



InsuranceSuite 10 Fundamentals: Kickstart

Student Workbook

Labs and Tutorials

Guidewire Proprietary & Confidential – DO NOT DISTRIBUTE

Copyright © 2018. Guidewire Software, Inc.

For information about Guidewire's trademarks, visit <http://www.guidewire.com/legal-notices>.

Document Published: 2020-09-10

Table of Contents

Introduction	6
Lesson 1 Introduction to Guidewire Configuration	7
1.1 Prerequisites.....	7
1.2 Lab: Working with Guidewire Studio	7
1.3 Lab: Explore TrainingApp	8
1.4 Lab: Create contacts.....	9
1.5 Lab Solution: Working with Guidewire Studio	11
1.6 Lab Solution: Explore TrainingApp	13
1.7 Lab Solution: Create contacts	14
Lesson 2 Introduction to the Data Model.....	20
2.1 Prerequisites.....	20
2.2 Lab: Explore the Data Dictionary.....	20
2.3 Lab Solution: Explore the Data Dictionary	22
Lesson 3 Extending the Data Model.....	25
3.1 Prerequisites.....	25
3.2 Lab: Modify an existing Entity Extension	25
3.3 Lab: Extend a base application Typelist	27
3.3.1 Configuration.....	28
3.3.2 Verification.....	29
3.4 Lab Solution: Modify an existing Entity Extension	30
3.5 Lab Solution: Extend a base application Typelist	34
Lesson 4 The User Interface Architecture.....	37
4.1 Prerequisites.....	37
4.1.1 Investigation: Explore the page structure	38
4.2 Lab: Write it down	38
4.3 Lab: Modify the Basics card on the Summary page	40
4.3.1 Configuration.....	41
4.3.2 Verification.....	41
4.4 Lab Solution: Write it down	42
4.5 Lab Solution: Modify the Basics card on the Summary page	46
Lesson 5 Atomic widgets	49
5.1 Prerequisites.....	50
5.2 Lab: Practice the Dot notation	50

5.2.1	Write it down	50
5.3	Lab: Determining Widgets and their requirements	51
5.4	Lab: Add new Atomic Widgets	53
5.4.1	Verification	55
5.5	Bonus Lab: Investigate optional widget properties	56
5.5.1	Write it down	57
5.6	Lab Solution: Practice Dot notation	58
5.7	Lab Solution: Determining widgets and their requirements.....	59
5.8	Lab Solution: Add new atomic widgets.....	60
5.9	Bonus Lab Solution: Investigate optional widget properties	66
Lesson 6	Detail Views.....	68
6.1	Prerequisites.....	69
6.2	Lab: Reusable Detail View Panel	69
6.2.1	Create the reusable Detail View Panel.....	70
6.2.2	Reference the new Detail View Panel on the Summary screen's Analysis Card.....	71
6.2.3	Reference the new Detail View Panel on the Analysis page.....	71
6.2.4	Verification	72
6.3	Lab: Toolbar and edit buttons.....	74
6.3.1	Verification	75
Lab:	Write it down	75
6.4	Lab Solution: Reusable Detail View Panel.....	76
6.5	Lab Solution: Reference the new Detail View Panel on the Summary screen's Analysis Card... ..	79
6.5.1	Solution: Reference the new Detail View Panel on the Analysis page	83
6.6	Lab Solution: Toolbar and edit buttons.....	84
6.7	Lab Solution: Write it down	85
Lesson 7	Introduction to Locations	87
7.1	Prerequisites.....	88
7.2	Lab: Location user story #1	88
7.2.1	Write it down	89
7.2.2	Configuration.....	89
7.2.3	Verification	91
7.3	Lab: Location user story #2	92
7.3.1	Write it down	92
7.3.2	Configuration.....	93
7.3.3	Verification	95
7.4	Lab Solution: Location user story #1	97
7.4.1	Write it down	97

7.4.2 Configuration.....	97
7.5 Lab Solution: Location user story #2	98
7.5.1 Write it down	98
7.5.2 Configuration.....	99
Lesson 8 Introduction to Gosu.....	100
8.1 Prerequisites.....	100
8.2 Lab: Coding with Gosu.....	100
8.2.1 Write it down	100
8.2.2 Configuration.....	101
8.2.3 Verification.....	101
8.3 Lab: Working with arrays	102
8.3.1 Write it down	102
8.3.2 Configuration.....	103
8.3.3 Verification.....	103
8.4 Lab Solution: Coding with Gosu	104
8.4.1 Write it down	104
8.4.2 Configuration.....	105
8.5 Lab Solution: Working with arrays	106
8.5.1 Write it down	106
8.5.2 Configuration.....	106
Lesson 9 Gosu Rules.....	108
9.1 Prerequisites.....	109
9.2 Lab: Create a new Gosu Rule.....	109
9.2.1 Write it down	109
9.2.2 Configuration.....	111
9.2.3 Verification.....	112
9.3 Lab Solution: Create a new Gosu Rule	112
9.3.1 Write it down	113
9.3.2 Configuration.....	113
Lesson 10 Enhancements	119
10.1 Prerequisites.....	119
10.2 Lab: Create a new enhancement	120
10.2.1 Modify the PCF configuration	120
10.2.2 Verification.....	122
10.3 Lab Solution: Create a new enhancement	124
10.3.1 Modify the PCF configuration	127

Lesson 11 Code Generation and Debugging	131
11.1 Prerequisites.....	132
11.2 Lab: Fix a bug.....	132
11.2.1 Lab: Write it down.....	133
11.2.2 Lab: Implement the fix	134
11.2.3 Verification	135
11.3 Lab Solution: Fix a bug.....	135
11.3.1 Lab Solution: Write it down	136
11.3.2 Lab Solution: Implement the fix.....	138
Appendix A: Gosu Training Cheat Sheet.....	142
Basics	144
Gosu classes, properties and functions.....	152
Arrays, loops and blocks.....	155
Appendix B: Quick reference	159
Guidewire TrainingApp Example List.....	159
Gradle	160
Studio productivity	160
Guidewire Application.....	161
Data Model.....	163
Widget reference table	165
Location reference table	166
Code generation, validation and deployment.....	167
Appendix C: Troubleshooting and processes	171
Restoring the development database	171
Address already in use: bind	172

Introduction

Welcome to the Guidewire InsuranceSuite™ 10.0 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**
 - b) Verify that the Debug console is open.
 - c) 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

Field Name	Value
------------	-------

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

b) 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?



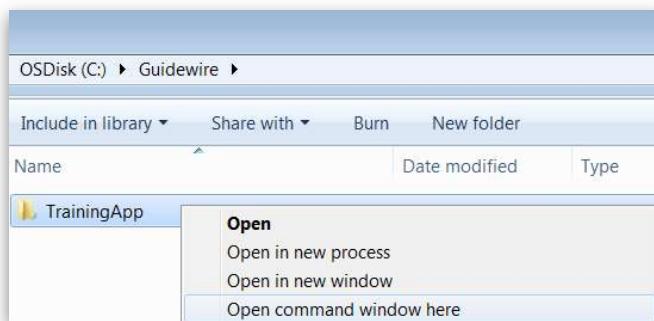
1.5 Lab Solution: Working with Guidewire Studio



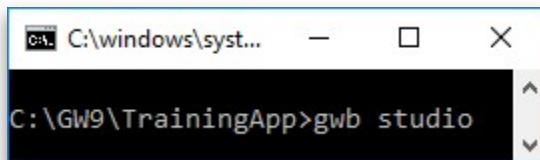
Solution

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.

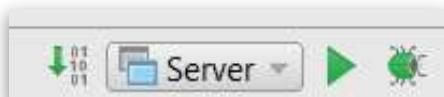


- b) Execute the following Gradle task: `gwb studio`

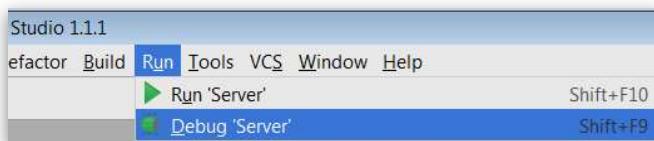


2. Start TrainingApp in debug mode

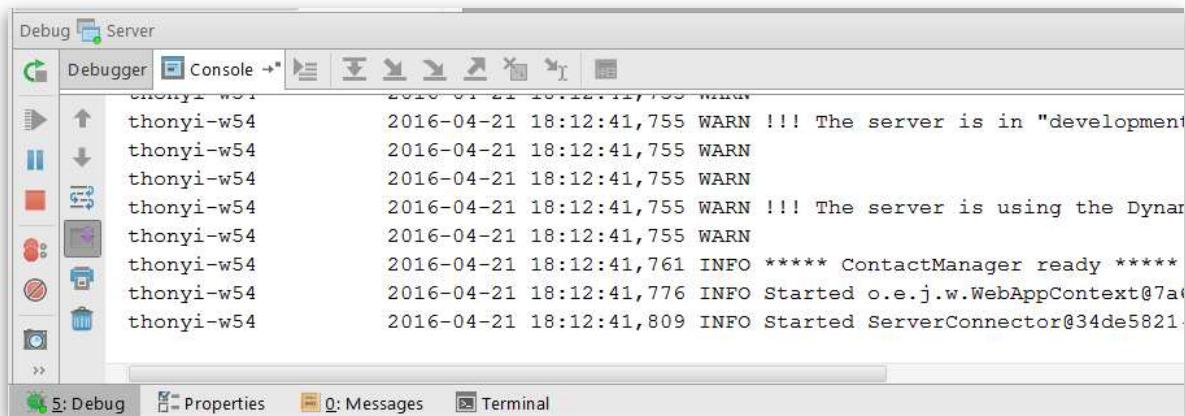
- a) To start TrainingApp in debug mode, either (1) click on the Debug icon or (2) from the menu select Run → Debug 'Server'



OR



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



The screenshot shows the Guidewire Studio interface with the 'Debug Server' window open. The 'Console' tab is selected. The log output shows several lines of server startup messages, including the line 'INFO ***** ContactManager ready *****' which indicates the application is running correctly.

Time	Message
2016-04-21 18:12:41,755	WARN !!! The server is in "development"
2016-04-21 18:12:41,755	WARN
2016-04-21 18:12:41,755	WARN
2016-04-21 18:12:41,755	WARN !!! The server is using the Dynan
2016-04-21 18:12:41,755	WARN
2016-04-21 18:12:41,761	INFO ***** ContactManager ready *****
2016-04-21 18:12:41,776	INFO Started o.e.j.w.WebAppContext@7a
2016-04-21 18:12:41,809	INFO Started ServerConnector@34de5821

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

Search

Contact Type	*	Contact	Location
Company/Last Name	Smith	Country	<none>
Minimum Score	<none>	City	
		State	<none>
		ZIP Code	#####-####

Search **Reset**

Search Results **Delete**

<input type="checkbox"/>	Name	Address	City	State	ZIP Code	Phone #	Contact Type
<input type="checkbox"/>	Quaiche Smith	435 Duarte Ave	Arcadia	California	91006	647-569-9708	Policy Person
<input type="checkbox"/>	Robert Smith	425 Duarte Ave	Arcadia	California	91006	647-560-0708	Person

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

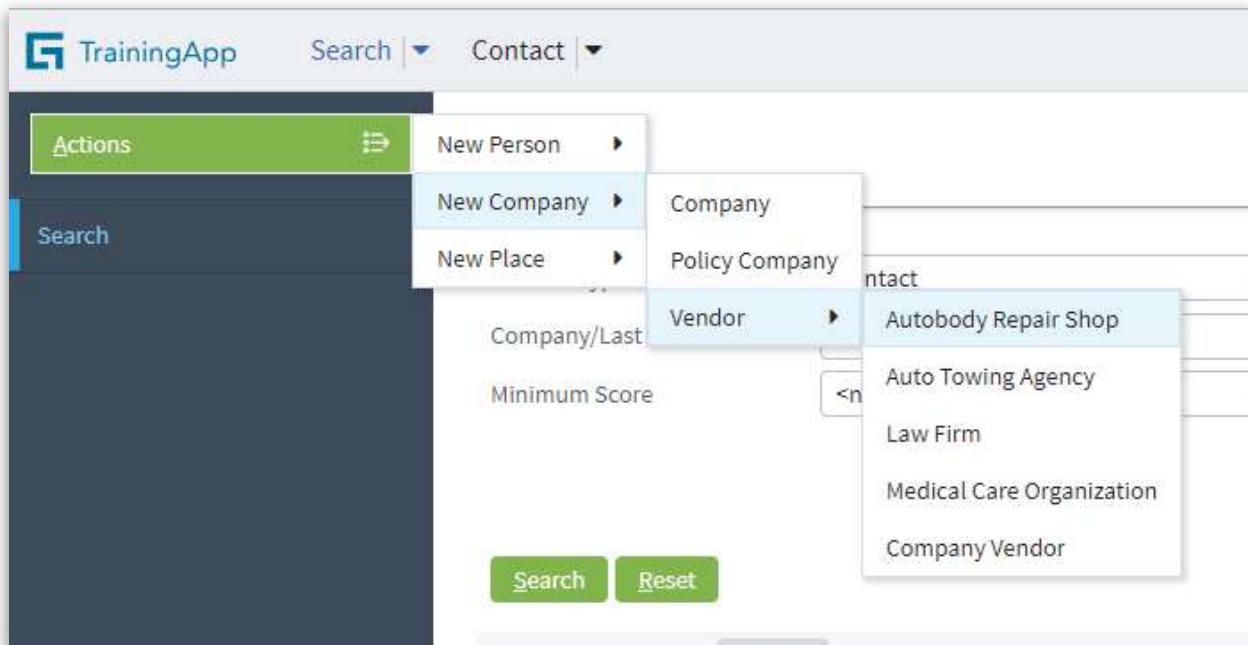
1. Create a contact of the type Law Firm

The screenshot shows the 'TrainingApp' interface with the 'Contact' module selected. In the top navigation bar, the 'Actions' button is highlighted. A dropdown menu is open under 'Actions', showing options like 'New Person', 'New Company', 'New Place', 'Company', 'Policy Company', 'Vendor', 'Company/Last', and 'Minimum Score'. The 'New Company' option is currently selected. A sub-menu for 'New Company' is displayed, listing categories such as 'Autobody Repair Shop', 'Auto Towing Agency', 'Law Firm', 'Medical Care Organization', and 'Company Vendor'. The 'Law Firm' category is highlighted. Below the sub-menu, there are 'Search' and 'Reset' buttons, and a 'Search Results' section with a 'Delete' button.

New Law Firm [Return to Search](#)

Law Firm		Additional Info	
Name *	<input type="text"/>	Preferred Vendor?	<input type="radio"/> Yes <input checked="" type="radio"/> No
Primary Address		Legal Specialty	
Country	<input type="text" value="United States"/>	<input type="text" value="<none>"/>	Contact Info
Address 1	<input type="text"/>	<input type="text"/>	Primary Contact <input type="text" value="<none selected>"/>
Address 2	<input type="text"/>	<input type="text"/>	Phone <input type="text"/>
Address 3	<input type="text"/>	<input type="text"/>	Fax <input type="text"/>
City	<input type="text"/>	<input type="text"/>	Main Email <input type="text"/>
County	<input type="text"/>	<input type="text"/>	Alternate Email <input type="text"/>
State	<input type="text" value='<none>"/'/>	<input type="text"/>	Tax Info
ZIP Code	<input type="text" value="#####-####"/>	<input type="text"/>	Tax ID (EIN) <input type="text"/>
Address Type	<input type="text" value="<none>"/>	<input type="text"/>	W-9 Received? <input type="radio"/> Yes <input checked="" type="radio"/> No
Description	<input type="text"/>		

2. Create a contact of the type Autobody Repair Shop



New Auto Repair Shop

[Return to Search](#)

[Update](#) [Cancel](#)

Auto Repair Shop

Name *

Primary Address

Country

Address 1

Address 2

Address 3

City [Map](#)

County

State

ZIP Code [Map](#)

Address Type

Additional Info

Preferred Vend

Business Licens

Contact Info

Primary Contact

Phone

Fax

Main Email

Alternate Email

Tax Info

Tax ID (EIN)

3. Search for contacts with the name Celebrity

Search

Contact Type	* Contact	Location
Company/Last Name	Celebrity	Country
Minimum Score	<none>	City
		State
		ZIP Code

Search Results

<input type="checkbox"/>	Name	Address	City	State	ZIP Code	Phone #	Contact Type
<input type="checkbox"/>	Celebrity Collision Repair		Hollywood	California			Auto Repair Shop
<input type="checkbox"/>	Celebrity Law Office		Hollywood	California			Law Firm

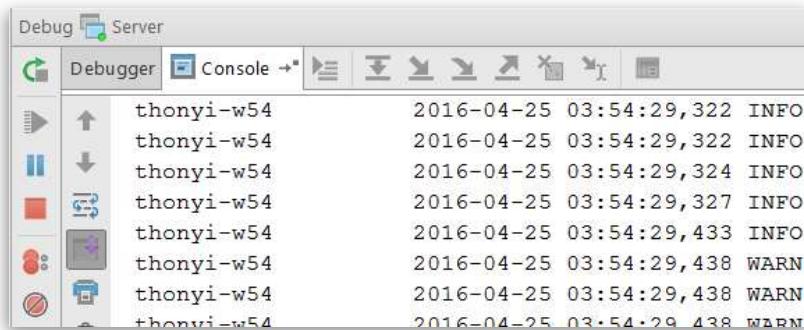
Search **Reset**

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.



genDataDictionary

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



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.

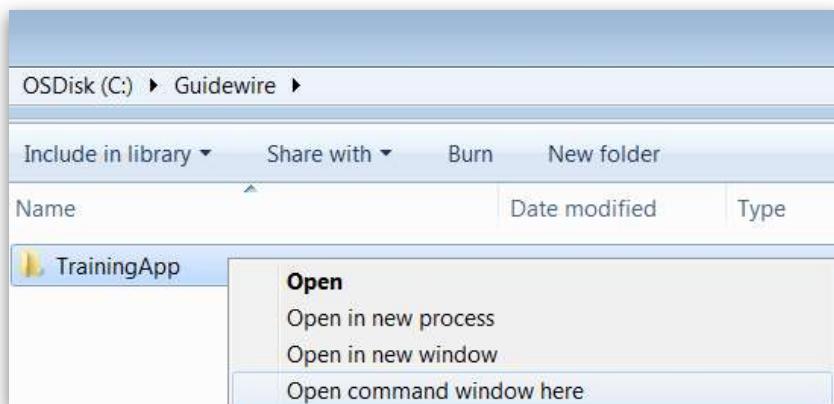


2.3 Lab Solution: Explore the Data Dictionary



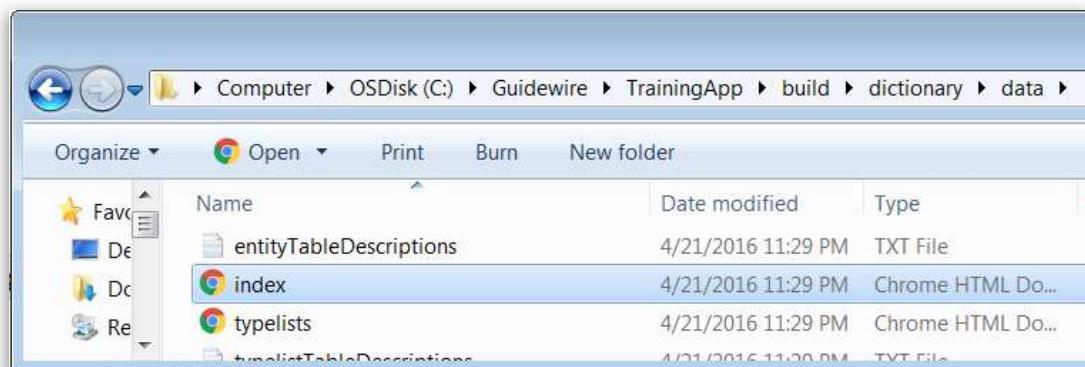
Solution

1. Build the Data Dictionary from the command window



```
Administrator: C:\windows\system32\cmd.exe
C:\Guidewire\TrainingApp>gwb genDataDictionary
```

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.

