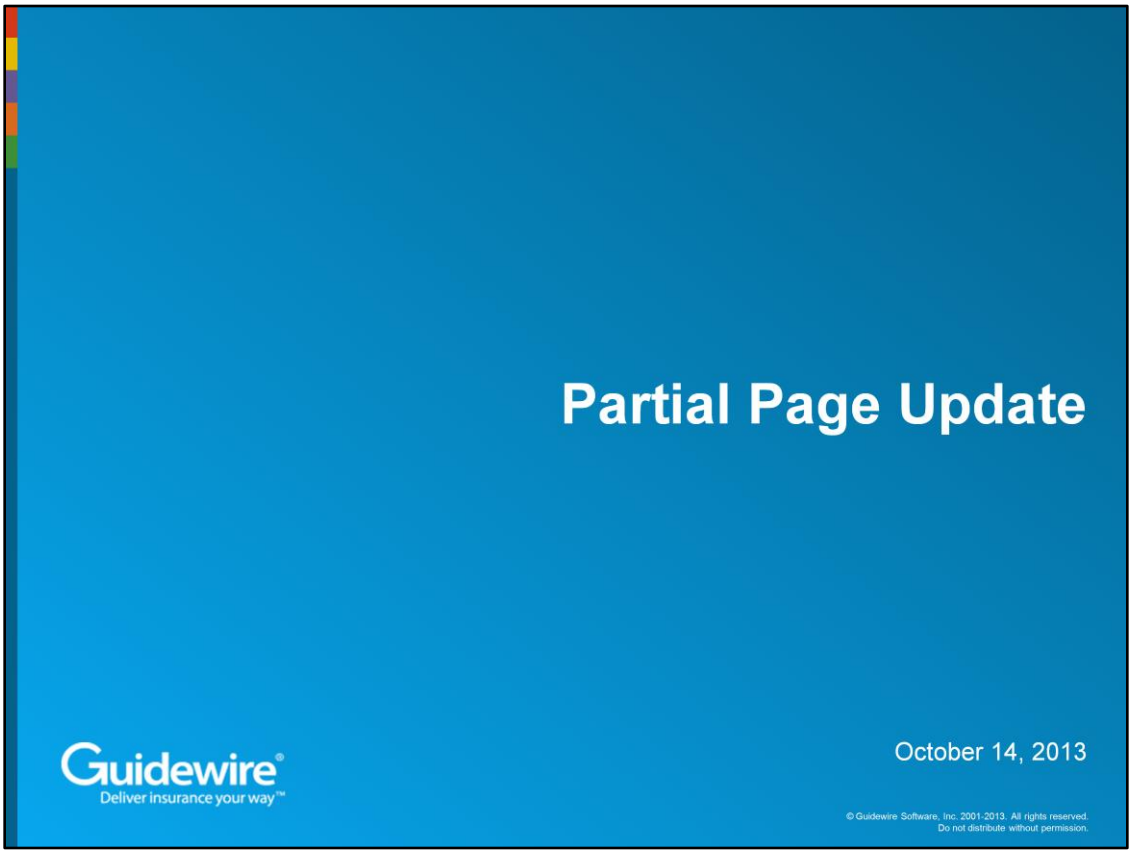


Guidewire training materials contain Guidewire proprietary information that is subject to confidentiality and non-disclosure agreements. You agree to use the information in this manual solely for the purpose of training to implement Guidewire software solutions. You also agree not to disclose the information 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.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Lesson objectives

- By the end of this lesson, you should be able to:
  - Identify common ways to configure a partial page update
  - Enable targeted Post On Change for an input and define properties that are best performing
  - Differentiate between targeted Post On Change and client reflection
  - Replace client reflection with targeted Post On Change

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. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Lesson outline

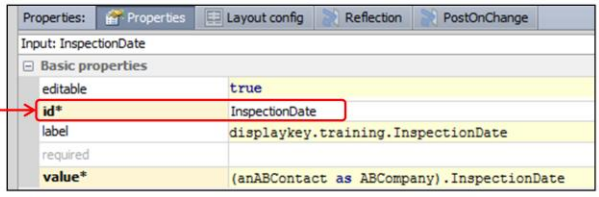
- Partial page update
- Configuring targeted Post On Change
- Replacing client reflection

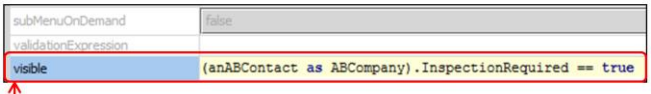
© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Static and dynamic widget properties

- Some widget properties must be set to static values
  - They define aspects of widget that never change





- Some widget properties can be set to Gosu expressions
  - They define dynamic behaviors

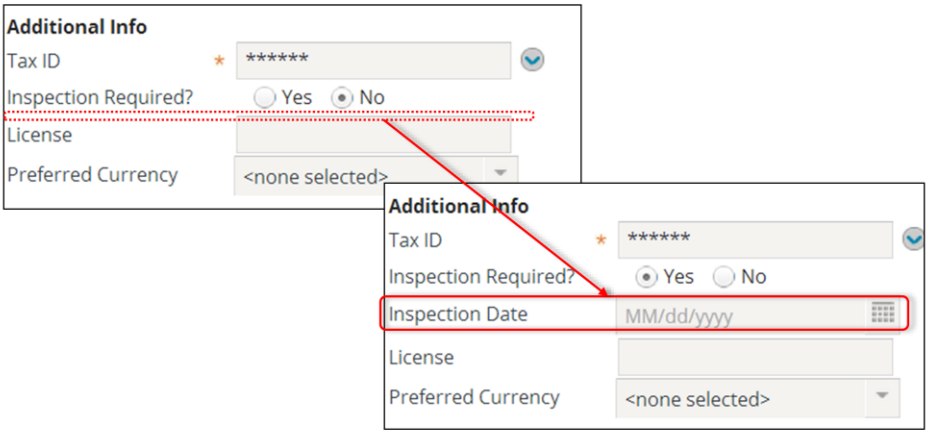
© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

In the example above, the "id" property of a widget is a static property. The widget always has the same ID, regardless of the state of the application or the values of any business data. However, the "visible" property is a dynamic property. It evaluates to either true (if the InspectionRequired field is true) or false (if the InspectionRequired field is false). Theoretically, the visible condition could be set to "(ABContact as ABCompany).InspectionRequired". The "== true" has been added only to clarify to students that the InspectionRequired field is a boolean.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Dynamic widget behavior

- Dynamic widget behavior refers to widget behavior that responds to changes in business data
- Examples:



© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

In the example above, the Company Info card has both an Inspection Required? field and Inspection Date field. However, the Inspection Date field is visible only if the Inspection Required? field is set to Yes.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Why implement dynamic widget behavior?

- To hide (or show) widgets and containers irrelevant (or relevant) to current business process
- To conditionally prevent (or allow) widget values from being modified
- To conditionally make a widget's value required (or optional)
- To update data immediately after a user changes a given widget's value

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

Widgets have properties that can be controlled by dynamic values. The examples in the slide reference the widget properties (respectively):

Visible, Available
Editable, Available
Required
onChange

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Identifying properties that are Gosu expressions

Guidewire PCF Format Reference

Input top

A generic Input, which can be bound to any value. The type of the value widget (e.g., textbox, dropdown or checkbox) will be determined dynamically based on the value type, for example:

- Boolean - BooleanRadioInput
- Date - DateInput
- TypeKey - TypeKeyInput

If no such mapping is found, the default is to use a TextInput. In general, this tag should be used where possible in preference to a more specific tag, unless the special attributes of the specific tag are needed.

Child elements: Reflect AbstractMenuItem

Attribute:	Type:	Default:	Description:
action	statement		An optional action to take when the user clicks the value of this widget when the value is not being edited. See ActionBase for list of available action prefixes. When specified, the value of this widget will be rendered as a link when the value is not being edited.
id	String		Reference ID of the widget. This does not have to be globally unique; it only has to be unique among the widget's siblings
visible	Expression (Boolean)		If used instead of "editable", specifies whether the Input is editable when data is being created

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

properties

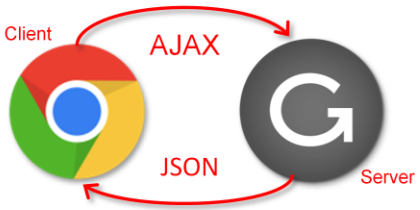
property

property property

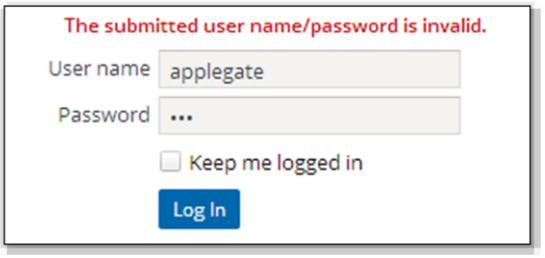
The PCF Format Reference is located in <ApplicationRootDirectory>\modules. It is called "pcf.htm".

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

How does processing occur?



