



# List View Panels



January 30, 2014

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

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 functionality of list view panels
  - Create a new list view panel
  - Create and modify row iterator, row, and cell widgets
  - Reference list view panels

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

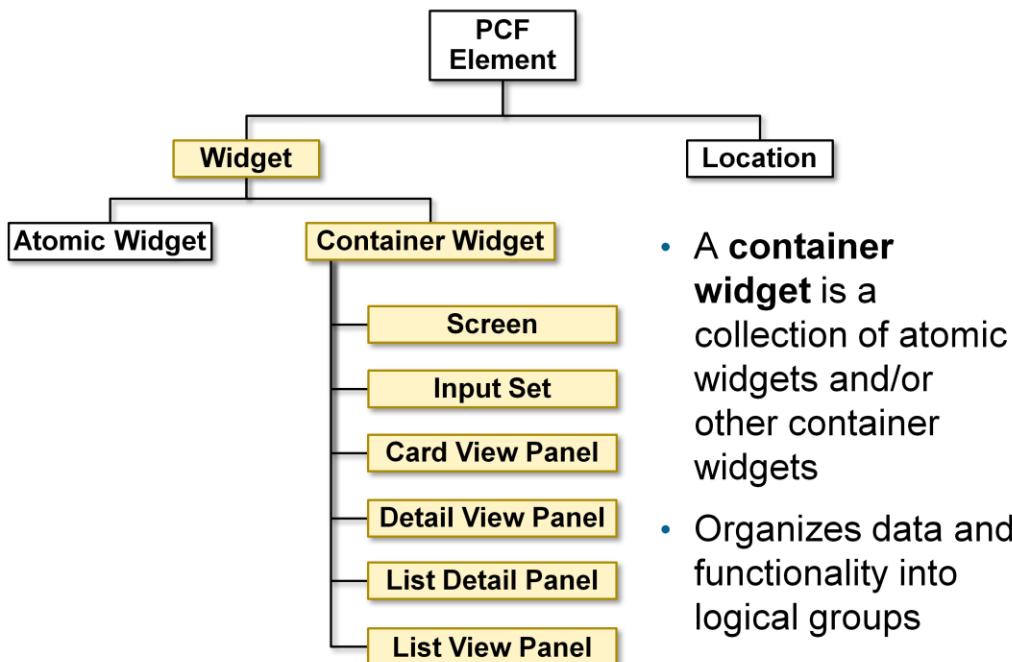
G U I D E W I R E



## Lesson outline

- List view panel fundamentals
- Create list view panels
- Reference list view panels

# Container widgets



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

4

G U I D E W I R E

Container widgets hold other widgets. Each one can be defined either in its own file or as a child container within some other PCF element file.

Both Widget and Location are conceptual representations in this diagram. There are no <Widget /> or <Location /> elements. Similarly, both Atomic Widget and Container Widget are conceptual representations. There are no <Atomic Widget /> or <Container Widget /> elements.

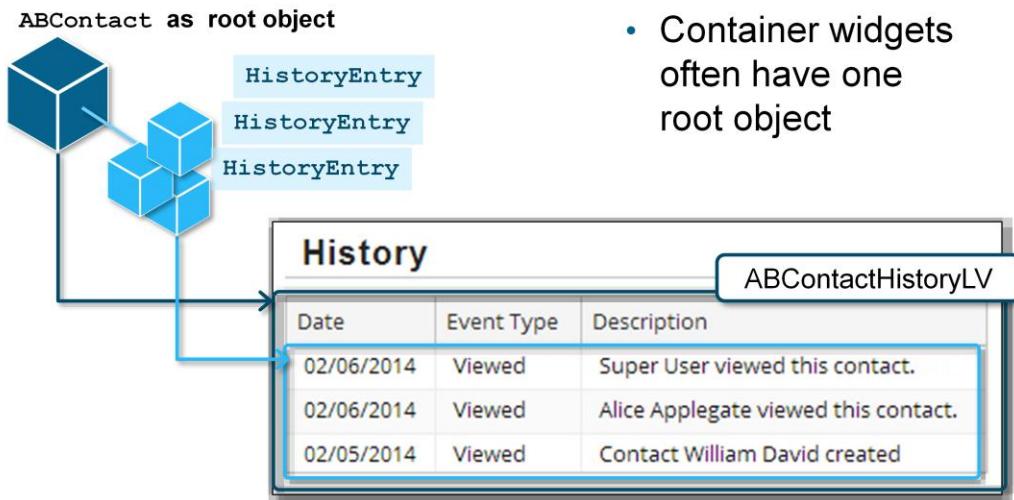
The PCF object model is container-based. Each screen element is modeled as an object, which may contain other objects. The hierarchical structure simplifies the task of locating and modifying visual elements. Furthermore, each element can be declared as an independent and therefore reusable element.

# List view panels

History		
Date	Event Type	Description
02/06/2014	Viewed	Super User viewed this contact.
02/06/2014	Viewed	Alice Applegate viewed this contact.
02/05/2014	Viewed	Contact William David created

- A **list view panel** is a container widget that often displays a set of rows that are related to one object or one query
- Users can view and, in some cases, edit the data

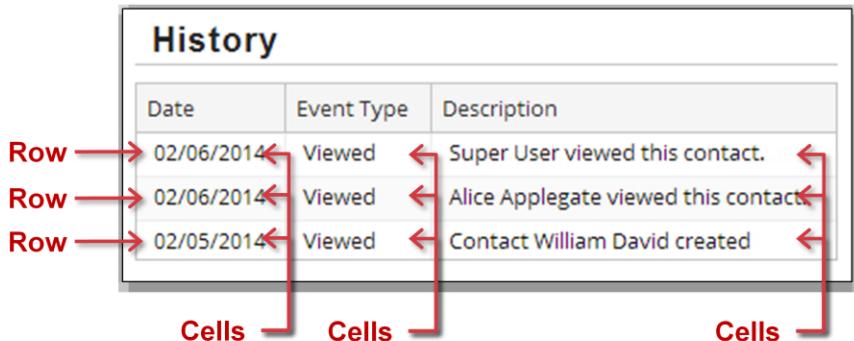
## List view panel root objects



- Container widgets often have one root object

- A **list view panel** can display an array of objects that are associated with the root object

## List view panel anatomy



- Cell widgets display individual fields of data
- Row widget organize cells
- Entire structure generated by row iterator

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

7

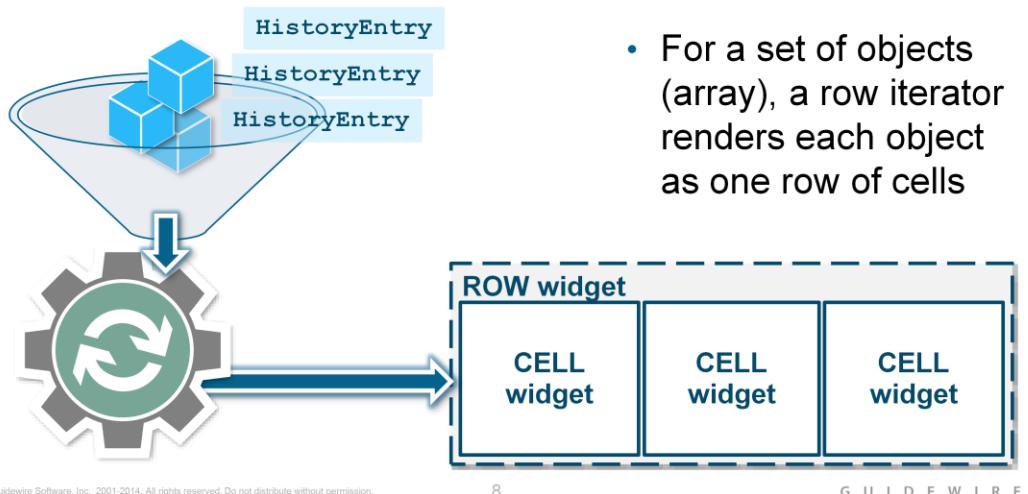
G U I D E W I R E

Row widgets organize the layout of the cells for an object instance when the row iterator is manipulating that instance. Row widgets do not perform any "navigation" within the collection of object instances. Adding a second row widget would not display two different objects when the page is rendered, but would arrange the cells for each instance across two rows. The generated markup for a row widget is a <TR> HTML tag.

