rules into a useful set to consider as a group. A Rule Set defines the root entity type:

- All the Rules in the set will be attached to the same the triggering entity (root entity)

3. Question: What will be your rule's name?

4. Question: What are the two logical conditions in the requirement?

1.

2.

5. Question: What will be the rule's action?

Review: What is a Rule?

A Rule, also known as a **Gosu Rule,** consists of a root entity, an expression that resolves to true or false, and an action that is executed if the condition is true.

Stop

9.2.2 Configuration

As a configuration developer, you want to create new Gosu Rules. In this exercise, you will first create a preupdate rule that creates a flag entry for a contact of the type ABDoctor that does not have a defined doctor specialty. Then, you will verify the execution of your rule.

1. **Open Guidewire Studio**
2. **Create an ABContactPreupdate rule for contacts of the subtype ABDoctor that creates a flag entry when a contact of the type ABDoctor does not have a defined specialty**
   a) Create the new rule using the name you specified in the first part of this exercise.
   b) In the Rule Set Editor, write Gosu code that meets the following condition criteria:
      - The contact is of the type ABDoctor.
      - The Specialty field is null.
   c) In the Rule Set Editor, write Gosu code that performs the following actions:
      - Creates a new flag entry.
      - Specifies the flag date as the current date.
      - Specifies the flag entry reason as an unspecified specialty for the doctor.
      - Adds the flag entry to the flag entries array for the given contact.

Important!

Gosu Rules are within transaction scope. This means you don't need to manually create, commit or rollback transactions (bundles) in Gosu Rules. However, if an exception is raised in your Rule, then all the other changes made by different Rules will be rolled back and the exception will propagate back to the caller (which in this example is the UI).

9.2.3 Verification

Activity

Verify the work you have done

1. **Deploy the rule**
2. **Log in to TrainingApp**
   a) From Studio, if your server is not already running, start the server using Debug 'Server'.
   b) Review the Debug console for errors and verify that the application is running in the Debug console.
   c) Log in as Super User.
3. **Change the doctor specialty for Rebecca Stevens**
   a) Open the Rebecca Stevens contact.
   b) In the Details page, in the Person Info tab, in the Medical Specialty input set, change the Specialty to <none> (if it is not already)
   c) Click Update.
4. **Edit the flag entry**
   a) In TrainingApp, view the summary page for Rebecca Stevens.
   b) In the Flag Entries list view, for the row with a warning flag and the "Doctor Specialty is unspecified" reason, click View/Edit.
   c) In the Flag Entry popup, in the Resolution field, enter your name.
   d) Click Update.
   e) Verify that the flag entry is no longer flagged.

Stop

9.3 Lab Solution: Create a new Gosu Rule

Solution

9.3.1 Write it down

1. **Question: Which of the 3 Rule Set Categories in TrainingApp is the best to implement the requirement: EventMessage, Preupdate or Validation?**

Preupdate

2. **Question: Which existing Rule Set in the Rule Set Category will be used? What is the Root entity type?**

Rule Set name: ABContactPreupdate, Root entity type: ABContact

3. **Question: What will be your rule's name?**

ABPU5000 – Subtype ABDoctor without doctor specialty

4. **Question: What are the two logical conditions in the requirement?**

1. The contact is an ABDoctor 2. Doctor specialty field is not specified

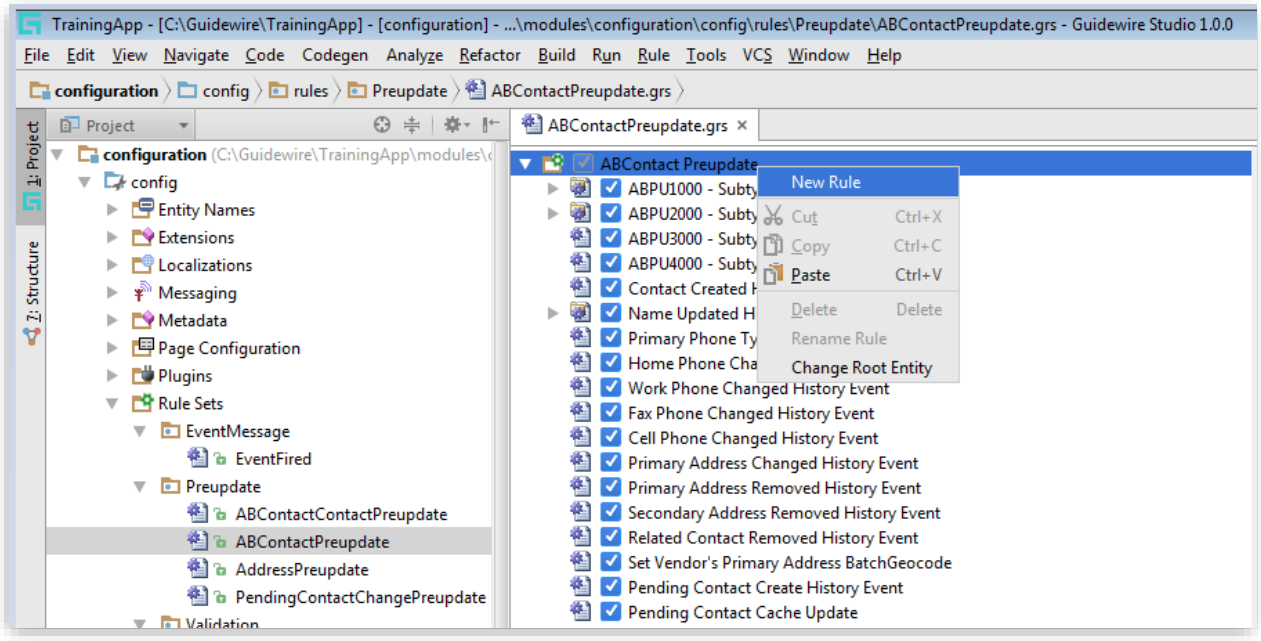
5. **Question: What will be the rule's action?**

Creating an initializing a new FlagEntry entity

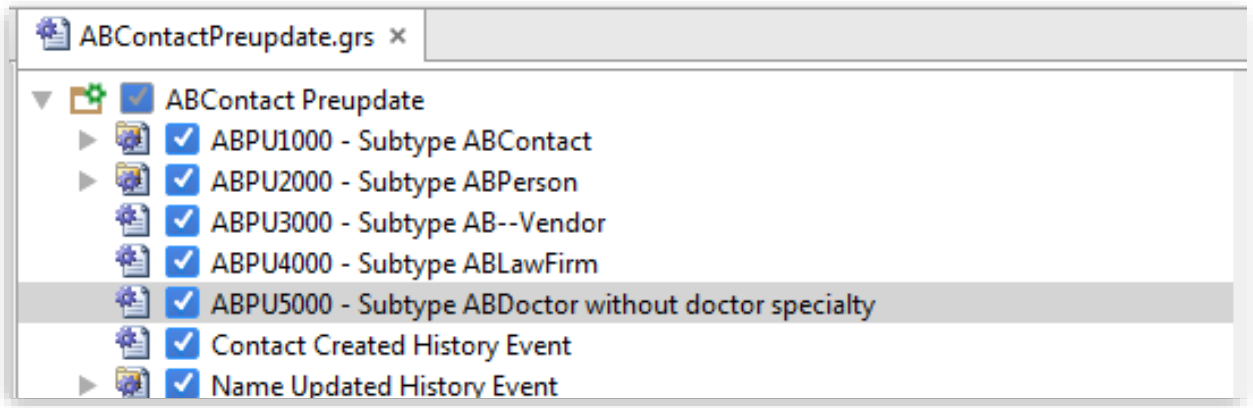
9.3.2 Configuration

Solution

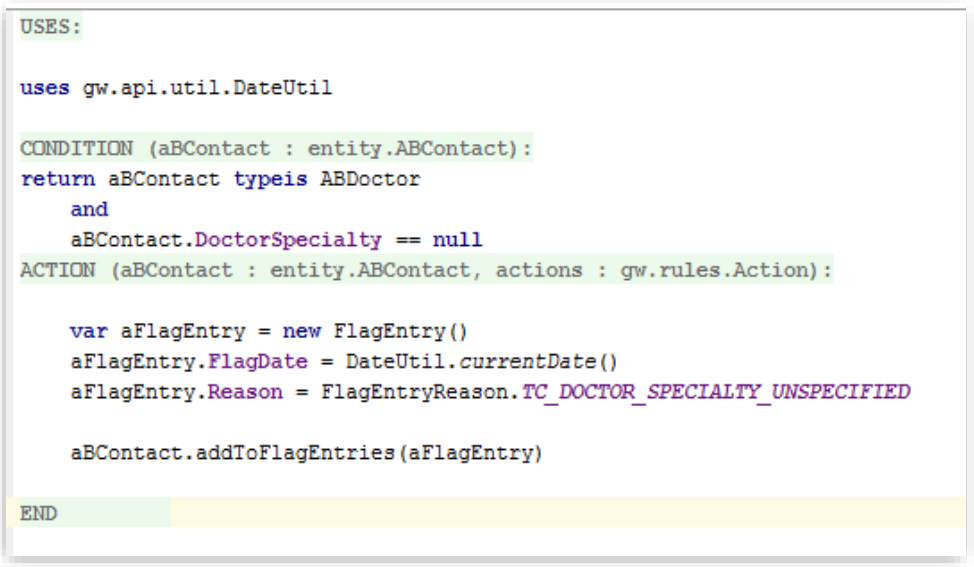
1. **To create the rule, right click on the ABContactPreupdate Rule Set and select New Rule. Important: make sure that you do not create the rule in the AB<u>ContactContact</u>Preupdate because you will have to delete and recreate it later.**



2. **Specify the rule name, then drag and drop it to the logically correct position between ABPU4000 - Subtype ABLawFirm and Contact Created History Event rules.**



3. Specify the sections of the rule:



```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
return aBContact typeis ABDoctor
    and
    aBContact.DoctorSpecialty == null
ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```

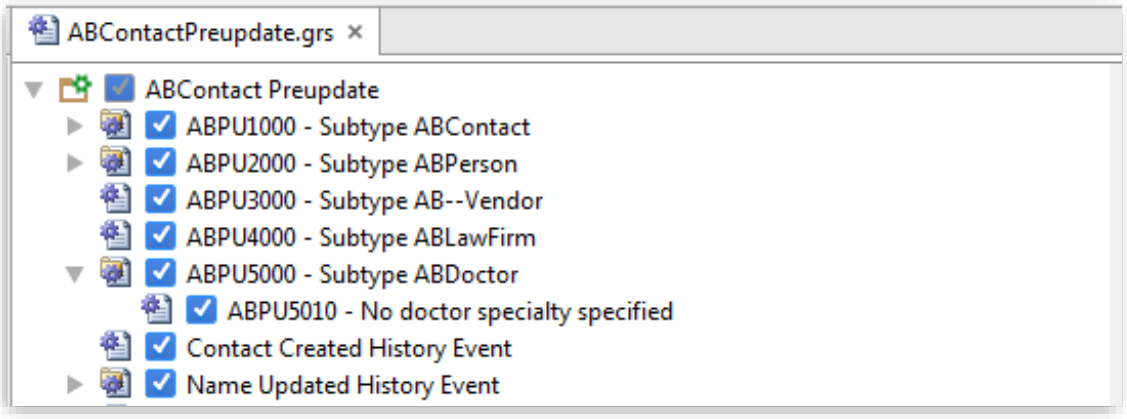
Note: You also have the option to copy and paste it from below. However, it is recommended that you type the code in based on the screenshot.

Important: please make sure that you copy and paste the sections individually without the parts highlighted in dark gray.

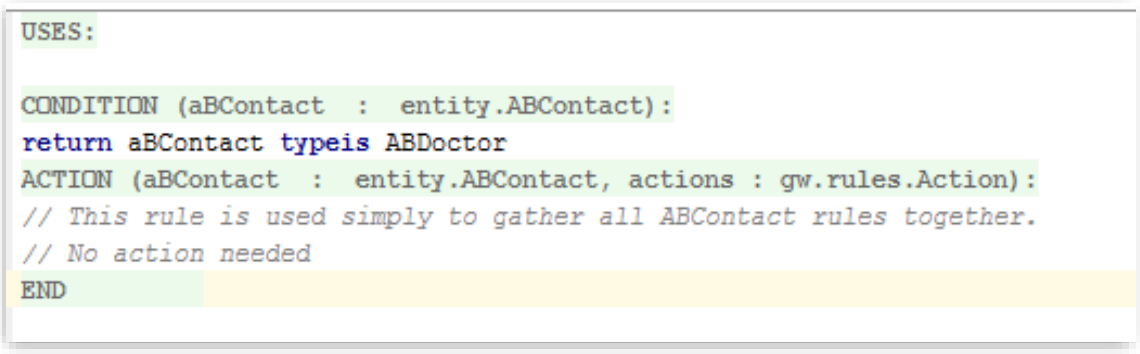
```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
  return aBContact typeis ABDoctor
    and
    aBContact.DoctorSpecialty == null

ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```

Solution 2

Note: You could also use two rules (parent-child) to implement this requirement. Recall the benefit of this is that the children can share the parent rule's condition, so we don't have to duplicate it.



1. Specify the sections of the parent rule:



```
USES:

CONDITION (aBContact  :  entity.ABContact):
return aBContact typeis ABDoctor
ACTION (aBContact  :  entity.ABContact, actions : gw.rules.Action):
// This rule is used simply to gather all ABContact rules together.
// No action needed
END
```

Note: You also have the option to copy and paste it from below. However, it is recommended that you type the code in based on the screenshot.

Important: please make sure that you copy and paste the sections individually without the parts highlighted in dark gray.

```
USES:

CONDITION (aBContact : entity.ABContact):
        return aBContact typeis ABDoctor
        ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):
// This rule is used simply to gather all ABContact rules together.
// No action needed

END
```

2. Then Specify the sections of the child rule:

```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
// Note that in this case the typecast is required because
// the typeis operator is not in the same condition section anymore
return (aBContact as ABDoctor).DoctorSpecialty == null
ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```

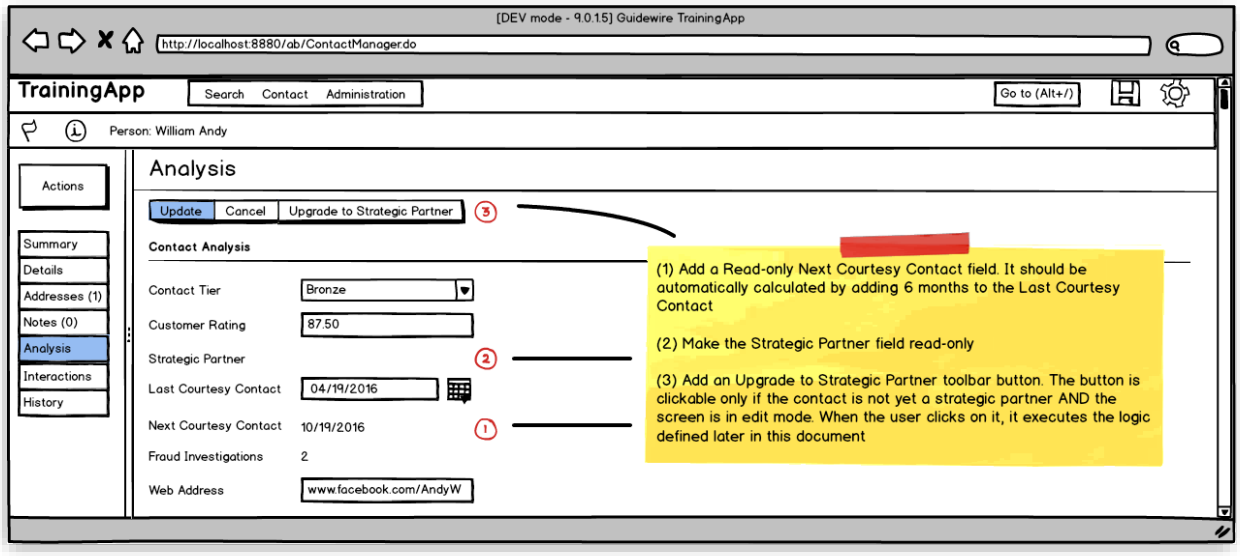
Note: You also have the option to copy and paste it from below. However, it is recommended that you type the code in based on the screenshot.

Important: please make sure that you copy and paste the sections individually without the parts highlighted in dark gray.