- Client makes AJAX server request and server responds to client with JSON
- Client JavaScript libraries process JSON to update page data and layout
- Reduced network traffic
  - Content length = 1479 octets



JSON

```
{"cmd":"replaceItems",
 "items":[{
    "id":"Login:LoginScreen:0",
    "frame":false,
    "xtype":"gcontainer",
    "html":"The submitted user
     name/password is invalid.",
    "border":false }]
}
```

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.
8
GUIDEWIRE

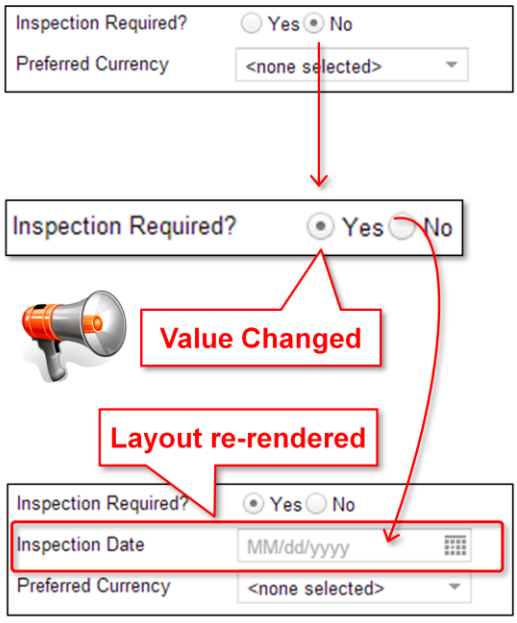
AJAX is an acronym for Asynchronous JavaScript and XML. The term generically describes how an application send data to and retrieve data from a server either asynchronously (in the background) or synchronously. XML is not a required format. Guidewire applications use JSON instead of XML for AJAX requests. JSON stands for JavaScript Object Notation. JSON is a text-data interchange format that is language independent and self-describing. JSON is like XML in that it is plain text, self-describing, hierarchal, and can be transported using AJAX. JSON is unlike XML in that it uses arrays, has no reserved keywords, and is quicker to read and parse.

In this example, it is important to observe that to display a simple error message to the user requires little server side processing and low levels of network traffic. This example is from the Emerald version of TrainingApp. Server-side processing involves the client making a HTTP Post or HTTP Get request to the Guidewire application server, the server renders a complete HTML page and responds to the client with the complete HTML, and the client browser then parses the document object model for the HTML and renders the resulting page.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Partial page update type: Layout re-render

- No data committed
- Applies to following widget properties
  - Visibility
  - Editability
  - Availability
  - Required
- Layout re-rendered for the page

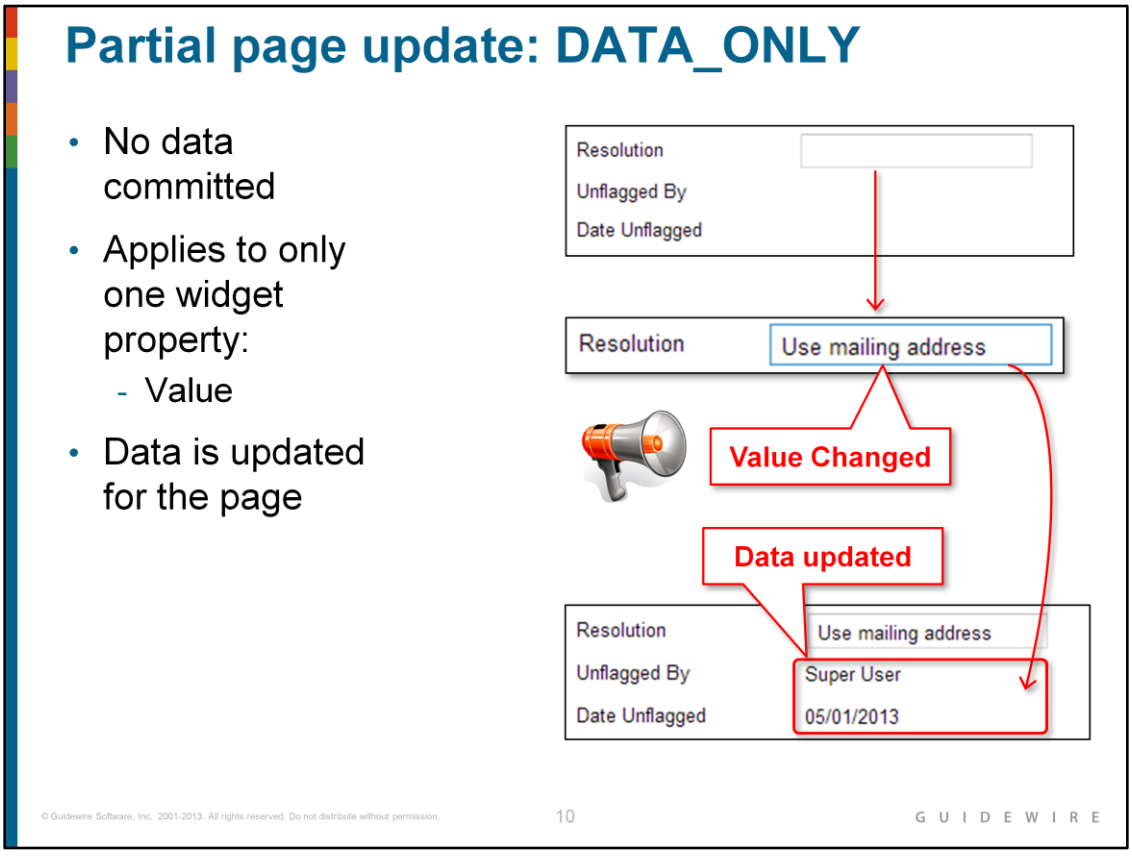


© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

There are two general and non-exclusive categories for a partial page update: DATA_ONLY and a layout re-render.

In the layout re-render example above, whenever the value of the InspectionRequired field is changed, all user editable data is sent from the client to the application server via AJAX. No data is committed. The server process the request and returns the user editable data to the client in JSON format. The client renders a partial page update that shows an Inspection Date calendar input.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

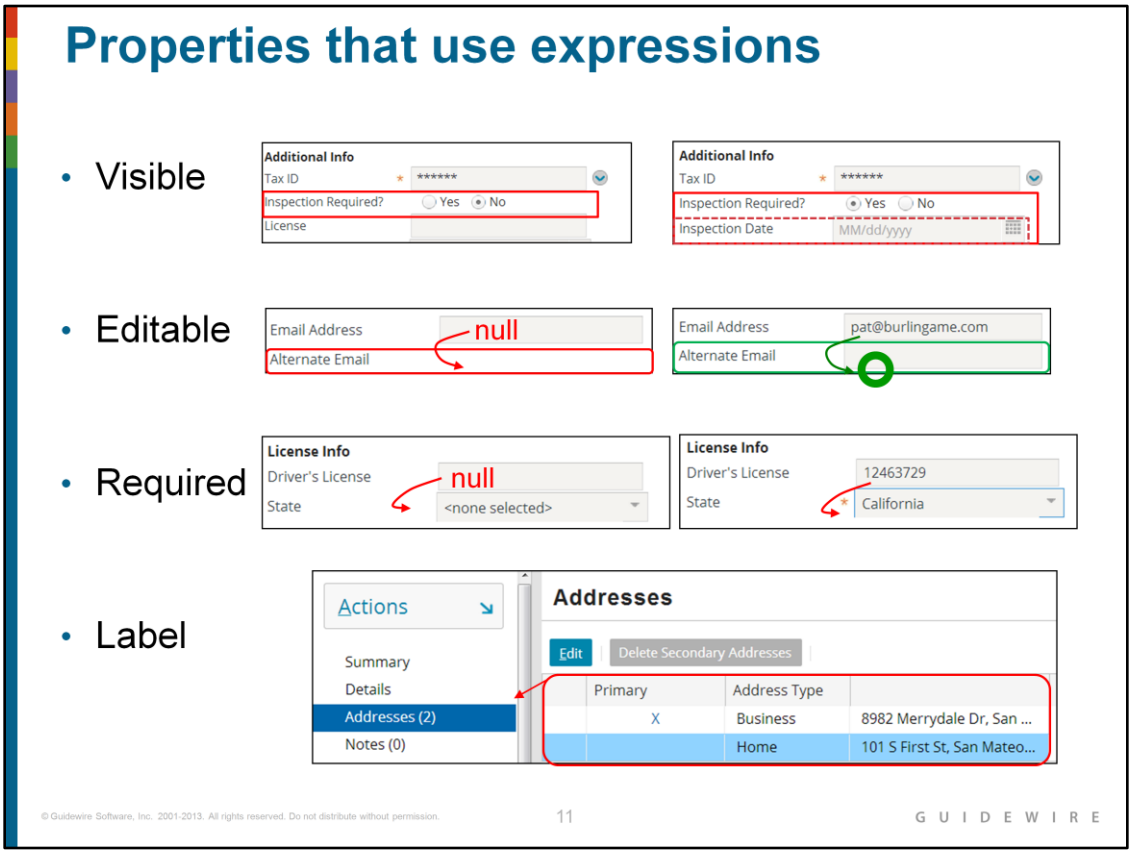


There are two general and non-exclusive categories for a partial page update: DATA_ONLY and a layout re-render.