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

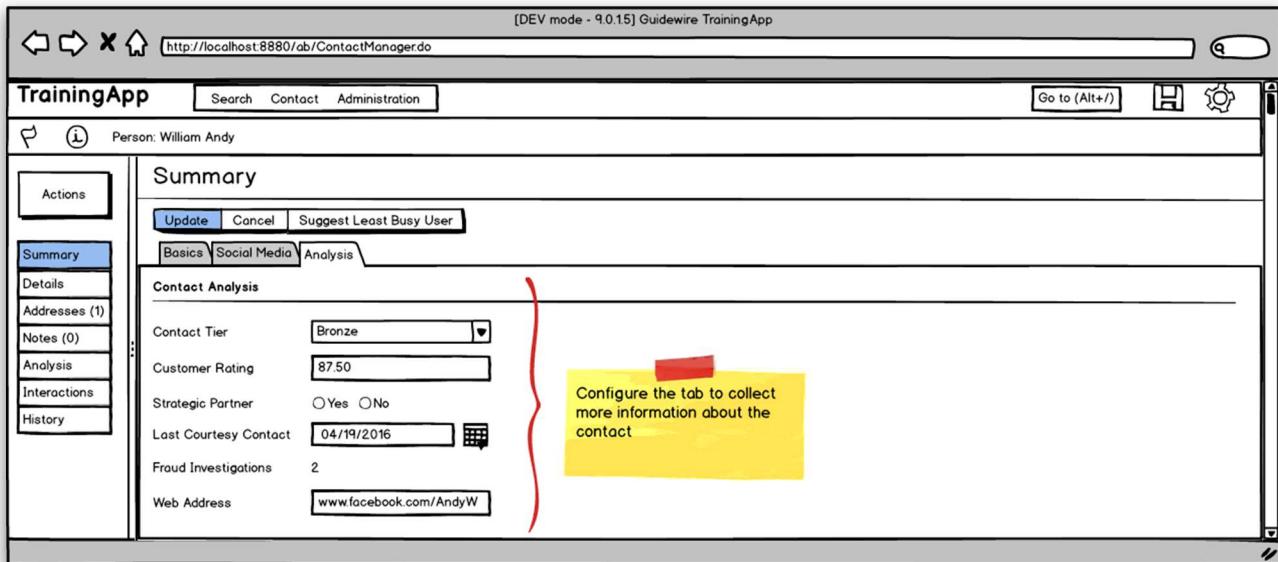
8. How many entities have a field named Organization?

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 (or start 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.etc

3. Add fields to ABContact.etc 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 (a Boolean value)	true

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

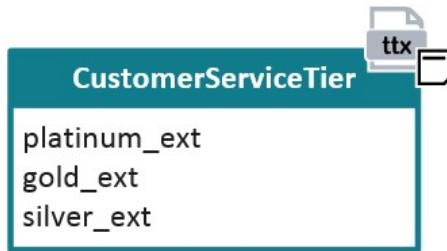
- Either click on the validate icon in the Entity Editor or use CTRL + S to save your changes
- If the code generation was successful (no errors shown in the Codegen tool window) then open the generated Java class and verify you can find your newly added fields

5. Deploy your changes

- From studio, restart the server using Debug 'Server'.
- If open, review the Messages Make window for compilation errors.
- Review the Debug console for errors.
- 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. E.g.: 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 (or start 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

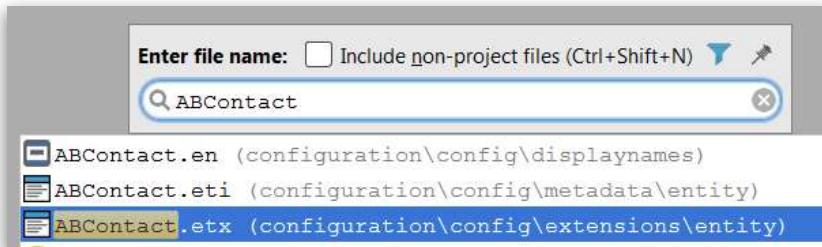


Solution

1. Open Guidewire Studio

```
C:\> Administrator: C:\Windows\system32\cmd.exe  
C:\Guidewire\TrainingApp>gwb studio
```

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



a)

3. Add the new elements (and subelements) to ABContact.etx

4. Add the new WebAddress_Ext column

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

ABContact.etx			ABContact.eti	
Element	Primary Value	Secondary Value	Name	Value
column	WebAddress_Ext	varchar	name	size
columnParam	size	40	value	40

5. Add the new FraudInvestigationNum_Ext integer column:

ABContact.etx			ABContact.eti	
Element	Primary Value	Secondary Value	Name	Value
column	WebAddress_Ext	varchar	name	FraudInvestigationNum_Ext
columnParam	size	40	type	integer
column	FraudInvestigationN...	integer	nullok	true
implementsInter	com.guidewire.pl.do...	com.guidewire.ab.d...	desc	

6. Add a new LastCourtesyContact_Ext column:

ABContact.etx			ABContact.eti	
Element	Primary Value	Secondary Value	Name	Value
column	WebAddress_Ext	varchar	name	LastCourtesyContact_Ext
column	FraudInvestigationN...	integer	type	datetime
column	LastCourtesyContact...	datetime	nullok	true
implementsInter	com.guidewire.pl.do...	com.guidewire.ab.d...	desc	Date and Time of courtesy interaction

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

ABContact.etx			ABContact.eti	
Element	Primary Value	Secondary Value	Name	Value
column	LastCourtesyContact...	datetime	name	CustomerRating_Ext
column	CustomerRating_Ext	decimal	type	decimal
implementsInter	com.guidewire.pl.do...	com.guidewire.ab.d...	nullok	true
implementsInter	com.guidewire.pl.sy...	com.guidewire.ab.d...	desc	Customer Rating

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

The screenshot shows the ABContact.etx extension in a software interface. A columnParam named 'precision' is selected, with its value set to 4.

Element	Primary Value	Secondary Value	Name	Value
column	LastCourtesyConta...	datetime	name	precision
column	CustomerRating_Ext	decimal	value	4
columnParam	precision	4		

- 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

The screenshot shows the ABContact.etx extension in a software interface. Two columnParams are defined: 'precision' with value 4 and 'scale' with value 1.

Element	Primary Value	Secondary Value	Name	Value
column	LastCourtesyConta...	datetime	name	precision
column	CustomerRating_Ext	decimal	value	4
columnParam	precision	4	name	scale
columnParam	scale	1	value	1

- Add the new IsStrategicPartner_Ext column with the following details:

The screenshot shows the ABContact.etx extension in a software interface. A new column 'IsStrategicPartner_Ext' is added with the following properties:

Element	Primary Value	Secondary Value	Name	Value
column	IsStrategicPartner_Ext	bit	name	IsStrategicPartner_Ext
implementsInterface	com.guidewire.pl.d...	com.guidewire.ab....	type	bit
implementsInterface	com.guidewire.pl.s...	com.guidewire.ab....	nullOk	true
implementsInterface	gw.ab.addressbook...	com.guidewire.ab....	desc	Is Strategic Partner?

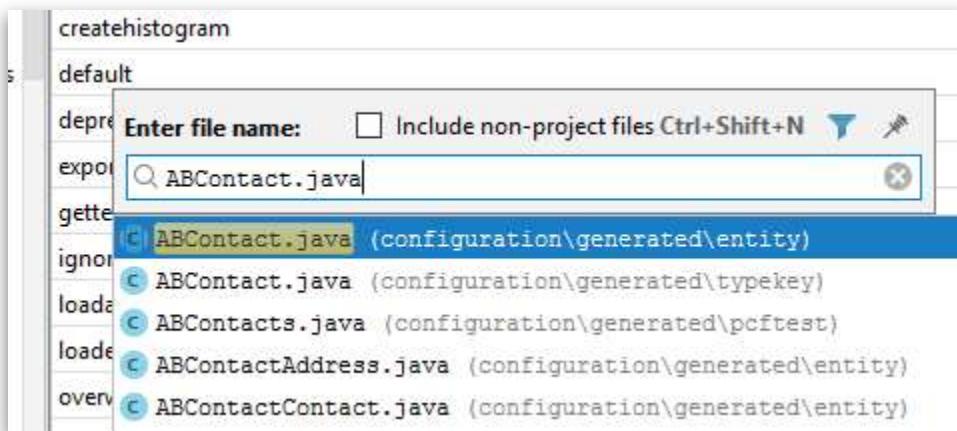
- Validate changes in Studio and generate the Java class

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

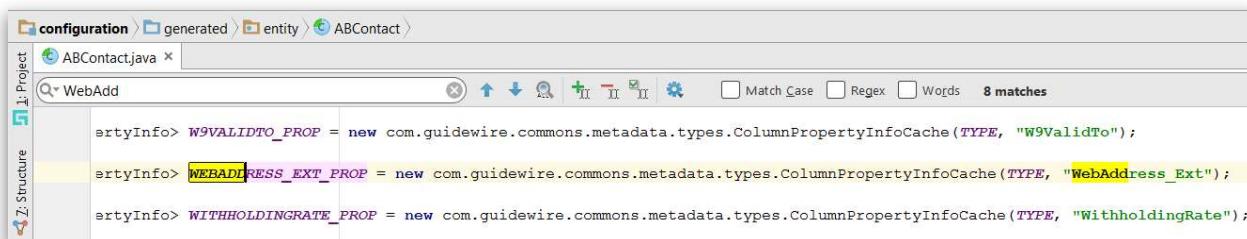


12. If the code generation was successful (no errors shown in the codegen tool window) then open the generated Java class and verify you can find your newly added fields.

- 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:

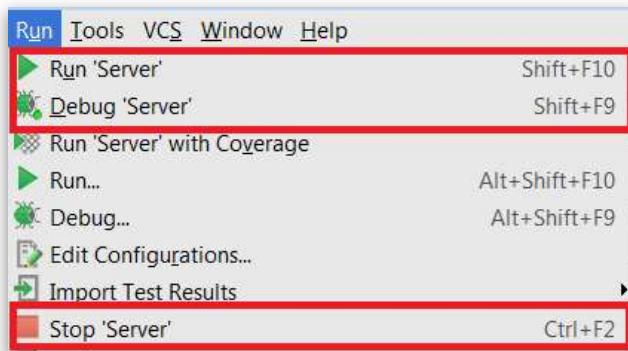


13. Build the Data Dictionary

```
Administrator: C:\windows\system32\cmd.exe
C:\Guidewire\TrainingApp>gwb genDataDictionary
```

14. Deploy changes

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

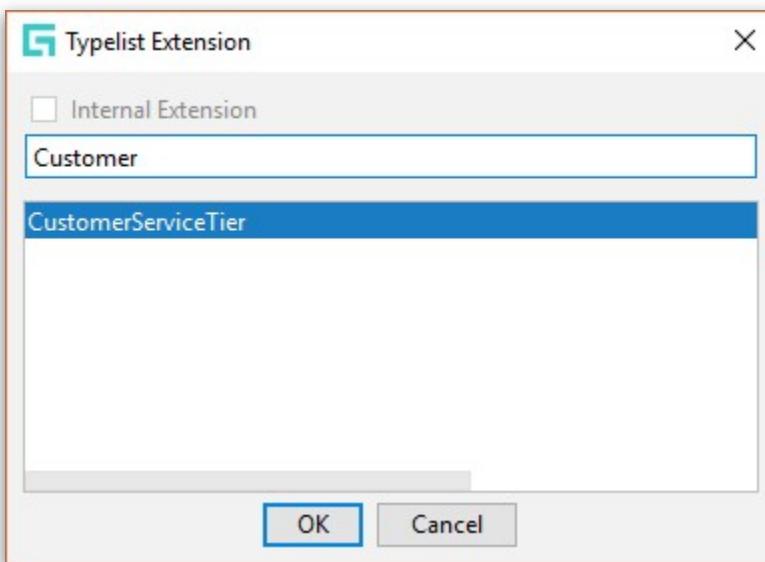


3.5 Lab Solution: Extend a base application Typelist



Solution

1. Create the Typelist Extension



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

The screenshot shows the Guidewire configuration interface. The top navigation bar shows the path: configuration > config > extensions > typelist > CustomerServiceTier.ttx. The main area displays a table of typecodes. A new row is being added, indicated by a green plus icon and the text 'typecode' in the first column. The table has columns for Element, Code, Name, Priority, and detailed properties. The 'Element' column shows 'typelistextension' and 'CustomerServ...'. The 'Code' column shows 'platinum_ext'. The 'Name' column shows 'Platinum'. The 'Priority' column shows '10'. The detailed properties table shows the following values:

Name	
code	platinum_ext
name	Platinum
desc	Platinum
identifierCode	
priority	10
retired	false

The screenshot shows the Guidewire configuration interface. The top navigation bar shows the path: configuration > config > extensions > typelist > CustomerServiceTier.ttx. The main area displays a table of typecodes. A new row is being added, indicated by a green plus icon and the text 'typecode' in the first column. The table has columns for Element, Code, Name, Priority, and detailed properties. The 'Element' column shows 'typelistextension' and 'CustomerServ...'. The 'Code' column shows 'gold_ext'. The 'Name' column shows 'Gold'. The 'Priority' column shows '20'. The detailed properties table shows the following values:

Name	
code	gold_ext
name	Gold
desc	Gold
identifierCode	
priority	20
retired	false

The screenshot shows the Guidewire configuration interface for the 'CustomerServiceTier.ttx' file. The left sidebar has two sections: '1: Project' and '2: Structure'. The '2: Structure' section is currently active, displaying a table of typecodes.

The table has the following columns:

- Element
- Code
- Name
- Priority
- Name

The rows show the following data:

typecode	platinum_ext	Platinum	10	code silver_ext
typecode	gold_ext	Gold	20	name Silver
typecode	silver_ext	Silver	30	desc Silver

A detailed view of the selected 'silver_ext' typecode is shown on the right side of the interface. It includes the following properties:

code	silver_ext
name	Silver
desc	Silver
identifierCode	
priority	30
retired	false

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.

The screenshot shows the TrainingApp interface. At the top, there is a navigation bar with the logo 'TrainingApp', a search bar, and a contact dropdown. On the left, a sidebar menu lists 'Actions', 'Summary' (which is selected and highlighted in blue), 'Details', 'Addresses (3)', 'Notes (5)', 'Social Media', 'Analysis', 'Interactions', and 'History'. The main content area has a title 'Person: William Andy' and a section titled 'Summary'. Under 'Basic Information', it shows 'Name: William Andy' and 'Public ID: ab:5'. Below that, under 'Assigned User', there is a dropdown menu set to 'Alice Applegate'. The 'Email Address' field contains 'willandy@albertsons.com'. Under 'Created On', it shows the date '06/19/2018'. In the 'Primary Address' section, the 'Country' dropdown is set to 'United States' and the 'Address 1' field contains '345 Fir Lane'.

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 10.0, Guidewire Studio 10.0, 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 anABCContact 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.

The screenshot shows the 'Summary' page for a contact named William Andy. The 'Basics' tab is selected. The 'Basic Information' section contains fields for Name (William Andy) and Public ID (ab:5). The 'Flag Entries' section displays a single entry: 'View' (button), 'Date Flagged' (06/19/2018), 'Reason' (No email address for this contact.), and 'Date Unflagged' (button).

4.3.1 Configuration

1. Modify the order of the widgets in Basic Information

- Move the Created On widget below the Assigned User widget.
- 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

- In TrainingApp, reload the changes to the PCF file(s).
- 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.



4.4 Lab Solution: Write it down



Solution

Exact details on how to complete the exercise.

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

Location Info for ABContactSummaryPage/Basics

Smoke Test Step: ABContactSummaryPage/Basics

Current User: Alice Applegate (aapplegate)

Application: ABContactSummaryPage/Basics

Schema Version:

---- File Structure ----

Page "ABContactSummaryPage" ([ABContactSummaryPage.pcf:9](#))
Screen "ABContactSummaryScreen" (ABContactSummaryPage.pcf)
CardView "ABContactSummaryCV" ([ABContactSummaryCV.pcf:6](#))
DetailView "ABContactSummaryDV" ([ABContactSummaryDV.pcf:7](#))
InputSet "AddressOwnerInputSet" ([AddressOwnerInputSet.pcf:6](#))
InputSet "GlobalAddressInputSet" ([GlobalAddressInputSet.default.pcf:7](#))
ListView "FlagEntriesLV" ([FlagEntriesLV.pcf:6](#))

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.

Page Include Structure (Location Stack):

- [ABContactLG.pcf](#)
 - var anABContact : ABContact = "William Andy"
- [ABContactInfoBar.pcf](#)
- [ABContactLGMenuActions.pcf](#)
- [NewContactPickerMenuItemSet.pcf](#)
- [ABContactSummaryPage.pcf](#)
 - var anABContact : ABContact = "William Andy"
- [ABContactSummaryCV.pcf](#)
- [ABContactSummaryDV.pcf](#)
- [AddressOwnerInputSet.pcf](#)
- [FlagEntriesLV.pcf](#)

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?