```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
// Note that in this case the typecast is required because
// the typeis operator is not in the same condition section anymore
return (aBContact as ABDoctor).DoctorSpecialty == null

ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```

Lesson 10 Enhancements

At the insurance company, business requirements necessitate that all contacts of the type ABContact should receive a courtesy call once every six months. This information needs to appear with contact analysis details. In addition, business requirements require that there is easy way to upgrade contacts to the level of strategic partner. We have just received the following wireframe and user stories from the business analysts:



"At our insurance company, we want our end users to be able to see the recommended Next Courtesy Contact date in the UI. This date should be calculated automatically and the users shouldn't be able to edit it. We also want the Strategic Partner field to be read-only because we want to automatically calculate the customer rating when a contact a becomes a strategic partner. To achieve this, add a new button that will calculate the customer rating and perform the upgrade." – Insurance company business analysts

In this lab, you will create a Gosu enhancement for the ABContact entity and make related changes for various PCF Files.

10.1 Prerequisites

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

- Extending Base Application Entities
- Atomic Widgets
- Detail Views

For this exercise, 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`.

10.2 Lab: Create a new enhancement

As a configuration developer, you want to be able to create or modify entity enhancements; and work with getter/setter properties and functions. In this exercise, you will create an enhancement for the ABContact entity related to analysis. The enhancement will contain a getter property and a method.

1. **Open Guidewire Studio for TrainingApp**
   a) From Studio, if your server is not already running, start the server using Debug 'Server'.
   b) Review the Debug console for errors and verify that the application is running in the Debug console.
2. **In Guidewire Studio, create the package and the enhancement**
   a) In Guidewire Studio, create a package called `succeedlab.ta.enhancements.entity`
   b) Create an enhancement for the ABContact entity that conveys that this enhancement relates to analysis.
3. **Create a getter property named NextCourtesyContact()**
   a) Create a getter property named NextCourtesyContact() that returns a date that is six (6) months in the future of the Last Courtesy Contact date. If Last Courtesy Contact date is null, then the property must return null.
4. **Create a method named upgradeToStrategicPartner()**
   a) Create a method named upgradeToStrategicPartner() that, for a given contact, does the following:
      - Sets the IsStrategicPartner_Ext field to true.
      - If the value of the CustomerRating_Ext field is null, sets the value to 25.
      - If the value of the CustomerRating_Ext field is between 0 and 989.9, increments that value by 10.
      - If the value of the CustomerRating_Ext field is more than 989.9, sets the value to 999.9.

10.2.1 Modify the PCF configuration

As a configuration developer, you want to use atomic widgets to display enhancement properties and to invoke enhancement functions. In this exercise, you will modify the ABContactAnalysisDV.pcf file to display the value of the getter property. Next, you will modify the ABContactAnalysisPage.pcf file. You will add a toolbar button to the toolbar. You will configure the button to call the enhancement method.

Review

How to work with atomic widgets

Note: If needed, revisit the *Atomic Widgets* lesson for examples, tips and keyboard shortcuts.

1. **Modify ABContactAnalysisDV**
   a) In Studio, open ABContactAnalysisDV.pcf.
   b) Modify the Strategic Partner widget to be read-only.
   c) To display Next Courtesy Contact, add the appropriate read-only input widget below Last Courtesy Contact.
   d) Create a displaykey for the widget's label property that reads Next Courtesy Contact.
   e) Configure the widget to display the enhancement property value.
2. **Modify ABContactAnalysisPage**
   a) In Studio, open ABContactAnalysisPage.pcf.
   b) In the existing toolbar, add a toolbar button.
   c) Create a displaykey for the button's label property that reads Upgrade to Strategic Partner.
   d) Configure the toolbar button so that it calls the upgradeToStrategicPartner() method when clicked.
   e) Configure the button so that it is only clickable when the contact is not a strategic partner and when the page is in edit mode.

Hint

No special syntax is required to reference entity enhancement elements

To reference properties or methods, just simply use the dot notation.

Examples:

```
var anABPerson = trainingapp.base.QueryUtil.findPerson("ab:5")
print(anABPerson.Age) // defined in acme.ta.enhancements.entity.ABPersonEnhancement
print(anABPerson.DateOfBirth) //defined in ABPerson.eti
```

Guidewire API

Test if the Location is in edit more or not.

`CurrentLocation` is an implicit object that represents the currently displayed location in the browser and it is available to use in every PCF file. You could use it to determine if the location - that the end user is looking at – is in edit mode or read-only mode.

Syntax: `CurrentLocation.InEditMode`

This expression:

- Returns true if current location is in edit mode
- Returns false if current location is in read-only mode

Guidewire API

Making a widget clickable or not clickable based on a condition

Most atomic widgets have the `available` property. This can be set to a Boolean expression which, if false, grays out the widget and all its children. For example, to disable an action in read-only mode, you can set

```
available=CurrentLocation.InEditMode || <action-available-condition>
```

or

```
available=CurrentLocation.InEditMode && <action-available-condition>
```

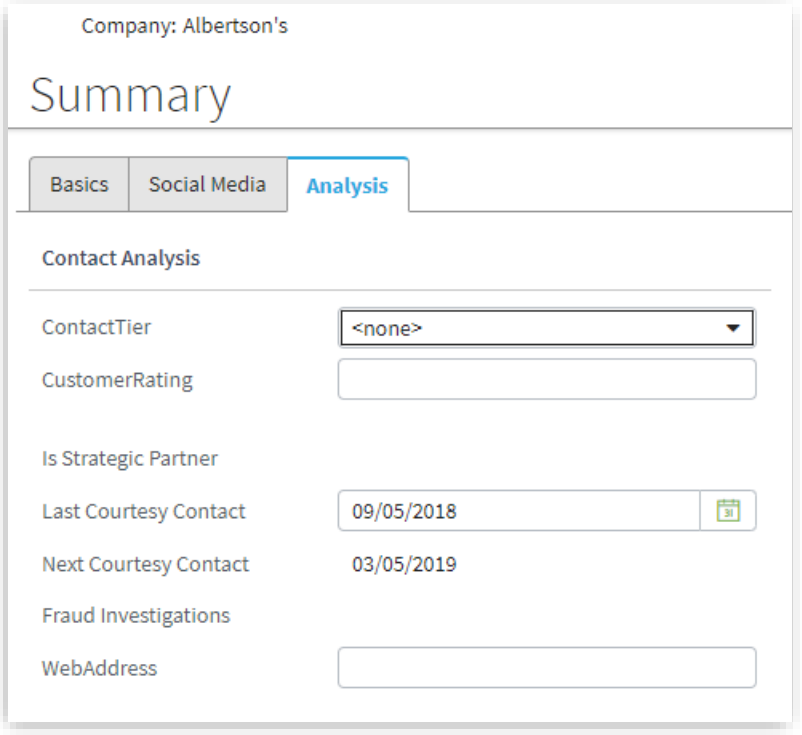
10.2.2 Verification

Activity

Verify the work you have done

1. **Deploy your changes**
   a) Restart the server (Stop / Debug 'Server') in Guidewire Studio.
2. **Verify the enhancement method and PCF behavior**
   a) Log in to TrainingApp as Alice Applegate.
   b) Search for and open the Albertson's contact.
   c) For the contact, navigate to the Analysis page.
   d) Click Edit.
   e) Set the Last Courtesy Contact field to today's date. Click Update.

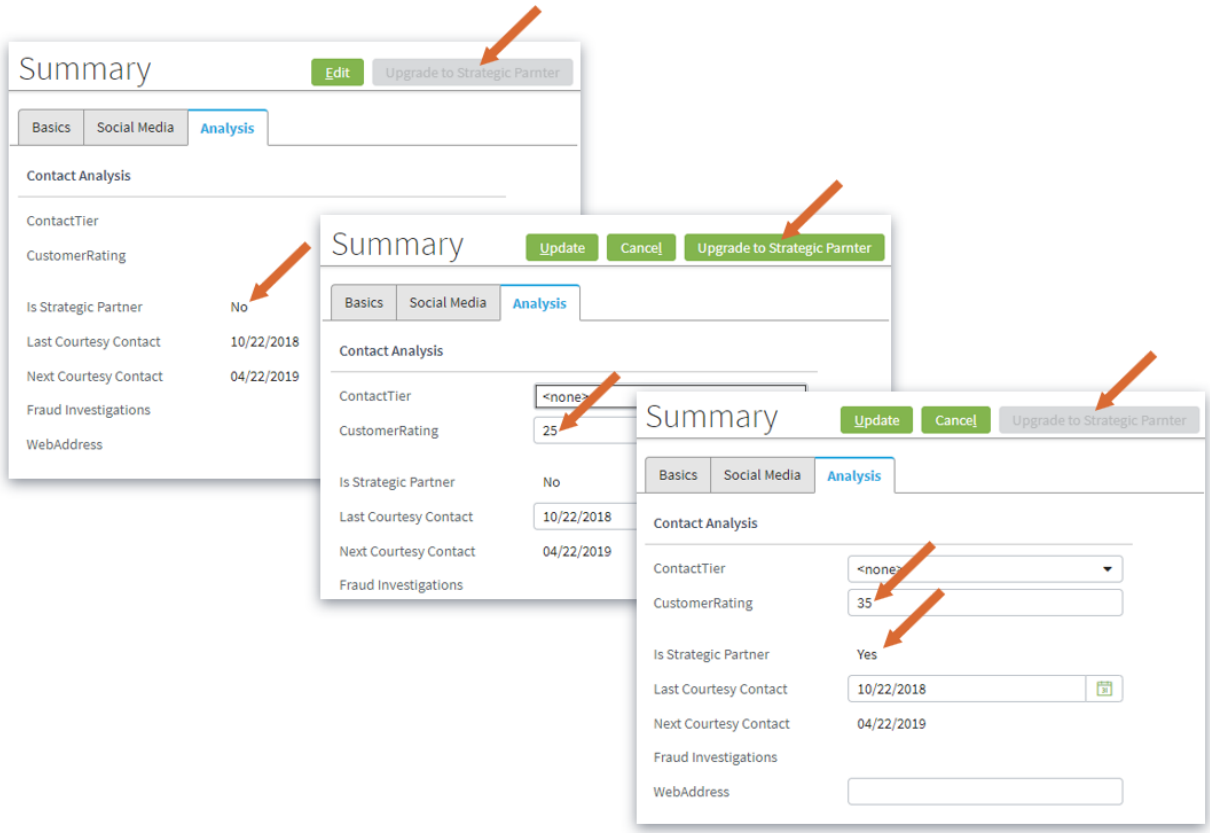
f) Verify that the Next Courtesy Contact field shows a date that is six (6) months from today's date.



g) Search for and open the Albertson's contact.

h) For the contact, navigate to the Analysis page.

i) Verify that the Upgrade to Strategic Partner button is clickable only when the user is editing the page and the Strategic Partner field is not equal to Yes.

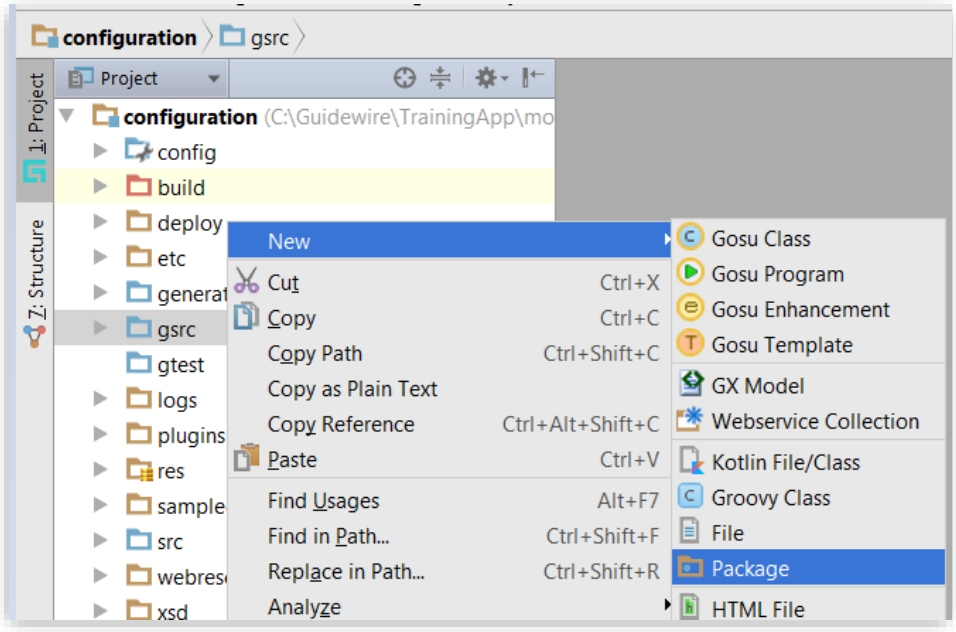


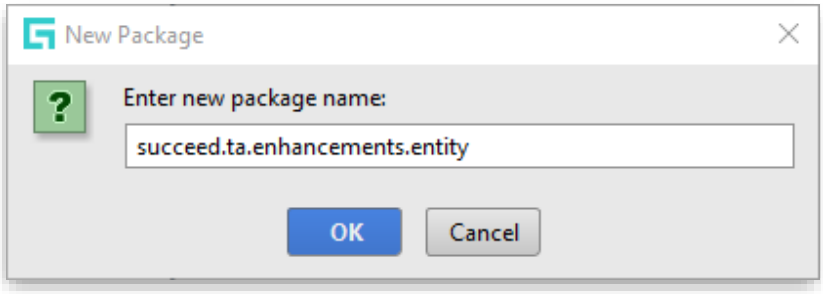
10.3 Lab Solution: Create a new enhancement

Solution

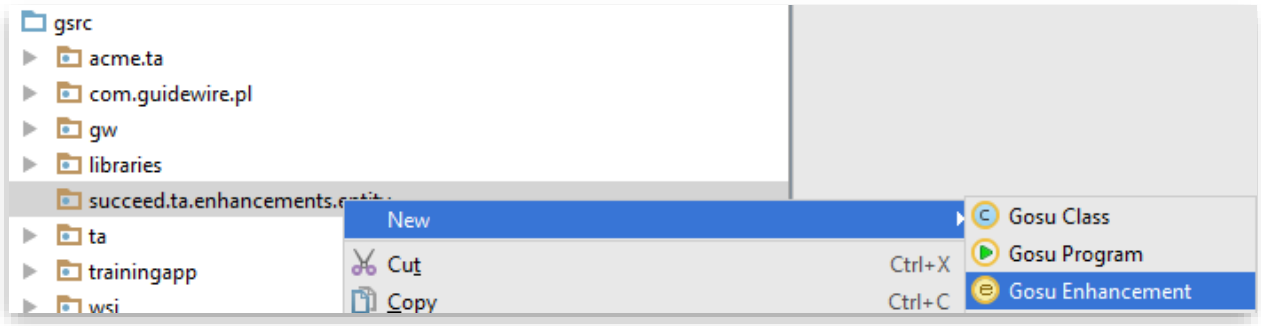
1. In Guidewire Studio, create the package and the enhancement

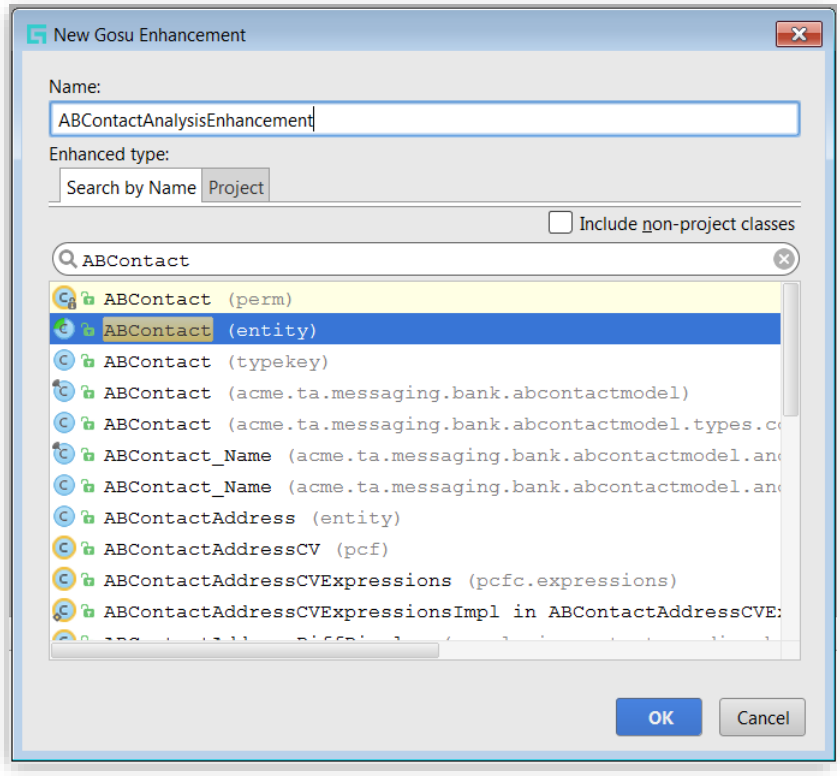
a) In Guidewire Studio, create a package called `succeed.ta.enhancements.entity`





b) Create an enhancement for the ABContact entity that conveys that this enhancement relates to analysis.