In the DATA_ONLY example, whenever the value of the Resolution field is changed, all user editable data is sent from the client to the application server via AJAX. No data is committed. The server processes the request and returns the user editable data to the client in JSON format. The client renders a partial page update that shows new data for the Unflagged By and Date Unflagged fields.

DATA_ONLY applies to the value property of a widget. Examples include refreshing an input group and refreshing a set of fields on a cell change in a list view.

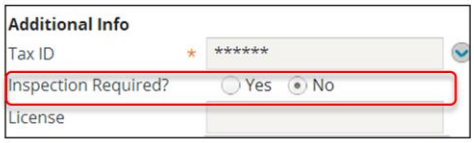
© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

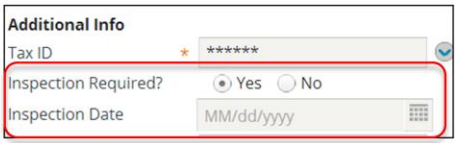


The visible property is used to dynamically control whether a widget is displayed.
The editable property is used to dynamically control whether a widget can be edited.
The required property is used to dynamically control whether a widget value is required.
The label property is used to dynamically control either which label is used or what is displayed in the label.
The properties listed above are not the only properties that use expressions, but they are properties that are commonly used with expressions.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Example 1: Conditionally visible





Display "Inspection Date" only if inspection is required

Properties: | Properties | PostOnChange | Layout config | Reflection

Input: InspectionDate

Basic properties

editable	`true`
id*	InspectionDate
label	`displaykey.training.InspectionDate`
required	
value*	`(anABContact as ABCompany).InspectionDate`
visible	`(anABContact as ABCompany).InspectionRequired == true`

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

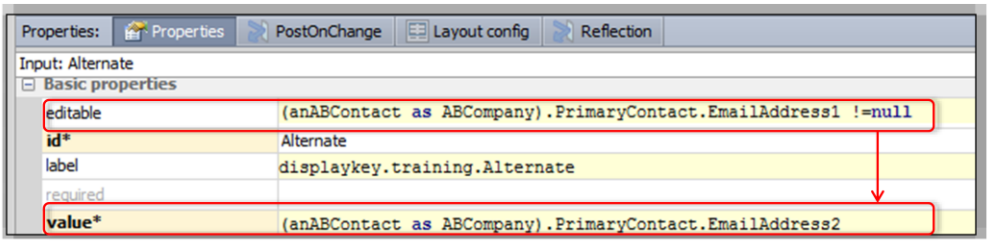
In the example above, the visible property resolves to true only if the contact's InspectionRequired field is true. (Theoretically, the visible condition could be set to simply "(ABContact as ABCompany).InspectionRequired". The "== true" has been added only to clarify to students that the InspectionRequired field is a boolean.)

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Example 2: Conditionally editable

Alternate email can be added only if main email exists

If the condition in the editable property is met, then the value for the Alternate Email field is added

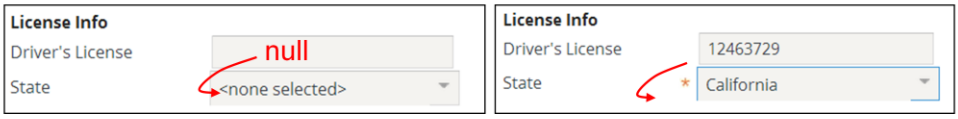


© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

In the example above, the editable property resolves to true only if the contact's EmailAddress1 field has a non-null value. This example is not implemented in TrainingApp and only serves as a possible scenario.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Example 3: Conditionally required



State is not required unless a driver's license number is provided

Properties: | Properties | PostOnChange | Layout config | Reflection

Input: State

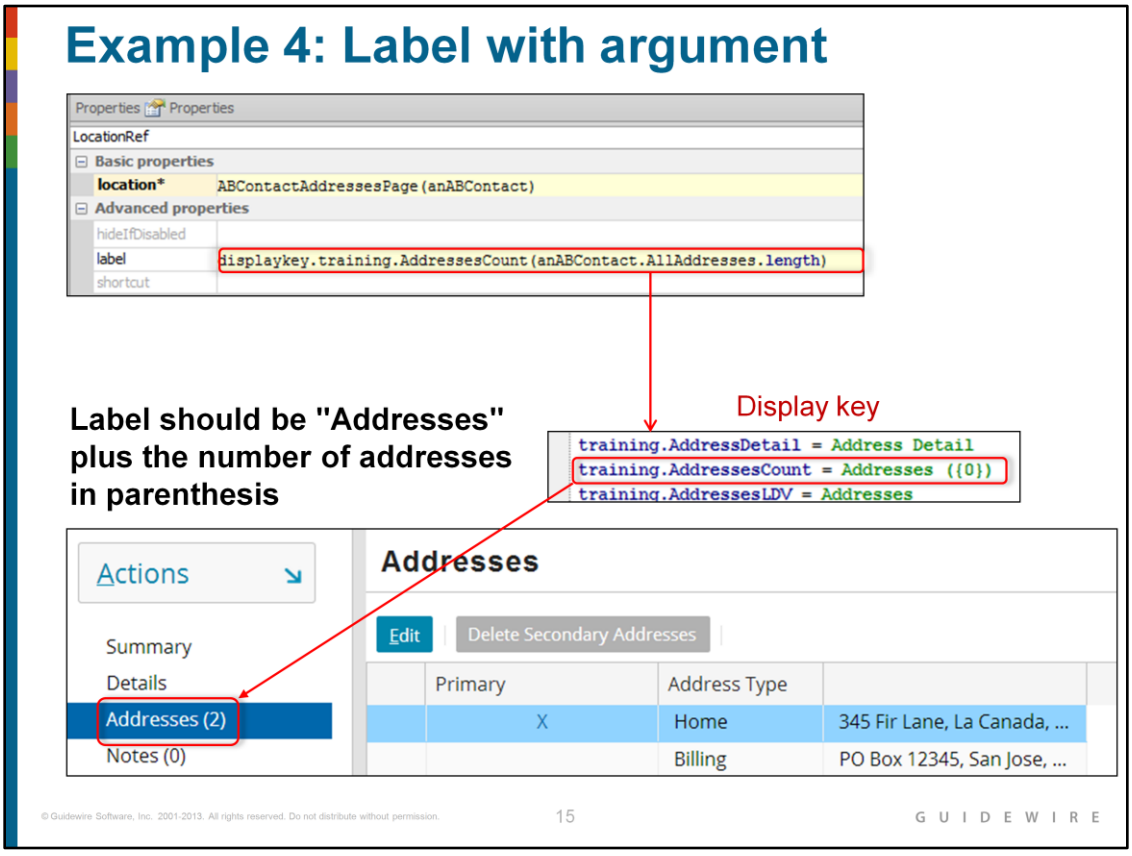
Basic properties

editable	true
id*	State
label	displaykey.training.State
required	(anABContact as ABPerson).LicenseNumber != null
value*	(anABContact as ABPerson).LicenseState

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

In the example above, the required property resolves to true only if the contact's license number field has a non-null value.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.



In the example above, the label property specifies a display key with an argument. It also passes in a value, specifically the length of the AllAddresses array. The label is rendered using the named display key with the argument is the designated location. Recall that a display key argument is noted by "{*X*}", where *X* is an integer. If there is only one argument, then the integer should be 0. If there are multiple arguments, then the arguments should be numbered starting with 0. When there are multiple arguments, the first argument listed in the label property is assigned to the {0} position, the second to the {1} position, and so on.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Lesson outline

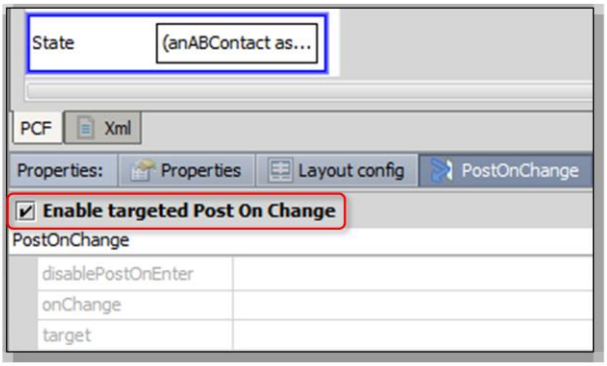
- Partial page update
- Configuring targeted Post On Change
- Replacing client reflection

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

PostOnChange tab property

- PostOnChange tab property
- Mark the Enable targeted Post On Change checkbox
- Configurable widgets
  - Cell
  - Input
  - InputGroup
- See pcf.xsd for schema support



- Three (3) configurable properties
  - disablePostOnEnter
  - onChange
  - target

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

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 updated 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.

