Lesson 4   Bundles and Database Transactions

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: Create a contact summary note

Exercise

Succeed Insurance wants to create and save contact notes with summary information for a given contact.

4.1.1   Requirements

Spec 1   Create the saveSummaryNote method that meets the following criteria:

- If the ABContactSummary's ContactID is null, do nothing, otherwise, for a non-null ContactID, create a contact note.
- Set the note subject to ABContact Summary.
- Set the note contact type to General.
- Where <x> is the appropriate value, the note body must contain the following lines (use "\n" for line breaks):
  - External ID: <x>
  - Name: <x>
  - Number of checking accounts: <x>

Spec 2   Query for the ABContact whose public ID matches the ABContactSummary's ContactID.

Spec 3   Add the contact note to the ABContact's ContactNotes array.

Spec 4   Commit the new note to the database.

4.1.2   Tasks

1. **Modify ABContact Summary class by inserting a new saveSummaryNote method.**
2. **Deploy code changes.**
3. **Perform verification steps.**

4.1.3   Verification steps

1. **Generate console output for ABContactSummary 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 **ABContactSummary**.
- Initialize the **ExternalID** sequence counter.
- Execute the **loadSummaryData** method using the given contact.
- Execute the **saveSummaryNote** method.

2. **Verify the note was created.**

a) In TrainingApp, navigate to the contact.

b) In the sidebar menu, click the Notes menu link.

c) Click Contact Note to view the entire note.

d) Verify the note was created with the correct subject and body.

Solution 1: Create a contact summary note

1. **Modify ABContact Summary class by inserting a new saveSummaryNote method.**

```
package si.ta.classes.entity

uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.api.system.database.SequenceUtil
uses gw.transaction.Transaction

/**
 * Training Lab
 */
class ABContactSummary {

  construct(i: int) {
    ExternalID = i
  }

  // Declare ExternalID property using getter and setter
  var _externalID: int  // Don't use Integer because it will initialize as null.

  property get ExternalID(): int {
    return _externalID
  }

  property set ExternalID(externalId: int) {
    if(externalId == 0) {
      _externalID = SequenceUtil.next(1000, "externalID") as int
    } else
      _externalID = externalId
  }

  // Declare properties using shorthand syntax
  var _contactID: String as ContactID
  var _name: String as Name
  var _numCheckingAccounts: int as NumCheckingAccounts
  var _assignedUserWorkload: int as AssignedUserWorkload

  // Create functions

  /**
   * Function that initializes ABContact properties.
   */
  @Param("contact", "Input parameter of type entity.ABContact")
  function loadSummaryData(contact: ABContact): void {
    this._contactID = contact.PublicID
    this._name = contact.DisplayName
    this._numCheckingAccounts = contact.BankAccounts.countWhere(\account ->
        account.AccountType == typekey.BankAccountType.TC_CHECKING)
    // check if the contact actually has an assigned user
    if(contact.AssignedUser != null) {
      var queryObj = Query.make(ABContact)
      queryObj.compare(ABContact#AssignedUser, Relop.Equals, contact.AssignedUser)
      this._assignedUserWorkload = queryObj.select().Count
    } else {
      _assignedUserWorkload = 0
    }
  }

  /**
   * Function that builds a comma-delimited list.
   */
  @Returns("A comma-delimited list of property values")
```

```
function buildConcatenatedSummary(): String {
  return String.format("%s,%s,%s,%s,%s",
      {_externalID, _contactID, _name, _numCheckingAccounts, _assignedUserWorkload})
}

/**
 * Function that create a summary note.
 */
function saveSummaryNote(): void {
  if (this._contactID != null) {
    var queryObj = Query.make(ABContact)
    queryObj.compare(ABContact#PublicID, Relop.Equals, this._contactID)
    var targetContact = queryObj.select().AtMostOneRow
    Transaction.runWithNewBundle(\newBundle -> {
      var newNote = new ContactNote()
      newNote.Subject = "ABContact Summary"
      newNote.ContactNoteType = ContactNoteType.TC_GENERAL
      newNote.Body = "External ID: " + this._externalID + "\n" +
          "Name: " + this._name + "\n" +
          "Number of checking accounts: " + this._numCheckingAccounts
      targetContact.addToContactNotes(newNote)
    }, "su")
  }
}

}
```

2. **Deploy code changes.**

    a) Restart server.

3. **Perform verification steps.**

```
uses gw.api.database.Query
uses gw.api.database.Relop
uses si.ta.classes.entity.ABContactSummary

var queryObj = Query.make(ABContact)
queryObj.compare(ABContact#PublicID, Relop.Equals, "ab:5")
var targetContact = queryObj.select().AtMostOneRow

var newSummary = new ABContactSummary(0)
newSummary.loadSummaryData(targetContact)
newSummary.saveSummaryNote()
```