Policy Center

Lesson Outline

- Entity Names
- Pattern Validation
- UI Validation
- Organisation Structure

Entity Names

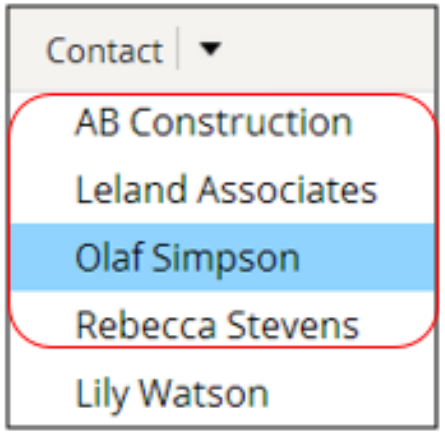
Entity Names

- Every entity has internally defined DisplayName field.

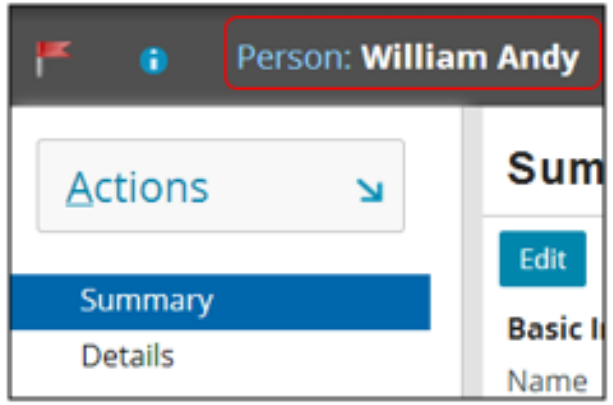
- For Example :
  - The display name for a Company is simply the value of the Company's Name field.
  - The display name for a Person is a concatenation of the first name plus a blank space plus the last name. If the person's name also has a suffix, such as “Mr.", that is also concatenated to the value.

Where entity names are used

- Whenever object as a whole is displayed
  - For example, dropdown that lists ABContacts



- Whenever object's DisplayName is explicitly referenced
  - For example, info bar widget with value property set to ABContact.DisplayName



Complex entity names

- Entity names are defined in Gosu
  - Can make use of any logic available in Gosu

```
if(person.PublicID == "ab:109")
  {
  output = "Mr." + person.DisplayName
  print(output)
  }

if(person.PublicID == "ab:70")
  {
  output = "Ms. " + person.DisplayName
  print(output)
  }
}
```

```
Ms. Rebecca Stevens
Mr.Perry Newton
```

Example of conditional logic for ABPerson

If person is male, append "Mr. "

If person is female, append "Ms. "

Otherwise, append nothing

Entity name editor

TrainingApp - [C:\Guidewire\TrainingApp] - [configuration] - ...\modules\configuration\config\displaynames\ABPerson.en - Guidewire Studio 1.0.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

configuration > config > displaynames > ABPerson.en

Project

configuration (C:\Guidewire
- config
  - Entity Names
    - ABCompany.en
    - ABContact.en
    - ABContactTag.en
    - ABPerson.en
    - ABPlace.en
    - Address.en
    - AssignableQueue.
    - Attribute.en
    - BusinessWeek.en
    - Company.en
    - Contact.en
    - ContactContact.e
    - ContactNote.en
    - Credential.en
    - EFTData.en
    - Group.en
    - GroupUser.en
    - Person.en
    - Place.en
    - Region.en
    - Role.en
    - SecurityZone.en
    - SpecialistService.e
    - UpdateBatch.en
    - User.en

ABPerson.en

Name	Entity Path	Sort Path
lastName	ABPerson.LastName	ABPerson.LastNameDenorm
firstName	ABPerson.FirstName	ABPerson.FirstNameDenorm
suffix	ABPerson.Suffix	
lastNameKanji	ABPerson.LastNameKanji	
firstNameKanji	ABPerson.FirstNameKanji	
prefix	ABPerson.Prefix	
particle	ABPerson.Particle	
middleName	ABPerson.MiddleName	

