Lab: Gosu and XML

In this lab, you will manipulate untyped and strong typed XML. In the first exercies, you will work with untyped XML and Gosu. In the second exercise, you will work with strongly types XML and Gosu.

Requirements

This lab requires that you use TrainingApp 8.0 and Guidewire Studio 8.0. To view, edit, and delete various contacts, log in to TrainingApp as Super User. The default URL for TrainingApp is <http://localhost:8880/ab/ContactManager.do>. The login/password for Super User is su/gw.

1. Untyped XML

Acme has an integration point that needs to manipulate untyped XML files pertaining to policy holders. For each file, Acme wants a subset of information displayed to the console. One additional XML element, <RiskAssessment>, must also be added to each file.

Investigation

1. Open XmlFile1.xml and XmlFile2.xml in Guidewire Studio
2. Both files are simple XML files storing information about a policy holder.
3. Both files contain the same XML elements, but not in the same order.
4. Move XmlFile1.xml and XmlFile2.xml
5. Move both files to the C:\Guidewire\TrainingApp directory.

Configuration

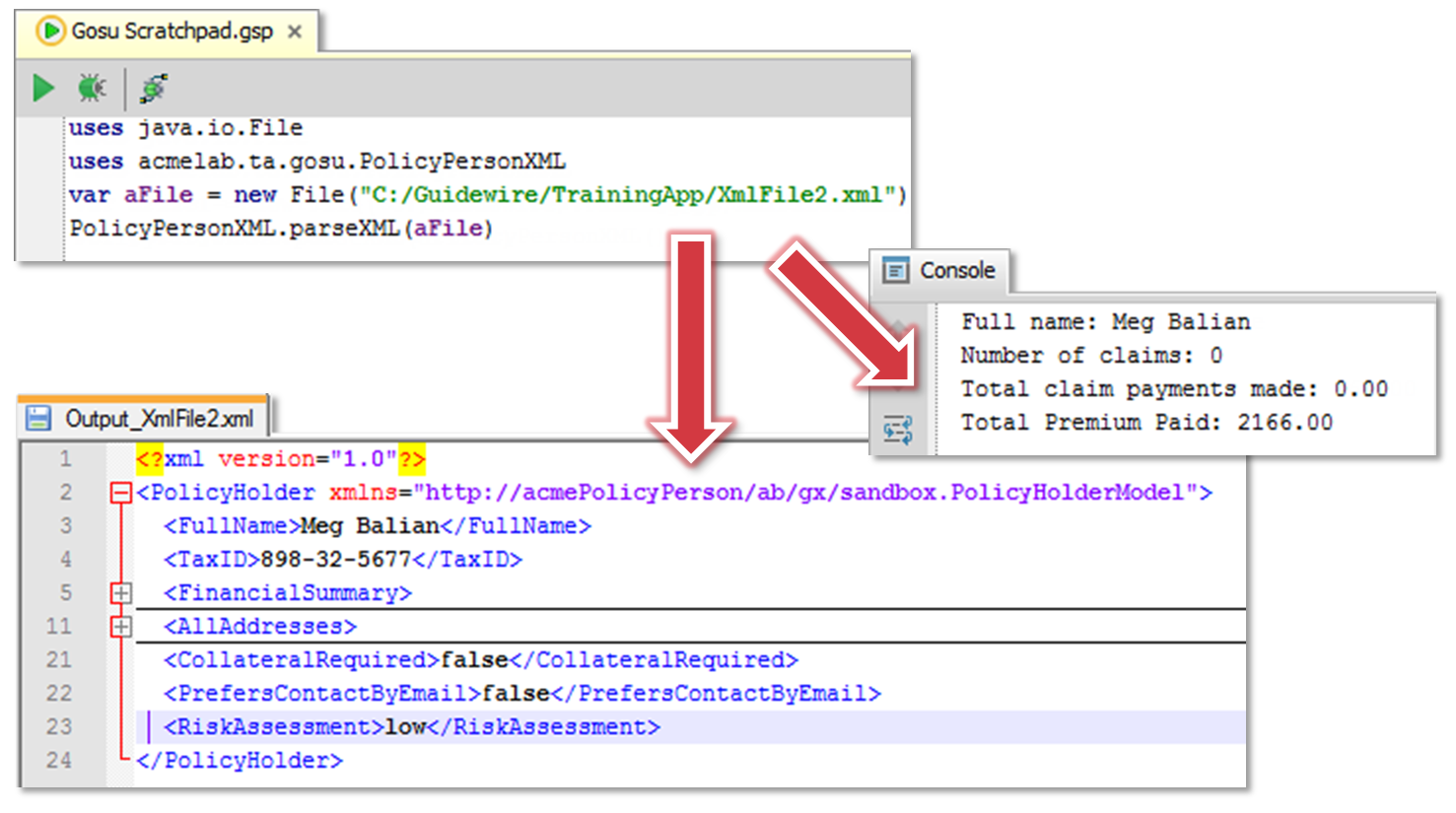
1. In Guidewire Studio, create the PolicyPersonXML class
2. Create the class in a package named acmelab.ta.gosu.
3. Create a method to read and modify untyped XML
4. Create a method in the PolicyPersonXML class that takes a java.io.File XML file as an input.
5. Extracts information and prints it to the console:
   * Person's full name
   * Total number of claims
   * Total claim payments made
6. Adds the <RiskAssessment>value</RiskAssessment element to the root element where the value is "high" if the total claim payments made is greater than the total premium paid and is "low" if the total claim payments made is less than or equal to the total premium paid
7. Writes the modified XML to the C:\Guidewire\TrainingApp directory as Output\_<xmlFileName>.txt where <xmlFileName> is the name of the input XML file name
8. Deploy your changes from Guidewire Studio
9. If your server is not already running, start the server using Debug 'Server'.
10. If your server is running, stop the server, and then start the server using Debug 'Server'.
11. Review the Debug console for errors and verify that the application is running in the Debug console.