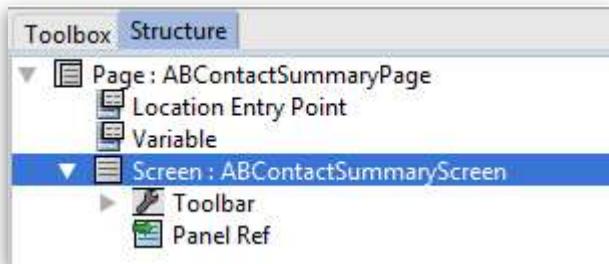
Collapse	Widget	ID	Render ID	File	Label
Page		ABContactSummaryPage		ABContactSummaryPage.pcf.9	
Screen		ABContactSummaryScreen	ABContactSummaryPage-ABContactSummaryScreen	ABContactSummaryPage.pcf.16	
TitleBar		ttlBar	ABContactSummaryPage-ABContactSummaryScreen-ttlBar	ABContactSummaryPage.pcf.16	
Messages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

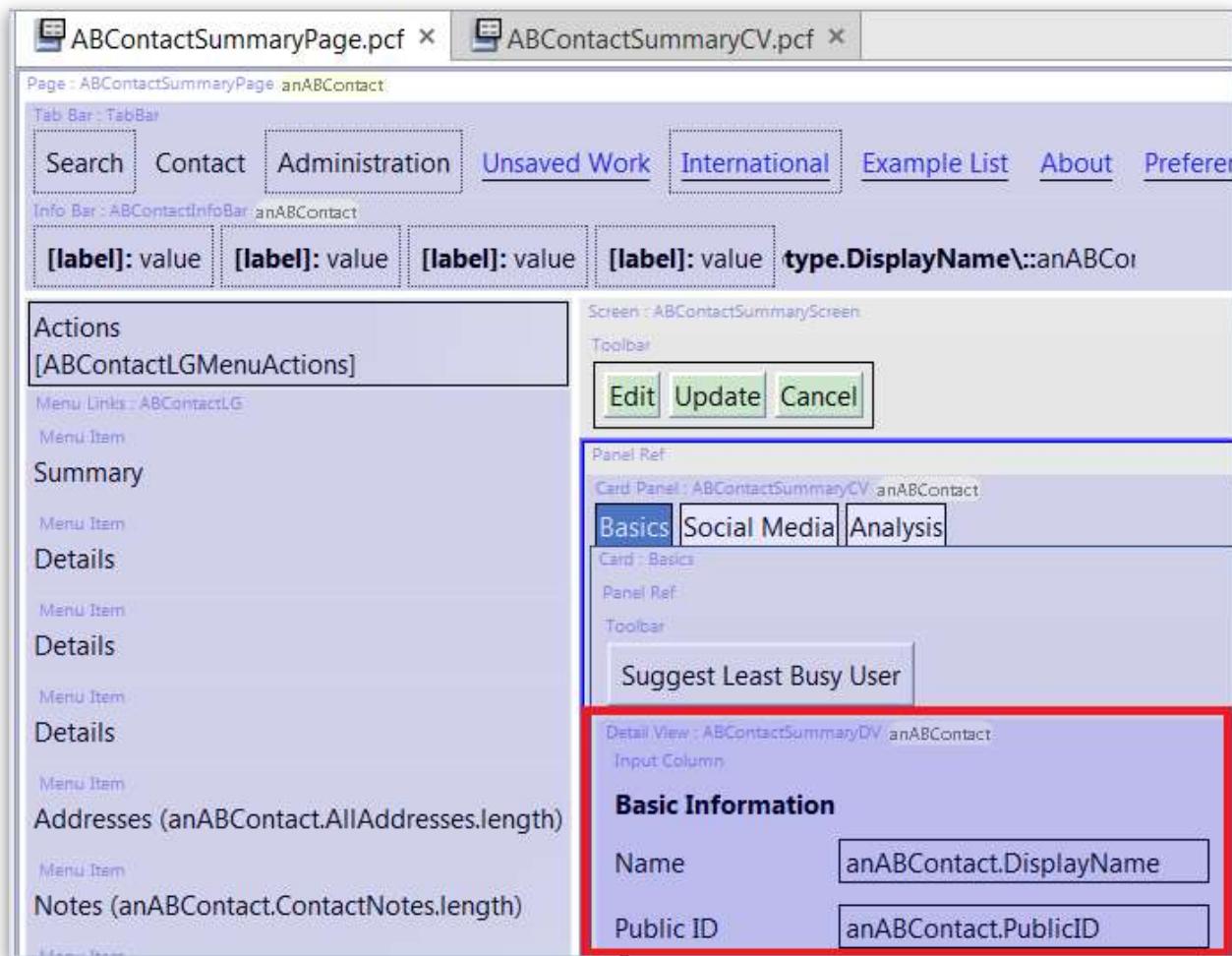
Action: Use ALT + SHIFT + E to open the page in Guidewire Studio.

- 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



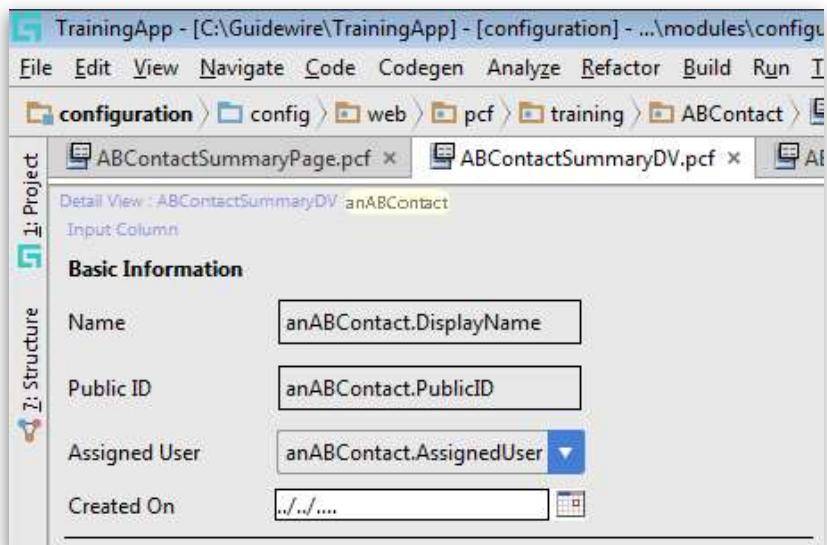
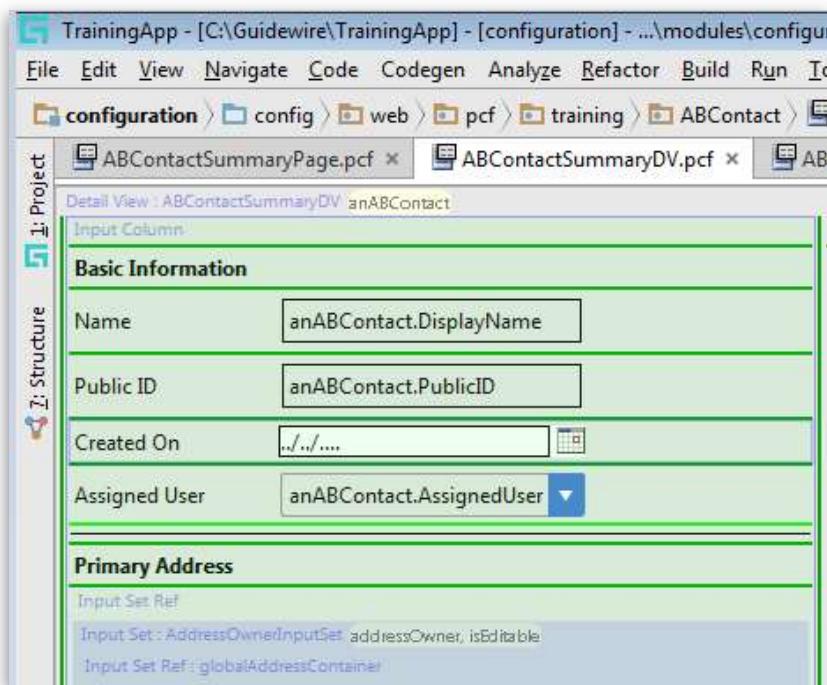
8. Question: Review the ABContactSummaryDV.pcf in the canvas. How are you able to distinguish included sections from the widgets specified in the file itself?

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.

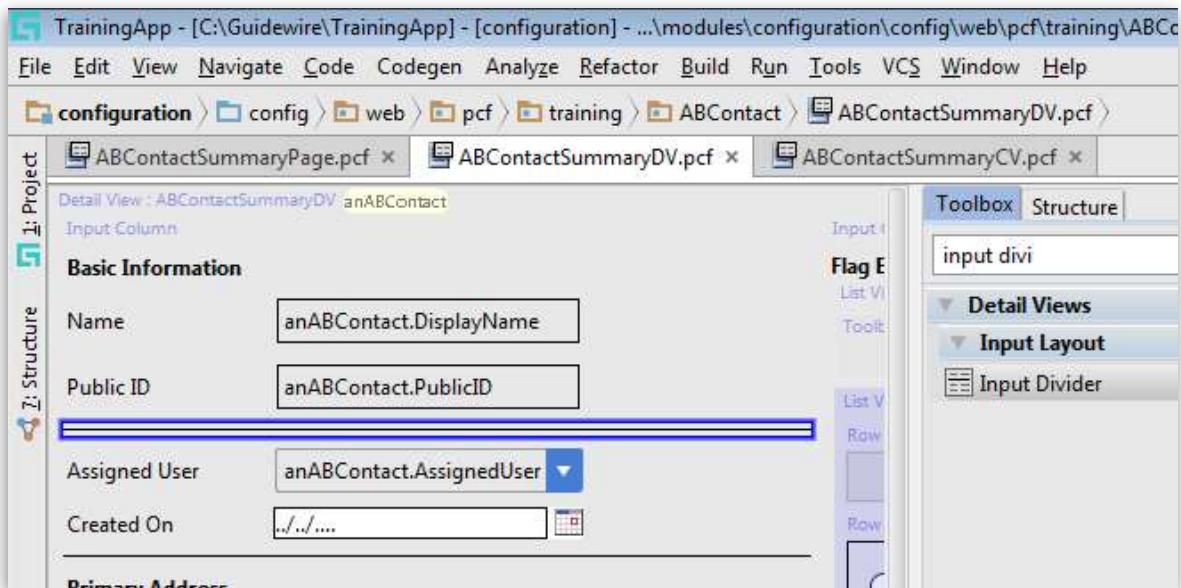
4.5 Lab Solution: Modify the Basics card on the Summary page

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.



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



- d) Reload the PCF changes using ALT + SHIFT + L in the browser.

The screenshot shows the 'Summary' PCF page. At the top, there is a navigation bar with tabs: Basics (selected), Social Media, and Analysis. Below the tabs is a button labeled 'Suggest Least Busy User'. The main content area is titled 'Basic Information' and contains the following data:

Name	William Andy
Public ID	ab:5
Assigned User	
Created On	06/19/2018

At the bottom of the page is a section titled 'Primary Address'.

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.

The screenshot shows the TrainingApp interface. At the top, there is a header with the logo 'TrainingApp', a search bar, and a 'Contact' dropdown. On the left, a sidebar menu lists 'Actions', 'Summary', 'Details', 'Addresses (3)', 'Notes (5)', 'Social Media', 'Analysis', 'Interactions', and 'History'. The 'Summary' tab is currently active. In the main content area, the title 'Person: William Andy' is displayed above a 'Summary' section. Below the summary, there is a tab bar with 'Basics', 'Social Media', and 'Analysis', where 'Analysis' is selected. The 'Contact Analysis' section contains several fields: 'ContactTier' (dropdown menu showing '<none>'), 'CustomerRating' (text input field), 'IsStrategicPartner' (radio buttons for 'Yes' and 'No' with 'Yes' selected), 'LastCourtesyContact' (date input field showing '09/25/2018'), 'Fraud Investigations' (text input field), and 'WebAddress' (text input field).

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 10.0, Guidewire Studio 10.0, 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 have 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

- 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.



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	<i>OtherEntityType</i>
typekey	TypeKey Input TypeKey Radio Button Input	Typekey Cell	typekey. <i>TypelistName</i>

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				
6	Last Courtesy Contact	Yes				

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



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

- c) 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

- In the PCF Editor, open the ABContactSummaryCV.pcf file.

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

- Use the Toolbox to add widgets.
- Use the Properties window to define the necessary widget properties when such as id*, editable, and value*.
- When defining labels, create display keys.
- 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. Generally speaking 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

Just 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.