Add Name Duplicate Name Remove Name

Default

Type

```
1   uses ...
4
5   var person = new ABPersonNameFieldImpl(){
6     :FirstName = firstName,
7     :MiddleName = middleName,
8     :LastName = lastName,
9     :FirstNameKanji = firstNameKanji,
10    :LastNameKanji = lastNameKanji,
11    :Particle = particle,
12    :Prefix = prefix,
13    :Suffix = suffix
14  }
15
16  return new NameFormatter().format(person, " ", NameLocaleSettings.DEFAULT
```

Variable table

ABPerson.en

Name	Entity Path	Sort Path	Sort Order
lastName	ABPerson.LastName	ABPerson.LastNameDenorm	1
firstName	ABPerson.FirstName	ABPerson.FirstNameDenorm	3
suffix	ABPerson.Suffix		5
lastNameKanji	ABPerson.LastNameKanji		2
firstNameKanji	ABPerson.FirstNameKanji		4
prefix	ABPerson.Prefix		6
particle	ABPerson.Particle		7
middleName	ABPerson.MiddleName		

Add Name Duplicate Name Remove Name

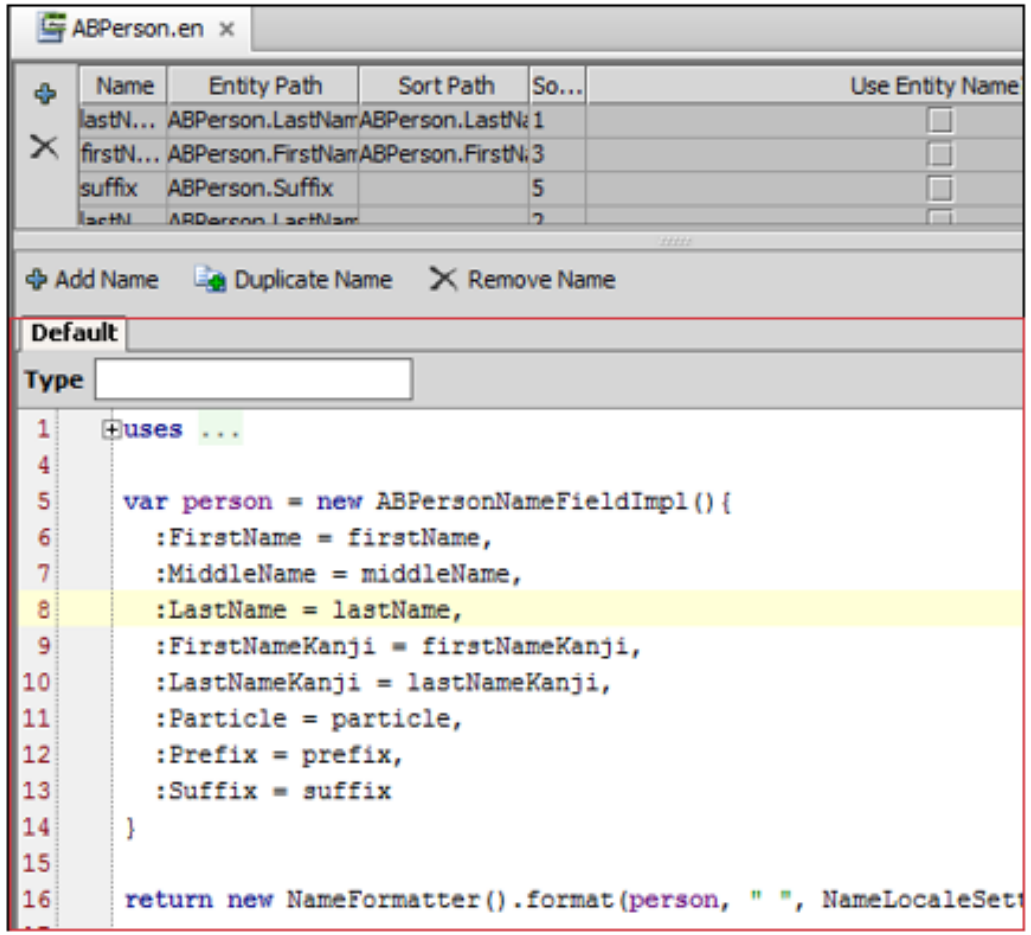
Default

Type

```
1   uses ...
4
5   var person = new ABPersonNameFieldImpl(){
6     :FirstName = firstName,
7     :MiddleName = middleName,
```

- Defines variables for use in method
- Also defines default sort order for entity

Return value pane



- Determines value to return when object's display name is needed
- If necessary, application converts value to string

Task

- Add the below fields in HouseDetails_Ext
  - First Name
  - Last Name
  - Gender

- Create a display key for HouseDetails_Ext entity.

- Display Name format : “Mr/Mrs.” + “FirstName” + “LastName”

Field Validation

Validation

- **Validation** is a general application behavior that prevents a user from saving invalid business data

- For example, specifying a policy expiration date that is prior to the policy's effective date