2. Create a getter property named NextCourtesyContact()

a) Create a getter property named NextCourtesyContact() that returns a date that is six (6) months in the future of the Last Courtesy Contact date. If Last Courtesy Contact date is null, then the property must return null.

```
property get NextCourtesyContact() : Date {
  var nextCourtesyDate : Date = null

  if(this.LastCourtesyContact_Ext != null) {
    nextCourtesyDate = this.LastCourtesyContact_Ext.addMonths(6)
  }

  return nextCourtesyDate
}
```

3. Create a method named upgradeToStrategicPartner()

a) Create a method named upgradeToStrategicPartner() that, for a given contact, does the following:

- Sets the IsStrategicPartner_Ext field to true.
- If the value of the CustomerRating_Ext field is null, sets the value to 25.
- If the value of the CustomerRating_Ext field is between 0 and 989.9, increments that value by 10.
- If the value of the CustomerRating_Ext field is more than 989.9, sets the value to 999.9.

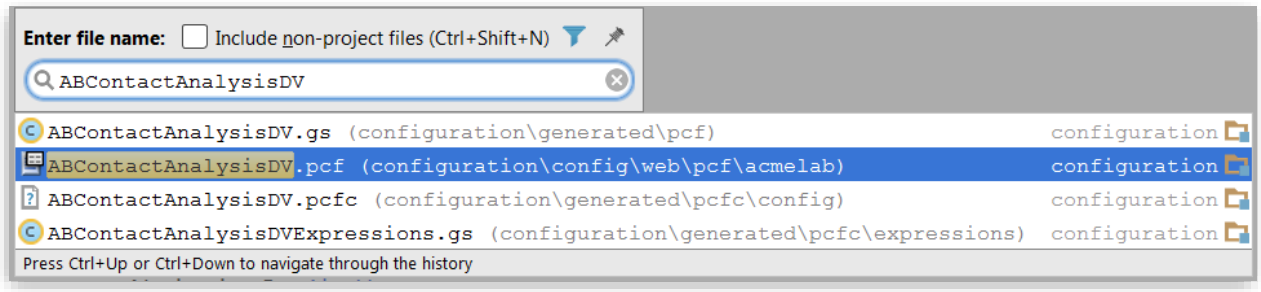
```
function upgradeToStrategicPartner() : void {
  if (this.IsStrategicPartner_Ext != true) {
    this.IsStrategicPartner_Ext = true
  }

  if (this.CustomerRating_Ext == null) {
    this.CustomerRating_Ext = 25
  }  else if (this.CustomerRating_Ext >= 0 and this.CustomerRating_Ext <= 989.9) {
    this.CustomerRating_Ext += 10
  } else {
    this.CustomerRating_Ext = 999.9
  }
}
```

10.3.1 Modify the PCF configuration

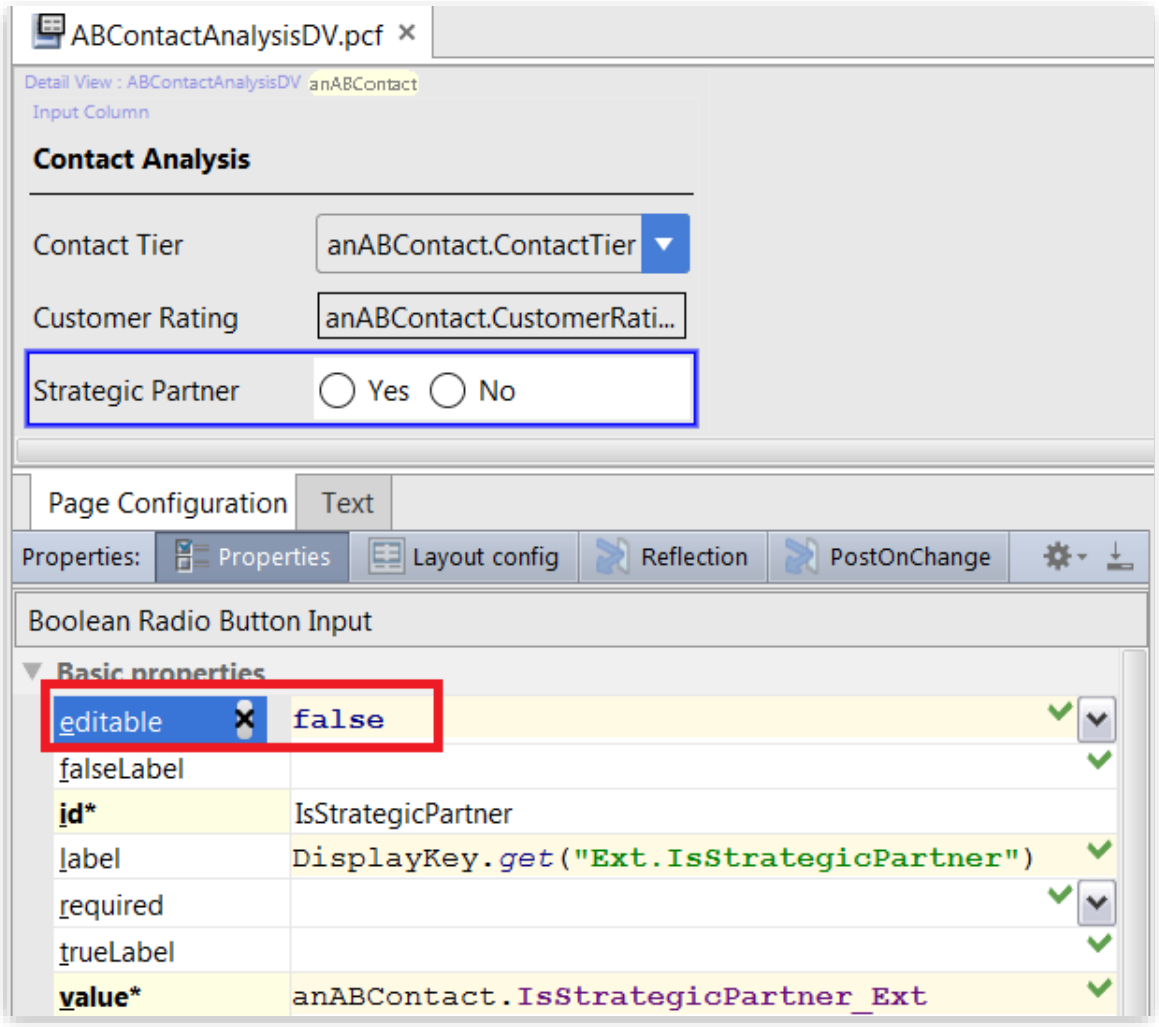
Solution

1. Modify ABContactAnalysisDV

a) In Studio, open ABContactAnalysisDV.pcf. Use `CTRL + SHIFT + N` shortcut in Studio to search for the PCF file:



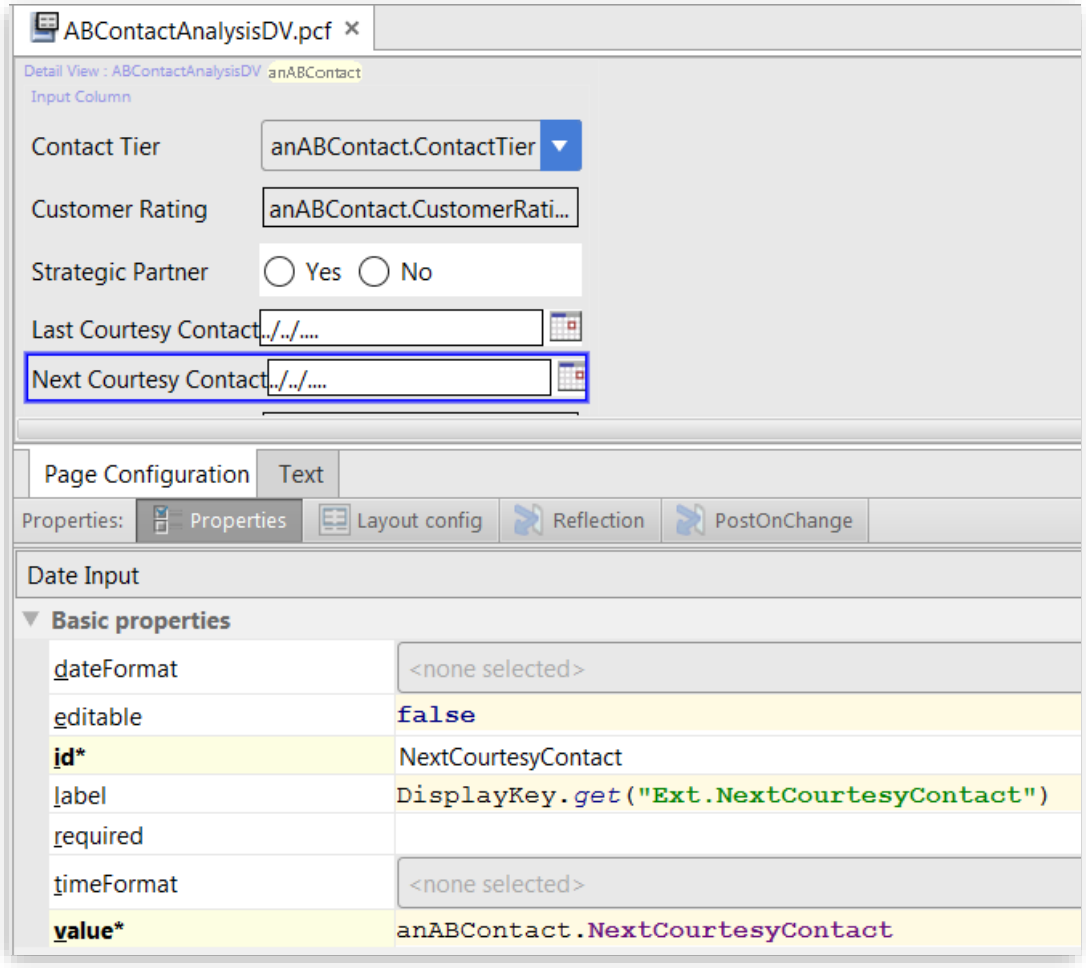
b) Modify the Strategic Partner widget to be read-only.



c) Add the appropriate read-only input widget below Last Courtesy Contact.

d) Create a displaykey for the widget's label property that reads Next Courtesy Contact.

e) Configure the widget to display the enhancement property value.



2. Modify ABContactAnalysisPage

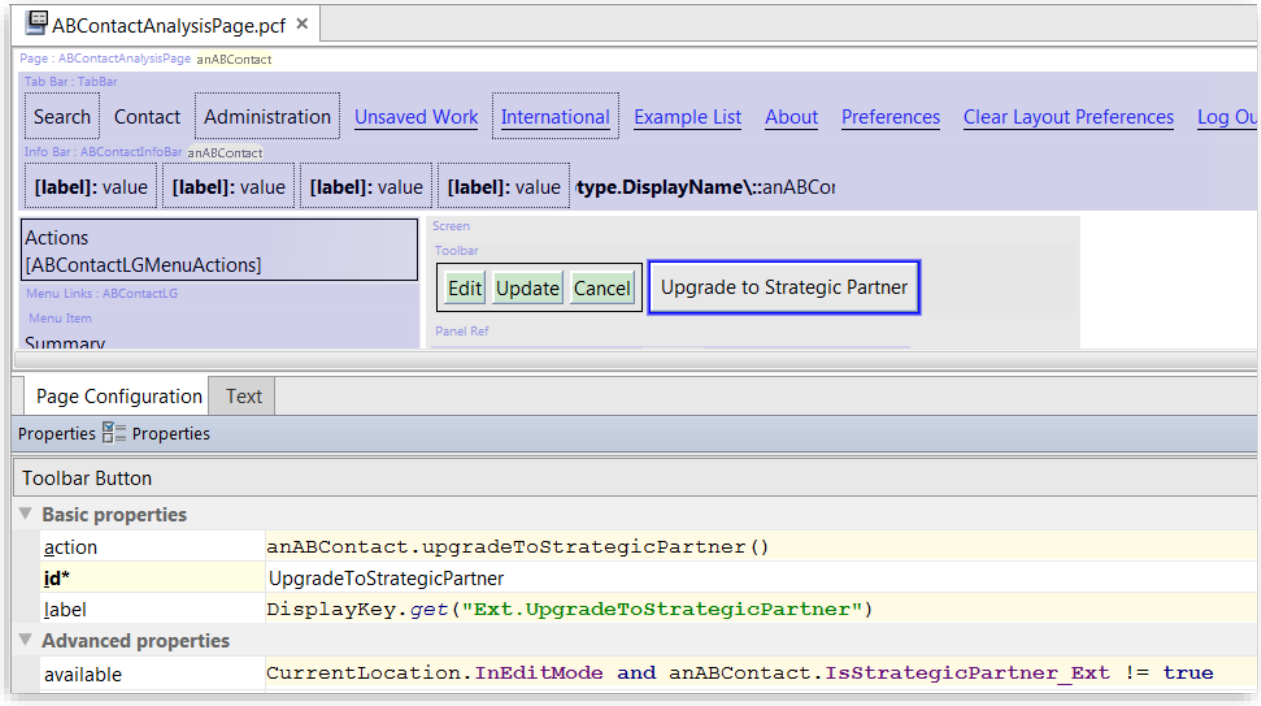
a) In Studio, open ABContactAnalysisPage.pcf.

b) In the existing toolbar, add a toolbar button.

c) Create a displaykey for the button's label property that reads Upgrade to Strategic Partner.

d) Configure the toolbar button so that it calls the upgradeToStrategicPartner() method when clicked.

e) Configure the button so that it is only clickable when the contact is not a strategic partner and when the page is in edit mode.