Row iterators are discussed in the next slide.

## Row iterator

- An **iterator** is a widget that takes a set of items and performs the same set of actions for each member
- A **row iterator** is an iterator for list view panels



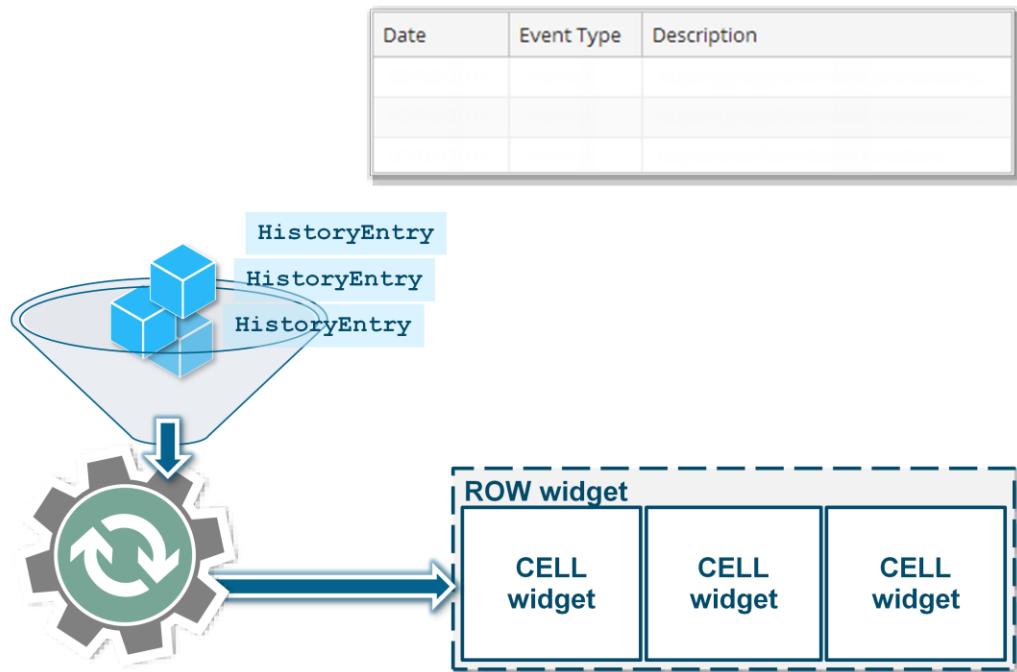
8

G U I D E W I R E

The PCF architecture has several different types of iterators, including:

- Menu item iterators, which take a set of objects and generate one menu item for each.
- Panel iterators, which take a set of objects and generate one panel (typically, one detail view) for each.
- PolicyCenter coverage iterators, which take a set of coverages associated with a covered item (such as a vehicle) and generate one coverage input for each coverage.

## Row iterator: no objects processed



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

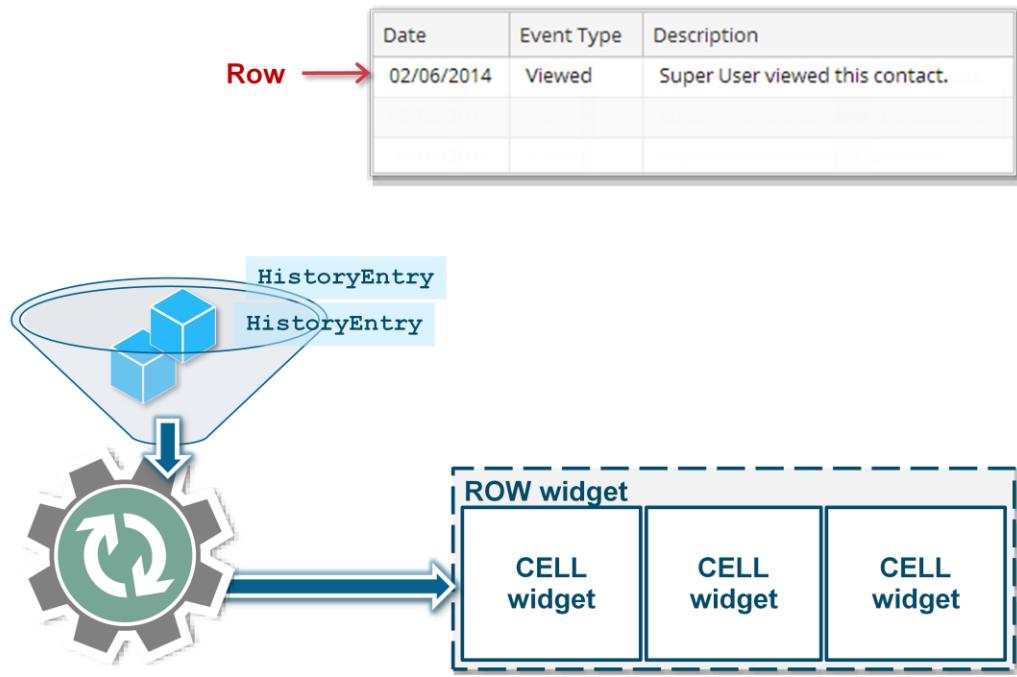
9

G U I D E W I R E

The PCF architecture has several different types of iterators, including:

- Menu item iterators, which take a set of objects and generate one menu item for each.
- Panel iterators, which take a set of objects and generate one panel (typically, one detail view) for each.
- PolicyCenter coverage iterators, which take a set of coverages associated with a covered item (such as a vehicle) and generate one coverage input for each coverage.

## Row iterator: first object processed



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

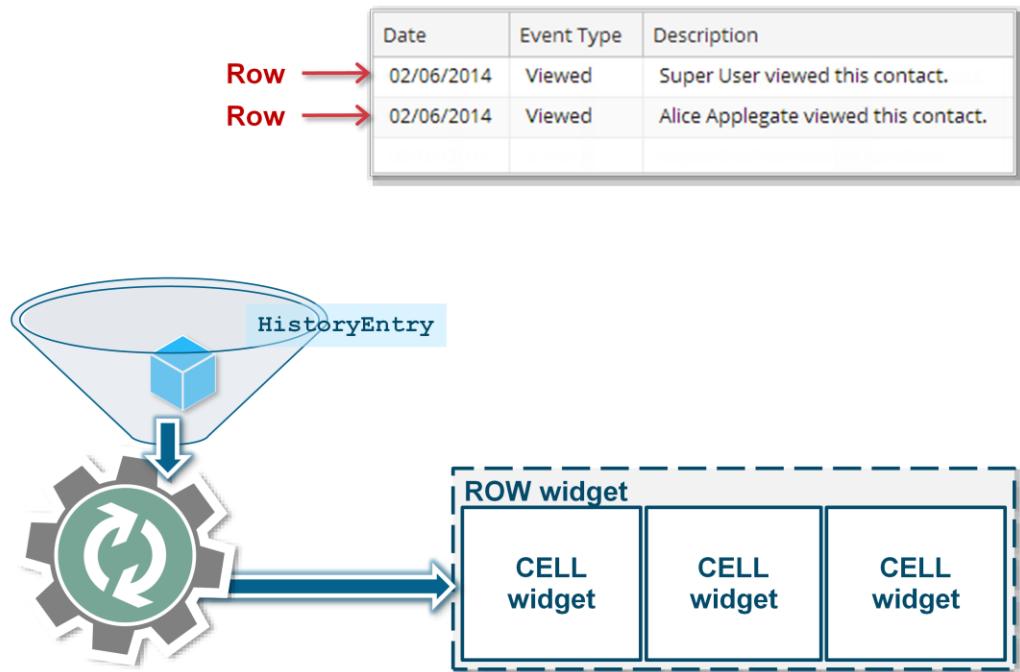
10

G U I D E W I R E

The PCF architecture has several different types of iterators, including:

- Menu item iterators, which take a set of objects and generate one menu item for each.
- Panel iterators, which take a set of objects and generate one panel (typically, one detail view) for each.
- PolicyCenter coverage iterators, which take a set of coverages associated with a covered item (such as a vehicle) and generate one coverage input for each coverage.