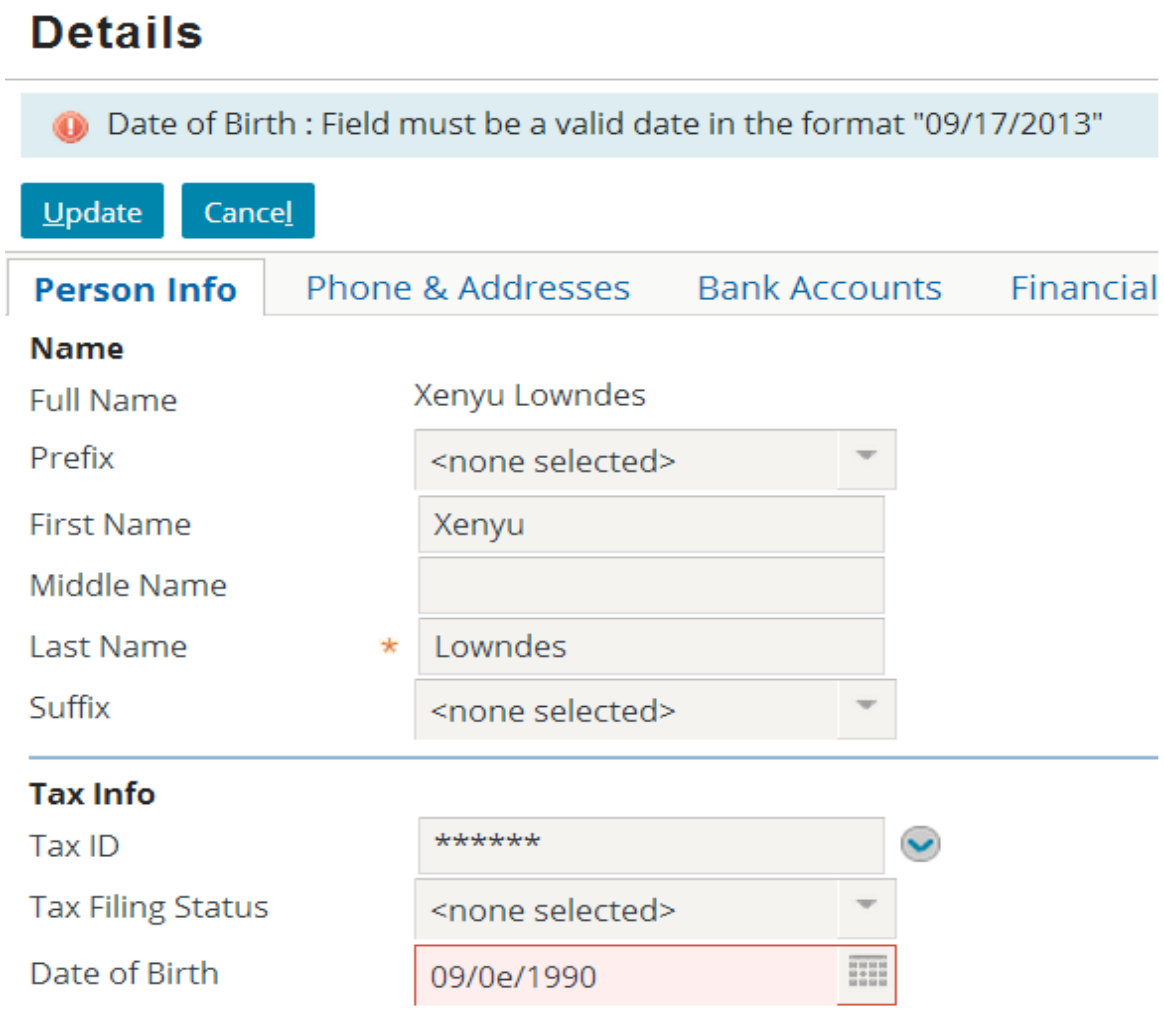
Field-level validation

- **Field-level validation** is a validation behavior tied to one or more specific fields, which can be implemented at:

- Two Levels of Validation
  - Data model level
    - Datatypes
    - Field validators
  - UI level
    - Validation expressions