You can determine which widgets support targeted Post On Change. In the pcf.xsd file, you can see if a specific widgets has the following element: <xsd:element ref="PostOnChange" …/>. The pcf.xsd file is in the parent directory of the Guidewire application project for Guidewire Studio.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

<PostOnChange /> XML example

```
1  <Input
2      editable="true"
3      id="DriversLicense"
4      label="displaykey.training.DriversLicense"
5      value="(anABContact as ABPerson).LicenseNumber">
6    <PostOnChange/>
7  </Input>
```

- **<PostOnChange/>** is an XML element in PCF
- Properties in Guidewire Studio are XML attributes
  - disablePostOnEnter
  - onChange
  - target

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

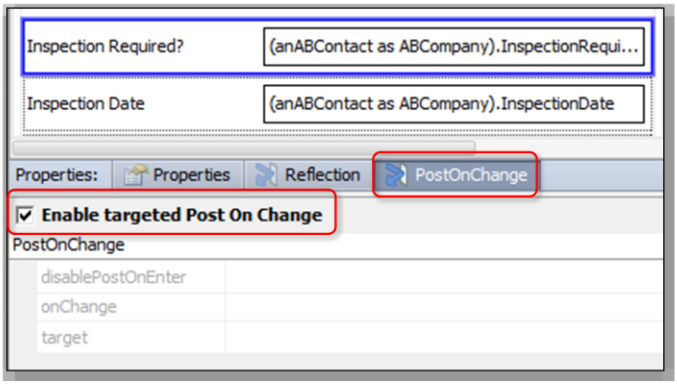
In the example above, Line 6 details the <PostOnChange /> element with no defined attributes.

PostOnChange is an XML element in PCF files for specific input and cell widgets. The XML element is the equivalent to a checked “Enable targeted Post On Change” checkbox for a given widget. The defined onChange attribute is equivalent to a specified onChange property in the Post On Change properties tab.

When upgrading previous applications, the upgrade tools will create the <PostOnChange/> XML element for widgets where the PostOnChange property equals true.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Targeted Post On Change



- Enable targeted Post On Change
- NO defined properties

- Data Behavior
  - Refresh of user-editable data
  - No data committed
- Layout Behavior
  - Entire page layout re-rendered
  - northPanel, westPanel, centerPanel, and southPanel

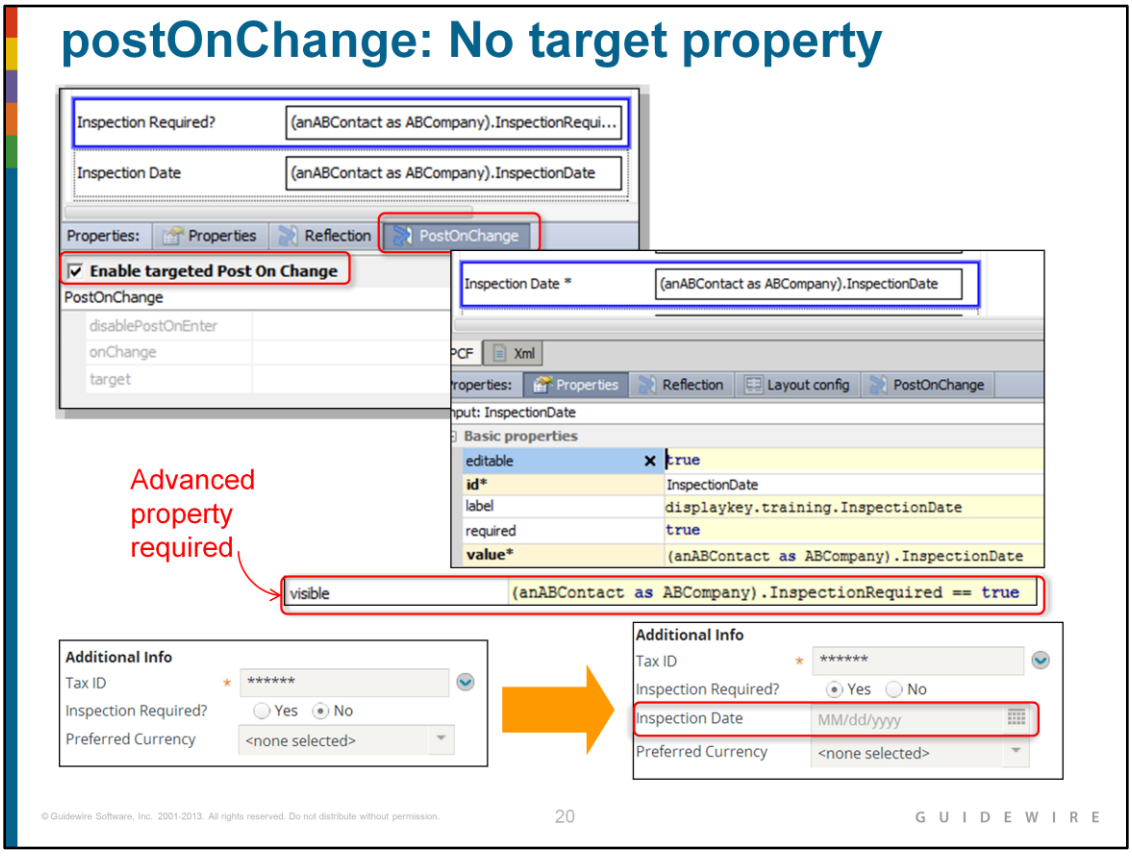
© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

If targeted Post On Change is enabled, the target property defines the scope and behavior of a page update, even if unspecified. Three types of values are accepted: No Value, specific Widget ID, and DATA_ONLY.

When no value is specified, the entire page is re-rendered and all user-editable data is refreshed. **There is a high performance cost** for implementing targeted Post On Change without specifying a target and any other properties. An unspecified value (no value) provides backwards compatibility for upgraded Guidewire applications.

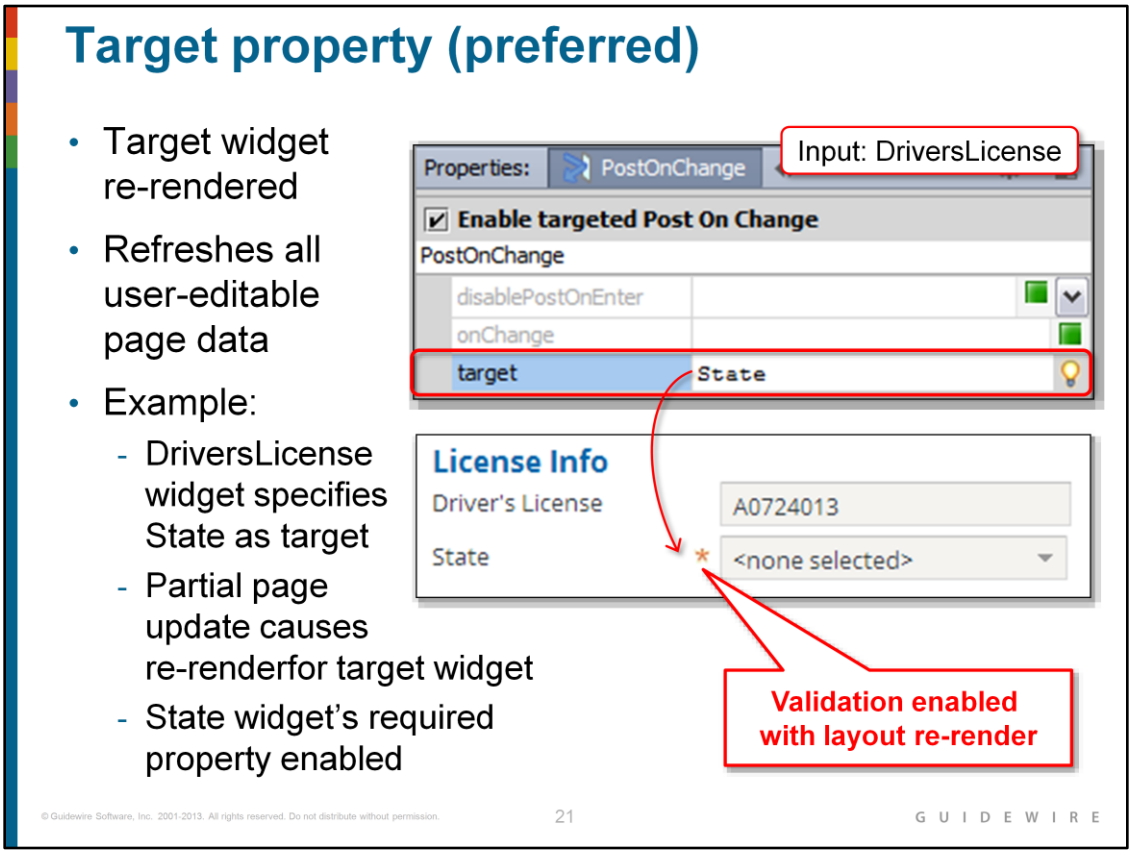
A page layout in Guidewire 8.0 applications consist of four defined panels: northPanel, westPanel, centerPanel, and SouthPanel. In this example, the application server renders the InspectionDate widget because the visible property of the InspectionDate widget is based on the boolean evaluation of anABContact's InspectionRequired property.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.