## Row iterator: next object processed



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

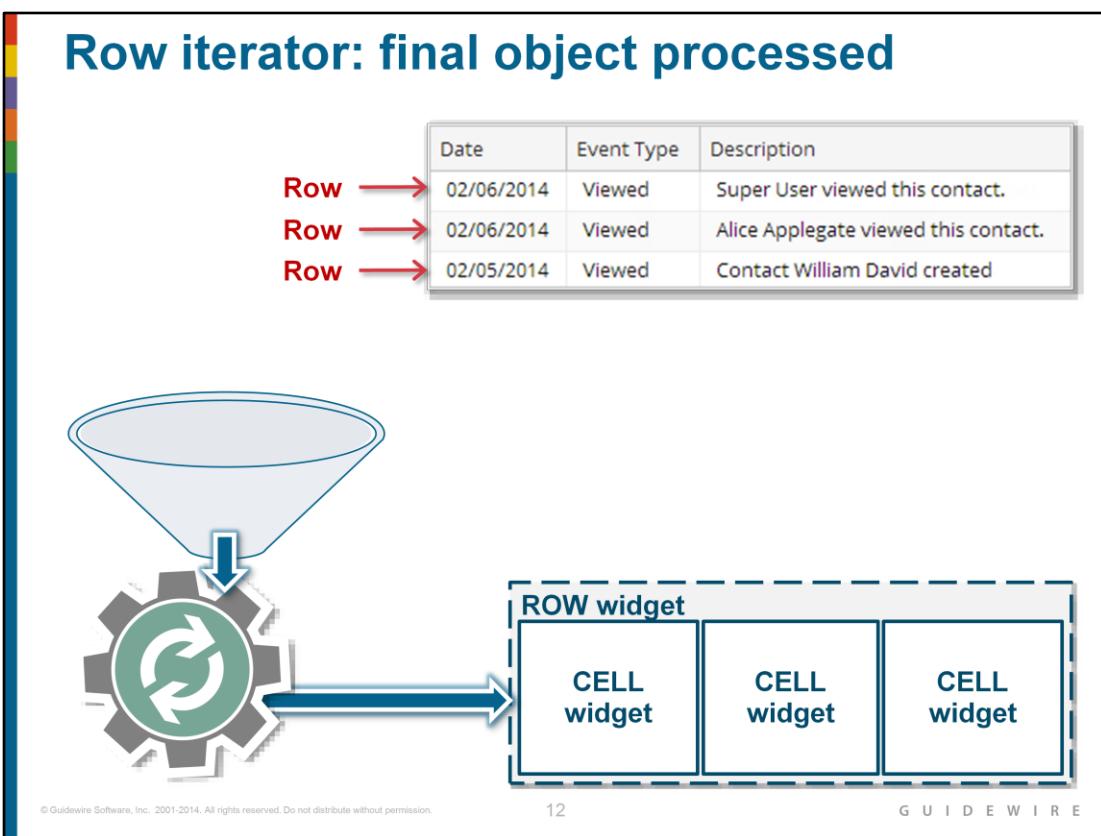
11

G U I D E W I R E

The PCF architecture has several different types of iterators, including:

- Menu item iterators, which take a set of objects and generate one menu item for each.
- Panel iterators, which take a set of objects and generate one panel (typically, one detail view) for each.
- PolicyCenter coverage iterators, which take a set of coverages associated with a covered item (such as a vehicle) and generate one coverage input for each coverage.

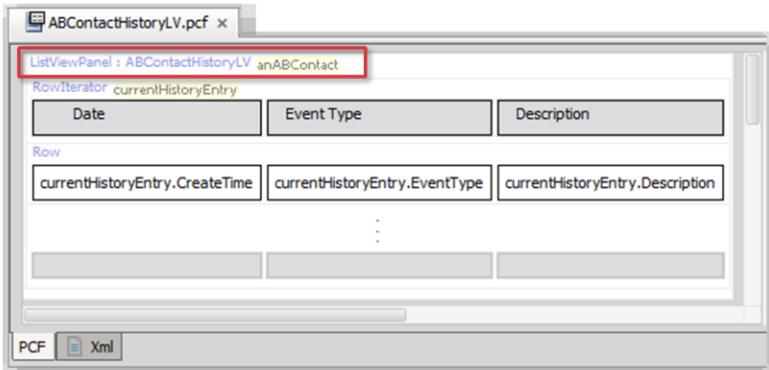
## Row iterator: final object processed



The PCF architecture has several different types of iterators, including:

- Menu item iterators, which take a set of objects and generate one menu item for each.
- Panel iterators, which take a set of objects and generate one panel (typically, one detail view) for each.
- PolicyCenter coverage iterators, which take a set of coverages associated with a covered item (such as a vehicle) and generate one coverage input for each coverage.

# Reusable containers



- PCF file contains a top-level container

- Other PCF files can reference the reusable container
- If the container is likely to be needed in multiple places, create as PCF file for container!

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

13

G U I D E W I R E

An inline container is not reusable. In the slide example, ABContactHistoryLV is a top-level container. Other containers can therefore reference it. In TrainingApp, the ABContactHistoryPage contains a Screen with a Panel Ref that references ABContactHistoryLV.

## File and widget

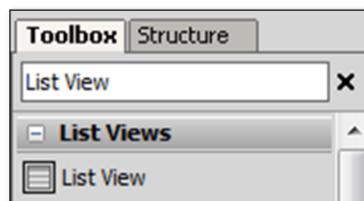
### List View Panel PCF file

- <ListViewPanel/> is a top-level PCF element
- File name ends with LV
- Define root object



### ListViewPanel widget

- Widget is defined in a Screen, Card View Panel, or a List Detail Panel



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

14

G U I D E W I R E

It is possible to define a variable for a ListViewPanel widget. In many cases, however, a ListViewPanel widget inherits the root object associated with its parent.

# Reusability and inline

## Reusability

- List View Panel is PCF file

- Ideal for multiple references
  - Other PCF files use a reference widget to reference the reusable container

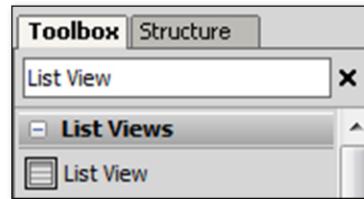
## Inline

- Defined as widget

- Single instance usage
  - Not possible to reference in another container



List View Panel  
PCF



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

15

G U I D E W I R E



## Lesson outline

- List view panel fundamentals
- Create list view panels
- Reference list view panels

# Steps to create a List View Panel PCF

1. Create the List View Panel PCF file
2. Specify the required variables
3. Specify additional properties
4. Add a row iterator
5. Add a row
6. Add cell widgets
7. Deploy PCFs

\* Slides do not cover the details of creating an inline List View Panel. See notes.

