Policy Retrieval Integration

Overview

In this lab, you will complete a portion of the policy retrieval portion of the plugin for the Personal Auto line of business.

This lab should be done on ClaimCenter and involves integrating with ExternalApp, which will serve as a Policy Administration System.

To run ExternalApp, navigate to c:\Guidewire\ExternalApp and double-click the ExternalApp shortcut. The start-up process is complete when the message "\*\*\*\* ContactManager ready \*\*\*\*" appears in the console.

Tasks

The AcmeIPolicySearchAdapter does policy retrieval only for the Homeowners line of business. Modify the plugin so that it also does policy retrieval for the Personal Auto line of business.

Part 1: Typecode Mapping

1. Navigate to ClaimCenter/modules/configuration/config/typelists/mapping. (This is part of the PASintegrationWithExternalApp.zip file and should have been copied over to ClaimCenter in the previous lab.) Open the typecodemapping.xml file.
2. The following is a list of codes for Personal Auto coverages as they appear in ExternalApp. In typecodemapping.xml, add <mapping> tags to the CoverageType <typelist> tag so that the ExternalApp typecodes map to the appropriate Guidewire typecodes. You can use either the LOB Model editor in Studio or the Data Dictionary to determine what the Guidewire typecode codes are.

**ExternalApp Personal Auto Coverage Codes**

* Liability - Bodily Injury and Property Damage
* Uninsured Motorist - Bodily Injury
* Uninsured Motorist - Property Damage
* Collision
* Comprehensive
* Towing (map this to the code for Guidewire's "Towing and Labor" coverage)

1. Restart ClaimCenter.
2. To test your work, open the Gosu Scratchpad and execute the following for each code listed above, where "xxx" is the code. Ensure that the correct Guidewire typecode is returned.

uses gw.api.util.TypecodeMapperUtil

print (TypecodeMapperUtil.getTypecodeMapper().  
getInternalCodeByAlias("CoverageType", "acme:pas", "xxx"))





Part 2: Enabling the partial Personal Auto retrieval method

1. In Studio, open the acme.cc.plugin.policy.PersonalAuto class. Notice that there are two implementations of the retrievePersonalAutoPolicy() method. The first is a stub method. The second is a partially complete method that has been commented out.
2. Delete the stub implementation, and remove the "/\*" and "\*/" comment delimiters surrounding the second method. Then save the class. Finally, restart ClaimCenter.
3. To test your work, create and save a Personal Auto claim (such as a claim based on the Anna Williams personal auto policy). Once the claim has been saved, you should be able to verify that the policy has information for the insured, the policy-level coverages, and the endorsements. There should not be any detailed information on vehicles.

**Insured information and policy-level coverages:**



**Endorsements:**



Note that vehicle information is not available.

Part 3: Complete the vehicle portion of the Personal Auto retrieval method

1. The retrievePersonalAutoPolicy() method has a section for vehicles. But there is no code that actually creates vehicles or vehicle coverages. Instead, it is just a set of comments. Replace the comments with code that will successfully retrieve vehicle information for personal auto claims.

You should review the code in acme.cc.plugin.policy.Homeowners.retrieveHomeownersPolicy() that creates locations and location coverages to help determine how to complete the vehicle code. The following is a sample of vehicle XML returned from the PAS. (The XML includes data on deductibles, but you do not need to load this into ClaimCenter.)

<PASPolicy>

...

<Vehicles>

<Entry>

<Color>White</Color>

<Make>Mazda</Make>

<Model>MX-5 Miata</Model>

<VehicleCoverages>

<Entry>

<CoverageName>Towing</CoverageName>

</Entry>

<Entry>

<CoverageName>Collision</CoverageName>

<Deductible>1000</Deductible>

</Entry>

<Entry>

<CoverageName>Comprehensive</CoverageName>

<Deductible>1000</Deductible>

</Entry>

</VehicleCoverages>

<VIN>FHJM3ASPL93MN6F7U</VIN>

<Year>2007</Year>

</Entry>

<Entry>

...

</Entry>

</Vehicles>

...

</PASPolicy>

Syntax for strongly-typed XML

*object*.Children[*n*].Text references the text of the *n*th child of the element referenced by *object*.

In other words, if anEntry references the first <Entry> element under <Vehicles> as shown above, then:

Gosu expression Value

* anEntry.Children[0] The <Color> element.
* anEntry.Children[0].Text The string "White"
* anEntry.Children[1] The <Make> element
* anEntry.Children[1].Text The string "Mazda"

The following is a copy of the coverages loop:

// Vehicle data (including vehicles, vehicle risk units and

// vehicle coverages)...

for (currentVehicleEntry in xml.Vehicles.$Children index i) {

// Vehicle data

// --> TO DO: create a vehicle object

// --> TO DO: set the vehicle's color, year, make, model, and VIN

// Vehicle Risk Unit data

// --> TO DO: create a vehicle risk unit object

// --> TO DO: set the vehicle's risk unit number (the first vehicle should

// be 1, the second 2, and so on)

// --> TO DO: associate the vehicle risk unit to the vehicle

// Vehicle coverage data

for (currentVehicleCoverageEntry in currentVehicleEntry.Children[3].Children) {

// --> TO DO: create a vehicle coverage object

// --> TO DO: set the vehicle coverage's coverage type using the

// currentVehicleCoverage's coverage type

// --> TO DO: associate the vehicle coverage to the vehicle risk unit

} // end vehicle coverage loop

// --> TO DO: associate the vehicle risk unit to the policy

} // end vehicle loop

1. To test your work, create and save a Personal Auto claim based on the same policy as the one used in the previous part. Once the claim has been saved, verify that there is now detailed information on vehicles and vehicle coverages.

