Lesson 6 XML Modeler and Strongly Typed XML

This exercise requires that you use **TrainingApp**, Guidewire Studio, and a supported web browser. Start **Guidewire Studio for TrainingApp**. Start the server as **Debug** 'Server'.

The default URL for **TrainingApp** is: http://localhost:8880/ab/ContactManager.do. Log in to **TrainingApp** as Alice Applegate whose login/password is **aapplegate/gw**.

Exercise 1: Use an XML model to generate an XML message payload

Exercise

For existing contacts with a modified tax ID and for newly created contacts of the type ABPerson, **TrainingApp** must determine if the given contact has been involved with a previous act of insurance fraud. The external system that checks for fraud requires that there is specific format for message payloads for various types of entities.

6.1.1 Requirements

Spec 1 The external system requires the following format for an XML message payload for a contact of the type ABCompany:

- Parent XML element for ABContact.
- Sub element for PublicID as Key property.
- Sub elements for DisplayName and TaxID as Normal properties.
- Sub elements contain respective values for TaxID and DisplayName.

```
<?xml version="1.0"?>
<ABContact xmlns="si.ta.xmlmodels.abcontactxmlmodel">
  <PublicID>java.lang.String</PublicID>
  <DisplayName>java.lang.String</DisplayName>
  <TaxID>java.lang.String</TaxID>
</ABContact>
```

Spec 2 Create an xmlmodel package with the fully qualified name of: si.ta.xmlmodel

Spec 3 Create an XML model called ABContactModel in the xmlmodel package.

Spec 4 Verification requirements:

- Export properties even if their value is null.
- Export properties even if their value has not changed.
- XML serialization should not perform sort or validation.

6.1.2 Tasks

1. **Create an xmlmodel package.**
2. **Create an XML model.**
3. **Deploy code changes.**
4. **Perform verification steps.**

6.1.3 Verification steps

1. **Generate debug console output using Gosu Scratchpad.**

   a) In **Studio**, open **Gosu Scratchpad** by clicking **Tools ➔ Gosu Scratchpad**.

   b) Write code that will test the solution:

   - Create a query that references contact William Andy whose **publicID** is **ab:5**.
   - Make sure only one result is retrieved, otherwise throw an exception.
   - Create a new payload variable the generates XML output utilizing the XML model.
   - Configure GXOptions.
   - Configure XML serialization options.
   - Output the payload to the debug console.

   c) Verify that the output in the debug console has the correct format.

```
<?xml version="1.0"?>
<ABContact xmlns="http://guidewire.com/ab/gx/si.ta.xmlmodels.abcontactxmlmodel">
  <PublicID>ab:5</PublicID>
  <DisplayName>William Andy</DisplayName>
  <TaxID>123-45-6793</TaxID>
</ABContact>
```

Solution 1: Use an XML model to generate an XML message payload

1. **Create an xmlmodel package.**

   a) Right-click on **si.ta** folder and select **New ➔ Package**.

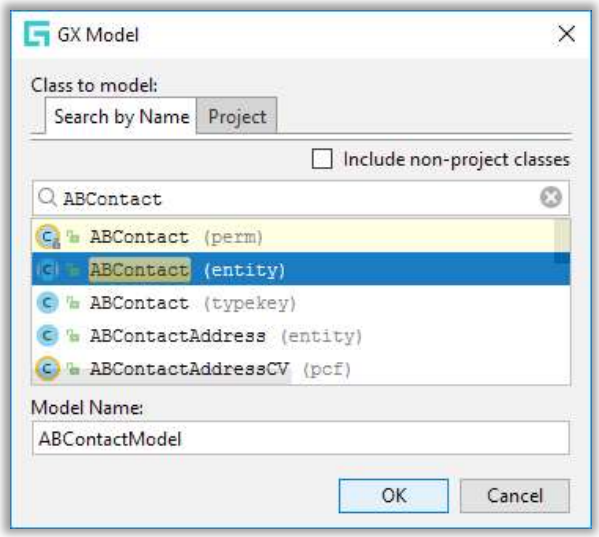
   b) Enter **xmlmodel** as the new package name.

2. **Create an XML model.**

   a) Right-click on the **xmlmodel** package and select **New ➔ GX Model**.

b) Search for the **ABContact** entity.

c) Enter **ABContactModel** as the model name.

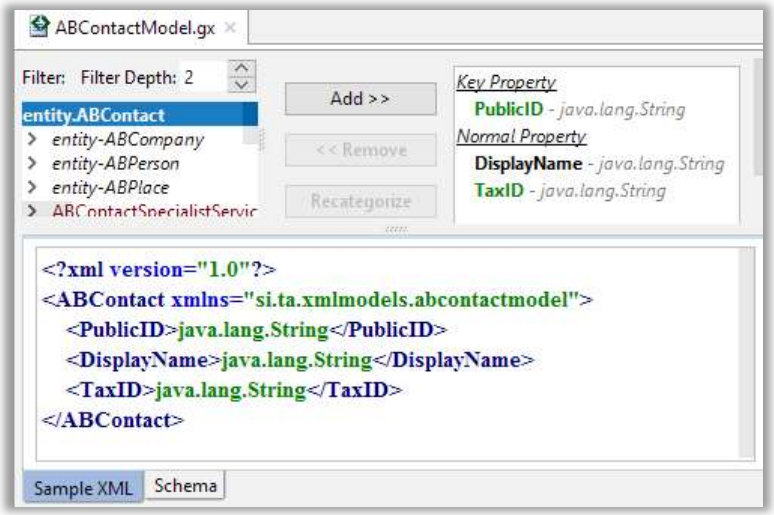


d) Click **OK**.

e) Select **PublicID** and add as a **Key Property**.

f) Select **DisplayName** and add as a **Normal Property**.

g) Select **TaxID** and add as **Normal Property**.