ABContactAnalysisPage.pcf
Page : ABContactAnalysisPage anABContact
Tab Bar : TabBar
Search
Contact
Administration
Unsaved Work
International
Example List
About
Preferences
Clear Layout Preferences
Log Ou
Info Bar : ABContactInfoBar anABContact
[label]: value
[label]: value
[label]: value
[label]: value
type.DisplayName\::anABCor
Actions
[ABContactLGMenuActions]
Menu Links : ABContactLG
Menu Item
Summary
Screen
Toolbar
Edit
Update
Cancel
Upgrade to Strategic Partner
Panel Ref
Page Configuration
Text
Properties
Properties
Toolbar Button
Basic properties
action
anABContact.upgradeToStrategicPartner()
id*
UpgradeToStrategicPartner
label
DisplayKey.get("Ext.UpgradeToStrategicPartner")
Advanced properties
available
CurrentLocation.InEditMode and anABContact.IsStrategicPartner_Ext != true

Lesson 11 Code Generation and Debugging

We have just received the following bug report from the testers:

User Story: *An insurance company wants to be able to "cascade" the ABCompany's email address to all employees who do not have an individual email address.*

Bug Description: *This user story has already been implemented, but it is not working correctly. If I click on the 'Cascade Email Address' button then the application cascades the company's email address to all employees and not only to the ones that do not have an email address already. This results in overriding the person's email address which is not the desired behavior. Please fix it ASAP.*

Priority: *High*

Screenshots:

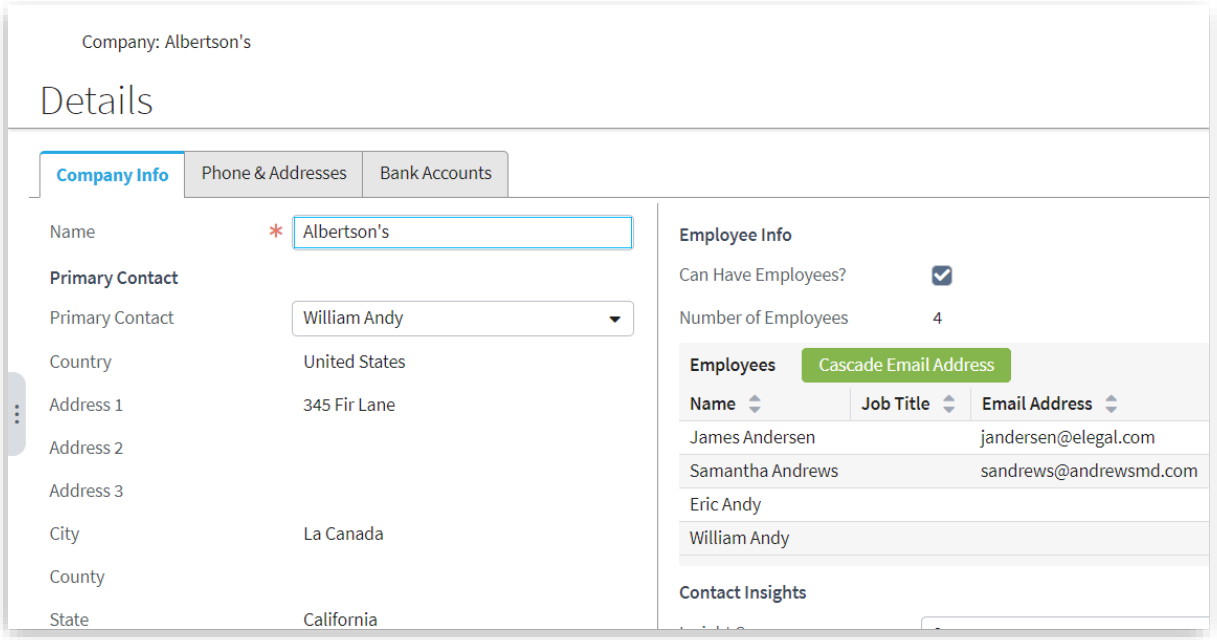


Figure 1 Before clicking on Cascade Email Address

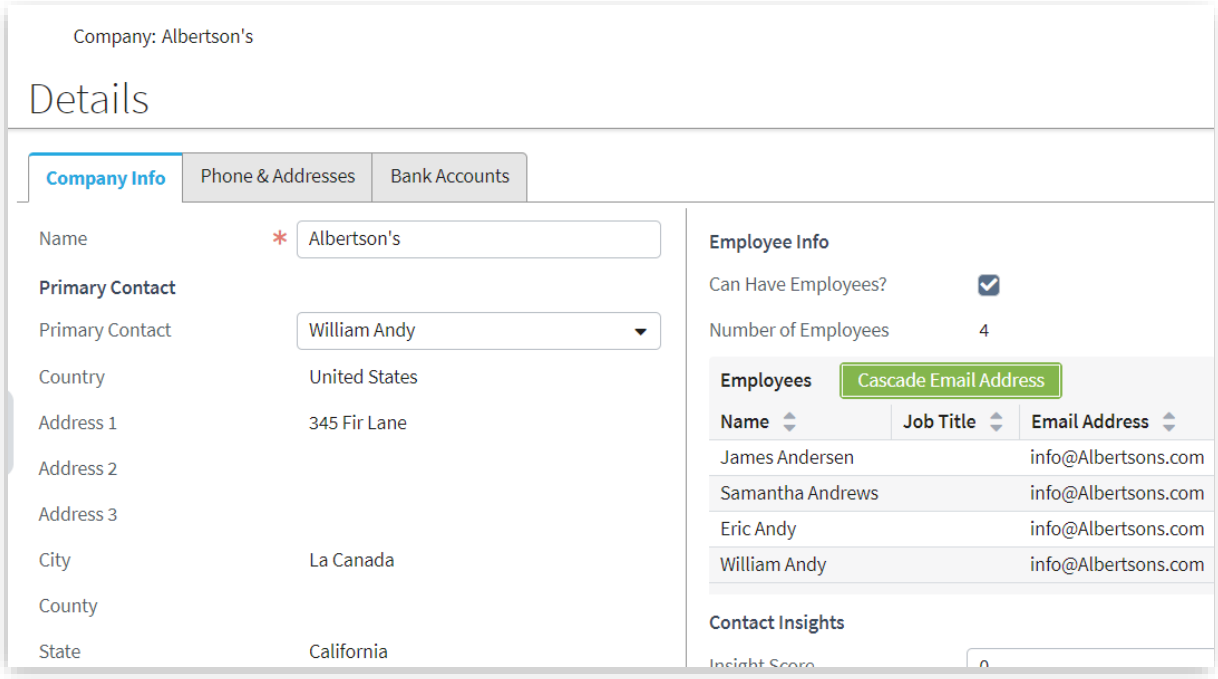


Figure 2 After clicking on Cascade Email Address

In this lab, you will debug a PCF method to identify a bug. Then, you will modify the PCF method and invoke the code generation. Finally, you will deploy the fixed PCF file and re-test the user story.

11.1 Prerequisites

For this exercise, 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`.

Verify that Albertson's has 4 employees as you can see it in Figure 1 above. Restore the development database if Albertson's has less than 4 employees or if all the employees have an email address.

11.2 Lab: Fix a bug

As a configuration developer, you want to be able to debug Guidewire applications. In this exercise you will use the debugger to debug a PCF method and identify an issue.

1. Open Guidewire Studio for TrainingApp

a) From Studio, if your server is not already running, start the server using Debug 'Server'.

b) Review the Debug console for errors and verify that the application is running in the Debug console.

2. **Locate the function called by the Cascade Email Address button**

a) Log in to TrainingApp as Alice Applegate and go to the Details screen of Albertson's.

b) Open the PCF file in Studio and identify the function specified in the action property of the Cascade Email Address button.

11.2.1 Lab: Write it down

1. **Question: What is the name of the method called by the Cascade Email Address button?**

2. **Go to the function declaration and verify it is a PCF function. CTRL + Left click on the function name.**

a) Verify that the Code tab opens in the PCF file

Guidewire API

The featured Guidewire API reference.

PCF methods are plain old Gosu functions defined in PCF files. The syntax is the same regardless where you define the function (PCF file, Enhancement or Gosu class). However, the semantics is slightly different: a PCF method…

- …is local (private) in the PCF file that defines it (not even referenced files can access it)
- …cannot be debugged at the place of definition (you have to open the generated `<PCFFileName>Expressions.gs` file after code generation, locate the method and put a breakpoint there)
- …is called from a widget property (e.g.: `action`, `value`, `available`, `editable`, etc.)

PCF methods are defined on the Code tab of the main PCF Element – container or location – and have direct access to the root object(s).

Why would you write code in a PCF file as opposed to an Enhancement or Gosu class? Because the logic the function implements is tied to one or more widgets in that PCF file and it is not needed anywhere else. For example:

- determining the label (View or View/Edit) for a button
- determining if a TextInput widget is editable or not editable
- executing some initialization logic after navigating to a location

3. **Open the appropriate Expressions.gs file and use the debugger to identify the issue**

a) Use a breakpoint, the Variables pane and 'Step over' button

4. **Question: What is the issue? What solution would you propose?**

Stop

11.2.2 Lab: Implement the fix

Reference

PCF file code generation

Remember, PCF files support both incremental and bulk code generation.

- Save the changed (CTRL + S) PCF file or select **Build** menu → **Make the project** to kick off incremental code generation.
- Select **CodeGen** menu → **Generate Page Configuration Classes** or **Build** menu → **Rebuild project** to kick off bulk code generation.

The code generation of a PCF file results in the following:

- The .gs file is a Gosu class that represents the PCF type.
- The .pcfc is a binary file that describes the PCF file for the new PCF runtime.

1. **Implement your proposed fix**

   a) Fix the PCF method implementation in the ABContactDetailsCompanyDV.pcf.

   Important: do not modify the Expressions.gs file.

   b) Kick off the code generation process by saving the file (`CTRL + S`).

2. **Deploy the fix**

   a) Verify that you can see your change in the ABContactDetailsCompanyDVExpressions.gs. Reload the PCF changes (`ALT + SHIFT + L`).

11.2.3 Verification

Activity

Verify the work you have done

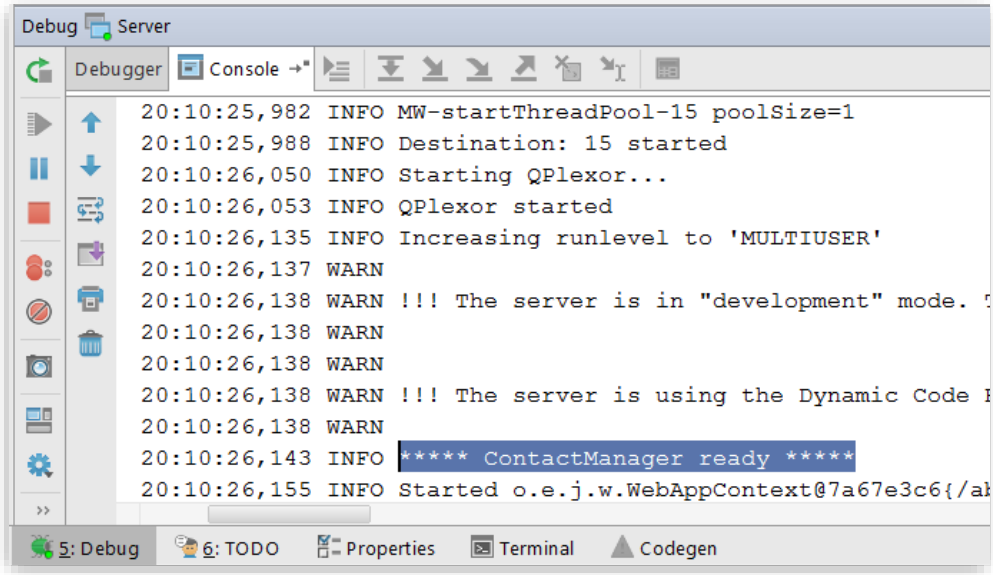
1. **Verify the issue has been resolved**
   a) Log in to TrainingApp as Alice Applegate.
   b) Search for and open the Albertson's contact.
   c) For the contact, navigate to the Details page.
   d) Click Edit.
   e) Click on Cascade Email Address button.
   f) Verify that only contacts without an email address have been updated

