Lesson 8 Validation Classes

8.1 Validation chaining

8.1.1 Investigation

Observe the validation sequence and validation chaining used in the business auto line of business.

Activity

Answer the following questions by drilling into the `BALineValidation.gs` class for the Commercial Auto line in Studio.

a) List the primary validate method in that class.

b) List all methods called from the validate method in sequence.

Stop

8.1.2 Solutions

Solution

Answer the following questions by drilling into the `BALineValidation.gs` class for the Commercial Auto line in Studio.

a) List the first method that is called in that class?

doValidate()

b) List all methods called from the validate method in sequence:

- *additionalInsuredAndTypeUnique()*
- *vinIsUnique()*
- *fleetTypeMatchesVehicleInfo()*
- *checkLiabilityCoverage()*
- *atLeastOneHiredAutoState()*
- *atLeastOneNonownedState()*
- *nonOwnedBasisSumGreaterThanZero()*

8.2 Configure validation classes

8.2.1 Requirement

Enigma Fire & Casualty Insurance wants to make sure that users cannot buy Personal Auto insurance for more than two vehicles.

Specifications:

Spec 1 Add a validation check that warns users when they create more than two vehicles that they are not allowed to do that.

Spec 2 The warning must appear when the user leaves the Vehicles Wizard step.

Spec 3 The validation must prevent users from quoting the policy when there are more than two vehicles on the policy.

8.2.2 Configuration

Activity

The following are some tips:

1. **PALineValidation uses a class called PALineVehiclesValidator.**

   The method can be placed there.

2. **Validation level typecodes should be used instead of the level code as a string, for example:**

   ```
   default → ValidationLevel.TC_DEFAULT
   quickquote → ValidationLevel.TC_QUICKQUOTABLE
   ```

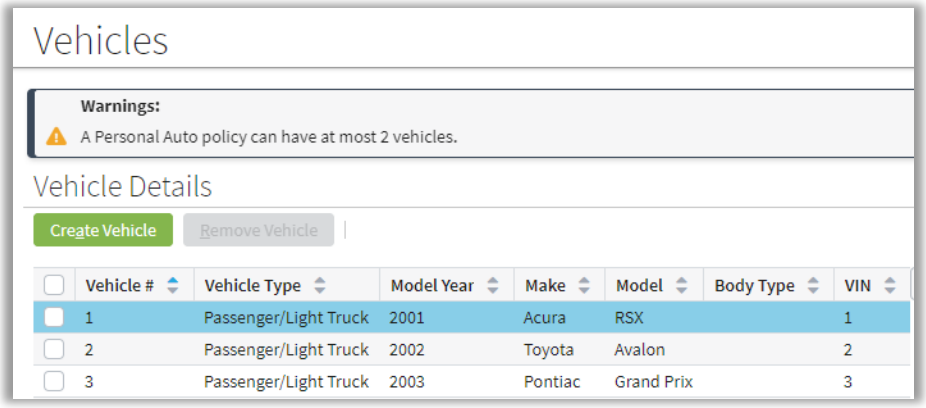
   and so on…

8.2.3 Verification

1. **Log in as aapplegate/gw.**
2. **Create a personal auto submission with three vehicles.**

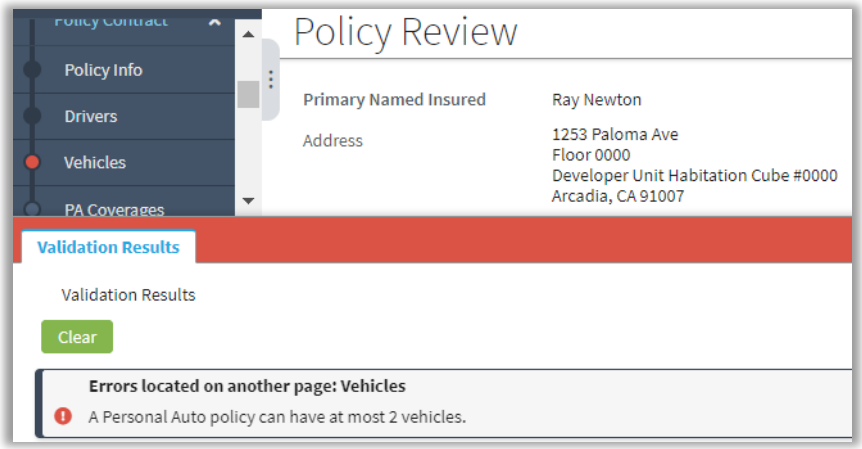
   You can open an existing policy and click Actions → Copy Submission, and then edit it to have three vehicles. Or create a new submission with three vehicles.

