



Guidewire training materials contain Guidewire proprietary information that is subject to confidentiality and non-disclosure agreements. You agree to use the information in this manual solely for the purpose of training to implement Guidewire software solutions. You also agree not to disclose the information in this manual to third parties or copy this manual without prior written consent from Guidewire. Guidewire training may be given only by Guidewire employees or certified Guidewire partners under the appropriate agreement with Guidewire.

# Lesson objectives

- By the end of this lesson, you should be able to:
  - Describe the different ways methods can be written
  - Create PCF methods
  - Implement common use cases for PCF methods

This lesson uses the notes section for additional explanation and information.  
To view the notes in PowerPoint, select View → Normal or View → Notes Page.  
When printing notes, select Note Pages and Print hidden slides.

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

2

G U I D E W I R E

## Lesson outline

- Overview of Gosu methods
- Creating PCF methods
- Common use cases for PCF methods

## Methods

- A method is a set of statements that executes a logical unit of work
- Example logic:

```
function popupButtonText
  if flag entry's IsEditable boolean is true
    return "View/Edit"
  else
    return "View"
```

The diagram illustrates the logic of the `popupButtonText` method. It shows two overlapping 'Flag Entries' tables. The top table represents the general state, and the bottom table shows a specific entry. A red box highlights the 'View/Edit' button in the top table, and a red arrow points to the 'View' button in the bottom table, demonstrating how the method's logic applies to the specific entry's `IsEditable` field.

Flag Entries				
	View	Date Flagged ↓	Reason	Date Unflagged
⚠	View/Edit	11/18/2013	No email	

Flag Entries				
	View	Date Flagged ↓	Reason	Date Unflagged
	View	11/18/2013	No email address for this contact.	04/24/2014

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

4

GUIDEWIRE

The slide example illustrates the FlagEntriesLV PCF Method, `popupButtonText()`.

For a given flag entry, the method returns "View/Edit" if the flag entry is editable or "View" if the flag entry is not editable.

A flag entry's `IsEditable` field is true if the flag entry is open and the user has sufficient permissions to resolve the flag entry.

# Method syntax

```
...3 function popupButtonText(aFlagEntry : FlagEntry) : String {  
4     var buttonText = displaykey.Training.View  
5     if (aFlagEntry.IsEditable) {  
6         buttonText = displaykey.Training.ViewEdit  
7     }  
8     return buttonText  
9 }
```

- Syntax:

- **function name(param1 : datatype) : returnType {  
 // code to execute method  
 return returnValue  
}**

- Example

- Line 3: Defines function signature
- Line 8: Return statement

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

5

GUIDEWIRE

You can find the `popupButtonText()` PCF Method in `FlagEntriesLV.pcf`.

The recommended capitalization convention for methods is to use camelCase with an initial lower-case letter such as `popupButtonText`, `calculateAvailability`, or `assignToNextAvailableUser`.

A method can define input parameters of any data type. In the slide example, the function only defines one parameter. It is possible to have no input parameters. Multiple input parameters are comma delimited. A method can return any type of value found in Gosu, including Boolean, String, Integer, Object from the data model, Arrays and Void (no value returned). However, you can only return one type. It is possible to return a generic list or array of objects that can be of different types.

You can precede the function keyword with one or more modifier keywords. Guidewire modifiers include `public` and `private`, which are access modifiers. A method is public by default, meaning that it can be referenced from anywhere in a Guidewire application that uses Gosu. In contrast, a private method can be referenced only within the library in which it is defined.

A method signature consists of the name, input parameters and return type. In Gosu, methods can be overloaded. When you overload a method, the name of the method is the same, but a) the number of parameters is different for the methods or b) the parameter data types are different. In all cases, the return type stays the same.

## Method with return values

```
...3 function popupButtonText(aFlagEntry : FlagEntry) : String {  
4     var buttonText = displaykey.Training.View  
5     if (aFlagEntry.IsEditable) {  
6         buttonText = displaykey.Training.ViewEdit  
7     }  
8     return buttonText  
9 }
```

- When defined, all code paths must contain a return statement
  - Return value must be of the declared return type
  - It is possible to have more than one return statement
- Example:
  - Line 3: defines the return type as String
  - Line 4: defines variable of the type String
  - Line 8: returns variable that is either View or View\Edit

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

6

GUIDEWIRE

If a method's return type is not void, all code paths must return a value. Gosu requires a return statement for all possible paths through the method including all choices for conditional execution, such as if and switch statements.

It is possible to have more than one return statement. There are situations in which you want execution of the method to stop when a certain condition is met.

Gosu also mandates that a value specified in a return statement match the return type declared in the method. A missing return type or a mismatched return value generates a compiler error.