Stop

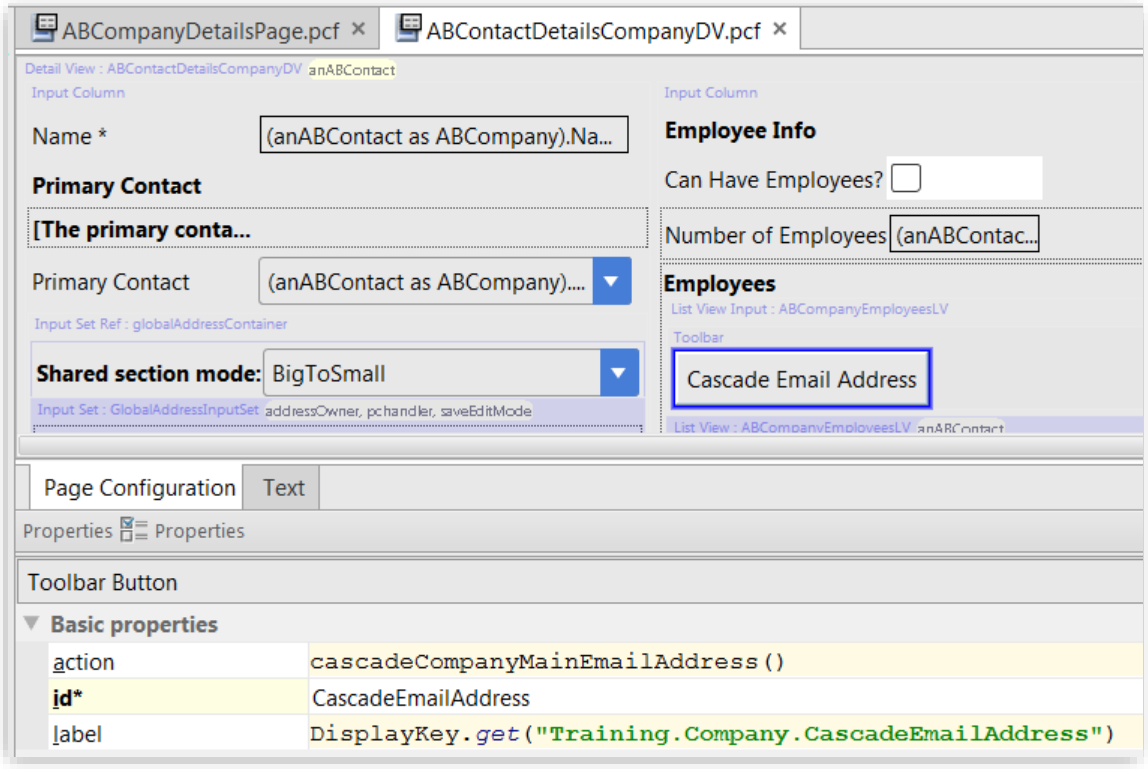
11.3 Lab Solution: Fix a bug

Solution

1. **Open Guidewire Studio for TrainingApp**



2. Locate the function called by the Cascade Email Address button

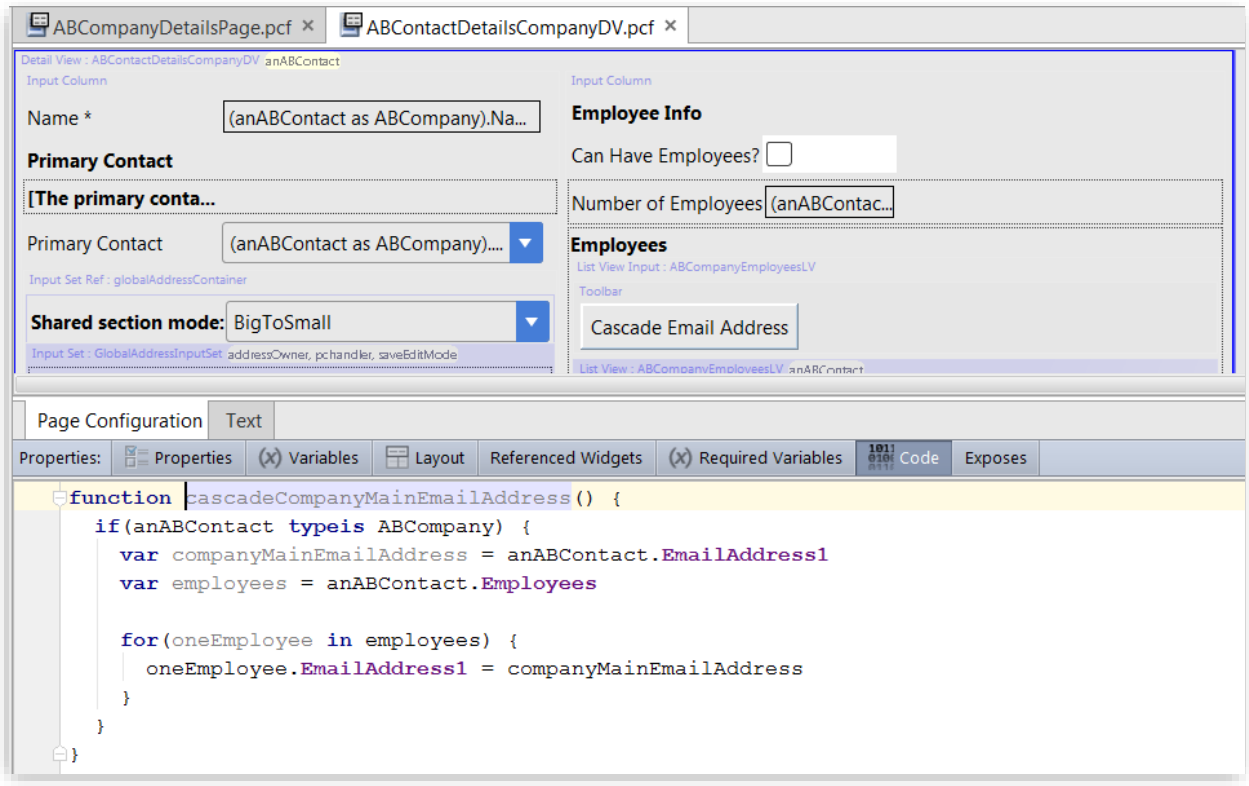


11.3.1 Lab Solution: Write it down

1. Question: What is the name of the method called by the Cascade Email Address button?

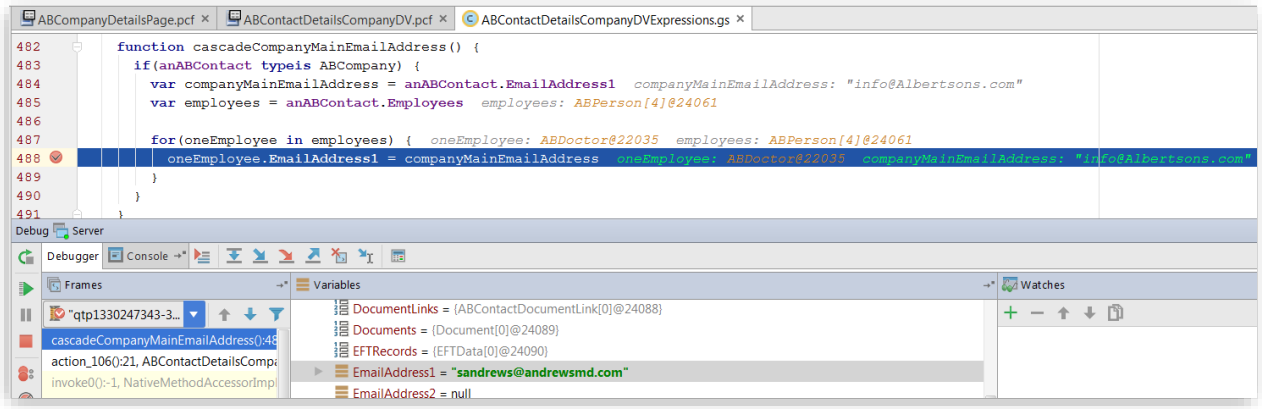
cascadeCompanyMainEmailAddress()

2. Go to the function declaration and verify it is a PCF function



3. **Open the appropriate Expressions.gs file and use the debugger to identify the issue**

   a) Use a breakpoint, the Variables pane and 'Step over' button



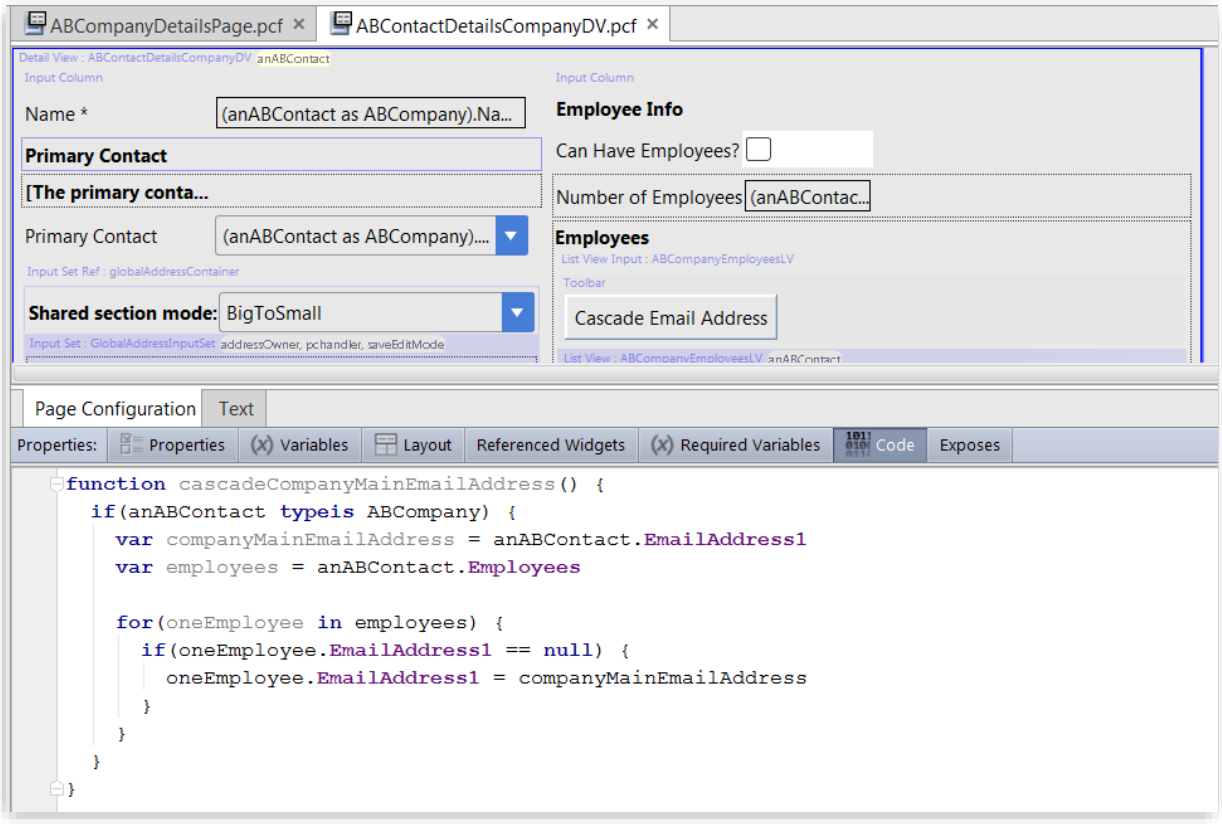
4. **Question: What is the issue? What solution would you propose?**

The highlighted line in the above screenshot will be executed, even though we can see that the EmailAddress1 field of the oneEmployee is already set to sandrews@andrewsmd.com. Suggested solution: put the line in an if statement and execute it only if the oneEmployee.EmailAddress1 is null.

11.3.2 Lab Solution: Implement the fix

Solution

1. Implement your proposed fix

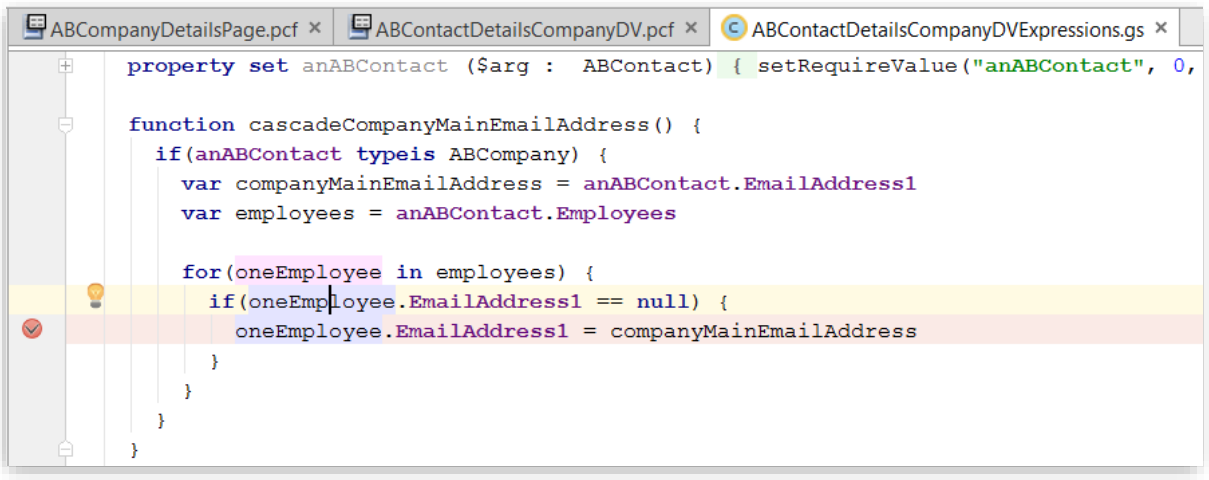


You also have the option to copy and paste it from below. However, it is recommended that you type the code in.

```
function cascadeCompanyMainEmailAddress() {
```

```
  if(anABContact typeis ABCompany) {
    var companyMainEmailAddress = anABContact.EmailAddress1
    var employees = anABContact.Employees

    for(oneEmployee in employees) {
      if(oneEmployee.EmailAddress1 == null) {
        oneEmployee.EmailAddress1 = companyMainEmailAddress
      }
    }
  }
}
```

2. Deploy the fix



Appendix A: Gosu Training Cheat Sheet

This section gives you a quick overview of the basic Gosu syntax needed to complete the exercises and included here only as a cheat sheet for the training. This cheat sheet is not part of the official Guidewire documentation and should not be considered as a complete syntax reference. The language is not static across time and it changes with product versions. (It may change even within the same major release.) Customers should always find and use the official ***Gosu Reference Guide*** included with the product. It is recommended to read the ***Gosu Introduction*** chapter of the *Gosu Reference Guide* to get a more expanded tour through the most commonly used language features.

Tips

Important!

Semicolon is allowed but ignored

Compiler determines end of statements based on syntax, thus no terminator required. Semicolon is allowed but ignored. The Guidewire recommendation is to NOT use semicolons. Guidewire also recommends using carriage returns and white space to make it clear to others reading the code where a given statement ends.

Important!

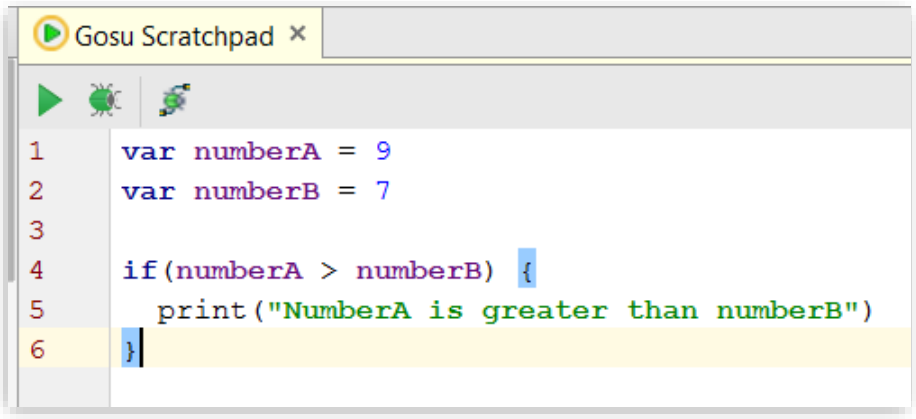
Gosu is case sensitive

Refer to variables, types and symbols using the exact same case every time, otherwise the application won't compile.

Hint

{ and }

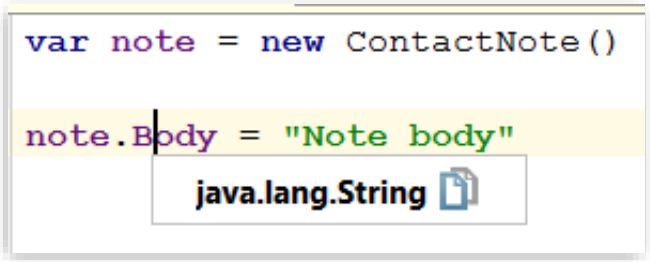
In Gosu, each opening '{' must have a matching '}'. Moving the flashing cursor after the '{' or '}' highlights the pairs, so it's easier to identify any errors.



Hint

How can I easily determine the type of Gosu elements (variables, properties, function return types) in a Gosu editor?

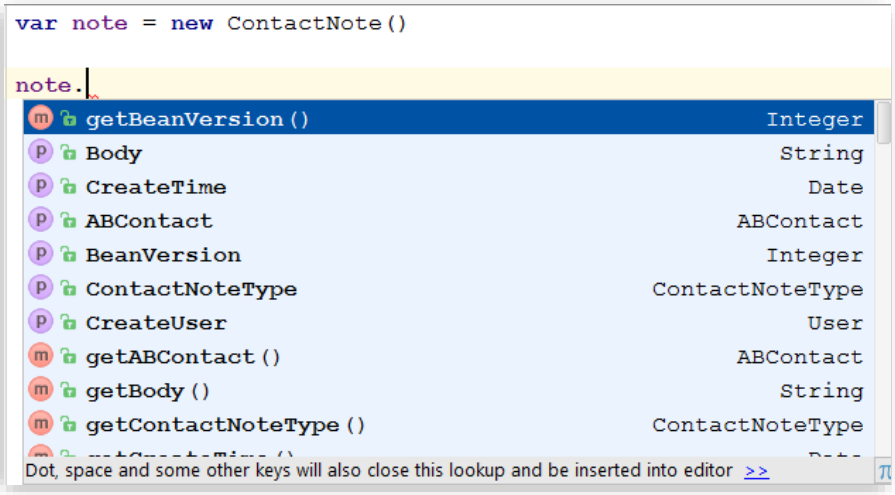
Use `CTRL + SHIFT + G`

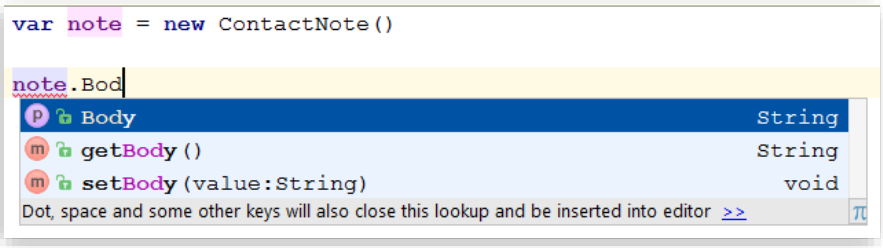


Hint

How can I use the code completion feature in a Gosu editor?