Data Model Level Validation

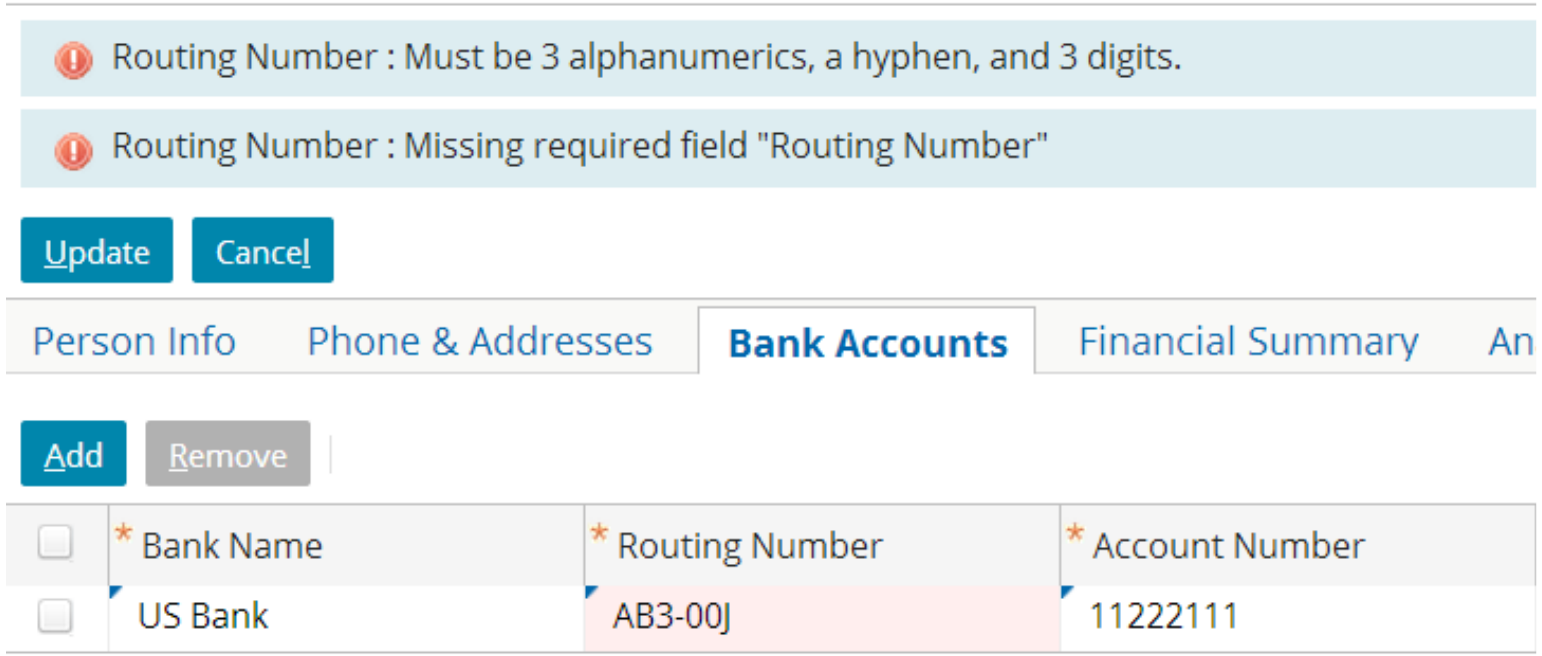
1. Datatypes



- Guidewire inherently requires all fields to have values that are legal for the field's underlying datatype
- Guidewire automatically prevents data of the wrong datatype from being saved and warns the user of the error

2. Field Validators

- A **field validator** is a pattern that is tied to one or more fields in the data model
- If field value does not match pattern, data cannot be saved and error message is displayed



Steps to implement field validator

1. Create error message display key
2. Create field validator
3. Associate field validator with entity field
4. Deploy the changes

Task

- Create an input “License” of type varchar

- The field should only accept the value of the format :