In the slide example, buttonText is defined as a String and is returned for all code paths. The String data type is implicit in the variable declaration because all displaykeys are Strings.

## Methods with no return value

```
...4 function deleteABContactSecondaryAddresses(): void {  
5     for (currentAddress in anABContact.SecondaryAddresses) {  
6         anABContact.removeAddress(currentAddress)  
7     }  
8 }
```

- If method does not need to return a value:
  - Set return value to void
  - No return statement is required
- Example:
  - Line 4: defines the return type as void

You can find the `deleteABContactSecondaryAddresses()` PCF Method in `ABContactAddressesLDV.pcf`.

A return statement is not require when no return type is defined in the function signature.

# Where can you declare Gosu methods?



- PCFs
  - Method can be used only within that PCF
  - Discussed in this lesson



- Enhancements
  - Method associated to given type (such as an entity)
  - Can only be used by instances of that type
  - Discussed in "Enhancements" lesson



- Classes
  - Method associated with given class
  - If declared as static methods, can be used anywhere
  - Discussed in "Gosu Classes" lesson

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

8

GUIDEWIRE

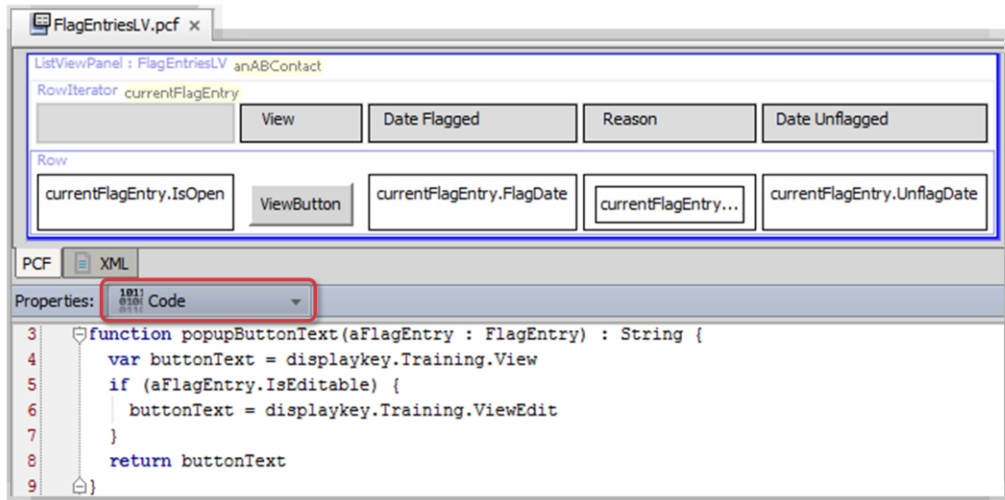
Class methods can also be non-static. In this case, the method can only be called from an instance of the class. Typically, configuration developers do not write an extensive amount of non-static class methods. Integration developers, however, typically do write both static and non-static class methods.



## Lesson outline

- Overview of Gosu methods
- Creating PCF methods
- Common use cases for PCF methods

# PCF methods



- A **PCF method** is a method declared on a given PCF file's Code tab
  - Properties of PCF file or its widgets can call the PCF Method

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

10

GUIDEWIRE

Any required variable (declared on the Required Variables tab) or non-required variable (declared on the Variables tab) is inherently known to all PCF methods. You do not need to pass a variable declared on either of these tabs to the PCF method. For example, if the method above required data from the root anABContact object, the method could simply reference anABContact. The object would not need to be included in the method declaration. PCF methods are available only within the scope of the PCF in which they are declared. They behave as if they have the private access modifier. You can explicitly add an access modifier, but this does not change the inherent scope of the PCF method. You can see this example in FlagEntriesLV in Studio.

Locations such as pages and popups and atomic widgets such as inputs, cells, and buttons have properties that can execute methods.

Container widgets such as detail view panels, list view panels, card view panels, and list detail panels do not have any properties that can execute methods, however.

## Lesson outline

- Overview of Gosu methods
- Creating PCF methods
- Common use cases for PCF methods

## Calling PCF methods

- Syntax:  
**functionName(inputParameters)**
- Call method from both location and widget properties
- Common use cases for widgets includes what happens...
  - For a button click
  - When rendering a widget
  - When a widget's value changes
- Common use cases for locations include when...
  - A user navigates to that location
  - Data modifications in that location are committed
- This Lesson focuses on defining the method in the PCF, hence the term, PCF Method

Although any method can be referenced, the remainder of this lesson focuses on PCF methods.