3. **When you leave the Vehicles step of the job wizard, the following warning should be displayed.**



4. **Move through the remaining wizard steps until you can quote the policy.**

   When you click the Quote button, your validation class should display an error with a link to the Vehicles page as shown in the following screen. A quote should not be displayed. The error should prevent you from getting a quote.



Stop

8.2.4 Solutions

Solution

1. **In `PALineVehiclesValidator.gs`, add a function to check for at most two vehicles and add the method to the function `doValidate()`.**

```
9  class PALineVehiclesValidator extends PolicyLineValidation<entity.PersonalAutoLine> {

31   override function doValidate() {
32    atLeastOneVehicle()
33    atMostNumVehicles(2)
34    allGaragesInSameState()
35    vinIsUniqueInPeriod()
36    validateEachVehicle()
37   }

116  public function atMostNumVehicles(num: int) {
117   Context.addToVisited(this, "atMostTwoVehicles()")
118   if (paLine.Vehicles.Count > num and Context.isAtLeast(ValidationLevel.TC_DEFAULT)){
119    var msg = DisplayKey.get("Ext.Validator.AtMostTwoVehicles", num)
120    if ( Context.isAtLeast(ValidationLevel.TC_QUICKQUOTABLE))
121     Result.addError(paLine, ValidationLevel.TC_QUICKQUOTABLE, msg, VEHICLES_WIZARD_STEP)
122    else
123     Result.addWarning(paLine, ValidationLevel.TC_DEFAULT, msg, VEHICLES_WIZARD_STEP)
124   }
125  }
126
127 }
```

2. **Create the display key.**

   Press Alt-Enter on the display key used in the method above and create the new display key.

   ```
   Ext.Validator.AtMostTwoVehicles = A Personal Auto policy can have at most {0}
   vehicles.
   ```

3. **Stop and restart server using Run -> Debug Server or simply Reload Changed Classes in Studio.**

8.3 References

Review

Creating a new validation class:

New class should extend `PCValidationBase` or `PCValidation`. Use an existing validation class as a reference.

For new entities:

- Create new validation class for entity you want to validate or
- Create new validation class when you create a new LOB

For existing entities:

- If a validation class for the entity exists add validation check methods to that class
- If the validation class does not exist, then create new validation class

Review

Steps to add a validation check:

1. **Edit/add appropriate validation class.**

   First you will have to decide whether this validation can be added to an existing class or a new validation class needs to be created. If, on the other hand, you needed to add a validation check for a new entity that didn't already have any validation checks and no validation checks exist for it, you would need to create a new validation class to validate that specific entity.

2. **Add the validation check method.**

   You will need to restart the server because of display key addition.

3. **Call method from `validationImpl()`.**

   The `validateImpl` method of any class calls all the methods in a logical sequence.

   Note: In case where you had to create a new validation class then you will also have to create a `validateImpl()` method and then add the method call statement to the method. For the example, we are creating a validation check for a personal vehicle and adding the method call statement to the `validateImpl()` method of the `PersonalVehicleValidation` class.

   If you are extending `PolicyLineValidation` class, you will have to create a `doValidate()` method and call the methods that validate a single issue in a logical order.

4. **Invoke the method.**

   The validation class methods must be invoked explicitly and are not called automatically. If it is already in the validation chaining path, you do not have to invoke it.

5. **Verify the result in the UI.**

Review

Invoking class-based validation:

Class-based validation must be explicitly invoked through code, job processes, wizard steps, workflow steps, integration plug-ins, or from any Gosu code.

To invoke validation from a particular PolicyCenter location:

- Wizard steps - use beforeSave attribute for that step.
- Pop-ups - use beforeCommit attribute for that pop-up.

Review

Validation Chaining is the process of one validation class calling another validation class to perform additional validation checks.

- Call `validate()` method which in turn calls other methods.
- Invoke validate method on another validation class (may have to loop through a set of objects).
- Classes chain to validations of entities they hold.
- Refer to `PolicyPeriodValidation` class for examples of validation chaining. It calls other validation classes, such as:

  `PolicyContactRoleValidation`,

  `PolicyContactRoleForSameContactValidation`,

  `PolicyLocationValidation`, `AnswerValidation` and

  `PolicyLineValidation`.

Stop