In this example, InspectionRequired's postOnChange property is enabled but **no value** (disablePostOnEnter, onChange, nor target) **is defined**. Therefore, whenever the widget value changes, the change is announced to all other widgets. This **is not optimal.** The InspectionDate's visible property is based on the InspectionRequired value (An advanced property). Consequently, whenever the InspectionRequired field is changed, the InspectionDate visibility is immediately updated.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

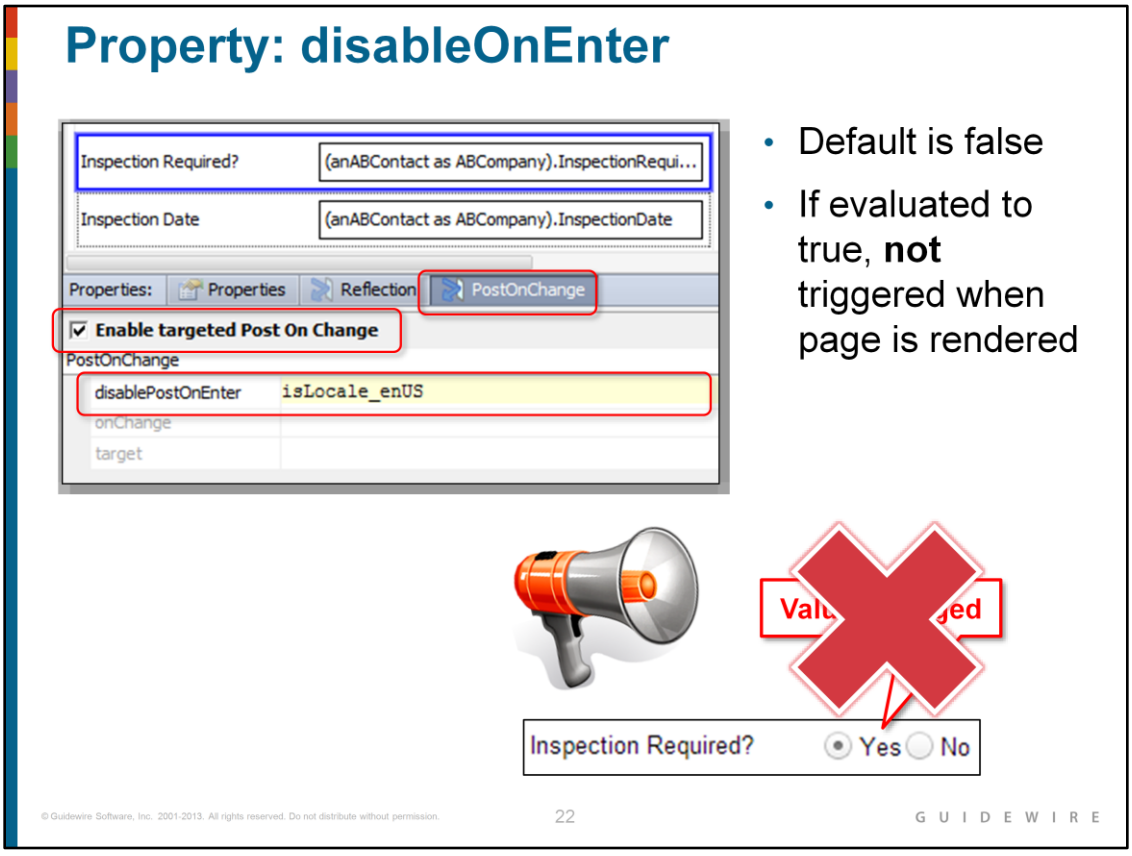


The target property defines the scope and behavior of a page update. Three types of values are accepted: No Value, specific Widget ID, and DATA_ONLY. Triggering refreshes all user-editable data, but **only the target widget is re-rendered** with a new layout. The following widget types are supported targets:

- inline input widget (individual input)
- inline input set
- inline input panel (DV panel or LV panel)
- included panel (Panel ref)
- exposed panel or input set
- label widget

In the slide example, targeted Post On Change is enabled for the DriversLicense input widget. The target is the State widget. When a user edits the Driver's License field, the change is announced to the State field. Only the layout of the State field is updated, causing the validation expression to equal to true as it is dependent on the existence of a value in the Driver's License field. All other user-editable data is also refreshed.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.



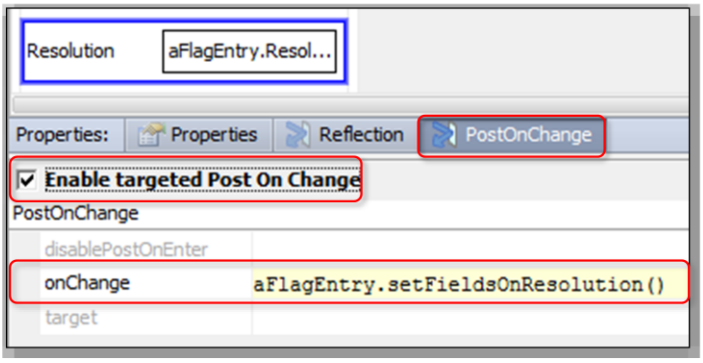
In certain cases, you may want to toggle (disable or enable) targeted Post On Change behavior for a specific widget.

In the layout example above, targeted Post On Change is enabled for the InspectionRequired input widget. However, the disablePostOnEnter property is set to the isLocale_enUS variable. A variable evaluation of true disables the targeted Post On Change behavior for the InspectionRequired widget. When the disablePostOnEnter property is true, the InspectionRequired widget will not announce a change in its value; data will not make a round-trip between client and the application server; the page will not be re-rendered; and unless the InspectionRequired field is already true for the given ABContact, the InspectionDate widget will not be visible.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Property: onChange

- Defines the Gosu expression to invoke
- Applicable when user changes the editable value of a widget



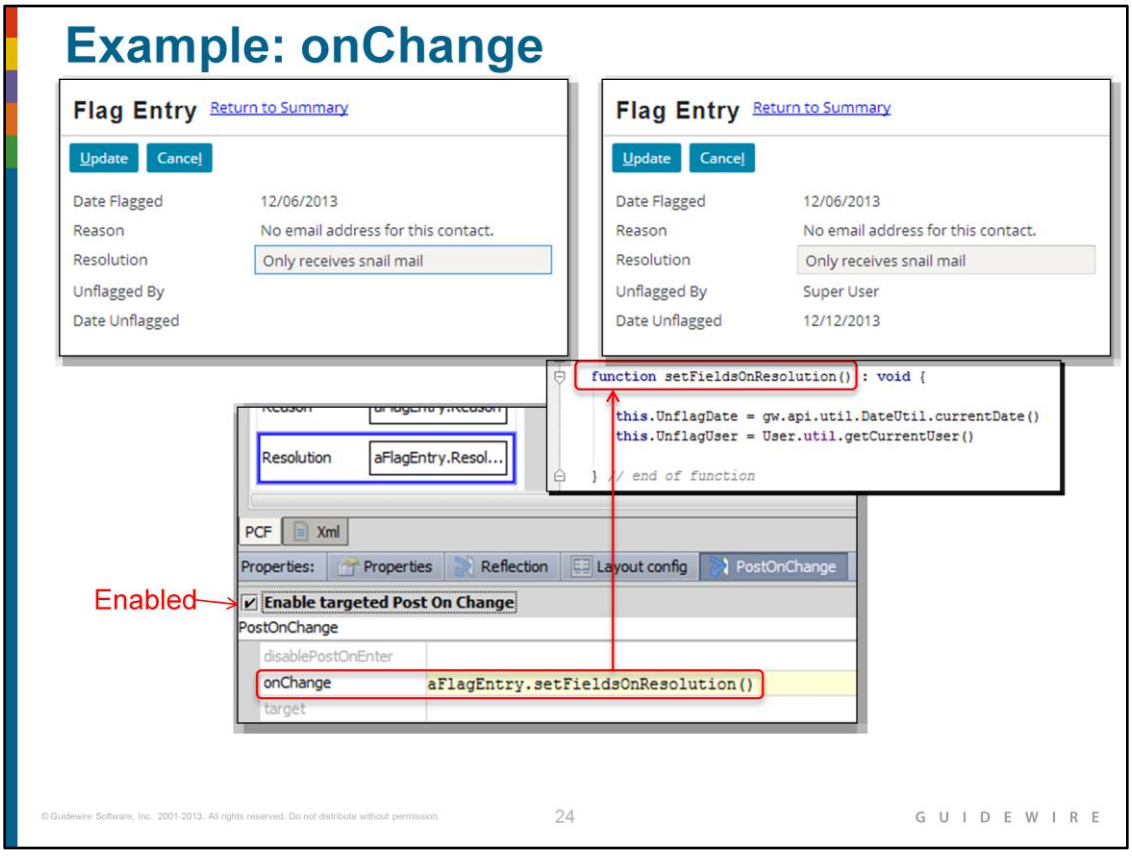
- Causes immediate post back to server
  - High cost to server-side processing in PCF configuration, especially if complex
- Upgrade support for onChange widget property

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

In the slide example, the Gosu expression defined for the onChange property invokes the setFieldsOnResolution() function. The function is defined in the entity enhancement, FlagEntryEnhancement.gsx.

```
16  /* This function is called when a FlagEntry's resolution field
17       is set. This function sets the UnflagDate and UnflagUser
18       fields. This serves the role of a "FlagEntry Pre-Update"
19       rule set.
20  */
21   function setFieldsOnResolution() : void {
22     this.UnflagDate = gw.api.util.DateUtil.currentDate()
23     this.UnflagUser = User.util.getCurrentUser()
24   } // end of function
```

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.