Person: William Andy

Summary

Basics Social Media **Analysis**

ContactAnalysis

ContactTier <none>

CustomerRating

IsStrategicPartner Yes No

LastCourtesyContact MM/dd/yyyy

Fraud Investigations

WebAddress

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?**





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.

```
anABContactBuildings_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

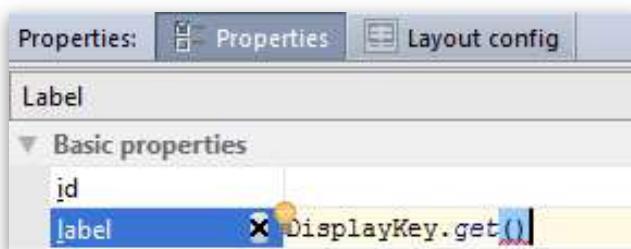
	Label/Info	Editable	Field	Datatype (Data Model)	Widget	valueType (Gosu)
7	Fraud Investigation Number	No	FraudInvestigationNum_Ext	integer	Text Input	java.lang.Integer
8	Web Address	Yes	WebAddress_Ext	varchar(60)	Text Input	java.lang.String

5.8 Lab Solution: Add new atomic widgets

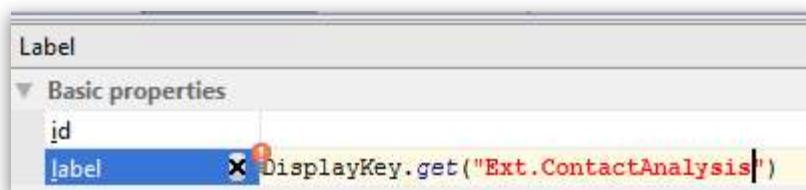


Solution

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



- 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:

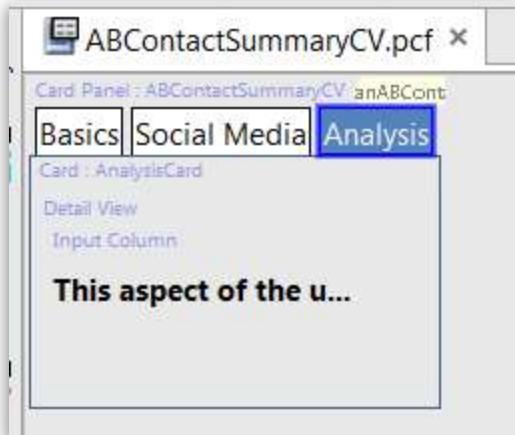


4. Then enter the text value for the locale and click OK



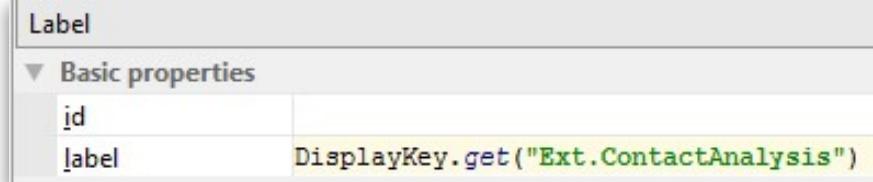
5. Open the PCF file:

- Open ABContactSummaryCV.pcf and select the Analysis card.

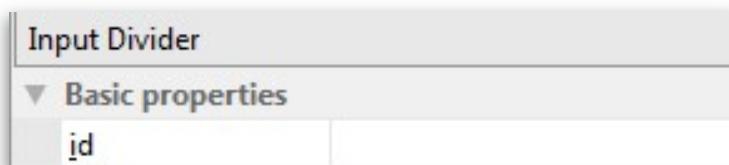


- b) To delete the existing label, right click on it and select Delete from the context menu.
6. Drag and drop widgets inside the existing Input Column

- a) Label widget for Contact Analysis label:



- b) Input Divider widget to display a horizontal line:



- c) TypeKey Input widget for Contact Tier

TypeKey Input	
▼ Basic properties	
editable	true
id*	ContactTier
label	DisplayKey.get("Ext.ContactTier")
required	
value*	anABContact.ContactTier
valueType*	typekey.ContactTier

7. Text Input widget for Customer Rating:

Properties:	Properties	Layout config	Reflection	PostOnChange
Text Input				
▼ Basic properties				
editable	true			
id*	CustomerRating			
label	DisplayKey.get("Ext.CustomerRating")			
required				
value*	anABContact.CustomerRating_Ext			

...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	
<u>editable</u>	true
<u>falseLabel</u>	
<u>id*</u>	IsStrategicPartner
<u>label</u>	DisplayKey.get("Ext.IsStrategicPartner")
<u>required</u>	
<u>trueLabel</u>	
<u>value*</u>	anABContact.IsStrategicPartner_Ext

9. Date Input widget for Last Courtesy Contact:

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

10. Text Input widget for Number Of Fraud Investigations...

Text Input	
Basic properties	
<u>editable</u>	false
<u>id*</u>	NumberOfFraudInvestigations
<u>label</u>	DisplayKey.get("Ext.NumOfFraudInv")
<u>required</u>	
<u>value*</u>	anABContact.FraudInvestigationNum_Ext

...and set the valueType to java.lang.Integer

validationExpression	
valueType	java.lang.Integer
visible	

11. Text Input widget for Web Address:

The screenshot shows the 'Properties' panel for a 'Text Input' widget. The 'Properties' tab is selected. Below it, the 'Basic properties' section is expanded, showing the following configuration:

editable	true
id*	WebAddress
label	DisplayKey.get("Ext.WebAddress")
required	
value*	anABContact.WebAddress_Ext

The completed design:

ABContactSummaryCV.pcf

Card Panel : ABContactSummaryCV anABContact

Basics Social Media Analysis

Contact Analysis

Contact Tier: anABContact.ContactTier

Customer Rating: anABContact.CustomerRati...

Strategic Partner: Yes No

Last Courtesy Contact: /.../....

Fraud Investigations: anABContact.FraudInvestig...

Web Address: anABContact.WebAddress_...

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.

- 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

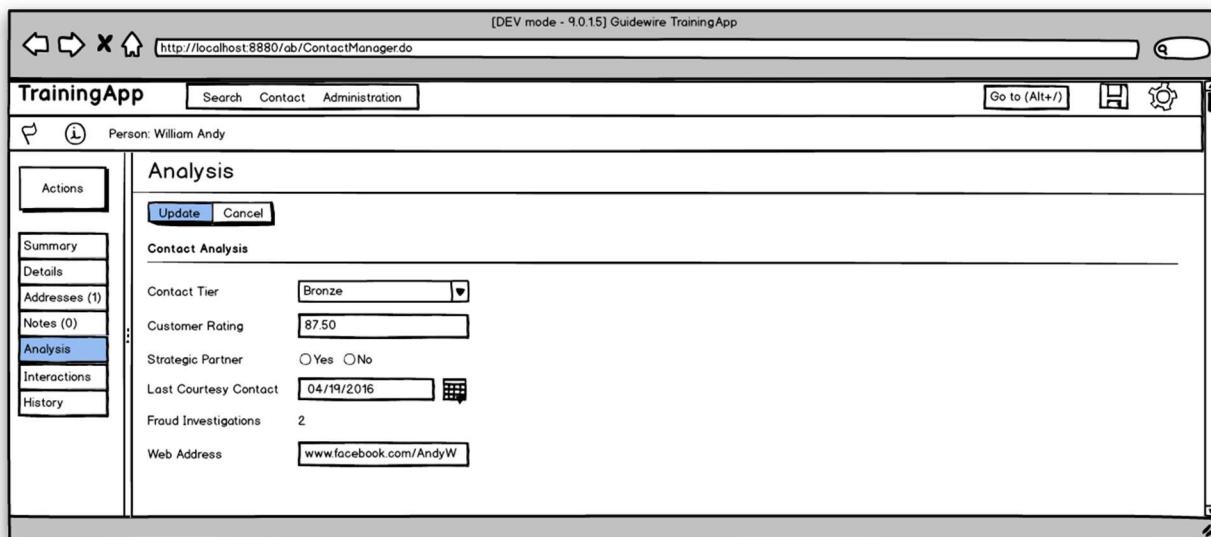
Lesson 6

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.

The screenshot shows the Guidewire TrainingApp interface. At the top, there's a header with the logo 'TrainingApp', a search bar, and a contact dropdown. Below the header is a toolbar with icons for edit, settings, and go to. The main area has a sidebar on the left with tabs for Actions, Summary, Details, Addresses (3), Notes (5), Social Media, Analysis, Interactions, and History. The 'Analysis' tab is currently selected. The main content area displays a summary for a contact named 'Person: William Andy'. It includes tabs for Basics, Social Media, and Analysis. Under the Analysis tab, there are fields for ContactTier (set to <none>), CustomerRating (empty input), IsStrategicPartner (radio buttons for Yes and No), LastCourtesyContact (date input with a calendar icon), Fraud Investigations (empty input), and WebAddress (empty input). At the bottom right of the main area are 'Update' and 'Cancel' buttons.

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 10.0, Guidewire Studio 10.0, 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.



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

- 4. Log in to TrainingApp**
 - a) Log in as Alice Applegate.
- 5. Reload the PCF changes**
 - a) In TrainingApp, reload the changes to the PCF file(s).
- 6. 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.

The screenshot shows the Guidewire TrainingApp interface. At the top, there is a navigation bar with the 'TrainingApp' logo, a search bar, and a contact dropdown. On the left, a sidebar menu lists various contact details: Actions, Summary, Details, Addresses (3), Notes (5), Social Media, Analysis, Interactions, and History. The 'Analysis' tab is currently selected. The main content area is titled 'Summary' and shows a sub-section titled 'Contact Analysis'. This section contains several input fields and controls: 'ContactTier' with a dropdown menu set to '<none>', 'CustomerRating' with a text input field, 'IsStrategicPartner' with a radio button group where 'Yes' is selected, 'LastCourtesyContact' with a date input field set to '09/25/2018', 'Fraud Investigations' with a text input field, and 'WebAddress' with an empty text input field.

7. 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.

The screenshot shows a web-based application interface. At the top, there is a header with the logo 'TrainingApp' and navigation links for 'Search' and 'Contact'. On the left side, there is a vertical sidebar with a green header bar containing the word 'Actions'. Below this, there are several menu items: 'Summary', 'Details', 'Addresses (3)', 'Notes (5)', 'Social Media', 'Analysis' (which is currently selected and highlighted in blue), 'Interactions', and 'History'. To the right of the sidebar, the main content area has a title 'Person: William Andy' at the top. Below it, the word 'Analysis' is displayed in large letters. Under the 'Analysis' section, there is a list of fields: 'Contact Analysis', 'Contact Tier', 'Customer Rating', 'Is Strategic Partner?', 'Last Courtesy Contact', and 'Web Address'. The 'Analysis' menu item in the sidebar is also highlighted with a blue background.



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?



6.5 Lab Solution: Reusable Detail View Panel



Solution

1. Open Guidewire Studio

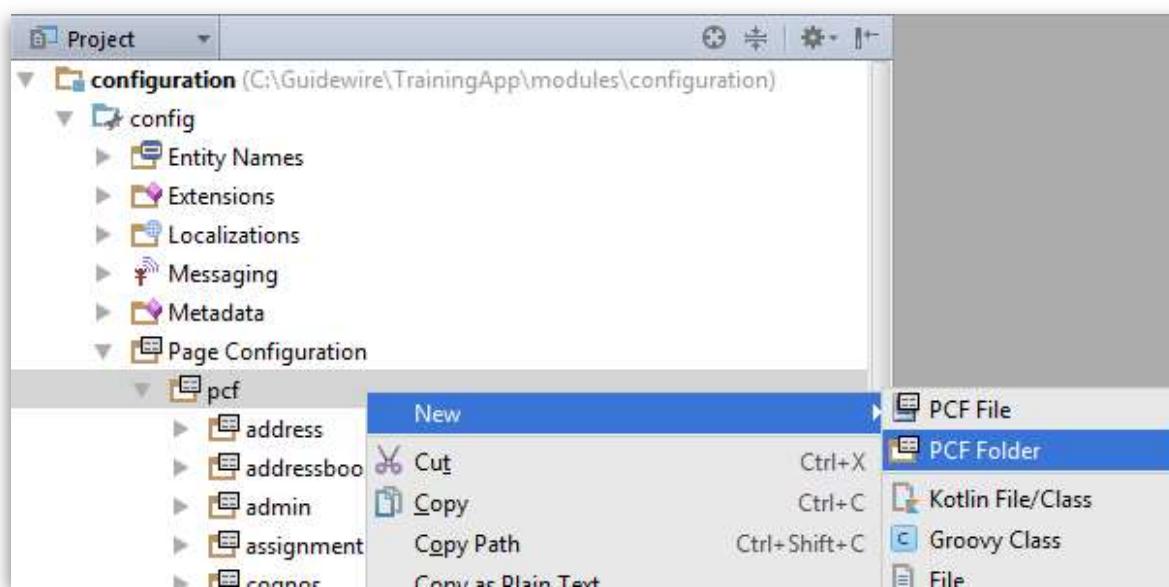
- See the Introduction to Guidewire Configuration lesson, if needed

2. Run or Debug the TrainingApp project]

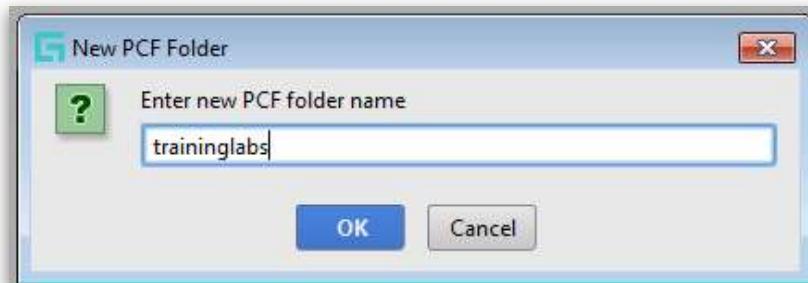
- See the Introduction to Guidewire Configuration lesson, if needed.

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

- Right click on the pcf node and select New → PCF Folder from the context menu.

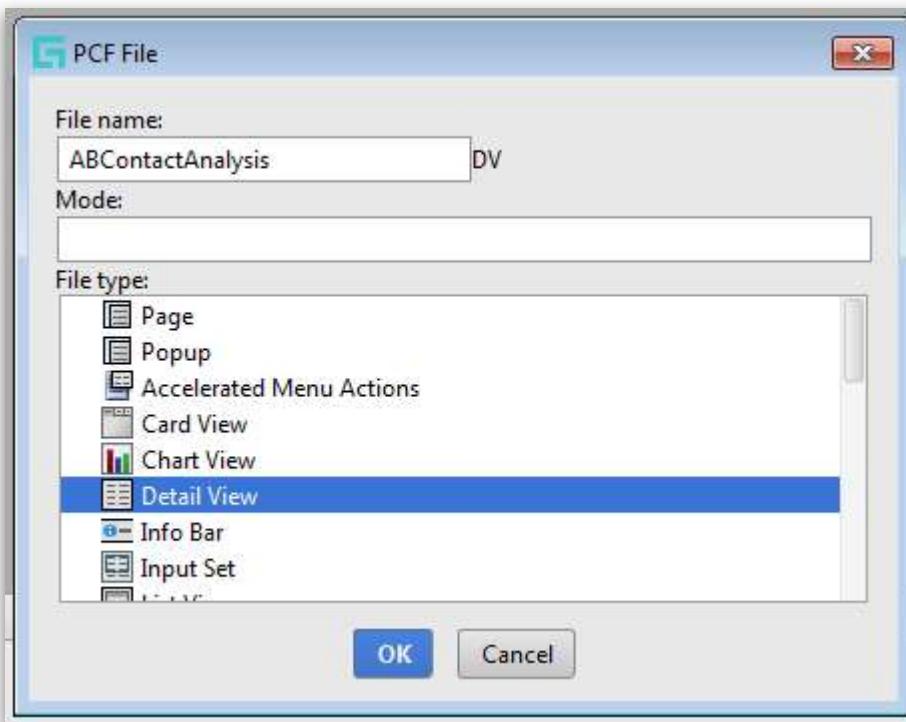


4. Enter the traininglabs name in the popup – no spaces

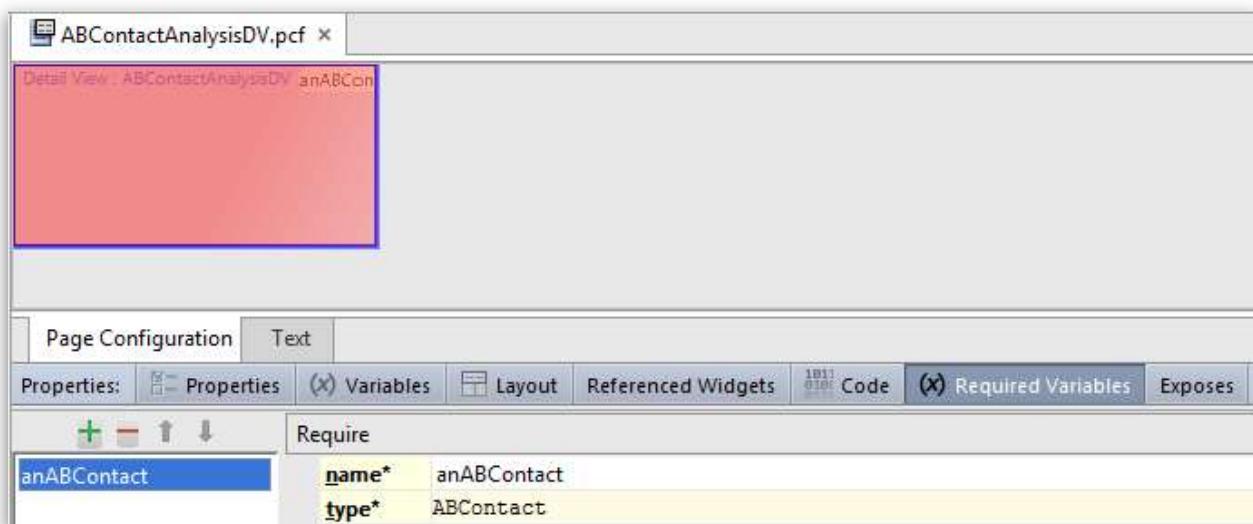


5. Create the ABContactAnalysisDV PCF file

- 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.

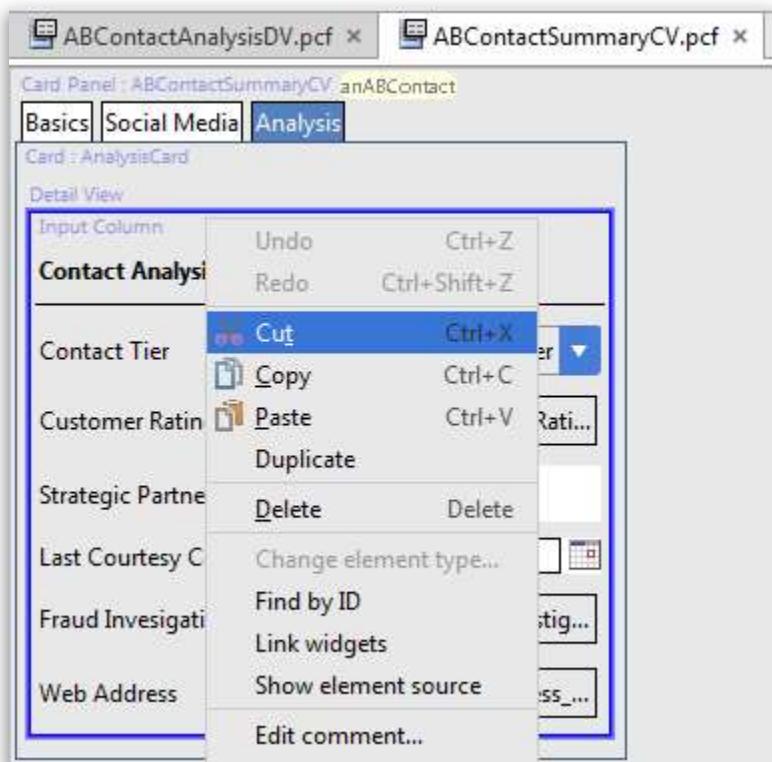


- b) Specify a root object of the type ABContact.



6. Navigate to the Analysis tab in ABContactSummaryCV.pcf

- 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.

The screenshot shows the 'Detail View : ABContactAnalysisDV' panel. At the top, there are two tabs: 'ABContactAnalysisDV.pcf' and 'ABContactSummaryCV.pcf'. Below the tabs, the title 'Detail View : ABContactAnalysisDV anABCContact' is displayed. A section titled 'Input Column' contains several fields:

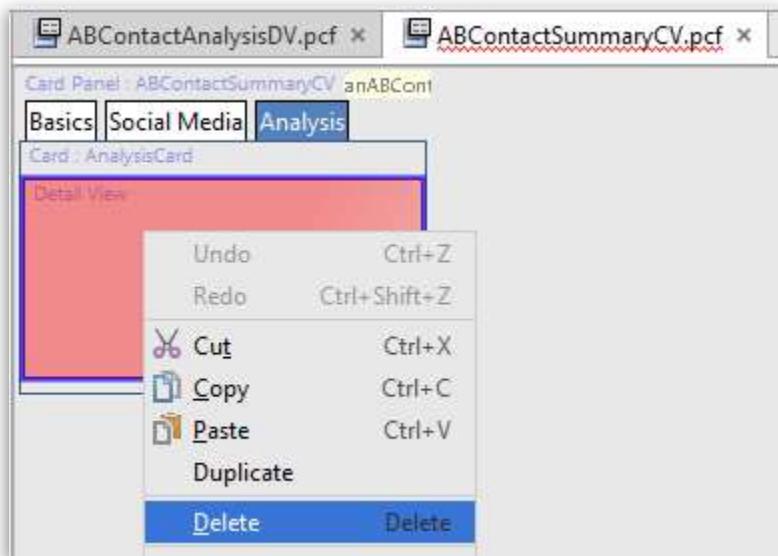
- Contact Tier: A dropdown menu currently set to 'anABCContact.ContactTier'.
- Customer Rating: A dropdown menu currently set to 'anABCContact.CustomerRati...'.
- Strategic Partner: A radio button group with 'Yes' selected.
- Last Courtesy Contact: An input field with a date picker icon, currently empty.
- Fraud Investigations: A dropdown menu currently set to 'anABCContact.FraudInvestig...'.
- Web Address: An input field currently empty.

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



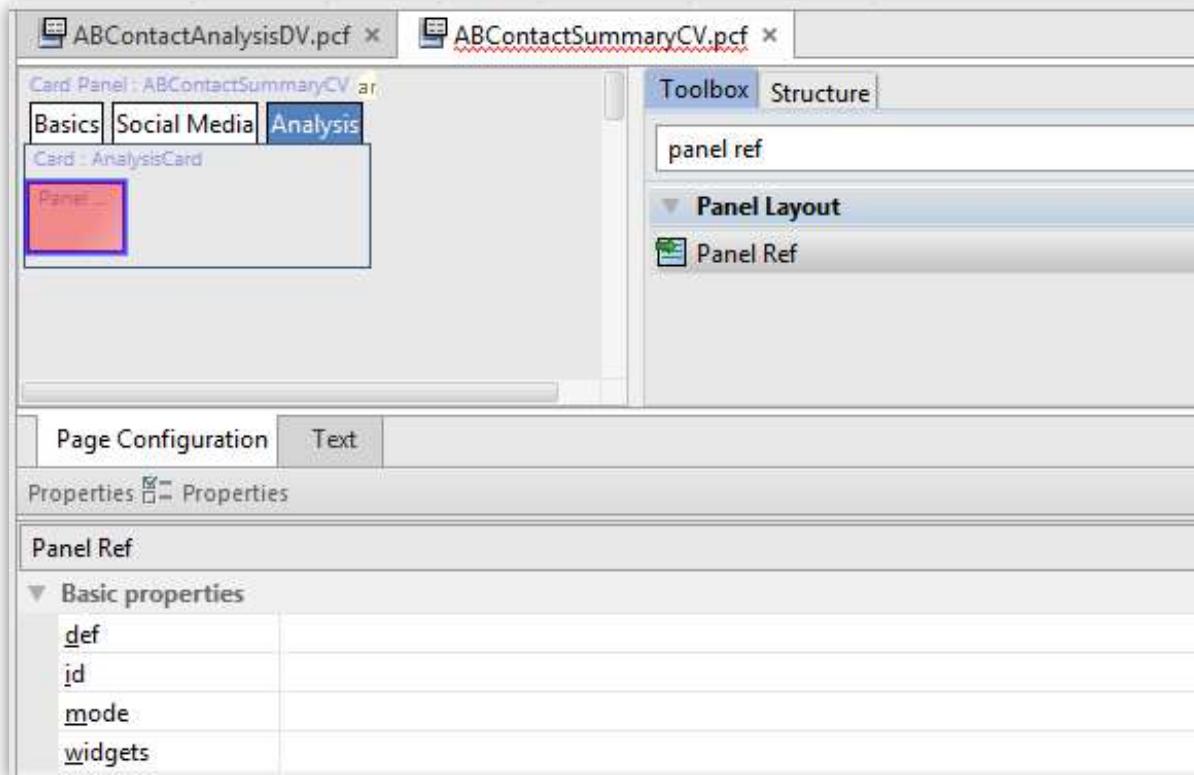
Solution

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

- Reference the ABContactAnalysisDV in the Panel Ref.

The screenshot shows the Guidewire Studio interface with the following details:

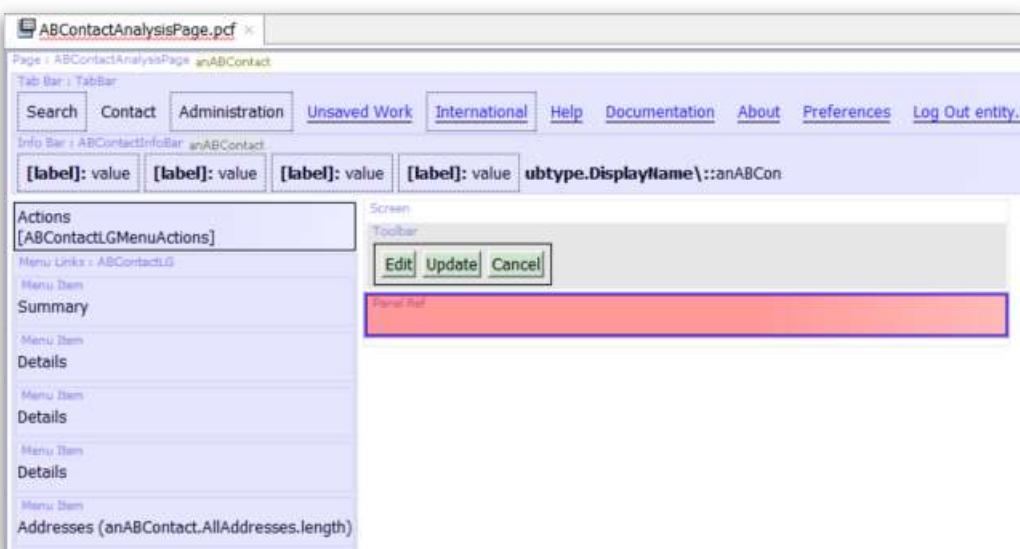
- Card Panel:** ABContactAnalysisDV.pcf
- Card:** Card Panel : ABContactSummaryCV anABContact
- Tab:** Analysis (selected)
- Panel Ref:** Detail View : ABContactAnalysisDV anABContact
- Input Column:** Contact Analysis
- Form Fields (Visible):**
 - Contact Tier: anABContact.ContactTier
 - Customer Rating: anABContact.CustomerRati...
 - Strategic Partner: Yes No
 - Last Courtesy Contact: anABContact.LastCous...
 - Fraud Investigations: anABContact.FraudInvestig...
 - Web Address: anABContact.WebAddress_...
- Panel Ref Properties:**
 - Basic properties:**
 - def: ABContactAnalysisDV(anABContact) (highlighted)
 - id
 - mode
 - widgets

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

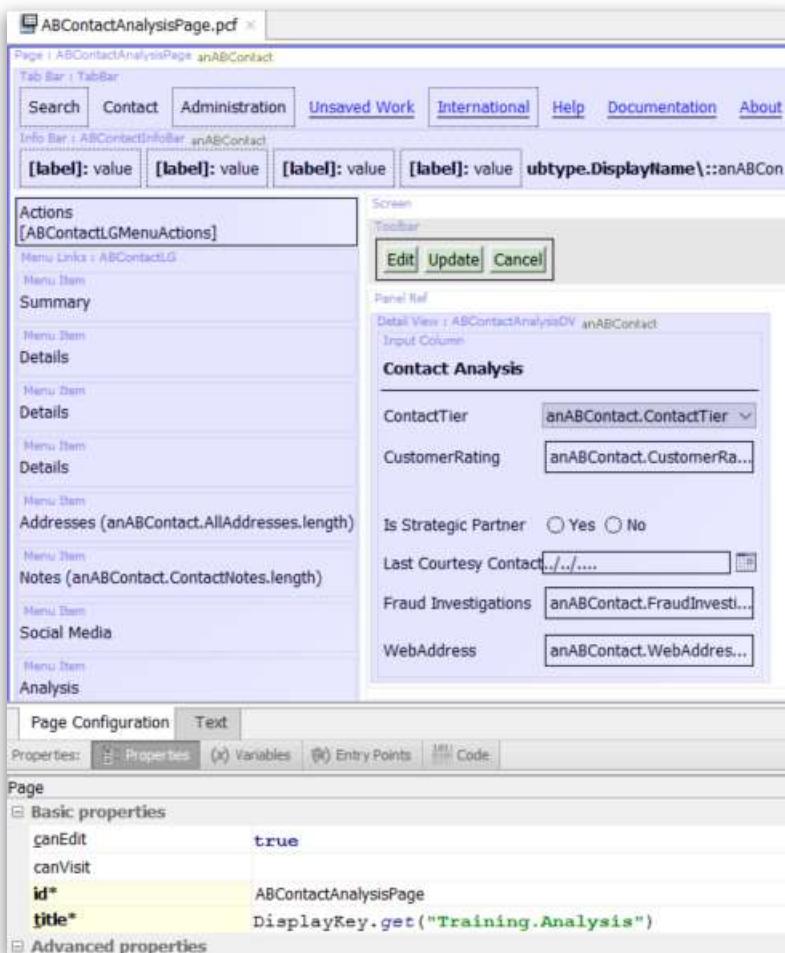


Solution

- 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

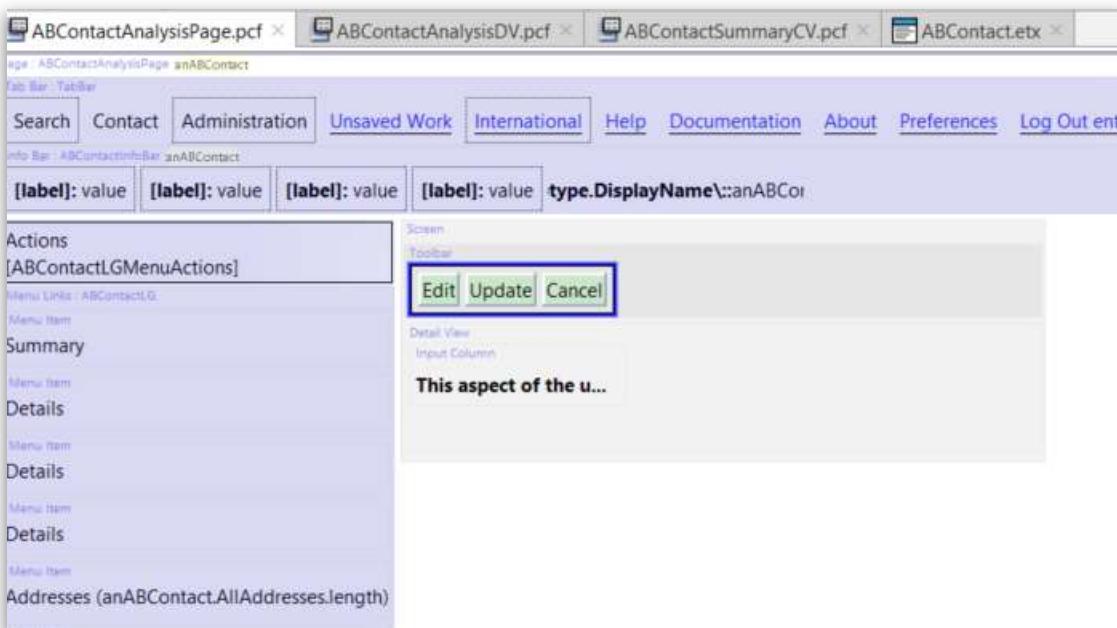


Solution

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.



Lesson 7

Introduction to Locations

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

The screenshot shows a web application interface for 'TrainingApp'. At the top, there is a navigation bar with the logo, a search bar, and a contact dropdown. The main content area has a sidebar on the left with links like 'Actions', 'Summary', 'Details' (which is currently selected), 'Addresses (3)', 'Notes (5)', 'Social Media', 'Analysis', 'Interactions', and 'History'. The main panel displays 'Person: William Andy' with a 'Details' section. Under 'Details', there are tabs for 'Person Info' (which is selected), 'Phone & Addresses', and 'Bank Accounts'. The 'Person Info' tab shows fields for Name (Full Name: William Andy, Prefix: , First Name: William, Middle Name: , Last Name: Andy, Suffix:), Employment Info (Occupation: , Employer: Albertson's), License Info (Driver's License: , State:), and Contact Insights.

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

The screenshot shows a web-based application interface for 'TrainingApp'. At the top, there is a navigation bar with the 'TrainingApp' logo, a search bar, and a contact dropdown. On the left, a sidebar menu lists various sections: Actions, Summary, Details (which is currently selected), Addresses (1), Notes (0), Social Media, Analysis, Interactions, and History. The main content area is titled 'Details' and shows information for 'Company: Albertson's'. A tabbed section at the top of this area includes 'Company Info' (which is active), 'Phone & Addresses', and 'Bank Accounts'. Below these tabs, the 'Company Info' section contains fields for Name (Albertson's), Primary Contact (William Andy), Address (345 Fir Lane, La Canada, CA 91352), Email Address, and Additional Info (Tax ID: *****). There is also a field for Inspection Required? and Preferred Currency. To the right of the main content area, there is a sidebar titled 'Employee Info' which lists names: James Anders, Samantha Andy, Eric Andy, and William Andy. Below this is a section titled 'Contact Insight'.

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 10.0, Guidewire Studio 10.0, 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?



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

InsuranceSuite 10 Fundamentals: Kickstart - Student Workbook

The screenshot shows a user interface for managing a person record. At the top, there's a header with the 'TrainingApp' logo, a search bar, and a contact dropdown. On the left is a sidebar with tabs: 'Actions', 'Summary', 'Details', 'Addresses (3)', 'Notes (5)', 'Social Media', and 'Analysis'. The 'Details' tab is selected. Below it, a sub-menu has 'Person Info' selected, along with 'Phone & Addresses' and 'Bank Accounts'. The main content area displays 'Person: William Andy' and a 'Details' section. This section includes a table with columns for 'Name' (Full Name: William Andy, Prefix: William) and 'Employment Info' (Occupation: , Employer: Albertson's). There are also sections for 'Notes' and 'Social Media'.

Figure 1 Source

The screenshot shows a user interface for managing a company record. At the top, there's a header with the 'TrainingApp' logo, a search bar, and a contact dropdown. On the left is a sidebar with tabs: 'Actions', 'Summary', 'Details', 'Addresses (1)', 'Notes (0)', 'Social Media', 'Analysis', 'Interactions', and 'History'. The 'Details' tab is selected. Below it, a sub-menu has 'Company Info' selected, along with 'Phone & Addresses' and 'Bank Accounts'. The main content area displays 'Company: Albertson's' and a 'Details' section. This section includes a table with columns for 'Name' (Albertson's), 'Primary Contact' (Primary Contact: William Andy, Address: 345 Fir Lane, La Canada, CA 91352), and 'Additional Info' (Email Address: , Tax ID: *****). To the right, there's a sidebar titled 'Employee Info' with a list of names: James Anders, Samantha And, William Andy, Eric Andy, and Contact Insight.

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

The screenshot shows the 'Details' page for a person named William Andy. The left sidebar has 'Actions' at the top, followed by 'Summary', 'Details' (which is selected), 'Addresses (3)', 'Notes (5)', 'Social Media', and 'Analysis'. The main content area has a header 'Person: William Andy' and a sub-header 'Details'. Below this are three tabs: 'Person Info' (selected), 'Phone & Addresses', and 'Bank Accounts'. The 'Person Info' tab displays Name (Full Name: William Andy, Prefix: William), Employment Info (Occupation: , Employer: Albertson's), and License Info. The 'Phone & Addresses' and 'Bank Accounts' tabs are currently empty.

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

The screenshot shows the 'Details' page for a company named Albertson's. The left sidebar has 'Actions' at the top, followed by 'Summary', 'Details' (selected), 'Addresses (1)', 'Notes (0)', 'Social Media', 'Analysis', 'Interactions', and 'History'. The main content area has a header 'Company: Albertson's' and a sub-header 'Details'. Below this are three tabs: 'Company Info' (selected), 'Phone & Addresses', and 'Bank Accounts'. The 'Company Info' tab displays Name (Albertson's), Primary Contact (William Andy), Address (345 Fir Lane, La Canada, CA 91352), Email Address, Additional Info (Tax ID: *****), Inspection Required? (No), and Preferred Currency. To the right of the main content, there is a sidebar titled 'Employee Info' which includes 'Can Have Employees?' (Yes), 'Number of Employees' (4), and a table titled 'Employees' with entries for James Andersen, Samantha Andrews, William Andy, and Eric Andy. There is also a section titled 'Contact Insights'.

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?**



7.3.2 Configuration

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

InsuranceSuite 10 Fundamentals: Kickstart - Student Workbook

The screenshot shows a company record for "Albertson's". The top navigation bar includes "TrainingApp", "Search", and "Contact". The left sidebar has sections for "Actions", "Summary", "Details", "Addresses (1)", "Notes (0)", "Social Media", "Analysis", "Interactions", and "History". The main content area displays "Company: Albertson's" and "Details" for "Company Info". The "Company Info" tab is selected, showing fields like Name (Albertson's), Primary Contact (William Andy), Address (345 Fir Lane, La Canada, CA 91352), and Email Address. Below this is an "Additional Info" section with Tax ID (*****). The right side contains "Employee Info" with a list of employees: James Anders, Samantha Andy, Eric Andy, and William Andy. At the bottom is a "Contact Insight" section.

Company Info		Phone & Addresses	Bank Accounts
Name	Albertson's		
Primary Contact	William Andy		
Address	345 Fir Lane La Canada, CA 91352		
Email Address			
Additional Info			
Tax ID	*****		
Inspection Required?			
Preferred Currency			

Figure 1 Source

The screenshot shows a person record for "William Andy". The top navigation bar includes "TrainingApp", "Search", and "Contact". The left sidebar has sections for "Actions", "Summary", "Details", "Addresses (3)", "Notes (5)", "Social Media", "Analysis", "Interactions", and "History". The main content area displays "Person: William Andy" and "Details" for "Person Info". The "Person Info" tab is selected, showing fields like Full Name (William Andy), Prefix, First Name (William), Middle Name, Last Name (Andy), and Suffix. The right side contains "Employment Info" with Employer (Albertson's), "License Info" with Driver's License and State, and "Contact Insights".

Person Info		Phone & Addresses	Bank Accounts
Name			
Full Name	William Andy		
Prefix			
First Name	William		
Middle Name			
Last Name	Andy		
Suffix			

Figure 2 Destination



Hint

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

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. You will have to configure the `available` property to fix this:

```
(anABCContact as ABCCompany).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

The screenshot shows the TrainingApp interface with the title "Company: Albertson's". The left sidebar has sections like Actions, Summary, Details, Addresses (1), Notes (0), Social Media, Analysis, Interactions, and History. The main area is titled "Details" and contains tabs for Company Info, Phone & Addresses, and Bank Accounts. Under Company Info, it shows Name (Albertson's), Primary Contact (William Andy), Address (345 Fir Lane, La Canada, CA 91352), Email Address, and Additional Info (Tax ID: *****). To the right, there are sections for Employee Info, Can Have Employees, Number of Employees, Employees, and Contact Insight, which lists names like James Anders, Samantha Andy, Eric Andy, and William Andy.

Company Info		Phone & Addresses	Bank Accounts
Name	Albertson's		
Primary Contact	William Andy		
Address	345 Fir Lane La Canada, CA 91352		
Email Address			
Additional Info			
Tax ID	*****		
Inspection Required?			
Preferred Currency			

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

The screenshot shows the TrainingApp interface with the title "Person: William Andy". The left sidebar has sections like Actions, Summary, Details, Addresses (3), Notes (5), Social Media, Analysis, Interactions, and History. The main area is titled "Details" and contains tabs for Person Info, Phone & Addresses, and Bank Accounts. Under Person Info, it shows Name (Full Name: William Andy), Prefix, First Name (William), Middle Name, Last Name (Andy), and Suffix. To the right, there are sections for Employment Info (Occupation, Employer: Albertson's), License Info (Driver's License, State), and Contact Insights.

Person Info		Phone & Addresses	Bank Accounts
Name	Full Name	William Andy	
Prefix	First Name	William	
Middle Name	Last Name	Andy	
Suffix			



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)

Name

Full Name: (anABContact as ABPerson).FullNa...

Prefix: (anABContact as ABPerson).Pre...

First Name: (anABContact as ABPerson).FirstNa...

Middle Name: (anABContact as ABPerson).Middle...

Employment Info

Occupation: (anABContact as ABPerson).Occup...

Employer: (anABContact as ABPerson).Em...

Input Column

Input Set Ref

Shared section mode: ARAdjudicator

Page Configuration | **Text**

Properties: Properties | Layout config | Reflection | PostOnChange

Range Input

Basic properties

- editable: true
- id***: Employer
- label: DisplayKey.get("Training.Employer")
- required
- value***: (anABContact as ABPerson).Employer
- valueRange***: queryForCompaniesWhoCanHaveEmployees()
- valueType***: ABCompany

Advanced properties

- action: ABCompanyDetailsPage.go((anABContact as ABPerson).Employer)
- actionAvailable
- align: <none selected>
- available: (anABContact as ABPerson).Employer != null or CurrentLocation.InEditMode

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 ABCCompany).PrimaryContact)
```

Basic properties	
editable	true
id*	PrimaryContact
label	DisplayKey.get("Training.PrimaryContact")
required	
value*	(anABContact as ABCCompany).PrimaryContact
valueRange*	(anABContact as ABCCompany).Employees
valueType*	ABContact
Advanced properties	
action	ABPersonDetailsPage.go((anABContact as ABCCompany).PrimaryContact)
actionAvailable	
align	<none selected>
available	(anABContact as ABCCompany).PrimaryContact != null or CurrentLocation.InEditMode

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 10.0, Guidewire Studio 10.0, 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

- Verify that the *Run in Debug Process* icon is available in Gosu Scratchpad.

2. Write Gosu code

- Use the `trainingapp.base.QueryUtil.findContact()` method to retrieve the contact using the PublicID of the contact.
- 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.

The screenshot shows the Gosu Scratchpad interface with a toolbar at the top and a list of log entries below. The toolbar includes icons for Debug, Server, Debugger, and Console. The log entries are:

```
Burlingame Saab was created 2016-04-12.  
Primary address state is CA.  
The contact is of the subtype Auto Repair Shop and is NOT a strategic partner.  
Contact is NOT of the type Doctor.  
  
Rebecca Stevens was created 2016-04-12.  
Primary address state is CA.  
The contact is of the subtype Doctor and is NOT a strategic partner.  
Doctor category is General Care and doctor specialty is Neurology.
```

At the bottom, there are tabs for Run, Debug, TODO, Properties, Terminal, Messages, and Codegen. The 'Debug' tab is selected.



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

- Verify that the *Run in Debug Process* icon is available in Gosu Scratchpad.

2. Write Gosu code

- Use the `trainingapp.base.QueryUtil.findPerson()` method to retrieve each contact.
- 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

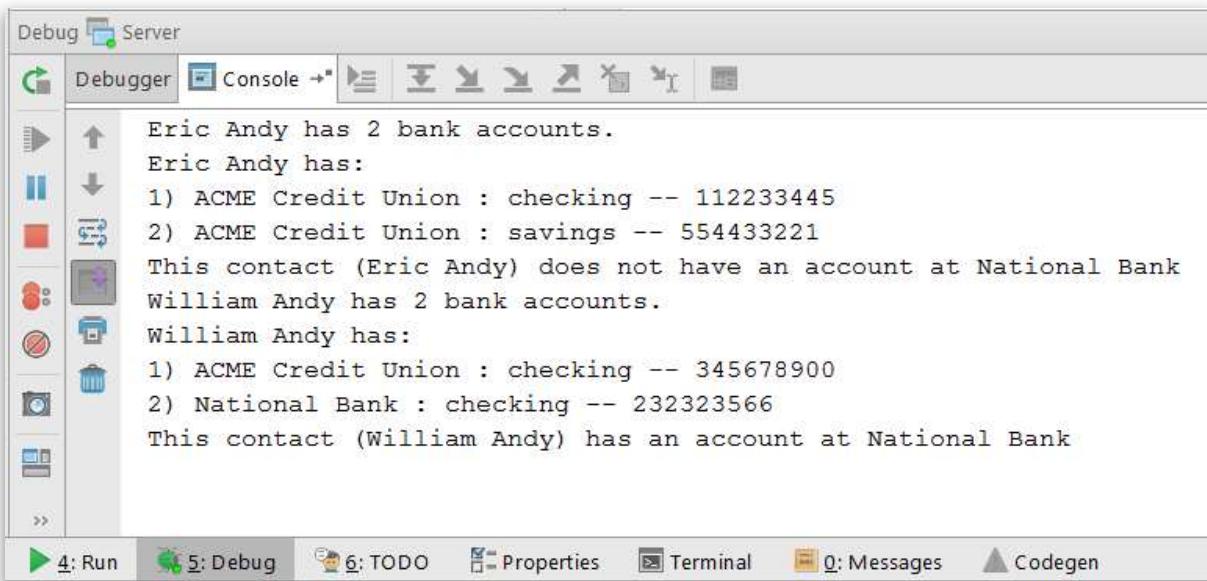


Activity

Verify the work you have done

1. Execute your Gosu Code in Gosu Scratchpad

- a) Verify the console output.



The screenshot shows the Gosu Scratchpad interface with the 'Console' tab selected. The output window displays the following text:

```
Eric Andy has 2 bank accounts.  
Eric Andy has:  
1) ACME Credit Union : checking -- 112233445  
2) ACME Credit Union : savings -- 554433221  
This contact (Eric Andy) does not have an account at National Bank  
William Andy has 2 bank accounts.  
William Andy has:  
1) ACME Credit Union : checking -- 345678900  
2) National Bank : checking -- 232323566  
This contact (William Andy) has an account at National Bank
```

The interface includes a toolbar with icons for run, debug, and server, and a bottom bar with tabs for Run, Debug, TODO, Properties, Terminal, Messages, and Codegen.



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

- 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)
        }
    }
}
```

Lesson 9

Gosu Rules

An insurance company wants to ensure that every ABDoc has a value specified for the doctor specialty field in the application. TrainingApp users can continue creating ABDocs 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:

TrainingApp

Actions

Summary

Details

Addresses (1)

Notes (0)

Social Media

Analysis

Interactions

History

Summary

Update Cancel

Basics Social Media Analysis

Suggest Least Busy User

Basic Information

Name	William Andy
Public ID	ob:5
Assigned User	Alice Applegate
Created On	03/28/2016

Primary Address

Country	United States
Address 1	411 Lost Way Ave
Address 2	
Address 3	
City	Foster City
County	
State	California
ZIP Code	94101
Address Type	Home
Description	Home is where the heart is
Valid Until	02/03/2019

Flagged Entries

	View	Date Flagged	Reason	Date Unflagged
!	View/Edit	03/28/2016	Doctor specialty is unspecified.	(1)

Every time an ABDoc 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.