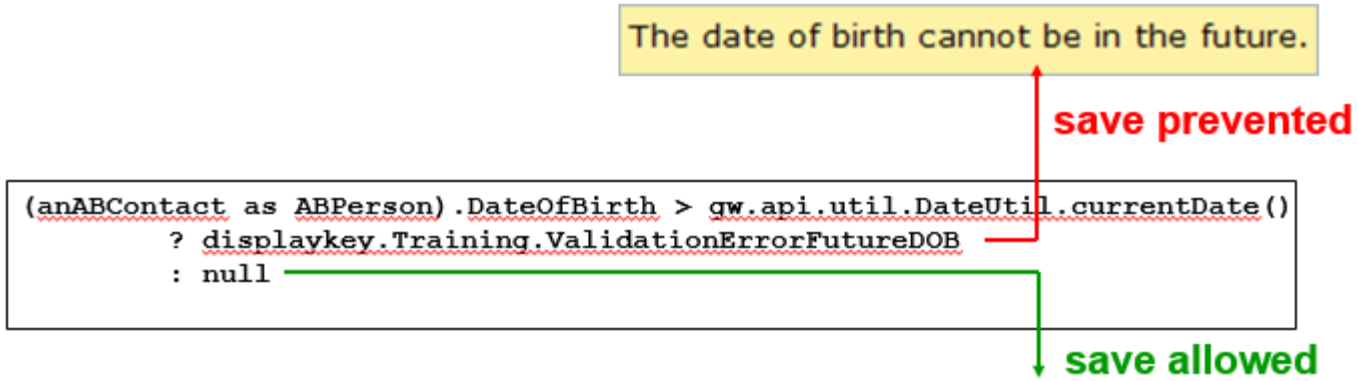
  AB-32-12345

UI Level Validation

Validation expression

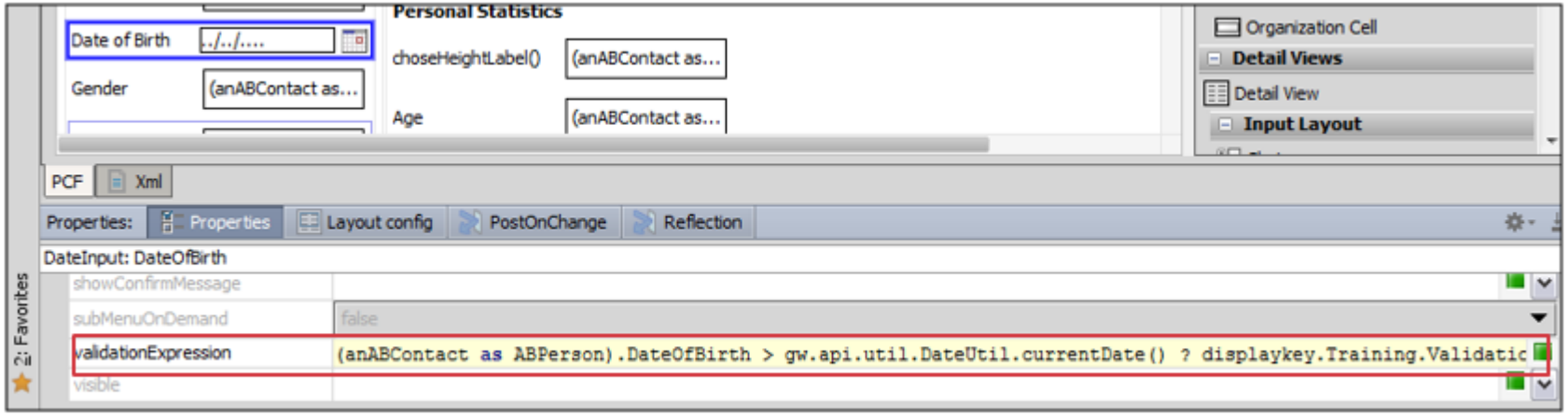
- A **validation expression** is an expression associated with a single atomic widget that implements field-level validation
  - If expression returns NULL, save is allowed
  - If expression returns string (an error message), then save is prevented, field is flagged, and message is displayed at top of screen

- Example:



The validationExpression attribute

- Syntax: *condition* ? NULL : *errorMessage*, or
  condition ? *errorMessage* : NULL
  - Typically written using ternary operator
  - NULL and error message can come in either order



Task

- Add a new field “Inauguration Date”
- Add a validation to check if the Inauguration Date is after Foundation Date.
- If the Inauguration Date is before Foundation Date, throw a message “Inauguration Date cannot be before Foundation Date”

Organization Structure

Guidewire implementation team

- Business architect
  - Writes business requirements documents
- Configuration developer
  - Configure product data model, user interface, and business logic
- Integration developer
  - Create integration points to external systems to share data with Guidewire application
- Reporting developer
  - Develops data warehouses , portals, reports

Development Model

- Waterfall model
  - Prepare Test plan / Test Case document
  - Testing conducted in Dev Environment

- Agile model
  - Update Rally for Test Case
  - Developer and tester do peer testing
  - Unit testing in developer’s machine

Environments in Project

- Local Environment – Unit Testing
- Dev Environment - Integration Testing
- Test Environment – For Testing team to test
- UAT Environment – For Business Users to test
- Load Environment – Tested for 1000 Users
- Production Environment – Deployed to client system

Testing Phase

- Code deployed to SystemTest environment

- Types of Testing
  - Smoke Test – Create a claim
  - Manual Test – Populate dropdowns, visibility
  - Compatibility Test – Different browsers and versions
  - Integration Test – check payload for data, access Policy center and claim center application to verify the values populated

Testing Tools

- Log defects
  - HP Quality Center – Waterfall/Agile Model
  - Rally – Agile Model
- Automate Testing
  - Selenium – Simple Web browser testing
  - HP Unified Functional Test (UFT) – More complex testing
- Load Testing
  - Apache JMeter – 30 Users login to call Repairer service

Summary

- Entity Names – DisplayName property
- Pattern Validation – using fieldValidators.xml
- UI Validation – validationExpression property
- Organisation Structure

Thank You