## Use case 1: When widget is clicked

The screenshot illustrates the configuration of a toolbar button. On the left, a 'Delete Secondary Addresses' button is highlighted in a red box. A red arrow points from this button to the 'ToolbarButton properties' panel on the right. This panel shows the 'action' property set to 'deleteABContactSecondaryAddresses()'. Another red arrow points from this action property to the 'ListDetailPanel properties: Code tab' at the bottom, which displays the implementation of the 'deleteABContactSecondaryAddresses()' method. The method iterates through the 'SecondaryAddresses' array and removes each address using the 'removeAddress()' method.

Addresses

Update Cancel

Add Remove Delete Secondary Addresses

<input type="checkbox"/>	Primary	Address Type	Address
<input checked="" type="checkbox"/>	Home	444 Ave Maria Stairway St, San Francisco, CA	
<input type="checkbox"/>	Other	9032 Flave Ste	
<input type="checkbox"/>	Other	444 Ave Maria	
<input type="checkbox"/>	Other	55 Straight Sta	

ToolbarButton properties

Properties

ToolbarButton: DeleteSecondaryAddresses

Basic properties

action deleteABContactSecondaryAddresses()

id\* DeleteSecondaryAddresses

label displaykey.Training.DeleteSecondaryAddresses

ListDetailPanel properties: Code tab

Properties: Code

```
/* This function deletes all addresses in the root object's SecondaryAddresses array. */
function deleteABContactSecondaryAddresses(): void {
    for (currentAddress in anABContact.SecondaryAddresses) {
        anABContact.removeAddress(currentAddress)
    }
}
```

- Toolbar button action attribute specifies PCF Method to execute

In the slide example, the ToolbarButton properties specifies an action property to use the deleteABContactSecondaryAddresses() method. The ABContactAddressesLDV is a ListDetailPanel that defines the PCF Method named deleteABContactSecondaryAddresses(). The method removes an address for given contact using the removeAddress() method for a contact.

## Widget availability

- Available property expression determines if widget is available for user interactions
  - Grayed out mean
- Toolbar button example:
  - Only evaluates to true when in edit mode AND when there are secondary addresses

The figure consists of three screenshots of a contact management interface, illustrating widget availability based on the 'Available' property expression.

**Top Screenshot (Available=true):** The 'Delete Secondary Addresses' button is active (blue). The interface shows a table with two rows of addresses. The first row is selected.

<input type="checkbox"/>	Primary	Address Type	
<input checked="" type="checkbox"/>		Home	444 Ave Maria Stairway St, San Francisco, CA
<input type="checkbox"/>		Other	9032 Flave Steep Str, San Francisco, CA 94104

**Middle Screenshot (Available=false):** The 'Delete Secondary Addresses' button is grayed out. The interface shows a table with one row of addresses. The first row is selected.

<input type="checkbox"/>	Primary	Address Type	
<input checked="" type="checkbox"/>	X	Home	444 Ave Maria Stairway St, San Francisco, CA

**Bottom Screenshot (Available=false):** The 'Delete Secondary Addresses' button is grayed out. The interface shows a table with one row of addresses. The first row is selected.

<input type="checkbox"/>	Primary	Address Type	
<input checked="" type="checkbox"/>		Home	444 Ave Maria Stairway St, San Francisco, CA

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

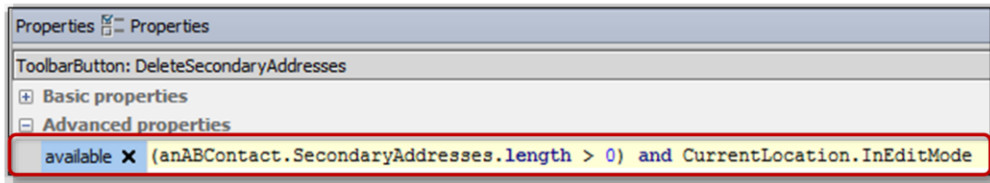
14

G U I D E W I R E

When the available property expression is true, the widget is available for user interactions. When the available property expression is false, the widget is unavailable for user interactions.

In the slide example, the Delete Secondary Addresses button is available only when the page is in edit mode AND only if the contact has secondary addresses.