“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 10.0, Guidewire Studio 10.0, 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.



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.



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?

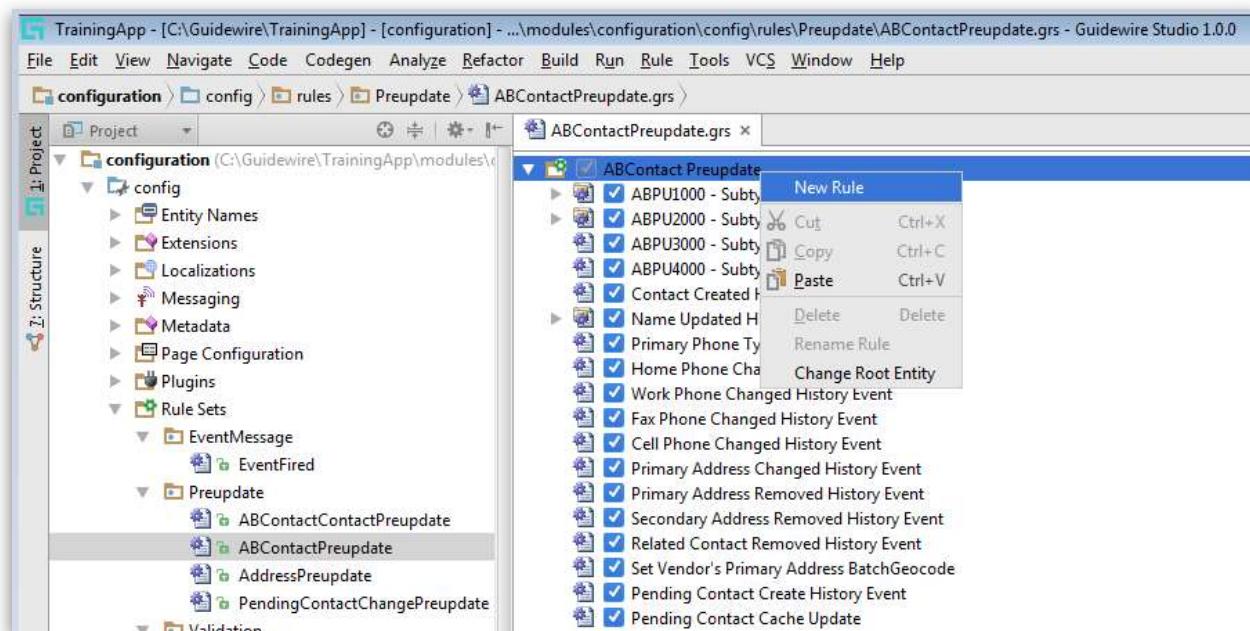
Creating an initializing a new FlagEntry entity

9.3.2 Configuration

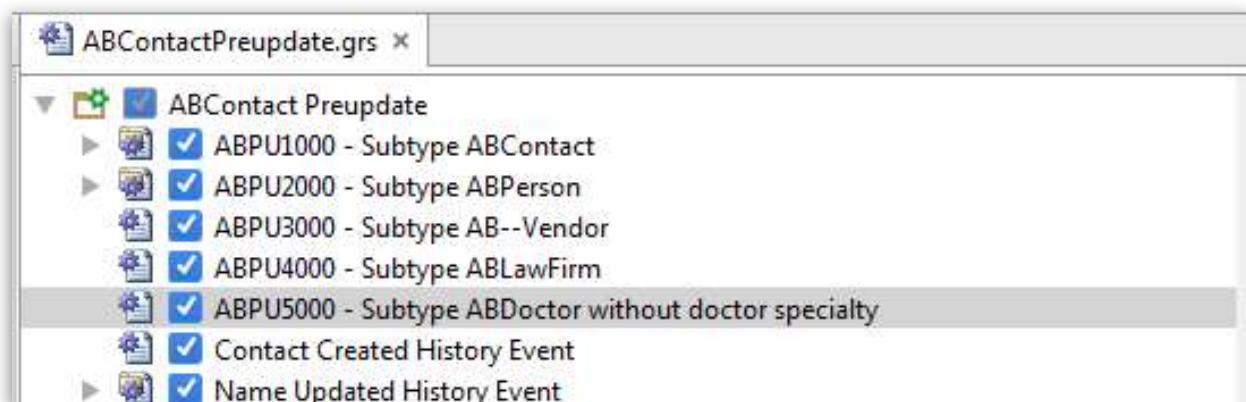


Solution 1

- 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 ABContactContactPreupdate because you will have to delete and recreate it later.



- 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.

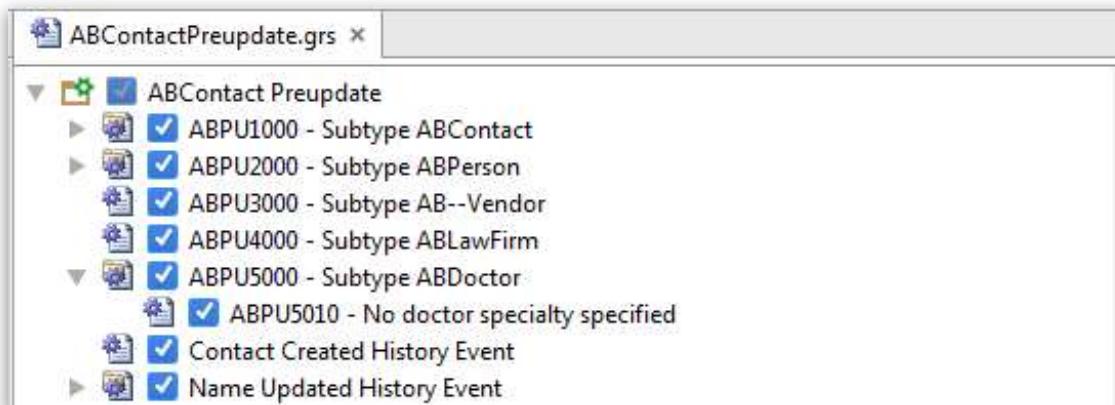
Important: please make sure that you copy and paste the sections individually without the parts highlighted in dark gray.

```
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.currentTimeMillis()
    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.currentTimeMillis()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)
```

InsuranceSuite 10 Fundamentals: Kickstart - Student Workbook

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 an 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:

The wireframe shows a contact record for 'Person: William Andy'. The 'Analysis' tab is selected. The 'Strategic Partner' field is highlighted with a red box. A yellow callout box contains the following tasks:

- (1) Add a Read-only Next Courtesy Contact field. It should be automatically calculated by adding 6 months to the Last Courtesy Contact.
- (2) Make the Strategic Partner field read-only
- (3) Add an Upgrade to Strategic Partner toolbar button. The button is clickable only if the contact is not yet a strategic partner AND the screen is in edit mode. When the user clicks on it, it executes the logic defined later in this document.

"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 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 10.0, Guidewire Studio 10.0, 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

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

- In Guidewire Studio, create a package called `succeedlab.ta.enhancements.entity`
- Create an enhancement for the ABContact entity that conveys that this enhancement relates to analysis.

3. Create a getter property named `NextCourtesyContact()`

- 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()`

- 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 mode 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.

Company: Albertson's

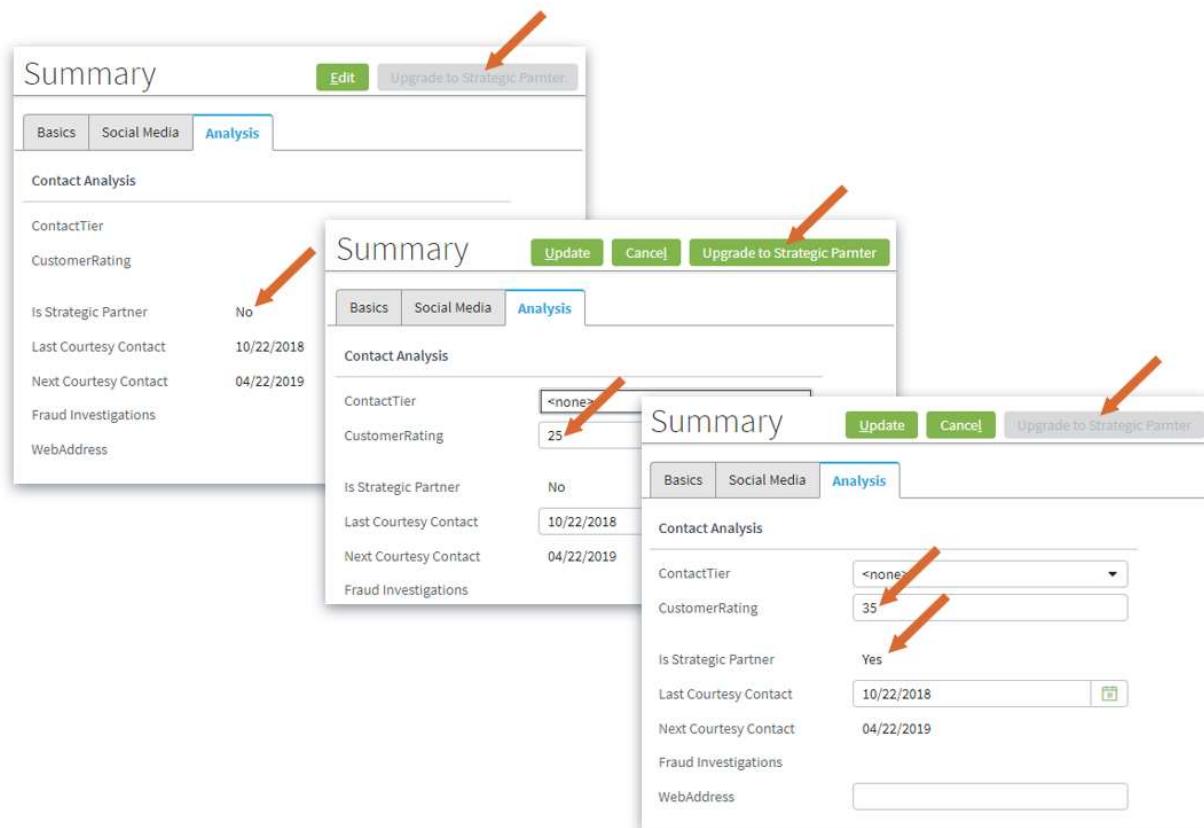
Summary

Basics Social Media **Analysis**

Contact Analysis

ContactTier	<none>
CustomerRating	
Is Strategic Partner	
Last Courtesy Contact	09/05/2018
Next Courtesy Contact	03/05/2019
Fraud Investigations	
WebAddress	

- 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.



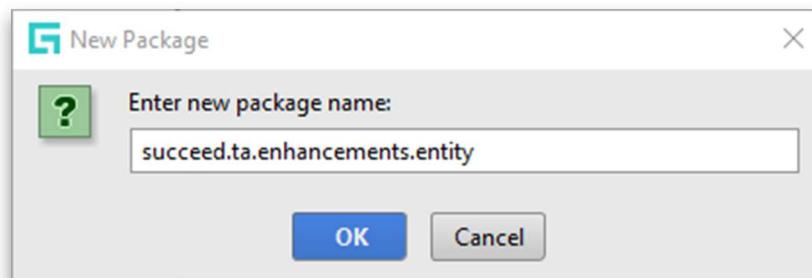
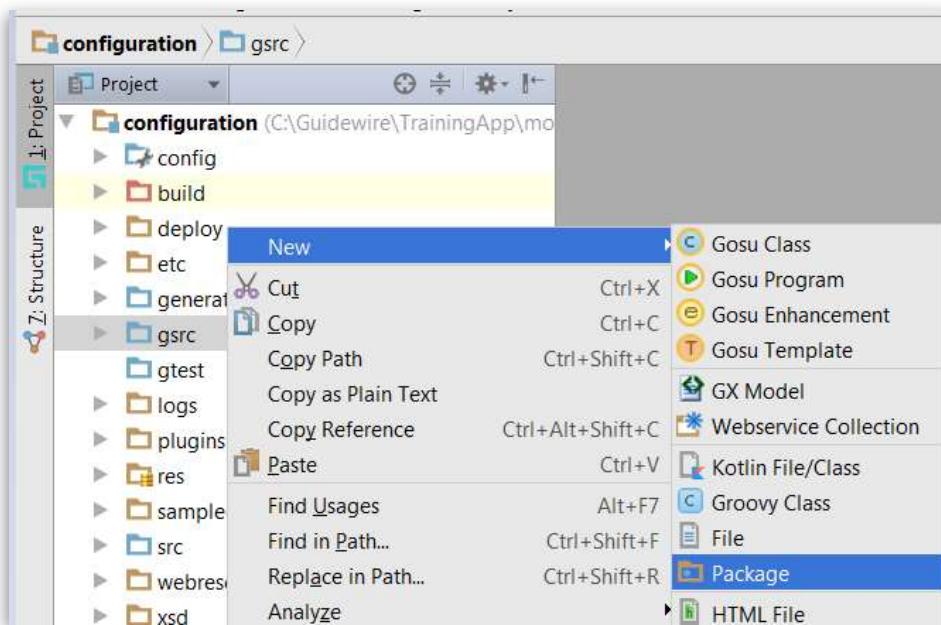
10.3 Lab Solution: Create a new enhancement



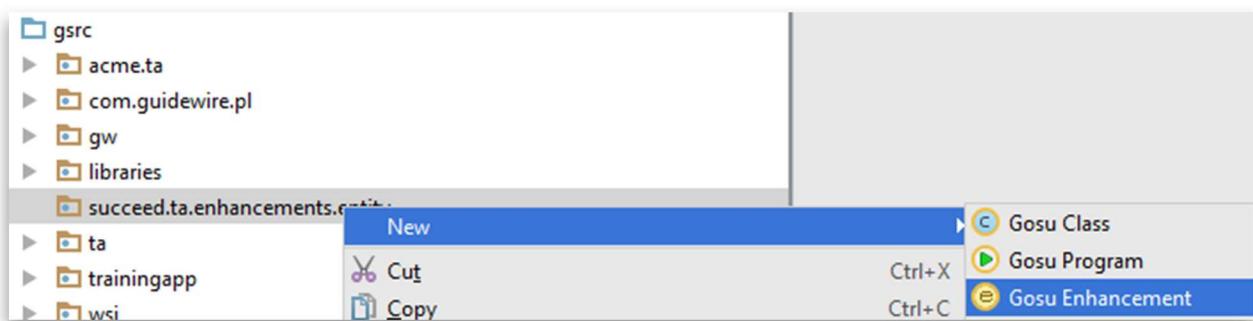
Solution

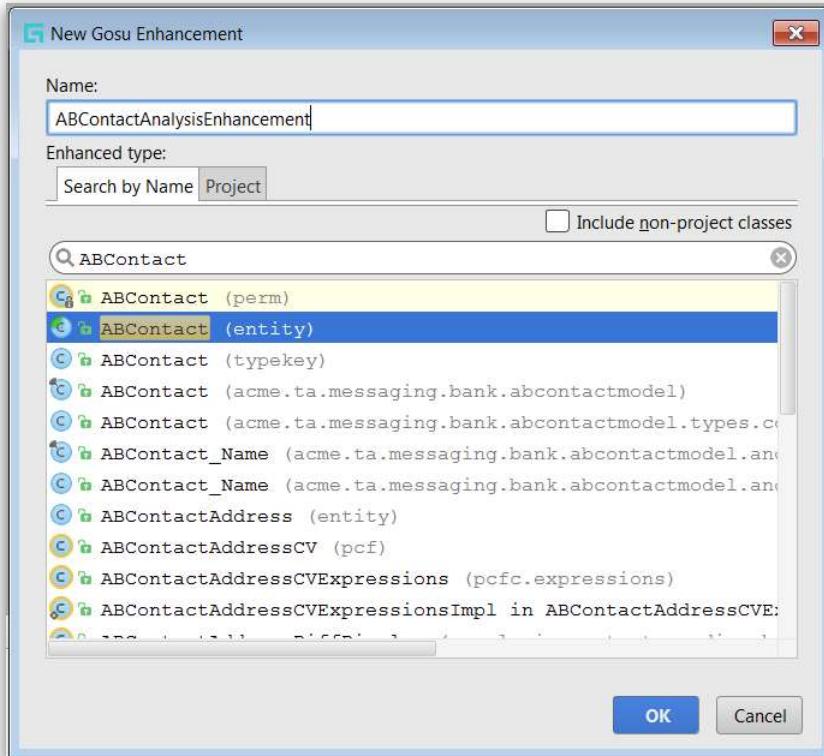
1. In Guidewire Studio, create the package and the enhancement

- 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()

- 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()