Move your flashing cursor after the dot character and hit `CTRL + SPACE` This will display the smart help. Typing will filter and narrow the list of suggestions. Just hit enter to select the property or function from the list.





Basics

GOSU Quick overview of basic Gosu syntax

Feature	Syntax	Try it!

Gosu primitives	Gosu primitives exist for compatibility with the Java language.
	
	Primitive types cannot hold the null value. (null is a special value that means empty)
	
	Primitive types can be coerced to non-primitive versions or back again in Gosu. During the conversion the null value will be converted to the default value of the primitive type (0
		`// declaring a primitive with initial value`
		`var i : int = 5`
		
		`// declaring non-primitive with null value`
		`var i2 : Integer = null`
		
		`// non-primitive type is coerced to primitive type`
		`// null value will be converted to 0`
		`i = i2`

	for any numeric data and false for boolean data). Primitives: `int`, `byte`, `char`, `short`, `long`, `float`, `double`, `boolean`	`print(i)`
Print message	**Important: never use it in production, use Loggers instead** `print( message/value to print)`	`print("Hello World")`

Comments	One line comment:
	
	Highlight the text in Studio and use the `CTRL + /` keystroke
	
	Multiple lines comment:
	
	Highlight the text in Studio and use the `CTRL + SHIFT + /` keystroke
		`//one line comment`
		
		`/* multiple`
		
		`lines`
		
		`comment */`

Variable declaration	`var variableName = initialValue`
	
	`var variableName : Type`
	
	`var variableName : Type = initialValue`
		`//****Variable declaration****`
		`var variableToStoreANumber : int`
		`var variableToStoreACharacterSequence : String`
		`var anAdjudicator : ABAdjudicator`
		`var anAddress : Address`
		`// Declaring a variable with type and initial value`
		`var booleanVariable : boolean = true`
		`// Declaring variables with initial value - the type isn't required.`
		`// Gosu is statically typed, but uses type inference to eliminate the vast majority of syntax`
		`// overhead usually involved with static typing`
		
		`//a Number`
		`var myVar = 5`
		`//a String`
		`var myStringVariable = "Hello World"`
		`// Assigning new value to a variable '=' is the assignment operator`

		`myVar = 8`
		`// Printing out the variable`
		`print(myVar)`
		
		`//Must declare a type because it can't be inferred`
		`var guess : String = null`

If statement	`if(condition) {`
	
	`block of code`
	
	`}`
		`var numberA = 9`
		`var numberB = 7`
		
		`// Each opening '{' must have a closing '}' - Moving the flashing cursor after the '{' or '}' highlights the pairs`
		`if(numberA > numberB) {`
		`print("NumberA is greater than numberB")`
		`}`

If-else statement	`if(condition) {`
	
	`block of code`
	
	`} else {`
	
	`block of code`
	
	`}`
		`var numberA = 9`
		`var numberB = 7`
		
		`if(numberA > numberB) {`
		`print("NumberA is greater than numberB")`
		`} else {`
		`print("NumberA is NOT greater than numberB")`
		`}`

If-elseif-else statement	There could be multiple else if statements.
	
	`if(condition) {`
	
	`block of code`
	
	`} else if(condition) {`
	
	`block of code`
	
	`} else {`
	
	`block of code`
		`var numberA = 9`
		`var numberB = 7`
		
		`if(numberA > numberB) {`
		`print("NumberA is greater than numberB")`
		`} else if(numberA == numberB) {`
		`print("Number A and NumberB are equal")`
		`} else {`
		`print("NumberA is less than numberB")`
		`}`

	`}`	

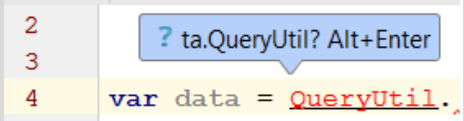
Ternary operator	Similar to if-else, but can only return expressions and cannot execute statements
	
	`condition ? valueiftrue : valueiffalse`
		`var numberA = 9`
		`var numberB = 7`
		
		`// Expressing the previous IF-ELSE with the ternary operator: the true and false cases MUST return a value, cannot call void functions there, e.g. print()`
		`var result = numberA > numberB ? "NumberA is greater than numberB" : "NumberA is NOT greater than numberB"`
		`print(result)`

Working with Strings	`+` concatenates two or more values into a single String
	
	`+=` concatenates variable with a right side expression
	
	`${expression}` evaluates an expression and inserts it in a String
		`var anABContact : ABContact`
		
		`anABContact = QueryUtil.findPerson("ab:5")`
		
		`print("The contact lives in " + anABContact.PrimaryAddress.State + " and he loves it!")`
		`print("This contact lives in ${anABContact.PrimaryAddress.State} and he loves it!")`
		
		`var contactStateOutput = "The contact lives in "`
		`contactStateOutput += anABContact.PrimaryAddress.State`
		`contactStateOutput += " and he loves it!"`

Logical operators	Equality: `==, !=`
	
	Relational: `>, <, >=, <=`
	
	Logical operators: `&&, and, ||, or, !, not`
	
	Notes:
		`var counter1 : int`
		`var counter2 = 0`
		
		`counter1 = counter2 + 1`
		
		`if (counter1 == 1) {`
		`  print("equal to 1")`
		`}`
		`if(counter1 != counter2) {`
		`  print("not equal to counter2")`
		`}`
		`if((counter1 > 0) and (counter1 < 10)) {`

	`==` - in case of objects, it tests for object equality, just like `.equals()`
	
	`===` - in case of objects, it tests for referential equality
		`print("greater than 0, less than 10")`
		`}`
		`if((counter1 >= 0) or (counter1 <= 10)) {`
		`print("equal to or greater than 0")`
		`print("or less than or equal to 10")`
		`}`

Guidewire static method libraries	Guidewire applications have static method libraries to solve common problems related to Dates, Numbers, Strings. It is recommended to check the available functions of these classes.
	
	`gw.api.util.DateUtil`
	
	`gw.api.util.Math`
	
	`gw.api.util.StringUtil`
		`var currentDate = gw.api.util.DateUtil.currentDate()`
		`var randomNumber = gw.api.util.Math.random(1000)`
		`var output = gw.api.util.StringUtil.formatDate(currentDate, "YYYY-MM-DD")`
		`if (randomNumber > 500) {`
		`print (currentDate + " " + randomNumber + " " + output )`
		`}`

Uses operator	To use types and namespaces in Gosu without fully qualifying the full class name including the package, use the Gosu `uses` operator.
	
	You can type in the class name and method and Studio will try to resolve the fully qualified name for you if the package or namespace has not been already declared. Use `ALT+ENTER` to see the Import Class options.
	
	 
		`uses gw.api.util.StringUtil`
		`uses gw.api.util.DateUtil`
		`uses gw.api.util.Math`
		
		`var currentDate = DateUtil.currentDate()`
		`var randomNumber = Math.random(1000)`
		`var output = StringUtil.formatDate(currentDate, "YYYY-MM-DD")`
		`if (randomNumber > 500) {`
		`print (currentDate + " " + randomNumber + " " + output )`
		`}`

Loading contacts from the Database	TrainingApp has a utility class with static methods to load contacts from the database. This is only for educational purposes and not available in other applications.	`var anABContact : ABContact`
		
		`//ALT + ENTER - to import the full package path for the class`
		`//CTRL + Q - quick documentation for a method`
		`// Shortcut to load a Contact from the Database based on the PublicID`

	`ta.QueryUtil.findXXX(public ID)`	`anABContact = ta.QueryUtil.findPerson("ab:5")`

Creating new objects	`var variableName = new Type()`	`var anABPerson = new ABPerson()`
		`var primaryAddress = new Address()`
		`var list = new java.util.List()`

Type cast	Lets you access properties on the subtype level. Usually a type check is executed to be sure the object is compatible with the new type.
	
	`(object as ChildSubType).PropertyOrMethod`
		`var anABContact : ABContact = ta.QueryUtil.findPerson("ab:5")`
		
		`print((anABContact as ABPerson).Age)`
		`//successful`
		`print((anABContact as ABCompany).NumberOfEmployees) // fails during runtime`

Type check	Tests if the object is of a give type. It supports automatic downcasting (No need to cast to object explicitly).
	
	`object typeis Type`
	
	The expression will return true if the object is of the Type **or any of its subtypes.**
		`var anABContact : ABContact = ta.QueryUtil.findPerson("ab:5")`
		
		`// Testing for subtypes`
		`if(anABContact typeis ABPerson) {`
		`// Inside the block anABContact behaves like an ABPerson since it passed the type test with the typeis operator`
		`// This case will be executed for any of the ABPerson subtypes (e.g. ABDoctor, ABAttorney, ABPolicyPerson, etc.)`
		`  print(anABContact.Age)`
		`} else {`
		`  print("Not a person")`
		`}`
		
		`// Attempting to use the same expression as above outside the scope of the THEN clause without casting to the ABPerson type will cause a compilation error. Try uncommenting the next line to see this.`
		`//print(anABContact.Age)`

Type check using the Subtype property	Subtyped and subtype entities automatically have the Subtype property which identifies their correct type. You can use the Subtype	`var anABContact : ABContact = ta.QueryUtil.findPerson("ab:5")`
		
		`if(anABContact.Subtype == typekey.ABContact.TC_ABPERSON) {`
		`  print((anABContact as ABPerson).Age)`

	property to execute a type check similarly to the typeis operator:
	
	`object.Subtype == typekey.Type.Subtype`
	
	However, there are two important differences:
	- using the Subtype for typecheck doesn't automatically downcast
	- the expression will return true only if the object is of the specified Type **and not for its subtypes**
		`} else {`
		`// This case will be executed for any of the ABPerson subtypes (e.g. ABDoctor, ABAttorney, ABPolicyPerson, etc.)`
		`  print("Not a person")`
		`}`

UI friendly names	`entity.DisplayName`
	
	`typekey.DisplayName`
		`var anABContact : ABContact = ta.QueryUtil.findPerson("ab:5")`
		
		`//these two are the same, they both invoke the Entity Name for the given entity (which is similar to toString() in Java)`
		`print(anABContact)`
		`print(anABContact.DisplayName)`
		
		`// The printed format of typecodes and dates depends on how we print them (concatenating? DisplayName? Code?)`
		`print(anABContact.PrimaryAddress.State) // long format`
		`print(anABContact.PrimaryAddress.State.Code) // short format`
		`print("This is the state of the primary address: " + anABContact.PrimaryAddress.State) // short format`
		`print("This is the state of the primary address: " + anABContact.PrimaryAddress.State.DisplayName) // long format`

		`var todayDate = DateUtil.currentDate()`
		`print(todayDate) // long format`
		`print("Today is: " + todayDate) // short format`
		`print("Today is: " + todayDate.toString()) // long format`

Working with typecodes and dates as string	The application reformats typecodes and dates when concatenated with a String.	`// The printed format of typecodes and dates depends on how we print them (concatenating? DisplayName? Code?)`
		`print(anABContact.PrimaryAddress.State) // long format`
		`print(anABContact.PrimaryAddress.State.Code) // short format`
		`print("This is the state of the primary address: " + anABContact.PrimaryAddress.State) // short format`
		`print("This is the state of the primary address: " + anABContact.PrimaryAddress.State.DisplayName) // long format`
		
		`var todayDate = DateUtil.currentDate()`
		`print(todayDate) // long format`
		`print("Today is: " + todayDate) // short format`
		`print("Today is: " + todayDate.toString()) // long format`

Null Safe invocation	`variable?.method()`	`// The following code can cause a NullPointerException if either the list or`
		`// the first element in the list is null. We can address this by`
		`// using the null-safe invocation operator ?.:`
		`var aList : java.util.LinkedList<String> = null`
		`if(aList?.get(0)?.isEmpty()) {`
		`  print("The first string is empty")`
		`}`
		
		`print("First List element: " + aList?.get(0))`
		`print("Is first list element an empty string? " + aList?.get(0)?.isEmpty())`

Null-safe Default Operator	Sometimes you want a default value if an expression is null. For this use case, Gosu supports the Elvis Operator:
	
	`variable = expression ?: defaultvalue`
		`var anABContact : ABContact = ta.QueryUtil.findPerson("ab:5")`
		
		`var middleName = anABPerson.MiddleName ?: "Unknown"`
		`print("MiddleName: " + middleName)`

Working with typekey fields	Typekey fields are defined in ETI, EIX and ETX files. They always get their values from the associated typelist.
	
	Assigning a new value:
	
	`entity.typekeyfield = Typelist.TC_Typecode`
	
	Testing for equality:
	
	`entity.typekeyfield == Typelist.TC_Typecode`
		`Important: the following code is not complete and cannot be executed in Gosu Scratchpad as presented here.`
		
		`var note = new ContactNote()`
		
		`note.ContactNoteType = ContactNoteType.TC_GENERAL`
		
		`if(note.ContactNoteType == ContactNoteType.TC_GENERAL) {`
		`  print("This is a general contact note")`
		`}`

Gosu classes, properties and functions

Quick overview of Gosu class, properties and function syntax

Feature	Syntax	Try it!

Class definition	A class is a template to create objects. A class can have properties and functions. The class could extend another class and implement any number of interfaces.	`class Card {`
		
		`}`

	class ClassName {
	
	}	

Variables	Instance variable definitions usually start with _
	
	The default visibility for instance variables is private
	
	Visibility levels:
	
	- private
	- internal
	- protected
	- public
		`class Card {`
		
		`  var _suit : String`
		`  var _rank : String`
		`  var _isFaceUp : Boolean`
		
		`}`

Properties	**Note:** Same syntax can be used in Enhancements
	
	Getter:
	
	`property get PropertyName() : Type {`
	
	`  return this._propertyName`
	
	`}`
	
	Setter:
	
	`property set PropertyName( newValue : Type ) {`
	
	`  this._propertyName = newValue`
	
	`}`
		`class Card {`
		
		`  var _suit : String`
		`  var _rank : String`
		`  var _isFaceUp : Boolean`
		
		`  property get Suit() : String {`
		`  return this._suit`
		`  }`
		
		`  property set Suit( newSuiteValue : String ) {`
		`  this._suit = newSuiteValue`
		`  }`
		
		`}`

Properties shortcut	There is a shorter way to expose properties using the as keyword	`class Card {`
		
		`  var _suit : String as Suite`
		`  var _rank : String as Rank`
		`  var _isFaceUp : Boolean as IsFaceUp`

	`var _variableName : Type as PropertyName` This will automatically generate a getter and setter.	`}`

Read-only Properties shortcut	`var _variableName : Type as readonly PropertyName`	`class Card {`
		
		`  var _suit : String as readonly Suite`
		
		`}`

Function	`function functionName( [parameters] ) : returnType {`
	
	`}`
	
	Notes:
	- the same function syntax can be used in Enhancements and PCF files as well
	- void keyword is optional
	- default visibility is public
		`class Card {`
		
		`  var _suit : String as Suite`
		`  var _rank : String as Rank`
		`  var _isFaceUp : Boolean as IsFaceUp`
		
		`  function turnFaceDown() : void {`
		`    this._isFaceUp = false`
		`  }`
		
		`}`
		
		`// Invoke a function`
		`var king = new Card()`
		`king.turnFaceDown()`

Static function	`static function functionName( [parameters] ) : returnType {`
	
	`}`
	
	Notes:
	- the same function syntax can be used in Enhancements and PCF files as well
	- void keyword is optional
		Important: this code cannot be executed in Gosu Scratchpad as presented here.
		