# Configuring widget availability



- If available property expression evaluates to true, widget is available for user interaction (toolbar button is clickable)
  - Default value for property is true
  - Evaluated only when action property is not null
- **CurrentLocation.InEditMode**
  - Control availability based on edit mode of location
  - Returns true if current location is in edit mode
  - Returns false if current location is in read-only mode

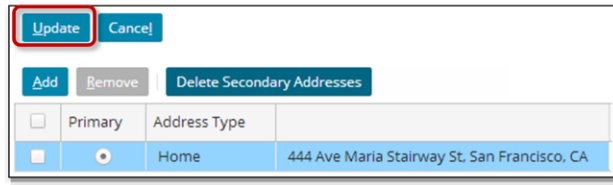
You can also use a combination of widgets and Gosu to affect whether or not a page is editable. Do the following to create a custom button that puts the current location into edit mode just like an edit button:

1. In Studio, add a Toolbar Button widget to the desired toolbar.
2. Click the Properties tab.
3. Specify `CurrentLocation.startEditing()` for the action property.
4. Configure other properties of the widget as desired.
5. Reload PCFs

## Changing data in PCF methods

- Example:

- Deletion of addresses committed only when user clicks Update



The screenshot shows a user interface for managing addresses. At the top, there are buttons for 'Update' (highlighted with a red box), 'Cancel', 'Add', 'Remove', and 'Delete Secondary Addresses'. Below these buttons is a table with columns for selection, 'Primary', 'Address Type', and the address details.

<input type="checkbox"/>	Primary	Address Type	
<input checked="" type="checkbox"/>	<input checked="" type="radio"/>	Home	444 Ave Maria Stairway St, San Francisco, CA

- When in edit mode, if a PCF method changes data, the data changes commits only when the user clicks Update
  - When a PCF method changes data, those changes are not automatically written to database
  - If a PCF method must change data while in read-only mode, the method code must commit the changes explicitly

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

16

G U I D E W I R E

A bundle is a collection of Gosu objects that corresponds to a database transaction. Changes to data in the bundle's objects either succeed as a unit or fail as a unit. In order to manually commit data, the code must manipulate the current bundle. For more information, consult the Gosu Reference Guide.

A PCF method that is executed in read-only mode can modify data and not manually commit it. In other words, data changes to the Gosu objects are not saved to the database.

Locations have `startEditing()` and `commit()` methods. Guidewire recommends that you NOT use these methods in combination to manually commit data for a location in read-only mode. This is because the commit could throw an error if for some reason the data cannot be saved to the database resulting in the location unexpectedly in edit mode with the data unsaved.



## Use case 2: Input or cell button label

The screenshot illustrates the configuration of a button cell in a table. At the top, a table titled "Flag Entries" has columns: View, Date Flagged, Reason, and Date Unflagged. The first row shows a warning icon, a "View/Edit" button, the date "11/18/2013", and the reason "No email address for this contact." A red box highlights the "View/Edit" button, with a red arrow pointing to the "ButtonCell properties" panel below. This panel shows the "Basic properties" for a "ButtonCell: ViewButton". The "action" is "FlagEntryPopup.push(currentFlagEntry)", "id\*" is "ViewButton", "label" is "displaykey.Training.View", and "value\*" is "popupButtonText(currentFlagEntry)". A red arrow points from the "value\*" field to the "ListViewPanel properties: Code tab" panel. This panel shows a code snippet for the "popupButtonText" function.

Flag Entries

View	Date Flagged ↓	Reason	Date Unflagged
⚠ View/Edit	11/18/2013	No email address for this contact.	

ButtonCell properties

Properties: ButtonCell: ViewButton

Basic properties

- action: FlagEntryPopup.push(currentFlagEntry)
- id\*: ViewButton
- label: displaykey.Training.View
- value\*: popupButtonText(currentFlagEntry)

ListViewPanel properties: Code tab

```
3 function popupButtonText(aFlagEntry : FlagEntry) : String {  
4   var buttonText = displaykey.Training.View  
5   if (aFlagEntry.IsEditable) {  
6     buttonText = displaykey.Training.ViewEdit  
7   }  
8   return buttonText  
9 }
```

- Value attribute species the text to display on a button
- PCF Method can set value

For input buttons and cell buttons, the value attribute specifies the text to display on the button. This attribute is not available for toolbar buttons.

Note that the label field for cells contains the label for the column, not for the individual cells.

## Use case 3: Targeted Post On Change

The diagram illustrates the 'Targeted Post On Change' functionality. It shows two 'Flag Entry' forms. The first form has a 'Resolution' field with the value 'Only receives snail mail'. A red arrow points from this field to a second 'Flag Entry' form. In the second form, the 'Unflagged By' field is updated to 'Super User' and the 'Date Unflagged' field is updated to '12/12/2013', both indicated by red arrows.