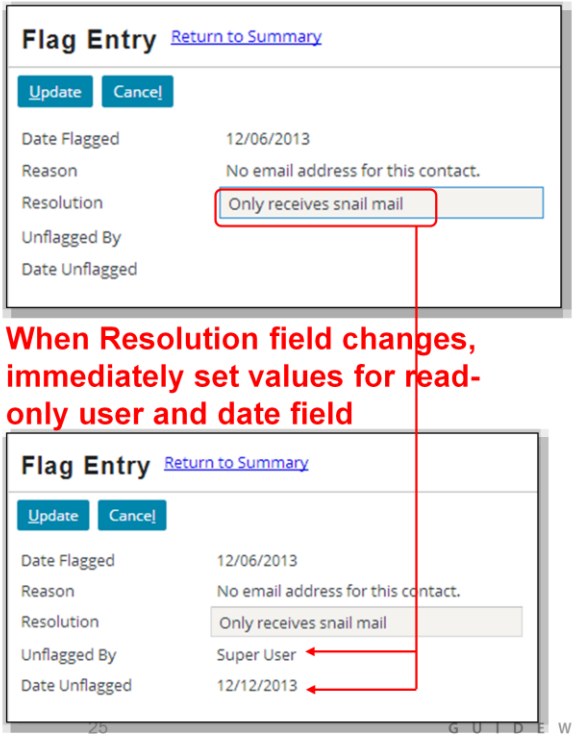
In the example above, whenever the value of the Resolution field changes and the user makes some other widget active, the setFieldsOnResolution function executes.

If the code executed by onChange modifies any field values, these modifications are done to the run-time objects only . In other words, the new values will be displayed in the user interface, but the values are not yet committed to the database. In fact, the changed values will never be committed unless something else explicitly commits the data (such as being in Edit mode and then clicking the Update button).

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Complex reactions to widget changes

- Some widget behavior cannot be implemented with the visible/editable/ required properties and postOnChange
  - For example, setting fields to current date and user



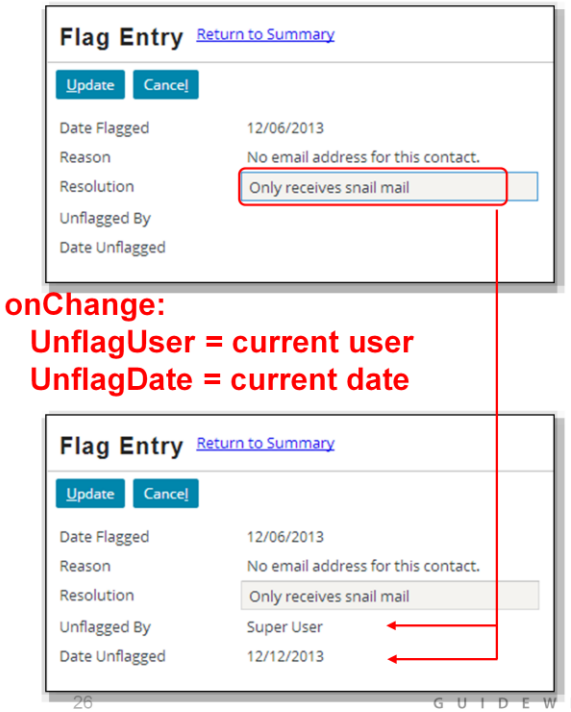
© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

Gosu expression properties and postOnChange are useful when you want to toggle boolean widget properties, such as visible or required, between true and false. However, they cannot be used to provide more open-ended functionality, such as setting a given field to the current date when a given event occurs.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

onChange property

- **onChange** is a widget property that executes code whenever widget's value changes
  - postOnChange must be true for onChange to trigger



© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

The onChange property is also discussed in the "PCF Methods" lesson.

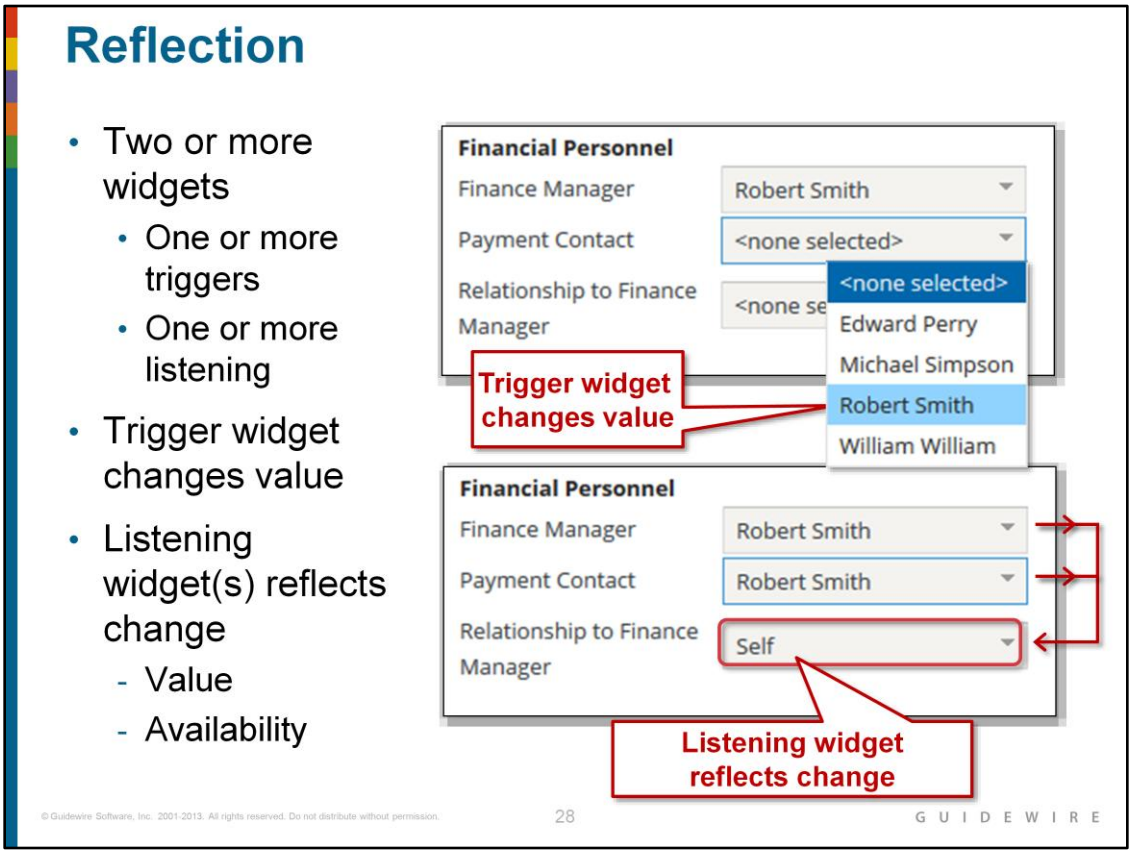
© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Lesson outline

- Partial page update
- Configuring targeted Post On Change
- Replacing client reflection

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

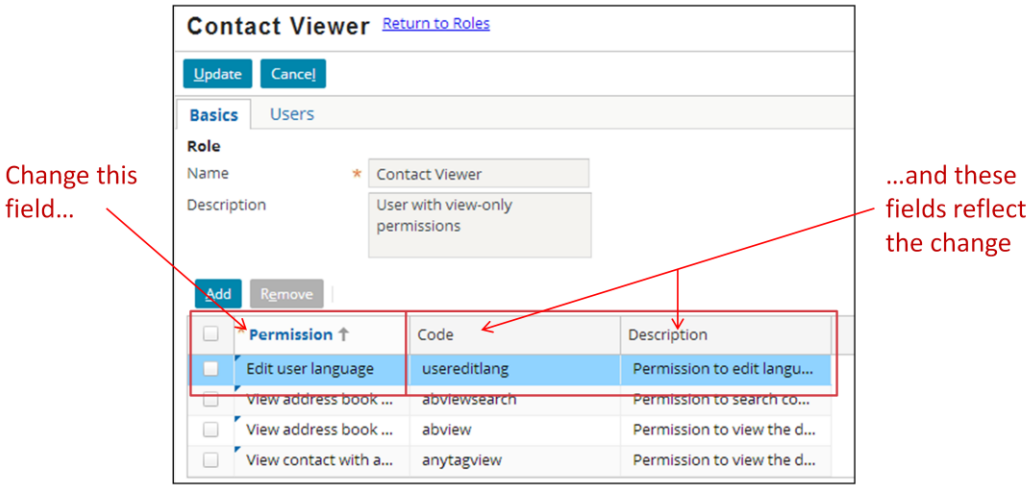


In the example above, whenever a change is made to the either the Finance Manager or Payment Contact, the Relationship to Finance Manager field reflects the change.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Reflection capabilities