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

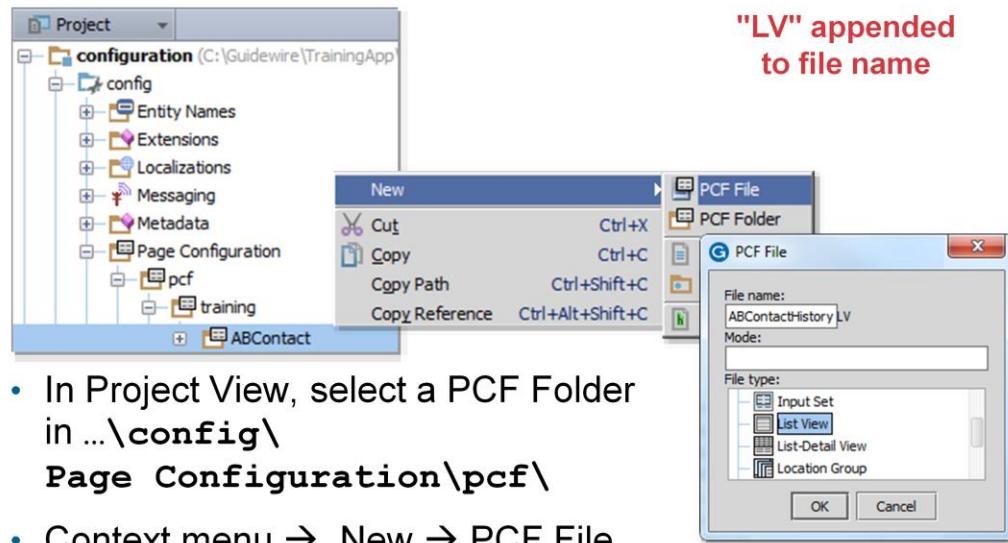
17

G U I D E W I R E

## Inline list view panel

1. Add the ListViewPanel widget to the parent container
2. Optionally specify additional properties
4. Add a row iterator widget
5. Add a row widget
6. Add cell widgets
7. Deploy PCFs

# Step 1: Create a list view panel PCF



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

18

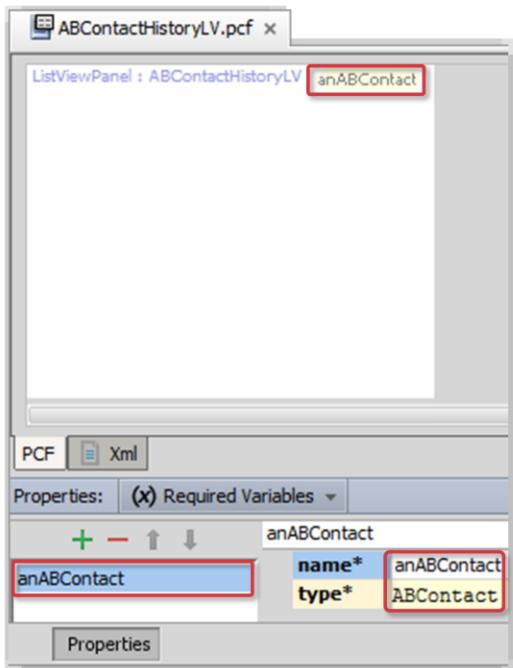
G U I D E W I R E

The initial list view panel always displays an error because every list view panel needs a row element. A newly created list view panel is empty.

## Inline list view panels

To create an inline list view panel, find the ListViewPanel widget in the PCF Editor toolbox. Drag the ListViewPanel widget onto an existing screen, list detail panel, or card view panel.

## Step 2: Specify required variable(s)



- Required Variables tab
  - Defines data object variable name and type
  - Example:  
anABContact is of type ABContact
- Object data can be
  - Data backed (database)
  - Virtual property
- Container data comes from defined variable object(s)
  - Typically, at least one variable
  - Not required



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

19

G U I D E W I R E

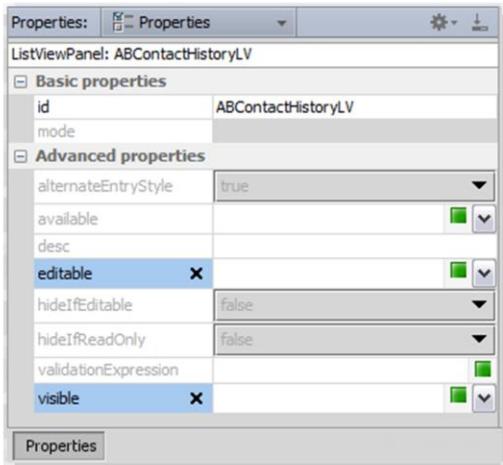
Most container widgets have at least one required object that contains data fields. One way to think of this is that there is at least one root object for a given container. Root objects must be specified on the list view panel's Required Variables tab.

The initial list view panel always displays an error because every list view panel needs a row element. A newly created list view panel is empty. In the slide example, the screenshot has been modified to not show the error.

### Inline list view panels

You do not need to specify root objects for inline list view panel. Because an inline list view panel can have only one parent container, it automatically inherits the root objects of its parent.

## Step 3: Specify additional properties



- **Editable**
  - Makes container and children widget editable
  - Not all container widgets have an explicit editable property
- **Visible**
  - Shows container and all children
  - If false, then hidden

- Blank is default and means that the property inherits the value from parent container or location
- If not defined in hierarchy, then true

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

20

G U I D E W I R E

There are two states that a list view panel can be in: edit mode and read-only mode. If the editable property for a list view panel is true (or blank, which defaults to true), then the list view panel can be put into either read-only mode or edit mode. If the editable property for a list view panel is false, then the list view panel cannot be put into edit mode. It is always in read-only mode.

### Inline list view panels

This step is the same for standalone and inline list view panels.

## Step 4: Add a row iterator

The screenshot shows the Guidewire PCF editor interface. On the left is the canvas with a red-bordered 'RowIterato...' component. To its right is the 'Toolbox' panel, which has 'Row Iterator' selected under 'List Views / Row and Cell Layout'. Below the canvas is the 'Properties' window, specifically the 'RowIterator' tab. It displays three properties in the 'Basic properties' section:

editable*	false
elementName*	currentHistoryEntry
value*	anABContact.HistoryEntries

A red arrow points from the 'Toolbox' panel to the 'RowIterato...' component on the canvas.

- Add RowIterator widget to list view panel
- editable= false
  - child cells not editable
- elementName= current HistoryEntry
  - Symbol name for cell object
- anABContact. HistoryEntries
  - root object array for row iterator

In the slide example, the RowIterator has already been placed in the ListViewPanel in the PCF editor canvas. The Properties window shows the Properties tab. Some attributes in the Basic properties have been removed. An asterisk denotes a required property. RowIterator widgets have three required properties: editable, elementName, and value.

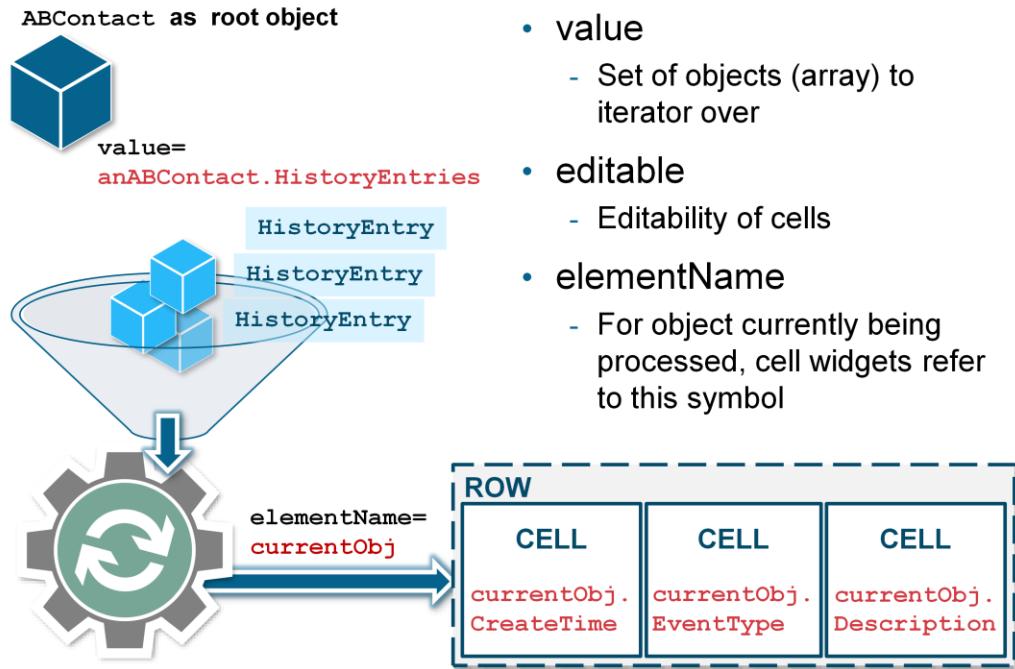
In the slide example...