		`class MyStringUtil {`
		
		`  // Utility class`
		
		`  static function countSpaces(line: String): int {`
		`    var count = 0`
		`    for (ch in line) {`
		`      if (ch == ' ') {`
		`        count++`
		`      }`
		`    }`
		`    return count`
		`  }`

		}
		
		// Invoke a static function
		print(MyStringUtil.countSpaces("this is a test"))

Creating and initializing new object	var variableName = new Type() {
	
	:propertyName = initialValue,
	
	:propertyName = initialValue,…
	
	}
		class Card {
		
		var _suit : String as Suite
		var _rank : String as Rank
		var _isFaceUp : Boolean as IsFaceUp
		
		function turnFaceDown() : void {
		this._isFaceUp = false
		}
		
		}
		
		var heartKing = new Card() {
		:Suite = "Heart",
		:Rank = "King",
		:IsFaceUp = true
		}
		
		heartKing.turnFaceDown()

Arrays, loops and blocks

GOSU **Quick overview of array, loop and block syntax**

Feature	Syntax	Try it!
Declaring programmatic arrays	var arrayName : Type[] var arrayName : Type[] = new Type[size] var arrayName = new Type[size]	// Declaring programmatic array var myArray : String[] = new String[2] // Declaring programmatic array var arrayFoo : int[] arrayFoo = new int[5]

Accessing elements	`var element = array[index]`
	
	`array[index] = new value`
	
	Arrays are indexed starting from 0
		`var myArray : String[] = new String[2]`
		
		`myArray[0] = "First"`
		`myArray[1] = "Second"`
		
		`print(myArray[1])`

Size of the array	`array.length`
	
	`array.Count`
		`// Loading a contact from the Database (William Andy)`
		`var aContact = ta.QueryUtil.findPerson("ab:5")`
		
		`// Getting all the contactnotes (ContactNote[])`
		`var contactNotes = aContact.ContactNotes`
		
		`// Number of contact notes`
		`print(contactNotes.length)`

Data model arrays	Data model arrays are declared in the ETI, EIX and ETX files using the <array> tag. Every <array> tag must have an associated reverse <foreignkey> on the other entity.
	
	The framework automatically generates the following two methods:
	
	`parentEntity.addToArrayName (newElement)`
	
	`parentEntity.removeFromArrayName(elementToRemove)`
		Important: the following code is not complete and won't work from Gosu Scratchpad. Visit the Solution section of the Business Rules lesson for the complete example.
		
		`var note = new ContactNote()`
		`note.Subject = DisplayKey.get("Ext.ContactAssigned")`
		`note.Body = DisplayKey.get("Ext.NewContactWithFlagEntriesAutoAssigned"`
		`, aBContact.AssignedUser, aBContact.FlagEntries.length)`
		`note.ContactNoteType = ContactNoteType.TC_GENERAL`
		
		`aBContact.addToContactNotes(note)`

For loop	For loop allows you to iterate over both arrays and anything that implements the java.lang.Iterable interface
	
	`for( anObject in collection/range ) {`
		`// Loading a contact from the Database (William Andy)`
		`var aContact = ta.QueryUtil.findPerson("ab:5")`
		
		`// Getting all the contactnotes (ContactNote[])`
		`var contactNotes = aContact.ContactNotes`

	block of code referencing anObject
	
	}
	
	You can specify ranges using the .. operator.
	
	start..end
	
	Examples:
	
	1..10
	
	"01/01/2016".toDate().."02/01/2016".toDate()
		// Traversing the contactnotes array and printing out in '1/1/2013 - General' format
		for(oneContactNote in contactNotes) {
		print(oneContactNote.CreateTime + " - " + oneContactNote.ContactNoteType)
		}
		
		// Range operator: printing number from 1 to 10
		for(i in 1..10) {
		print(i)
		}

Indexed for loop	Index starts from 0 and will be incremented by 1 in every loop cycle
	
	for(anObject in collection index i) {
	
	block of code referencing anObject
	
	}
		// Loading a contact from the Database (William Andy)
		var aContact = ta.QueryUtil.findPerson("ab:5")
		
		// Getting all the contactnotes (ContactNote[])
		var contactNotes = aContact.ContactNotes
		
		// Traversing the contactnotes array and printing out in '1 - 1/1/2013 - General' format
		for(oneContactNote in contactNotes index i) {
		print((i +1) + " - " + oneContactNote.CreateTime + " - " + oneContactNote.ContactNoteType)
		}

Block	Blocks (also called closures or lambda expressions) are a simple way to specify an inline function. They have a lot of uses, but they really shine in data structure manipulation
	
	\ variableName -> condition
		// Loading a contact from the Database (William Andy)
		var aContact = ta.QueryUtil.findPerson("ab:5")
		
		// Getting all the contactnotes (ContactNote[])
		var contactNotes = aContact.ContactNotes

	Frequently used array functions:
	- `hasMatch`: determines if any element in the array matches the condition
	- `where`: Retrieves a target array that consists of all the members of a source array that match a given condition
	- `firstWhere`: returns the first element that matches the condition
	- `countWhere`: return the count of elements that match a given condition
		`// Finds the first contact note in the array that has the type General`
		`var isThereAContactNoteWithTypeGeneral = contactNotes.hasMatch( \ oneContactNote -> oneContactNote.ContactNoteType == ContactNoteType.TC_GENERAL )`
		`print("isThereAContactNoteWithTypeGeneralShorter: " + isThereAContactNoteWithTypeGeneralShorter)`

Appendix B: Quick reference

Gradle

Gradle

What is Gradle and how to work with it?

Gradle is an open-source build automation system based on a Groovy DSL (Domain Specific Language). It is a very flexible general purpose build tool. Gradle has three phases of build development: initialization, configuration, and execution. You can add source sets and add dependency configurations easily. Gradle supports incremental builds. It intelligently determines which parts of the build tree are up to date so that any tasks dependent upon those parts will not need to be re-executed. InsuranceSuite 9.0 applications implement the Gradle wrapper. The Gradle wrapper executes the required Gradle build on machines where Gradle is not installed. Using the Gradle wrapper enforces the usage of a particular Gradle version thus minimizing support issues.

Read me: All the commands below should be executed from Command Prompt (Command line) pointing to the project root. Follow these steps to quickly open a command prompt from the project root: SHIFT + right click on `C:\GW9\TrainingApp` and select '`Open command window here`' from the context menu. This works the

same way in all the other Guidewire InsuranceSuite 9.0 applications (PolicyCenter, BillingCenter, ClaimCenter and ContactManager).

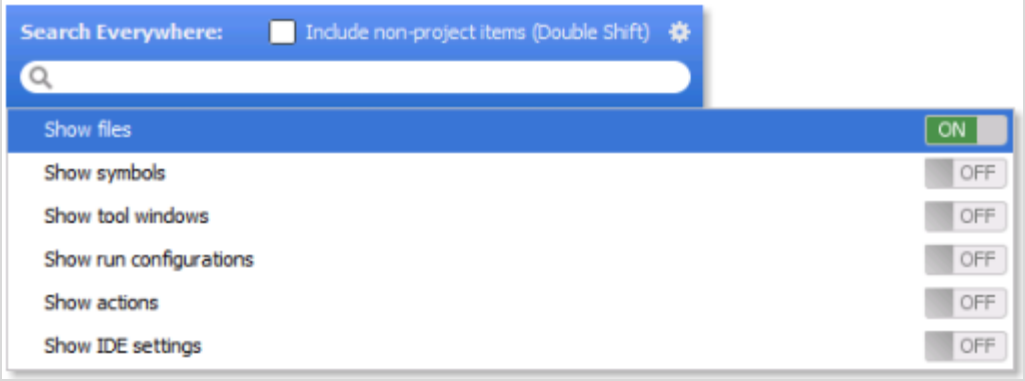
How to…	Command
…list all Gradle tasks?	`gwb tasks`
…start Studio?	`gwb studio`
…start the Server?	`gwb runServer -Dgw.port=8880`
…stop the Server	`gwb stopServer`
…regenerate the Data Dictionary?	`gwb genDataDictionary`
…regenerate Gosu Documentation?	`gwb gosudoc`
…drop the database?	`gwb dropDb`

Studio productivity

Studio

What are the most helpful shortcuts in Studio?

Keystroke	Description
`SHIFT + SHIFT` (double shift)	Search Everywhere – Guidewire Studio makes it possible to look for any item of the source code in a single action. In the popup window that opens, start typing the search string to narrow the suggestion list. It is also possible to configure the scope of the search everywhere. In the dialog, click the gear. Turn on or off the desired search scopes.

	
`CTRL + SHIFT + N`	To open any **file** in the editor quickly. (Navigate to file)
`CTRL + N`	To open any **class** in the editor quickly. (Navigate to class)
`CTRL + E`	To navigate to a recently **opened** file.
`CTRL + SHIFT + E`	To navigate to a recently **modified** file.
`ALT + HOME`	Activate navigation bar to navigate to a file.
`CTRL + SHIFT + F`	To initiate a text search in the specified path. (Find in path)
`CTRL + Q`	Shows the Gosu documentation of a function or property. Use it when the cursor is flashing between two letters in the function/property name.
`ALT + ENTER`	1) Creates displaykey 2) Generates uses statement 3) Generates method stubs

Guidewire Application

Guidewire Application URLs

URL syntax to access a Guidewire application: **http://hostName:port/appCode**

Application	URL
TrainingApp	**http://localhost:8880/ab**
BilingCenter	**http://localhost:8580/bc**
ClaimCenter	**http://localhost:8080/cc**
ContactManager	**http://localhost:8280/ab**
PolicyCenter	**http://localhost:8180/pc**

Guidewire Application shortcuts

These shortcuts and tools are in the platform, thus they work in all of the Guidewire InsuranceSuite 9.0 applications. (PolicyCenter, BillingCenter, ClaimCenter and ContactManager)

Read me: Use this shortcuts in the browser! These keystrokes work only if the `EnableInternalDebugTools` parameter is set to true in the `config.xml` file.

```
<param name="EnableInternalDebugTools" value="true"/>
```

Keystroke	Description
`ALT + SHIFT + I`	**To open Location Information** The window details the file structure and details the hierarchy of the location, screen, and any child container widgets. For each location and container widget, the name of the file in which it is referenced is listed. Location Info is also useful when Studio is not running.
`ALT + SHIFT + W`	**To open Widget Inspector** The Widget Inspector shows all the PCF files and widgets referenced in the active application browser window except for the workspace area. The Widget Inspector also shows the available variables and their current values.

`ALT + SHIFT + E`	**To open a PCF file in Studio from the Browser.** If you are running an open application project in Guidewire Studio and if internal tools are enabled, you can automatically open the location being viewed in the application.
`ALT + SHIFT + L`	**To reload PCF and displaykey changes without restarting the Server** If you are running an open application project in Guidewire Studio and if internal tools are enabled, you can reload all the page configuration files and display keys for the server. **Important:** If you reload PCF files while in edit mode, you may experience unpredictable results. For the current location, where there is a data modification in progress, the new PCFs may not be reloaded. Therefore, Guidewire recommends reloading PCF files while in read-only mode as it provides for more predictable results.
`ALT + SHIFT + T`	**To access the Server Tools and Internal Tools** You can execute administrative tasks from the Server Tools tab: - Running Batch Processes - Setting log level - Viewing logs, etc. You can perform different development tasks from the Internal Tools tab (**important**: this should be disabled in production!) - Reload development resources (PCF files, Web templates, Workflow engine, Display Names) - Load sample data - Test system clock (set it to the future)

Data Model

Entity definition files

Base application (Out-of-the-box) files are in `…configuration/config/Metadata/Entities` – All the files in this folder are read-only. It contains ETI and EIX files.

Custom files are in `…configuration/config/Extensions/Entities` – It contains ETI and ETX files.

File type	Description	Naming convention
ETI	A single Guidewire or custom entity declaration from scratch. The name of the file corresponds to the name of the entity being declared. This entity could be a top-level entity or a subtype.	New custom ETI file names should end with a suffix. E.g.: Interaction_Ext.eti, MedicalCase_Ext, PhoneCall_Ext, etc
EIX	A single Guidewire entity extension. The name of the file corresponds to the name of the Guidewire entity being extended. The EIX file contains extensions to the platform-level entities that are required for the base application and cannot be modified. EIX files are extensions to platform-layer entities created by Guidewire development to meet the needs of a given application's base data model.	*EIX files neither created nor modified by configuration developers.*
ETX	A single Guidewire or custom entity extension. The name of the file corresponds to the name of the entity being extended. Base application entities are read-only to prevent any conflicts during upgrade to the next version of the Guidewire application. The concepts of an Entity extension allows customers to safely add new fields to base application entities or to override certain attributes of existing base application entity fields. **An entity can have at most 1 ETX file.** Guidewire provides certain entity extensions as part of the base application configuration. Many of the extension index definitions address performance issues. Other extensions provide the ability to configure the data model in ways that would not be possible if the extension was part of the base data model. Do not simply overwrite a Guidewire extension with your own extension without understanding the full implications of the change.	An ETX file has the exact same name as the entity (ETI file) that it extends. E.g.: ABContact.eti + ABContact.etx, Claim.eti + Claim.etx Fields in the ETX file should end with a suffix. E.g.: WebAddress_Ext, IsStrategicPartner_Ext fields in ABContact.etx

Entity definition elements

These elements could be used in the ETI or ETX files to define new data fields or relationship fields in the entity.

Element	Description	Notes
<column>	Defines a field that stores a single data value.	If the datatype is varchar, the <column> must have a <columParam> subelement to specify the size. If the datatype is decimal, the <column> must have two <columnParam> subelements to specify the precision and scale.
<foreignkey>	Defines a one-to-one unidirectional relationship.	If you want to define a one-to-one bidirectional, then <foreignkey> has to be used in one of the entities and <one-to-one> must be used in the other. <one-to-one> identifies the owner of the relationship – the entity that could exist without the other.
<array>	An array defines a set of additional entities of the same type to associate with the main entity. The application automatically generates `addToArrayName(newObject)` and `removeFromArrayName(object)` functions on the parent entity to add and remove elements.	An array requires a reverse foreignkey because the array is virtual and maintained in code. The code assembles the array by executing queries against the database. In order to do this, the application must be able to query for all members of a given array. It can do this only if each member has a foreign key referencing its parent. For example, an ABContact entity includes an array of BankAccount entities. Thus, the BankAccount entity must have a foreignkey pointing to ABContact.
<typekey>	A typekey field is an entity defined field associated with a specific typelist. Referenced typelist contains typecodes whose values are the only possible value for the typekey field. A typekey can point to exactly one value in a typelist.	It is often rendered as a dropdown in the UI. <typekey> can have <typefilter> associated to it. A typefilter further limits the set of possible values for the typekey. It allows the typekey to get its possible values only from a specific subset of all the typelist values.

<column-override>	It allows configuration developers to override certain data fields on read-only base application entities (defined by the <column> element). **It can only be used in ETX files.**	It can override the following attributes**:** createhistogram, default, nullok, size, supportsLinguisticSearch, type. The overridden attribute should be less restrictive, otherwise it would result in potential data loss. If the overridden attribute is more restrictive then typically the database table structure needs to be updated manually.

Widget reference table

Widget reference table

This table helps you identify the correct atomic widget type based on the data type of the field.

The third column also tells you what should be specified as the value in the valueType attribute of the atomic widget.