- Reflection **can** be used to set listening widget's:
  Value, Availability (and indirectly its Editability)

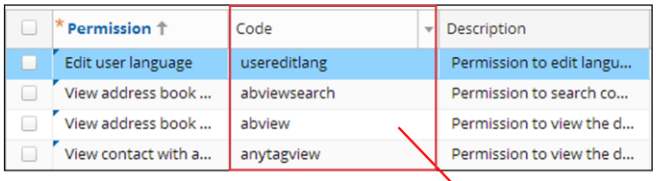


- Reflection **cannot** be used to set listening widget's:
  - Visibility, Requiredness, Label

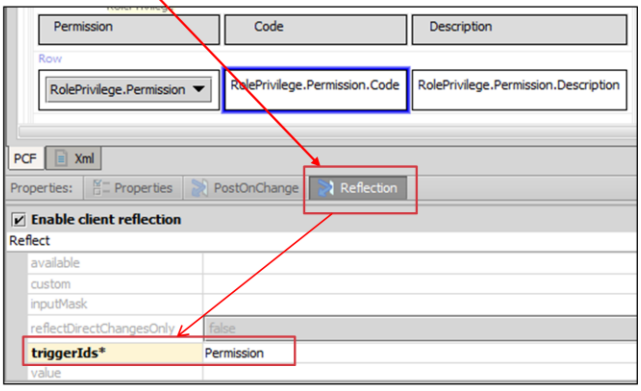
© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Reflection tab



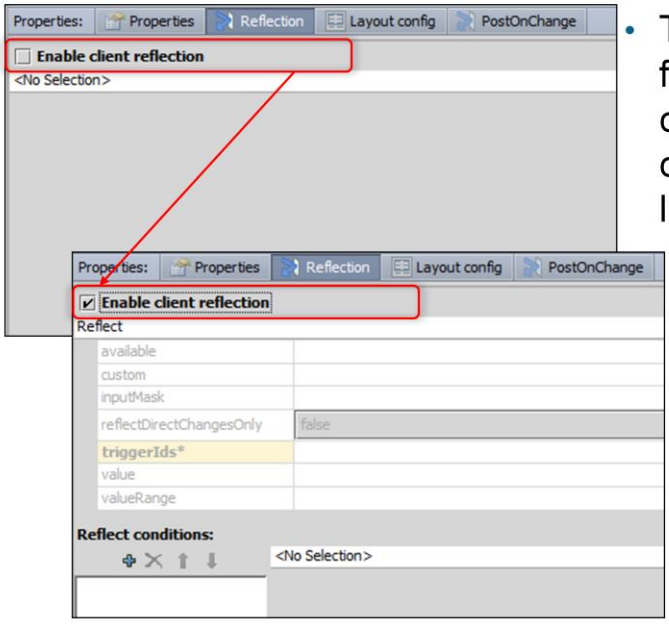
- Reflection is configured on listening widget's Reflection tab



© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

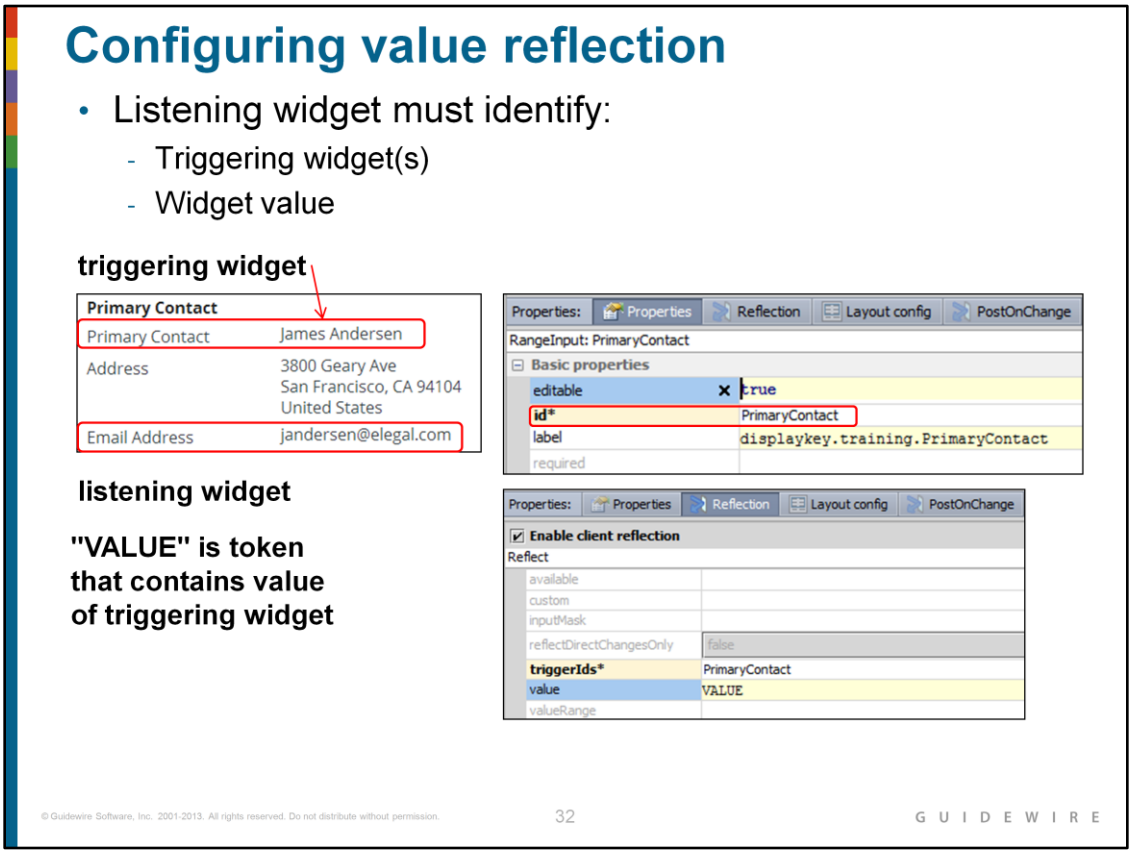
Enabling reflection



- To enable reflection, first select "Enable client reflection" checkbox on listening widget

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.



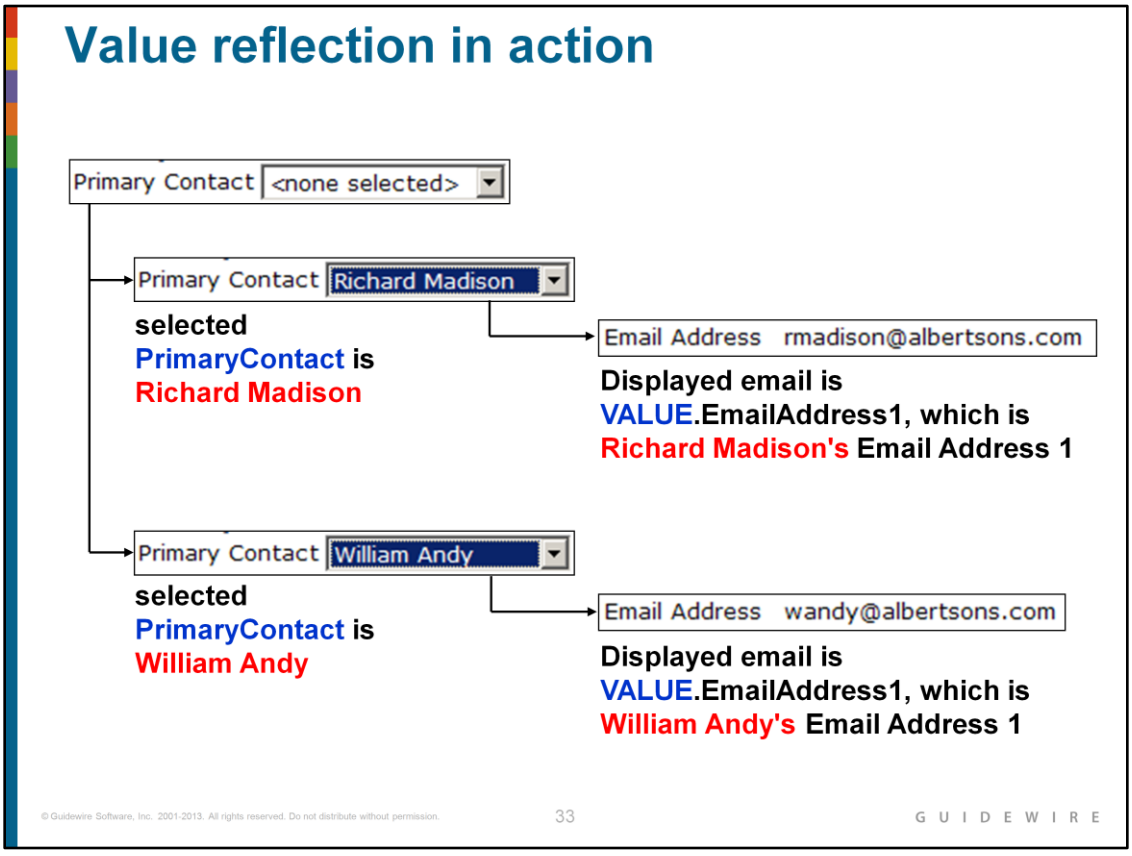
The VALUE token represents the value of the widget that is referenced by the triggerIds property.
It is possible for a single widget to reflect changes from multiple triggering widgets. Each argument becomes a numbered token as in VALUE1, VALUE2, VALUE3, etc. The value field for the listening widget can be a Gosu expression.