- Editable is false—the cell value cannot be edited.
- The name of the symbol used to reference the object currently being processed is currentHistoryEntry.
- The object set to be processed is anABContact's HistoryEntries array.

### Inline list view panels

This step is the same for standalone and inline list view panels.

# Row Iterator required properties



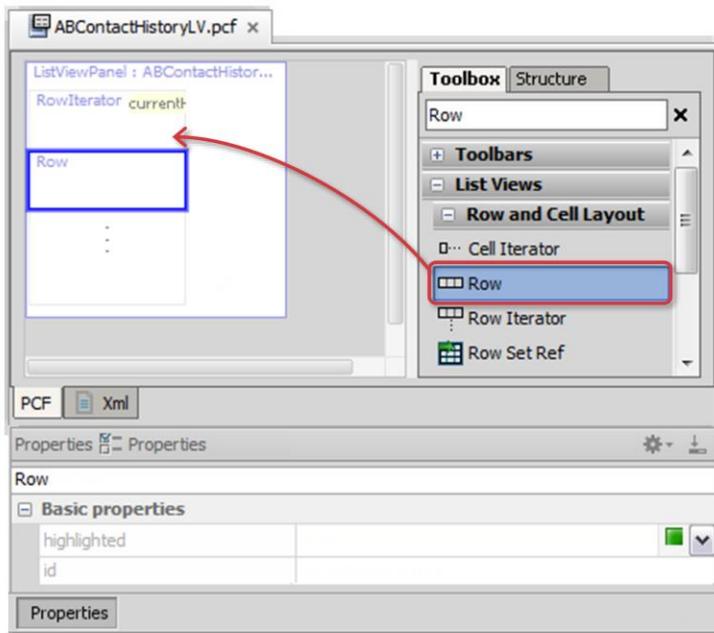
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

22

G U I D E W I R E

Each cell widget inside the row widget is used to render data for multiple objects. For example, the first cell in the diagram above renders the CreateDate field for all three HistoryEntry objects in the array. The cell widget must therefore be bound to an abstract name that references each object in the array one at a time. The Element Name property of the row iterator defines what this name is. The value you select for the Element Name property is arbitrary, but whatever value you select must be used by the cell widgets in the row iterator. To improve readability, use element names such as "current<object>" or "this<object>".

## Step 5: Add a row



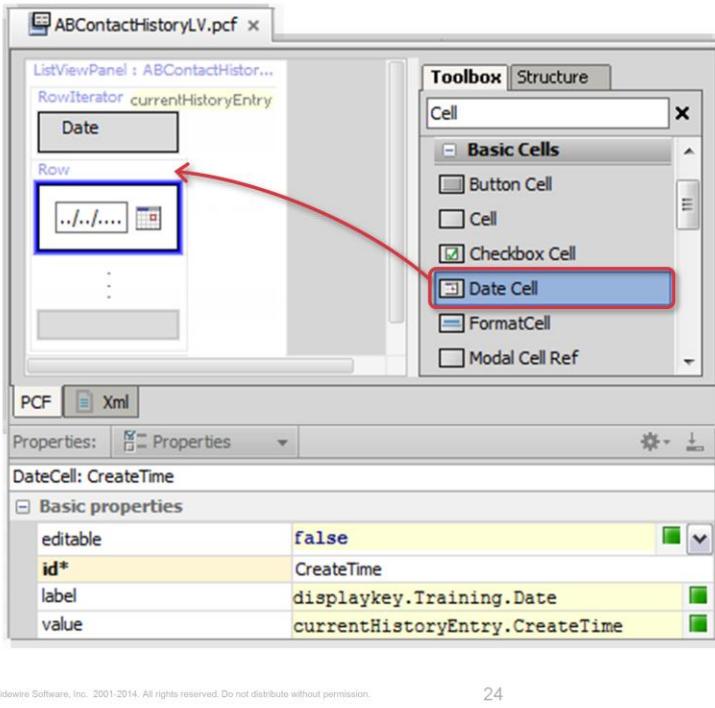
- Add to RowIterator widget
- Row is required
- Not necessary to define properties in most cases for row

In the slide example, the Row has already been placed in the Row Iterator in the PCF editor canvas. The Properties window shows the Properties tab.

### Inline list view panels

This step is the same for standalone and inline list view panels.

## Step 6: Add cell widgets



- Add cell widgets to row
- **editable**
  - default is false
- **Id**
  - required, but only unique to row
- **label**
  - specifies column header
- **value**
  - reference to iterator's element name and object property

In the slide example, the DateCell widget has already been placed in the Row in the PCF editor canvas. The DateCell widget offers additional formatting options for dates and times.

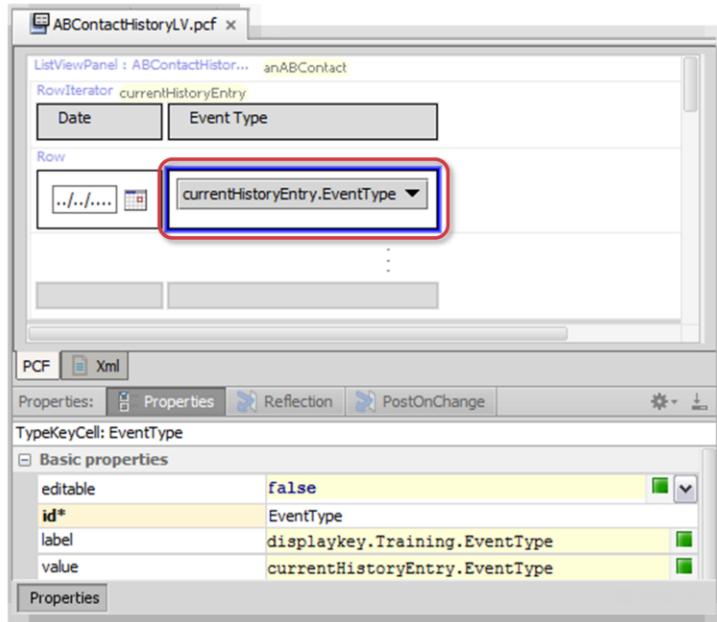
The Properties window shows the Properties tab. Some attributes in the Basic properties have been removed. An asterisk denotes a required property. Cell input widgets require a unique ID value. However, this value need only be unique to the inputs in the Row.

A cell widget is inside row iterator, which is used to process multiple rows. A cell's value property must reference row iterator's element name. The label property specifies the label for column header.

### Inline list view panels

This step is the same for standalone and inline list view panels.