Data type (Data Model)	Recommended widget(s) in a Detail View	Recommended widget(s) in a List View	Data type (Gosu) This should be the valueType of the widget
varchar	Text Input	Text Cell	java.lang.String
integer	Text Input	Text Cell	java.lang.Integer
decimal	Text Input	Text Cell	java.math.BigDecimal
text	Text Input	Text Cell	java.lang.String
	Text Area Input	Text Area Cell	
date, datetime	Date Input	Date Cell	java.util.Date
bit	Boolean Dropdown Input	Boolean Radio Cell	java.lang.Boolean
	Boolean Radio Button Input	Checkbox Cell	
	Check Box Input		
foreignkey	Range Input	Range Cell	*OtherEntityType*

	Range Radio Button Input		
typekey	TypeKey Input	TypeKey Cell	typekey.*TypelistName*
	TypeKey Radio Button Input		

Location reference table

Location reference table

This table summarizes the main characteristics of the different locations. Locations define the navigation. Every Location has at least one entrypoint that defines the required input parameters for the location.

Location	Description	Typical navigation method	Initially displays	In
Page	contains a single Screen, used in Location Groups. It has its own info bar, tab bar and actions menu.	go()	Screen	Screen area
Location Group	collection of Pages	go()	First child Page	Screen area
Wizard	an ordered collection of Screens. It has its own info bar, tab bar and actions menu. It can also have wizard buttons and two separate side bar sections (dependent and independent)	go() push()	First Screen	Screen area
Popup	contains a single screen and returns the user to the previous location once the popup is closed	push()	Screen	Originating frame
Worksheet	contains a single screen, rendered in the workspace area.	goInWorkspace()	Screen	Workspace frame

Forward	contains logic to execute before navigating to another location	go()	Nothing	N / A
Exit point	points to a URL outside of the Guidewire application	push	External web site	New browser window

Code generation, validation and deployment

Code generation and validation

How does code generation and validation work?

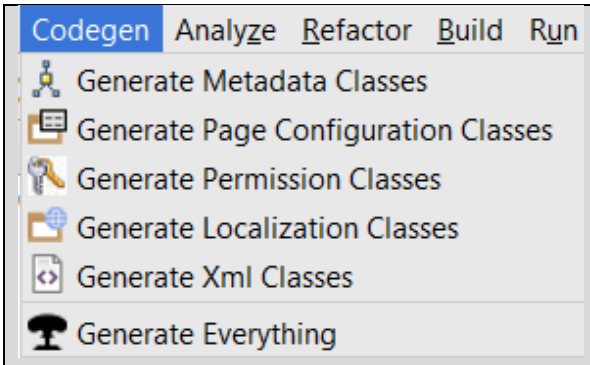
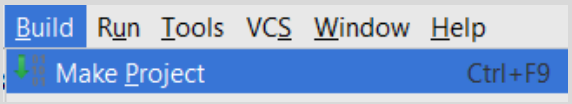
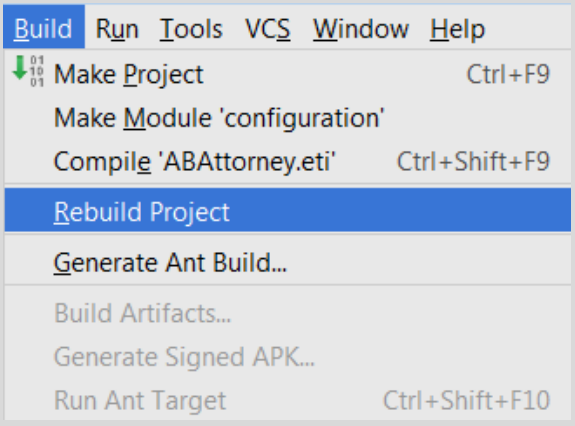
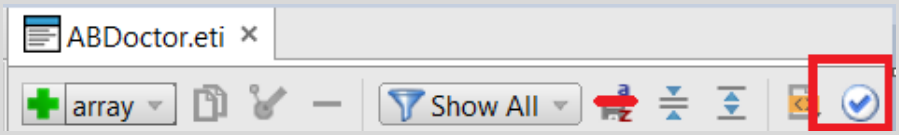
In InsuranceSuite 9.0, Guidewire Studio makes the process of code generation explicit and the generated code itself accessible. Internal code generators process resources like PCFs, entities and typelists in order to produce Gosu and Java classes.

There are two types of code generators: incremental and bulk. Both are supported, but incremental code generation is not an option for all resource types.

Incremental code generators process one or more changed resources files in isolation without the need to read all similar resources.

Bulk code generators cannot process individual resources files in isolation from the whole set. Bulk code generators are invoked explicitly and, to process the set of resources as a whole, they take longer.

How to invoke code generation?	Description	Incremental or bulk?
On Save – CTRL + S	Saving one or more modified resources files will invoke the associated code generator for the modified resource(s) only. Studio reports any errors in the Codegen tool window.	Only incremental code generators are invoked on save for performance reasons.
From Codegen menu	You can invoke individual code generators using one of the menu items. Studio reports any errors in the Codegen tool window.	Invoked this way, all code generators run in *bulk mode*, meaning

<table>
<tr>
<td> 
</td>
<td></td>
<td>Studio processes all the relevant resources.</td>
</tr>
<tr>
<td>During project make
 
</td>
<td>Studio invokes incremental code generators for the sets of the files that changed since the last project make or rebuild. Studio reports errors in the Messages tool window.</td>
<td><u>Incremental</u></td>
</tr>
<tr>
<td>During project rebuild
 
</td>
<td>Studio invokes the code generators in bulk mode on all resources. Studio reports errors in the Messages tool window.</td>
<td><u>Bulk mode</u></td>
</tr>
<tr>
<td>For Data Model entities only you can also click on the validate icon.
 
</td>
<td>This invokes the associated code generator for the current entity. Studio reports any errors in the Codegen tool window.</td>
<td>Only an <u>incremental</u> code generator is invoked for the open entity.</td>
</tr>
</table>

File types	Description	Output of code generation
PCF files	Studio supports incremental code generation for PCF files.	The code generation of the PCF files results in the following: - A .gs file in the …/configuration/generated/pcf package. This file a Gosu class that represents the PCF type - A .pcfc file in the …/configuration/generated/pcfc/config package. This is a binary file that describes the PCF file for the PCF runtime. The file is not human readable. - The Expression.gs file in the …/configuration/generated/pcfc/expressions package. This is a Gosu class that contains all container and widget expressions. Files in the pcfc.expressions package are also for the PCF runtime. You can debug these files.
ETI / ETX / TTI / TTX files	Studio supports incremental code generation for Entities and Typelists.	For all Entities and Typelists Studio generates Java classes.
XSD / WSDL and GX Model files	Studio performs bulk code generation for XSD / WSDL and GX Model files.	Studio creates Java classes in schema-dependent packages.

Deployment

How to deploy different configuration resources?

Guidewire Studio's incremental and bulk code generation plays a significant role in how developers prepare to deploy resources in a local development environment. Code generation creates the classes required for compilation and runtime.

Deploying Gosu changes in a local development environment, only works if the DCEVM (Dynamic Code Evolution Virtual Machine) is installed on your development machine.

Note: Restarting the server always deploys all changed resources.

Resource Name	New	Modified
Entity Name	**RESTART**	**RESTART**
Entity / Entity Extension	**RESTART**	**RESTART**
Typelist / Typelist Extension	**RESTART**	**RESTART**
Display Key	**ALT + SHIFT + L**	**ALT + SHIFT + L**
Page Configuration File	**ALT + SHIFT + L / RELOAD**	**ALT + SHIFT + L / RELOAD**
Rule	**RELOAD (RESTART – If a new Rule and new child rule were both created)**	**RELOAD**
Gosu class	**RELOAD (RESTART – If the Gosu class was created in a new package)**	**RELOAD**
Gosu Enhancement	**RELOAD (RESTART – If the enhancement was created in a new package)**	**RELOAD**

Appendix C: Troubleshooting and processes

In this section you will find how to resolve common errors and how to execute different development related processes.

Restoring the development database

Restoring the development database

How to restore the database? The database becomes corrupted if the database upgrade fails for any reason. This will prevent the server from starting. In development environments it is usually safe to just simply drop the database and restore it to its initial state. Remember, if you drop the database you just only loose the data (contacts that you might have created) and **not** your configuration. Your database configuration is in the Data Model, defined by the entities and typelists.

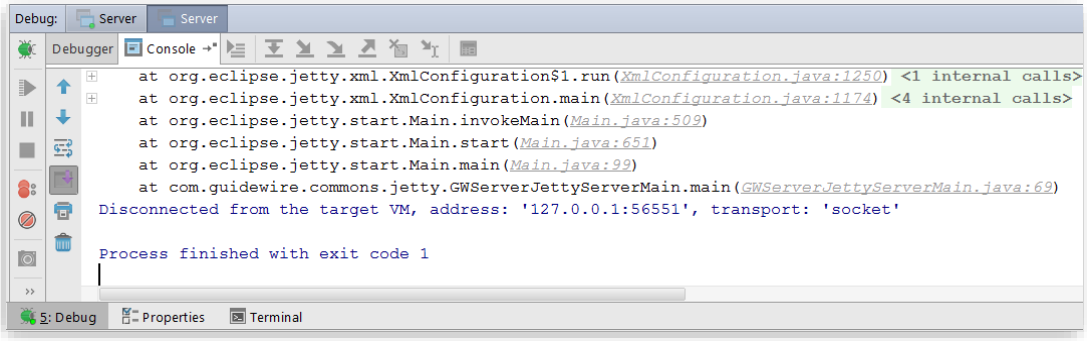
1) Open a command window pointing to the project root folder
2) From the command window, execute the `gwb dropDb` Gradle task
3) Start the server using the Run/Debug server button in Studio
4) Open the browser and log in to the application with **su/gw** after you see the `***[Guidewire application name] ready***` message in the server log
5) Go to `Administration` → `Monitoring` → `Message Queues`. Select all the message destinations and suspend them
6) Use the `ALT + SHIFT + T` keystroke to access the Internal Tools. Go the Internal Tools -> Sample data and load the sample data by clicking on one of the buttons – depending on which dataset you want to load. Wait until you see the Sample data successfully loaded message in the UI – this may take several minutes, depending on the dataset size.
7) Return to the Guidewire application by clicking on `Actions` → `Return to application`
8) Go to `Administration` → `Messaging`. Select all the message destinations and resume them
9) Log out and log back in as **aapplegate/gw**

Address already in use: bind

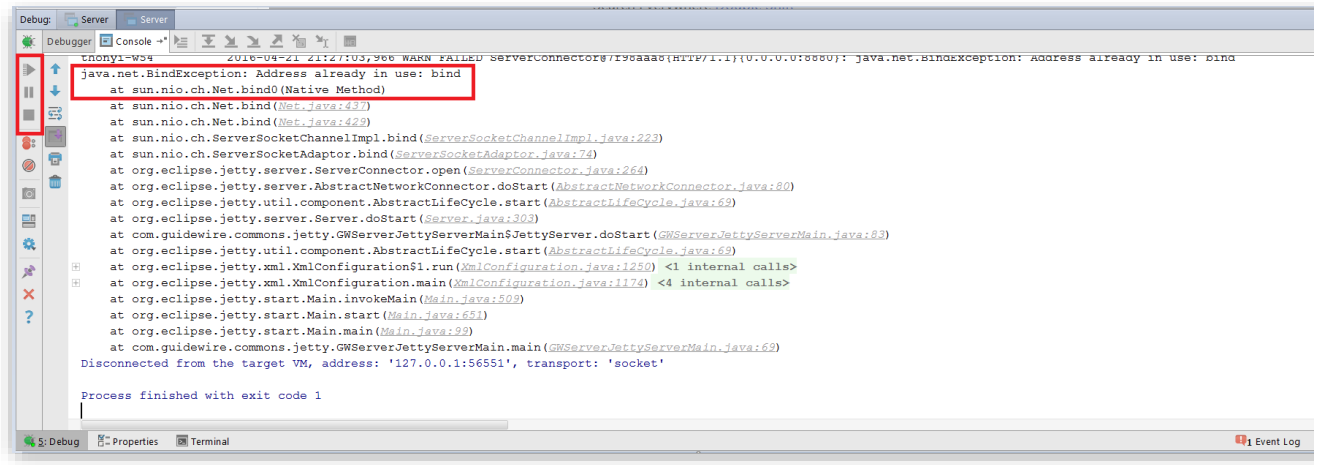
Server fails to start/restart and server logs shows error message: java.net.BindException: Address already in use: bind

You have been waiting forever for the server log to show the **** [ApplicationName] ready **** message? Sometimes the server doesn't start/restart successfully. This could be because of any configuration error in the project; or because there is another server instance already running.

If you see multiple lines starting with the ‘at’ word in your server log, that means there was an error. For example:

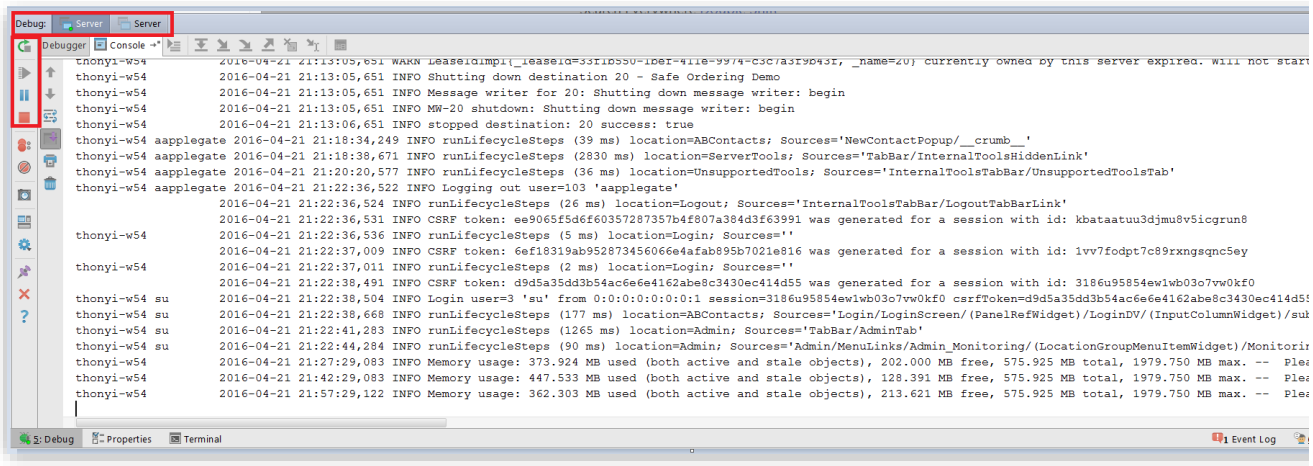


First, always make sure that you can see enough information in your server log to be able to correctly identify the issue. You can always just simply resize the Debug tool window by hovering over the top and dragging it. Then start scrolling up in the server log, to the line just right before the first ‘at …..’. That line identifies the issue. For example, in our case it’s a **java.net.BindException: Address already in use: bind:**



This usually means another server instance is already running. Notice that on the left hand side the resume, pause and stop icons are inactive, indicating that the server didn’t start.

Also notice there are two server tabs at the top of the Debug window! Select the first tab, and you will see that the icons on the left hand side are active:



This means a server instance is already running from Studio. Click on the stop button to stop it and then you will be able to start the server the usual way.

Note: The second server tab (failed) stays open even after successfully restarting the server. To avoid confusion in the future, just right click on the second tab and select `Close tab`.

(Sometimes the same error could happen if a third party software is blocking the same port. If that is the case, please ask your Guidewire Instructor to help you.)