- 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
    }
}
```

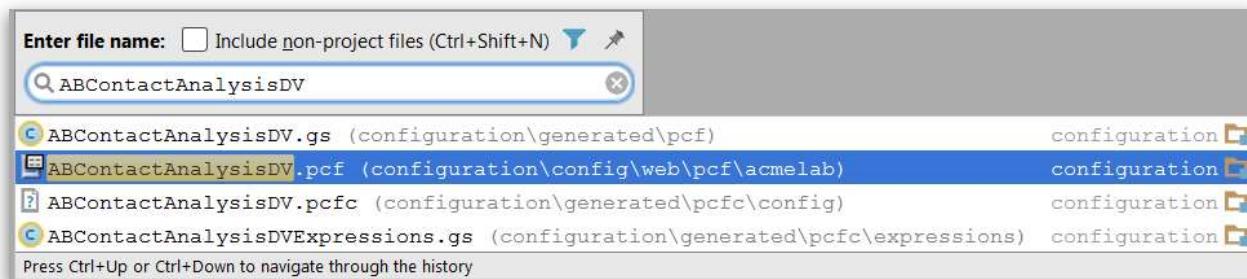
10.3.1 Modify the PCF configuration



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.

The screenshot shows the 'ABContactAnalysisDV.pcf' page configuration interface. At the top, there's a header bar with tabs for 'Detail View : ABContactAnalysisDV' and 'anABContact'. Below this, the main content area has a section titled 'Contact Analysis'.

In the 'Contact Analysis' section, there are three input fields:

- 'Contact Tier' dropdown: anABContact.ContactTier
- 'Customer Rating' dropdown: anABContact.CustomerRati...
- 'Strategic Partner' radio button group: with options 'Yes' and 'No'.

The 'Strategic Partner' group is highlighted with a blue border. Below this section, there's a toolbar with tabs for 'Page Configuration' (selected) and 'Text'. Underneath the toolbar, the title 'Boolean Radio Button Input' is displayed.

The 'Properties' tab is selected in the toolbar. In the properties grid, under the 'Basic properties' section, the 'editable' property is set to 'false' (highlighted with a red box). Other properties listed include:

Property	Value
falseLabel	
id*	IsStrategicPartner
label	DisplayKey.get("Ext.IsStrategicPartner")
required	
trueLabel	
value*	anABContact.IsStrategicPartner_Ext

- Add the appropriate read-only input widget below Last Courtesy Contact.
- Create a displaykey for the widget's label property that reads Next Courtesy Contact.
- Configure the widget to display the enhancement property value.

Basic properties	
<u>dateFormat</u>	<none selected>
<u>editable</u>	false
<u>id*</u>	NextCourtesyContact
<u>label</u>	DisplayKey.get("Ext.NextCourtesyContact")
<u>required</u>	
<u>timeFormat</u>	<none selected>
<u>value*</u>	anABContact.NextCourtesyContact

2. Modify ABContactAnalysisPage

- In Studio, open ABContactAnalysisPage.pcf.
- In the existing toolbar, add a toolbar button.
- Create a displaykey for the button's label property that reads Upgrade to Strategic Partner.
- Configure the toolbar button so that it calls the upgradeToStrategicPartner() method when clicked.
- 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.

Basic properties	
action	<code>anABContact.upgradeToStrategicPartner()</code>
id*	<code>UpgradeToStrategicPartner</code>
label	<code>DisplayKey.get("Ext.UpgradeToStrategicPartner")</code>

Advanced properties	
available	<code>CurrentLocation.InEditMode and anABContact.IsStrategicPartner_Ext != true</code>

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:

The screenshot shows a company details page for 'Albertson's'. The 'Company Info' tab is selected. The company name is 'Albertson's'. The primary contact is 'William Andy'. The company has 4 employees. The 'Employee Info' section shows four employees: James Andersen, Samantha Andrews, Eric Andy, and William Andy. The 'Cascade Email Address' button is highlighted in green.

Name	Job Title	Email Address
James Andersen		jandersen@elegal.com
Samantha Andrews		sandrews@andrewsmrd.com
Eric Andy		
William Andy		

Figure 1 Before clicking on Cascade Email Address

Company: Albertson's

Details

Company Info	Phone & Addresses	Bank Accounts
Name * Albertson's		
Primary Contact	William Andy	
Primary Contact	William Andy	
Country	United States	
Address 1	345 Fir Lane	
Address 2		
Address 3		
City	La Canada	
County		
State	California	

Employee Info		
Can Have Employees?	<input checked="" type="checkbox"/>	
Number of Employees	4	
Employees Cascade Email Address		
Name	Job Title	Email Address
James Andersen		info@Albertsons.com
Samantha Andrews		info@Albertsons.com
Eric Andy		info@Albertsons.com
William Andy		info@Albertsons.com

Contact Insights

Insight Score
0

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 10.0, Guidewire Studio 10.0, 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?



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



11.3 Lab Solution: Fix a bug



Solution

1. Open Guidewire Studio for TrainingApp

```

20:10:25,982 INFO MW-startThreadPool-15 poolSize=1
20:10:25,988 INFO Destination: 15 started
20:10:26,050 INFO Starting QPlexor...
20:10:26,053 INFO QPlexor started
20:10:26,135 INFO Increasing runlevel to 'MULTIUSER'
20:10:26,137 WARN
20:10:26,138 WARN !!! The server is in "development" mode.
20:10:26,138 WARN
20:10:26,138 WARN
20:10:26,138 WARN !!! The server is using the Dynamic Code Inlining feature.
20:10:26,138 WARN
20:10:26,143 INFO ***** ContactManager ready *****
20:10:26,155 INFO Started o.e.j.w.WebAppContext@7a67e3c6{/abc}

```

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

action	<code>cascadeCompanyMainEmailAddress ()</code>
id*	<code>CascadeEmailAddress</code>
label	<code>DisplayKey.get("Training.Company.CascadeEmailAddress")</code>

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()

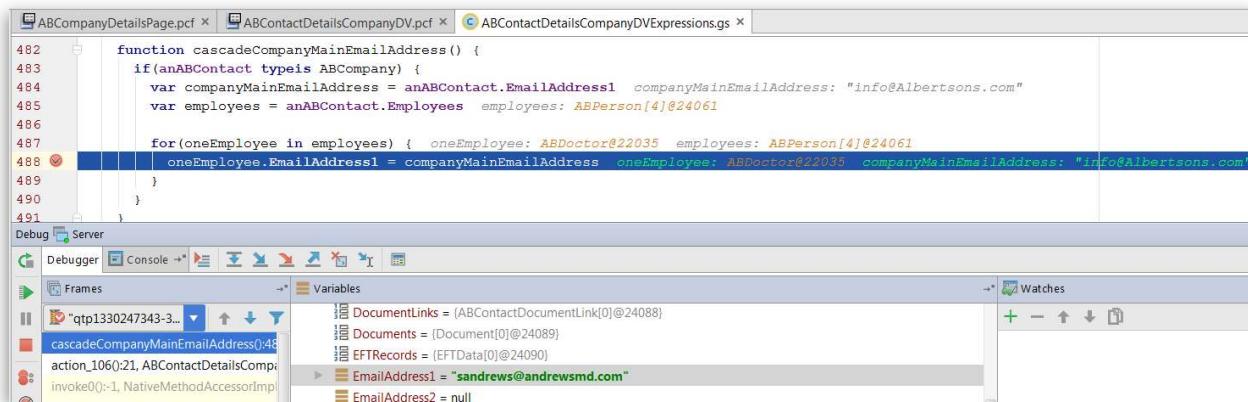
2. Go to the function declaration and verify it is a PCF function

The screenshot shows the PCF page editor interface. At the top, there are two tabs: "ABCompanyDetailsPage.pcf" and "ABContactDetailsCompanyDV.pcf". The "ABContactDetailsCompanyDV.pcf" tab is active. Below the tabs, the page content is displayed in two columns. The left column contains fields for "Name *", "Primary Contact" (with a dropdown menu showing "(anABContact as ABCompany)...."), and "Shared section mode: BigToSmall". The right column contains sections for "Employee Info" (with checkboxes for "Can Have Employees?" and "Number of Employees"), "Employees" (with a toolbar), and a "Cascade Email Address" button. At the bottom of the page content, there is a "Page Configuration" tab and a "Text" tab. The "Text" tab is selected, showing the source code of the function. The code is as follows:

```
function cascadeCompanyMainEmailAddress() {  
    if(anABContact typeis ABCompany) {  
        var companyMainEmailAddress = anABContact.EmailAddress1  
        var employees = anABContact.Employees  
  
        for(oneEmployee in employees) {  
            oneEmployee.EmailAddress1 = companyMainEmailAddress  
        }  
    }  
}
```

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



The screenshot shows a debugger interface with the following details:

- Code Editor:** Displays a snippet of JavaScript code. The highlighted line is at row 488: `oneEmployee.EmailAddress1 = companyMainEmailAddress;`
- Variables Pane:** Shows the current state of variables:
 - DocumentLinks = (ABContactDocumentLink[0]@24088)
 - Documents = {Document[0]@24089}
 - EFTRecords = {EFTData[0]@24090}
 - EmailAddress1 = "sandrews@andrewsmd.com"
 - EmailAddress2 = null

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

```

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
            }
        }
    }
}

```

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



The screenshot shows a code editor window with three tabs at the top: "ABCompanyDetailsPage.pcf", "ABContactDetailsCompanyDV.pcf", and "ABContactDetailsCompanyDVExpressions.gs". The "ABContactDetailsCompanyDVExpressions.gs" tab is active, displaying the following Groovy script:

```
property set anABCContact ($arg : ABContact) { setRequireValue("anABCContact", 0, true)

function cascadeCompanyMainEmailAddress() {
    if(anABCContact typeis ABCompany) {
        var companyMainEmailAddress = anABCContact.EmailAddress1
        var employees = anABCContact.Employees

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

The code implements a function `cascadeCompanyMainEmailAddress` that checks if the current contact is a company. If it is, it sets the `EmailAddress1` field for all employees to the company's main email address. A yellow lightbulb icon is shown next to the first line of the loop, indicating a potential issue or warning.

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

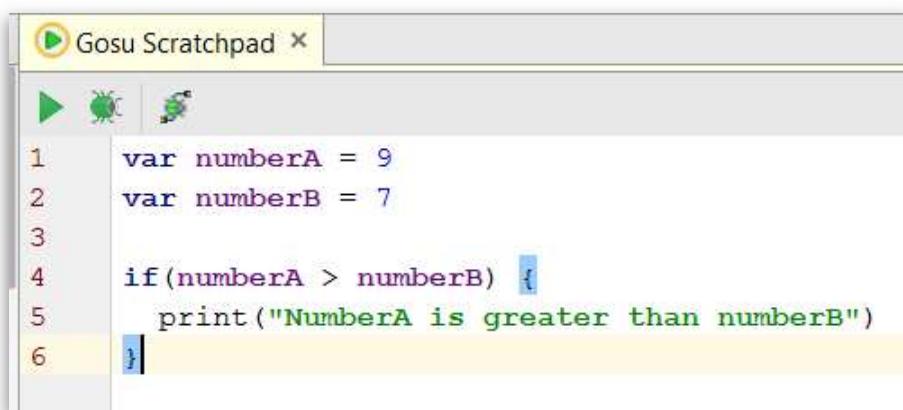
Refer to variables, types and symbols using the exact same case every time, otherwise the application won't compile.



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.



The screenshot shows a Gosu Scratchpad window. The code is as follows:

```
1 var numberA = 9
2 var numberB = 7
3
4 if(numberA > numberB) {
5     print("NumberA is greater than numberB")
6 }
```

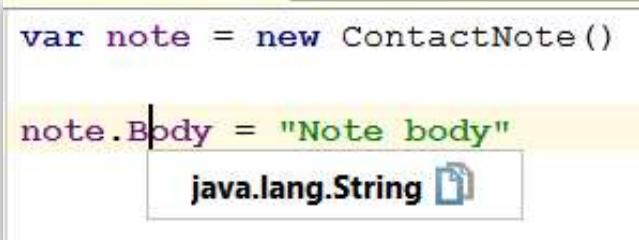
A yellow rectangular highlight is placed over the entire body of the if-block, from the opening brace on line 4 to the closing brace on line 6.



Hint

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

Use CTRL + SHIFT + G



The screenshot shows a Gosu code editor with the following code:

```
var note = new ContactNote()

note.Body = "Note body"
```

The cursor is positioned at the end of "note.Body" on the second line. A tooltip-like box appears below the cursor, containing the text "java.lang.String" and a small icon of a document.



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.

```
var note = new ContactNote()

note.|
```

The screenshot shows a code editor with the following Gosu code:

```
var note = new ContactNote()

note.|
```

A code completion dropdown is open, listing properties and methods of the `ContactNote` class. The dropdown includes:

- `getBeanVersion()`: Integer
- `Body`: String
- `CreateTime`: Date
- `ABContact`: ABContact
- `BeanVersion`: Integer
- `ContactNoteType`: ContactNoteType
- `CreateUser`: User
- `getABContact()`: ABContact
- `getBody()`: String
- `getContactNoteType()`: ContactNoteType
- `... (truncated)`

Below the dropdown, a note says: "Dot, space and some other keys will also close this lookup and be inserted into editor >>".

```
var note = new ContactNote()

note.Body|
```

The screenshot shows a code editor with the following Gosu code:

```
var note = new ContactNote()

note.Body|
```

A code completion dropdown is open, listing properties and methods of the `Body` field. The dropdown includes:

- `Body`: String
- `getBody()`: String
- `setBody(value:String)`: void

Below the dropdown, a note says: "Dot, space and some other keys will also close this lookup and be inserted into editor >>".

Basics



Quick overview of basic Gosu syntax

Feature	Syntax	Try it!
Gosu primitives	<p>Gosu primitives exist for compatibility with the Java language.</p> <p>Primitive types cannot hold the null value. (null is a special value that means empty)</p> <p>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 for any</p>	<pre>// declaring a primitive with initial value var i : int = 5</pre> <pre>// declaring non-primitive with null value var i2 : Integer = null</pre> <pre>// non-primitive type is coerced to primitive type // null value will be converted to 0 <i>i = i2</i></pre>

	<p>numeric data and false for boolean data).</p> <p>Primitives: int, byte, char, short, long, float, double, boolean</p>	<pre>print(i)</pre>
Print message	<p>Important: never use it in production, use Loggers instead</p> <p>print(message/value to print)</p>	<pre>print("Hello World")</pre>
Comments	<p>One line comment: Highlight the text in Studio and use the CTRL + / keystroke</p> <p>Multiple lines comment: Highlight the text in Studio and use the CTRL + SHIFT + / keystroke</p>	<pre>//one line comment</pre> <pre>/* multiple lines comment */</pre>
Variable declaration	<pre>var variableName = initialValue var variableName : Type var variableName : Type = initialValue</pre>	<pre>//****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</pre>

		<pre>print(myVar) //Must declare a type because it can't be inferred var guess : String = null</pre>
If statement	if(condition) { block of code }	<pre>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") }</pre>
If-else statement	if(condition) { block of code } else { block of code }	<pre>var numberA = 9 var numberB = 7 if(numberA > numberB) { print("NumberA is greater than numberB") } else { print("NumberA is NOT greater than numberB") }</pre>
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 }	<pre>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") }</pre>
Ternary operator	Similar to if-else, but can only return expressions and cannot execute statements condition ? valueiftrue : valueiffalse	<pre>var numberA = 9 var numberB = 7 // Expressing the previous IF-ELSE // with the ternary operator: the // true and false cases MUST return a</pre>

		<pre>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)</pre>
Working with Strings	<ul style="list-style-type: none"> + 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 	<pre>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!"</pre>
Logical operators	<p>Equality: ==, !=</p> <p>Relational: >, <, >=, <=</p> <p>Logical operators: &&, and, , or, !, not</p> <p>Notes:</p> <p>== - in case of objects, it tests for object equality, just like .equals()</p> <p>== - in case of objects, it tests for referential equality</p>	<pre>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)) { 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</pre>

		<pre>10") }</pre>
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	<pre>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) }</pre>
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. 2 ? ta.QueryUtil? Alt+Enter 3 4 var data = <u>QueryUtil</u>	<pre>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) }</pre>
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. ta.QueryUtil.findXXX(publicID)	<pre>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 anABContact = ta.QueryUtil.findPerson("ab:5")</pre>
Creating new objects	var variableName = new Type()	<pre>var anABPerson = new ABPerson() var primaryAddress = new Address() var list = new java.util.List()</pre>
Type cast	Lets you access properties on the subtype level. Usually a type check is executed to be sure the	<pre>var anABContact : ABContact = ta. QueryUtil.findPerson("ab:5")</pre>

	<p>object is compatible with the new type.</p> <pre>(object as ChildSubType).PropertyOrMethod</pre>	<pre>print((anABContact as ABPerson).Age) //successful print((anABContact as ABCompany).NumberOfEmployees) //fails during runtime</pre>
Type check	<p>Tests if the object is of a given type. It supports automatic downcasting (No need to cast to object explicitly).</p> <p>object typeis Type</p> <p>The expression will return true if the object is of the Type <u>or any of its subtypes</u>.</p>	<pre>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)</pre>
Type check using the Subtype property	<p>Subtyped and subtype entities automatically have the Subtype property which identifies their correct type. You can use the Subtype property to execute a type check similarly to the typeis operator:</p> <pre>object.Subtype == typekey.Type.Subtype</pre> <p>However, there are two important differences:</p>	<pre>var anABContact : ABContact = ta.QueryUtil.findPerson("ab:5") if(anABContact.Subtype == typekey.ABContact.TC_ABPERSON) { print((anABContact as ABPerson).Age) } else { // This case will be executed for any of the ABPerson subtypes (e.g. ABDoctor, ABAttorney, ABPolicyPerson, etc.) print("Not a person") }</pre>

	<ul style="list-style-type: none"> - using the Subtype for typecheck doesn't automatically downcast - the expression will return true only if the object is of the specified Type <u>and</u> <u>not for its subtypes</u> 	
UI friendly names	<pre>entity.DisplayName typekey.DisplayName</pre>	<pre>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.S tate) // long format print(anABContact.PrimaryAddress.S tate.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.D isplayName) // long format var todayDate = DateUtil.currentDate() print(todayDate) // long format print("Today is: " + todayDate) // short format print("Today is: " + todayDate.toString()) // long format</pre>

Working with typecodes and dates as string	The application reformats typecodes and dates when concatenated with a String.	<pre>// 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</pre>
Null Safe invocation	variable?.method()	<pre>// 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())</pre>
Null-safe Default Operator	Sometimes you want a default value if an expression is null. For	<pre>var anABContact : ABContact = ta. QueryUtil.findPerson("ab:5")</pre>

	<p>this use case, Gosu supports the Elvis Operator: variable = expression ?: defaultvalue</p>	<pre>var middleName = anABPerson.MiddleName ?: "Unknown" print("MiddleName: " + middleName)</pre>
Working with typekey fields	<p>Typekey fields are defined in ETI, EIX and ETX files. They always get their values from the associated typelist.</p> <p>Assigning a new value: entity.typekeyfield = Typelist.TC_Typecode</p> <p>Testing for equality: entity.typekeyfield == Typelist.TC_Typecode</p>	<p>Important: the following code is not complete and cannot be executed in Gosu Scratchpad as presented here.</p> <pre>var note = new ContactNote() note.ContactNoteType = ContactNoteType.TC_GENERAL if(note.ContactNoteType == ContactNoteType.TC_GENERAL) { print("This is a general contact note") }</pre>