## Example: Second cell



- Add cell to row
  - Order determines default column order
- Cell widget is generic widget that resolves the type
- Use the best widget for data type
- TypeKeyCell widget
  - Binds to a typekey object value

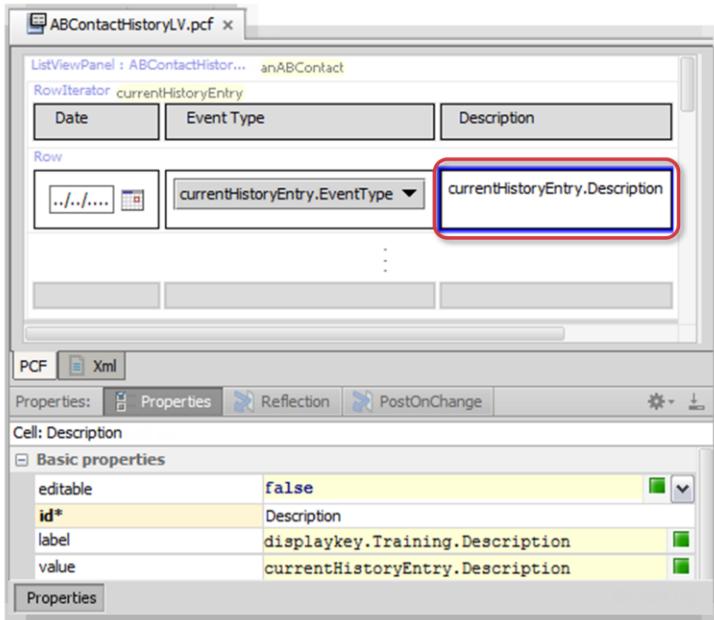
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

25

G U I D E W I R E

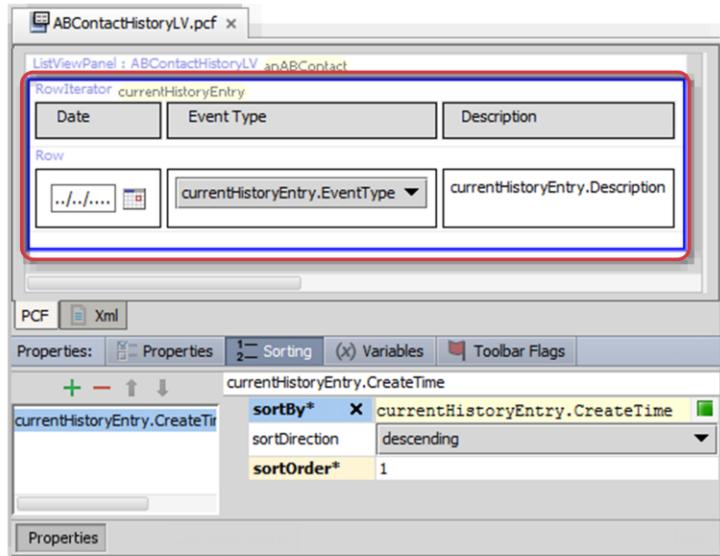
In the slide example, the EventType Cell widget has already been placed in the Row in the PCF editor canvas. The Properties window shows the Properties tab. Some attributes in the Basic properties have been removed. An asterisk denotes a required property. Cell input widgets require a unique ID value. However, this value need only be unique to the inputs in the Row.

## Example: Third cell



- Add cell to row
  - Order determines default column order
- Cell widget is generic widget
  - Use the best widget for the field data type
  - Default is to use TextCell

# Row iterator sorting



- To define the default behavior of the row iterator sort, use the Sorting properties tab
  - sortBy
    - Specify field
  - sortDirection
    - Specify ascending or descending
  - sortOrder
    - Specify sort precedence

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

27

GUIDEWIRE

The row iterator can sort the objects (elementName property) in the array (value property). To define the default sorting criteria, use the row iterator Sorting properties tab to specify the sorting criteria.

For each sort criterion, you must specify the sortBy value, the direction of the sort (ascending or descending), and precedence of the criteria (the sortOrder value). It is possible to have multiple sort criterion. If a column has a sort criteria with sort order 1 (and the sortBy column is visible in the list), then it is initially rendered with an up-facing or down-facing arrow head next to the label. The direction of the arrow reflects the sortDirection value. Any criteria with a sort order of 2 or greater are rendered with an up-facing or down-facing double arrow head next to the header label.

For properties that are typecode fields, rows are sorted based on the priority and then name of the typecodes. For example, if a list of buildings includes a "BuildingType" column with possible values of "residential - single dwelling" (priority 10) and "residential - multiple dwelling" (priority 20) and the column has a defined ascending sort order, all "residential - single dwelling" buildings will be listed before any "residential - multiple dwelling" buildings.

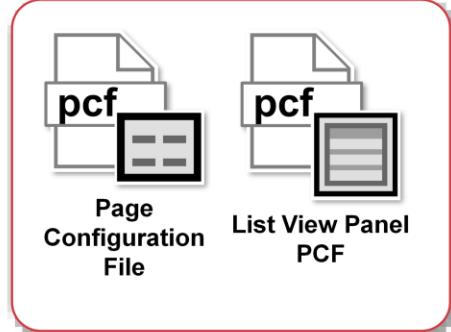
You can create sort criterion that reference values not displayed in the list. For example, a list of history entries could be sorted by create date even though the create date isn't displayed in the list.

In order for a user to override the defined sorting criteria for a row iterator, you must configure the enableSort property for a given cell widget. If true, a user can sort the values in the column by clicking the column header.

## Step 7: Deploy PCFs

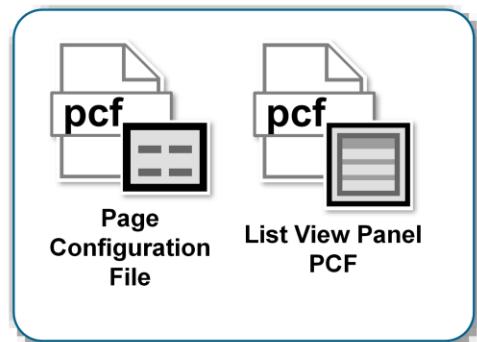
### Restart Server

- PCFs read at server startup



### Reload PCFs

- ALT+SHIFT+L
  - Internal debug tools enabled
- Internal Tools
  - Reload → Reload PCF Files



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

28

G U I D E W I R E

It is also possible to reload PCF files using the Guidewire API and/or internal server tools. The Reload PCF command can be found on the Reload page in Internal Tools. To access Internal Tools, you must log in as an administrator user, e.g., su/gw. Then, use ALT+SHIFT+T. In the tab bar, select Internal Tools → Reload. On the Reload page, click the Reload PCF Files button. The Reload PCF Files button calls the static method `gw.api.tools.InternalToolsUtil.reloadPCFs()`.

### Inline list view panels

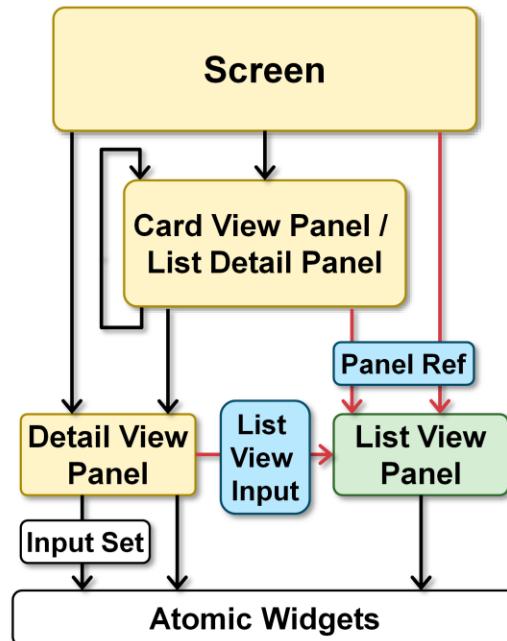
This step is the same for standalone and inline list view panels. However, for inline list view panels, only the location PCF files get deployed.



## Lesson outline

- List view panel fundamentals
- Create list view panels
- Reference list view panels

# Referencing list view panels



- A reference widget in a parent container references a PCF File as an embedded child container
- A PanelRef widget can reference:
  - Detail View Panel
  - List View Panel
  - Card View Panel
  - List Detail Panel
- A List View Input in a Detail View Panel can reference:
  - List View Panel

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

30

G U I D E W I R E

List view panels can be referenced by four types of containers: screens, card view panels, list detail panels, and detail view panels.