Flag Entry <a href="#">Return to Summary</a>	
<a href="#">Update</a>	<a href="#">Cancel</a>
Date Flagged	12/06/2013
Reason	No email address for this contact.
Resolution	Only receives snail mail
Unflagged By	
Date Unflagged	

Flag Entry <a href="#">Return to Summary</a>	
<a href="#">Update</a>	<a href="#">Cancel</a>
Date Flagged	12/06/2013
Reason	No email address for this contact.
Resolution	Only receives snail mail
Unflagged By	Super User
Date Unflagged	12/12/2013

- Targeted Post On Change onChange property method
  - Widget value change triggers changes to other widgets prior to the value being saved
  - Discussed in detail in the Partial Page Update lesson

In order to improve data entry efficiency and reduce page refresh, you can configure input widgets and cell widgets to dynamically react to user input using targeted Post On Change. With targeted Post On Change enabled, it is possible to update field values on the page and/or re-render the entire layout or a partial page layout.

Post On Change is a Boolean tab property in Guidewire Studio and allows you to define three additional properties: `disablePostOnEnter`, `onChange`, and `target`. The `PostOnChange` property tab contains the `Enable targeted Post On Change` checkbox. When checked, the three additional properties are editable.

## Use case 4 : Navigating to locations

The screenshot shows a 'Search Results' table on the left with a 'Delete' button and a search input. The table has columns for checkboxes, 'Name', and a search result 'William Andy'. A red box highlights 'William Andy' and a red arrow points to the 'History' tab in the 'Person: William Andy' detailed view on the right. The 'History' tab shows a list of events with columns for Date, Event Type, and Description.

Date ↑	Event Type	Description
11/18/2013	Created	Contact William Andy created
04/05/2014	Viewed	Super User viewed this contact.
04/05/2014	Viewed	Super User viewed this contact.
04/05/2014	Viewed	Super User viewed this contact.
04/10/2014	Viewed	Super User viewed this contact.
04/10/2014	Viewed	Super User viewed this contact.
04/16/2014	Viewed	Super User viewed this contact.

- Location PCF Method creates a history entry
  - Contact created
  - User viewed this contact

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

19

G U I D E W I R E

The location properties that can call PCF methods include:

- **afterEnter**: Executed immediately after application enters location
- **beforeCancel**: Executed before application cancels out of edit mode
- **afterCancel**: Executed after application cancels out of edit mode
- **beforeCommit**: Executed before application commits out of edit mode
- **afterCommit**: Executed after application commits out of edit mode
- **afterReturnFromPopup**: Executed after application returns to location from popup

Locations such as pages and popups and atomic widgets such as inputs, cells, and buttons have properties that can execute methods.

Container widgets such as detail view panels, list view panels, card view panels, and list detail panels do NOT have any properties that can execute methods.

## Lesson objectives review

- You should now be able to:
  - Describe the different ways methods can be written
  - Create PCF methods
  - Implement common use cases for PCF methods

## Review questions

1. Is it required or optional for a method to:
  - a) Receive input parameters?
  - b) Return a value?
2. Can an atomic widget reference:
  - a) A PCF method declared in the same file?
  - b) A PCF method declared in another PCF file?
3. What does a widget's "available" property control?

### Answers

1a) Optional

1b) Optional; if a method returns no value, its return type should be listed as "void".

2a) Yes

2b) No

3) This property determines if the widget can be clicked or not.

# Notices

**Copyright © 2001-2014 Guidewire Software, Inc. All rights reserved.**

Guidewire, Guidewire Software, Guidewire ClaimCenter, Guidewire PolicyCenter, Guidewire BillingCenter, Guidewire Reinsurance Management, Guidewire ContactManager, Guidewire Vendor Data Management, Guidewire Client Data Management, Guidewire Rating Management, Guidewire InsuranceSuite, Guidewire ContactCenter, Guidewire Studio, Guidewire Product Designer, Guidewire Live, Guidewire DataHub, Guidewire InfoCenter, Guidewire Standard Reporting, Guidewire ExampleCenter, Guidewire Account Manager Portal, Guidewire Claim Portal, Guidewire Policyholder Portal, ClaimCenter, BillingCenter, PolicyCenter, InsuranceSuite, Gosu, Deliver Insurance Your Way, and the Guidewire logo are trademarks, service marks, or registered trademarks of Guidewire Software, Inc. in the United States and/or other countries.

All other trademarks are the property of their respective owners.

**This material is confidential and proprietary to Guidewire and subject to the confidentiality terms in the applicable license agreement and/or separate nondisclosure agreement.**

This file and the contents herein are the property of Guidewire Software, Inc. Use of this course material is restricted to students officially registered in this specific Guidewire-instructed course, or for other use expressly authorized by Guidewire. Replication or distribution of this course material in electronic, paper, or other format is prohibited without express permission from Guidewire.

Guidewire products are protected by one or more United States patents.