Gosu classes, properties and functions



Quick overview of Gosu class, properties and function syntax

Feature	Syntax	Try it!
Class definition	<p>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.</p> <pre>class ClassName { }</pre>	<pre>class Card { }</pre>
Variables	<p>Instance variable definitions usually start with _</p> <p>The default visibility for instance variables is private</p> <p>Visibility levels:</p> <ul style="list-style-type: none"> - private 	<pre>class Card { var _suit : String var _rank : String var _isFaceUp : Boolean }</pre>

	<ul style="list-style-type: none"> - internal - protected - public 	
Properties	<p>Note: Same syntax can be used in Enhancements</p> <p>Getter:</p> <pre>property get PropertyName() : Type { return this._propertyName }</pre> <p>Setter:</p> <pre>property set PropertyName(newValue : Type) { this._propertyName = newValue }</pre>	<pre>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 } }</pre>
Properties shortcut	<p>There is a shorter way to expose properties using the as keyword</p> <pre>var _variableName : Type as PropertyName</pre> <p>This will automatically generate a getter and setter.</p>	<pre>class Card { var _suit : String as Suite var _rank : String as Rank var _isFaceUp : Boolean as IsFaceUp }</pre>
Read-only Properties shortcut	<pre>var _variableName : Type as readonly PropertyName</pre>	<pre>class Card { var _suit : String as readonly Suite }</pre>
Function	<pre>function functionName([parameters]) : returnType { }</pre> <p>Notes:</p> <ul style="list-style-type: none"> - the same function syntax can be used in 	<pre>class Card { var _suit : String as Suite var _rank : String as Rank var _isFaceUp : Boolean as IsFaceUp function turnFaceDown() : void { this._isFaceUp = false } }</pre>

	<p>Enhancements and PCF files as well</p> <ul style="list-style-type: none"> - void keyword is optional - default visibility is public 	<pre> } } <i>// Invoke a function</i> var king = new Card() king.turnFaceDown() </pre>
Static function	<pre> static function functionName([parameters]) : returnType { } </pre> <p>Notes:</p> <ul style="list-style-type: none"> - the same function syntax can be used in Enhancements and PCF files as well - void keyword is optional 	<p>Important: this code cannot be executed in Gosu Scratchpad as presented here.</p> <pre> class MyStringUtil { <i>// Utility class</i> static function countSpaces(line: String): int { var count = 0 for (ch in line) { if (ch == ' ') { count++ } } return count } <i>}</i> <i>// Invoke a static function</i> print(MyStringUtil.countSpaces("this is a test")) </pre>
Creating and initializing new object	<pre> var variableName = new Type() { :propertyName = initialValue, :propertyName = initialValue,... } </pre>	<pre> 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", </pre>

```

        :IsFaceUp = true
    }

heartKing.turnFaceDown()

```

Arrays, loops and blocks



Quick overview of array, loop and block syntax

Feature	Syntax	Try it!
Declaring programmatic arrays	<pre> var arrayName : Type[] var arrayName : Type[] = new Type[size] var arrayName = new Type[size] </pre>	<pre> // Declaring programmatic array var myArray : String[] = new String[2] // Declaring programmatic array var arrayFoo : int[] arrayFoo = new int[5] </pre>
Accessing elements	<pre> var element = array[index] array[index] = new value </pre> <p>Arrays are indexed starting from 0</p>	<pre> var myArray : String[] = new String[2] myArray[0] = "First" myArray[1] = "Second" print(myArray[1]) </pre>
Size of the array	<pre> array.length array.Count </pre>	<pre> // 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) </pre>
Data model arrays	<p>Data model arrays are declared in the ETI, EIX and ETX files using the <array> tag. Every <array> tag</p>	<p>Important: the following code is not complete and won't work from Gosu Scratchpad. Visit the</p>

	<p>must have an associated reverse <foreignkey> on the other entity. The framework automatically generates the following two methods:</p> <pre>parentEntity.addToArrayName(newElement) parentEntity.removeFromArrayName(elementToRemove)</pre>	<p>Solution section of the Business Rules lesson for the complete example.</p> <pre>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)</pre>
For loop	<p>For loop allows you to iterate over both arrays and anything that implements the <code>java.lang.Iterable</code> interface</p> <pre>for(anObject in collection/range) { block of code referencing anObject }</pre> <p>You can specify ranges using the .. operator.</p> <p>start..end</p> <p>Examples:</p> <pre>1..10 "01/01/2016".toDate() .. "02/01/2016".toDate()</pre>	<pre>// 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 '1/1/2013 - 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) }</pre>
Indexed for loop	<p>Index starts from 0 and will be incremented by 1 in every loop cycle</p> <pre>for(anObject in collection index i) {</pre>	<pre>// Loading a contact from the Database (William Andy) var aContact = ta.QueryUtil.findPerson("ab:5")</pre>

	<pre>block of code referencing anObject }</pre>	<pre>// 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) }</pre>
Block	<p>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</p> <pre>\ variableName -> condition</pre> <p>Frequently used array functions:</p> <ul style="list-style-type: none"> - 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 	<pre>// Loading a contact from the Database (William Andy) var aContact = ta.QueryUtil.findPerson("ab:5") // Getting all the contactnotes (ContactNote[]) var contactNotes = aContact.ContactNotes // Finds the first contact note in the array that has the type General var isThereAContactNoteWithTypeGeneral = contactNotes.hasMatch(\ oneContactNote -> oneContactNote.ContactNoteType == ContactNoteType.TC_GENERAL) print("isThereAContactNoteWithType GeneralShorter: " + isThereAContactNoteWithTypeGeneral Shorter)</pre>

- | | | |
|--|--|--|
| | <ul style="list-style-type: none">- <code>countWhere</code>: return the count of elements that match a given condition | |
|--|--|--|

the count of elements
that match a given
condition

Appendix B: Quick reference

Guidewire TrainingApp Example List



Example List

What is Example List?

TrainingApp has at least one example of virtually every objective in the InsuranceSuite 9.0 Fundamentals – Kickstart and InsuranceSuite 9.0 Fundamentals – Essentials courses. To find an example of a specific type of configuration, in Settings, select Example List. The Example List page organizes examples alphabetically within a functionality group: Data Model, User Interface, Gosu, and Cross-Section Functionality.

The screenshot shows the Guidewire TrainingApp Example List interface. At the top, there's a navigation bar with the 'GUIDEWIRE TrainingApp' logo, a search bar, and a dropdown menu containing 'International', 'Example List' (which is highlighted in blue), and 'About'. Below the navigation is a search bar with the word 'Search' and an 'Actions' button. The main content area has a title 'TrainingApp9-General-1-9.0.0 (ContactManager 9.0.0 Ferrite (GA) Patch 1)' and a 'Welcome to TrainingApp' message. Underneath, there's a section titled 'Example List' which contains a table:

Functionality	Example	Related Config Files
Delegate	Verifiable_Ext delegate (implemented by VendorEvaluation)	Verifiable_Ext.eti and VendorEvaluation.eti
Edge foreign key	ABCompanyVendor entity (FinanceManager and PaymentContact are self-referencing fields)	ABCompanyVendor.etx
Entity (creating subtype)	ABAutoScrapYard_Ext entity	ABAutoScrapYard_Ext.eti
Entity (creating "top-level")	Building_Ext entity	Building_Ext.eti
Entity (extending base subtype)	ABPersonVendor entity	ABPersonVendor.etx

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

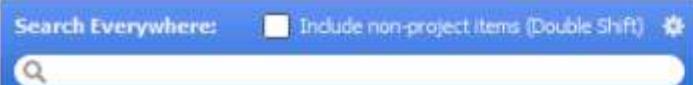
Studio productivity



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.

 <input checked="" type="checkbox"/> Show files ON	
Show symbols	<input type="checkbox"/> OFF
Show tool windows	<input type="checkbox"/> OFF
Show run configurations	<input type="checkbox"/> OFF
Show actions	<input type="checkbox"/> OFF
Show IDE settings	<input type="checkbox"/> OFF
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	<p>To open Location Information</p> <p>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.</p> <p>Location Info is also useful when Studio is not running.</p>
ALT + SHIFT + W	<p>To open Widget Inspector</p> <p>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.</p>
ALT + SHIFT + E	<p>To open a PCF file in Studio from the Browser.</p> <p>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.</p>
ALT + SHIFT + L	<p>To reload PCF and displaykey changes without restarting the Server</p> <p>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.</p> <p>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.</p>
ALT + SHIFT + T	<p>To access the Server Tools and Internal Tools</p> <p>You can execute administrative tasks from the Server Tools tab:</p> <ul style="list-style-type: none"> - Running Batch Processes - Setting log level - Viewing logs, etc. <p>You can perform different development tasks from the Internal Tools tab (important: this should be disabled in production!)</p> <ul style="list-style-type: none"> - Reload development resources (PCF files, Web templates, Workflow engine, Display Names) - Load sample data - Test system clock (set it to the future)

Data Model

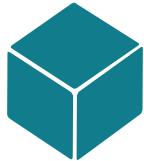


Entity definition files

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.	<i>EIX files neither created nor modified by configuration developers.</i>
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 <columParam> 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>	<p>An array defines a set of additional entities of the same type to associate with the main entity.</p> <p>The application automatically generates <code>addToArrayName(newObject)</code> and <code>removeFromArrayName(object)</code> functions on the parent entity to add and remove elements.</p>	<p>An array requires a reverse foreignkey because the array is virtual and maintained in code.</p> <p>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.</p> <p>For example, an ABContact entity includes an array of BankAccount entities. Thus, the BankAccount entity must have a foreignkey pointing to ABContact.</p>
<typekey>	<p>A typekey field is an entity defined field associated with a specific typelist.</p> <p>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.</p>	<p>It is often rendered as a dropdown in the UI.</p> <p><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</p>

		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 Area Input	Text Cell Text Area Cell	java.lang.String
date, datetime	Date Input	Date Cell	java.util.Date
bit	Boolean Dropdown Input Boolean Radio Button Input Check Box Input	Boolean Radio Cell Checkbox Cell	java.lang.Boolean
foreignkey	Range Input	Range Cell	OtherEntityType

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

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

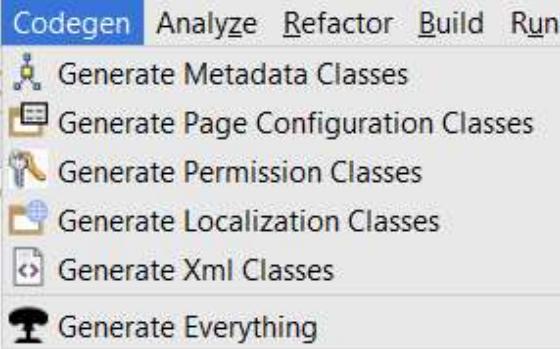
How does code generation and validation work?

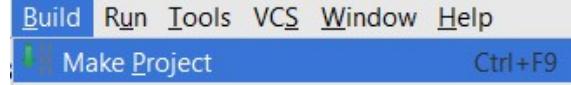
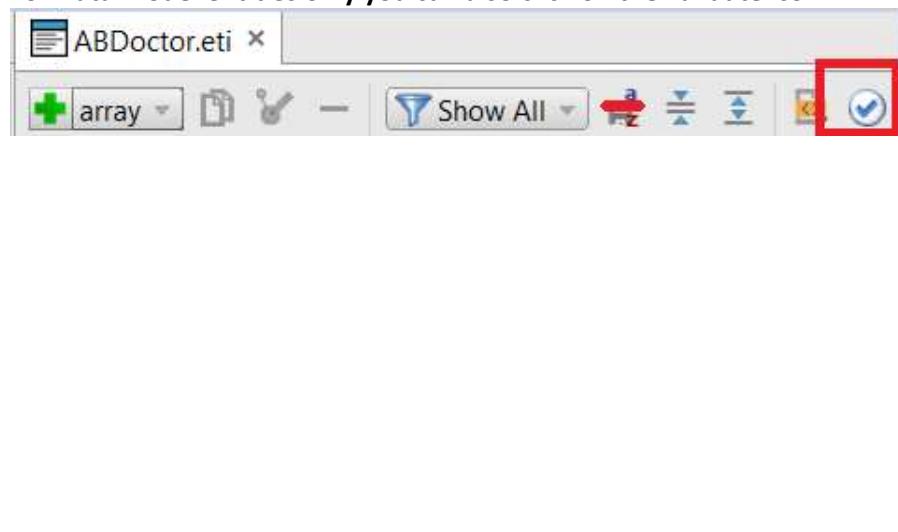
In InsuranceSuite 9.0, Guidewire Studio makes the process of code generation explicit and the generated code itself accessible. Internal code generators process resources like PCFs, entities and typelists in order to produce Gosu and Java classes.

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 <i>incremental</i> 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 <i>bulk mode</i> , meaning Studio processes all the relevant resources.
During project make	Studio invokes incremental code generators for the sets of the files	<i>Incremental</i>

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

File types	Description	Output of code generation
PCF files	<p>Studio supports incremental code generation for PCF files.</p>	<p>The code generation of the PCF files results in the following:</p> <ul style="list-style-type: none"> - 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/pcf/config

		<p>package. This is a binary file that describes the PCF file for the PCF runtime. The file is not human readable.</p> <ul style="list-style-type: none"> - The Expression.gs file in the .../configuration/generated/pcf/expressions package. This is a Gosu class that contains all container and widget expressions. Files in the pcf.expressions package are also for the PCF runtime. <p>You can debug these files.</p>
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:

```

at org.eclipse.jetty.xml.XmlConfiguration$1.run(XmlConfiguration.java:1250) <1 internal calls>
at org.eclipse.jetty.xml.XmlConfiguration.main(XmlConfiguration.java:1174) <4 internal calls>
at org.eclipse.jetty.start.Main.invokeMain(Main.java:509)
at org.eclipse.jetty.start.Main.start(Main.java:651)
at org.eclipse.jetty.start.Main.main(Main.java:99)
at com.guidewire.commons.jetty.GWServerJettyServerMain.main(GWServerJettyServerMain.java:69)
Disconnected from the target VM, address: '127.0.0.1:56551', transport: 'socket'

Process finished with exit code 1

```

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**:

```

tnonyi-W34 ZULB-U4-ZL-Z1Z47US,966 WARN FAILED ServerConnector@1988aa0[HTTP/1.1](0.0.0.0:8880): java.net.BindException: Address already in use: bind
at sun.nio.ch.Net.bind0(Native Method)
at sun.nio.ch.Net.bind(Net.java:437)
at sun.nio.ch.Net.bind(Net.java:429)
at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:223)
at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:74)
at org.eclipse.jetty.server.ServerConnector.open(ServerConnector.java:264)
at org.eclipse.jetty.server.AbstractNetworkConnector.doStart(AbstractNetworkConnector.java:80)
at org.eclipse.jetty.util.component.AbstractLifeCycle.start(AbstractLifeCycle.java:69)
at org.eclipse.jetty.server.Server.doStart(Server.java:303)
at com.guidewire.commons.jetty.GWServerJettyServerMain$JettyServer.doStart(GWServerJettyServerMain.java:83)
at org.eclipse.jetty.util.component.AbstractLifeCycle.start(AbstractLifeCycle.java:69)
at org.eclipse.jetty.xml.XmlConfiguration$1.run(XmlConfiguration.java:1250) <1 internal calls>
at org.eclipse.jetty.xml.XmlConfiguration.main(XmlConfiguration.java:1174) <4 internal calls>
at org.eclipse.jetty.start.Main.invokeMain(Main.java:509)
at org.eclipse.jetty.start.Main.start(Main.java:651)
at org.eclipse.jetty.start.Main.main(Main.java:99)
at com.guidewire.commons.jetty.GWServerJettyServerMain.main(GWServerJettyServerMain.java:69)
Disconnected from the target VM, address: '127.0.0.1:56551', transport: 'socket'

Process finished with exit code 1

```

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:

```

Debug: Server Server
Debugger Console
tnonyi-w54 2016-04-21 21:13:05,651 WARN LeaseCompilease10F33LDSSU-10E1-4116-9974-C3C7A31YD04JL,_name=zu) currently owned by this server expired.
tnonyi-w54 2016-04-21 21:13:05,651 INFO Shutting down destination 20 - Safe Ordering Demo
tnonyi-w54 2016-04-21 21:13:05,651 INFO Message writer for 20: Shutting down message writer: begin
tnonyi-w54 2016-04-21 21:13:05,651 INFO MW-20 shutdown: Shutting down message writer: begin
tnonyi-w54 2016-04-21 21:13:06,651 INFO stopped destination: 20 success: true
tnonyi-w54 aapplegate 2016-04-21 21:18:34,249 INFO runLifecycleSteps (39 ms) location=ABContacts; Sources='NewContactPopup/_crumb_'
tnonyi-w54 aapplegate 2016-04-21 21:18:38,671 INFO runLifecycleSteps (2830 ms) location=ServerTools; Sources='tabBar/internalToolsHiddenLink'
tnonyi-w54 aapplegate 2016-04-21 21:20:20,577 INFO runLifecycleSteps (36 ms) location=UnsupportedTools; Sources='internalToolsTabBar/unsupportedToolsTab'
tnonyi-w54 aapplegate 2016-04-21 21:22:36,522 INFO Logging out user=103 'aapplegate'
2016-04-21 21:22:36,524 INFO runLifecycleSteps (26 ms) location=Logout; Sources='internalToolsTabBar/logoutTabBarLink'
2016-04-21 21:22:36,531 INFO CSRF token: ee9065f5d6f60357287357b4f07a384d3f63991 was generated for a session with id: kbataatuu3djmu8v5ic
thonyi-w54 2016-04-21 21:22:36,536 INFO runLifecycleSteps (5 ms) location=Logout; Sources=''
2016-04-21 21:22:37,009 INFO CSRF token: 6ef18319ab952873456066e4afab895b7021e816 was generated for a session with id: 1vv7fodpt7c89rxngsq
thonyi-w54 2016-04-21 21:22:37,011 INFO runLifecycleSteps (2 ms) location=Logout; Sources=''
2016-04-21 21:22:38,491 INFO CSRF token: d9d5a35dd3b54ac6664162abe8c3430ec414d55 was generated for a session with id: 3186u95854ew1wb03o7
thonyi-w54 su 2016-04-21 21:22:38,501 INFO Login user=3 'su' from 0:0:0:0:0:1 session=3186u95854ew1wb03o7vw0kf0 csrfToken=d9d5a35dd3b54ac6664162abe
thonyi-w54 su 2016-04-21 21:22:41,283 INFO runLifecycleSteps (1265 ms) location=Admin; Sources='tabBar/AdminTab'
thonyi-w54 su 2016-04-21 21:22:44,284 INFO runLifecycleSteps (90 ms) location=Admin; Sources='Admin/MenuLinks/Admin_Monitoring/(LocationGroupMenuItemWid
thonyi-w54 2016-04-21 21:27:29,083 INFO Memory usage: 373.924 MB used (both active and stale objects), 202.000 MB free, 575.925 MB total, 1979.750 MB
thonyi-w54 2016-04-21 21:42:29,083 INFO Memory usage: 447.533 MB used (both active and stale objects), 128.391 MB free, 575.925 MB total, 1979.750 MB
thonyi-w54 2016-04-21 21:57:29,122 INFO Memory usage: 362.303 MB used (both active and stale objects), 213.621 MB free, 575.925 MB total, 1979.750 MB

```

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.)