The methods for referencing a list view panel in a screen, card view panel or list detail panels is identical: you use a PanelRef widget. To embed a list view panel in a detail view panel you use a List View Input widget.

# Reference a List View Panel

Screen, Card View Panel,  
or List Detail Panel

List View Input

1. Add panel ref
2. Specify ref properties
3. Add toolbar
4. Deploy PCFs

1. Add list view input
2. Specify ref properties
3. Add toolbar
4. Deploy PCFs

## Panel Ref

The screenshot shows a 'History' panel with a title bar 'Panel Ref'. Below it is a toolbar with navigation icons (back, forward, search) and a page indicator 'Page 1 of 3'. A red arrow points from the text 'Toolbar' to the toolbar area. A callout box labeled 'List View Panel' points to a table below. The table has columns 'Date', 'Event Type', and 'Description'. The data is as follows:

Date	Event Type	Description
02/06/2014	Viewed	Super User viewed this contact.
02/06/2014	Viewed	Alice Applegate viewed this contact.
02/05/2014	Viewed	Contact William David created

- A **Panel Ref** includes a reference to a "panel" container
  - Card View Panel, Detail View Panel, List Detail Panel, List View Panel, or Panel Set
  - Optionally supplies referenced panel with Title, Toolbar, Help Text
- If applicable, a toolbar automatically applies paging

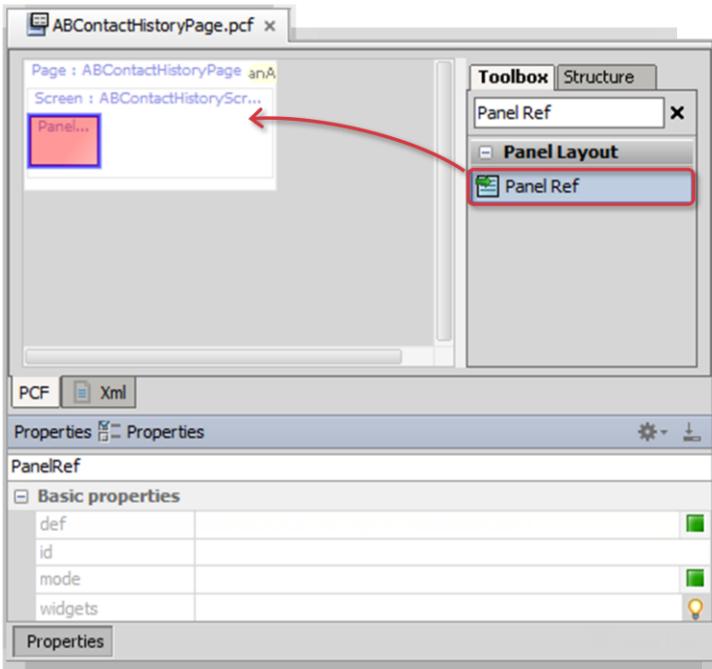
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

32

G U I D E W I R E

A Panel Ref requires a reference to a panel such as a Detail View Panel, List View Panel, Panel Set or Card View Panel. A Panel Ref supplies the referenced panel with title, toolbar or instructional text.

## Step 1: Add panel ref

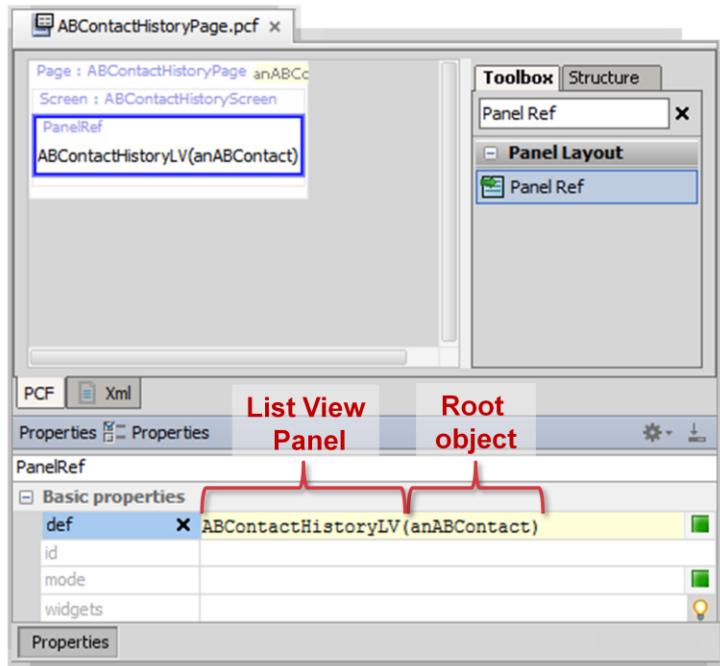


- Add a PanelRef widget to the parent container
- Panel Ref requires a def property value

A PanelRef widget can reference a list view panel. To reference a list view panel from a parent container, add a Panel Ref in the appropriate place in the parent container.

In the slide example, the Panel Ref has already been placed in the Screen in the PCF editor canvas. The Properties window shows the Properties tab of the Panel Ref.

## Step 2: Reference the list view panel



- Define the def property to specify the list view panel
- Pass the required root object type as an argument

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

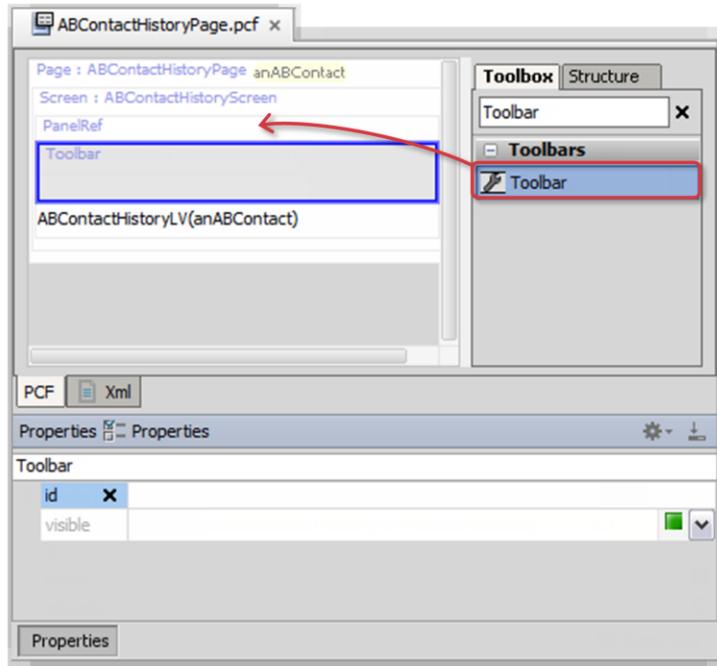
34

G U I D E W I R E

In the PanelRef's def property, specify the list view panel. Specify the required object(s) to pass to the list view panel in parentheses.

In the slide example, ABContactHistoryPage defines a root object named anABContact. ABContactHistoryPage contains a newly added Panel Ref. The Panel Ref requires a value for the def property. The def property references the list view panel named ABContactHistoryLV. The def property passes the anABContact root object as an argument to ABContactHistoryLV.

## Step 3: Add toolbar for paging



- Add a Toolbar to PanelRef widget
- Toolbar widget provides paging controls for list view panel

In the slide example, the Toolbar has already been placed in the PanelRef in the PCF editor canvas. The Properties window shows the Properties tab for the Toolbar.

A toolbar is a row of widgets associated with a container widget that lets the user take action on the data in the container widget. A toolbar can be directly add to a Screen, Panel Ref, or List View Input. Toolbars can also be associated with Detail View Panels, Card View Panels, List Detail Panels, and List View Panels.