Verification

1. Open Gosu Scratchpad
2. Execute the PolicyPersonXML method to parse XmlFile1
3. In Gosu Scratchpad, call your class method to parse XmlFile1.
4. Since your code is not data-backed, it is not necessary to run in a debug 'server' processs.
5. Verify the output to the console.
6. Open the Output\_XmlFile1.txt and verify that the addition of the RiskAssessment element value is high.



1. Execute the PolicyPersonXML method to parse XmlFile2
2. In Gosu Scratchpad, call your class method to parse XmlFile2.
3. Since your code is not data-backed, it is not necessary to run in a debug 'server' processs.
4. Verify the output to the console.
5. Open the Output\_XmlFile2.txt and verify that the addition of the RiskAssessment element value is low.



1. Strongly Typed XML

Acme has an integration point that needs to create XML about policy holders for an external system. In the external system, policy holder information is structured differently than it is in TrainingApp. The external system has a policyholder XSD that describes how to structure the information. In this exercise, you will write code that maps TrainingApp data into the necessary and valid XML structure for the external system.

Investigation

1. Open policyholder.xsd in Guidewire Studio
2. Review the XSD file.
3. Move the policyholder.xsd
4. Move the policyholder.xsd to the appropiate directory in the TrainingApp project in Guidewire Studio.
5. Deploy your changes from Guidewire Studio
6. If your server is not already running, start the server using Debug 'Server'.
7. If your server is running, stop the server, and then start the server using Debug 'Server'.

Configuration

1. Create a new method for PolicyPersonXML class
2. Create a method that takes a public ID as an input parameter and queries for the ABPolicyPerson using that public ID.
3. If there is one ABPolicyPerson with that public ID, then print to the console in valid XML format the following
   * Person's full name (using the value in ABPolicyPerson.DisplayName)
   * Tax ID (using the value in ABPolicyPerson.TaxID)
   * Risk Assessment where the value is "high" if the total claim payments made is greater than the total premium paid and is "low" if the total claim payments made is less than or equal to the total premium paid (using ABPolicyPerson.FinancialSummary)

Hints

**Hint (1)**Example of query for a ABPolicyPerson:

**1 uses gw.api.database.Query**

**2 uses gw.api.database.Relop**

**3**

**4 var publicID : String = "ab:146"**

**5 var queryObj = Query.make(ABPolicyPerson)**

**6 queryObj.compare (ABPolicyPerson#PublicID, Relop.Equals, publicID)**

**7 var resultsObj = queryObj.select()**

1. Deploy your changes from Guidewire Studio
2. Reload changed classes.

Verification

1. If needed, create a financial summary for an ABPolicyPerson in TrainingApp
2. In Training App, navigate to a given contact (such as Christy Webster).
3. Note the ABContact's Public ID.
4. Navigate to the Detail screen's Financial Summary card.
5. If there are no values for the Premium and Claim Payment fields, edit the Financial Summary and enter values for all fields. Make sure that the Total Claim Payments Made is greater than the Total Policy Premium Paid.
6. Click Update.
7. Open Gosu Scratchpad
8. Verify that the Run in Debug Process icon is available in Gosu Scratchpad.
9. Execute the PolicyPersonXML method to generate XML for a given ABPolicyPerson
10. In Gosu Scratchpad, call your class method to generate XML for a given ABPolicyPerson.
11. Since your code is data-backed, it is necessary to run in a debug 'server' processs.
12. Verify the output to the console.



|  |  |
| --- | --- |
|  | Stop and ask your instructor to review your completed lab. |