There is one circumstance where the value property on the Reflection tab is optional. If the listening widget is a property on an object and the triggering widget is a foreign key to that object, then the value property on the Reflection tab is not required. In this situation, a Guidewire application can infer the value from the value property on the Properties tab. For example, if the listening widget is the EmailAddress1 field on a contact object, and the triggering widget PrimaryContact points to that same object, then a Guidewire application can infer that the Reflection value should be PrimaryContact.EmailAddress1. However, including the value property does not cause problems and may make the configuration more readable.
The value field is required if:

The triggering widget is not a foreign key field pointing to an object, or
The triggering widget is a foreign key field pointing to an object, but the reflecting widget reflects a value that is not a field on that object

Note that the example shown on this and the next slide is not representative of what is actually configured in TrainingApp, but has been configured to demonstrate value reflection.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.



In the example above, the Email Address widget's client reflection has been enabled with the following properties set as stated:

triggerIds is set to "PrimaryContact"
value is set to "VALUE"

Changes made to the primary contact widget are immediately reflected in the EmailAddress widget.

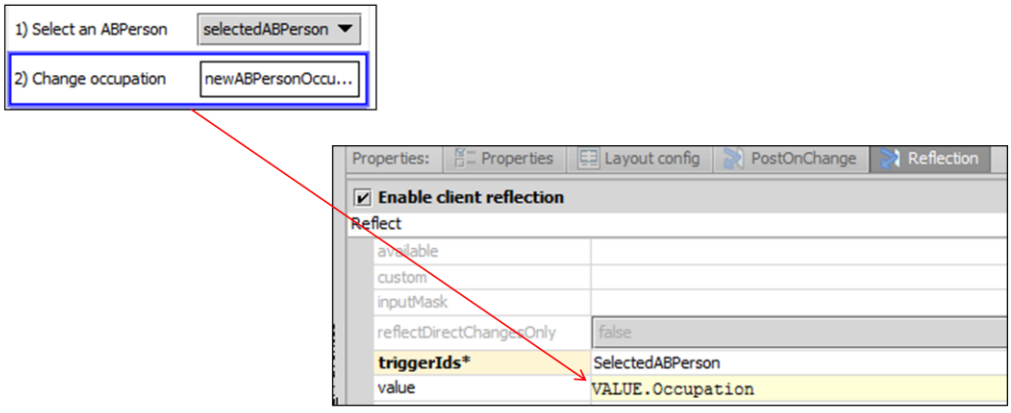
Keep in mind that the example shows the *selected* value of Primary Contact and the *displayed* email address. Because these changes are occurring in Edit mode, the *values* of Primary Contact and Email Address remain null until the Update button is clicked and changes are committed.

Note that the example shown on this and the previous slide is not representative of what is actually configured in TrainingApp, but has been configured to demonstrate value reflection.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Value reflection: additional information

- If the value of a VALUE token has properties, they can be accessed by using standard dot notation (VALUE.property)

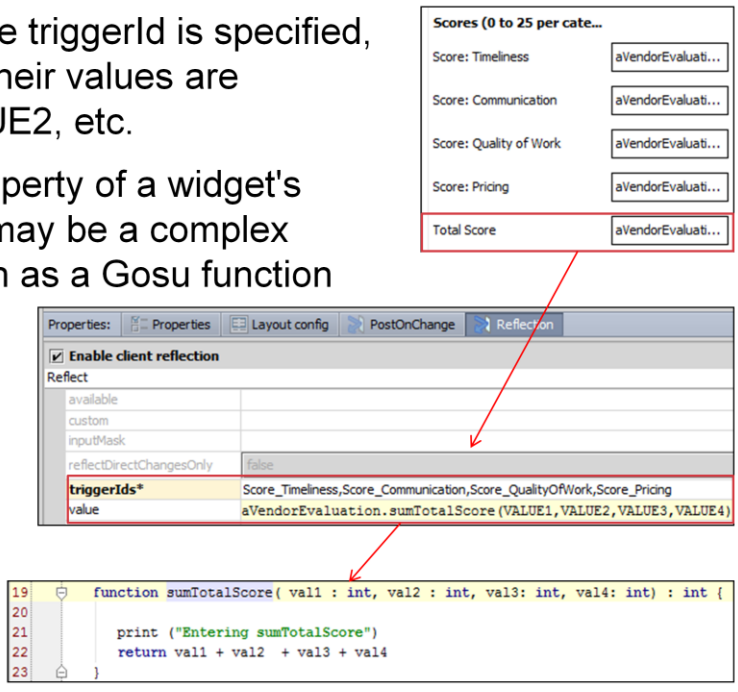


© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Complex value reflection

- If more than one triggerId is specified, the tokens for their values are VALUE1, VALUE2, etc.
- The "value" property of a widget's Reflection tab may be a complex statement, such as a Gosu function



© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

In the example shown, the Total Score widget listens to four other widgets. A change to any of these triggering widgets causes the value of the listening widget to update using the sumTotalScore function from VendorEvaluationEnhancement.gsx.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Basic implementation comparison

Client Reflection

- On listening widget
- Reflection tab
- Define triggerIds property
  - One widget ID
- Specify Value property
  - References specific triggerId field
  - Optional VALUE token

Targeted Post On Change

- On triggering widget
- PostOnChange tab
- Define target property
  - One widget ID
- Assign value with onChange
  - Variable

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Advanced implementation comparison

Client Reflection

- On listening widget
- Reflection tab
- Define triggerIds property
  - Comma delimited list of widget IDs
- Specify Value property
  - VALUEx tokens
  - References specific TriggerId field value

Targeted Post On Change

- On triggering widget
- PostOnChange tab
- Define target property
  - InputSet, InputGroup, ListViewPanel, or Ref id
- Assign value with onChange
  - Expression
  - Method

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Deprecated server-side client reflection

- Two kinds of client reflection
  - Client-side
  - Server-side
- Client-side still supported as partial page update option
  - Targeted post onChange is easier to manage and configure in most cases
- Client reflection with server-side expressions are deprecated
  - Replace with targeted post on change

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Replacing reflection with targeted Post On Change

1. Identify the listening widgets configured for client reflection
2. Identify the announcing widgets, e.g., the triggerIds, for client reflection
3. Enable targeted Post On Change for announcing widgets
4. Specify required properties for best performance
   - disableOnEnter
   - onChange
   - Target
5. Disable client reflection for listening widgets
6. Test and verify behavior

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Lesson objectives review

- You should now be able to:
  - Identify common ways to configure a partial page update
  - Enable targeted Post On Change for an input and define properties that are best performing
  - Differentiate between targeted Post On Change and client reflection
  - Replace client reflection with targeted Post On Change

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Review questions

1. How is a partial page update implemented?
2. What are the three properties that you can configure for a widget with targeted Post On Change enabled?
3. What are the three possibilities for the target property? Which is the best performing? Why?
4. What is the primary configuration difference between client reflection and targeted Post On Change?
5. How can you most effectively refresh of layout of several related widgets by enabling targeted Post on Change on one trigger widget?

© Guidewire Software, Inc. 2001-2013. All rights reserved. Do not distribute without permission.

1) A partial page update response is typically a JSON response to an AJAX request.

2) disableOnEnter, onChange, and target.

3) No value (unspecified value), DATA_ONLY, or a specific id? DATA_ONLY refreshes only the user-editable data and is often the best performing.

4) You configure client reflection on listening widgets. You configure targeted Post On Change on the trigger widget.

5) If there are several widgets that for which you need to refresh the layout or update the data, consider wrapping the related widgets into an Input Set widget.

© Guidewire Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.

Notices

Copyright © 2001-2013 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 ExampleCenter, 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. Guidewire products are protected by one or more United States patents.

This material is Guidewire proprietary and confidential. The contents of this material, including product architecture details and APIs, are considered confidential and are fully protected by customer licensing confidentiality agreements and signed Non-Disclosure Agreements (NDAs).

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 Software, Inc. 2001-2013. All rights reserved.
Do not distribute without permission.