When a toolbar is associated with a list view panel, the toolbar may not have any button widgets. In this specific case, the toolbar association may be required to provide button paging controls. Paging button controls allow users to view the list view rows in sizable chunks and "page" through the results. A user can move to the first, last, previous and next page. A user can also specify a specific page number to view.

# List View Input

The screenshot shows a 'Employee Info' section with two items: 'Can Have Employees?' (Yes) and 'Number of Employees' (8). Below this is a 'Employees' section. A 'List View Input' button is highlighted with a purple box. The 'List View Panel' is shown below, containing a table with three columns: Name, Job Title, and Email Address. The table has four rows with data: William Dan (Accountant, William.Dan@albertsons), Adam Hinds (Cashier, Adam.Hinds@albertsons), and Mary Leflore (General Manager, Mary.Leflore@albertsons). A red arrow points to the toolbar above the table, labeled 'Toolbar'. The toolbar includes navigation icons (double arrows, left, right), a page number (1 of 2), and a 'List View Input' button.

Name	Job Title	Email Address
William Dan	Accountant	William.Dan@albertsons
Adam Hinds	Cashier	Adam.Hinds@albertsons
Mary Leflore	General Manager	Mary.Leflore@albertsons

- A **List View Input** references a List View Panel for a Detail View Panel
  - An optional toolbar automatically applies paging

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

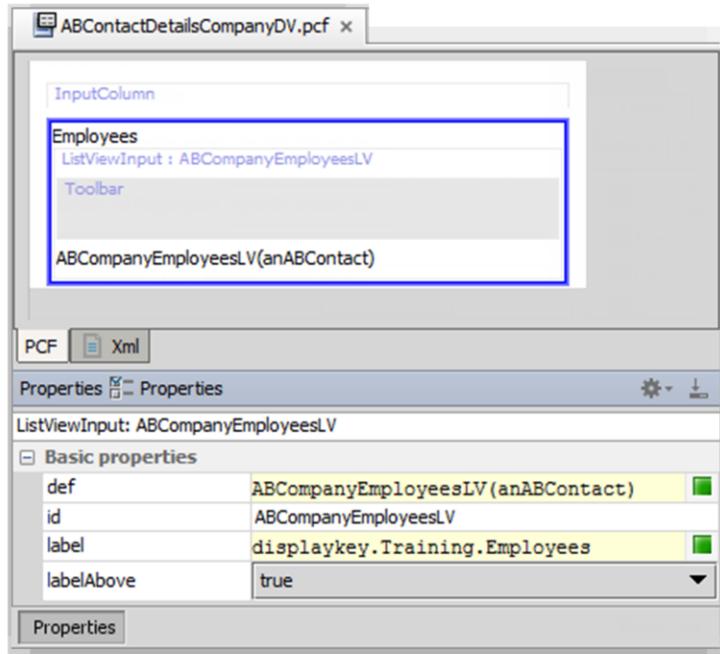
36

G U I D E W I R E

A ListViewInput element references a List View Panel and supplies it with an optional toolbar. A List View Input allows for an List View Panel to be placed in a Detail View Panel.

In the slide example, the page is in read-only mode. The toolbar automatically adds paging of the list in both read-only and edit modes.

# List view input properties



- **def**
  - references list view panel
- **label**
  - Value to display in UI
  - Use displaykey
- **labelAbove**
  - Affects label position
  - True is above
  - False is left of list view

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

37

G U I D E W I R E

# Properties: label and labelAbove

- Default
  - No label
  - labelAbove=true

« «   Page 1 of 2   » »		
Name	Job Title	Email Address
William Andy	Manager	William.Andy@albertsons
William David	Associate	William.David@albertsons

- Offset
  - No label
  - labelAbove=false

« «   Page 1 of 2   » »		
Name	Job Title	Email Address
William Andy	Manager	William.Andy@albertsons
William David	Associate	William.David@albertsons

- Above
  - label
  - labelAbove=true

Employees		
« «   Page 1 of 2   » »		
Name	Job Title	Email Address
William Andy	Manager	William.Andy@albertsons
William David	Associate	William.David@albertsons

- Offset Above
  - label
  - labelAbove=false

Employees		
« «   Page 1 of 2   » »		
Name	Job Title	Email Address
William Andy	Manager	William.Andy@albertsons
William David	Associate	William.David@albertsons

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

38

G U I D E W I R E

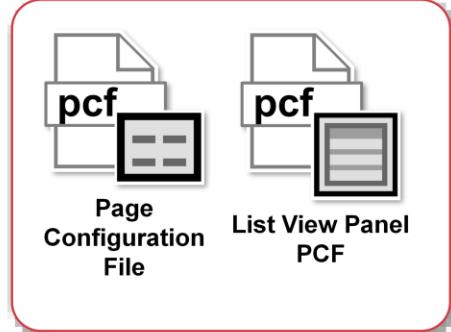
When labelAbove is set to false, the application offsets the list view from the left side of the input column to make room for the label.

If you do not plan to use a label for the list view, you may want to set the labelAbove to true to make better use of the space on the screen.

## Step 4: Deploy PCFs

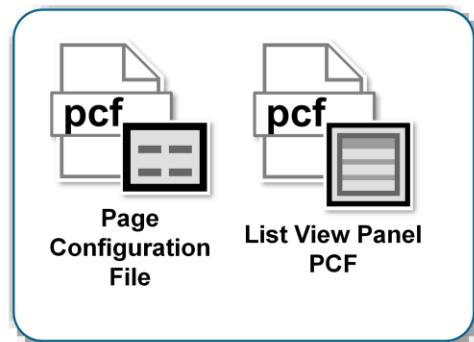
### Restart Server

- PCFs read at server startup



### Reload PCFs

- ALT+SHIFT+L
  - Internal debug tools enabled
- Internal Tools
  - Reload → Reload PCF Files



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

39

G U I D E W I R E

It is also possible to reload PCF files using the Guidewire API and/or internal server tools. The Reload PCF command can be found on the Reload page in Internal Tools. To access Internal Tools, you must log in as an administrator user, e.g., su/gw. Then, use ALT+SHIFT+T. In the tab bar, select Internal Tools → Reload. On the Reload page, click the Reload PCF Files button. The Reload PCF Files button calls the static method `gw.api.tools.InternalToolsUtil.reloadPCFs()`.

## Lesson objectives review

- You should now be able to:
  - Describe the functionality of list view panels
  - Create a new list view panel
  - Create and modify row iterator, row, and cell widgets
  - Reference list view panels

# Review questions

1. In a list view panel:
  - a) What type of widget displays individual fields of data?
  - b) What type of widget organizes the individual fields of data?
2. Assume you have a list view panel with "anABContact" as the root object. The list view panel displays the contact's addresses.
  - a) What would the row iterator's "value" property be set to?
  - b) What would the row iterator's "element name" be set to?
  - c) What other widgets would make use of the element name?
3. A list view typically needs a toolbar, even if it is read-only. Why?
4. In what way is embedding a list view panel in a detail view panel different from embedding a list view panel in a screen?

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

41

G U I D E W I R E

## Answers

- 1a) Cell widgets
- 1b) Row widgets
- 2a) anABContact.Addresses (assuming that the name of the addresses array is "Addresses")
- 2b) The value is arbitrary, but it would probably be set to something like "currentAddress".
- 2c) The cell widgets inside the row iterator's row. For example, a cell displaying the street would have a value of "currentAddress.Street".
- 3) The toolbar is needed for the paging controls. These controls are used to view each page of rows if the number of rows is greater than what can be displayed at one time.
- 4) To embed a list view panel in a detail view panel , use a ListViewInput widget. To embed a list view panel in a screen (or card view panle or list detail panel , you use a PanelRef widget.

# 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.

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

42

G U I D E W I R E