3. Deploy code changes.

a) From the **Studio** menu, **Restart** the server.

4. **Perform verification steps.**

```
uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.api.gx.GXOptions
uses gw.xml.XmlSerializationOptions
uses si.ta.xmlmodel.abcontactmodel.ABContact

var queryObj = Query.make(ABPerson)
queryObj.compare(ABPerson#PublicID, Relop.Equals, "ab:5")
var targetPerson = queryObj.select().AtMostOneRow
// Configure GXOptions
var gxOpts = new GXOptions() {
  :Incremental = false,
  :Verbose = true
}
// Configure serialization options
var sOpts = new XmlSerializationOptions() {
  :Sort = false,
  :Validate = false
}
// Create payload and display to console
var xml = ABContact.create(targetPerson, gxOpts)
var payload = xml.asUTFString(sOpts)
print(payload)
```

Exercise 2: Generate an XML message payload using an XSD

Exercise

Succeed Insurance 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 the information should be structured. You need to create code that maps **TrainingApp** data into this XML structure.

6.1.4 Requirements

Spec 1 Create a new package called xml.

Spec 2 Create a new package called xsd.

Spec 3 Create a new Gosu class called PolicyPersonXML.

Spec 4 Create a new function called generateXML for the PolicyPersonXML class that meets the following specifications:

- It takes a public ID as an input parameter.
- It queries for the ABPolicyPerson with that public ID.
- If there is no ABPolicyPerson with that ID, then it prints to console ABPerson with PublicID of <x> not found.
- If there is one ABPolicyPerson with that ID, then it prints to the console the XML for that person. This XML should be created using the policyholder.xsd file found at <root>\training.

Spec 5 The following information should be included in the XML:

- Full name
- Tax ID
- Risk assessment
- Set to high if the total claim payments made is greater than the total premium paid; otherwise, set to low

Hints

Keep in mind that premium amounts are not on ABPolicyPerson directly, but rather on ABPolicyPerson.FinancialSummary. If the ABPolicyPerson has no financial summary, then output ABPolicyPerson with PublicID <x> does not have a Financial Summary.

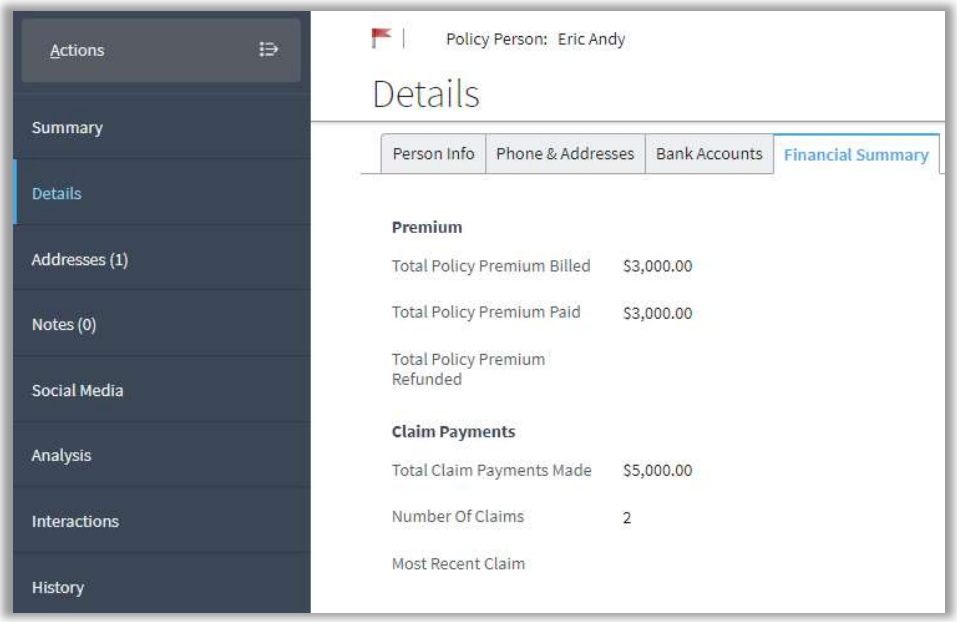
Spec 6 XML serialization should not perform sort or validation.

6.1.5 Tasks

1. **Create new packages.**
2. **Copy policyholder.xsd file located at <root>\training folder to xsd package.**
3. **Create new Gosu class and method.**
4. **Deploy code changes.**
5. **Perform verification steps.**

6.1.6 Verification steps

1. **Add financial summary information to Eric Andy in TrainingApp.**



2. **Generate debug console output using Gosu Scratchpad.**

   In **Studio**, open **Gosu Scratchpad** by clicking **Tools ➔ Gosu Scratchpad**.

   - ab:98 (Eric Andy)

```
<?xml version="1.0"?>
<PolicyHolder xmlns="http://acmePolicyPerson/ab/gx/sandbox.PolicyHolderModel">
  <FullName>Eric Andy</FullName>
  <TaxID>123-84-7704</TaxID>
  <RiskAssessment>high</RiskAssessment>
</PolicyHolder>
```

   - ab:145 (Betty Yalobusha)

```
ABPolicyPerson with PublicID ab:145 does not have a Financial Summary
<?xml version="1.0"?>
<PolicyHolder xmlns="http://acmePolicyPerson/ab/gx/sandbox.PolicyHolderModel">
  <FullName>Betty Yalobusha</FullName>
  <TaxID>123-84-7751</TaxID>
</PolicyHolder>
```

   - ab:100000 (No contact exists)